



Features

- For use in WiMAX systems. Fully compliant with the WiMAX standard IEEE 802.16-2004
- Double Buffered input enables high speed operation
- Generic memory interface for easy ASIC integration
- Simple handshake protocol for reliable interfacing
- Fully synchronous design
- Full WiMAX compliant hardware interleaver. No external calculation required.
- Interleaver frame size selectable on a frame-by-frame basis
- Comprehensive verification plan provided
- All frame sizes supported
- All rates (1/2, 2/3, 3/4, 5/6, 7/8) supported

General Description

The SAL50100E consists of verilog IP for implementing the parallel concatenated convolutional (turbo) encoder as defined by the WiMAX standard. Refer to Fig. 1 below.

As shown in the figure, the encoder consists of a constituent Duo-Binary Recursive Systematic Convolutional (DBRSC) encoder operating on the input data frame. The DBRSC encoder operates on the data first in natural order and then operates on a permuted version of the input data. The permutation is accomplished in the interleaver block as described in detail in the IEEE std 802.16-2004 specification using an interleave address generator.

The encoder outputs a number of bits for every input bit, depending on the code rate: an unencoded copy of the data, parity bits from the first convolutional coder, and parity bits from the second convolutional coder.

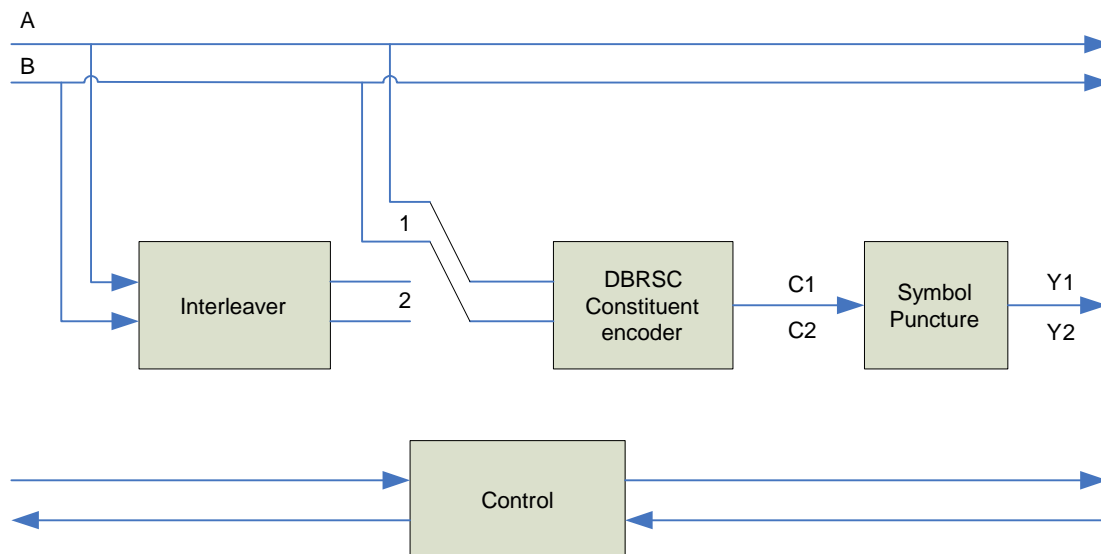


Figure 1: Encoder Block Diagram

Theory of Operation

The constituent Duo-Binary Circular Recursive Systematic Convolutional encoder is shown in more detail in fig. 2 below. The bits of the data to be encoded are alternately fed to inputs A and B, starting with the MSB of the first byte being fed to A.

Puncturing

Various code rates are achieved through selective deletion (puncturing) of parity bits. The puncture patterns are identical for both normal and interleaved code streams.

CTC Modulation Mapping

The encoded bit is fed into the mapper for BPSK and QPSK modulation in the following order:

$$A_0, B_0, \dots, A_{N-1}, B_{N-1}, Y_{1,0}, Y_{1,1}, \dots, Y_{1,M}, Y_{2,0}, Y_{2,1}, \dots, Y_{2,M}$$

Where M is the number of parity bits and the I channel is fed first. The order in which bits are fed to the mapper for 16-QAM, 64-QAM, and 256-QAM is:

$$A_0, B_0, \dots, A_{Rn}, B_{Rn}, Y_{1,0}, A_{Rn+1}, B_{Rn+2}, \dots, A_{2Rn}, B_{2Rn}, Y_{2,0}, \dots$$

Rate	Y															
1/4	1	1														
2/3	1	0	1	0												
3/4	1	0	0	1	0	0										
5/6	1	0	0	0	0	1	0	0	0	0						
7/8	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 1. Puncture patterns

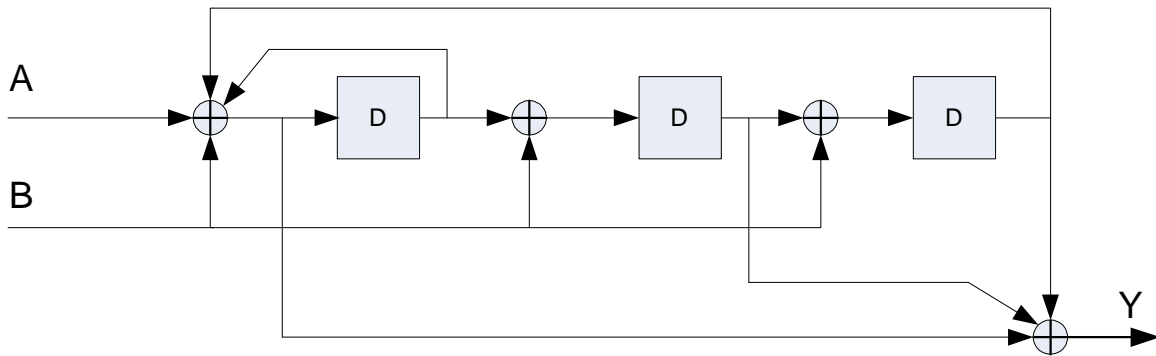


Figure 2 Encoder details

Specifying the code rate

The code rate is specified using the rate signal according to table 1 below.

rate signal	code rate
000	1/2
001	2/3
010	3/4
011	4/6
100	7/8
Other Values	1/2

Table 2. Code rate

Allowable information block lengths

The encoder is fed by blocks of k bits or N couples ($k=2N$ bits). For all frame sizes, k is a multiple of 8 (N is a multiple of 4). N is limited

to $8 \leq N/4 \leq 256$. Zero padding should be used for block sizes less than 32-bytes.

Encoder state initialization

The encoder is first initialized to the circulation state $Sc1$. The encoder is then fed the data sequence in natural order. This first encoding is called the C1 encoding. Then the encoder, after initialization with the circulation state $Sc2$, is fed the data again, this time in interleaved order. This is called C2 encoding.

The state of the encoder is denoted S (with $0 \leq S \leq 7$), with the S value read binary left-to-right out of the constituent encoder registers (see fig 2). The circulation states $Sc1$ and $Sc2$ are determined internally to the SAL50100E device using the following procedure:

- 1) Initialize the encoder with state 0. Encode the sequence in natural order for the determination of $Sc1$ or in interleaved order for $Sc2$. In both cases, call the final state of the encoder S_{0n-1} .
- 2) According to the length N of the sequence, use table 2 below to find $Sc1$ or $Sc2$.

N mod 7	S_{0N-1}							
	0	1	2	3	4	5	6	7
1	0	6	4	2	7	1	3	5
2	0	3	7	4	5	6	2	1
3	0	5	3	6	2	7	1	4
4	0	4	1	5	6	2	7	3
5	0	2	5	7	1	3	4	6
6	0	7	6	1	3	4	5	2

Table 3. Circulation state table lookup

Signal Descriptions

The module pin out is shown in the figure below, and in table 1. The signals are conveniently organized into functional groups as follows:

Clock and Reset

The design is fully synchronous with a single clock signal. The reset signal is synchronous and needs to be asserted for at least one full clock cycle to reset internal logic.

Control signals

Init, *rate*, and *start_encode* are control signals whose operation is described in table 3 below.

Data signals

The serial data are shifted in on *din*. The encoded data are shifted out on *dout*, as shown in figure 3, and as shown in the waveforms below.

Input Flow Control

Two signals control the flow of data into the device. *Din_rdyin_n* is an active low signal indicating to the device that valid data is ready on *din*. *Din_rdyout_n* is an active low signal supplied by the device that indicates that it's ready to accept data in on *din*. Valid data are flowing into the device whenever both *din_rdyin_n* and *din_rdyout_n* are low.

Output Flow Control

Two signals control the flow of data out of the device. *Dout_rdyin_n* is an active low signal indicating to the device that the interface can receive data from the outputs. *Dout_rdyout_n* is an active low signal supplied by the device that indicates that it has valid data to shift out. Valid data are flowing out of the device whenever both *dout_rdyin_n* and *dout_rdyout_n* are low.

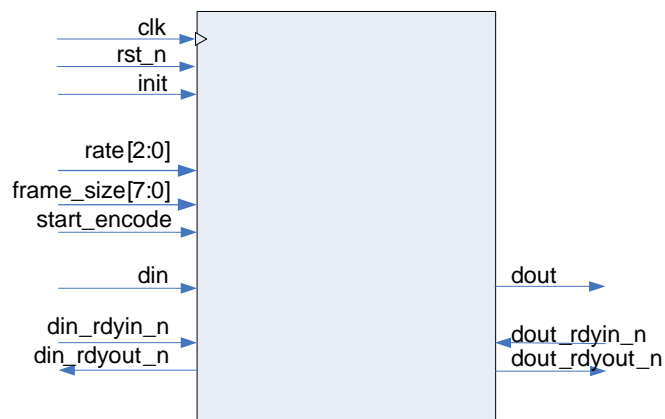


Figure 3 Component Pinout

Pin	Sense	Width	Description
clk	in	1	Clock
rst_n	in	1	Synchronous reset
din	in	1	Serial data (message) in
din_rdyin_n	in	1	Indicates serial data in is valid
din_rdyout_n	out	1	Indicates module ready to accept data (not all input buffers full)
dout	out	1	Encoder data out
dout_rdyin_n	in	1	Indicates to the module that it's ok to shift data out
dout_rdyout_n	out	1	Indicates that data is available to be shifted out
init	in	1	Initializes the interleaver and loads the block size
rate	in	3	Code rate: 00 = 1/2, 01 = 1/3, 10 = 1/4, 11 = 1/6
frame_size	in	7	Indicates size of current block. Initialized with "init" signal.
start_encode	in	1	Triggers the encode operation
Table 4. Module Pinout			

Waveforms

Initialization

The frame size defaults to 256 bytes (the minimum number specified in the WiMAX spec). Use the *init* signal and the *frame_size* signal as shown in fig. 4 below to specify a new frame size (in bits) for the component. To ensure proper operation of the device, make sure the input buffers are empty before specifying a new frame size, as the interleaver parameters are computed each time a new value is specified for the frame size. Thus the *init* signal can also be used to perform a device “soft” reset by asserting *init* without changing the *frame_size*.

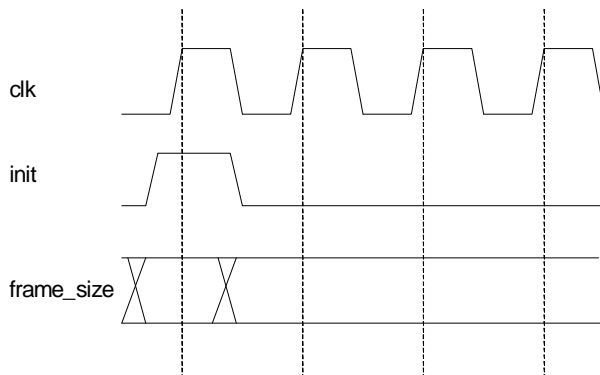


Figure 4. Initialization

In order to read and operate on both normal and interleaved data, an entire frame should be loaded into the input buffer before the encoding process will start. The encoding will start when the *start_encode* signal is asserted. This signal can also be used to re-send a data frame, or to force early termination of a data frame.

If a data frame is terminated with *start_encode* before the number of bits indicated by *frame_size* have been loaded, the device will automatically pad the end of the data frame with zeros.

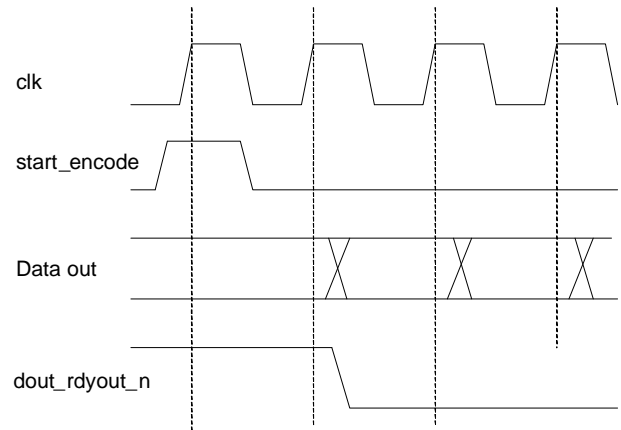


Figure 5. Start Encode

Input Flow Control

The input flow control signals consist of *din_rdyin_n* and *din_rdyout_n*. These signals accompany the data in signal *din*. A ‘0’ on the *din_rdyin_n* indicates to the device that valid data is being shifted into the device on *din*. The device uses *din_rdyout_n* to indicate that it’s able to accept more data. Proper input operation is simple: valid data is flowing into the device’s input buffers when both *din_rdyin_n* and *din_rdyout_n* are logic ‘0’. Because there is an input “pre-buffering” circuit that allows zero-wait-state input operation, the *din_rdyout_n* signal can effectively be ignored in the middle of loading a frame into the device.

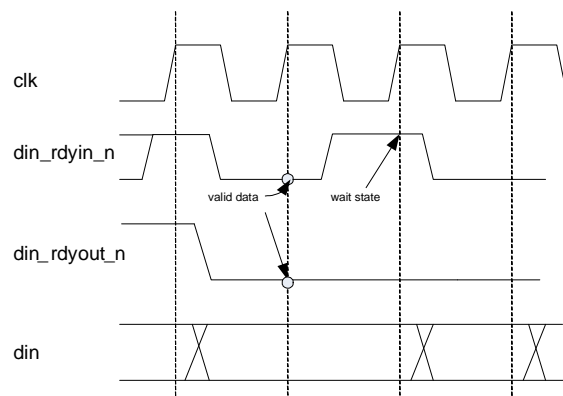


Figure 6. Input timing

Output Flow Control

The output flow control signals consist of *dout_rdyin_n* and *dout_rdyout_n*. These signals accompany the data signals *zk*, *zk*, and *zk_prime* out of the device. A '0' on the *dout_rdyin_n* indicates to the device that it's OK to shift data out of the device. The device uses *dout_rdyout_n* to indicate that it has valid data to send. Proper output operation is simple: valid data is flowing out of the device when both *dout_rdyin_n* and *dout_rdyout_n* are logic '0'. The *dout_rdyin_n* signal can be tied low for fastest operation.

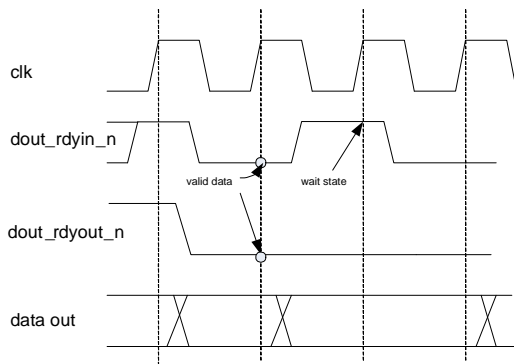


Figure 7. Output timing

Module Verification

The SAL50100E has been subjected to extensive verification to ensure the highest quality product possible. A comprehensive test plan was implemented which included the following:

- All WiMAX turbo code frame sizes

- High-quality random data source
- High-quality random noise source
- Extensive flow-control simulations
- Verification of operation against known data sequences

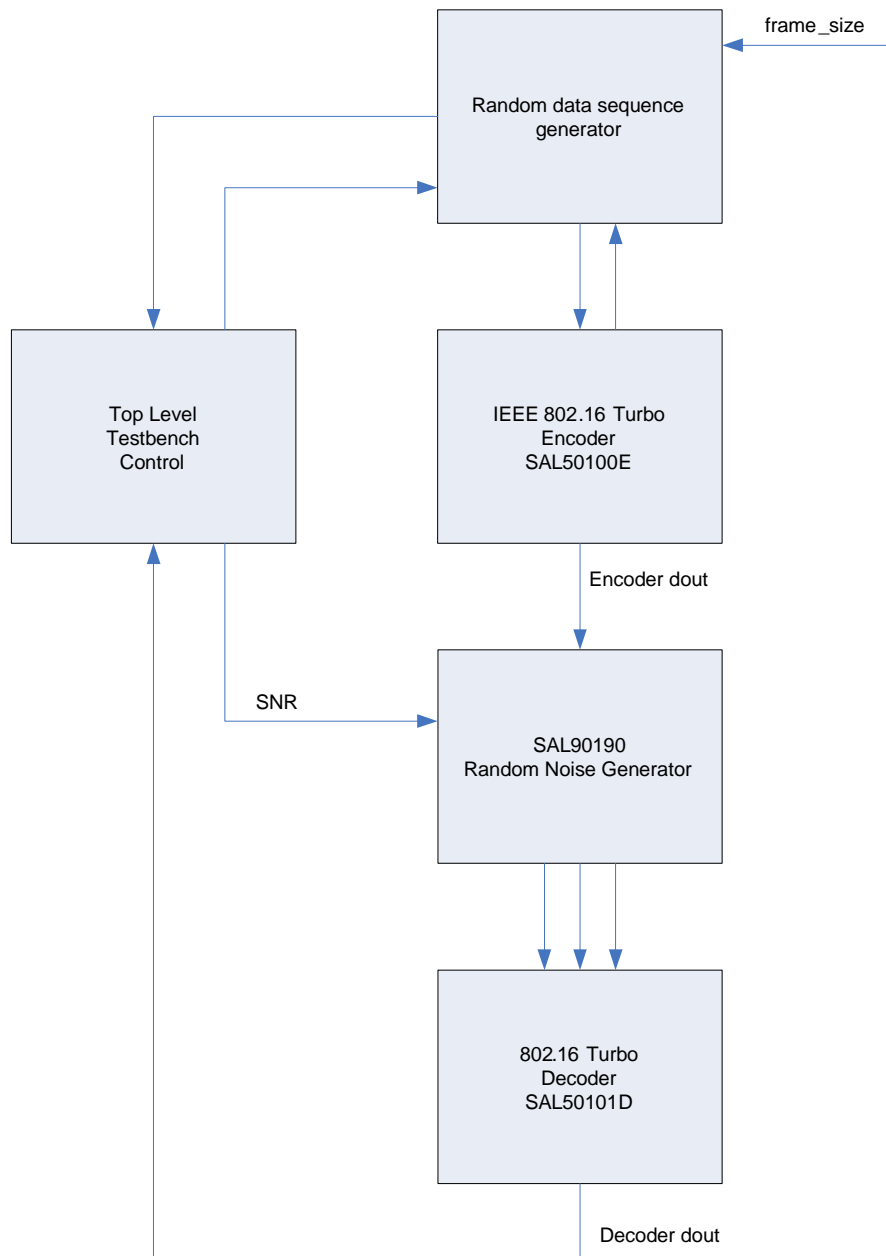
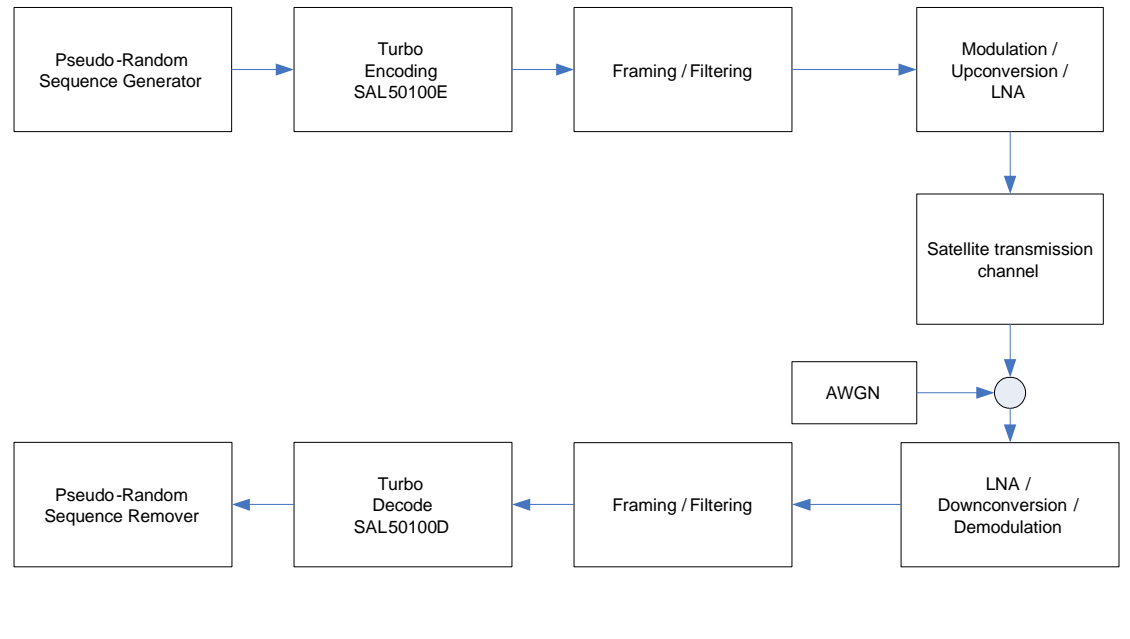


Figure 8: Testbench Block Diagram

Application: Wireless Internet

Parallel Concatenated Convolutional Codes (PCCC) offer the best error correction performance of any known codes for code rates < 0.7 and moderate frame sizes (~ 1024 bits). The SAL50100E is designed specifically for use in wireless WAN systems.



Ordering Information

Salamander Error Correction currently has 1 WiMAX-compatible turbo encoder IP module available:

SAL50100E 802.16-2004 Compatible Turbo Encoder

About Salamander:

Salamander Error Correction develops and sells error correction modules of the highest quality worldwide.

Salamander Error Correction is a division of Komodo Industries, Inc.

Salamander Error Correction:
3364 Via Alicante
La Jolla, CA 92037

sales@salamander-ecc.com