



Features

- CCSDS 131.0-B-1 compatible
- Integrated channel coding solution
- Reed-Solomon encoder
 - N = 255, k = 239, t = 8
 - 8-bit symbols
 - Interleaving
 - Field generator polynomial:
 $F(x) = x^8 + x^7 + x^2 + x + 1$
 - Generator polynomial:
 $g(x) = \prod (x - \alpha^{11})$
- Data randomizer
- Attached Sync Marker insertion
- Convolutional encoder
 - Constraint length 7
 - Nominal code rate 1/2
 - Punctured rates 2/3, 3/4, 5/6, 7/8
- Low, 2 clock cycle latency
- Simple handshake protocol for reliable interfacing
- Fully synchronous design
- Very high speed operation > 200 MHz
- Comprehensive verification plan provided

General Description

The SAL40400E consists of verilog IP for implementing the CCSDS channel encoding.

The device consists of an 8-error-correcting Reed Solomon encoder, data randomizer, Attached Sync Marker (ASM) insertion, and a convolutional encoder.

The device consists of multiple modules, each capable of being bypassed. Modules that are bypassed are disabled to save power.

Incoming data are first (optionally) encoded by the Reed-Solomon encoder. The data are then (optionally) passed to the data randomizer module, which scrambles the data to ensure sufficient transitions in the data. An Attached Sync Marker is then (optionally) appended to the data, to allow data frame synchronization. Finally, the data are (optionally) passed to the convolutional encoder module, which further encodes the data stream with optional puncturing.

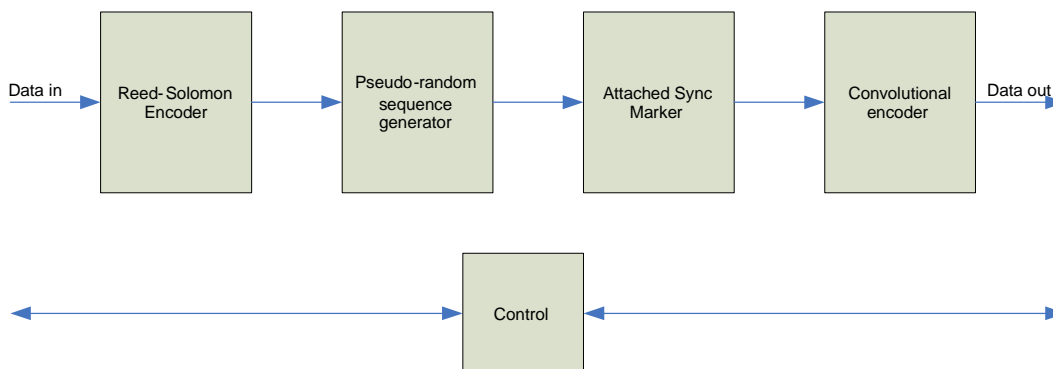


Figure 1. Block diagram

Theory of Operation

Reed-Solomon Encoder

The CCSDS recommendations specify two Reed-Solomon constructions, differing in their error correction ability. As one might expect, the greater the error correcting ability, the greater the overhead in terms of redundant symbols. The specification additionally outlines an interleaving scheme, which allows a larger data block to be defined. The SAL40400E device supports all the options defined in the CCSDS spec:

- $n = 255, k = 239, t = 8$
- 8-bit symbols
- Interleaving
- Field generator polynomial:
 $F(x) = x^8 + x^7 + x^2 + x + 1$
- Generator polynomial:
 $g(x) = \prod (x - \alpha^{1j})$

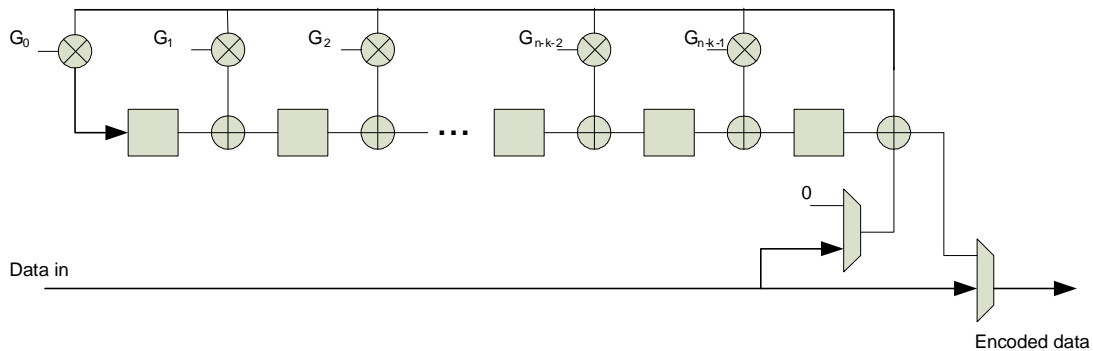


Figure 2. Reed-Solomon block

Pseudo-Randomizer

In order to ensure proper receiver operation, the data stream must be sufficiently random. The pseudo-randomizer in the SAL40400 products is the preferred method to ensure sufficient randomness for all combinations of CCSDS-recommended modulation and coding schemes. The presence or absence of pseudo-randomization is fixed for a physical channel, and its existence is understood *a priori* by the receiver.

Randomization is achieved by an exclusive OR of each bit of the Codeblock or Transfer Frame with a standard pseudo-random sequence. On the receiving end it is again applied to de-randomize the data after convolutional coding (if used) and Codeblock synchronization.

The pseudo-random sequence is generated using the circuit shown below, which implements the following polynomial:

$$h(x) = x^8 + x^7 + x^5 + x^3 + 1$$

This sequence begins at the first bit of the Codeblock or Transfer Frame and repeats after 255 bits, continuing repeatedly until the end of the Codeblock or Transfer Frame.

The sequence generator is initialized to all-ones state at the start of each Codeblock or Transfer Frame.

The first 40 bits of the pseudo-random sequence from the generator are:

1111 1111 0100 1000 0000 1110 1100 0000
1001 1010

The leftmost bit is the first bit of the sequence to be XOR'd with the first bit of the Codeblock or Transfer Frame, etc.

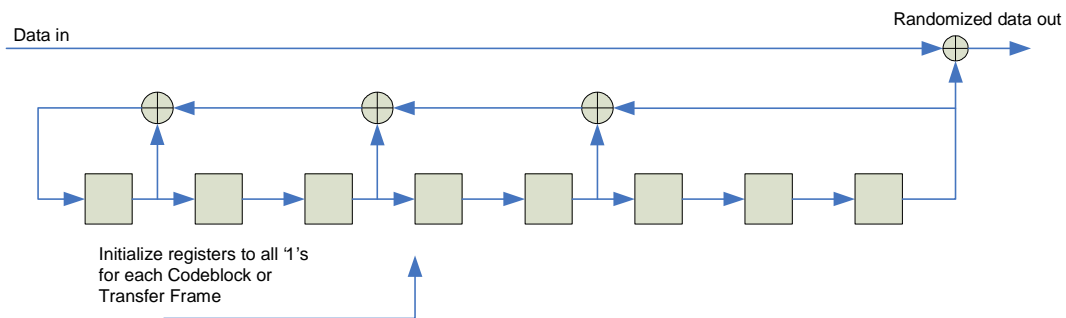


Figure 3. Randomizer block

Attached Sync Marker

Synchronization of the data transferred across a CCSDS compliant communication system is achieved by using a stream of fixed length data blocks separated by a specific known bit pattern between them. This bit pattern is known as an Attached Sync Marker, or ASM. The data blocks are known as Codeblocks if Reed-Solomon (or turbo) coding is used on the data prior to randomization. Otherwise, the data block is known as a Transfer Frame. After the attachment of ASM, the data block is known as a Channel Access Data Unit.

Although not optional according to the CCSDS recommendations, the ASM is optional in this device to provide maximum flexibility for the user. The default behavior is for the ASM to be appended to the RS-coded and randomized Codeblock or the Transfer Frame, if Reed-Solomon coding is bypassed.

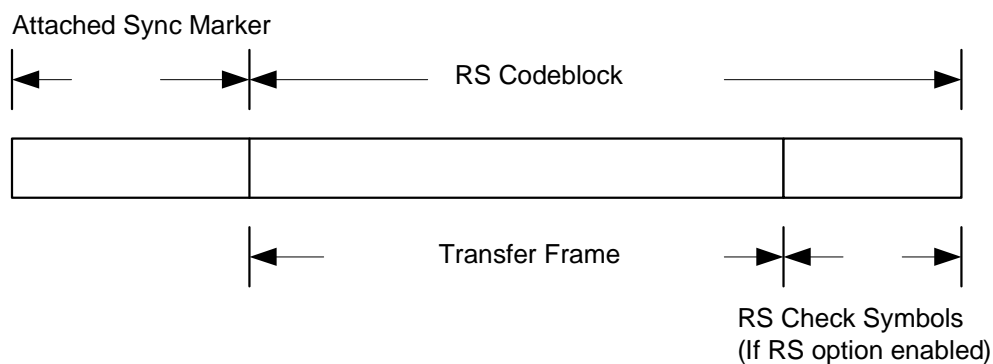


Figure 4. Sync Marker format

Convolutional Encoder

The basic code is a rate 1/2 constraint length 7 transparent code which is well suited to channels with predominantly Gaussian noise. The device supports the nominal rate 1/2 code as well as all the punctured rates as defined in the spec.

The encoder is of the non-systematic non-recursive type, with connection polynomials $g_1 = 171$ and $g_2 = 133$. The output of g_2 is inverted in the non-punctured (rate 1/2) mode.

Its operation is very simple: a serial stream of data is shifted into the device, and an encoded serial data stream is shifted out of the device.

Refer to figure 1. If the **punct** bit is not set, then puncturing is not enabled and the code is a rate 1/2 code, i.e., 2 bits are shifted out of the encoder for every bit shifted into the encoder.

The bits are output in the order $c_1(1), c_2(1)/, c_1(2), c_2(2)/, \text{etc.}$, where the slash indicates that the c_2 bits are inverted. The c_2 bits are inverted to ensure that there are sufficient bit transitions in the output of the encoder to allow the receiver to achieve bit synchronization.

If the **punct** bit is set, then some of the bits are removed from the nominal rate 1/2 output by a technique known as puncturing to achieve a higher code rate. Available code rates are 2/3, 3/4, 5/6, and 7/8. Although these rates make more efficient use of the available channel bandwidth, they suffer in their error correcting performance due to the removal of some redundant bits in the output. The puncture patterns are shown below:

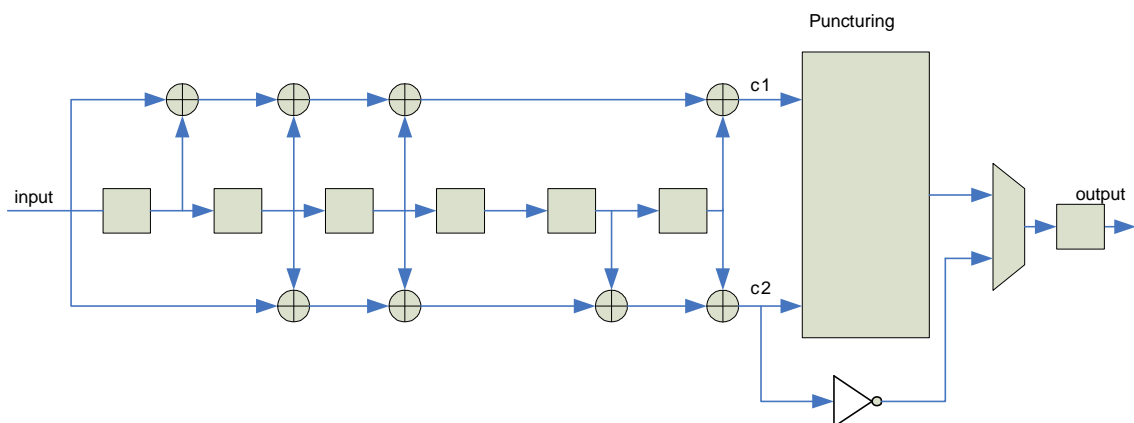


Figure 5. Convolutional encoder

Puncture pattern 1 = transmit 0 = don't transmit	Code Rate	Output Sequence
C1: 1 0 C2: 1 1	2/3	c1(1), c2(1), c2(2), ...
C1: 1 0 1 C2: 1 1 0	3/4	c1(1), c2(1), c2(2), c1(3), ...
C1: 1 0 1 0 1 C2: 1 1 0 1 0	5/6	c1(1), c2(1), c2(2), c1(3), c2(4), c1(5), ...
C1: 1 0 0 0 1 0 1 C2: 1 1 1 1 0 1 0	7/8	c1(1), c2(1), c2(2), c2(3), c2(4), c1(5), c2(6), c1(7), ...
Table 1. Puncture patterns		

Signal Descriptions

The module pinout 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.

Flow Control signals

Three signals control flow of data into the device, *din_rdyin_n* and *din_rdyout_n*. The *din_rdyin_n* signal indicates that data into the device is valid. The *din_rdyout_n* signal indicates that the device is ready to receive data. Valid data is flowing into the device if *din_rdyin_n* and *din_rdyout_n* are both asserted (low). Additionally, because the device can support short frames, the *in_dp_n* signal is used to indicate to the device which input data are part of the current data block. The *in_dp_n* signal is low during the parity interval.

Two signals control flow of data out of the device, *dout_rdyin_n* and *dout_rdyout_n*. The *din_rdyin_n* signal is used to indicate to the device that the external interface is ready to receive data. The *dout_rdyout_n* signal indicates that data out of the device is valid. Valid data is flowing out of the device if *dout_rdyin_n* and *dout_rdyout_n* are both asserted (low).

Data signals

The data are clocked in on *din* and clocked out on *dout*.

Configuration signals

In addition to the above defined signals, there are a number of mode bits which are defined in table 2 below.

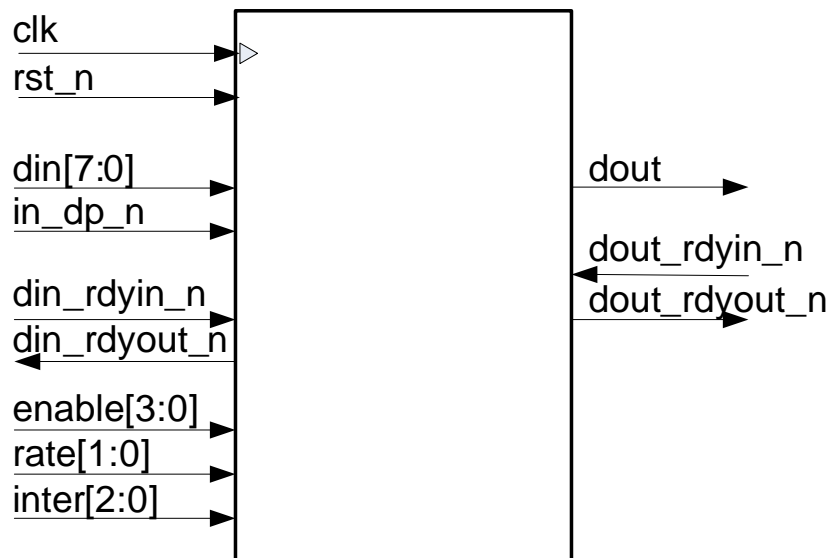


Figure 6: Component pinout

Pin	Sense	Width	Description
clk	in	1	Clock
rst_n	in	1	Synchronous reset
din	in	8	Serial data (message) in
din_rdyin_n	in	1	Indicates serial data in is valid
din_rdyout_n	in	1	Indicates device is ready to receive data in
in_dp_n	in	1	'1' indicates data in, '0' = calculate Reed-Solomon parity
dout	out	1	Data out
dout_rdyout_n	out	1	Indicates that data is available to be shifted out
enable	in	4	Individual internal module enables: enable[0] = enable Reed-Solomon enable[1] = enable Pseudo-Randomizer enable[2] = enable Attached Sync Marker enable[3] = enable convolutional coding
rate	in	2	Convolutional code rate: 00 = 2/3, 01 = 3/4, 10 = 5/6, 11 = 7/8
inter	in	3	Reed-Solomon Interleave: 001/110/111 = 1, 010 = 2, 011 = 3, 100 = 4, 101 = 5, 000 = 8
Table 2. Component pinout			

Waveforms

Input

The input functional timing is shown below. *Din_rdyin_n* is used as an input data enable, *in_dp_n* is used to indicate when data (as opposed to parity) is being shifted into the device.

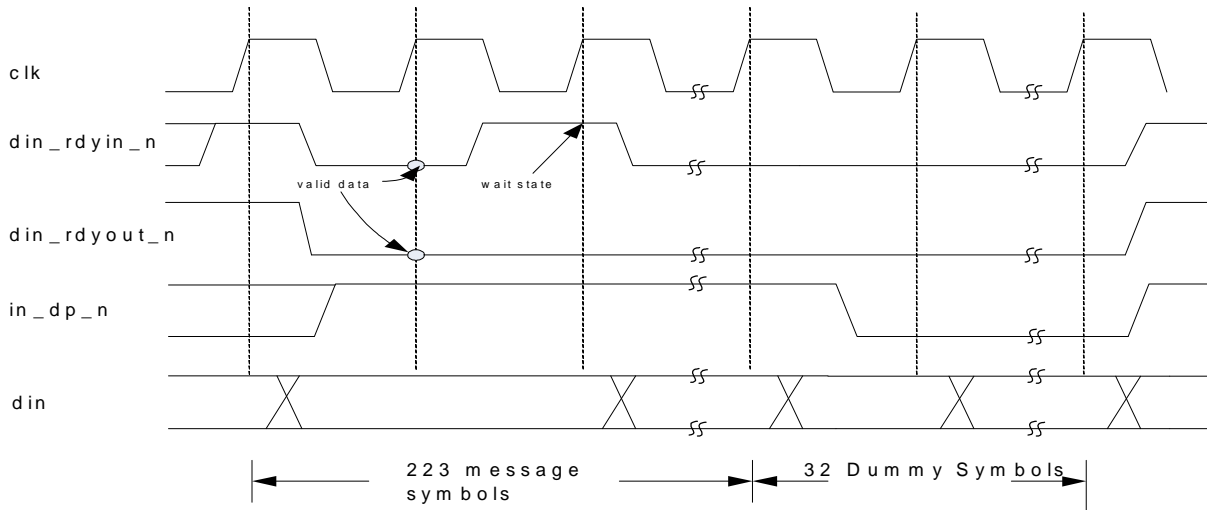


Figure 3: Input timing

Output

The output functional timing is shown below. *Dout_rdyout_n* is used as an output data ready indication, *out_dp_n* is used to indicate when data (as opposed to parity) is being shifted out of the device.

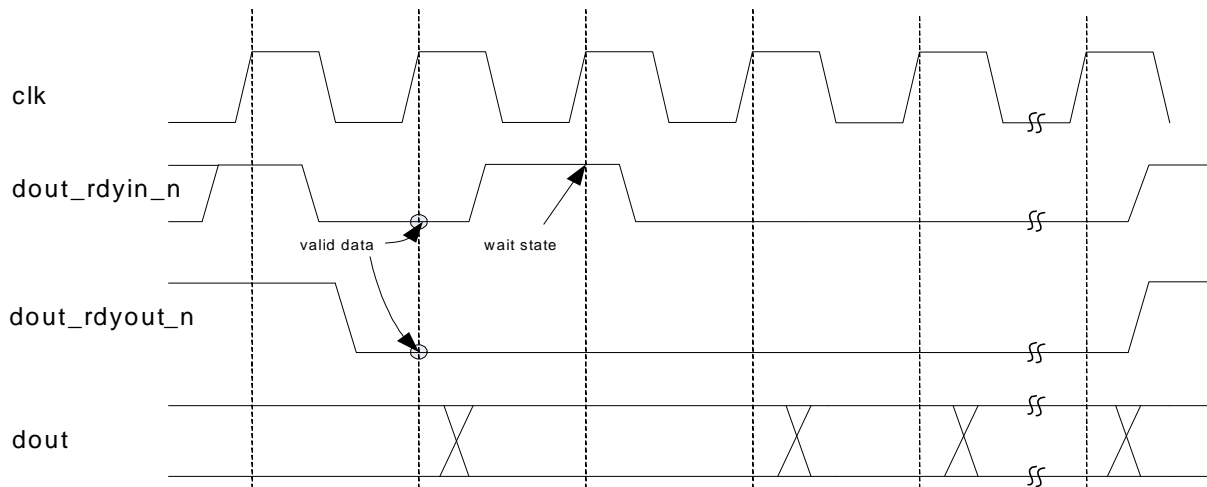


Figure 4: Output timing

Module Verification

The SAL40400E has been subjected to extensive verification to ensure the highest quality product possible. A comprehensive test plan was implemented which included the following:

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

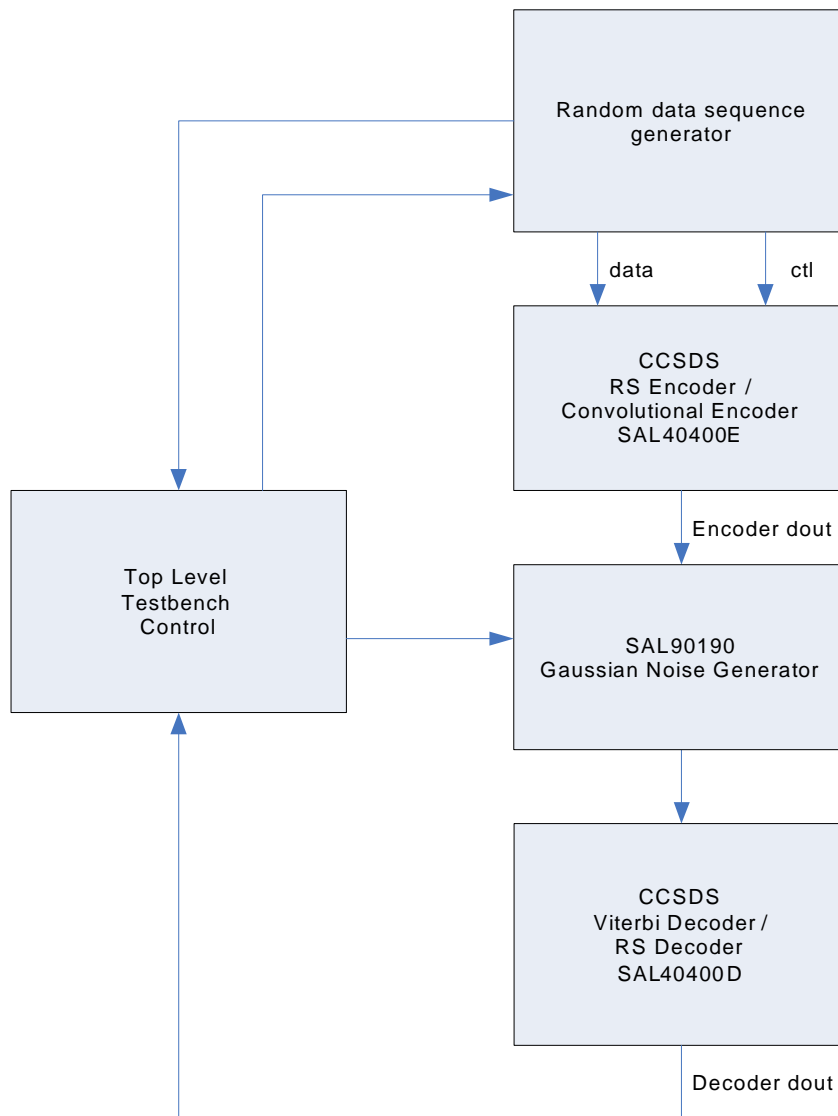
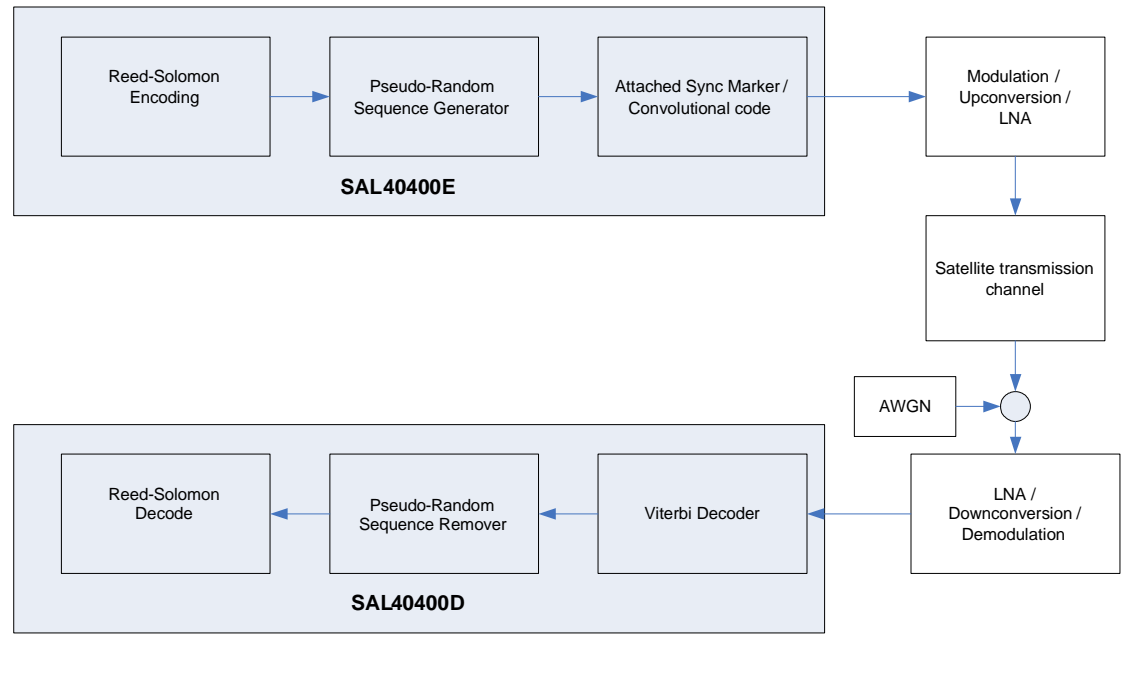


Figure 5: Testbench Block Diagram

Application: Satellite System

The CCSDS channel coding forms an integral part of many satellite telemetry systems.



Ordering Information

Salamander Error Correction currently has 2 CCSDS-compatible Reed-Solomon / Convolutional encoder IP modules available:

SAL40400E t = 8 CCSDS RS / Convolutional encoder

SAL40401E t = 16 CCSDS RS / Convolutional 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