



### Features

- Constraint length 7
- Nominal code rate  $\frac{1}{2}$
- Punctured rates  $\frac{2}{3}$ ,  $\frac{3}{4}$ ,  $\frac{5}{6}$ ,  $\frac{7}{8}$
- Simple handshake protocol for reliable interfacing
- Fully synchronous design
- Medium speed operation
- Erasure support
- 4-bit soft input
- medium speed – 16 clocks-per-bit
- Frame synchronization mode
- Comprehensive verification plan provided

### General Description

The SALxx301D consists of verilog IP for implementing a 64-state Viterbi decoder. The basic code is a rate  $\frac{1}{2}$  constraint length 7 transparent code which is well suited to channels with predominantly Gaussian noise. The device supports the nominal rate  $\frac{1}{2}$  code as well as all the punctured rates as defined in the spec.

The decoder uses the maximum-likelihood (Viterbi) algorithm to decode the  $(7, \frac{1}{2})$  convolutional code. The code is of the non-systematic non-recursive type, with default connection polynomials  $g1 = 171$  and  $g2 = 133$ .

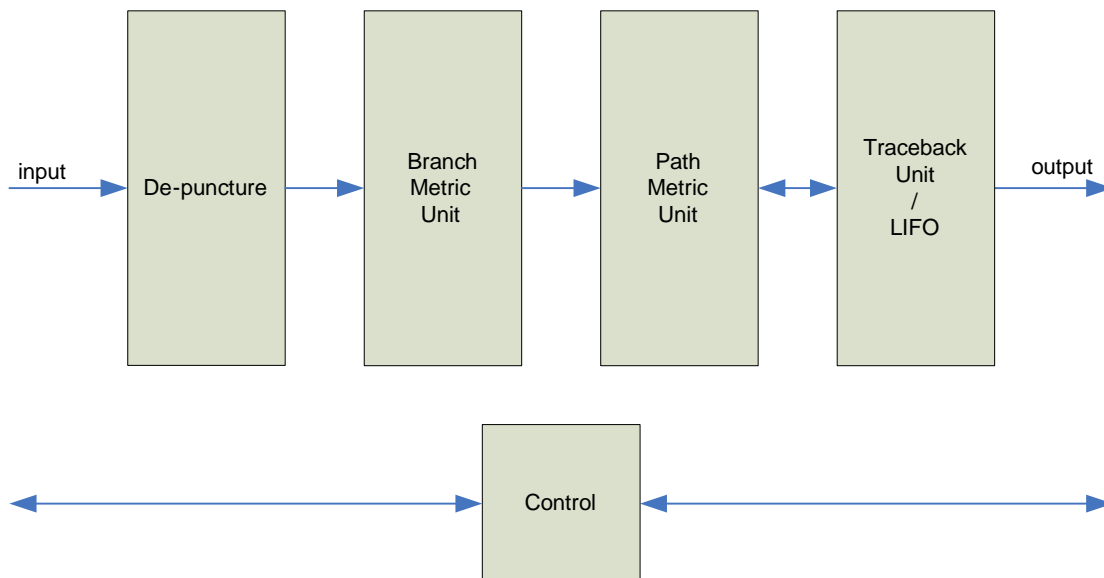


Figure 1: Decoder Block Diagram

## Theory of Operation

The SALxx301D is a (7, 1/2) Viterbi decoder designed as a trade-off between speed and complexity. The device accepts 4-bit soft input data samples formatted as shown in table 1. The data are optionally de-punctured depending on the code rate going into the device, inserting zero samples where necessary to re-make the nominal rate 1/2 code, as shown in table 2 below.

The branch metric unit assigns to the incoming data a set of metrics based on different assumptions of the current state of the code trellis. The branch metrics are then passed to the path metric unit and are added to the accumulated path metric and the most likely path metric is selected and stored for subsequent comparison.

After a traceback depth's worth of data is captured, the traceback unit works backward through the trellis, pulling the most likely bits from the survivor memory as it goes. The data are stored in a Last-In-First-Out (LIFO) buffer and made available to the output of the device at the end of the traceback sequence.

| in_mode[1:0] | Input format   |
|--------------|----------------|
| 00           | 2's compliment |
| 01           | sign-magnitude |
| 10           | binary         |
| 11           | hard input     |

Table 1. input mode

| Puncture pattern<br>1 = accept bit<br>0 = insert '0' | Code Rate | input Sequence                                      |
|--|-----------|---|
| X: 1 0<br>Y: 1 1                                     | 2/3       | X(1), Y(1), Y(2), ...                               |
| X: 1 0 1<br>Y: 1 1 0                                 | 3/4       | X(1), Y(1), Y(2), X(3), ...                         |
| X: 1 0 1 0 1<br>Y: 1 1 0 1 0                         | 5/6       | X(1), Y(1), Y(2), X(3), Y(4), X(5), ...             |
| X: 1 0 0 0 1 0 1<br>Y: 1 1 1 1 0 1 0                 | 7/8       | X(1), Y(1), Y(2), Y(3), Y(4), X(5), Y(6), X(7), ... |

Table 2. 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.

The init signal is used as a “soft” reset signal, to return the device to the zero state of the trellis.

### Control signals

Two 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.

Two signals control flow of data out of the device, *dout\_rdyout\_n* and *dout\_rdyin\_n*. The *dout\_rdyout\_n* signal indicates that data out of the device is valid. The *dout\_rdyin\_n* signal indicates to the device that it's OK to shift data out of the device.

### Sync

The sync signal is used when the bit synchronization is not known. Asserting this signal causes the device to make multiple attempts to decode the data based on initial state assumptions and to adjust those assumptions based on the relative values of the path metrics. This signal should not be used in normal operation when the frame sync is known, because its use affects the throughput and power consumption of the device.

### Data signals

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

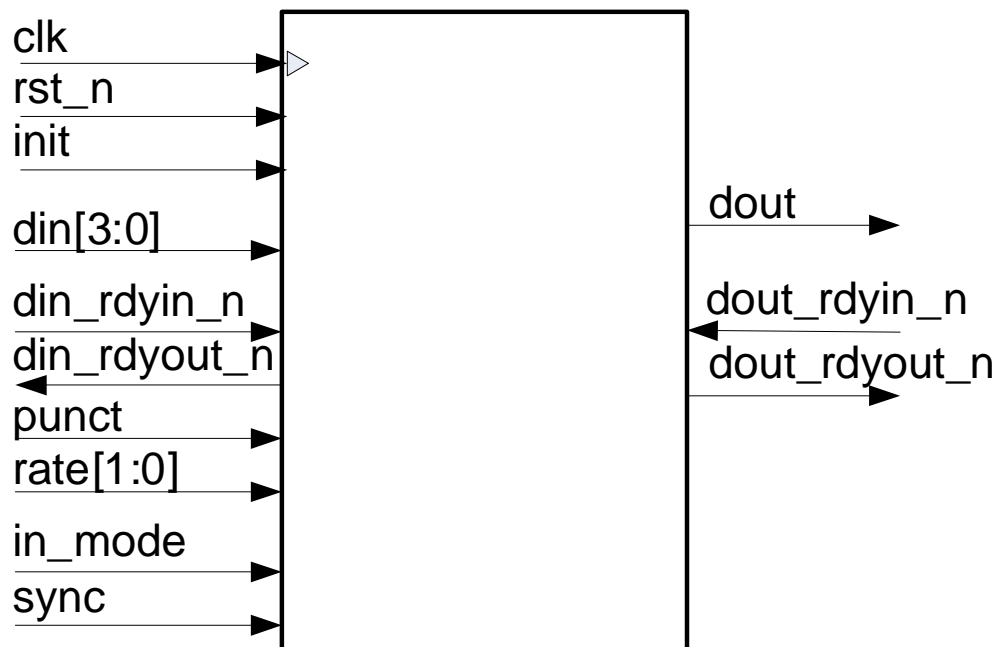


Figure 2: Component pinout

| Pin                       | Sense | Width | Description  |
|---------------------------|-------|-------|--|
| clk                       | in    | 1     | Clock  |
| rst_n                     | in    | 1     | Synchronous reset  |
| init                      | in    | 1     | Soft reset. Sets the trellis state to the zero state.            |
| punct                     | in    | 1     | When set, code is punctured, else code is rate 1/2               |
| rate                      | in    | 2     | Code rate: 00 = 2/3, 01 = 3/4, 10 = 5/6, 11 = 7/8                |
| in_mode                   | in    | 2     | Input data format (see table 1)                                  |
| sync                      | in    | 1     | Instructs device to acquire synchronization of input data stream |
| din                       | in    | 4     | Serial data (message) in   |
| din_rdyin_n               | in    | 1     | Indicates serial data in is valid                                |
| din_rdyout_n              | out   | 1     | Indicates device is ready to receive data in                     |
| dout                      | out   | 1     | Data out   |
| dout_rdyin_n              | in    | 1     | Indicates to device that it's ok to shift data out               |
| dout_rdyout_n             | out   | 1     | Indicates that data is available to be shifted out               |
| Table 3. Component pinout |       |       |  |

## Waveforms

### Input

The input functional timing is shown below. *Din\_rdyin\_n* is used as an input data enable, *din\_rdyout\_n* is used to indicate when the device is ready to receive data.

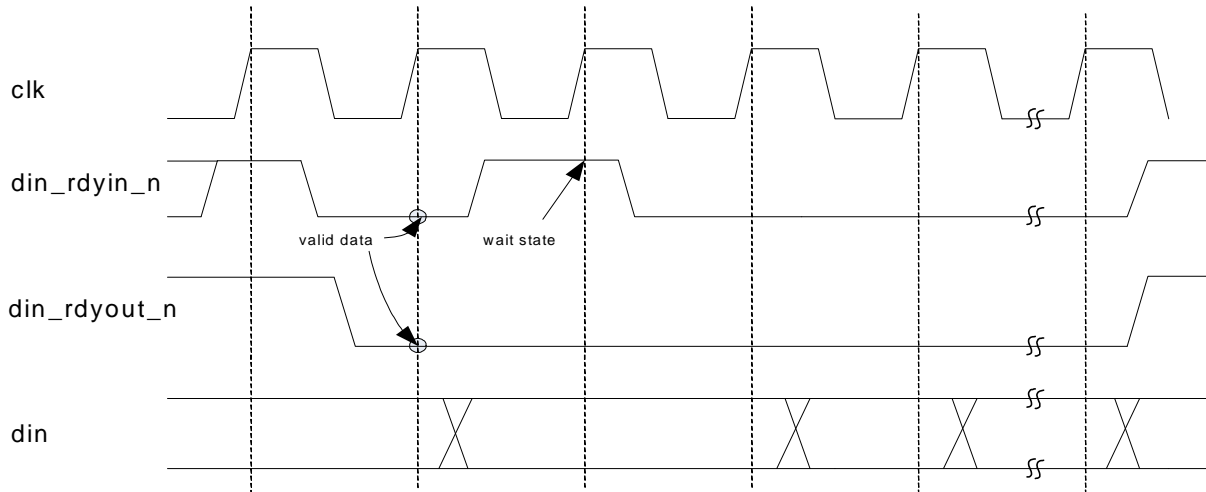


Figure 3: Input timing

### Output

The output functional timing is shown below. *Dout\_rdyout\_n* is used as an output data ready indication, *dout\_rdyin\_n* is used to indicate to the device that it's OK to shift data out.

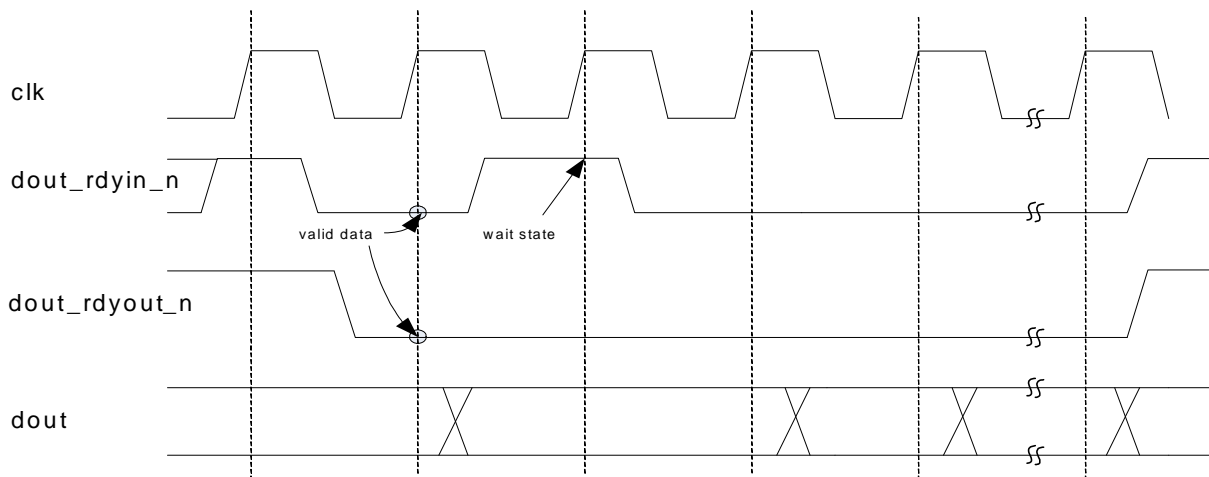


Figure 4. Output timing

## Module Verification

The SALxx301D 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

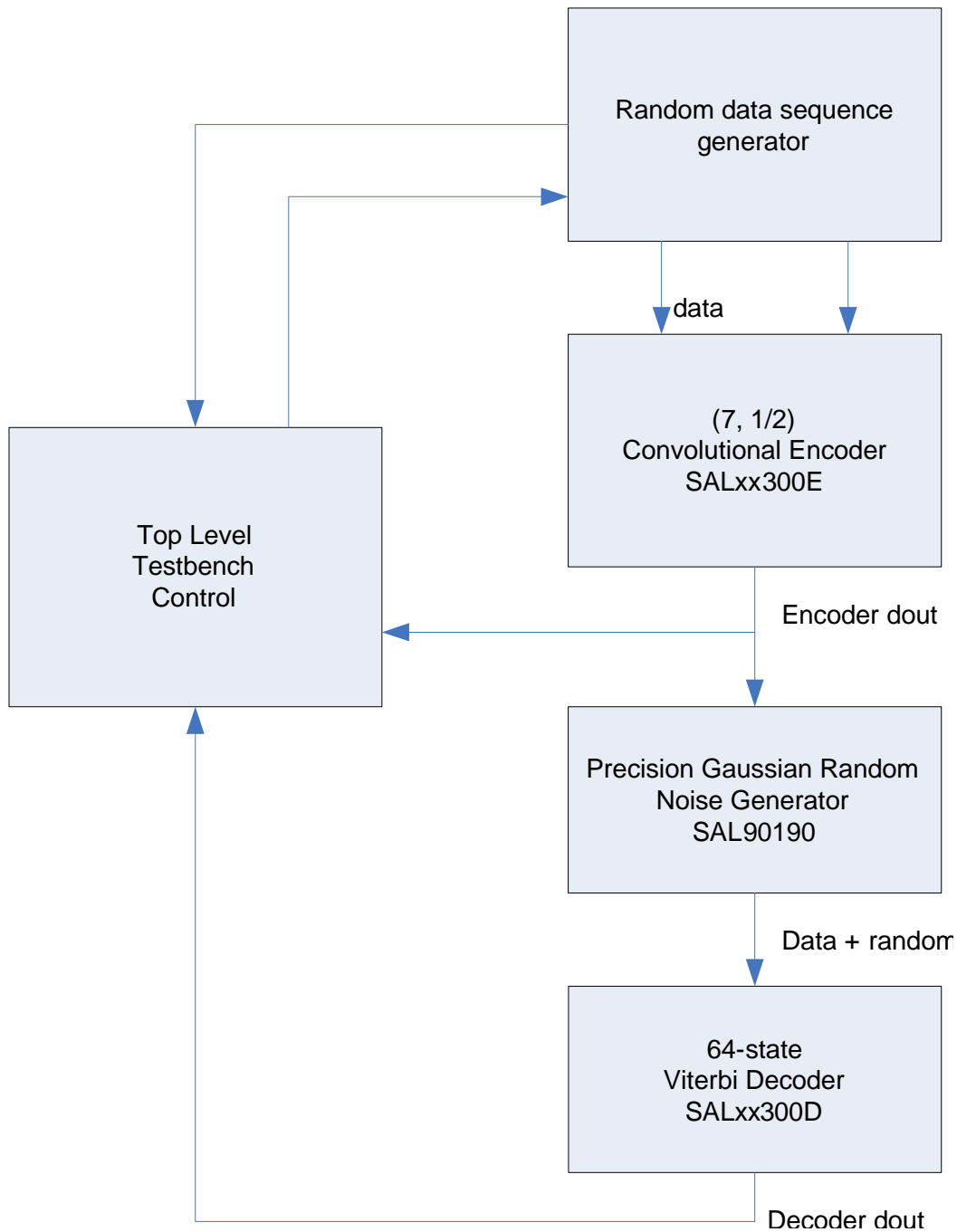
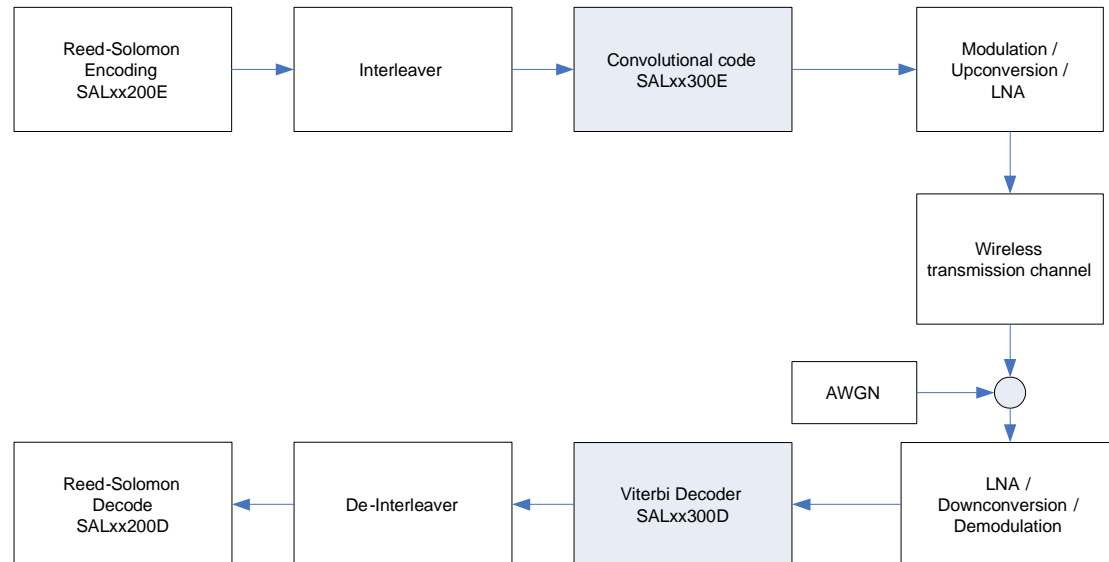


Figure 5: Testbench Block Diagram

## Application: Wireless Internet System

A Viterbi decoder forms an integral part of a wireless Internet telemetry system.



## Ordering Information

Salamander Error Correction currently has 6 generic

**SALxx300D** Viterbi decoder, low speed

**SALxx301D** Viterbi decoder, medium speed

**SALxx302D** Viterbi decoder, high speed

**SALxx320D** Viterbi decoder, constraint length 9, low speed

**SALxx321D** Viterbi decoder, constraint length 9, medium speed

**SALxx322D** Viterbi decoder, constraint length 9 Viterbi decoder IP modules available:

## 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:  
5330 Carroll Canyon Rd  
San Diego, CA 92121  
(858) 373-2112  
fax: (858) 373-1224  
sales@salamander-ecc.com