

Features

- For use in 3GPP2 (CDMA2000) systems. Fully compliant with the 3GPP2 standard C.S0002-D v1.0
- Double Buffered input enables high speed operation
- Generic memory interface for easy ASIC integration
- Simple handshake protocol for reliable interfacing
- Fully synchronous design
- Full 3GPP2 compliant hardware interleaver. No external calculation required.
- Interleaver frame size selectable on a frame-by-frame basis
- Buffer-ready interrupt provided
- Comprehensive verification plan provided
- Full block size range of 40 – 5114 bits supported

General Description

The SAL21101D consists of verilog IP for decoding the parallel concatenated convolutional (turbo) code defined by the 3GPP2 standard. Refer to Fig. 1 below.

The decoder takes in 6-bit values for the unencoded (“systematic”) data and two parity values, one which is the result of encoding the data with a simple convolutional encoder and one which is the result of encoding a permuted version of the data through an identical simple convolutional encoder. The permutation is described in detail in the 3GPP2 specification.

The data are operated on iteratively by the Maximum A Posteriori (MAP) decoder module, first looking at the normal ordered data and parity, then looking at the interleaved sequence and parity. At each step in the process a correction factor is refined and stored as “extrinsic” data for use in the following step.

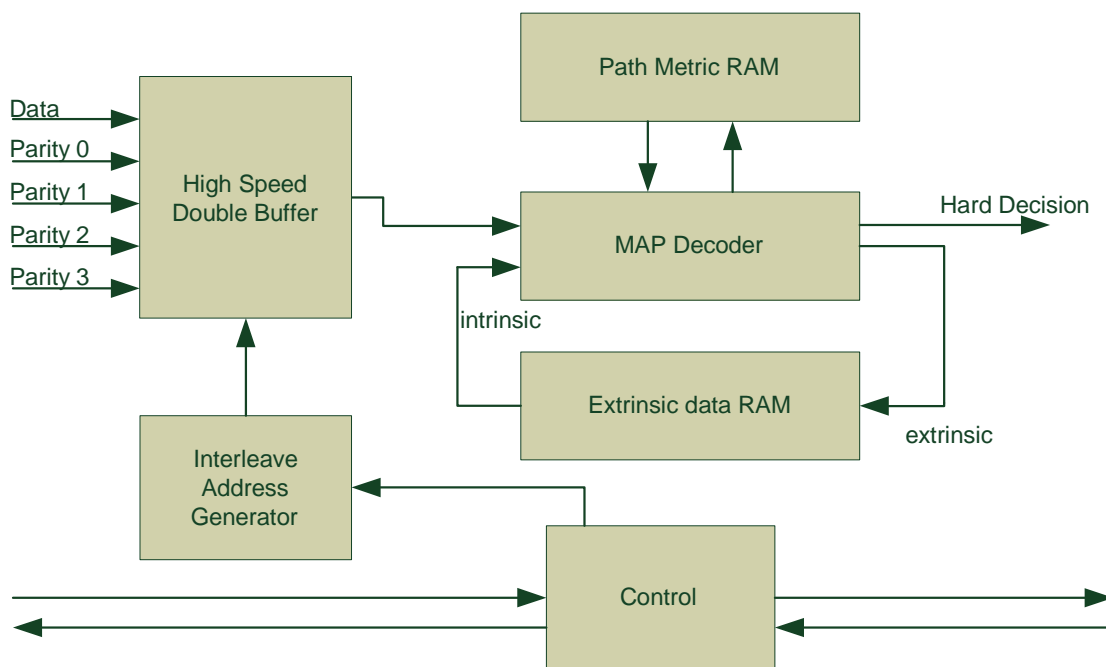


Figure 1: Decoder Block Diagram

Theory of Operation

The SAL21101D is a self-contained, high quality error correction coder designed specifically for use in CDMA2000 systems. The emphasis for the SAL21101D product is on achieving superior error correction performance at a moderate complexity. The reader is referred to the SAL21100D product for our lowest power and most efficient device. For the highest bitrate performance (at the expense of higher complexity) the reader is referred to the SAL21102D product.

The device receives 6-bit 2's-compliment data for unencoded data, parity, and interleaved parity. Data can also be received as signed-magnitude or unsigned binary, based on the *input_mode* signal.

input_mode[1:0]	Definition
00	6-bit 2's compliment
01	6-bit signed magnitude
10	6-bit unsigned binary
11	6-bit 2's compliment

Table 1. input_mode

The heart of the decoder is a soft-in / soft-out (SISO) decoder module that performs the full log-MAP algorithm iteratively.

The Decoder Constant

Because the decoder core operates in the logarithmic domain, the input data need to be converted to a logarithmic (log likelihood ratio) form. This reduces to multiplication of the input data by a constant based on the estimated signal to noise ratio of the channel.

The value of the decoder constant is calculated using the formula:

$$C = A\sigma^2\sqrt{m}/2 \quad (1)$$

Where σ^2 is the normalized noise variance. It has been shown [2] that no noise estimation is better than poor noise estimation, so the input scaling can optionally be disabled.

Max Extrinsic

In addition to input scaling, the extrinsic correction term can be limited to limit the rate at which the decoder converges on a solution.

Num Iterations

The number of decoder iterations can be specified using the *num_its* signal. The default value is 4.5. Optionally the device can be allowed to perform early termination, if the decoder converges on a solution before the specified number of iterations.

Mode Bits

There are a number of mode bits beside the input and output mode bits, and they are shown in the table below.

mode[4:0]	Definition, if set
mode[0]	Use decoder_const
mode[1]	Use max_extr
mode[2]	Use num_its, else use 4.5 iterations
mode[3]	Use early termination, if possible
mode[4]	Use force_decode (see below)

Table 2. mode bits

Output Mode

Data output can be either serial or parallel, depending on the value of the **out_mode** signal.

out_mode	Definition, if set
out_mode	Data out is 32-bit parallel

Table 3. output mode

If the **out_mode** bit is not set, the data appear one bit at a time on **dout[0]**. If the bit is set, the data come out 32 bits at a time on **dout[31:0]**.

Code Rate

The 3GPP2 standard supports 4 code rates $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, and $\frac{1}{5}$. The code rate currently being used is specified using the **rate** signal vector.

rate[1:0]	Definition, if set
00	rate = 1/2
01	rate = 1/3
10	rate = 1/4
11	rate = 1/5

Table 3. code rate

Device Operation

If the “Use force_decode” option (mode[4]) is set, decoding begins when the **force_decode** signal is asserted, otherwise decoding will begin automatically when a frame’s worth of data (plus 12 tail bits) has been loaded into the device. In devices with double-buffered input the device will accept a second buffer while processing the first.

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.

Control signals

Init, frame_size, d_const, max_extr, in_mode and out_mode are control signals whose operation is described below.

Data signals

The *din_s*, *din_p0*-*din_p3* are 6-bit input signals representing the systematic data and the two parity signals defined in the spec. *Dout* is the serial or parallel output decoded data.

The data input to the device are stored internally in FIFO order, the first bit being bit 0 of the data frame. In addition to the data frame, the device expects to see trellis termination or “tail” bits as defined in the spec.

The data out of the device are in FIFO order, and therefore don’t require an address to access them. The data can either be read out in serial fashion on bit 0 of dout, or in parallel fashion, depending on the value of out_mode.

Flow control signals

The “rdy_in_n” signals are active low indicators to the device that data is available on the input or that the external logic is able to accept data.

The “rdyout_n” signals are active low indicators from the device that it is ready to accept data on the input or to shift out data on the output

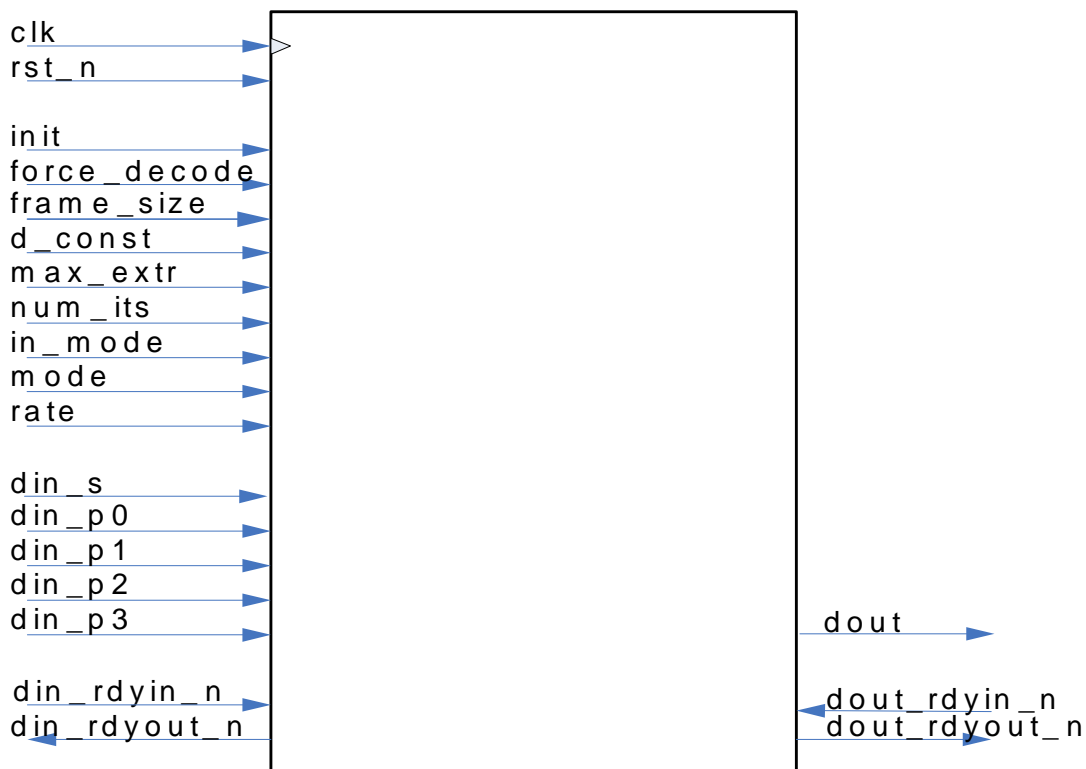


Figure 2: Component Pinout

Pin	Sense	Width	Description
clk	in	1	clock
rst_n	in	1	synchronous reset
din_s	in	6	serial data (message) in
din_p0	in	6	parity data in
din_p1	in	6	parity data in
din_p2	in	6	parity data in
din_p3	in	6	parity data 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	32	decoded data out (serial or parallel depending on mode)
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
force_decode	in	1	when enabled, decoding begins when this signal is asserted
frame_size	in	13	indicates size of current block. Initialized with "init" signal.
dec_const	in	8	decoder constant, based on estimated noise (see description above)
max_extr	in	8	maximum allowable extrinsic data (see description above)
in_mode	in	2	input mode: specify the data format of input bits
mode	in	5	various mode affecting device operation
out_mode	in	1	specify serial or parallel output format
rate	in	2	code rate

Table 4. Pin Definitions

Waveforms

Initialization

The frame size defaults to 17 bits (the minimum number specified in the 3GPP2 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*.

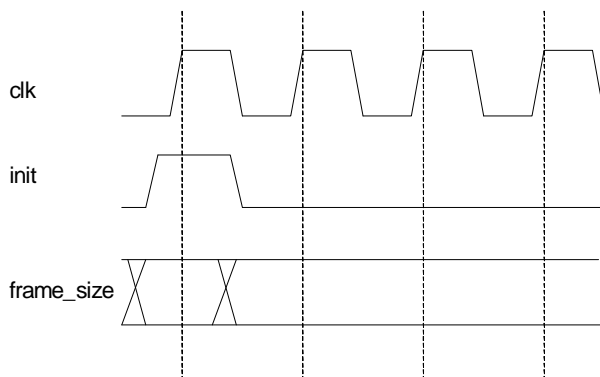


Figure 3: Initialization Timing

In order to decode using both normal and interleaved data, an entire frame must be loaded into the input buffer before the decoding process will start. The decoding will start automatically when the correct number of bits are shifted into the component. Alternatively, the *force_decode* signal can be used to force the decoder to begin decoding the data.

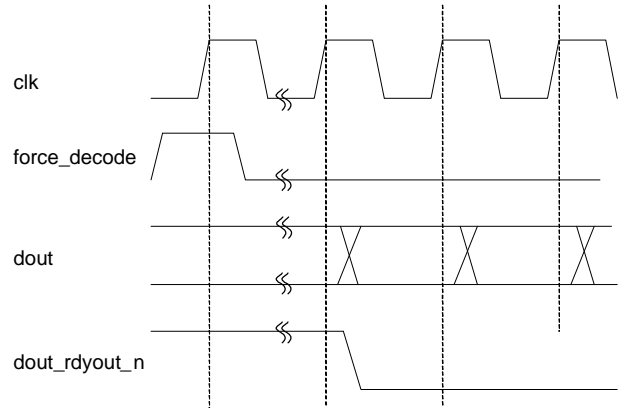


Figure 4: Force Decode Operation

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.

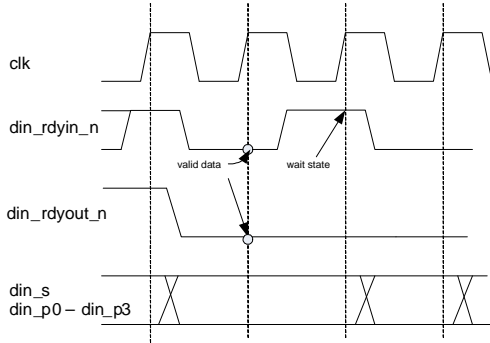


Figure 5: Input Flow Control

Output Flow Control

The output flow control signals consist of *dout_rdyin_n* and *dout_rdyout_n*. These signals accompany the data signals on *dout* 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 tied low for fastest operation.

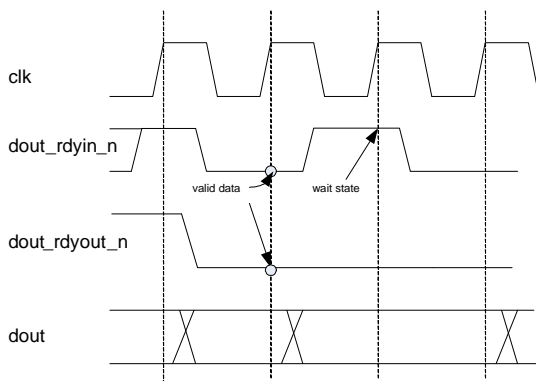


Figure 6: Output Flow Control

Tail Bits

The parallel convolutional turbo encoding process is completed by forcing the constituent coders to a known state (the 000 state, in this case). To do this, additional bits called tail bits must be sent by the turbo encoder. The actual bit pattern used is determined by the code rate as follows:

input bit	1/2	1/3	1/4	1/5
din_s	111000	222000	222000	333000
din_p0	111000	111000	111000	111000
din_p1	000000	000000	111000	111000
din_s	000111	000222	000222	000333
din_p2	000111	000111	000111	000111
din_p3	000000	000000	000111	000111

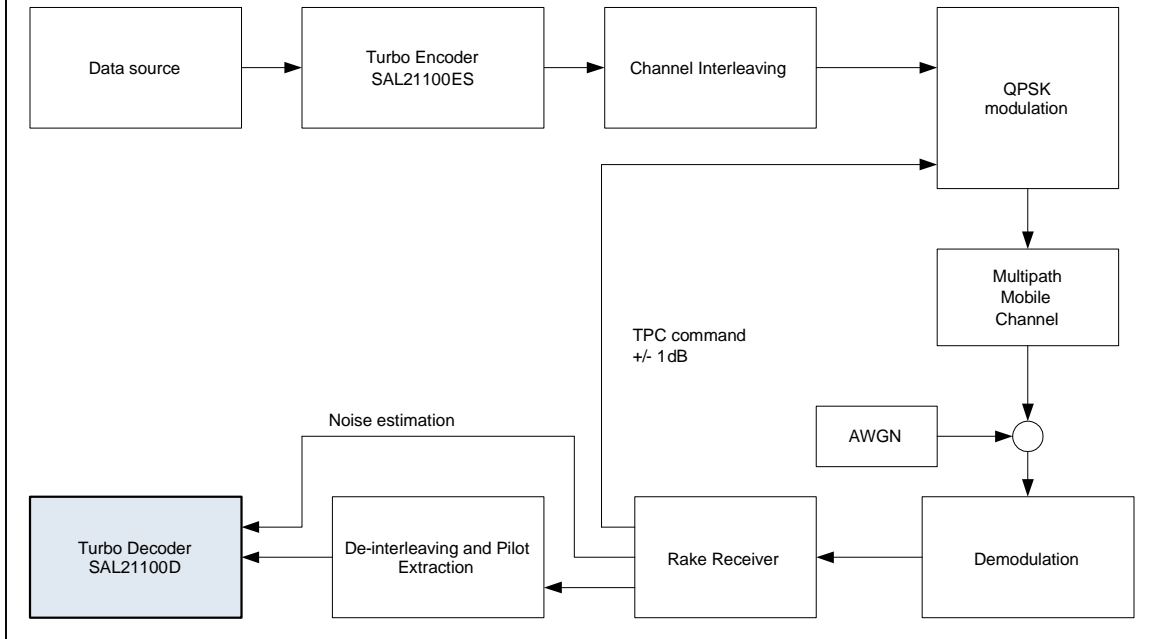
Table 5. tail bit puncture patterns

Figure 7: Tail Bits

These tail bits are received as shown in the figure, and the decoder will use the information to establish the correct starting point when working backward through the code trellis. Note that the *din_s* signals are repeated. This is because the tail bits are actually output from the encoder on the 'X' output, but that output is otherwise unused. The decoder expects to receive those bits on the *din_s* input.

Application: CDMA2000

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 SAL21101D is designed specifically for use in CDMA2000 systems.



Ordering Information

Salamander Error Correction currently has 3 3GPP2-compatible turbo decoder IP modules available:

SAL21100D Low Power

SAL21101D Standard Version

SAL21102D High Performance

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