

Features

- **Compliant with Standards:**
 - 3GPP TS 25.212 version 4.2.0
 - CCSDS 101.0-B-5
- **Throughput of 2Mbps for 3GPP at 30MHz, 7 Iterations**
- **Two's Complement Data/Parity Input**
- **Supports Depuncturing**
- **Variable Soft-Widths for Input Symbols, Branch Metrics, Path Metrics and LLRs**
- **User-Defined Number of States**
- **Variable Block Sizes During Runtime**
- **Programmable Number of Iterations**
- **Optional Hard Decision Storage**
- **Selectable Max-Log-Map or Log-Map Algorithm**
- **Programmable Pipe Stages for Convenient Memory Interfacing**
- **Optional Double Buffering**

General Description

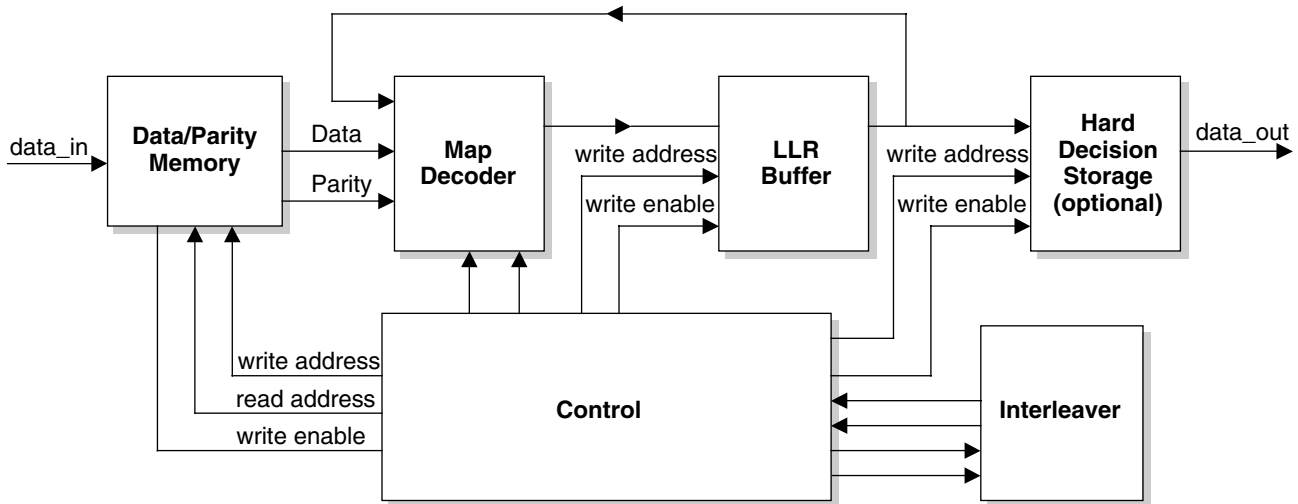
Turbo coding is an advanced error correction technique widely used in the communications industry. Turbo encoders and decoders are key elements in today's communication systems to achieve the best possible data reception with the fewest possible errors. The basis of turbo coding is to introduce redundancy in the data to be transmitted through a channel. The redundant data helps to recover original data from the received data. In data transmission, turbo coding helps achieve near Shannon limit performance.

Lattice provides a Turbo Decoder IP core that is both flexible and compliant with two different standards, 3GPP and CCSDS. 3GPP is widely used in WCDMA and MC-CDMA applications while CCSDS is most commonly found in telemetry and space communications.

Lattice also supplies users with a Turbo Encoder core providing users a complete state of the art error correction solution.

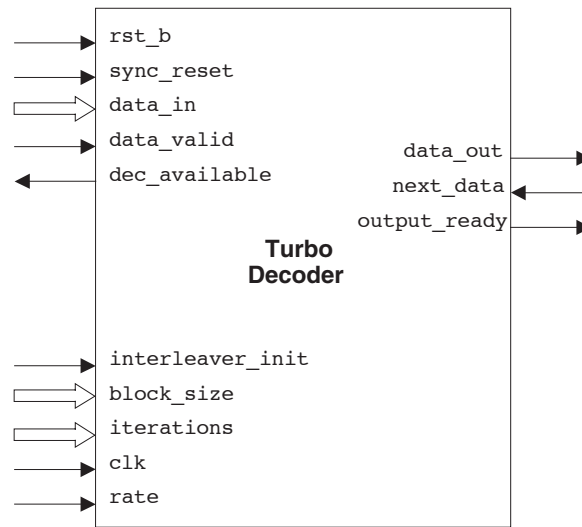
Functional Block Diagram

Figure 1. Turbo Decoder Conceptual Functional Block Diagram



Block Diagram

Figure 2. Turbo Decoder I/O Block Diagram



Note: Additional I/O signals are required if either an external memory or double buffer is selected. Please refer to the *Turbo Decoder User's Guide* for a detailed description of those signals.

MAP Algorithm

Turbo decoding is based on the principle of comparing the probability of a received soft input data being a '1' and '0'. The Lattice Turbo Decoder uses a decoding scheme called the MAP (Maximum A Posteriori Probability) algorithm. The algorithm determines the probability of whether each received data symbol is a '1' as well as a '0'. This is done with the help of the data, parity symbols, and the decoder knowledge of the encoder trellis. A trellis is a form of a state transition table, of the encoder input/output. Based on the data and parity information, the MAP decoder computes the probability of the encoder being in a particular state. Depending on the soft data, parity value and the weight from the previous state, the probability that the data is a '1' or '0' can be computed. The MAP decoder computes the weight for each data symbol in a given block for both the forward and reverse directions. This results in the computation of forward and reverse metrics. Using these two values the probabilities are computed. After the probabilities are determined they are compared and a decision is made. The Turbo Decoder IP core uses the logarithm of the probability to reduce computation; this is known as Log Likelihood Ratio (LLR). The computation of the probabilities is done iteratively to obtain a reliable result. Once the result is considered reliable, one can make a final decision as to whether the data symbol is a '1' or '0'. The Lattice Turbo Decoder can implement both the Log-Map and Max-Log-Map algorithm. The Log-Map algorithm gives a slightly better performance than the Max-Log-Map but utilizes more resources and runs at a slower frequency.

The Log Likelihood Ratio is the probability that the received data bit is a '0' divided by the probability that the received data bit is a '1'.

$$L(D) = \log \frac{P(D=0)}{P(D=1)}$$

Thus, taking the logarithm we will have a positive value if $P(D=1) > P(D=0)$, and negative value for the opposite. A positive value means the data value is a '1', otherwise a '0'. For one complete cycle of iteration, one needs to compute the LLR using parity for non-interleaved as well as interleaved data.

Functional Description

The Turbo Decoder consists of four main components, the control module, decoder and interleaver, output data handshaking, and memory buffers.

Control Module

The control module takes care of the interface, pipelining and handshake communication between various blocks and I/O pins. Data and parity are read serially into the memory and it is assumed that the data is received in the same order as it was transmitted from the encoder. Signal `interleaver_init` initializes the `block_size` by specifying the size of the block to be input to the decoder. Input data can be given only when `dec_available` is asserted. Input data has to be qualified with `data_valid` to be accepted by the core.

Decoder and Interleaver

Once the data is input into the decoder, the decoder starts computing the LLR of each data symbol. The LLR is computed for the block sequence twice, once using the non-interleaved data and the corresponding parity and then using the interleaved data and the corresponding parity. One round of this computation is called an iteration. Each iteration is divided into two sections, an ODD window and an EVEN window. The LLR for systematic parity is computed during the ODD window and the LLR for interleaved data is computed during the EVEN window. When both ODD and EVEN window computations are done, one iteration is complete. The user can set the number of iterations for each block on the `iterations` pin. During the second half of the iteration, EVEN window, the LLR computed in the first half is improved upon by using previous computations. Every window makes use of LLR information computed in the previous window and tries to improve on the estimate of LLR. The interleaver is used in the second half of iteration to generate the interleaved address. This address is used to address the data memory and read the interleaved data for the second half of the iteration. It is also used to address the LLR memory unit and get the previously computed LLR information. At the end of one iteration, the decoder has a set of LLR for each input data. The second iteration starts with again using the non-interleaved parity bits and data and the previously computed LLR to get a new estimate of the LLR for the data. Once the decoder completes the number of iterations required to be done, the LLR memory buffer has the final LLR values. The sign of the LLR values determines whether the data is a '1' or '0'. A positive sign means the data value is a '1', otherwise, it is a '0'.

Output Data Handshaking

When the decoder is ready to output data, signal `output_ready` is asserted high after the decoder has completed the specified number of iterations. The user can then assert signal `next_data` to read the decoded data, which then allows data to be output on `data_out`.

An optional synchronous reset signal, `sync_reset` is available to reinitialize the Turbo Decoder in the middle of a block processing. The current block being processed will be completely discarded during this reset. This can be done at any point of time during the operation of the decoder.

Memory Buffer

The memory buffering for this IP splits into four sections. These sections are described in detail below.

Input Data/Parity Memory

The Turbo Decoder core requires a large amount of memory to store the input data block. Since data memory requirements are large, external memory is recommended so that on-chip memory can be used for other purposes. An external memory interface is provided in the Turbo Decoder IP core. A single or double buffer memory mode may be selected depending on the available external memory at hand. Double buffer memory allows one block of data to be processed while another block is written and read. Double buffer memory delivers better performance than the single buffer selection by minimizing delay between the processing of each block.

Internal Memory

Some internal memory is required to implement the interleaver and other necessary functions of the Turbo Decoder. Lattice's Turbo Decoder requires a small amount of memory for internal purposes. For example, the 3GPP configuration uses approximately 4.6Kb spread between four memory blocks.

LLR Memory

After the Turbo Decoder completes the required number of iterations, the LLR memory buffer stores the final LLR values. The size of the LLR memory buffer is dependant on configuration and block size.

Hard Decision Storage Memory

Lattice's Turbo Decoder IP core offers optional hard decision storage. When LLR memory is used as an output buffer, the decoder cannot go on to process the next block of data until the current LLR values of the previous block are completely read out. This results in an extra processing delay of B cycles (B = `block_size`). To minimize delay, output data after hard decision can be stored in a separate memory to allow the decoder to operate on a new data block if memory can be spared.

Parameter Descriptions

The Turbo Decoder IP supports many parameters to enhance its flexibility. Table 1 lists the parameters available in this IP.

Table 1. Parameter Descriptions

Parameter/Notation	Description	3GPP	3GPP2	CCSDS
THREE_GPP, THREE_GPP2OR CCSDS	Decoder type	THREE_GPP	THREE_GPP2	CCSDS
NUM_STATES/STATENUM	Number of states	8	8	16
RATE	Rate	Range: 2-6 Variable	Range: 2-6 Variable	Range: 2-6 Variable
MAX_BLK_SIZE/ MAXBLKSIZE	Maximum block size	Default: 5114	Default: 20730	Default: 8920
INPUT_DATA_WIDTH/ INPWIDTH	Input data width	Range: 3-6 Default: 6	Range: 3-6 Default: 6	Range: 3-6 Default: 6
WINDOW_SIZE/WINSIZE	Window size	Range: 16 or 32 Default: 16	Range: 16 or 32 Default: 16	Range: 16 or 32 Default: 16
BUFFER_EXTERNAL/ EXTMEM	External memory buffer	Range: Yes or No Default: Yes	Range: Yes or No Default: Yes	Range: Yes or No Default: Yes
DOUBLE_BUFFER/ DBLBUFF	Double memory buffer	Range: Yes or No Default: Yes	Range: Yes or No Default: Yes	Range: Yes or No Default: Yes
NUM_PARITY	Number of parities	Range: 1-3 Default: 1	Range: 1-3 Default: 2	Range: 1-3 Default: 3
DEC_ALGORITHM/ALGO	Decoder algorithm	1=LOGMAP, 0=MAXLOGMAP Default = 0	1=LOGMAP, 0=MAXLOGMAP Default = 0	1=LOGMAP, 0=MAXLOGMAP Default = 0
BM_SOFTBITS/NBM	Branch metric softbits	Range: 8-12 Default =10	Range: 8-12 Default =10	Range: 8-12 Default =10
PM_SOFTBITS/NPM	Path metric softbits	Range: 8-12 Default =10	Range: 8-12 Default =10	Range: 8-12 Default =10
LR_SOFTBITS/NLR	Log likelihood ratio softbits	Range: 8-12 Default =10	Range: 8-12 Default =10	Range: 8-12 Default =10
EXTMEM_PIPELINE/ EXTPIPELINE	External memory pipeline	Range: PIPE1-PIPE6 Default = PIPE1	Range: PIPE1-PIPE6 Default = PIPE1	Range: PIPE1-PIPE6 Default = PIPE1
HARD_DECISION_STORAGE/ HARDSTORAGE	Hard decision storage	Range: Yes or No Default: Yes	Range: Yes or No Default: Yes	Range: Yes or No Default: Yes

Signal Descriptions

Table 2 lists the various input and output signals for the Turbo Decoder IP.

Table 2. Turbo Decoder Signal Definitions

Port Name	I/O Type	Width	Signal Description
clk	Input	1	System Clock
rst_b	Input	1	Active Low Asynchronous Reset
sync_rst	Input	1	Synchronous Reset (optional signal)
data_in	Input	3-6	Data Input (soft encoded data from demodulator)
data_out	Output	1	Data Output
iterations	Input	4	Input bus that can be set to a value from 0-15 to set the number of iterations needed to decode the data being input.
interleaver_init	Input	1	Interleaver initialization. <code>block_size</code> on input pins can be changed and accepted when this signal is asserted.
data_valid	Input	1	Enables the encoder to read the data at <code>data_in</code> when asserted.
dec_available	Output	1	Active High signal. Indicates that decoder is available to accept data. Note, this signal is de-asserted one data symbol before buffer is full.
block_size	Input	11-15	Block size up to 2^{15} bits can be set depending on the configuration selected and <code>MAX_BLK_SIZE</code> allowed for each configuration.
next_data	Input	1	Asserted to indicate successful reading of encoded data from <code>data_out</code> .
output_ready	Output	1	When asserted encoded data is ready and available at <code>data_out</code> .
rate	Input	1 or 2	The rate of data output from the decoder.

Custom Core Configurations

For Turbo Decoder core configurations that are not available in the Evaluation Package, please contact your Lattice sales office to request a custom configuration.

Related Information

For more information regarding core usage and design verification, refer to the *Turbo Decoder User's Guide*, available on the Lattice web site at www.latticesemi.com.

Appendix for ORCA Series 4 FPGAs

Table 3 details the performance and utilization information for the configurations provided in the ORCA Series 4 Turbo Decoder Evaluation Package.

Table 3. Performance and Utilization¹

Parameter File	Mode	Parameters	ORCA 4 PFUs	LUTs	Registers	PIO	EBR	f _{MAX}
turbo_deco_o4_1_001.lpc	3GPP	See Table 2	1235	3750	3569	184	N/A	46 MHz
turbo_deco_o4_1_003.lpc	CCSDS	See Table 2	1674	3292	4807	191	N/A	36 MHz

1. Performance and utilization characteristics are generated targeting an OR4E06-2BA352 in ispLEVER v.3.0 software.

Supplied Netlist Configurations

The Ordering Part Number (OPN) for all configurations of the Turbo Decoder core targeting ORCA Series 4 devices is TURBO-DECO-O4-N1. Table 3 lists the netlists that are available in Evaluation Package, which can be downloaded from the Lattice web site at www.latticesemi.com.