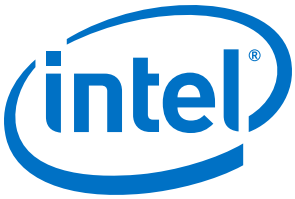


Erasure Decoder Reference Design

Updated for Intel® Quartus® Prime Design Suite: **17.0**

Subscribe
Send Feedback

UG-20072 | 2017.05.02
Latest document on the web: **PDF** | **HTML**

Contents

1. About the Erasure Decoder Reference Design

The Erasure Decoder is a particular type of Reed-Solomon decoder that uses a non-binary, cyclic, linear block error correction code.

In a Reed-Solomon decoder with erasure decoding capability, the number of errors (*E*) and erasures (*E'*) that you can correct is:

$n - k = 2E + E'$

Where *n* is the block length and *k* is the message length (*n*-*k* equals the number of parity symbols).

The Erasure Decoder only considers erasures, so the correction capability can reach the maximum given by *n*-*k*. The decoder receives as input the erasure locations, typically provided by the demodulator within the coding system, which can indicate certain received code symbols as unreliable. The design should not not exceed the erasure correction capability. The design treats symbols that it indicates as erasure as zero value.

Features

- Targets Stratix® 10 devices
- Corrects erasures
- Parallel operation
- Flow control

2. Erasure Decoder Functional Description

The Erasure Decoder does not correct errors, only erasures. It avoids the complexity of finding error locations, which Reed-Solomon decoding requires.

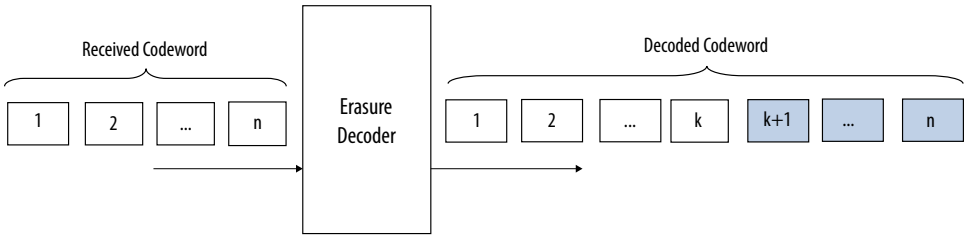
The design algorithm and architecture is different than a Reed-Solomon decoder. Erasure decoding is a form of encoding. It tries to fill up the input with $p=n-k$ symbols to form a valid codeword, by fulfilling the parity equations. The parity matrix and the generator matrix define the parity equations.

The design only works with small Reed-Solomon codes, such as RS(14,10), RS(16,12), RS(12,8) or RS(10,6). For a small number of parity symbols ($p < k$) use this design; for a large number of parity symbols ($p > k-p$), you should use a generator matrix.

The erasure pattern (represented by the n-bits wide `in_era` input) addresses the ROM where the design stores parity submatrices. The design only has $\binom{n}{p} = \frac{n!}{k!(n-k)!}$ possible erasure patterns. Therefore, the design uses an address compression module. The design encodes the address with the number of addresses that are smaller than the address and have exactly p bits set.

The Erasure Decoder receives at its input any rate of incoming symbols, up to the total block length n per cycle for the maximum throughput. You can configure parallelism and the number of channels, so that the design multiplies the incoming symbols by the number of channels in parallel that correspond to different codewords arriving at the same time. The erasure decoder produces the full decoded codeword, including check symbols, in one cycle (several codewords for several channels).

Figure 1. Codeword Decoding



An input buffer allows you to have the number of parallel symbols per channel fewer than the total block length (n). Intel recommends you use the input bandwidth, unless the parallelism depends on your interface requirements.

3. Erasure Decoder IP Core Parameters

Table 1. Parameters

Parameter	Legal Values	Default Value	Description
Number of channels	1 to 16	1	The number of input channels (*C*) to process.
Number of bits per symbol	3 to 12	4	The number of bits per symbol (*M*).
Number of symbols per codeword	1 to *2M*–1	14	The total number of symbols per codeword (*N*).
Number of check symbols per codeword	1 to *N*–1	4	The number of check symbols per codeword (*R*).
Number of parallel symbols per channel	1 to *N*	14	The number of symbols that arrive in parallel at the input for each codeword (*PAR*)
Field Polynomial	Any valid polynomial	19	Specifies the primitive polynomial defining the Galois field.

4. Erasure Decoder Interfaces and Signals

The Avalon-ST interface supports backpressure, which is a flow control mechanism, where a sink can indicate to a source to stop sending data.

The ready latency on the Avalon-ST input interface is 0; the number of symbols per beat is fixed to 1.

The clock and reset interfaces drive or receive the clock and reset signal to synchronize the Avalon-ST interfaces.

4.1. Avalon-ST Interfaces in DSP IP Cores

Avalon-ST interfaces define a standard, flexible, and modular protocol for data transfers from a source interface to a sink interface.

The input interface is an Avalon-ST sink and the output interface is an Avalon-ST source. The Avalon-ST interface supports packet transfers with packets interleaved across multiple channels.

Avalon-ST interface signals can describe traditional streaming interfaces supporting a single stream of data without knowledge of channels or packet boundaries. Such interfaces typically contain data, ready, and valid signals. Avalon-ST interfaces can also support more complex protocols for burst and packet transfers with packets interleaved across multiple channels. The Avalon-ST interface inherently synchronizes multichannel designs, which allows you to achieve efficient, time-multiplexed implementations without having to implement complex control logic.

Avalon-ST interfaces support backpressure, which is a flow control mechanism where a sink can signal to a source to stop sending data. The sink typically uses backpressure to stop the flow of data when its FIFO buffers are full or when it has congestion on its output.

Related Information

Avalon Interface Specifications

4.2. Erasure Decoder IP Core Signals

Table 2. Clock and Reset Signals

Name	Avalon-ST Type	Direction	Description
clk_clk	clk	Input	The main system clock. The whole IP core operates on the rising edge of clk_clk .
reset_reset_n	reset_n	Input	An active low signal that resets the entire system when asserted. You can assert this signal asynchronously. However, you must deassert it synchronous to the clk_clk signal. When the IP core recovers from reset, ensure that the data it receives is a complete packet.

Table 3. Avalon-ST Input and Output Interface Signals

Name	Avalon-ST Type	Direction	Description
in_ready	ready	Output	Data transfer ready signal to indicate that the sink is ready to accept data. The sink interface drives the in_ready signal to control the flow of data across the interface. The sink interface captures the data interface signals on the current clk rising edge.
in_valid	valid	Input	Data valid signal to indicate the validity of the data signals. When you assert the in_valid signal, the Avalon-ST data interface signals are valid. When you deassert the in_valid signal, the Avalon-ST data interface signals are invalid and must be disregarded. You can assert the in_valid signal whenever data is available. However, the sink only captures the data from the source when the IP core asserts the in_ready signal.
in_data[]	data	Input	Data input containing the codeword symbols. Valid only when in_valid is asserted. The in_data signal is a vector containing *C* x *PAR* symbols. If *PAR* < *N*, the codeword of each channel arrives over several cycles.
in_era	data	Input	Data input that indicates which symbols are erasures. Valid only when in_valid is asserted. It is a vector containing *C* x *PAR* bits.
out_ready	ready	Input	Data transfer ready signal to indicate that the downstream module is ready to accept data. The source provides new data (if available) when you assert the out_ready signal and stops providing new data when you deassert the out_ready signal.
out_valid	valid	Output	Data valid signal. The IP core asserts the out_valid signal high, whenever a valid output is on out_data.
out_data	data	Output	Contains decoded output when the IP core asserts the out_valid signal. The corrected symbols are in the same order that they are entered. It is a vector containing *C* x *N* symbols.
out_error	error	Output	Indicates non-correctable codeword.

An asserted in_valid signal indicates valid data.

Each codeword can arrive over several cycles, depending on the parallelism parameter. The design tracks the structure of the input, so it requires no packet boundaries on the interface. The design's **Number of channels in parallel** increases throughput by replicating the functional units for all the concurrent channels. This design does not use Avalon-ST interface multiple channel support.

When the decoder asserts the `out_valid` signal, it provides valid data on `out_data`. It outputs *C* codewords per cycle, where *C* is the number of channels in parallel. The IP core asserts `out_error` signal when it receives a non-correctable codeword, i.e.: when the IP core exceeds erasure correction capability.