Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Reed-Solomon Decoder

User's Guide

October 2005
ipug07_04.0

Introduction

Lattice's Reed-Solomon Decoder core provides an ideal solution that meets the needs of today's forward error correction applications. The Reed-Solomon Decoder core provides a customizable solution allowing forward error correction of data in many communication applications. This core allows designers to focus on the application rather than the Reed-Solomon Decoder, resulting in a faster time to market.

Reed-Solomon codes are widely used in various applications for forward error correction and detection. Lattice's Reed-Solomon Decoder core is a fully synchronous core developed in conjunction with Lattice's Reed-Solomon Encoder core to provide a complimentary pair. For more information on these and other Lattice products, refer to the Lattice web site at www.latticesemi.com.

This user's guide explains the functionality of the Reed-Solomon Decoder core and how it can be implemented to provide decoding on any data transmission. It also explains how to achieve the maximum level of performance.

The Reed-Solomon Decoder core comes with the documentation and files listed below:

- Data sheet
- Lattice gate level netlist
- RTL simulation model
- Core instantiation template

Features

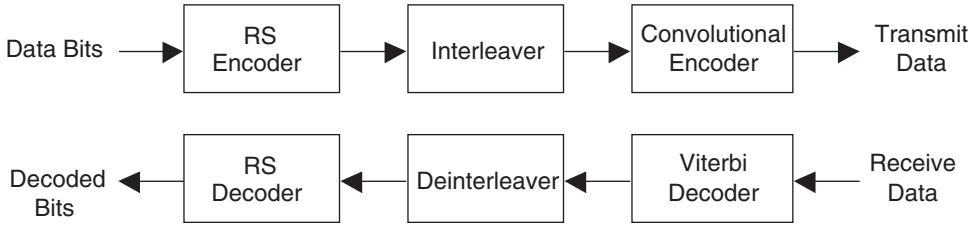
- Forward Error Correction (FEC) for communication and common applications
- Selectable Reed-Solomon standards:
  - CCSDS (255,223)
  - ATSC (207,187)
  - DVB (204,188)
  - OC192 (255,239)
- Shortened codes supported
- Fully synchronous design
- Error/erasures supported
- Supports symbol widths from 3 to 12 bits, corresponding to GF(8) and GF(4096) respectively
- User-defined and default field and generator polynomials supported
- Generates failure flags and measurement information

General Description

Reed-Solomon codes are used to perform Forward Error Correction. FEC encoders introduce redundancy in data before it is transmitted. The redundant data (check symbols) are transmitted along with the original data through the channel. A Reed-Solomon decoder at the receiver is used to recover any corrupted data This type of error correction is widely used in data communications applications such as hard disk and media storage (CD) systems, Digital Video Broadcast (DVB) and Optical Carriers (e.g. OC-192).

The codes are represented by the format RS(n,k) where n is the total number of s-bit wide symbols, and k is the number of s-bit wide information (data) symbols in a codeword. The Reed-Solomon Decoder performs detection and correction of encoded data available at the receiver after demodulation. The RS encoded data is then processed to determine whether any errors have occurred during transmission. Once the number of errors is determined, the decoder decides if they are within the range of correction. After determining this, the decoder corrects the errors in the received data. A typical application of space signal processing is shown in Figure 1.

Figure 1. Application of Reed-Solomon code in a Space Communication System



A Reed-Solomon Decoder can correct errors and erasures. The maximum number of correctable erroneous symbols in a codeword is t = (n-k)/2, and the maximum number of correctable erasures in a codeword is μ = n-k.

Reed-Solomon codes are based on finite field arithmetic, also known as Galois Field. The size of the field is determined by the width `s` of the information symbol, and the field has 2^s elements. When n is less than the maximum value of 2^s -1, the code is called a shortened code.

Reed-Solomon codes are characterized by two polynomials: the field polynomial and the generator polynomial. The field polynomial defines the Galois Field to which the information and check symbols belong. The generator polynomial determines the check symbol generation, and is a prime polynomial for all codewords (i.e. all codewords are exactly divisible by the generator polynomial). Both the field and generator polynomials are user configurable.

Field Polynomial

The field polynomial can be specified as any prime polynomial up to 2^{s+1} - 1. The field polynomial is defined by its decimal value (f). The decimal value of a field polynomial is determined by setting x = 2 in the polynomial and calculating the result. For example, the polynomial $x^2 + x + 1$ in decimal value is $2^2 + 2 + 1 = 7$.

Generator Polynomial

The generator polynomial determines the value of the check symbols. The starting root (`gstart`) and root spacing (`rootspace`) variables define the generator polynomial. The general form of the generator polynomial is:

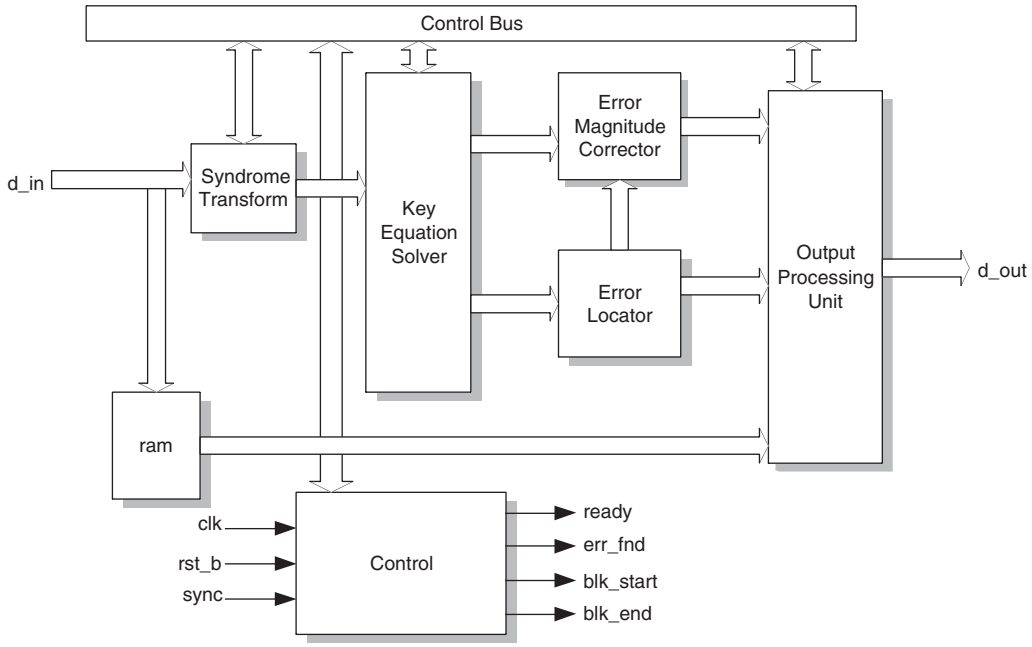
$$g(x) = \prod_{i=0}^{n-k-1} (x - \alpha^{rootspace * (gstart + i)}) \quad (1)$$

Shortened Codes

A shortened code has a lesser number of total symbols than the maximum possible for the given symbol width. An example of a shortened code would be, RS(204,188) where s = 8.

Block Diagram

Figure 2. Reed-Solomon Decoder Block Diagram



Functional Description

The data received by the RS Decoder is Reed-Solomon encoded data. This data is a representation of a polynomial in a Galois Field. If there are no errors in the received data, the data polynomial will evaluate to zero at the roots of the generator polynomial. This result is obtained because the roots of the generator polynomial and received data polynomial are the same when no errors are present. If the received data has been corrupted during the transmission, the polynomial will not evaluate to zero. The RS Decoder can construct the syndrome polynomial by evaluating the received polynomial at all the roots of the generator polynomial. Once the syndrome polynomial has been constructed, it can be used to solve the Error Locator polynomial and Error Evaluator polynomial. Using these two polynomials, the decoder can find the error locations and magnitudes. Finally, the decoder can correct the errors in the received data, provided the errors are in the range of possible correction (determined by the level of encoding that has been performed).

If there are errors in the received codeword, it can be expressed as follows:

$r(x) = c(x) + e(x)$

where:

- c(x) is the Transmitted codeword.
- r(x) is the Received codeword.
- e(x) is the Error polynomial.

The syndrome polynomial S(x) is obtained by evaluating the received word at each root of the generator polynomial. The Error Locator polynomial $\Lambda(x)$ is orthogonal to the syndrome polynomial in the Galois field. This can be represented as:

$S(x)\ \Lambda(x) = \Omega(x) \bmod x^{2t}$

where:

- $\Omega(x)$ is the Error Evaluator polynomial.
- 2t is the number of check symbols introduced by the decoder.

The following sections describe the function of each block of the RS Decoder.

Syndrome Transform

The Syndrome Transform (also called Syndrome Generation) block evaluates the received codeword of the generator polynomial. If the received data contains an error, the syndrome polynomial generated will be non-zero. If the received data has no error, the syndrome polynomial is zero, and the data is passed out of the decoder without any error correction.

Key Equation Solver

This is the heart of the Reed-Solomon Decoder. This block generates the Error Locator polynomial $\Lambda(x)$ (also known as the "Key Equation" as it is the key to solve the decoding problem). After the Error Locator polynomial has been found, it is used to determine the Error Evaluator polynomial $\Omega(x)$.

Error Locator

This block is implemented using the Chien-search method. Essentially, this method evaluates the Error Locator polynomial at all the elements in the Galois Field. The Error Locator polynomial evaluates to zero at its roots. The Chien-search takes m cycles, where m is the number of elements in the Galois Field. After m cycles, all roots have been determined. If the roots are determined before the m cycles are over, the search is terminated early.

Error Magnitude Corrector

Once the location of the error has been determined, the Error Magnitude Corrector evaluates the evaluator polynomial at the root. It uses the result to calculate the value of the error at the given location. Once this has been determined, the value is added to the received word to recover the corrected data. The addition occurs after the Error Locator evaluates to zero.

Control Circuit

The control circuit handles the interface, pipelining and handshaking communication between the various blocks and the I/O pins. The control circuit moves the data without processing it through the decoder when no error is detected. Similarly, when the number of errors exceeds the maximum range of correction, the control circuit stops all data processing activities. The control circuit interacts with the other blocks to generate the status signals like `fail`, `err_fnd`, `err_cnt`, `ers_cnt` and other handshaking signals. Once the block has been processed, the control circuit sends out the ready signal to the output to start the processing of the next data.

Parameter Descriptions

Table 1 lists the parameters used for configuring the RS Decoder. The values of these parameters are set prior to synthesis.

Table 1. RS Decoder Parameters

Parameter	Description	Value
n	The total number of symbols in the code word.	3 - 4095
wsymb	Width of input data d_in in bits. Ranges from 3 to 12 bits. The size of d_out equals the size of d_in.	3 - 12
k	Indicates the number of data symbols in the code.	1 - 4093
rootspace	Indicates space between the roots of the generator polynomial. The value of rootspace must satisy the following equation: GCD(rootspace, 2^{wsymb}-1) = 1. GCD is Greatest Common Divisor.	1 - 65535
gstart	Offset value of the generator polynomial. The starting value for the first root of the generator polynomial is calculated as rootspace * gstart.	0 - 65535
fpoly	Field polynomial. This must be a valid finite field polynomial in the Galois field GF(2^n). The default value is used if fpoly is not specified. Table 2 defines default field polynomials for different symbol widths.	11 - 8191
coretype	Type of the RS Decoder core. It can be one of the following optimized types: CCSDS, ATSC, DVB or OC192 or it can be a custom configuration. The details of the different core types are listed below: • Hard coded parameters for CCSDS mode will be used to derive the decoder if this option is selected. coretype = ccsds, wsymb = 8, n = 255, k = 223, gstart = 112, rootspace = 11, fpoly = 391, numerasrs = 0 • Hard coded parameters for ATSC mode will be used to derive the decoder if this option is selected. coretype = atsc, wsymb = 8, n = 207, k = 187, gstart = 0, rootspace = 1, fpoly = 285, numerasrs = 0 • Hard coded parameters for DVB mode will be used to derive the decoder if this option is selected. coretype = dvb, wsymb = 8, n = 204, k = 188, gstart = 0, rootspace = 1, fpoly = 285, numerasrs = 0 • Hard coded parameters for OC192 mode will be used to derive the decoder if this option is selected. coretype = oc192, wsymb = 8, n = 255, k = 239, gstart = 0, rootspace = 1, fpoly = 285, numerasrs = 0 • When the parameter coretype is defined as custom, the parameter values can be individually selected by the user in the IPexpress™ tool.	OC192, CCSDS, ATSC, DVB or Custom
numerasrs	Number of erasures.	0 - 4088
sync	Handshake for input block data. This parameter is used to configure the functionality of the sync port.	Pulse or DataEn

Table 2. Default Field Polynomials

Symbol Data Width	Default Field Polynomial	Decimal Value
3	$x^3 + x + 1$	11
4	$x^4 + x + 1$	19
5	$x^5 + x^2 + 1$	37
6	$x^6 + x + 1$	67
7	$x^7 + x^3 + 1$	137
8	$x^8 + x^4 + x^3 + x^2 + 1$	285
9	$x^9 + x^4 + 1$	529
10	$x^{10} + x^3 + 1$	1033
11	$x^{11} + x^2 + 1$	2053
12	$x^{12} + x^6 + x^4 + x + 1$	4179

Signal Descriptions

Table 3. RS Decoder Signal Descriptions

Port Name	I/O	Active State	Signal Description
Required Signals			
rst_b	Input	Low	Asynchronous reset. This resets all the flip-flops in the core and initializes the decoder. It has the highest priority.
sync	Input	High	Used to indicate the data boundaries for the block. It has two functions that can be selected as a parameter: (1) Pulse is high for first data and low for the rest. (2) DataEn is high for data symbols and low for the check symbols.
clk	Input	N/A	System clock.
d_in[wsymb-1:0]	Input	N/A	Input data bus. The bus width is set by the symbol width parameter. Every symbol is sampled at the rising edge of the clock only after the sync pin has been asserted. A copy of each symbol can be passed directly (without processing) to d_del, if it exists.
blk_start	Output	High	Asserted for one clock cycle to indicate the first decoded symbol on the output d_out.
blk_end	Output	High	Asserted for one clock cycle to indicate the last decoded symbol on the output d_out.
err_fnd	Output	High	Error Found Indicator. Asserted through the duration of the output data block when the block has at least one symbol in error. It is low otherwise.
ready	Output	High	Asserted whenever the decoder is ready to sample input data from the next block. It is de-asserted to prevent any input sampling when the decoder is still processing the previous block.
d_out[wsymb-1:0]	Output	N/A	Output data bus. Corrected data is presented on d_out one symbol at a time after processing. If the decoding has failed, a copy of the original input data block will be presented at d_out, and the fail pin will be asserted, if it exists.
Optional Signals (for custom core types, these optional signals can be selected or deselected in the IPexpress tool)			
ce	Input	High	Clock Enable. While this is de-asserted, the decoder will ignore all other synchronous inputs and maintain its current state.
sr	Input	High	Synchronous reset. Asserted for at least one symbol duration in order to re-initialize the decoder state. Input data symbols sampled before sr is asserted are given unprocessed at the output.
ers	Input	High	Erasure. Asserted to indicate the input data symbol at the d_in pin is erased.
fail	Output	High	Fail Indicator. Asserted through the duration of output data block to indicate that the block has more errors than the decoder can correct.
d_del[wsymb-1:0]	Output	N/A	Uncorrected Data Output. A delayed copy of the input data block. Data is presented on d_del concurrently with the decoded block on d_out.
err_cnt[err_width-1:0]	Output	N/A	Error Counter. Provides the number of corrected errors in the most recent output block. The bus width err_width is equal to the number of bits required to represent the maximum possible number of correctable errors as given in the following equation: err_width = ciel($\log_2$((n-k+1)/2)) when the parameter numerasrs = 0 err_width = ciel($\log_2$(n-k+1)) when the parameter numerasrs > 0 The operator ciel() stands for the next higher integer.
ers_cnt[ers_width-1:0]	Output	N/A	Erasure Counter. Provides a count of the number of erasures fed into the decoder in the most recent input data block. The bus width ers_width is equal to the number of bits required to represent the maximum possible number of correctable erasures as given in the following equation: ers_width = ciel($\log_2$(n-k+1)) The operator ciel() stands for the next higher integer.

Timing Diagrams

Every symbol of the RS Decoder is sampled at the rising edge of the clock when `sync` pin is asserted. There are two behaviors of the sync pin:

- **Start Pulse.** The `sync` pin is asserted for one symbol duration to indicate the start of the input data block.
- **Data Symbol Enable.** The `sync` pin is asserted for k-symbol duration from the start of the input data block, during which the information symbols are sampled in.

Figure 3 shows the I/O signal's status after rst_b is asserted at the beginning of the block. During the decoding process, the `ready` and `sync` signals are asserted HIGH to indicate the start of the decoding process. In order to output the decoded data, `blk_start` is asserted for one clock cycle to indicate that the first decoded symbol is currently at the `d_out` pin. The pin `blk_end` is asserted for one clock cycle to indicate the last decoded symbol is currently at the `d_out` pin.

Figure 3. I/O Signal Status of RS Decoder

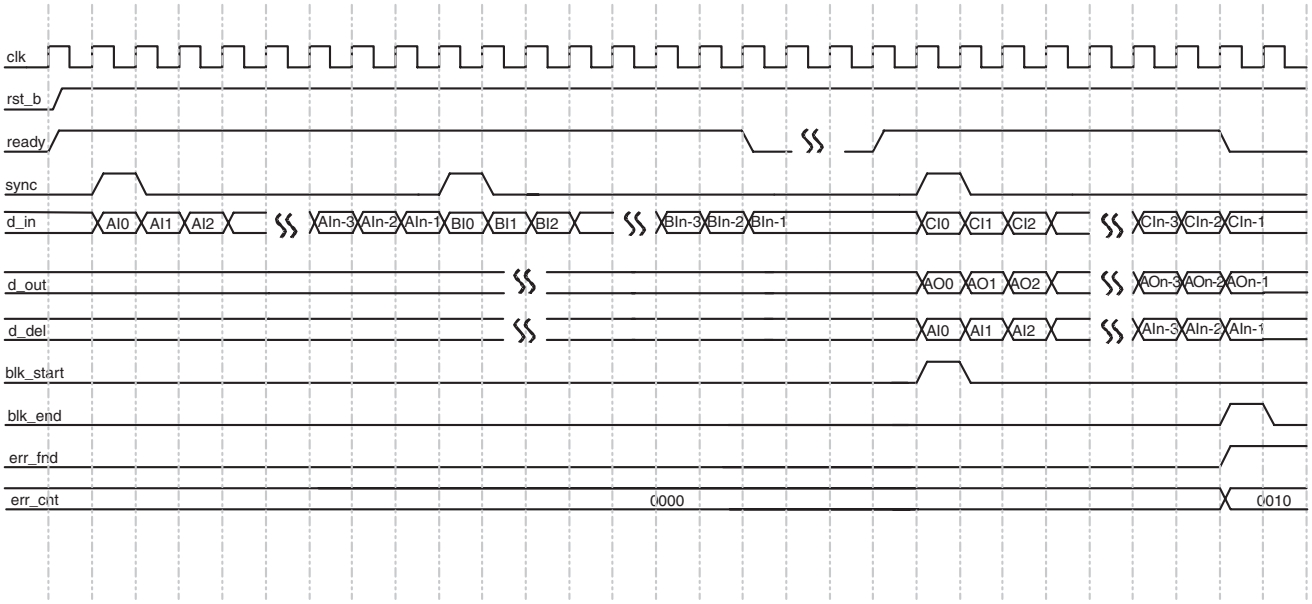


Figure 4 illustrates the output status when `sr` is asserted. When `sr` is asserted during the decoding process, it re-initializes the decoder state. It forces the data at `d_in` to be output to `d_out` without correction.

Figure 4. Effect of Synchronous Reset on the Output Data from the Decoder

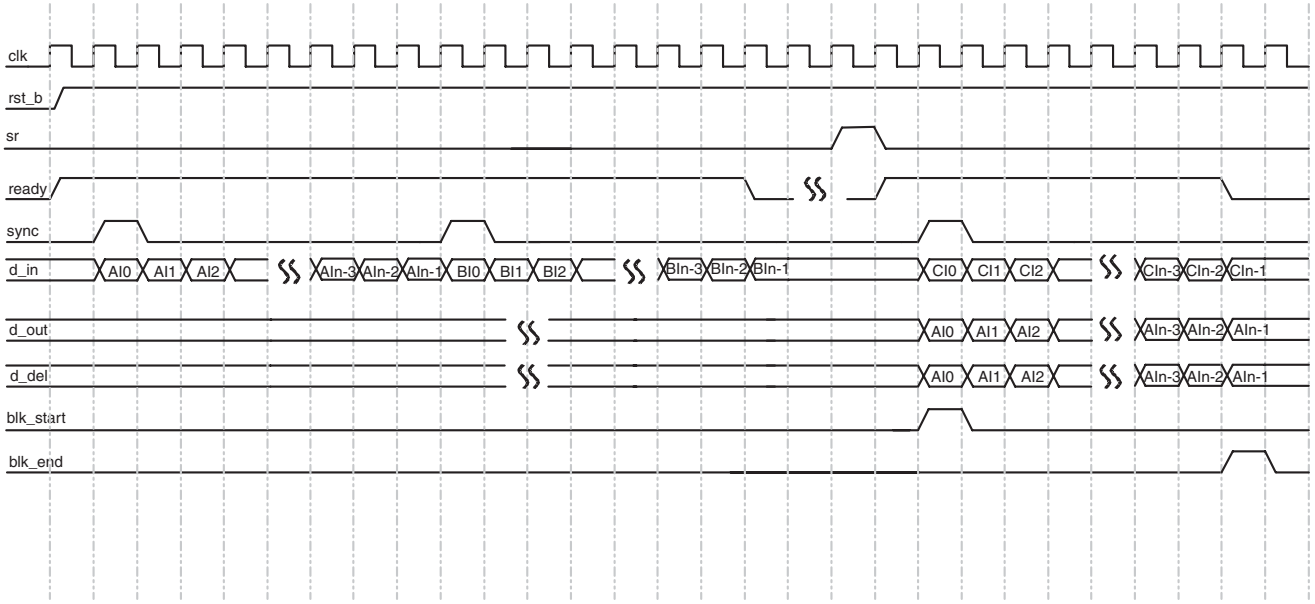
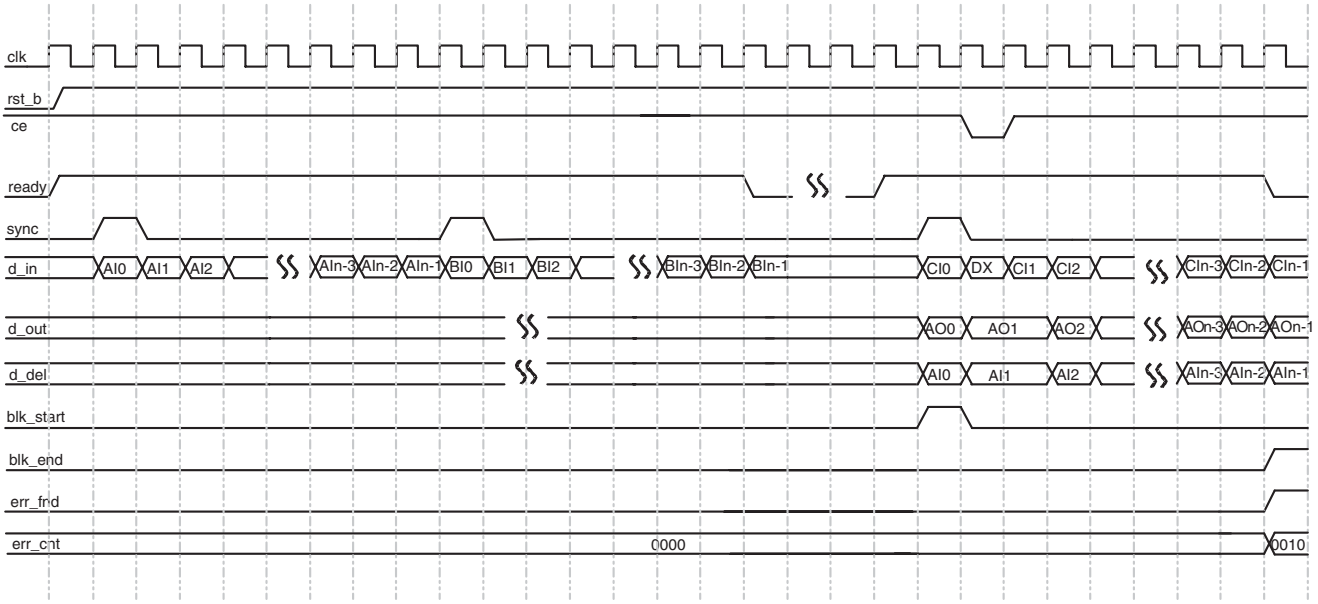


Figure 5 illustrates the effect of clock enable (`ce`) on the output data from RS Decoder. The decoder ignores all other synchronous inputs and remains in its current state when `ce` is de-asserted. When `ce` is asserted, the decoder goes back to the normal decoding process. In the figure, the data DX at d_in (that occurs during `ce` going low) is not recognized by the decoder.

Figure 5. Effect of Clock Enable on the Output Data from Decoder



References

- I. S. Reed, M. T. Shih, and T. K. Truong, "VLSI design of inverse-free Berlekamp-Massey algorithm," Proc. IEEE, Part E, vol. 138, pp. 295-298, September 1991.
- S. Kwon and H. Shin, "An area-efficient VLSI architecture of a Reed-Solomon decoder/encoder for digital VCRs," IEEE Trans. Consumer Electronics, pp. 1019-1027, Nov. 1997.
- Reed-Solomon Decoder Data Sheet, Lattice Semiconductor Corporation, 2002.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Appendix for ORCA® Series 4 FPGAs or FPSCs

Table 4. Performance and Resource Utilization[1]

Parameter File	Mode	ORCA 4 PFUs[2]	LUTs	Registers	I/Os[3]	sysMEM EBRs	f_{MAX} (MHz)
reeds_deco_o4_1_001.lpc	OC192	321	1595	918	35	2	83
reeds_deco_o4_1_002.lpc	CCSDS	621	2627	1565	36	2	76
reeds_deco_o4_1_003.lpc	DVB	330	1764	911	35	2	85
reeds_deco_o4_1_004.lpc	ATSC	423	2139	1072	35	2	84

1. Performance and utilization characteristics are generated using an OR4E042BA352. When using this IP core in different density, package, speed, or grade within ORCA Series 4 family, performance may vary.
2. PFU is a standard logic block of some Lattice devices. For more information, check the data sheet of the device.
3. The I/Os for these configurations include the optional signals `d_del` and `err_cnt`. The width of the `err_cnt` signal bus is 5 for the CCSDS configuration, and 4 for the other evaluation configurations.

Supplied Netlist Configurations

The Ordering Part Number (OPN) for all configurations of this core for use with ORCA Series 4 FPGAs is REEDS-DECO-O4-N1. Table 5 lists the Lattice-specific netlists available in the Evaluation Package, which can be downloaded from the Lattice web site at www.latticesemi.com.

Table 5. Core Configurations

Number	Parameter	Configurations			
		CCSDS	DVB	ATSC	OC192
1	wsymb	8	8	8	8
2	n	255	204	207	255
3	k	223	188	187	239
4	rootspace	11	1	1	1
5	fpoly	391	285	285	285
6	gstart	112	0	0	0
7	Numerasrs	0	0	0	0
8	Coretype	ccsds	dvb	atsc	oc192

You can use the IPexpress software tool to help generate new configurations of this IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER® design tools. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.

Appendix for ispXPGA® FPGAs

Table 6. Performance and Resource Utilization[1]

Parameter File	Mode	ispXPGA PFUs[2]	LUTs	Registers	I/Os[3]	sysMEM EBRs	f_{MAX} (MHz)
reeds_deco_xp_1_001.lpc	OC192	571	2034	1033	35	2	83
reeds_deco_xp_1_002.lpc	CCSDS	1011	3660	1717	36	2	76
reeds_deco_xp_1_003.lpc	DVB	586	2089	1049	35	2	84
reeds_deco_xp_1_004.lpc	ATSC	712	2536	1286	35	2	80

1. Performance and utilization characteristics are generated using LFX500B-04F516C in Lattice's ispLEVER 3.x software. The evaluation version of this IP core only works on this specific device density, package, and speed grade.
2. PFU is a standard logic block of some Lattice devices. For more information, refer to the data sheet of the device.
3. The I/Os for these configurations include the optional signals `d_del` and `err_cnt`. The width of the `err_cnt` signal bus is 5 for the CCSDS configuration, and 4 for the other evaluation configurations.

Supplied Netlist Configurations

The Ordering Part Number (OPN) for all configurations of this core for use with ispXPGA devices is REEDS-DECO-XP-N1. Table 7 lists the Lattice-specific netlists that are available in the Evaluation Package, which can be downloaded from the Lattice web site at www.latticesemi.com.

Table 7. Core Configurations

Number	Parameter	Configurations			
		CCSDS	DVB	ATSC	OC192
1	wsymb	8	8	8	8
2	n	255	204	207	255
3	k	223	188	187	239
4	rootspace	11	1	1	1
5	fpoly	391	285	285	285
6	gstart	112	0	0	0
7	Numerasrs	0	0	0	0
8	Coretype	ccsds	dvb	atsc	oc192

You can use the IPexpress software tool to help generate new configurations of this IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.

Appendix for LatticeECP™ and LatticeEC™ FPGAs

Table 8. Performance and Resource Utilization[1]

Parameter File	Mode	SLICEs	LUTs	Registers	I/Os[2]	sysMEM EBRs	f_{MAX} (MHz)
reeds_deco_e2_1_001.lpc	OC192	812	1188	792	35	2	110
reeds_deco_e2_1_002.lpc	CCSDS	1389	1950	1419	36	2	102
reeds_deco_e2_1_003.lpc	DVB	816	1202	793	35	2	113
reeds_deco_e2_1_004.lpc	ATSC	1053	1491	991	35	2	103

1. Performance and utilization characteristics are generated using LFEC20E-5F672C in Lattice's ispLEVER v.4.1 software. When using this IP core in a different device, density, package, or speed grade, performance may vary.
2. The I/Os for these configurations include the optional signals `d_del` and `err_cnt`. The width of the `err_cnt` signal bus is 5 for the CCSDS configuration, and 4 for the other evaluation configurations.

Supplied Netlist Configurations

The Ordering Part Number (OPN) for all configurations of this core for use with LatticeECP/EC FPGAs is REEDS-DECO-E2-N1. Table 9 lists the Lattice-specific netlists available in the Evaluation Package, which can be downloaded from the Lattice web site at www.latticesemi.com.

Table 9. Core Configurations

Number	Parameter	Configurations			
		CCSDS	DVB	ATSC	OC192
1	wsymb	8	8	8	8
2	n	255	204	207	255
3	k	223	188	187	239
4	rootspace	11	1	1	1
5	fpoly	391	285	285	285
6	gstart	112	0	0	0
7	Numerasrs	0	0	0	0
8	Coretype	ccsds	dvb	atsc	oc192

You can use the IPexpress software tool to help generate new configurations of this IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.