Dynamic Block Reed-Solomon Decoder

User Guide

FPGA-IPUG-02155-1.7

April 2021

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
ATSC	Advanced Television Systems Committee
CCSDS	Consultative Committee for Space Data Systems
DVB	Digital Video Broadcasting
I^2C	Inter-Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
LUT	Look-Up Table
OC	Optical Carrier
OFDM	Orthogonal Frequency Division Multiplexing
RS	Reed Solomon

1. Introduction

Reed-Solomon codes are widely used in various communications and storage applications for forward error correction. Reed-Solomon codes are well suited for burst error correction and are frequently used as outer codes in communication systems. A Reed-Solomon Decoder performs detection and correction of the encoded data at the receiver. Lattice Semiconductor's Dynamic Block Reed-Solomon Decoder (RS Decoder) IP core is compliant with several industry standards including the more recent IEEE 802.16-2004 and can be custom configured to support other non-standard applications as well. The RS Decoder supports a wide range of symbol widths and allows you to define the field polynomial, generator polynomial and several other parameters.

The newer standards like IEEE 802.16-2004 require the use of Reed-Solomon codes with dynamically varying block sizes. Lattice's RS Decoder IP core provides an ideal solution that meets such needs of today's forward error correction world. This core allows the block size and number of check symbols to be varied dynamically through input ports. Lattice's RS Decoder IP can be used with Lattice's RS Decoder for a complete Reed-Solomon code based forward error correction application. For more information on these and other IP products for forward error correction, refer to the Lattice web site at www.latticesemi.com/solutions.

1.1. Quick Facts

Table 1.1 through Table 1.9 provide quick facts about the RS Decoder IP core for LatticeEC™, LatticeECP™, LattceECP2™, LattticeSC™, LatticeSCM™, LatticeXP™, LatticeECP2M™, LatticeXP2™, and LatticeECP3™ devices.

Table 1.1. RS Decoder IP Core for LatticeEC Devices Quick Facts

		RS Decoder IP Configuration					
		OC-192	**CCSDS**	**DVB**	**ATSC**	**IEEE 802.16-2004 SCa/OFDM**	**IEEE 802.16-2004 SC**
Core Requirements	FPGA Family Supported	LatticeEC					
	Minimal Device Needed	LFEC1E	LFEC3E	LFEC1E	LFEC3E	LFEC3E	LFEC3E
Resource Utilization	Targeted Device	LFEC20E-5F672C					
	LUTs	1100	2000	1200	1500	1900	2100
	sysMEM EBRs	2	2	2	2	3	3
	Registers	900	1500	900	1100	1400	1600
Design Tool Support	Lattice Implementation	Lattice Diamond™ 1.0 or ispLEVER® 8.1					
	Synthesis	Synopsys® Synplify™ Pro for Lattice D-2009.12L-1					
	Simulation	Aldec® Active-HDL™ 8.2 Lattice Edition					
		Mentor Graphics® ModelSim™ SE 6.3F					

Table 1.2. RS Decoder IP Core for LatticeECP Devices Quick Facts

		RS Decoder IP Configuration					
		OC-192	CCSDS	DVB	ATSC	IEEE 802.16-2004 SCa/OFDM	IEEE 802.16-2004 SC
Core Requirements	FPGA Families Supported	LatticeECP					
	Minimal Device Needed	LFECP6E					
Resource Utilization	Targeted Device	LFECP20E-5F672C					
	LUTs	1100	2000	1200	1500	1900	2100
	sysMEM EBRs	2	2	2	2	3	3
	Registers	900	1500	900	1100	1400	1600
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1					
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1					
	Simulation	Aldec Active-HDL 8.2 Lattice Edition					
		Mentor Graphics ModelSim SE 6.3F					

Table 1.3. RS Decoder IP Core for LatticeECP2 Devices Quick Facts

		RS Decoder IP Configuration					
		OC-192	CCSDS	DVB	ATSC	IEEE 802.16-2004 SCa/OFDM	IEEE 802.16-2004 SC
Core Requirements	FPGA Families Supported	LatticeECP2					
	Minimal Device Needed	LFE2-6E					
Resource Utilization	Targeted Device	LFE2-50E-7F672C					
	LUTs	1100	1100	1100	1100	1100	1100
	sysMEM EBRs	2	2	2	2	2	2
	Registers	900	900	900	900	900	900
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1					
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1					
	Simulation	Aldec Active-HDL 8.2 Lattice Edition					
		Mentor Graphics ModelSim SE 6.3F					

Table 1.4. RS Decoder IP Core for LatticeECP2 Devices Quick Facts

		RS Decoder IP Configuration					
		OC-192	CCSDS	DVB	ATSC	IEEE 802.16-2004 SCa/OFDM	IEEE 802.16-2004 SC
Core Requirements	FPGA Families Supported	LatticeSC					
	Minimal Device Needed	LFSC3GA15E					
Resource Utilization	Targeted Device	LFSC3GA25E-7F900C					
	LUTs	1200	1200	1200	1200	1200	1200
	sysMEM EBRs	2	2	2	2	2	2
	Registers	900	900	900	900	900	900
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1					
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1					
	Simulation	Aldec Active-HDL 8.2 Lattice Edition					
		Mentor Graphics ModelSim SE 6.3F					

Table 1.5. RS Decoder IP Core for LatticeECP2 Devices Quick Facts

		RS Decoder IP Configuration					
		OC-192	**CCSDS**	**DVB**	**ATSC**	**IEEE 802.16-2004 SCa/OFDM**	**IEEE 802.16-2004 SC**
Core Requirements	FPGA Families Supported	Lattice SCM					
	Minimal Device Needed	LFSCM3GA15EP1					
Resource Utilization	Targeted Device	LFSCM3GA25EP1-7F900C					
	LUTs	1200	1200	1200	1200	1200	1200
	sysMEM EBRs	2	2	2	2	2	2
	Registers	900	900	900	900	900	900
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1					
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1					
	Simulation	Aldec Active-HDL 8.2 Lattice Edition					
		Mentor Graphics ModelSim SE 6.3F					

Table 1.6. RS Decoder IP Core for LatticeXP Devices Quick Facts

		RS Decoder IP Configuration					
		OC-192	**CCSDS**	**DVB**	**ATSC**	**IEEE 802.16-2004 SCa/OFDM**	**IEEE 802.16-2004 SC**
Core Requirements	FPGA Families Supported	Lattice XP					
	Minimal Device Needed	LFXP3E					
Resource Utilization	Targeted Device	LFXP20E-5F484C					
	LUTs	1100	2000	1200	1500	1900	2100
	sysMEM EBRs	2	2	2	2	3	3
	Registers	900	1500	900	1100	1400	1600
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1					
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1					
	Simulation	Aldec Active-HDL 8.2 Lattice Edition					
		Mentor Graphics ModelSim SE 6.3F					

Table 1.7. RS Decoder IP Core for LatticeECP2M Devices Quick Facts

		RS Decoder IP Configuration					
		OC-192	**CCSDS**	**DVB**	**ATSC**	**IEEE 802.16-2004 SCa/OFDM**	**IEEE 802.16-2004 SC**
Core Requirements	FPGA Families Supported	Lattice ECP2M					
	Minimal Device Needed	LFE2M20E					
Resource Utilization	Targeted Device	LFE2M35E-7F484C					
	LUTs	1100	1100	1100	1100	1100	1100
	sysMEM EBRs	2	2	2	2	2	2
	Registers	900	900	900	900	900	900
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1					
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1					
	Simulation	Aldec Active-HDL 8.2 Lattice Edition					
		Mentor Graphics ModelSim SE 6.3F					

Table 1.8. RS Decoder IP Core for LatticeXP2 Devices Quick Facts

		RS Decoder IP Configuration					
		OC-192	**CCSDS**	**DVB**	**ATSC**	**IEEE 802.16-2004 SCa/OFDM**	**IEEE 802.16-2004 SC**
Core Requirements	FPGA Families Supported	Lattice XP2					
	Minimal Device Needed	LFXP2-5E					
Resource Utilization	Targeted Device	LFXP2-17E-7FT256C					
	LUTs	1100	1100	1100	1100	1100	1100
	sysMEM EBRs	2	2	2	2	2	2
	Registers	900	900	900	900	900	900
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1					
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1					
	Simulation	Aldec Active-HDL 8.2 Lattice Edition					
		Mentor Graphics ModelSim SE 6.3F					

Table 1.9. RS Decoder IP Core for LatticeECP3 Devices Quick Facts

		RS Decoder IP Configuration					
		OC-192	**CCSDS**	**DVB**	**ATSC**	**IEEE 802.16-2004 SCa/OFDM**	**IEEE 802.16-2004 SC**
Core Requirements	FPGA Families Supported	Lattice ECP3					
	Minimal Device Needed	LFE3-35EA					
Resource Utilization	Targeted Device	LFE3-95E-8FN672CES					
	LUTs	1100	1100	1100	1100	1100	1100
	sysMEM EBRs	2	2	2	2	2	2
	Registers	900	900	900	900	900	900
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1					
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1					
	Simulation	Aldec Active-HDL 8.2 Lattice Edition					
		Mentor Graphics ModelSim SE 6.3F					

1.2. Features

- 3- to 12-Bit Symbol Width
- Configurable Field Polynomial
- Configurable Generator Polynomial: Starting Root and Root Spacing
- User-defined Codewords
  - Maximum of 4095 symbols
  - Maximum of 256 check symbols
  - Shortened codes
- Off-the-shelf Support for the Following Communication Standards:
  - OC-192
  - DVB
  - CCSDS
  - ATSC
  - IEEE 802.16-2004 WirelessMAN-SCa/OFDM
  - IEEE 802.16-2004 WirelessMAN-SC
- Fully Synchronous
- Systematic Decoder
- Full Handshaking Capability
- Dynamically Variable Block Size
- Dynamically Variable Check Symbols
- Error, Erasure, and Puncturing Modes
- Error Measurement Information

1.3. Block Diagram

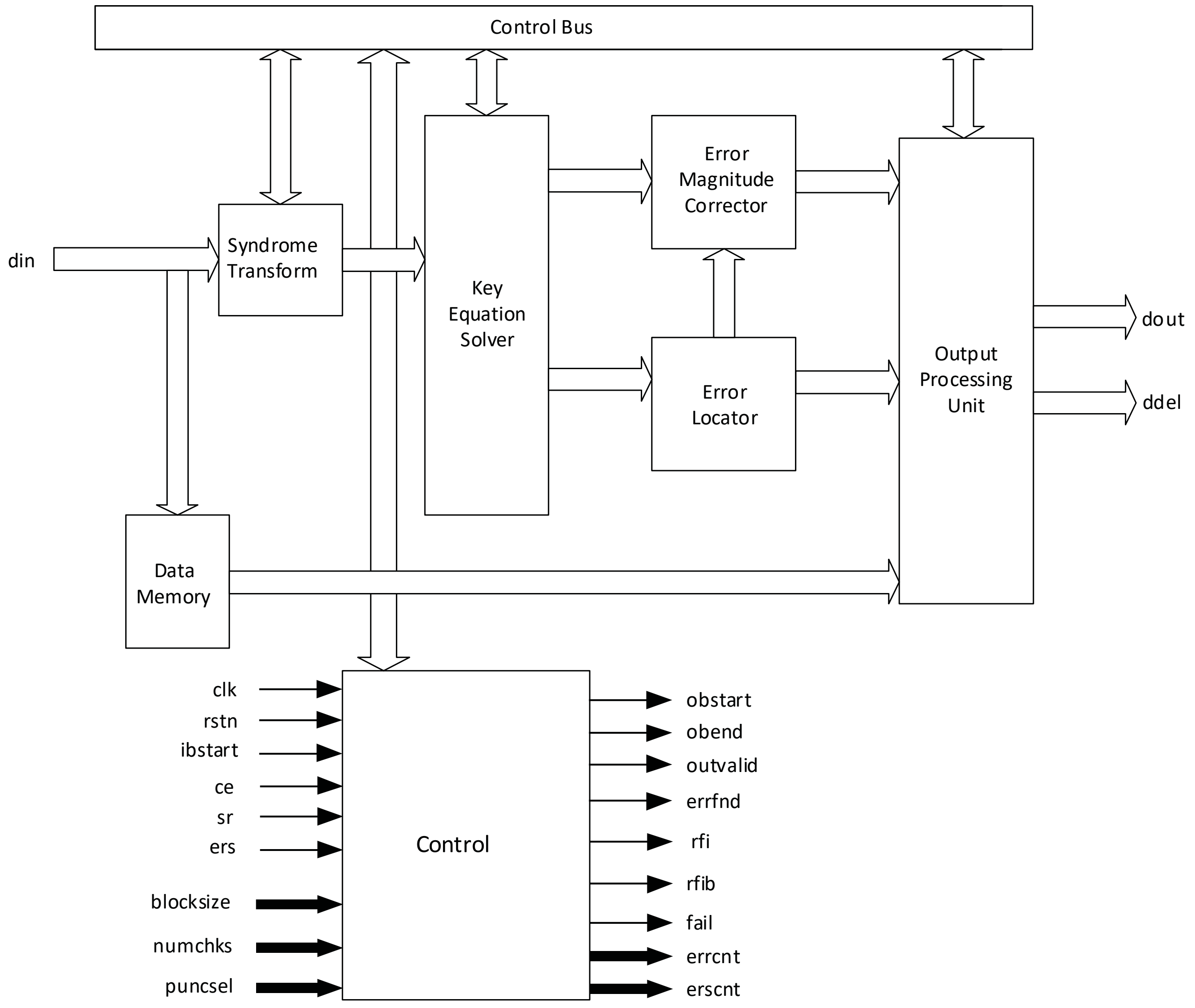


Figure 1.1. RS Decoder Block Diagram

1.3.1. General Description

Reed-Solomon codes are used to perform Forward Error Correction (FEC). FEC introduces controlled redundancy in the data before it is transmitted to allow error correction at the receiver. The redundant data (check symbols) are transmitted with the original data to the receiver. An RS Decoder is used in the receiver to correct any transmission errors. This type of error correction is widely used in data communications applications such as Digital Video Broadcasting (DVB) and Optical Carrier (such as OC-192).

Reed-Solomon codes are written in the format RS(n,k) where k is the number of information symbols and n is the total number of symbols in a codeword or block. Each symbol in the codeword is wsymb bits wide. The RS 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 1.2.

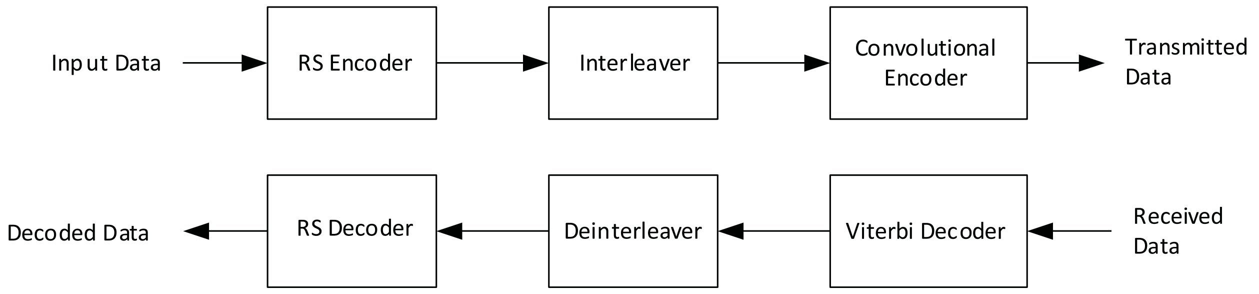


Figure 1.2. Application of Reed-Solomon Code in a Space Communication System

Reed-Solomon codes are defined on a finite field known as Galois field. The size of the field is determined by the symbol width, wsymb, and is equal to 2wsymb. When n is less than its maximum value of 2wsymb-1, the corresponding code RS(n,k) is referred to as a shortened code.

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

1.3.1.1. Field Polynomial

The field polynomial is defined by its decimal value (f). The decimal value of a field polynomial is obtained by set- ting x = 2 in the polynomial. For example, the polynomial x2 + x + 1 in decimal value is 22 + 2 + 1 = 7. The field polynomial can be specified as any prime polynomial with decimal value up to 2wsymb+1 - 1.

1.3.1.2. Generator Polynomial

The generator polynomial determines the value of the check symbols. The generator polynomial can be defined by the parameters starting root (gstart) and root spacing (rootspace). The general form of the generator polynomial is given by:

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

where α is called the primitive element of the field polynomial. For a binary Galois field GF(2), α is equal to 2.

1.3.2. Shortened Codes

When the size of the Reed-Solomon codewords, n, is less than the maximum possible size, 2wsymb-1, they are called shortened codes. For example, RS (204,188) when wsymb = 8 is a shortened code.

1.3.3. Systematic Decoder

The decoder can only decode data encoded by a systematic Reed-Solomon Encoder. In a systematic encoder, the information symbols are unchanged and are followed by check symbols in the output.

1.3.4. Decoding Modes

The decoder can support Error, Erasure, and Puncturing modes. In the error mode no information is available about the symbols in error. In this mode the decoder needs to compute both position and magnitude of the error symbols. In the erasure mode, you can dynamically indicate the erased symbols using the input porters. Erased symbols are those symbols in error whose positions are known in advance. Error mode can be thought of a special case of Erasure mode, when number of erased symbols is zero. Therefore it is not necessary to identify all correct- able errors as erasures through the input porters in the erasure mode and combinations of errors and erasures can be used. If erased symbols are known and the position of the erased symbols can be dynamically indicated using ers, then erasure mode is useful. The symbol correction capability of the decoder increases since the position of the symbol in error is already known and only the magnitude needs to be computed. Generally erasure sup- port substantially increases the decoder latency and resource utilization. Puncturing mode is an optimized version of the erasure mode and can be used when the position of the erased symbols is known in advance. You can define a maximum of (n - k) puncture patterns and can dynamically select one of these patterns using the input port puncsel. The format for the puncture pattern file is explained in a later section. The decoder will be able to correct the errors and erasures successfully if the following conditions are satisfied.

For the Error mode, the number of correctable errors Eerr is given by

Eerr = (n - k) /2, when Block size type is constant.

Eerr = (Number of check symbols) /2, when Block size type is variable and Variable check symbols is not defined.

Eerr = (value on numchks port) /2, when Variable check symbols is defined.

For the Erasure and Puncturing modes, the number of correctable errors Eerr and the number of correctable erasures Eers (given through the input port ers) are bound by the following relations

(2 * Eerr + Eers ð (n - k)) and (Eers ((n - k - 2)), when Block size type is constant.

(2 * Eerr + Eers ð (Number of check symbols)) and (Eers ð (Number of check symbols - 2)), when

Block size type is variable.

2. Functional Description

A block diagram of the RS Decoder is shown in Figure 1.1. The RS Decoder is comprised of the Syndrome Trans- form, Key Equation Solver, Error Locator, Error Magnitude Corrector, Data Memory and Output Processing blocks.

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 the received data polynomial are the same when there are no errors. 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 correctable errors (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 Λ(x) is orthogonal to the syndrome polynomial in the Galois field. This can be represented as:

S(x)Λ(x) = ¾(x) mod x2t

where:

¾(x) is the Error Evaluator polynomial.

2t is the number of check symbols introduced in the encoder.

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

2.1. 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.

2.2. Key Equation Solver

This is the heart of the RS Decoder. This block generates the Error Locator polynomial ⍰(x) (also known as the "Key Equation" as it is the key to solve the decoding problem). After the Error Locator polynomial has been deter- mined, it is used to compute the Error Evaluator polynomial ¾(x)

2.3. 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 up to m cycles, where m is the number of elements in the Galois Field, to determine all the roots. If the roots are determined before m cycles are over, the search is terminated early.

2.4. Error Magnitude Corrector

Once the location of the error has been determined, the Error Magnitude Corrector evaluates the evaluator polynomial at that 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 original data. The addition occurs only when the Error Locator polynomial evaluates to zero.

2.5. Control Unit

The control unit handles the interface, pipelining and handshaking communication between the various blocks and the I/O ports. 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 obstart, obend, outvalid, rfib, errfnd, errcnt, erscnt and fail. Once the block has been processed, the control circuit sends out the rfi signal to the output to start the processing of the next data block.

2.6. Basis Conversion Modules

When core configuration is selected as CCSDS, then two additional Basis Conversion modules are added to the RS Decoder. These modules comply with the CCSDS specification. Dual-basis to normal polynomial-basis conversion module is added after the din input port and normal polynomial-basis to dual-basis conversion module is added before the dout output port.

2.7. Variable Block Size

In the constant Block size type option, the block size value and number of information symbols are provided as constant values through the RS Decoder user interface before core generation. For variable Block size type option, the block size value is provided dynamically through the input port blocksize. The number of the information symbols is calculated from the block size value provided through the input port and the number of check symbols. The number of check symbols can be either constant and defined in the user interface or variable and given through the input port numchks, depending on the parameter Variable check symbols. Once block size value, number of check symbols and number of information symbols are known then the core operates in the same way as when block size was constant.

2.8. Variable Check Symbols

This option can be used when there is requirement for variable error correction capability. One example of this type of application is IEEE 802.16-2004 WirelessMAN-SC. In this option the number of check symbols value is provided dynamically through the input port numchks. Dynamically Variable check symbols option is available only for the Error Decoding mode.

3. Puncturing Pattern File Format

This file contains the pre-defined puncture patterns that are selected using the puncsel signal. This file is necessary when Decoding mode is selected as Puncturing. This file should have the ".cfg" extension. The RS Decoder IP core user interface requires this file during core configuration. The format and a sample of this file is given below, followed by a brief explanation.

Format:

<n1> <n2> <n3> ....<nN>   First line lists the number of punctured symbols in each pattern. N is the total number of puncture patterns. There should be N lines following this line, one for each pattern

<p1> <p2> .... <pn1>   These are the actual puncture patterns. Each line contains one pattern. p1, p2,

<p1> <p2> ... <pn2>   etc. in each pattern are the positions of the punctured symbols from the end of the

...   block for that pattern.

...

<p1> <p2> ... <pnN>

The number of punctured symbols in the first line, <ni>, can be set to zero to indicate there is no puncturing for that pattern.

Sample content:

```
0 4 8 12
0
0 1 2 3
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7 8 9 10 11
```

In the sample content above, the first line contains the number of punctured symbols for each puncture pattern. 0 indicates that for the first puncture pattern there are 0 symbols punctured. 4 indicates that for the second puncture pattern there are 4 symbols punctured. 8 indicates that for the third puncture pattern there are 8 symbols punctured. 12 indicates that for the fourth puncture pattern there are 12 symbols punctured. The first puncture pattern is defined on line 2 and the second puncture pattern is defined on line 3, and so on. Each puncture pattern lists the position of the punctured symbol starting from the end of the block. This means that the last symbol is numbered as

0 and the next to the last symbol is numbered as 1, and so on. Therefore, for the second puncture pattern defined on line 3, number 0 indicates the last symbol in the block is punctured and number 1 indicates the second from the last symbol is punctured. The values are entered in decimal format.

3.1. Default Field Polynomials

The default field polynomials used in the user interface for different symbol widths are given in Table 3.1. The user, however, can enter any valid polynomial.

Table 3.1. Default Field Polynomials

Symbol 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

3.2. Signal Descriptions

Table 3.2 shows the definitions of the interface signals available with the RS Decoder IP Core.

Table 3.2. Interface Signal Descriptions

Port	Bits	I/O	Description
All Configurations			
clk	1	I	System clock. This is the reference clock for input and output data.
rstn	1	I	System wide asynchronous active-low reset signal.
ibstart	1	I	Indicates that the data on din is the first information symbol of a new codeword.
din	3 - 12	I	Input data port. The wsymb parameter defines the port width of this signal.
dout	3 - 12	O	Output data port. The wsymb parameter defines the port width of this signal.
obstart	1	O	Output block start. Indicates the first output data of the codeword on the dout port.
obend	1	O	Output block end. Indicates the last output data of the codeword on the dout port.
outvalid	1	O	Output data valid. Indicates valid data is present on dout.
errfnd	1	O	Error found indicator. Asserted at the same time obend is asserted if the block has at least one symbol in error.
rfi	1	O	Ready for input. Indicates the decoder is ready to receive input data. Typically, this signal is high when the core is ready to read input symbols. This signal is low when the decoder is busy processing a previous block of data and cannot accept new block of data.
For Variable Check Symbols (When the Parameter Variable check symbols is Yes)			
rfib	1	O	Ready for input block. Indicates that the decoder is ready to receive the first information symbol in the block.
numchks	2 - 9	I	This port is used to provide the variable number of check symbols value. The width of this port is defined as the number of bits required to represent the Max. number of check symbols parameter value provided by the user. The width of this port is defined as follows: ceil(log2(Max. number of check sym- bols)). The value at this port is read only when ibstart is high. The operator ceil() stands for the next higher integer.
For Variable Block Size Type Only (When the Parameter Variable block size is "Yes")			
blocksize	3 - 12	I	Variable block size value. The value at this port is read only when ibstart is high. The wsymb parameter defines the port width of this signal.
For Puncturing Mode (when Decoding mode is "Puncturing" and Number of puncture patterns is More than 1)			
puncsel	1 - 8	I	Puncture pattern select signal. The value on this port selects the puncturing pat- tern from the number of predefined patterns for the current block of data. The width of this port is defined as ceil(log2(Number of puncture patterns)). The value at this port is read only when ibstart is high.
Optional I/O			
ce	1	O	Clock enable. While this is de-asserted, the decoder will ignore all other synchronous inputs and maintain its current state.
sr	1	O	Synchronous reset. Asserted for at least one symbol duration in order to reinitialize the decoder state. Input data symbols sampled before sr is asserted are not given at the output.
ddel	1	O	Original uncorrected data output. A delayed copy of the input data block. Data is presented on ddel concurrently with the decoded block on dout. The wsymb parameter defines the port width of this signal.

Port	Bits	I/O	Description
errcnt	1-8	O	Error Counter. Provides the number of corrected errors in the most recent out- put block. The bus width errwidth is equal to the number of bits required to represent the maximum possible number of correctable errors, as given in the following equation: When Block size type is Constant, errwidth is defined as errwidth = ciel(log2((n-k+1)/2)) when Error Decoding mode is selected. errwidth = ciel(log2(n-k-0.5)) when Erasure or Puncturing Decoding mode is selected. When Block size type is Variable and Variable check symbols is No, errwidth is defined as errwidth = ciel(log2((Number of check sym- bols+1)/2)) when Error Decoding mode is selected. errwidth = ciel(log2(Number of check symbols-0.5)) when Erasure or Puncturing Decoding mode is selected. When Variable check symbols is Yes, errwidth is defined as errwidth = ciel(log2((Max. number of check symbols+1)/2)) The operator ciel() stands for the next higher integer.
erscnt	1-8	O	Erasure Counter. Provides a count of the number of erasures fed into the decoder in the most recent input data block. The bus width erswidth is equal to the number of bits required to represent the maximum possible number of correctable erasures, as given in the following equation: When Block size type is Constant, erswidth is defined as erswidth = ciel(log2(n-k-1.5)). When Block size type is Variable, erswidth is defined as erswidth = ciel(log2(Number of check symbols -1.5)) The operator ciel() stands for the next higher integer.
fail	1	O	Decoding failure indicator. Asserted at the same time obend is asserted to indicate that the block has more errors than the decoder can correct.

3.3. Timing Specifications

The decoder receives the data in blocks. The assertion of signal ibstart indicates the first symbol of the new block of data at the input of the decoder. The ibstart signal should be asserted only during the first clock cycle of a data block. The ibstart signal should not be re-asserted until the decoder is ready to receive the next block of data as indicated by rfib going high. The signal rfib can be used to generate the ibstart signal. If a new block of input data has to be applied before the decoder is ready for a new block, the decoder operation should be reset using the synchronous reset signal sr.

Figure 3.1 shows the I/O signals' status after asynchronous reset rstn is asserted at the beginning of the block. The output rfi goes high after reset to indicate the core is ready to receive input data. After ibstart signal is asserted, the decoder reads in the data block sequentially and starts the decoding process. When the decoded data is given at the output, obstart is asserted for one clock cycle during the first decoded output symbol. The output obend is asserted for one clock cycle when the last decoded symbol is given at the dout port.

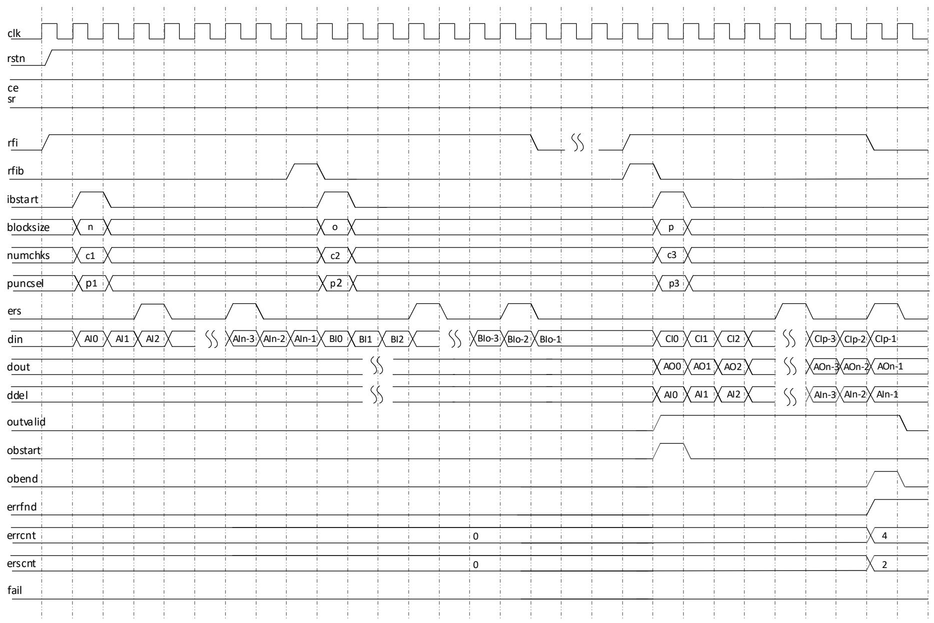


Figure 3.1. RS Decoder Normal Operation Timing Diagram

Figure 3.2 illustrates the output status when sr is asserted. When sr is asserted during the decoding process, it reinitializes the decoder state similar to power on reset state. The output data stops appearing at the output. The decoder operation can be started again by asserting the ibstart signal.

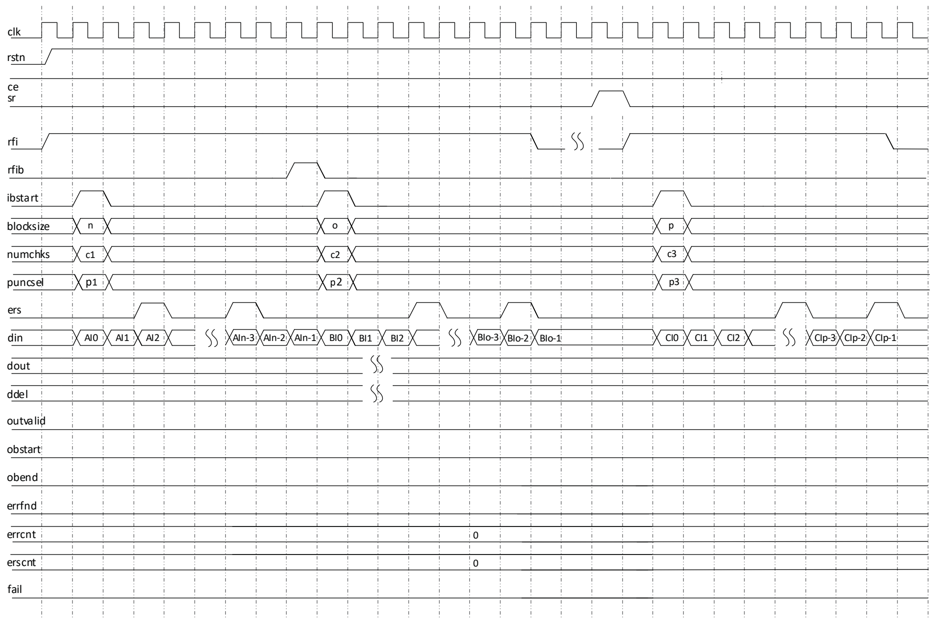


Figure 3.2. Effect of Synchronous Reset on the Output Data from the Decoder

Figure 3.3 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 din (that occurs during ce going low) is not recognized by the decoder.

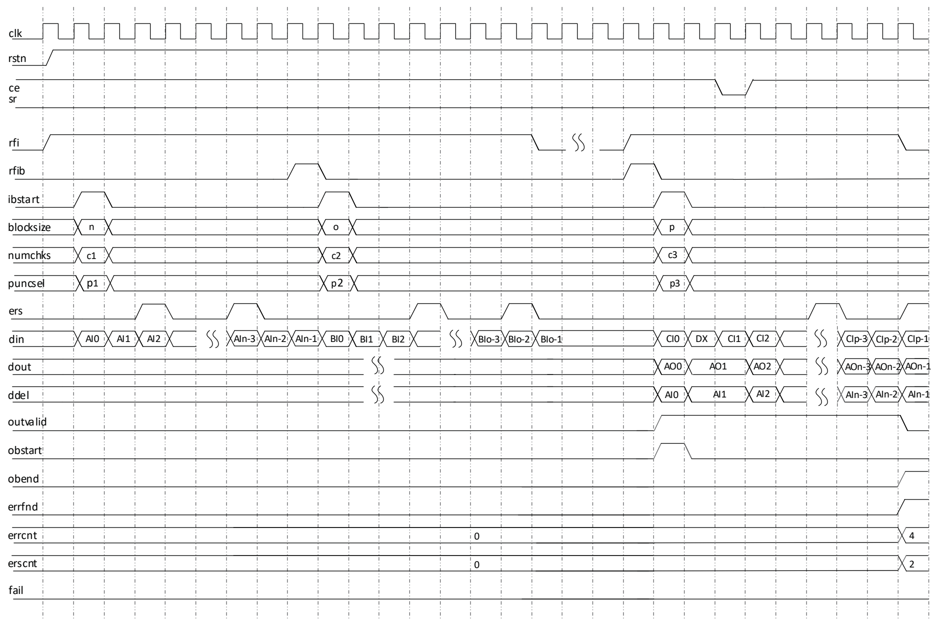


Figure 3.3. Effect of Clock Enable on the Output Data from Decoder

3.4. Parameter Settings

The IPexpress™ tool is used to create IP and architectural modules in the Diamond and ispLEVER software. Refer to the IP Core Generation section for a description on how to generate the IP.

The RS Decoder IP core Configuration user interface allows you to create a custom configuration or to select one of the standard configurations: OC-192, CCSDS, DVB, ATSC, IEEE 802.16-2004 WirelessMAN-SCa/OFDM and IEEE 802.16-2004 WirelessMAN-SC. Table 3.3 provides the list of user configurable parameters for the RS Decoder IP core.

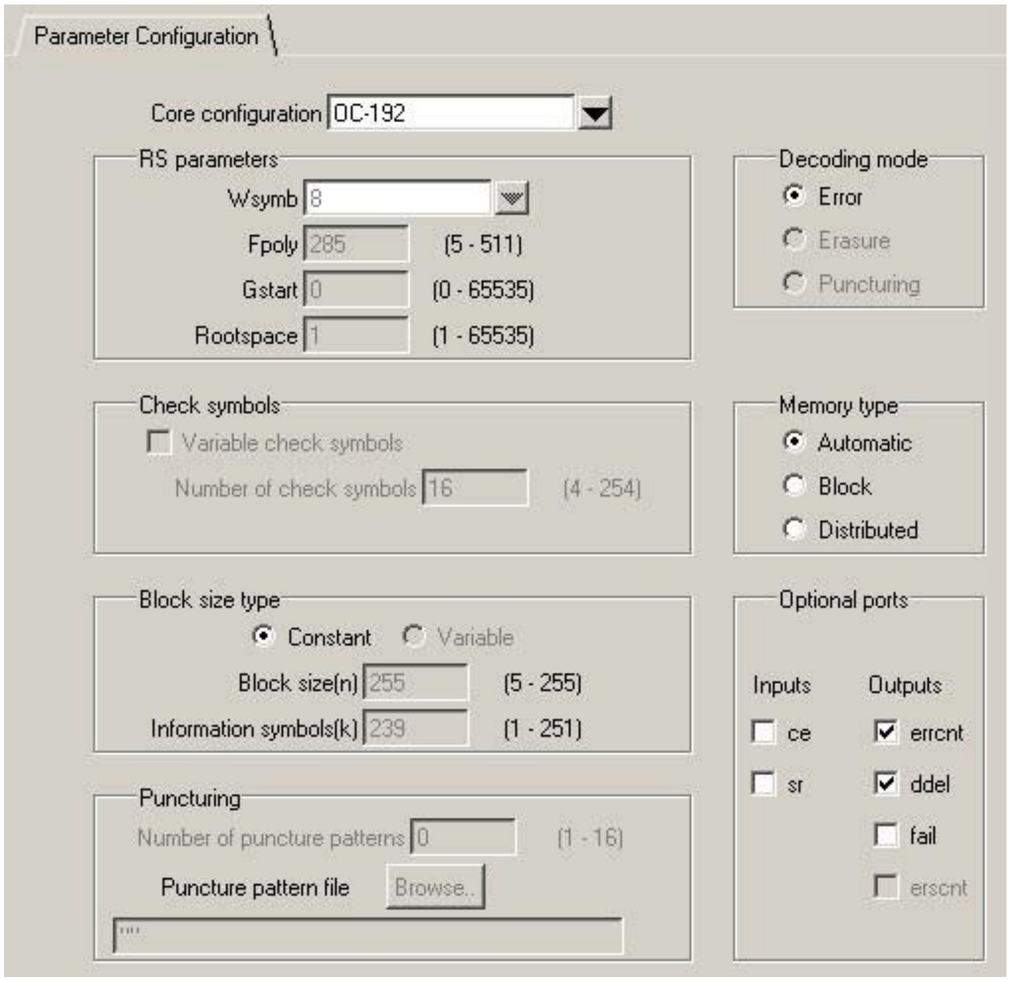
Table 3.3. User Configurable Parameters

Parameter	Range/Options	Default
Core Configuration		
Core configuration	Custom, OC-192, CCSDS, DVB, ATSC, IEEE 802.16-2004 SCa, IEEE 802.16-2004 SC	OC-192
Connect reset port to GSR	Yes/No	Yes
RS Parameters		
wsymb	3 - 12 bits	8 bits
fpoly	5 - 8191	285
gstart	0 - 65535	0
rootspace	1 - 65535	1
Check Symbols		
Variable check symbols	Yes/No	No
Number of check symbols	4 - ((2^{wsymb})-2). Maximum value is limited to 256.	16
Max. number of check symbols	4 - ((2^{wsymb})-2). Maximum value is limited to 128.	32
Block Size Type		
Block size type	Variable} if Variable check symbols is checked. {Constant, Variable} if Variable check symbols is not checked. {Constant} if wsymb is 3 and Decoding mode is Erasure.	Constant
Block size(n)	5 - ((2^{wsymb})-1) Default value is ((2^{wsymb})-1)	255
Information symbols(k)	1 - (n-4)	239

Puncturing		
Number of puncture patterns	1 - (n-k) when Block size is Constant. 1 - (Number of check symbols) when Block size is Variable.	1
Puncture pattern file	Edit field to enter the file name directly or indirectly by using the browse button.	
Decoding Mode		
Decoding mode	{Error, Erasure, Puncturing}	Error
Memory Type		
Memory type	{Automatic, Block, Distributed}	Automatic
Optional Ports		
ce	Yes/No	No
sr	Yes/No	No
errcnt	Yes/No	Yes
ddel	Yes/No	Yes
fail	Yes/No	No
erscnt	Yes/No	

3.5. RS Decoder Configuration User Interface

Figure 3.4 shows the contents of the RS Decoder IP core Configuration user interface.



Parameter Configuration

Core configuration: OC-192

RS parameters
- Wsymb: 8
- Fpoly: 285 (5 - 511)
- Gstart: 0 (0 - 65535)
- Rootspace: 1 (1 - 65535)

Decoding mode: [✓ Error] Erasure Puncturing

Check symbols
- ☐ Variable check symbols
- Number of check symbols: 16 (4 - 254)

Memory type: [✓ Automatic] Block Distributed

Block size type: [✓ Constant] Variable
- Block size(n): 255 (5 - 255)
- Information symbols(k): 239 (1 - 251)

Optional ports
- Inputs: ☐ ce ☐ sr
- Outputs: ☑ errcnt ☑ ddel ☐ fail ☐ erscnt

Puncturing
- Number of puncture patterns: 0 (1 - 16)
- Puncture pattern file: Browse..
- ""

Figure 3.4. RS Decoder IP Core Configuration User Interface

- **Core Configuration** – Selects between custom and pre-defined standard configurations. Table A.1 defines the fixed parameter values for different standard configurations.
- **RS Parameters**
  - **Wsymb** – This parameter sets symbol width.
  - **Fpoly** – This parameter sets the decimal value of the field polynomial. Table 3.1 provides the default field polynomial values for different symbol widths.
  - **Gstart** – This parameter sets the offset value of the generator polynomial. The starting value for the first root of the genera- tor polynomial is calculated as rootspace * gstart.
  - **Rootspace** – This parameter sets the root spacing of the generator polynomial. The value of rootspace must satisfy the following equation: GCD(rootspace, 2wsymb-1) = 1. GCD is Greatest Common Divisor.

- **Check Symbols**
  - **Variable Check Symbols** – This option allows the number of check symbols to be varied through the port in addition to varying the block size dynamically. In this case, the number of check symbols is defined through the input port numchks. This option is available only when Block size type is Variable and Decoding mode is Error.
  - **Number of Check Symbols** – Constant value for the Number of check symbols in the codeword. This parameter is available when Block size type is selected as Variable and Variable check symbols is not checked.
  - **Max. Number of Check Symbols** – Maximum value for number of check symbols provided through the input port numchks. This parameter selection is available only when Variable check symbols is checked.
- **Block Size Type** – This parameter specifies whether block size is provided as a constant value or varied through the input port. If Block size type is selected as Variable, the block size is read from the input port block size. Options depend on Variable check symbols.
  - **Block Size(n)** – This parameter specifies the total number of symbols in the codeword. Defined only if Block size type is Constant.
  - **Information Symbols(k)** – This parameter specifies the number of information symbols in the codeword. Defined only if Block size type is Constant. The value of k also depends on n as the maximum value of (n-k) is limited to 256
- **Puncturing**
  - **Number of Puncture Patterns** – This is the number of pre-defined puncture patterns that can be dynamically selected using puncsel. This parameter is enabled when Decoding mode is selected as Puncturing.
  - **Puncture Pattern File** – This is the file containing the pre-defined puncture patterns. The format of the puncture pattern file is described in the Puncture Pattern File Format section of this document. The browse button to load the puncture pattern file is enabled when Decoding mode is selected as Puncturing. The file should have a .cfg extension.
- **Decoding Mode** – Selects between different decoding modes. The selection of this parameter depends on the application requirements.
- **Memory Type** – Specifies the type of memory used for storing input data. If Memory type is selected as Block, then EBR memory is used. If Memory type is selected as Distributed then distributed memory is used. If Memory type is selected as Automatic then memory will be selected in a most optimized way depending on the other parameters selected.
- **Optional Ports**
  - **ce** – Determines whether the input port ce (clock enable) is present.
  - **sr** – Determines whether the input port sr (synchronous reset) is present.
  - **errcnt** – Determines whether the output port errcnt (error count) is present.
  - **ddel** – Determines whether the output port ddel (delayed data) is present.
  - **fail** – Determines whether the output port fail (decoding failure) is present.
  - **erscnt** – Determines whether the output port erscnt (erasure count) is present.

4. IP Core Generation

This chapter provides information on licensing the RS Decoder IP core, generating the core using the Diamond or ispLEVER software IPexpress tool, running functional simulation, and including the core in a top-level design. The Lattice RS Decoder IP core can be used in LatticeECP3, LatticeECP2/M, LatticeECP, LatticeSC/M, LatticeXP, and LatticeXP2 device families.

4.1. Licensing the IP Core

An IP license is required to enable full, unrestricted use of the RS Decoder IP core in a complete, top-level design. An IP license that specifies the IP core and device family is required to enable full use of the core in Lattice devices. Instructions on how to obtain licenses for Lattice IP cores are given at:

http://www.latticesemi.com/products/intellectualproperty/aboutip/isplevercoreonlinepurchas.cfm

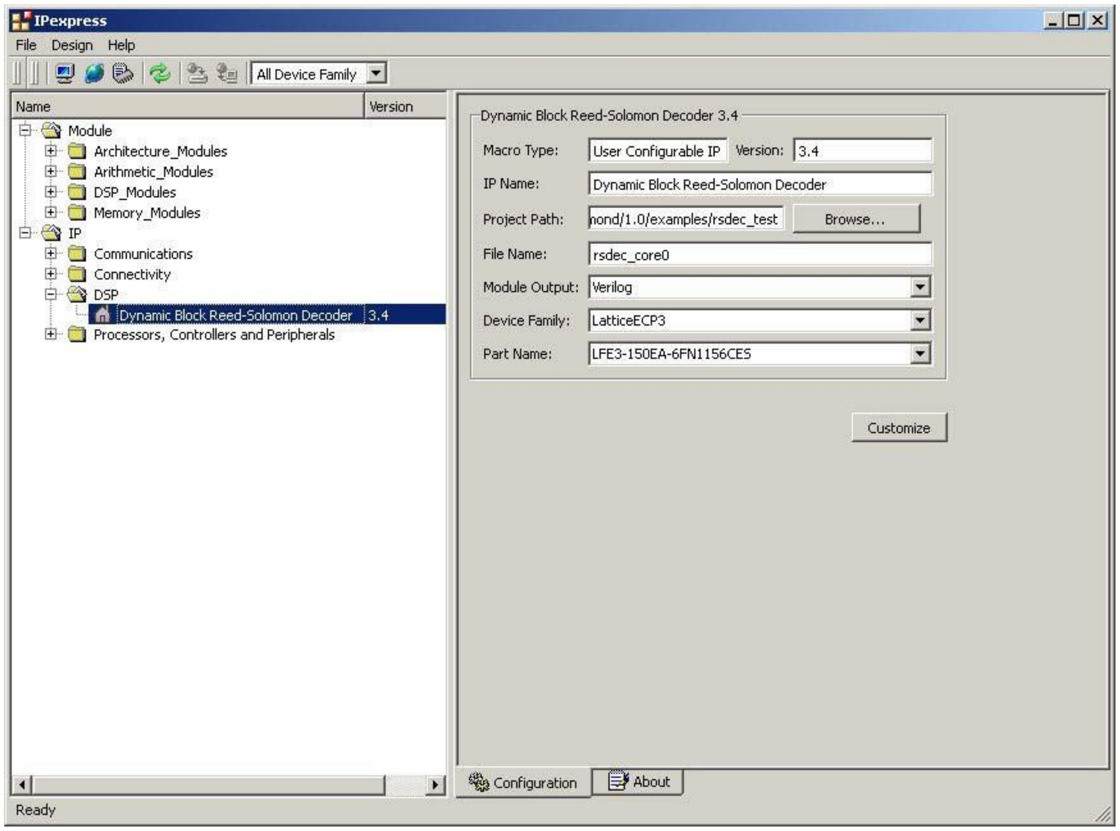
Users may download and generate the IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The RS Decoder IP core also supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license (see the Instantiating the Core section for further details). However, a license is required to enable timing simulation, to open the design in the Diamond or ispLEVER EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

4.2. Getting Started

The RS Decoder IP core is available for download from the Lattice IP Server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP core has been installed, the IP core will be available in the IPexpress user interface dialog box shown in Figure 4.1.

The IPexpress tool user interface dialog box for the RS Decoder IP core is shown in Figure 4.1. To generate a specific IP core configuration, specify:

- **Project Path** – Path to the directory where the generated IP files will be located.
- **File Name** – *username* designation given to the generated IP core and corresponding folders and files.
- (Diamond) **Module Output** – Verilog or VHDL.
- (ispLEVER) **Design Entry Type** – Verilog HDL or VHDL.
- **Device Family** – Device family to which IP is to be targeted (such as LatticeSCM, Lattice ECP2M, LatticeECP3, etc.). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.



Dynamic Block Reed-Solomon Decoder 3.4

Macro Type: User Configurable IP Version: 3.4

IP Name: Dynamic Block Reed-Solomon Decoder

Project Path: nond/1.0/examples/rsdec_test

File Name: rsdec_core0

Module Output: Verilog

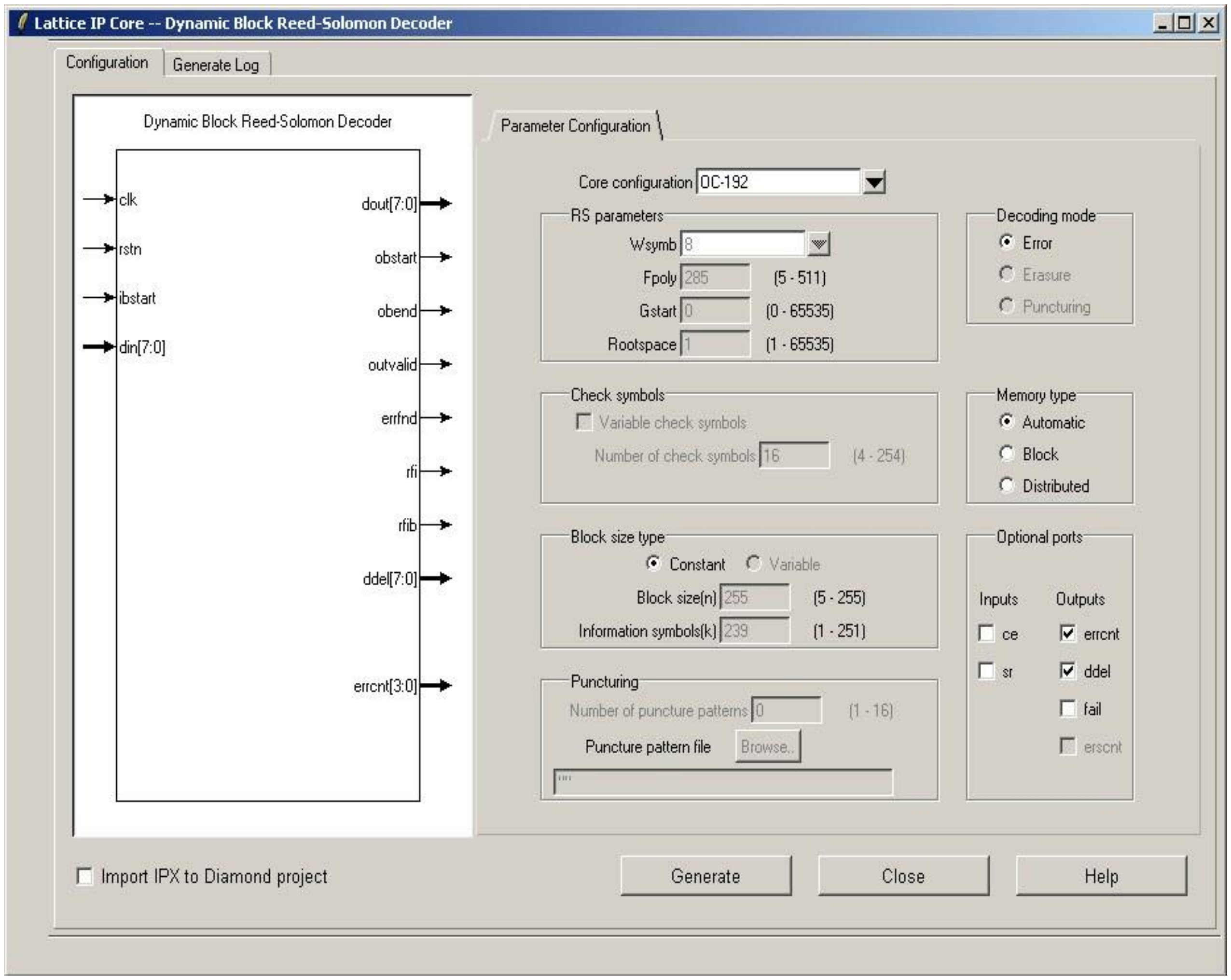
Device Family: LatticeECP3

Part Name: LFE3-150EA-6FN1156CES

Figure 4.1. IPexpress Tool Dialog Box (Diamond Version)

Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output (Design Entry in ispLEVER), Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration, click the Customize button in the IPexpress tool dialog box to display the RS Decoder IP core Configuration user interface, as shown in Figure 4.2. From this dialog box, you can select the IP parameter options specific to their application. Refer to the Parameter Settings section for more information on the parameter settings.



Core configuration: OC-192

RS parameters
- Wsymb: 8
- Fpoly: 285 (5 - 511)
- Gstart: 0 (0 - 65535)
- Rootspace: 1 (1 - 65535)

Decoding mode: [✓ Error] Erasure Puncturing

Check symbols
- Variable check symbols: ☐
- Number of check symbols: 16 (4 - 254)

Memory type: [✓ Automatic] Block Distributed

Block size type: [✓ Constant] Variable
- Block size(n): 255 (5 - 255)
- Information symbols(k): 239 (1 - 251)

Puncturing
- Number of puncture patterns: 0 (1 - 16)
- Puncture pattern file: Browse..
- ""

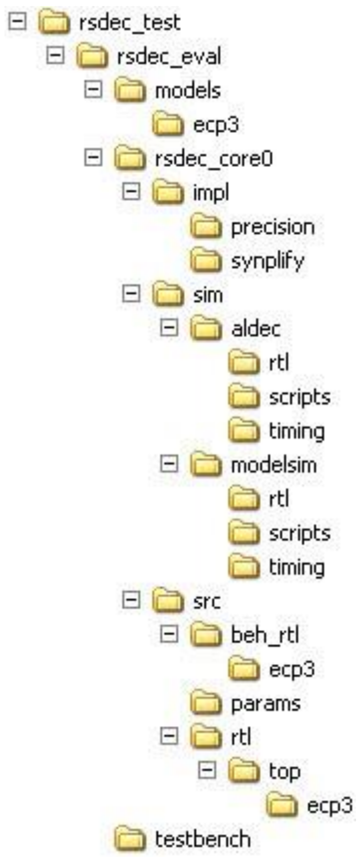
Optional ports
- Inputs: ☐ ce, ☐ sr
- Outputs: ☑ errcnt, ☑ ddel, ☐ fail, ☐ erscnt

☐ Import IPX to Diamond project

Figure 4.2. Configuration User Interface (Diamond Version)

4.3. IPexpress-Created Files and Top Level Directory Structure

When you click the Generate button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified *Project Path* directory. The directory structure of the generated files is shown in Figure 4.3.



```
rsdec_test
    rsdec_eval
        models
            ecp3
        rsdec_core0
            impl
                precision
                synplify
            sim
                aldec
                    rtl
                    scripts
                    timing
                modelsim
                    rtl
                    scripts
                    timing
            src
                beh_rtl
                    ecp3
                params
                rtl
                    top
                        ecp3
        testbench
```

Figure 4.3. Lattice RS Decoder IP Core Directory Structure

Table 4.1 provides a list of key files and directories created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name specified in the IPexpress tool.

Table 4.1. File List

File	Description
<username>_inst.v	This file provides an instance template for the IP.
<username>.v	This file provides the RS Decoder core for simulation.
<username>_beh.v	This file provides a behavioral simulation model for the RS Decoder core.
<username>_bb.v	This file provides the synthesis black box for the user's synthesis.
<username>.ngo	The ngo files provide the synthesized IP core.
<username>.lpc	This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool.
<username>.ipx	The IPX file holds references to all of the elements of an IP or Module after it is generated from the IPexpress tool (Diamond version only). The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP/Module generation user interface when an IP/Module is being re-generated.
<username>_top.[v,vhd]	This file provides a module which instantiates the RS Decoder core. This file can be easily modified for the user's instance of the RS Decoder core. This file is located in the rsdec_eval/<username>_/src/rtl/top/ directory.
<username>_generate.tcl	Created when the user interface Generate button is clicked, invokes generation, may be run from command line.
<username>_generate.log	IPexpress scripts log file.
<username>_gen.log	IPexpress IP generation log file

4.4. Instantiating the Core

The generated RS Decoder IP core package includes black-box (<username>_bb.v) and instance (<user- name>_inst.v) templates that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file that can be used as an instantiation template for the IP core is provided in \<project_dir>\rsdec_eval\<username>\src\rtl\top. Users may also use this top-level reference as the starting template for the top-level for their complete design.

4.5. Running Functional Simulation

Simulation support for the RS Decoder IP core is provided for Aldec Active-HDL (Verilog and VHDL) simulator, Mentor Graphics ModelSim simulator. The functional simulation includes a configuration-specific behavioral model of the RS Decoder IP core. The test bench sources stimulus to the core, and monitors output from the core. The generated IP core package includes the configuration-specific behavior model (<username>_beh.v) for functional simulation in the *Project Path* root directory. The simulation scripts supporting ModelSim evaluation simulation is provided in \<project_dir>\rsdec_eval\<username>\sim\modelsim\scripts. The simulation script supporting Aldec evaluation simulation is provided in \<project_dir>\rsdec_eval\<username>\sim\aldec\scripts. Both Modelsim and Aldec simulation is supported via test bench files provided in \<project_dir>\rsdec_eval\testbench. Models required for simulation are provided in the corresponding \models folder.

To run the Aldec evaluation simulation:

1. Open Active-HDL.
2. Under the **Tools** tab, select **Execute Macro**.
3. Browse to folder \<project_dir>\rsdec_eval\<username>\sim\aldec\scripts and execute one of the "do" scripts shown.

To run the ModelSim evaluation simulation:

1. Open ModelSim.
2. Under the **File** tab, select **Change Directory** and choose the folder <project_dir>\rsdec_eval\<username>\sim\modelsim\scripts.
3. Under the **Tools** tab, select **Execute Macro** and execute the ModelSim *do* script shown.

Note: When the simulation completes, a pop-up window will appear asking "Are you sure you want to finish?" Answer No to analyze the results (answering Yes closes ModelSim).

4.6. Synthesizing and Implementing the Core in a Top-Level Design

Synthesis support for the RS Decoder IP core is provided for Mentor Graphics Precision or Synopsys Synplify. The RS Decoder IP core itself is synthesized and is provided in NGO format when the core is generated in IPexpress. Users may synthesize the core in their own top-level design by instantiating the core in their top-level as described previously and then synthesizing the entire design with either Synplify or Precision RTL Synthesis. The following text describes the evaluation implementation flow for Windows platforms. The flow for Linux and UNIX platforms is described in the Readme file included with the IP core. The top-level files <username>_top.v are provided in

\<project_dir>\rsdec_eval\<username>\src\rtl\top. Push-button implementation of the reference design is supported via Diamond or ispLEVER project files, <username>.syn, located in the following directory:

\<project_dir>\rsdec_eval\<username>\impl\(synplify or precision). To use these project files using Synplify:

To use this project file in Diamond:

1. Choose **File > Open > Project**.
2. Browse to \<project_dir>\rsdec_eval\<username>\impl\synplify (or precision) in the **Open Project** dialog box.
3. Select and open <username>.ldf. All the files needed to support top-level synthesis and implementation are imported to the project.
4. Select the **Process** tab in the left-hand user interface window.
5. Implement the complete design via the standard Diamond user interface flow.

To use this project file in ispLEVER:

1. Choose **File > Open Project**.
2. Browse to \<project_dir>\rsdec_eval\<username>\impl\synplify (or precision) in the **Open Project** dialog box.
3. Select and open <username>.syn. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the device top-level entry in the left-hand user interface window.
5. Implement the complete design via the standard ispLEVER user interface flow.

4.7. Hardware Evaluation

The RS Decoder IP core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase of an IP license. It may also be used to evaluate the core in hardware in user-defined designs.

4.7.1. Enabling Hardware Evaluation in Diamond

Choose Project > Active Strategy > Translate Design Settings. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

4.7.2. Enabling Hardware Evaluation in ispLEVER

In the Processes for Current Source pane, right-click the Build Database process and choose Properties from the dropdown menu. The hardware evaluation capability may be enabled/disabled in the Properties dialog box. It is enabled by default.

5. Updating/Regenerating the IP Core

By regenerating an IP core with the IPexpress tool, you can modify any of its settings including device type, design entry method, and any of the options specific to the IP core. Regenerating can be done to modify an existing IP core or to create a new but similar one.

5.1. Regenerating an IP Core in Diamond

To regenerate an IP core in Diamond:

1. In IPexpress, click the **Regenerate** button.
2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.
3. IPexpress shows the current settings for the module or IP in the **Source** box. Make your new settings in the Target box.
4. To generate a new set of files in a new location, set the new location in the **IPX Target File** box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an .ipx extension.
5. Click **Regenerate**. The module's dialog box opens showing the current option settings.
6. In the dialog box, choose the desired options.

   Note: To get information about the options, click Help. Also, check the About tab in IPexpress for links to technical notes and user guides. IP may come with additional information. As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.
7. To import the module into your project, if it's not already there, select **Import IPX to Diamond Project** (not available in stand-alone mode).
8. Click **Generate**.
9. Check the **Generate Log** tab to check for warnings and error messages.
10. Click **Close**.

The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 above.

5.2. Regenerating an IP Core in ispLEVER

To regenerate an IP core in ispLEVER:

1. In the IPexpress tool, choose **Tools > Regenerate IP/Module**.
2. In the **Select a Parameter File** dialog box, choose the Lattice Parameter Configuration (.lpc) file of the IP core to regenerate, and click **Open**.
3. The **Select Target Core Version, Design Entry, and Device** dialog box shows the current settings for the IP core in the **Source Value** box. Make your new settings in the **Target Value** box.
4. To generate a new set of files in a new location, set the location in the **LPC Target File** box. The base of the .lpc file name will be the base of all the new file names. The LPC Target File must end with an .lpc extension.
5. Click **Next**. The IP core's dialog box opens showing the current option settings.
6. In the dialog box, choose desired options.

   Note: To get information about the options, click Help. Also, check the About tab in the IPexpress tool for links to technical notes and user guides. The IP core might come with additional information. As the options change, the schematic diagram of the IP core changes to show the I/O and the device resources the IP core will need.
7. Click **Generate**.
8. Click the **Generate Log** tab to check for warnings and error messages.

6. Ordering Part Number

- RSDEC-DBLK-E2- U3 – RS Decoder Core for LatticeECP/EC.
- RSDEC-DBLK-P2-U3 – RS Decoder Core for LatticeECP2/S
- RSDEC-DBLK-PM-U3 – RS Decoder Core for LatticeECP2M/S
- RSDEC-DBLK-E3-U3 – RS Decoder Core for LatticeECP3
- RSDEC-DBLK-XP-U3 – RS Decoder Core for LatticeXP
- RSDEC-DBLK-X2-U3 – RS Decoder Core for LatticeXP2
- RSDEC-DBLK-SC-U3 – RS Decoder Core for LatticeSC/M

7. Resource Utilization

This appendix gives resource utilization information for Lattice FPGAs using the RS Decoder IP core.

IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the Diamond and ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and Diamond or ispLEVER help system. For more information on the Diamond or ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.

Table A.1 provides the parameter settings for the standard configurations shown in Table A.2 through Table A.8.

Table A.1. Parameter Settings for Standard Configurations

Parameter Name	Core Configuration					
	OC-192 (config1)	**CCSDS (config2)**	**DVB (config3)**	**ATSC (config4)**	**IEEE 802.16-2004 WirelessMA-SCa or WirelessMA-OFDM (config5)**	**IEEE 802.16-2004 WirelessMAN-SC (config6)**
RS Parameters						
wsymb	8	8	8	8	8	8
fpoly	285	391	285	285	285	285
gstart	0	112	0	0	0	0
rootspace	1	11	1	1	1	1
Check Symbols						
Variable check symbols	No	No	No	No	No	Yes
Number of check symbols	NA	NA	NA	NA	16	NA
Max. number of check symbols	NA	NA	NA	NA	NA	32
Block Size Type						
Block size type	Constant	Constant	Constant	Constant	Variable	Variable
Block size(n)	255	255	204	207	NA	NA
InfoSymbols(k)	239	223	188	187	NA	NA
Puncturing						
Number of puncture patterns	NA	NA	NA	NA	4	NA
Decoding Mode						
Decoding mode	Error	Error	Error	Error	Puncturing	Error
Memory Type						
Memory type	Automatic	Automatic	Automatic	Automatic	Automatic	Automatic
Optional Input/Output Ports						
ce	No	No	No	No	No	No
sr	No	No	No	No	No	No
errcnt	Yes	Yes	Yes	Yes	Yes	Yes
ddel	Yes	Yes	Yes	Yes	Yes	Yes
fail	No	No	No	No	No	No
erscnt	NA	NA	NA	NA	Yes	NA

Table A.2. Performance and Resource Utilization

IPexpress User-Configurable Mode	Slices	LUTs	Registers	sysMEM™ EBRs	I/O	f_{MAX} (MHz)
OC-192	588	1171	795	2	37	123
CCSDS	980	1947	1349	2	38	114
DVB	604	1196	802	2	37	124
ATSC	766	1520	969	2	37	113
IEEE 802.16-2004 WirelessMAN SCa	927	1835	1279	2	51	116
IEEE 802.16-2004 WirelessMAN SC	1044	2066	1486	2	52	104

Note: Performance and utilization data are generated using an LFEC/P20E-5F672C device with Lattice Diamond 1.0 and Synplify Pro for Lattice D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP/EC family.

7.1. LatticeECP2 and LatticeECP2S FPGAs

Table A.3. Performance and Resource Utilization

IPexpress User-Configurable Mode	Slices	LUTs	Registers	sysMEM EBRs	I/O	fMAX (MHz)
OC-192	562	1117	791	2	37	175
CCSDS	963	1917	1322	2	38	157
DVB	591	1173	792	2	37	150
ATSC	756	1500	960	2	37	166
IEEE 802.16-2004 WirelessMAN SCa	917	1818	1252	2	51	152
IEEE 802.16-2004 WirelessMAN SC	1037	2056	1493	2	52	137

Note: Performance and utilization data are generated using an LFE2-50E/S-7F672C device with Lattice Diamond 1.0 and Synplify Pro for Lattice D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2/S family.

7.2. LatticeECP2M and LatticeECP2MS FPGAs

Table A.4. Performance and Resource Utilization

IPexpress User-Configurable Mode	Slices	LUTs	Registers	sysMEM EBRs	I/O	fMAX (MHz)
OC-192	562	1117	791	2	37	169
CCSDS	963	1917	1322	2	38	163
DVB	591	1173	792	2	37	178
ATSC	756	1500	960	2	37	160
IEEE 802.16-2004 WirelessMAN SCa	917	1818	1252	3	51	151
IEEE 802.16-2004 WirelessMAN SC	1037	2056	1493	3	52	145

Note: Performance and utilization data are generated using an LFE2M35E/SE-7F484C device with Lattice Diamond 1.0 and Synplify Pro for Lattice D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2M/S family.

7.3. LatticeECP3 FPGAs

Table A.5. Performance and Resource Utilization

IPexpress User-Configurable Mode	Slices	LUTs	Registers	sysMEM EBRs	I/O	fMAX (MHz)
OC-192	564	1062	791	2	37	148
CCSDS	990	1884	1322	2	38	149
DVB	591	1123	792	2	37	156
ATSC	776	1476	960	2	37	144
IEEE 802.16-2004 WirelessMAN SCa	912	1746	1252	2	51	145
IEEE 802.16-2004 WirelessMAN SC	1067	2031	1493	2	52	142

Note: Performance and utilization data are generated using an LFE3-95E-8FN672CES device with Lattice Diamond 1.0 and Synplify Pro for Lattice D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.

7.4. LatticeXP FPGAs

Table A.6. Performance and Resource Utilization

IPexpress User-Configurable Mode	Slices	LUTs	Registers	sysMEM EBRs	I/O	fMAX (MHz)
OC-192	588	1171	795	2	37	110
CCSDS	980	1947	1349	2	38	108
DVB	604	1196	802	2	37	111
ATSC	766	1520	969	2	37	103
IEEE 802.16-2004 WirelessMAN SCa	928	1837	1279	2	51	109
IEEE 802.16-2004 WirelessMAN SC	1044	2066	1486	2	52	85

Note: Performance and utilization data are generated using an LFXP20E-5F484C device with Lattice Diamond 1.0 and Synplify Pro for Lattice D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeXP family.

7.5. LatticeXP2 FPGAs

Table A.7. Performance and Resource Utilization

IPexpress User-Configurable Mode	Slices	LUTs	Registers	sysMEM EBRs	I/O	fMAX (MHz)
OC-192	562	1117	791	2	37	140
CCSDS	963	1917	1322	2	38	128
DVB	591	1173	792	2	37	157
ATSC	756	1500	960	2	37	128
IEEE 802.16-2004 WirelessMAN SCa	917	1818	1252	2	51	126
IEEE 802.16-2004 WirelessMAN SC	1037	2056	1493	2	52	127

Note: Performance and utilization data are generated using an LFXP2-30E-7F484C device with Lattice Diamond 1.0 and Synplify Pro for Lattice D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeXP2 family.

7.6. LatticeSC and LatticeSCM FPGAs

Table A.8. Performance and Resource Utilization

IPexpress User-Configurable Mode	Slices	LUTs	Registers	sysMEM EBRs	I/O	fMAX (MHz)
OC-192	591	1113	803	2	37	267
CCSDS	1033	1958	1348	2	38	245
DVB	642	1219	804	2	37	287
ATSC	804	1537	970	2	37	253
IEEE 802.16-2004 WirelessMAN SCa	965	1833	1276	2	51	240
IEEE 802.16-2004 WirelessMAN SC	1107	2101	1501	2	52	238

Note: Performance and utilization data are generated using an LFSC/M3GA25E-7F900C device with Lattice Diamond 1.0 and Synplify Pro for Lattice D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeSC/M family.

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.
- LatticeECP/EC Family Data Sheet (DS1000)
- LatticeECP2/M Family Data Sheet (DS1006)
- LatticeECP3 Family Data Sheet (DS1021)
- LatticeSC/M Family Data Sheet (DS1004)
- LatticeXP Family Data Sheet (DS1001)
- LatticeXP2 Family Data Sheet (FPGA-DS-02088)

Related Information

For more information regarding core usage and design verification, refer to the Reed-Solomon Decoder IP Core User’s Guide.

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 1.7, IP Version 3.4, April 2021

Section	Change Summary
All	• Changed document number from IPUG52 to FPGA-RD-02155. • Updated document template.
Disclaimers	Added this section.
Acronyms in This Document	Added this section.
Ordering Part Number	Corrected LatticeECP3 OPN to RSDEC-DBLK-E3-U3.
All	Minor editorial and style changes.

Revision 1.6, IP Version 3.4, December 2010

Section	Change Summary
All	Added support for Diamond software throughout.

Revision 1.5, IP Version 3.3, July 2010

Section	Change Summary
All	• Divided document into chapters. Added table of contents. • Added Quick Facts tables in the Introduction section. • Added new content in the IP Core Generation section.

Revision 1.4, IP Version 3.3, May 2009

Section	Change Summary
All	• Added support for LatticeECP3 FPGA family. • Added VHDL flow. • Added Aldec Active-HDL simulation and Linux/Solaris platform support.

Revision 1.3, IP Version 3.2, May 2007

Section	Change Summary
All	Added support for LatticeXP2 FPGA family..

Revision 1.2, IP Version 3.1, January 2007

Section	Change Summary
All	Updated LatticeECP/EC, LatticeECP2, LatticeXP and LatticeSC appendices. Added support for the LatticeECP2M FPGA family.

Revision 1.1, IP Version 3.0, August 2006

Section	Change Summary
All	Core version 3.0: Full support of IPexpress flow, including LatticeECP/EC, LatticeECP2, LatticeSC, and LatticeXP.

Revision 1.0, IP Version 2.0, March 2006

Section	Change Summary
All	Initial release.

www.latticesemi.com