
CoreRSENC v3.0 Handbook



Table of Contents

Introduction	3
Core Overview	3
Utilization and Performance	4
1 Functional Description	5
Theory of Operation	5
RS Encoder Block Diagram	7
2 Tool Flows	9
Licensing	9
SmartDesign	9
Simulation Flows	9
Synthesis in Libero IDE	10
Place-and-Route in Libero IDE	10
3 Interface Descriptions	11
Ports	11
I/O Signal Functionality	12
Configuration Parameters	13
4 Testbench Operation and Modification	15
User Testbench	15
5 List of Changes	17
A References	19
B Product Support	21
Customer Service	21
Customer Technical Support Center	21
Technical Support	21
Website	21
Contacting the Customer Technical Support Center	21
Index	23

Introduction

Core Overview

CoreRSENC is an RTL generator that produces a Microsemi SoC Products Group (formerly Actel) FPGA-optimized Reed-Solomon (RS) encoder core based on user-defined parameters.

RS codes are a class of error-correcting codes used to detect and correct errors that might be introduced into digital data when it is transmitted or stored. Error-correcting codes incorporate redundancy in data. With this redundancy, only a subset of all possible transmissions contains valid messages. This means the valid codes are separated from each other so errors are not likely to corrupt one valid code into another. The encoded data can then be transmitted or stored.

When recovering data, a decoder first determines if a received message is valid. This step is called error detection. If an error is detected, the decoder finds the valid message “closest” to the received one. Provided the number of corrupted words (symbols) does not exceed a specified range, the message found is the one that was transmitted. Thus, the decoder conducts error correction.

The number of errors the code can correct depends on the amount of redundancy added. In other words, the more errors are expected to occur, the more redundant symbols need to be added. Note that the number of redundant symbols directly impacts the complexity of the Reed-Solomon codec (encoder and especially decoder).

The RS encoder and decoder do not necessarily have to be coupled. Both encoder and decoder operate over an RS code that is entirely defined by a user via core configuration parameters. Once the same RS code parameters are defined, the encoder/decoder can communicate to a different decoder/encoder at logic level. The user may need to provide physical-level converters and minimal handshaking logic if necessary. Obviously, the RS encoder and decoder work with each other with no extra logic or converters needed.

A user configures the RS encoder via a configuration text file. The file contains all configurable parameters, and the user only needs to set desired parameter values. For a detailed description of the configuration parameters, refer to ["Configuration Parameters" on page 13](#).

An example of a digital communication system utilizing an RS codec is shown in [Figure 1](#). Data gets encoded then modulated and transmitted via a communication channel that may introduce one or more errors. At the receiver end, a demodulated message gets decoded with erroneous symbol(s) corrected. The recovered data goes to its destination.

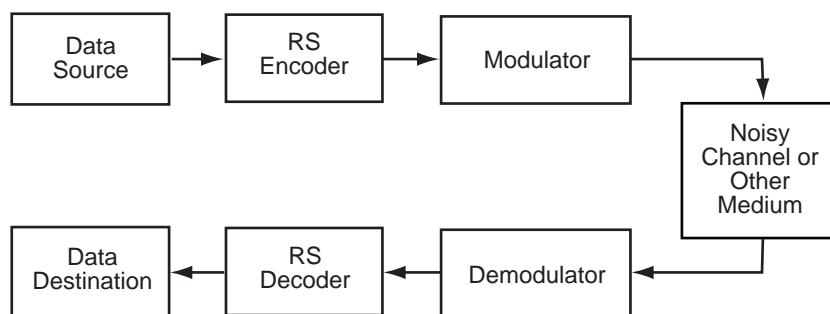


Figure 1 • An Example of a Digital Communication System

Utilization and Performance

CoreRSENC has been implemented in several Microsemi FPGA families. A summary of the utilization and performance data for CoreRSENC is given in [Table 1](#).

Table 1 • CoreRSENC Device Utilization and Performance

FPGA Family and Device	Config.	Cells or Tiles			Utilization %	Device Speed Grade	Clock Rate MHz	Throughput Mbps
		Comb.	Seq.	Total				
IGLOO [®] AGL125V5	1	338	225	563	18.30 %	STD	101	730
	2	425	321	746	24.30 %	STD	89	623
ProASIC [®] 3/E A3P250	1	342	221	563	9.2 %	-2	156	1,127
	2	451	319	770	12.5 %	-2	146	1,021
SmartFusion [™] A2F200M	1	320	202	543	11.78 %	-1	120	867
	2	399	298	718	15.58 %	-1	107	748
Fusion AFS250	1	344	221	565	9.20 %	-2	150	1,060
	2	443	319	762	12.40 %	-2	140	950
ProASIC ^{PLUS} [®] APA300	1	460	205	665	8.1 %	STD	78	564
	2	552	301	853	10.4 %	STD	82	574
Axcelerator [®] AX250	1	376	204	580	13.7 %	-2	155	1,120
	2	424	300	724	17.1 %	-2	151	1,056
RTAX-S RTAX250S	1	376	204	580	13.7 %	-1	112	784
	2	424	300	724	17.1 %	-1	99	715

Note: Data in this table were obtained using typical synthesis and layout settings.

CoreRSENC configuration parameters were set as shown in [Table 2](#). Configurations 1 and 2 are used in the ATSC (Advanced Television Systems Committee) and CCSDS (Consultative Committee for Space Data Systems) standards, respectively.

Table 2 • RS Encoder Test Configurations

Parameter		Configuration	
Name	Description	1	2
MM	Symbol width, bits	8	8
NN	Codeword length, symbols	207	255
TT	Number of corrupted symbols the RS code can correct	10	16
FR	First root of the primitive polynomial	0	112
PRIM_POLY	Primitive polynomial	285	391

1 – Functional Description

Theory of Operation

Properties of Reed-Solomon Codes

A Reed-Solomon code is a block code generally designated as $RS(n, k)$ with m -bit symbols, where k is the number of data symbols per block, n is the number of symbols the encoded message contains, and the symbol size s can be in a range from one to several bits. The encoded message called codeword has $n - k$ redundant parity symbols. The code can correct up to $t = (n - k) / 2$ symbols.

An RS code is also a systematic one, since the encoder simply appends the parity symbols to the otherwise unchanged original data sequence. Figure 1-1 shows the RS code structure.

An RS code is a linear code. In practice, this means that every possible s -bit word is a valid symbol. For instance, with 8-bit RS symbols, any 8-bit word can be transmitted directly in the data part of a codeword (Figure 1-1), so the encoder does not care what the nature of the data is, whether it is a binary stream separated into blocks of k 8-bit symbols, ASCII codes, etc. Given a symbol size m , the maximum RS codeword length is $n_{\max} = 2^s - 1$.

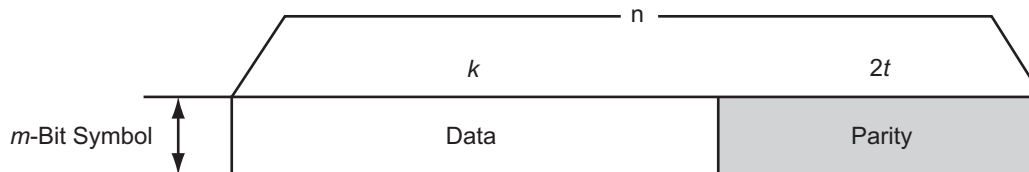


Figure 1-1 • The RS Code Structure

Consider an $RS(255, 223)$ code with 8-bit symbols utilized by many standards. Here each codeword contains 223 data bytes and 32 parity bytes, totaling a 255-byte codeword. The code is capable of correcting up to 16 corrupted symbols. Parameters of the code are as follows:

- $n = 255$
- $k = 223$
- $m = 8$
- $t = (255 - 223) / 2 = 16$

A corrupted symbol can have one or more (up to m) erroneous bits. In the above example, the RS code can correct up to 16 symbol errors when every erroneous symbol has one to eight corrupted bits. This property makes RS a powerful tool for protecting data impacted by burst errors.

Galois Field Math

RS codes are based on Galois fields (GFs), also called finite fields. Rules of the GF arithmetic are different from the usual arithmetic rules. For instance, GFs are finite fields. This means that any field element, as well as a result of the element addition and multiplication, can be presented by a fixed-length binary word. To generate and decode an RS code of s -bit symbols, an m -bit-wide Galois field is used. References 1 and 2 in "References" on page 19 provide a gentle introduction to GF math. Here only a few notes on GF are discussed—those that help configure CoreRSENC and CoreRSDec.

A Galois field used to generate an RS code is defined by RS symbol size s and a primitive polynomial. The polynomial has binary coefficients, i.e., either 0 or 1. For instance,

$$1 \cdot x^8 + 0 \cdot x^7 + 0 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x + 1$$

EQ 1-1

Depending on the size s , there might be one or more valid primitive polynomials. Different polynomials generate different GFs and thus different RS codes. Usually, particular standards—for example, 802.11—define the primitive polynomial to be used in the RS encoder/decoder. The Microsemi RS cores support any user-defined polynomial valid with any symbol size s . Polynomials are entered as decimal numbers. The bits of this number's binary image correspond to the polynomial coefficients. For example,

$$1 \cdot x^8 + 0 \cdot x^7 + 0 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x + 1 \Rightarrow 100011101 = 285$$

EQ 1-2

The core also checks the validity of the polynomial entered. If the polynomial is invalid, the core replaces it with a default primitive polynomial and issues an appropriate warning. Default polynomials are listed in [Table 1-1](#).

Table 1-1 • Default Primitive Polynomials

Symbol Size, s	Default Polynomial	Decimal Form
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

There is another important polynomial an RS codec directly utilizes: a Generator polynomial. This is derived from the primitive polynomial based on the first root of the Generator polynomial. Again, particular standards often define the first root value to be used in the RS codec. Most common first root values are 0 or 1, but the Microsemi RS encoder supports any value in the range from 0 to $n - 1$.

Shortened Codes

A shortened codeword contains fewer symbols than the maximum $n_{\max} = 2^m - 1$. The shortened codeword keeps the same number of parity symbols, $2t$, to correct up to t errors. Therefore, the number of data symbols in the shortened code is reduced by the same amount as the overall codeword length. For instance, RS(204, 188) is the shortened code of RS(255, 239). Both codes have a symbol width of 8 and use the same number of parity symbols, 16.

Conceptually, shortening a codeword is done by assuming initial extra data symbols of the maximum-length codeword are set to 0. Though efficiency of the shortened code is less than that of the maximum-length code, some standards require the RS codec to use it.

RS Encoder Block Diagram

Figure 1-2 shows a block diagram of the RS(n, k) encoder. For k clock cycles, it accepts m -bit input symbols. Since the RS code is systematic, the unaltered input data pass through the encoder via an output multiplexer. Once the k data symbols enter the encoder, the feedback shift registers reg0 – reg($n - k - 1$) finish calculating the parity symbols. These are appended to the data symbols with the help of the output multiplexer to form a complete codeword.

The feedback shift register generates parity symbols in such a way as to make the complete codeword a whole multiple of the RS Generator polynomial. At the decoder end, the codeword is divided by the Generator polynomial. If the remainder is zero, no errors are detected. Otherwise, there are errors in the received codeword. All calculations are performed using finite field arithmetic.

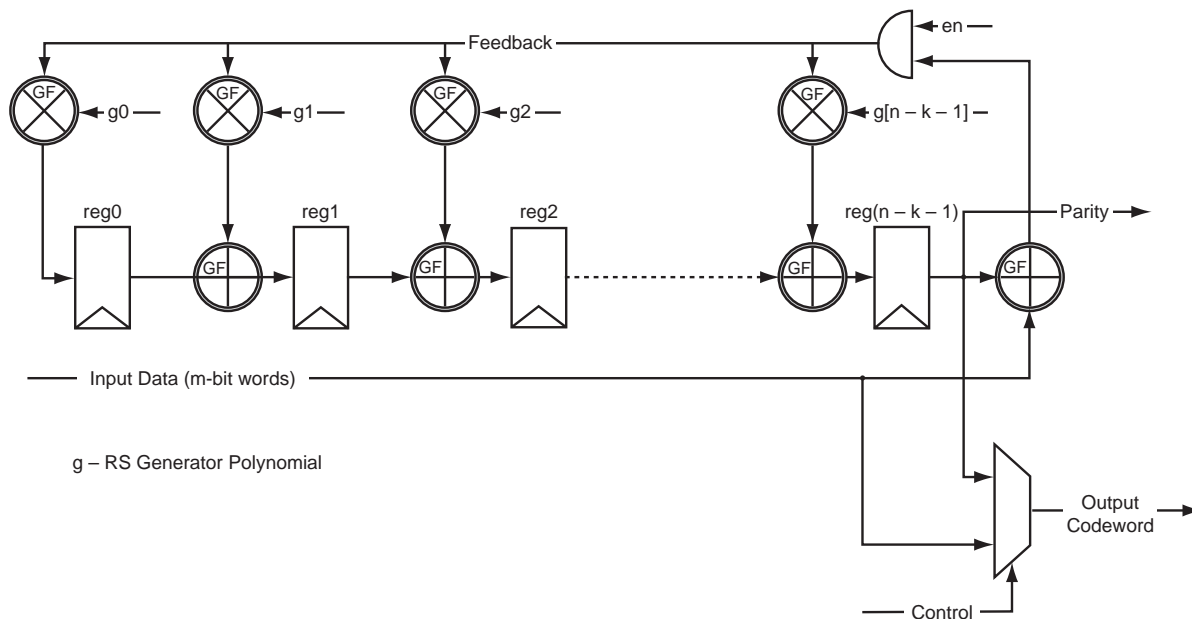


Figure 1-2 • RS Encoder Block Diagram

2 – Tool Flows

Licensing

CoreRSENC is provided with a full RTL license. Complete RTL source code is provided for the core and testbenches.

SmartDesign

CoreRSENC is preinstalled in the SmartDesign IP deployment design environment. The core can be configured using the configuration GUI within SmartDesign, as shown in [Figure 2-1](#).

For more information on using SmartDesign to instantiate and generate cores, refer to the [Using DirectCore in Libero® IDE User's Guide](#).

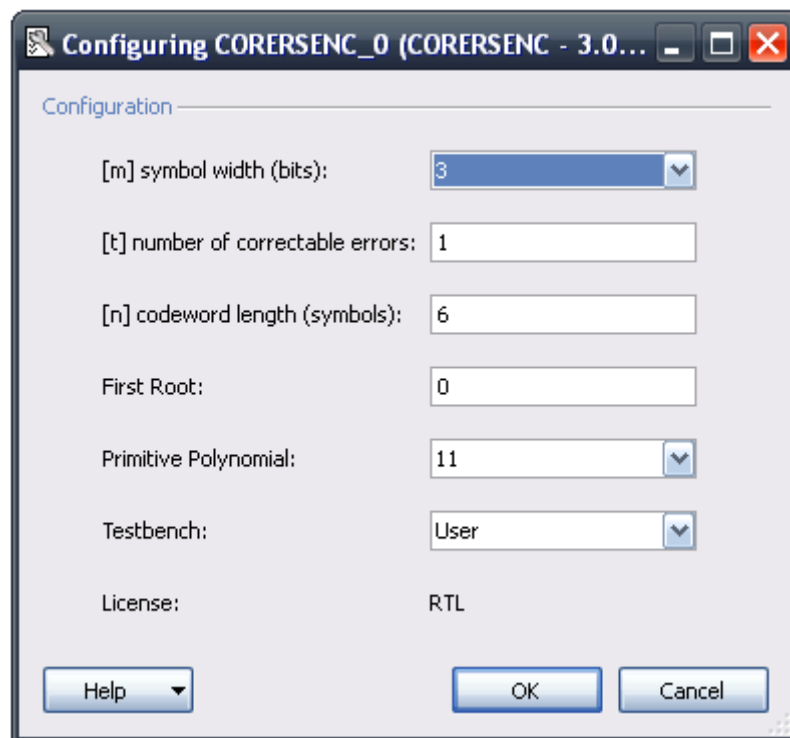


Figure 2-1 • SmartDesign CoreRSENC Configuration Window

Simulation Flows

The user testbench for CoreRSENC is included in this release.

To run simulations, select the user testbench flow within SmartDesign and click **Save & Generate** on the Generate pane. The user testbench is selected through the Configuration GUI shown in [Figure 2-1](#).

When SmartDesign generates the Libero IDE project, it will deploy the user testbench files.

To run the user testbench, set the design root to the CoreRSENC instantiation in the Libero IDE Design Hierarchy pane and click the **Simulation** icon in the Libero IDE Design Flow window. This will invoke ModelSim® and automatically run the simulation.

Synthesis in Libero IDE

Click the **Synthesis** icon in Libero IDE. The Synthesis window appears, displaying the Synplicity® project. Set Synplicity to use the Verilog 2001 standard if Verilog is being used. To run Synthesis, select the **Run** icon.

Place-and-Route in Libero IDE

Click the **Layout** icon in Libero IDE to invoke Designer. CoreRSENC requires no special place-and-route settings.

3 – Interface Descriptions

Ports

The port signals for CoreRSENC are defined in [Table 3-1](#) and illustrated in [Figure 3-1](#).

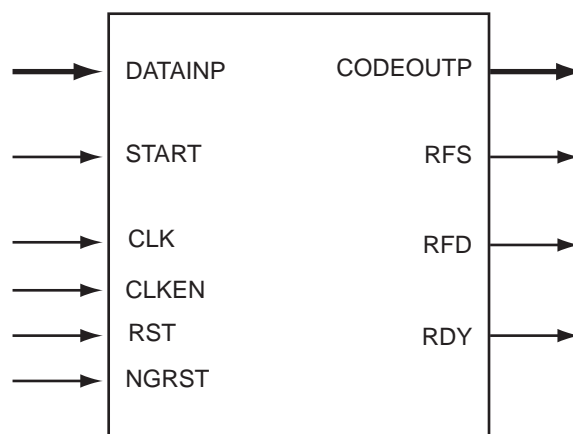


Figure 3-1 • CoreRSENC I/O Signals

Table 3-1 • I/O Signal Descriptions

Signal	Direction	Description
DATAINP[MM-1:0]	Input	Data input to be encoded. The input bus is m bits wide.
START	Input	This signal starts a new codeword cycle. It informs the encoder that at the next clock interval the first m -bit data symbol of a k -symbol sequence will appear at DATAINP.
CLK	Input	Encoder clock signal
CLKEN	Input	Encoder clock enable signal
RST	Input	Synchronous reset
NGRST	Input	Asynchronous reset; active low
CODEOUTP[MM-1:0]	Output	Encoded data (codeword) output. The output is m bits wide.
RFS	Output	Ready For Start. This signal is active when the encoder is ready to accept a new START signal.
RFD	Output	Ready For Data. This signal is active when the encoder is ready to accept a new data portion of k symbols on DATAINP.
RDY	Output	(Output RS Code) Ready. This signals that a valid codeword is present at the core output.

I/O Signal Functionality

NGRST, RST Inputs

Both reset signals reset all registers of the RS encoder to bring it to an initial state. In the initial state, the feedback registers $\text{reg}0 - \text{reg}(n - k - 1)$ (Figure 1-2 on page 7) are set to zero, signals RFS and RFD are active, and the RDY signal is inactive. The RS encoder is ready to accept fresh input data. NGRST is an asynchronous signal (active low), and RST (active high) is synchronous to rising edge of the clock signal. One or both must be used to prevent the very first codeword from being incorrect. If this is not a problem, the reset signals are optional.

CLKEN Input

When CLKEN is inactive (Low), the core is frozen. All inputs except NGRST are ignored, and the core retains its current state.

START Input

This signal starts a new codeword cycle. It informs the encoder that at the next clock interval the first m -bit data symbol DATAINP_0 of a k -symbol sequence will appear at the DATAINP bus (Figure 3-2). It is assumed that the CLKEN signal is active in Figure 3-2.

The START signal can be set at any time to launch another codeword generation. Normally, though, a data source is supposed to issue START once the RFS signal goes High. If the data source wishes to issue a START before the completion of the current codeword, it must first issue a reset by either asserting the RST signal or deasserting the NGRST signal. Otherwise, the subsequent codeword will be corrupted and must be discarded.

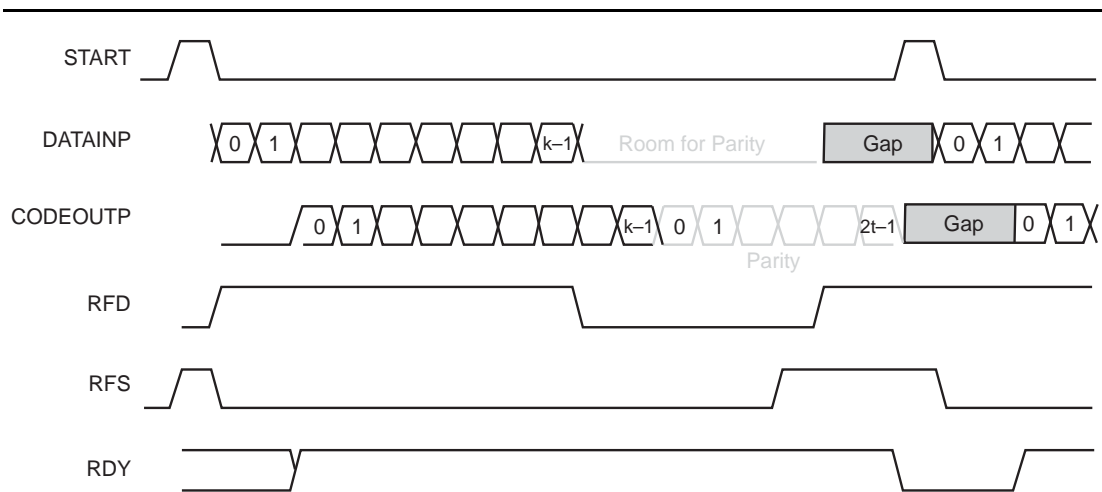


Figure 3-2 • RS Encoder timing

It can be seen in Figure 3-2, that START can be asserted as early as to repeat the RFS signal, or can otherwise be delayed arbitrarily with regard to a positive edge of RFS. Figure 3-2 shows an example of a three-clock-cycle delay.

RFS Output

The core asserts this output when it is ready to accept another START signal. With normal RS encoder functionality, the data source should wait for RFS to go High to issue the START signal. Logically, RFS signifies completion of the current codeword generation, but it gets asserted three clock cycles prior to actual codeword termination (Figure 3-2) due to the RS encoder latency.

RFD Output

This optional output signal is asserted when the RS encoder core is ready for fresh input data. Once the core fetches k symbols of data, RFD goes LOW, thus blocking input data when the core generates redundant parity symbols (Figure 3-2).

RDY Output

The optional RDY signal marks an interval of time when a codeword is present at the RS encoder output CODEOUTP (Figure 3-2 on page 12).

Configuration Parameters

CoreRSENC generates the RS encoder engine RTL code based on parameters set by the user. CoreRSENC supports the variations specified in Table 3-2.

Table 3-2 • CoreRSENC Configuration Parameters

Name	Valid Values	Description
MM	3 to 8	Symbol width, bits (m)
NN	5 to $2^m - 1$	Codeword length, symbols (n)
TT	1 to 16 as long as $t < n / 2 - 1$	Number of correctable errors (t)
PRIM_POLY	Arbitrary valid polynomial selectable from a drop-down menu	Primitive polynomial identifying Galois field
FR	0 to $n-1$	First root of the primitive polynomial
FPGA Family	IGLOO, IGLOOPLUS, IGLOOe, ProASIC3/E/L, SmartFusion, Fusion, ProASICPLUS, Axcelerator, RTAX-S	Identifies family of FPGA devices

4 – Testbench Operation and Modification

User Testbench

Included with the releases of CoreRSENC is a user testbench that verifies operation of the CoreRSENC engine. A simplified block diagram of the user testbench is shown in [Figure 4-1](#). The user testbench instantiates the RS encoder engine configured by the user, as well as behavioral, non-synthesizable models of an input test vector generator, a golden codeword generator, a comparator, and a signal generator that provides necessary clock, reset, and other signals. The testbench compares the actual RS encoder output codeword and the golden codeword vector. CoreRSENC automatically generates Verilog or VHDL testbench behavioral code based on the user selection of the core language.

The same testbench can be used for pre-synthesis and post-synthesis simulation. A simulation tool displays the verification result.

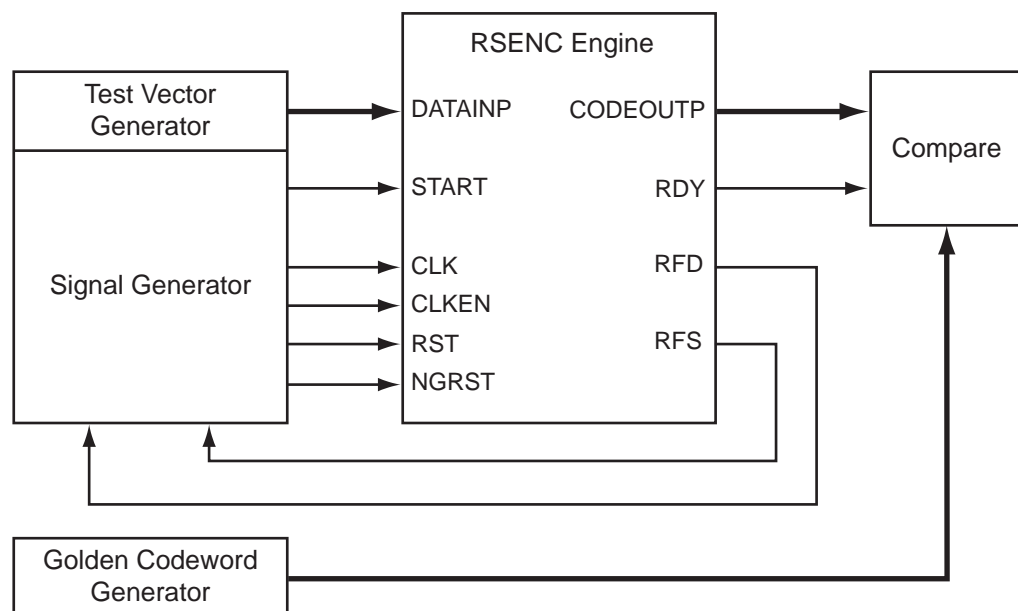


Figure 4-1 • CoreRSENC User Testbench

5 – List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Revision	Changes	Page
Revision 1 (February 2011)	The core version was updated from v2.0 to v3.0. SmartFusion numbers were added to Table 1 • CoreRSENC Device Utilization and Performance .	4
	The parameter names were changed in Table 2 • RS Encoder Test Configurations .	4
	The <i>s</i> -bit was changed to <i>M</i> -bit in Figure 1-1 • The RS Code Structure and the accompanying text.	5
	The "Tool Flows" section was replaced.	9
	[MM-1:0] was added to the signal names DATAINP and CODEOUTP in Table 3-1 • I/O Signal Descriptions .	11
	Port names in Figure 3-1 • CoreRSENC I/O Signals and Figure 3-2 • RS Encoder timing were changed from lower case to upper case. Occurrences of these port names throughout the document were revised accordingly.	11
	The "START Input" section was revised to explain necessary steps if the START signal will be initiated before completion of the current codeword.	12
	The names and descriptions of configuration parameters were revised in Table 3-2 • CoreRSENC Configuration Parameters . IGLOO PLUS and SmartFusion were added as valid FPGA families and ProASIC3/E was changed to ProASIC3/E/L. The HDL selection parameter was removed.	13
The terminology "verification testbench" was changed to "user testbench" in the "Testbench Operation and Modification" chapter.	15	

A – References

Rorabaugh, C. Britton. *Error Coding Cookbook*. McGraw-Hill, 1995.

Sweeney, Peter. *Error Control Coding*. John Wiley & Sons, 2002.

B – Product Support

Microsemi backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group (formerly Actel) and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650.318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650.318.8044**

Customer Technical Support Center

Microsemi staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.actel.com/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 a.m. to 6:00 p.m., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 a.m. to 6:00 p.m., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460

800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found on the website at www.actel.com/company/contact/default.aspx.

Index

C

configuration parameters 13
contacting Microsemi
 customer service 21
 electronic mail 21
 telephone 22
 web-based technical support 21
customer service 21

E

encoder block diagram 7

F

finite field math 5

G

Galois field math 5

I

I/O signals
 descriptions 11
 functionality 12
interface descriptions 11

M

Microsemi
 electronic mail 21
 telephone 22
 web-based technical support 21
 website 21

O

operation, theory of 5
overview 3

P

ports 11
primitive polynomials, default 6
product support 22
 customer service 21
 electronic mail 21
 technical support 21
 telephone 22
 website 21

R

Reed-Solomon codes
 defined 3
 encoder block diagram 7

 properties of 5
 shortened 6
 use in a digital communication system 3
references 19

S

shortened codes 6

T

technical support 21
test configurations 4
testbenches 15
 verification 15
theory of operation 5

U

utilization and performance 4

V

verification testbench 15

W

web-based technical support 21



Microsemi Corporate Headquarters
2381 Morse Avenue, Irvine, CA 92614
Phone: 949-221-7100 · Fax: 949-756-0308
www.microsemi.com

Microsemi Corporation (NASDAQ: MSCC) offers the industry's most comprehensive portfolio of semiconductor technology. Committed to solving the most critical system challenges, Microsemi's products include high-performance, high-reliability analog and RF devices, mixed signal integrated circuits, FPGAs and customizable SoCs, and complete subsystems. Microsemi serves leading system manufacturers around the world in the defense, security, aerospace, enterprise, commercial, and industrial markets. Learn more at www.microsemi.com.

© 2011 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.