Reed-Solomon Compiler MegaCore Function

Solution Brief 48 **September 2000, ver. 1.0**

Target Applications:
Wireless Communications, Satellite Communications

Family:
APEX™ 20K, ACEX™ 1K, FLEX® 10K, FLEX 8000, and FLEX 6000

Ordering Codes:
PLSM-RSENC
PLSM-RSDEC
PLSM-HC-RSENC/V
PLSM-HC-RSDEC/C
PLSM-HC-RSDEC/ERAS
PLSM-HC-RSDEC/V

Vendor:

101 Innovation Drive
San Jose, CA 95134
http://www.altera.com
Tel. (408) 544-7000

Features

- High-performance encoder/decoder for error detection and correction
- Fully parameterized Reed-Solomon (RS) function including:
  - Number of bits per symbol
  - Number of information symbols and check symbols per codeword
  - Field polynomial
  - First root of generator polynomial
  - Space between roots in generator polynomial
- MegaWizard® Plug-In Manager for easy parameterization
  - Generates parameterized encoder or decoder
  - Generates example test vectors
- VHDL simulation model
- Flexible decoder implementation
  - Discrete, streaming, or continuous architectures
  - Erasure-supporting option
  - Variable option
- Optimized for the APEX™ 20K, ACEX™ 1K, FLEX® 10K, FLEX 8000, and FLEX 6000 device architectures
- Efficient VHDL simulation model
- Flexible licensing: purchase only the required features

General Description

Reed-Solomon (RS) codes are used for detecting and correcting errors in data transmitted over a communications channel. RS codes break a data stream into a series of codewords comprised of several information symbols and check symbols (also known as parity bits). The Altera® Reed-Solomon compiler MegaCore® function supports four to ten bits per symbol. The encoder adds parity symbols to the data stream prior to its transmission over a communications channel. The decoder receives the data and checks for and corrects any errors. Errors are defined on a symbol basis (i.e., any number of bit errors within a symbol is considered as only one error). An RS decoder can correct one symbol error for every two check symbols in a codeword.

Functional Description

Table 1 shows the parameters for the Reed-Solomon compiler MegaCore function. The parameters define the specific RS code for the encoder or decoder, and are specified using the MegaWizard Plug-In Manager.

Table 1. Reed-Solomon Compiler MegaCore Function Parameters

Parameter	Range	Description
type_of_encoder	Standard or variable	This parameter specifies a standard encoder or variable encoder.
decoder_options	Erasure-supporting option and/or variable option	This parameter specifies the erasure-supporting option and/or variable option for the decoder. Erasure-supporting option increases the logic resources used.
architecture *(1)*	Discrete, streaming, or continuous	This parameter specifies discrete, streaming, or continuous decoder architecture.
keysize *(1), (2)*	Half or full	There is a trade-off between the amount of logic and the number of cycles used to determine the location and number of errors. A full design creates a function that uses more logic, but fewer clock cycles to process the codeword. A half design creates a function that uses less logic, but requires more cycles to process a codeword.

Notes:
(1) This parameter applies to the decoder only.
(2) This parameter was called speed in earlier versions.

Encoder Signals

To begin the encoding process, a high pulse of at least one clock cycle must be applied to start. When enable is asserted, data is entered at rsin[] and encoded at the rising clock edge (sysclk). The function outputs the first codeword symbol on rsout[] after the clock edge that clocked in the first information symbol. Check symbols are output after the information symbols, during the R clock cycles.

The enable signal allows you to suspend the continuous operation of the encoder. If enable is de-asserted, the encoder stops processing data; if enable is asserted, the encoder restarts processing data. Table 2 describes the signals used by the RS encoder.

Table 2. Encoder Signals

Name	Type	Description
sysclk	Input	System clock
reset	Input	Resets the encoder asynchronously
start	Input	Sets the encoder to process a new input codeword
enable	Input	Allows data into the encoder and onto the output data bus
rsin[]	Input	m-bit wide input data bus
rsout[]	Output	m-bit wide output data bus
numn[] *(1)*	Input	Variable value of N. Can be any value from the minimum allowable value of N up to the selected value of N.
numcheck *(1)*	Input	Variable number of check symbols. Can be any value from the minimum allowable value of R up to the selected value of R.

Note:
(1) This signal is used only when the variable option is selected.

Decoding Data

A discrete decoder processes one codeword at a time and must be reset between each codeword. The decoder receives the codeword's first symbol on the rising clock edge of `sysclk` after `reset` is de-asserted. The `rdyn` signal indicates a decoder is ready to accept a new codeword and is asserted until the decoder receives the last codeword. When `dsout` is asserted, the discrete decoder outputs one symbol on each rising clock edge until all data is transferred. If the discrete decoder detects more errors than it can correct, it presents the incorrect data on the signal `rsout[]`.

The streaming decoder interface is similar to the discrete decoder, however, the streaming decoder is only reset once. If `reset` is asserted while codewords are being decoded, the codewords are lost. The `rdyn` signal indicates when the streaming decoder can accept a new codeword; when `rdyn` and `dsin` are high, the decoder expects a new symbol at each rising clock edge. The streaming decoder has a pipeline depth of three codewords and therefore must receive three codewords before it places the first corrected codeword at `rsout[]`. If the streaming decoder detects more errors than it can correct, it presents the incorrect data on the signal `rsout[]`.

The continuous decoder also has a pipeline depth of three codewords but does not have idle gaps in between codewords; the codewords enter and are placed at the output continuously. The continuous decoder is suspended by de-asserting `ce` and applying `reset` for at least one clock cycle. When `ce` is re-asserted, the first symbol can be accepted by the decoder. If the continuous decoder detects more errors than it can correct, it presents the incorrect data on the signal `rsout[]`. There is no variable option for the continuous decoder.

Performance

Tables 3 and 4 show the standard and erasure-supporting function's performance and area utilization for FLEX 10KE devices. Overall resource requirements vary widely depending on the parameter values selected. The number of logic cells required to implement the function depends on both the field size and the number of check symbols. EAB (embedded array block) and ESB (embedded system block) usage varies between discrete and streaming decoders.

Table 3. Performance & Area Utilization in a FLEX 10KE Device—Standard

Architecture	Keysize	*m*	*N*	*R*	Utilization		Performance	
					Logic Elements	EABs/ESBs	Mbps	f_{MAX} (MHz)
streaming	full	4	15	4	516	6	150	82
streaming	full	5	31	6	772	6	259	78
streaming	half	6	55	6	906	6	381	74
streaming	half	8	204	16	2,431	6	455	61
continuous	half	8	204	16	2,118	7	488	61
discrete	half	8	204	16	2,194	3	115	55
streaming	half	8	207	20	3,439	6	388	53
streaming	half	8	225	46	6,603	6	178	45
continuous *(1)*	full	10	1,023	8	1,240	36	921	92

Table 4. Performance & Area Utilization in a FLEX 10KE Device—Erasure-Supporting								
Architecture	Keysize	m	N	R	Utilization		Performance	
					Logic Elements	EABs/ESBs	Mbps	f_{MAX} (MHz)
streaming	full	4	15	4	858	6	129	81
streaming	full	5	31	6	1,355	6	227	79
streaming	half	6	55	6	1,562	6	381	76
streaming	half	8	204	16	4,853	6	441	59
continuous	half	8	204	16	3,913	7	423	93
discrete	half	8	204	16	4,190	3	118	57
streaming	half	8	207	20	6,312	6	383	52
streaming *(1)*	half	8	225	45	13,101	6	149	37

Note to tables:
(1) Results are for architectures implemented in an APEX 20KE device.

Tables 5 and 6 show the function's performance and area utilization for the variable and erasure-supporting options implemented in a FLEX 20KE device.

Table 5. Performance & Area Utilization—Variable								
Architecture	Keysize	m	N	R	Utilization		Performance	
					Logic Elements	EABs/ESBs	Mbps	f_{MAX} (MHz)
streaming	full	8	255	40	7,288	6	480	69
discrete	half	5	31	6	1,355	6	227	79
streaming	half	6	55	6	1,562	6	381	76
streaming	full	8	255	40	13,865	7	321	37

Table 6. Performance & Area Utilization—Erasure-supporting & Variable								
Architecture	Keysize	m	N	R	Utilization		Performance	
					Logic Elements	EABs/ESBs	Mbps	f_{MAX} (MHz)
streaming	full	8	255	40	13,865	7	321	37

Generating a Custom RS Megafunction

Altera provides a MegaWizard Plug-In with the Reed-Solomon compiler MegaCore function. You can use the MegaWizard Plug-In Manager within the MAX+PLUS® II or Quartus™ software, or as a standalone application to create and integrate custom megafunctions without changing your design's source code. You can then simulate your design to verify compatibility and instantiate the custom megafunction in your design file.

To generate a custom RS megafunction, start the MegaWizard Plug-In Manager by choosing the **MegaWizard Plug-In Manager** command (File menu) in the MAX+PLUS II or Quartus software, or type the command `megawiz` at a command or UNIX prompt. Specify that you wish to create a new custom megafunction, and select **RS Compiler** in the **Communications** folder. Select a language for the output files, select either a decoder or an encoder, and specify whether you want the erasure-supporting (decoder only) or variable options. If you are creating a decoder, specify a discrete, streaming, or continuous architecture, and whether you wish to use a full- or half-size design.

Altera provides VHDL models that you can use to simulate the functionality of the RS compiler MegaCore function in your system. Users who have licensed the MegaCore function can also generate VHDL Output Files (**.vho**) or Verilog Output Files (**.vo**) for simulation in third-party simulators.

To learn more about Altera's RS compiler, refer to the *Reed-Solomon Compiler MegaCore Function User Guide.*

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com