XAPP952 (v1.0) December 5, 2007

Forward Error Correction on ITU-G.709 Networks using Reed-Solomon Solutions

Author: Michael Francis

Summary

The ITU-G.709, Interface for the Optical Transport Network (OTN) standard [Ref 1] describes the Forward Error Correction (FEC) requirement for an optical transport network. The purpose of FEC is to improve the quality of service on the network and ensure that the data is received without errors. In this application note, the error correction section of the ITU-G.709 standard is examined and implemented in both the Virtex™-4 and Virtex-5 Platform FPGA families using the LogiCORE™ Reed-Solomon (RS) Encoder and Decoder cores. The application note also discusses an example of an enhanced FEC scheme.

Introduction

The requirement for data traffic bandwidth continues to grow. To meet this increased demand higher data rates are needed, leading to optical networks. However, the mature networks, like SONET/SDH, still need to be supported, and customers seek higher performance with error-free transmission and, hence, better quality of service.

Within a Metropolitan Area Network (MAN), optical links may have to go over long distances. As the distance increases, the signals start to attenuate and degrade, resulting in loss of data, or data errors. The attenuation and degradation can be due to factors such as:

- **Chromatic Dispersion** - broadening of the input signal as it travels down the length of the fiber.
- **Four Wave Mixing** - when two or more frequencies (or, equivalently, wavelengths) of light propagate together through an optical fiber. Light at a new frequency is generated using optical power from the original frequencies.

As rates are increased, many of these degradation factors become more pronounced.

To overcome these factors, issue repeaters are used on the optical link. These repeaters are placed along the optical link so they can take an attenuated signal, amplify it, and retransmit it onto the next repeater. The disadvantage is the cost associated with the repeater. However, the use of FEC allows the following three advantages:

1. Extending the distance at which repeaters can be installed.
2. Increasing the rate at which data can be transmitted.
3. Improving the Quality of Service (QOS) indication.

The ITU-G.709, Interface for the Optical Transport Network (OTN), has recommended using error correction. The recommendation uses the Reed-Solomon algorithm for the FEC.

ITU-G.709 Overview

The ITU-G.709 recommendation has four layers of hierarchy as shown in Figure 1.

Data, such as SONET/SDH or Ethernet, are encapsulated in an Optical Payload Unit (OPU) consisting of a payload and overhead area. The Optical channel Data Unit (ODU) comprises the OPU data area and the OPU overhead. The ODU data and the FEC make up the Optical channel Transport Unit (OTU) *k* frame, where *k* is 1, 2, or 3, and represents one of three line rates. The OCh is the optical interface, and the overhead section is generally for operational and maintenance information.

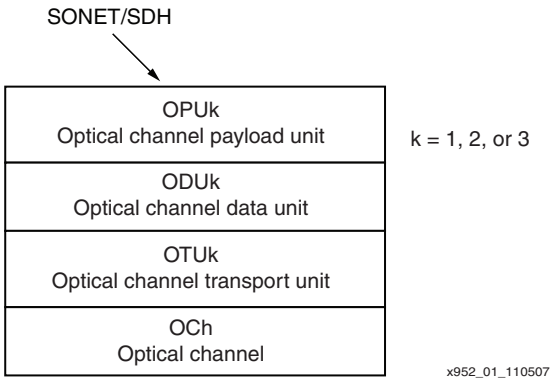

Figure 1: **OTN Hierarchy**

The overhead, due to the error protection, is approximately 7%. The ITU-G.709 recommendation outlines the lines rates shown in Table 1.

Table 1: **OTU Rates**

G.709 Interface	Line Rate (kbps)
OTU-1	2 666 057.143
OTU-2	10 709 225.316
OTU-3	43 018 413.559

For each line rate, the OTU frame is formatted in the same way and is shown in Figure 2. This is organized as four rows each of 4080 bytes comprising the Operation Administration and Maintenance (OAM), ODU payload, and the FEC bytes.

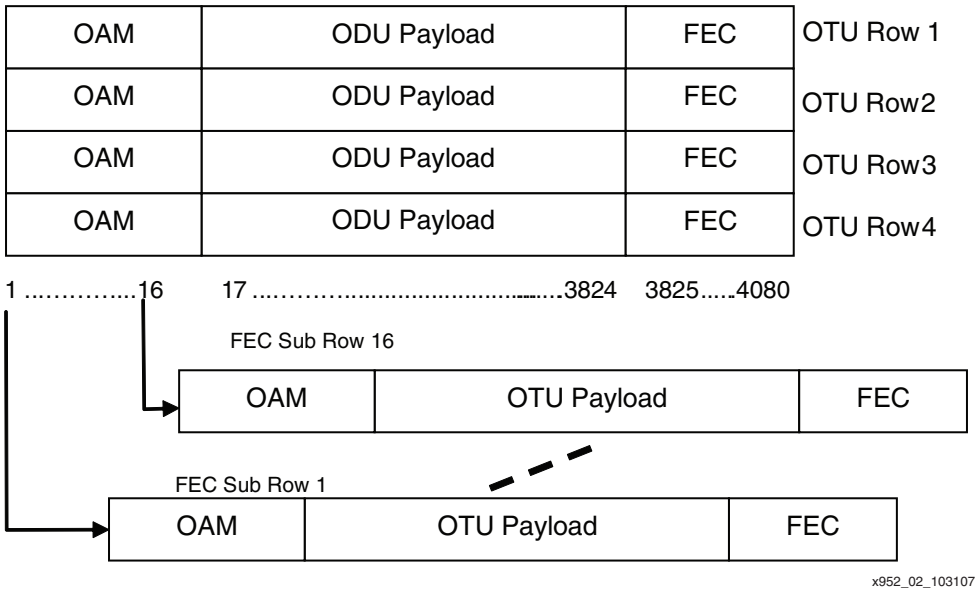

Figure 2: **OTU Frame**

Each OTU row is split into sixteen sub rows using byte interleaving. For example, the first sub row takes bytes 1, 17, 33, etc., while the second sub row consists of the 2nd, 17th, 34th bytes, etc. Each sub row is processed by a Reed-Solomon algorithm. The FEC check bytes are calculated over the information bytes 1 to 239 of each sub row and are transmitted in bytes 240 to 255 of the same sub row.

Reed-Solomon FEC

Reed-Solomon codes are linear block codes that can detect and correct burst errors. These are referred to as (*n*, *k*) with symbol width *s*-bit wide, as shown in Figure 3. An encoder takes *k* data, or information, symbols of s bits each and appends parity or check symbols to make an *n* symbol code word. There are *n* - *k* parity symbols of *s* bits each. The RS decoder processes each block and attempts to correct errors and recover the original data. The decoder can correct up to *t* symbols that contain errors in the code word, where $2t = n - k$. The number of errors detected can be used to indicate the status of the link.

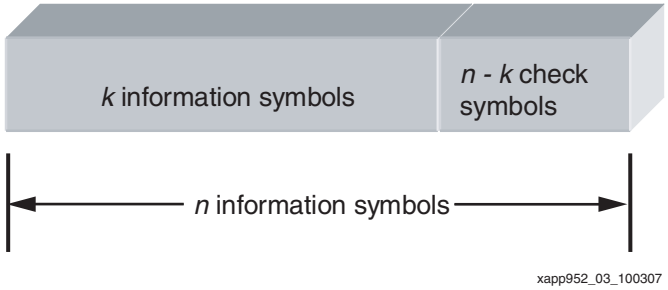

Figure 3: **Reed-Solomon Code Word**

A Brief Summary of Reed-Solomon Terminology

- *Symbol_Width s* is the number of bits per symbol.
- *Code word* is the block of *n* symbols.
- *RS (n, k)* code:
  - *n* is the total number of symbols per code word.
  - *k* is the number of information symbols per code word.
- *Code Rate* is equal to *k / n*
  - $r = (n – k)$ is the number of check symbols.
  - $t = (n – k) / 2$ is the maximum number of symbols with errors that can be corrected.

Reed-Solomon Encoders and Decoders

Reed-Solomon encoders and decoders are licensed cores available via the LogiCORE IP library. The full feature list is as follows:

- Implements many different RS coding standards
- Is a fully synchronous design using a single clock
- Supports continuous input data with no gap between code blocks
- Has symbol sizes from 3 bits to 12 bits
- Has a code block length variable that is up to 4095 symbols
- Has a code block length that can be dynamically varied on a block-by-block basis
- Supports shortened codes
- Supports error and erasure decoding
- Supports puncturing (as in IEEE 802.16d standard)
- Supports multiple channels
- Corrects parameterizable number of errors
- Supports any primitive field polynomial for a given symbol size
- Counts number of errors corrected and flags failures
- Has user-selectable control signal behavior

Further details on RS encoder and decoder architecture can be found in [Ref 2], [Ref 3], and [Ref 4].

ITU-G.709 Forward Error Correction

The ITU-G.709 FEC uses an RS (255,239) code. The size of the symbol is 8 bits, and the size of the block, or code word, is 255 bytes. There are 239 information symbols per block. 16 check symbols are appended to the 239 data symbols, allowing 8 symbol error corrections.

The FEC processing separates the OTU k (k =1, 2 or 3) row into 16 interleaved data streams, each consisting of 255 symbols. The interleaving of the data (see Figure 1) assists the RS decoder in handling longer noise bursts and improves the error-correcting capability within a frame.

For implementation of the ITU-G.709 FEC solution, only a basic parameterization of the core is required. The parameters for the encoder and decoder are entered in an easy-to-use GUI shown in Figure 4 and Figure 5.

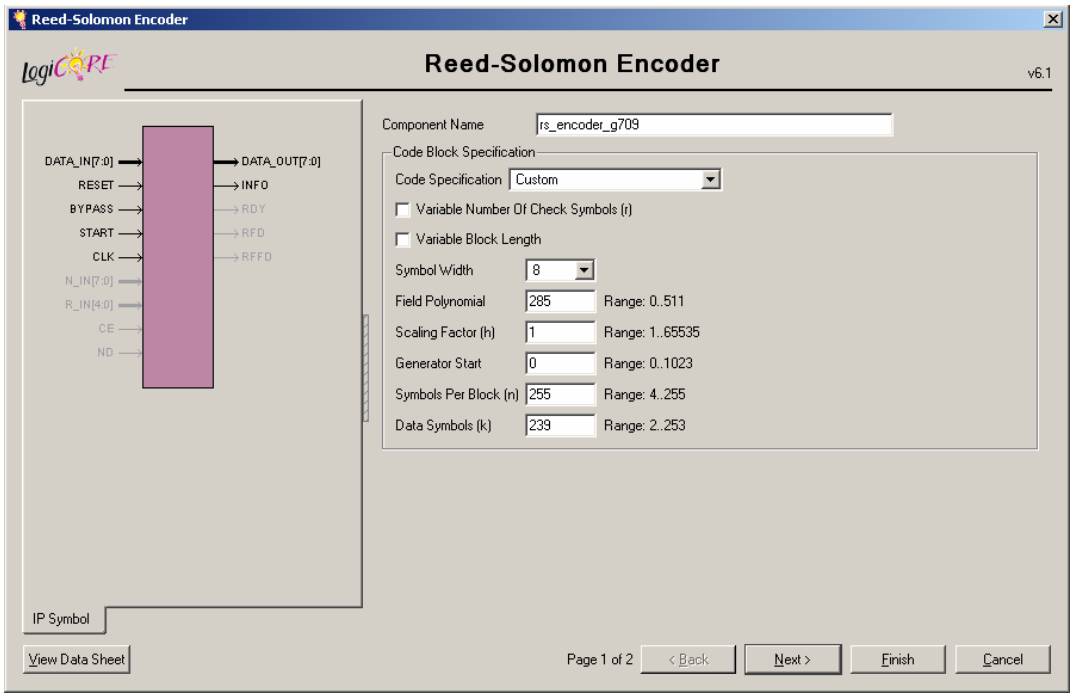

Figure 4: **Reed-Solomon Encoder GUI**

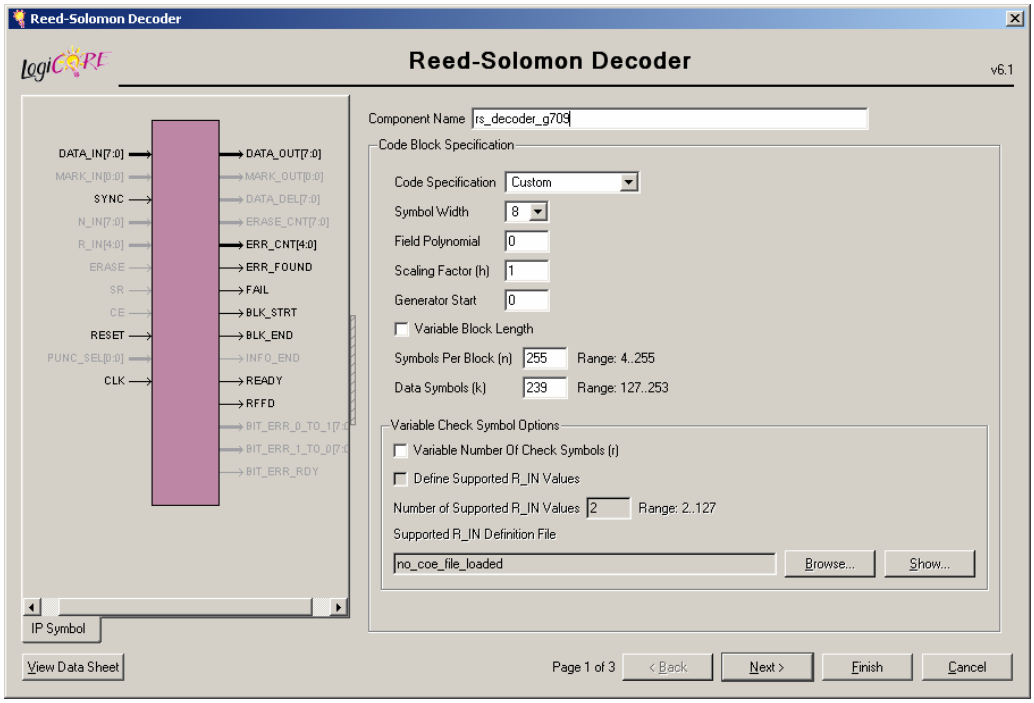

Figure 5: **Reed-Solomon Decoder GUI**

Design Overview

A complete set of design files that allows the user to simulate and, in some cases, run an FEC system with RS encoders/decoders on hardware is associated with this application note. A brief description of the files is contained in the "Reference Design Files" section.

The outline of the FEC simulation system is outlined in Figure 6.

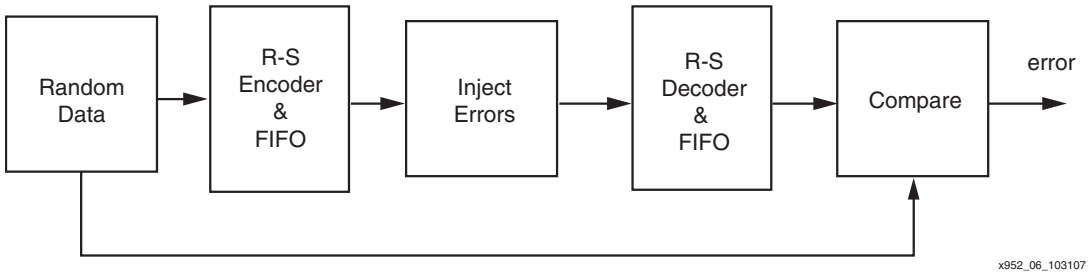

Figure 6: G.709 FEC System

Virtex-4 and Virtex-5 FPGA Implementation

Virtex-4 FPGAs allow the G.709 FEC processing to be implemented in either LX (logic), FX (full feature), or SX (signal processing) families. However, for the OTU-3 interface, the LX family has more slices available for implementation.

When looking at the maximum frequency (F_{max}) of the RS encoder/decoder, the decoder determines the maximum data rate that can be handled. The size of the encoder tends to be smaller than the decoder and has a higher performance.

The line rate of the interface and the F_{max} that can be obtained for the particular family/speed grade determines the number of RS decoders that are required. In the following case studies, each of the line rates are considered and implementation solutions are looked at.

The OTU-1 and OTU-3 interfaces shown target the Virtex-4 FPGA -11 speed grades, while the OTU-2 interface uses a -10 speed grade due to its higher performance at two-channels.

For Virtex-5 FPGAs, the higher performance allows savings to be made both in terms of area and speed grade. The design can be implemented in any of the Virtex-5 FPGA families.

OTU-1 Interface

For the OTU-1 interface, the line rate is 2,666,057.143 kilobit per second (kbps). This equates to 333,257,142.875 megabytes per second (MBps), the rate at which the RS encoders and decoders operate and which exceeds the RS encoders/decoders performance. Therefore, to process that amount of data two encoders and decoders are required to each operate at 167 MHz, and taking jitter into account a performance of 170 MHz could be required. Note that in a Virtex-5 FPGA mid-speed grade (-2), only a single encoder is needed, thereby saving area.

The data from the random data generator is organized as 16 byte-wide FEC sub rows, totalling 128 bits. Two sets of 8-1 multiplexers create 16 bits at 156 MHz which is written into the 512 x 16 LogiCORE FIFO Generator at 156 MHz. The FIFO has been set up to use two most significant bits from the FIFO read count, so that reading can be started when the FIFO is half-full. See Figure 7.

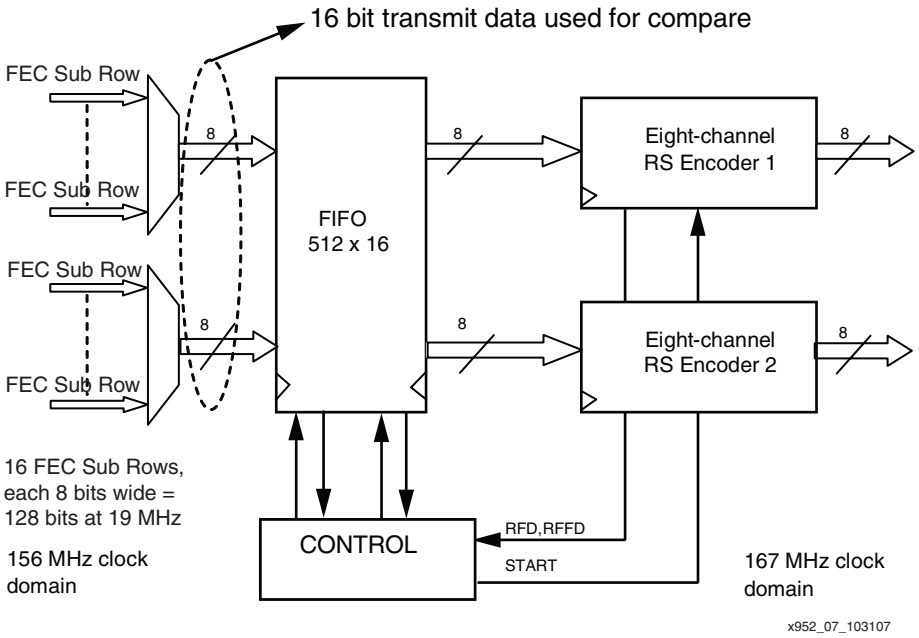

Figure 7: **OTU-1 Encoding**

Encoder

On the read port of the FIFO, the 16-bit data is read out at higher rate of 167 MHz. This is 156 MHz x (255/239). The 16-bit bus is split into two 8-bit data buses. Each 8-bit data bus goes to the DATA_IN port of the specific encoder, and the data is encoded. The RS encoder has been configured via the GUI with the following parameters:

- Data Symbols (k) = 239
- Symbols per Block (n) = 255
- Symbol Width = 8
- Code Specification = Custom
- Number of channels = 8
- Select RDY, RFD and RFFD. See [Ref 2] for definitions of the signals.

The Ready For Data (RFD) signal can be used as the read signal for the FIFO. All other parameters can be left at the default values. Each RS encoder has been set up for eight

channels as the number of sub rows that have been multiplexed together is eight. The *ITU-G.709 Recommendation* gives the field polynomial as x^8+.. x^4+.. x^3+.. x^2+.1. This is defined in the Field Polynomial area of the GUI as 285 dec, or 100011101. The 1s are corresponding to the Xs. The polynomial is specific to the symbol width, which, in this case, is 8 bits.

The control for the encoder is straightforward. There are no requirements for the N_IN and R_IN ports. These are used for variable length codes, where the number of information symbols is less than *k,* and the number of check symbol required is less than *(n-k).* While the encoder is streaming input data as fast as possible, the new data control signal (ND) is set High. The CE port can be used if there is a requirement to control power, and setting the CE to Low will disable the operation of the encoder. The BYPASS input can be disabled as all the data input will be encoded.

When the START pulse is asserted, the RS encoder encodes the data and appends the 16 check symbols required. See Figure 8. Even though the RS encoder is processing eight channels, the START pulse only has to be one clock period wide. To save logic, the Ready For First Data (RFFD) on the output of the encoder can be gated with an enable signal to produce the START signal.

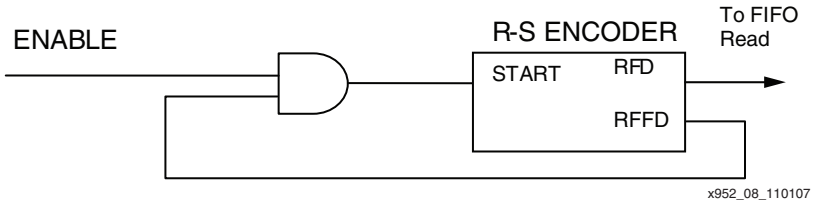

Figure 8: **RS Encoder START**

The encoder is set up for multiple-channel operation and the eight channels are interleaved on DATAOUT. After the eight channels of data, comes the eight sets of check symbols with each set consisting of 16 check symbols. Refer to [Ref 2] for details.

For implementation in the Virtex-5 FPGA, the slowest speed grade can be used.

Decoder

As mentioned earlier for the encoder, two decoders are used because of the data rate required. The decoders must operate at 170 MHz, therefore, the -11 speed grade is needed. Even with Virtex-5 FPGAs, two decoders are still required, but importantly, the slowest speed grade can be used. With the highest speed grade, only a single decoder is required. The RS decoder has been configured via the GUI with the following parameters:

- Data Symbols (k) = 239
- Symbols per Block (n) = 255
- Symbol Width = 8
- Code Specification = Custom
- Number of channels = 8

Select RDY, RFD, and RFFD. See [Ref 3] for definitions of the signals.

The N_IN and R_IN inputs are for variable length codes and are not required. To decode the input data, a SYNC pulse is required and is common to both decoders. For this application note, the encoder START pulse is used as the SYNC input on the decoder. This signal is a single-clock period width even for multi-channel applications. The decoder is set up with the same parameters as the encoder. The RFFD signal can be gated and fed back into the SYNC input of the decoder. The data from the encoders are sent, via error insertion logic, to the decoders. For Virtex-4 FPGAs, each encoder output goes to the data input of the corresponding decoder. For Virtex-5 FPGAs, the byte-wide bus goes to both decoders.

However, each decoder has its own SYNC pulse, unlike in the case of the Virtex-4 FPGAs where the signal is shared.

The DATA_OUT from each decoder is the corrected data. The number of symbols, or bytes, is the size of the block which is 255, and the data are interleaved.

On the output of the decoders, there are a number of status signals that can be used to control external logic. Two useful signals are the BLK_STRT signal, indicating the start of the corrected data, and the INFO_END signal which indicates the last information byte. As there are eight channels, therefore, the BLK_STRT and INFO_END are eight clock periods wide. These signals can be ANDed together and used to control the writing of the RS information data into the FIFO. The check symbol section of the output data is ignored.

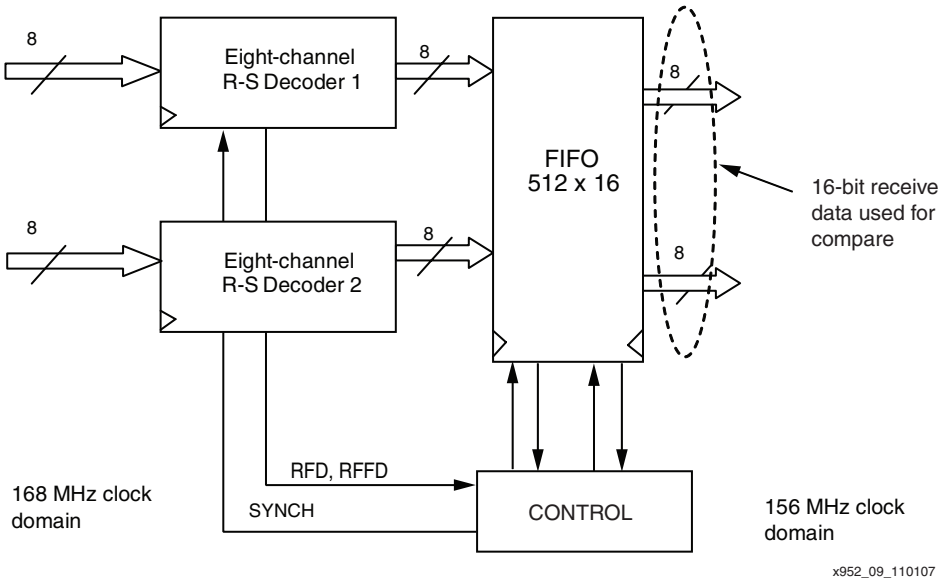

Figure 9: **OTU-1 Decoding**

The RS decoder provides details about the errors it found:

- ERR_CNT: This is error count output which indicates how many errors there are. Its width is set by the n-*k* value which is 16 for ITU-G.709.
- ERR_FOUND: Goes active when any errors are found.
- FAIL: If the number of errors is greater than the error correcting capability of the decoder (that is, eight symbols), this signal is active. The assertion of this signal means the ERR_CNT value should be ignored.

Each decoder has the above error status signals which can be used to give an indication of the quality of the interface. For example, the ERR_FOUND from all the decoders can be ORed together, and when active can be used to store the ERR_CNT values in a register, or a statistics buffer. Similarly, the FAIL output from all the decoders can be ORed together, indicating that the number of errors exceeded the correction capability of the decoder. This signal could be used as an interrupt, indicating the decoder's error capacity has been exceeded.

Even though the decoder gives the number of errors, within n-*k*/2, it does not give the locations of the errors. For locations of the errors, simple external logic is required. An optional port (DATA_DEL) on the decoder, which can be exposed, is the delayed input data. This output when XORed with the DATA_OUT can give an indication of where the errors occurred. When used with a counter, the error positions can also be stored in a statistics buffer. Note that for an eight channel decoder, three block RAMS are used per core.

The decoder data outputs are written to the 512x16 FIFO created by the FIFO Generator at 16 MHz and the read out at 156 MHz.

With the decoder, there are two parameters that need to be considered: latency and throughput. Latency is the number of sampled symbols on DATA_IN to the corrected data on DATA_OUT. How soon data can be sampled by the decoder is not dependent upon the latency, but the throughput. The throughput is defined by the processing delay which is determined by *n-k/2*, that is, 8. For RS (255,239), the delay is 204 samples which is less than the block size (255) and therefore continuous processing can be done. Refer to *Processing Delay* in [Ref 3].

OTU-2 Interface

For the OTU-2, the line rate is 10709225.316 kbps, which equates to 1338.65 MBps. To process that amount of data, eight encoders and decoders are required to each operate at 167 MHz. In the design files, this has been rounded up to 168 MHz.

With the OTU-2, which includes the FEC packets, setting the 128 bit input data is at 83.6658 MHz. In this case, this is organized as eight 2-to-1 multiplexers. Each multiplexer output is byte-wide at 156 MHz. The eight multiplexer outputs can be concatenated together to produce a 64-bit wide bus for the 512x64 FIFO. The 64-bit FIFO output goes to eight encoders The RS encoder/decoders are set up with the same parameters as OTU-1 except the number of channels is two. The total number is eight encoders and eight decoders. The RS decoder when selected for two channels has a higher performance than a single-channel decoder. Therefore, for the OTU-2, the decoder can run in excess of the 168 MHz and can be put in the slowest Virtex-4 FPGA speed grade. See Figure 10.

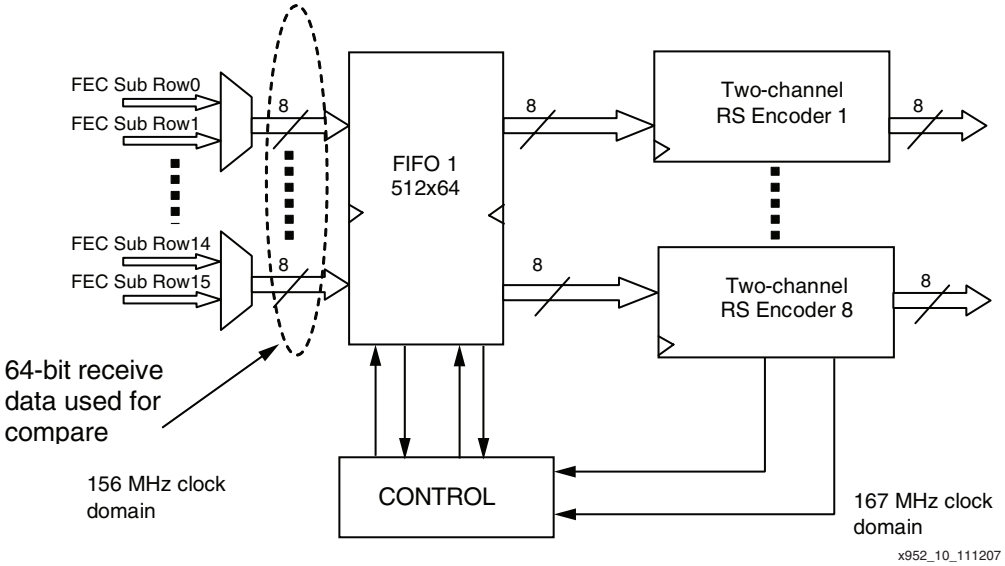

Figure 10: **OTU-2 Encoding**

For the decoder after the injection of errors, the RS decoder detects and corrects any errors and writes the information bytes into the 512x64 FIFO. See Figure 11.

By using the BLK_STRT and INFO_END, the FIFO write signal is created and is just a register with a set and reset.

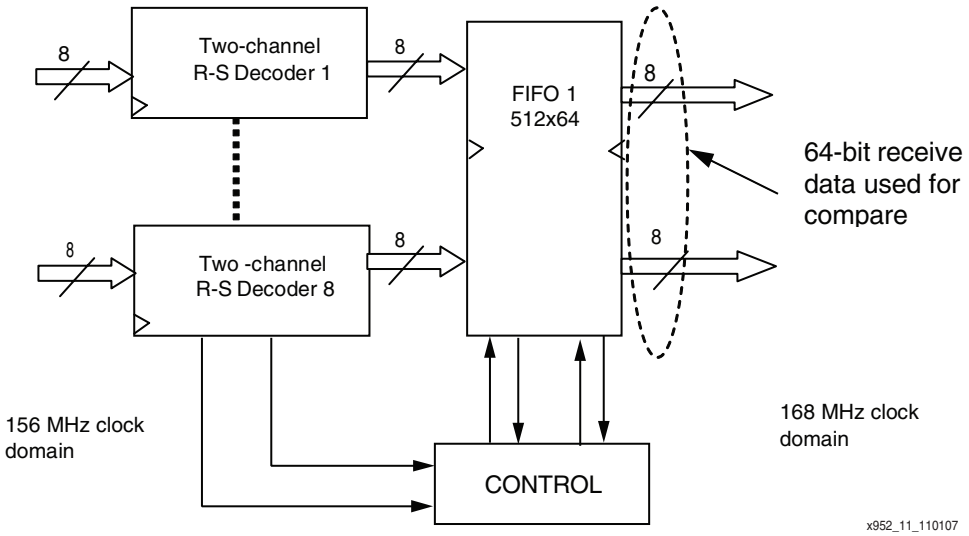

Figure 11: **OTU-2 Decoding**

OTU-3 Interface

For the OTU-3, the line rate is 43018413.559 kbps, which equates to 538 MBps. To process that amount of data 32 encoders and decoders are required to each operate at 167 MHz. Therefore, the data has to be demultiplexed into 256 bits at 167 MHz. The data are written into the 512x256 FIFO and from there into 32 encoders. The encoders are single-channel decoders operating at 167 MHz. To allow for jitter, the mid-speed grade Virtex-4 FPGA has to be used. See Figure 12. .

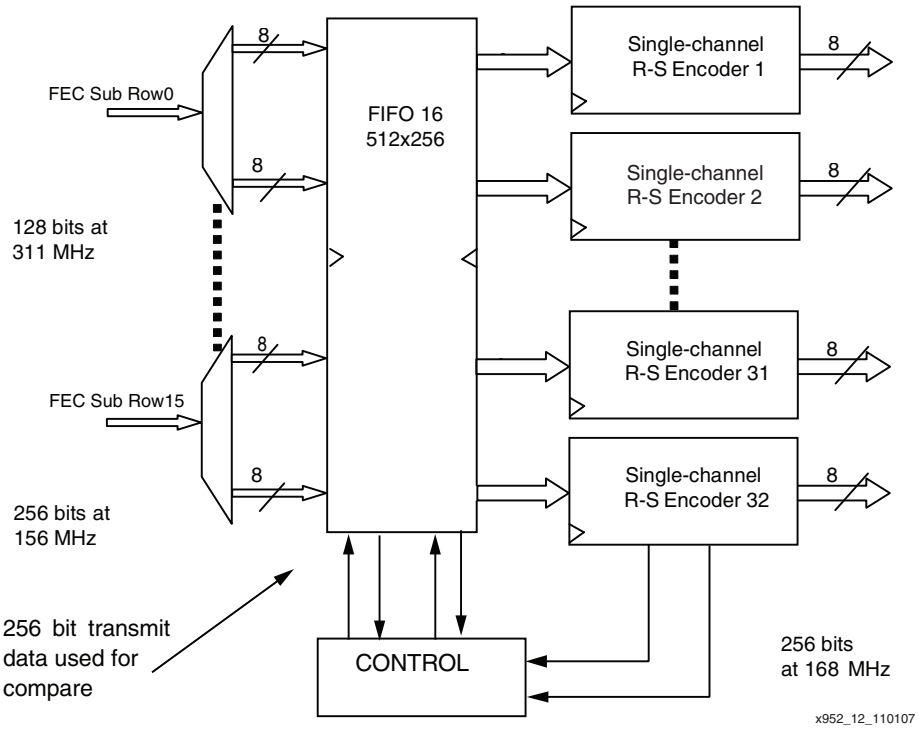

Figure 12: **OTU-3 Decoding**

For Virtex-5 FPGAs, the encoder can be configured as a single-channel encoder running at 312 MHz. In this case, the FIFO is built using the LogiCORE™ FIFO Generator and set up as 512x128. To use lowest speed grade, 32 encoders and decoders were used. In a mid-speed grade, this can be reduced to 16 encoders and decoders.

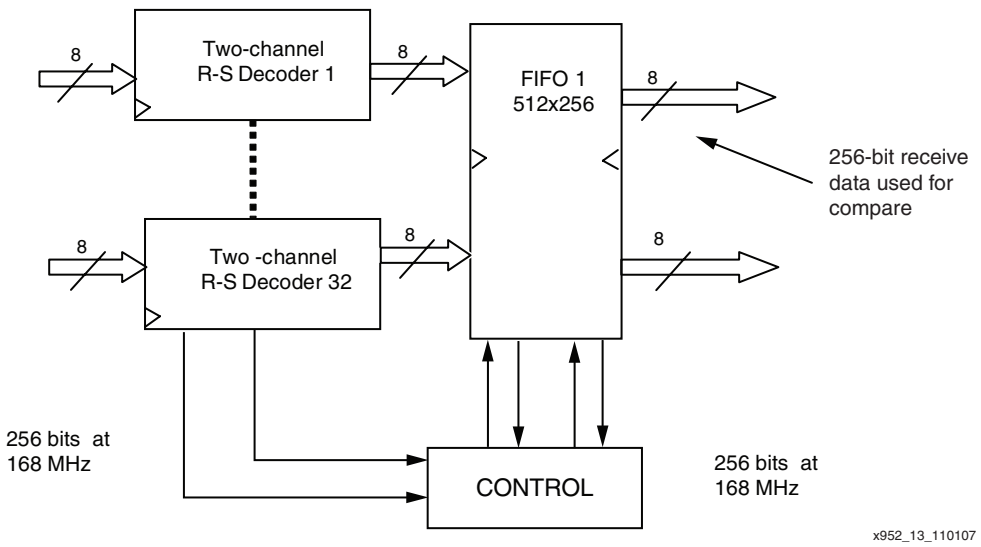

Figure 13: **OTU-3 Decoding**

Enhanced FEC

According to [Ref 5], the RS (255, 239), FEC code has a 5.6 dB net coding gain at 10^{-12} decoder output bit error rate (BER). The G.975.1 standard defines *super FEC* which has better correction capability than the RS (255, 239). There are a number of recommendations in the G.965.1 standard included in Table 2.

Table 2: **Super FEC Schemes**

G.975.1 Subclause	FEC Scheme	Description
I.2	Concatenated FEC	Outer code: RS (255 239): Inner code: CSOC (n0/k0=7/6 J=8)
I.3	Concatenated FEC	Outer code: BCH (3860 3824) Inner code: BCH (2040 1930)
I.4	Concatenated FEC	Outer code: RS (1023 1007) Inner code: BCH (2047 1952)
I.5	Concatenated FEC	Outer code: RS (1901 1855) Inner code: Extended Hamming Product Code (512 502) x (510 500)
I.6	Non-concatenated	FEC LDPC Code
I.7	Concatenated FEC	Two orthogonally concatenated BCH codes
I.8	Non-concatenated FEC	RS (2720 2550)
I.9	Concatenated FEC	Two interleaved extended BCH (1020 988) codes

The advantage of using Virtex-4/Virtex-5 FPGAs is that analysis can be done to investigate which scheme is providing the better protection. The various RS settings, e.g., RS (1023, 1007) and RS (1901, 1855) can be configured by the GUI. However, RS (2720, 2550) exceeds the max *n-k* value of 128 and so cannot be implemented using the CORE Generator GUI. Contact Xilinx if this is required.

For the RS (1023,1007) configuration in the Virtex-5 -1 FPGA, use 1,858 registers, 1,246 LUTs and 2 block RAMs. For the RS (1901,1855), use 4,701 registers, 3,768 LUTs, and 4 block RAMs.

Even though there is not a *Bose-Chaudhuri-Hochquenghem* code (BCH) core, the design can be written in HDL and targeted at the relevant family. Or, contact Xilinx for information on solutions from Xilinx partners on: www.xilinx.com.

So for a super FEC scheme, an arrangement can be made where via a select line either the RS (255,239) scheme could be used, or the RS (255,239) encoded data is sent to the Convolutional Self-Orthogonal Code (CSOC) encoder to implement an improved FEC scheme.

Resource Utilization

Table 3 and Table 4 outlines the Virtex-4 and Virtex-5 FPGA utilization figures for the implementation of the RS transmit consisting of FIFOs and RS encoders.

Table 3: **Virtex-4 FPGA Resource Utilization**

Data Rate	Device	LUTs	FFs	Block RAM	% LUTs	% FFs	% Block RAMs
Encoder							
2.5 Gbps	XC4VSX35 -11	882	751	1	2	2	1
10 Gbps	XC4VSX35 -10	2,035	2,561	2	6	8	1
43 Gbps	XC4VLX200 -11	6801	5,546	8	3	3	2
Decoder							
2.5 Gbps	XC4VSX35 -11	4,867	3,775	7(1)	15	12	3
10 Gbps	XC4VSX35 -10	10,411	12,141	18	33	39	9
43 Gbps	XC4VLX200 -11	35,636	24,901	72	20	13	21

Notes:

1. For eight channel decoder, three block RAMS are used.

Table 4: **Virtex-5 FPGA Resource Utilization**

Date Rate	Device	LUTs	FFs	Block RAMs	% LUTs	% FFs	% Block RAMs
Encoder							
2.5 Gbps	XC5VSX50 -1	625	751	1	2	2	1
10 Gbps	XC5VSX50 -1	1,567	2,561	1	4	7	1
43 Gbps	XC5VLX330 -1	5,561	5,545	4	2	2	1
Decoder							
2.5 Gbps	XC5VSX50 -1	2,978	3,788	4	9	11	3
10 Gbps	XC5VSX50 -1	8,401	12,196	9	25	37	6
43 Gbps	XC5VLX330 -1	26,687	24,644	36	11	12	11

Conclusion

In this application note, examples were given of ITU-G.709 FEC implementation in the Virtex-4 and Virtex-5 FPGA families. The use of the CORE Generator graphical user interface showed easy configuration of the encoder and decoder. The use of core multi-channel capability and core control signals allowed savings in area. The 10 Gbps line rate showed the higher performance of the two-channel decoder and allowed not only reduced area, but operation in lowest speed grade.

The RS IP can be easily configured to invoke different *n,k* values and, therefore, different error protection values can be used.

Reference Design Files

The following reference design source files are available for download from the Xilinx website at: www.xilinx.com/bvdocs/appnotes/xapp952.zip.

The ZIP file has six ISE projects for the OTU-1, OTU-2, and OTU-3 interfaces for Virtex-4 and Virtex-5 FPGA families. Each project consists of:

- ISE 9.2.02 ISE Project
- VHDL files
- CORE Generator™ XCO configuration files
- User constraints file
- Xilinx Architectural Wizard (XAW) file for the Digital Clock Manager
- Simulation DO files

To generate bit files, a 90-day evaluation license is needed. The license allows parameterization, generation, and instantiation of the IP in the ISE Project, as well as download and configuration of the FPGA, and functional and timing simulation can be done. The evaluation license allows for a bitstream to be generated which is fully functional for 2-3 hours.

When the project is opened, Regenerate All Cores creates the NGC netlist. Selecting Behavioral Simulation shows the operation of the cores.

For the OTU-2 reference design, there is the option to create a bitstream and target either an ML402 or an ML506 development board. ChipScope can be used to monitor internal signals to validate the operation of the encoder/decoder. Dual-in-line switches on the boards can be used to select between no inserted errors, errors within the decoder's error correcting capability, and errors which exceed the decoder's capability to correct errors.

References

1. ITU-G.709, *Interface for the Optical Transport Network*
2. Xilinx, DS251, *Reed-Solomon Encoder v6.1* Data Sheet
3. Xilinx, DS252, *Reed-Solomon Decoder v6.1* Data Sheet
4. Xilinx, WP110, *Reed-Solomon Solutions with Spartan-II FPGAs*, by Antolin Agatep
5. ITU-G.975.1, *Forward Error Correction for High Rate **DWDM** Submarine Systems*

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/05/07	1.0	Initial Xilinx release.

Notice of Disclaimer