A CCSDS REED-SOLOMON ENCODER CORE AND RADIATION HARD DEVICE

Sandi Habinc

European Space Agency, ESTEC WSM, Postbus 299, 2200 AG Noordwijk, The Netherlands
sandi@ws.estec.esa.nl

Abstract

With increasing telemetry data rates, there is a need for faster Reed-Solomon and Convolutional on-board encoders than currently exist, both as self standing components and as integrated parts of single chip telemetry systems and custom designs. An ongoing ESA activity aims at developing such reusable and technology independent encoder cores in VHDL, together with a radiation hard component replacing the discontinued MA1916 encoder.

1 INTRODUCTION

A telemetry channel is encoded to gain a higher data throughput at the same bit error rate as for uncoded transmission, but with less energy expended per information bit. The encoding allows the ground segment to perform error detection and correction on down-link data. Reed-Solomon and Convolutional encoding have been used widely on European spacecraft and are endorsed by European Space Agency (ESA) and Consultative Committee for Space Data Systems (CCSDS) standards. The encoding is normally carried out in the telemetry encoder of the spacecraft, just after the virtual channel multiplexer in systems using packet telemetry.

The MA1916 Reed Solomon and Convolutional encoder device (Ref. 6) from GEC Plessey Semiconductor (GB) has been flown on several spacecraft (e.g. SOHO) and will further be used on XMM and Integral. With the manufacturing of the MA1916 device being discontinued, several ongoing spacecraft developments have been left without an alternative replacement. The development of a new encoder named RESCUE has therefore been initiated. This new encoder could be utilised on commercial and scientific spacecraft such as Cobras/Samba and Rosetta.

Instead of directly developing an encoder device for a specific technology, it has been decided first to develop reusable encoder cores. As a second step, a device based on these cores will be developed and will be supported as an Application Specific Standard Product (ASSP) by the foundry. This approach has several advantages over the more classical way of designing, e.g. the intermediate output from the core development can be reused in future designs. There is however an increased effort when compared with classical design methods, since with cores being intended for reuse, requirements on verification and documentation tend to become tighter. Specifying the encoder core functionality involved little effort, since few system requirements covering aspects such as suitable and flexible interfaces etc. needed to be considered at that stage. The more difficult task of specifying the ASSP can now commence while the encoder cores are being developed.

The objective of the core development is to create technology independent Reed-Solomon and Convolutional encoders adhering to the ESA and CCSDS standards. These cores will be modelled in synthesizable VHDL (Very high speed integrated circuit Hardware Description Language). Each core will be fully synchronous with a single clock region and without gated clocks, featuring no combinatorial or feedback loops. The radiation hard MITEL CMOS/SOS5 process will be used when demonstrating the implementability of the cores. This core development will be finalised during 1Q97 with the main output being:

- synthesizable VHDL models (also for simulation);
- test benches with 100% code coverage;
- production test vectors with 98% fault coverage;
- functional specifications and user manuals.

The intention is to use the encoder cores as a virtual second source for the RESCUE device or other devices that will incorporate them, allowing fast transfer to another foundry if necessary. They are not intended for general use, but could be included in products where the agency believes they fit, e.g. in a single chip telemetry encoder.

Initial VHDL models of the encoder cores were developed by ESA. Further development is performed by Smartech (SF) under the prime contractor Dornier Satellitensysteme (D). Dornier will also perform extensive verification of the VHDL models versus the existing MA1916 device. These encoder cores are further discussed in sections 3 and 4.

This consortium together with MITEL Semiconductors (S) is also envisaged for the RESCUE device development, as further discussed in sections 5 through 7.

2 APPLICABLE STANDARDS

To ensure that the Reed-Solomon encoder being developed will be compliant with the existing standards, a review of the relevant ESA and CCSDS documents has been carried out. The encoder shall generate codewords according to the ESA Telemetry Channel Coding standard (Ref. 1). This standard specifies a block code with 8 bits per symbol, 255 symbols per codeword: first 223 symbols being the information symbols and last 32 symbols forming the check symbols. Shortened codewords lengths may be obtained using virtual fill. Implicitly, the CCSDS Telemetry Channel Coding standard (Ref. 2) will also be supported. Although the description of the encoding algorithm in the two standards differs, the same code is specified. This has been reconfirmed both analytically and through VHDL simulation. There are however some differences between the two standards concerning the allowed interleave depths. The ESA standard permits interleave depths 1 to 5 and 8, whereas the CCSDS standard only allows 1 to 5.

Furthermore, the ESA Packet Telemetry Standard (Ref. 3) allows the standard transfer frame lengths with 892, 1115 and 1784 octets when using Reed-Solomon encoding, corresponding to interleave depths 4, 5 and 8. The CCSDS Packet Telemetry standard (Ref. 4) allows the same interleave depths 1 to 5 as in (Ref. 2). Also, the CCSDS Advanced Orbiting Systems standard (Ref. 5) allows interleave depths 1 to 5. It allows shortening of codeword to accommodate compatibility with 32 bit microprocessor systems, where the allowed number of suppressed symbols must be a multiple of the selected interleave depth.

Moreover, the existing MA1916 device implements interleave depths 1, 4 and 5, according to (Ref. 6), and the Virtual Channel Multiplexer (VCM) device supports interleave depths 1, 2, 4 and 5, according to (Ref. 7).

By supporting interleave depths 1 to 8 and virtual fill with the number of suppressed symbols being a multiple of the interleave depth, the above standards and devices can all be covered by a single encoder core.

3 REED-SOLOMON ENCODER CORE

The Reed-Solomon encoder core being developed is targeted towards systems with low to medium data rates. It is a physically small core that will be suitable for future single chip telemetry systems where a small silicon area is a major requirement. The encoder core is also suitable for deep-space missions since the chosen architecture allows implementations with low power consumption and great interleave depths.

The encoder core implements a coding algorithm compliant with (Ref. 1):

- there are 8 bits per symbol;
- there are 255 symbols per codeword;
- the encoding is systematic: the first 223 symbols transmitted are information symbols, and the last 32 symbols transmitted are check symbols;
- the code can correct up to 16 symbol errors and detect up to 16 symbol errors;
- the field polynomial is

$$f(x) = x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$$

- the code generator polynomial is

$$g(x) = \prod_{i=112}^{143} (x + \alpha^i) = \sum_{j=0}^{32} g_j \cdot x^j$$

  with the highest power of x being transmitted first;

- interleaving is supported for depth I in range 1 to 8, where information symbols are encoded as I codewords with symbol numbers $i + j*I$ belonging to codeword i {where $0 \le i < I$ and $0 \le j < 255$};
- shortened codeword lengths are supported, allowing suppression of a number of information symbols equalling to any multiple of the selected interleave depth, where such suppressed symbols are assumed to be in the beginning of the codewords;
- the encoder input and output data are in a representation specified by the following transformation matrix

$$\begin{bmatrix}\iota_0 & \iota_1 & \iota_2 & \iota_3 & \iota_4 & \iota_5 & \iota_6 & \iota_7\end{bmatrix} = \begin{bmatrix}\alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0\end{bmatrix} \times \begin{bmatrix} 0&1&1&1&1&0&1&1 \\ 0&1&1&1&1&0&0&1 \\ 0&0&1&0&1&0&1&1 \\ 0&0&1&1&1&1&1&1 \\ 0&0&0&0&1&0&0&1 \\ 1&0&0&0&0&1&1&1 \\ 0&1&0&1&1&1&1&1 \\ 0&0&1&1&0&1&1&1 \end{bmatrix}$$

where ι_0 is transferred first, and with the following matrix specifying the reverse transformation

$$\begin{bmatrix}\alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0\end{bmatrix} = \begin{bmatrix}\iota_0 & \iota_1 & \iota_2 & \iota_3 & \iota_4 & \iota_5 & \iota_6 & \iota_7\end{bmatrix} \times \begin{bmatrix} 1&1&1&0&1&1&0&1 \\ 0&1&0&1&1&1&1&1 \\ 0&0&0&1&0&1&1&1 \\ 0&1&0&1&1&0&1&0 \\ 1&0&0&0&1&0&0&0 \\ 0&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&1&1 \\ 1&0&0&1&1&0&0&0 \end{bmatrix}$$

The encoder core generates Reed-Solomon codewords by receiving information symbols that are transmitted unmodified while calculating the corresponding check symbols that in their turn are transmitted after the information symbols. A diagram of the encoder core is shown in figure 2, which is directly mapping to the functional blocks of a linear feedback shift register implementing the encoding algorithm as shown in figure 1.

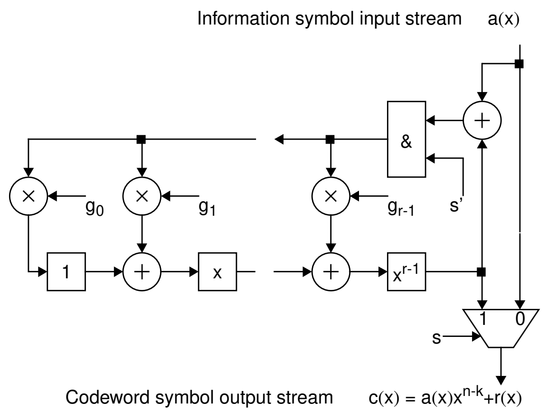

Figure 1: *Reed-Solomon encoder logic*

Check symbol calculation is independent of any previous codewords and correct codeword calculation can therefore begin immediately at the reception of the first information symbol after power-up. No explicit initialisation, such as resetting the check symbol memory elements, is required before correct operation can commence. This is achieved by suppressing the feedback from the check symbol memory or the multiplier when data are assumed to be zero.

To circumvent the effects of heavy particles when not manufactured in a radiation hard process, the control logic is reset and re-synchronised for each codeword. These implementation features will confine any error due to Single Event Upsets (SEU) to the affected codeword. Furthermore, the core does not require the value of the input symbols to be zero while check symbols are transmitted, being the case with some previous encoder designs.

The encoder core does not generate the Attached Synchronisation Marker specified in (Ref. 1). Instead, it has a means for bypassing the check symbol calculation and receiving and transmitting symbols without modifications.

The core does not include a Pseudo Randomiser (Ref. 1), since the necessary circuitry can be easily implemented outside the core if required in a specific device.

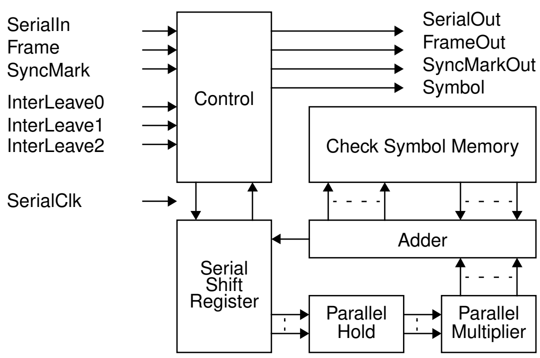

Figure 2: *Reed-Solomon encoder core*

The interface of the encoder core for data input and output is serial. Data and control signals of the core are directly compatible with the VCM output interface, reducing design effort and gate count for additional interface circuitry if integrating both on a single chip. All operations are made in the Galois field defined by the field polynomial described above. By employing the chosen architecture, resource sharing can be maximised which leads to a smaller gate count and area.

Translation between the two symbol representations specified previously is implemented directly in the parallel multiplier, which further reduces the silicon area required for the encoder core.

The encoder core can be configured for any maximum interleave depth I_{max} ranging from 1 to 8, selectable by means of a VHDL generic. For a specific instantiation of the encoder core, any interleave depth ranging from 1 to the chosen I_{max} is supported. Also, for any selected I_{max} the area of the encoder core is minimised, i.e. logic required for implementing interleave depths greater than I_{max} is not unnecessarily included in the synthesised core.

Potentially, it will be possible to select between two check symbol memory organisations for the encoder core; one based on latches or memory that are addressed by means of a bit and an interleave counter; and a second one based on flip-flops organised as synchronous serial shift registers. The first variant should be suitable to use with foundry specific memory macro-cells that could provide low power consumption and further reduce the silicon area needed.

The targeted performance of the encoder core is a data rate of 20 Mbits/s. Only a single phase clock with the same frequency as the bit rate is required. The core will have a gate and routing area of less than 5000 equivalent gates when implemented in MITEL CMOS/SOS5 technology.

4 CONVOLUTIONAL ENCODER CORE

The Convolutional encoder core being developed encodes data according to (Ref. 1) and (Ref. 2). The code has a constraint length of 7 bits and a code rate of 1/2 bit per symbol. It is generated by the two connection vectors G1=1111001 and G2=1011011, with symbol inversion on the output path of G2 and with G1 associated with the first symbol. A diagram of such an encoder is shown in figure 3.

The encoder core is driven by a symbol clock with twice the input bit frequency. Data are input and output synchronously with this clock, and a second clock acts as input bit delimiter. For each input bit two output symbols are generated.

The targeted performance of the encoder core is a 20 Mbits/s data input rate, corresponding to a symbol output rate of 40 Mbits/s.

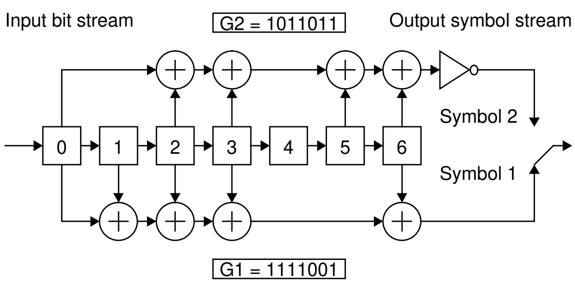

Figure 3: *Convolutional encoder core*

5 RESCUE: REED-SOLOMON AND CONVOLUTIONAL ENCODER

A new radiation hard Reed-Solomon and Convolutional encoder is needed to replace the discontinued MA1916 device for future spacecraft. This new device should not only act as a replacement, but should also have an increased functionality and raised performance. The objective is not to achieve full functional or timing compatibility between the two devices, but to develop a successor that could in most cases replace the MA1916 device with no or only minor modifications existing board designs.

The rest of this section forms a preliminary requirement specification of the new encoder device named RESCUE. The text has been purposely based on the MA1916 specification to allow users easily to recognise potential differences. As a baseline, the new device will be pin compatible with MA1916. The RESCUE device pinout is shown in figure 4 and is further discussed in section 7.

The RESCUE device will be based on the two encoder cores that have been presented previously. Hence, all advantages of assembling already verified blocks will be exploited, such as minimising the required development schedule. The RESCUE device will operate in two principal modes:

- Basic: largely compatible with MA1916;
- Advanced: with enhancements as listed further below.

The following enhancements over the MA1916 device are envisaged for both modes:

- synchronisation markers, information and check symbol output will always be deterministic after a reset (MA1916 generates an invalid frame after a reset);
- internal symbol clock will be re-synchronised for every frame (MA1916 only synchronises on the first frame after a reset, something that has caused problems in some previous board designs);
- errors due to SEUs will be contained within one frame and its check symbols thanks to two mechanisms: control logic will be re-initialised and re-synchronised for each frame, and check symbol calculation will be independent of any previous codewords;
- virtual fill and shortened codeword lengths will be supported, allowing 1 to 222 suppressed symbols per interleave depth (not supported in MA1916);
- deterministic relation between clock signals after reset;
- as a target, 20 Mbits/s data rate for the Reed-Solomon encoder, and 40 Mbits/s symbol output rate for the Convolutional encoder (5 and 10 Mbits/s for MA1916).

The following additional enhancements are envisaged for the Advanced mode:

- special input for synchronisation marker bypass, releasing the requirement on holding the Reed-Solomon data input at logical 0 while check symbols are being output (compatible with VCM);
- additional interleave depths 2 and 3 (supporting all VCM frame lengths);
- on-chip support for cascading two Reed-Solomon encoders, allowing interleave depths 6, 8 and 10 to be realised without additional external logic;

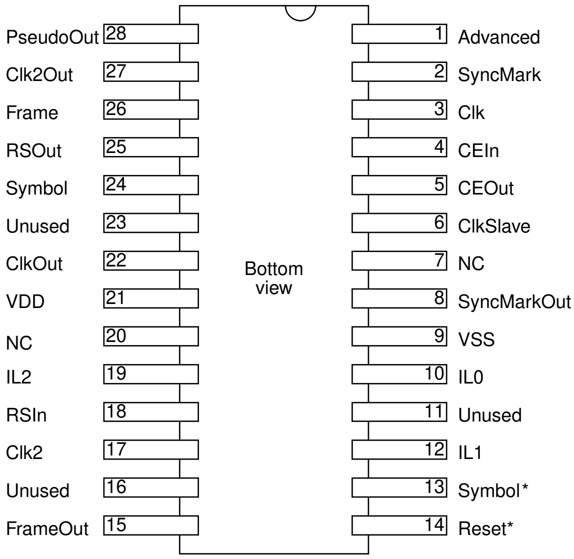

Figure 4: *Foreseen pinout for the RESCUE device*

- pseudo-random generator compliant with (Ref. 1) and (Ref. 2), implementing the polynomial

$$h(x) = x^8 + x^7 + x^5 + x^3 + 1$$

- clock divider always being active (only started on the first frame after a reset in Basic mode and for MA1916).

The envisaged differences compared to MA1916 are:

- no test pattern generator (since not widely used);
- no production test pins (used for other functions);
- changes in sampling and clocking schemes (to allow for a synchronous design and optimisation of the interfaces for higher data rates).

Preliminary characteristics of the RESCUE device are:

Compliance:	implementing ESA and CCSDS standards on channel encoding (for packet telemetry and advanced orbiting systems);
Compatibility:	pin compatible with the MA1916, input interface compatible with the VCM;
Performance:	Reed-Solomon encoder and pseudo-random generator with 20 Mbits/s data rate, Convolutional encoder with 20 Mbits/s input data rate and 40 Mbits/s output symbol rate (preliminary targets);
Interleaving:	1 to 5 (plus 6, 8 and 10 with cascading);
Technology:	MITEL 1.25 µm CMOS/SOS5 process (former ABB Hafo), 6000 equivalent gates
Radiation:	100 kRad guaranteed total dose tolerance, low SEU sensitivity and Latch-up immune
Power:	+5 V supply, low power consumption;
Package:	28 pin flat pack;
Schedule:	design start in 1Q97, prototypes in 3Q97, SOS5 capability assessment programme completion in 2Q97 and SOS5 Capability Approval in 2Q98;
Support:	supported by foundry as an ASSP with a complete data sheet.

6 DISCUSSION POINTS

The baseline functionality of the RESCUE device has been listed here above. There are however functions that have been assessed during specification writing which have not been included. In this section both these categories will be discussed, providing the rationale for inclusion or rejection of some of the more controversial features. Readers are invited to provide comments and suggestions on what the RESCUE device should implement to fit into their current and future applications. Suggestions should be provided before 15 January 1997.

For the Convolutional encoder no changes or enhancements with respect to given standards have been included in the RESCUE device. Means for interchanging G1 and G2 or the position of the inverter could be considered. Also codes with other constraint lengths or bit rates could be considered if at least one specific application requiring this is known before the design start.

Turbo codes have been suggested as a replacement for Convolutional - and sometimes Reed-Solomon - encoding. Such codes have been studied in other ESA activities, but no ESA or CCSDS standard yet covers Turbo coding. Inclusion of such an encoder would therefore only be feasible if at least one application requiring it is known.

For the Reed-Solomon encoder no changes to the code are envisaged since there is a fixed standard for telemetry applications. There are however some other aspects that need to be considered. The RESCUE device is targeted towards applications with low to medium data rates. Therefore, to support higher data rates, as would normally be the case for advanced orbiting systems, is not feasible.

Interleave depths 1 to 5 have been chosen for the RESCUE device, covering most applications except for some deep-space missions where an interleave depth of 8 is desirable. To support this interleave depth on-chip, an increase of some 50% of silicon area would have been required, resulting in a more expensive chip and unused functionality for the average application. Instead, the baseline is to support interleave depths 6, 8 and 10 by cascading two RESCUE devices as shown in figure 5. No additional external logic will be required for cascading.

In cascading mode, both the master and slave devices receive incoming data at bit rate frequency. Each device performs Reed-Solomon encoding on half the incoming data, taking every second information symbol as input and generating every second codeword symbol. The master works on even numbered interleave depths and the slave on odd numbered ones. The clock input on the slave encoder is driven by the special ClkSlave output from the master, generating a clock that is active only when odd numbered symbols are received or transmitted. Complete codewords (comprising both even and odd numbered symbols) are provided on the RSOut and PseudoOut outputs of the master encoder. A drawback with this cascading approach is that the clocking scheme of the device becomes more complicated. This could be mitigated if the encoder core is designed in a way allowing odd or even symbols to be ignored or suppressed, but that would adversely increase the core sized. Note that the symbols from the Convolutional encoder are provided on the CEOut output of the master.

To include additionally the RS(255, 239) code used for digital video broadcasting seems not feasible since it is based on a different field polynomial than the code used for telemetry. Consequently, this would result in a large area increase. Also, the desired data rates of some 200 Mbits/s are not compatible with the low power/small area approach chosen for the RESCUE device.

The clocking and sampling schemes of the RESCUE device will be adapted to the VCM for the Advanced mode, but will remain compatible with MA1916 for the Basic mode. As an example, the ClkOut output will always toggle when Clk2 is active, in contrast to MA1916 where this is started at the beginning of the first frame after a reset.

The RESCUE device will include a Pseudo Randomiser compliant with (Ref. 1) that will mix the Reed-Solomon output bit stream with a deterministic pseudo-random sequence. This function can be used to obtain the bit transition density required on a channel to allow the receiver on ground to maintain bit synchronisation.

Finally, the question of whether to provide a drop-in compatible replacement for the MA1916 with the same timing and functionality, including the less desirable features, is still open. The baseline is to support the functionality, but not to have exactly the same timing. This is the case for the undeterministic relationship between the input and output clock signals after reset that has been observed for the MA1916 device, something that will be improved in the RESCUE device.

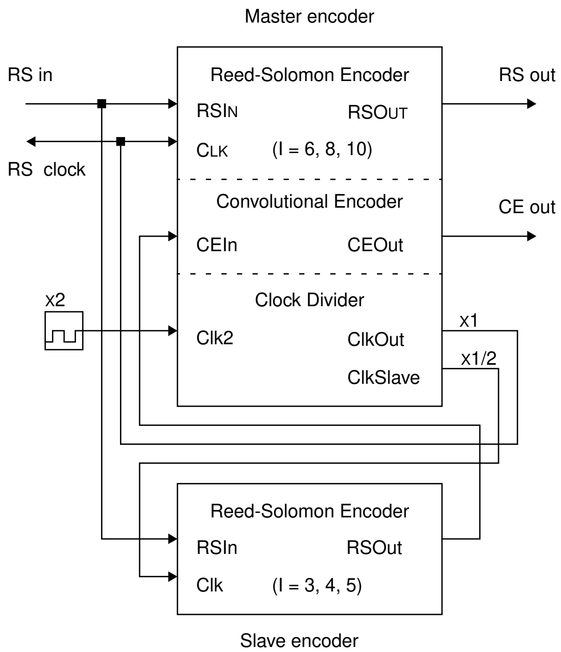

Figure 5: *Two cascaded RESCUE devices*

7 RESCUE PINOUT COMPARED TO MA1916

With pin compatibility between the RESCUE and MA1916 devices being an essential requirement, it was felt necessary to conduct a preliminary comparison. A large part of the MA1916 functionality is directly reflected on its pins and their definitions have therefore been repeated in the comparison below.

The intention is to provide the reader with a comprehensible description of the potential differences between the two devices. For a deeper understanding of the dissimilarities and their impact, the reader is recommended to read the MA1916 data sheet (Ref. 6). A summary of the pinout comparison is given in table 3.

Pin 1: Advanced ⇔ T2 (Test Pattern Select)

For MA1916, this input selects between patterns to be generated by the test generator.

For the RESCUE device, this input will select between two operational modes:

- Basic: largely compatible with MA1916 when tied to logical 0;
- Advanced: enhanced mode when tied to logical 1.

Pin 14: Reset* ⇔ n_RST (Reset)

No difference between the MA1916 and RESCUE devices.

Pin 17: Clk2 ⇔ CLK (Clock)

For MA1916, this clock input drives the Reed-Solomon and Convolutional encoders and the test generator.

For the RESCUE device, this input clock will drive the Convolutional encoder and the clock divider, being active on the rising edge.

Pin 27: Clk2Out ⇔ CLK_OUT (Clock Out)

For MA1916, this output is the buffered CLK input.

For the RESCUE device, this output will be the buffered Clk2 input (which is compatible with MA1916).

Pin 22: ClkOut ⇔ CLKS (Synchronisation Clock)

For MA1916, this output defines the timing of the Reed-Solomon encoder, acting as a bit delimiter. The toggling of this output begins at the first rising SMC edge after reset.

For the RESCUE device, this output will act in two modes:

- Basic: compatible with MA1916;
- Advanced: this output will carry a clock signal with half the Clk2 frequency, that will always toggle when the RESCUE device is supplied with a clock on the Clk2 input. This will allow the bit clock of the VCM device to be directly driven from the RESCUE device, without the need for an external clock divider as is the case with current VCM/MA1916 configurations.

Pin 3: Clk ⇔ CE_CLKS (Convolutional Clock)

For MA1916, this input defines the timing of the Convolutional encoder, acting as a bit delimiter.

For the RESCUE device, this clock will drive the Reed-Solomon encoder and act as a bit delimiter for the Convolutional encoder.

Pin 6: ClkSlave ⇔ TEST_POINT (Production Test)

For MA1916, this output is used for production testing.

For the RESCUE device, this output will carry a clock that should be connected to the slave encoder when cascaded.

Pin 10: IL0 ⇔ SEL_A (Interleave Depth Select)

For MA1916, this input is used for selecting the interleave depth of the Reed-Solomon encoder.

For the RESCUE device, this input together with IL1 and IL2 will select the Reed-Solomon encoding interleave depth, acting in two modes as defined in table 1 and table 2.

Basic mode (compliant with MA1916)				
IL0	IL1	IL2	Interleave depth	Remarks
0	0	-	5	IL2 can be tied to logical 0 or 1
0	1	-	4	IL2 can be tied to logical 0 or 1
1	0	-	1	IL2 can be tied to logical 0 or 1
1	1	-	5	IL2 can be tied to logical 0 or 1

Table 1: *Interleave depth selection: basic mode*

Advanced mode				
IL0	IL1	IL2	Interleave depth	Remarks
0	0	0	1	
0	0	1	2	
0	1	0	3	
0	1	1	4	
1	0	0	5	
1	0	1	6	slave set to 3 when cascading
1	1	0	8	slave set to 4 when cascading
1	1	1	10	slave set to 5 when cascading

Table 2: *Interleave depth selection: advanced mode*

Pin 12: IL1 ⇔ SEL_B (Interleave Depth Select)

For MA1916, this input is used for selecting the interleave depth of the Reed-Solomon encoder.

For the RESCUE device, see IL0 for a definition.

Pin 19: IL2 ⇔ T0 (Test Pattern Select)

For MA1916, this input selects between patterns to be generated by the test generator.

For the RESCUE device, see IL0 for a definition.

Pin 26: Frame ⇔ SMC (Select Message or Checksum)

For MA1916, this input selects between information symbol input or check symbol output. No distinction is made between check symbol output and synchronisation marker transfer.

For the RESCUE device, this input will act in two modes:
- Basic: the same functionality as for MA1916;
- Advanced: this input will also select between information symbol input or check symbol output, but the distinction between check symbol output and synchronisation marker transfer will be performed explicitly with the SyncMark input. This mode is compatible with the VCM output interface.

Pin 15: FrameOut ⇔ SMC_OUT (Message/Checksum)

For MA1916, this output is used with the test generator.

For the RESCUE device, this output will act in two modes:
- Basic: a logical 0 will be output;
- Advanced: the Frame input value will be output after being sampled on the rising Clk edge.

Pin 2: SyncMark ⇔ T1 (Test Pattern Select)

For MA1916, this input selects between patterns to be generated by the test generator.

For the RESCUE device, this input will act in two modes:
- Basic: can be tied to a logical 0 or 1;
- Advanced: this input will be used for indicating when a synchronisation marker is transferred.

Pin 8: SyncMarkOut ⇔ ST1 (RS Output Valid)

For MA1916, this output indicates when the output from the Reed-Solomon encoder is valid after a reset.

For the RESCUE device, this output will act in two modes:
- Basic: a logical 1 will be output, since the encoder will transfer unmodified synchronisation markers and produce valid codewords directly after reset;
- Advanced: the SyncMark input value will be output after being sampled on the rising Clk edge.

Pin 18: RSIn ⇔ MSG (Message)

For MA1916, this input carries input data and synchronisation markers, and must be held at logical 0 while check symbols are output.

For the RESCUE device, this input will also carry input data and synchronisation markers. But in the Advance mode it will not be required to stay at logical 0 while check symbols are output, since it is masked by the SyncMark input.

Pin 25: RSOut ⇔ RSE_OUT (RS Output)

For the MA1916 and RESCUE devices, this output carries codeword symbols and synchronisation markers.

Pin 28: PseudoOut ⇔ MSG_OUT (Test Message Out)

For MA1916, this output carries patterns generated by the test generator.

For the RESCUE device, this output will act in two modes:
- Basic: a logical 0 will be output;
- Advanced: this output will carry the Reed-Solomon encoder output mixed with a pseudo-random sequence.

Pin 24: Symbol ⇔ SYZ (Byte Rate Clock)

For the MA1916 and RESCUE devices, this output is a symbol clock used in the Reed-Solomon encoding. It is high during every eighth CLKS period and low at other times.

Pin 13: Symbol* ⇔ SZY (Byte Rate Clock)

For the MA1916 and RESCUE devices, this output has the inverted Symbol value (i.e. the SYZ value for MA1916).

Pin 4: CEIn ⇔ CE_IN (Convolutional Data In)

For the MA1916 and RESCUE devices, this input carries input data to the Convolutional encoder. This input also carries odd check symbols from the slave to the master when two RESCUE devices are cascaded.

Pin 5: CEOut ⇔ CE_OUT (Convolutional Output)

For the MA1916 and RESCUE devices, this output carries output symbols from the Convolutional encoder.

Pin 7 and 20: NC ⇔ N/C (Not connected)

No difference between the MA1916 and RESCUE devices.

Pin 11: Unused ⇔ ST2 (Production Test Output)

For MA1916, this output is used for production testing.

For the RESCUE device, a logical 0 will be output.

Pin 16: Unused ⇔ READY (Test Data Valid)

For MA1916, this output is used for test patter generation.

For the RESCUE device, a logical 1 will be output.

Pin 23: Unused ⇔ T3 (Production Test Input)

For MA1916, this input is used for production testing.

For the RESCUE device, it can be tied to a logical 0 or 1.

Pin 9: VSS ⇔ VSS

No difference between the MA1916 and RESCUE devices.

Pin 21: VDD ⇔ VDD

No difference between the MA1916 and RESCUE devices.

Pin number	I/O type	RESCUE pin name	MA1916 pin name	Basic mode	Advanced mode	Compatible with MA1916	Remarks
1	I	Advanced	T2	tie to logical 0	tie to logical 1	partially	mode selection
14	I	Reset*	n_RST	reset	as for Basic mode	yes	
17	I	Clk2	CLK	as CLK	as for Basic mode	partially	drives Convolutional encoder
27	O	Clk2Out	CLK_OUT	as CLK_OUT	as for Basic mode	yes	Clk2 buffered
22	O	ClkOut	CLKS	as CLKS	always toggled	yes	Clk2 divided by 2
3	I	Clk	CE_CLKS	as CE_CLKS	as for Basic mode	partially	drives Reed-Solomon encoder
6	O	ClkSlave	TEST_POINT	logical 0	slave clock	partially	used for cascading
10	I	IL0	SEL_A	as SEL_A	selects interleave depth	yes	supports 1-5 in advanced mode
12	I	IL1	SEL_B	as SEL_B	selects interleave depth	yes	supports 1-5 in advanced mode
19	I	IL2	T0	tie to logical 0 or 1	selects interleave depth	yes	supports 1-5 in advanced mode
26	I	Frame	SMC	as SMC	as for Basic mode	yes	compatible with VCM
15	O	FrameOut	SMC_OUT	SMC delayed	as for Basic mode	no	(old test generator output)
2	I	SyncMark	T1	tie to logical 0 or 1	bypass RS encoding	yes	compatible with VCM
8	O	SyncMarkOut	ST1	logical 1	SyncMark delayed	partially	(old production test output)
18	I	RSIn	MSG	as MSG	as for Basic mode	yes	
25	O	RSOut	RSE_OUT	as RSE_OUT	as for Basic mode	yes	
28	O	PseudoOut	MSG_OUT	logical 0	pseudo-random transitions	no	(old test generator output)
24	O	Symbol	SYZ	as SYZ	as for Basic mode	yes	
13	O	Symbol*	SZY	as SZY	as for Basic mode	yes	
4	I	CEIn	CE_IN	as CE_IN	as for Basic mode	yes	also used for cascading
5	O	CEOut	CE_OUT	as CE_OUT	as for Basic mode	yes	
7 & 20	N/C	NC	N/C			yes	
11	O	Unused	ST2	logical 0	logical 0	partially	value compatible with MA1916
16	O	Unused	READY	logical 1	logical 1	partially	value compatible with MA1916
23	I	Unused	T3	tie to logical 0 or 1	tie to logical 0 or 1	yes	(old production test input)
9	P	VSS	VSS			yes	
21	P	VDD	VDD			yes	

Table 3: *Comparison between the pinouts of the RESCUE and MA1916 devices*

8 SUMMARY

An ongoing development of reusable Reed-Solomon and Convolutional encoder cores has been presented. The encoders are oriented towards low or medium data rate applications such as packet telemetry. They are written in synthesizable VHDL and can be modified to accommodate higher data rates or other encoding algorithms if required.

The cores will potentially provide reduction of design effort for devices that incorporate them, since they already will have been verified for correct operation and implementability. If a device incorporating the encoders needs to be transferred to another foundry, the cores will act as a virtual second source reducing further design efforts.

In parallel with the current encoder core development, further specification of a radiation hard device that will replace the discontinued MA1916 device is necessary. Users are asked to provide detailed requirements on what functionality is needed and what new capabilities should be incorporated in the replacement device named RESCUE.

Please direct comments and suggestions regarding the preliminary RESCUE device specification to the author.

9 REFERENCES

1 Telemetry Channel Coding Standard, ESA PSS-04-103, Issue 1, September 1989
2 Telemetry Channel Coding, CCSDS 101.0-B-3, 1992
3 Packet Telemetry Standard, ESA PSS-04-106, Issue 1, January 1988
4 Packet Telemetry, CCSDS 102.0-B-4, 1995
5 Advanced Orbiting Systems, CCSDS 701.0-B-2, 1992
6 GPS SOS Radiation Hard Handbook, July 1995
7 MITEL VCM HAF_12396 Data Sheet, January 1994

10 ACRONYMS AND ABBREVIATIONS

ASSP Application Specific Standard Product
CCSDS Consultative Committee for Space Data Systems
CMOS Complementary Metal-Oxide Semiconductor
RS Reed-Solomon
SEU Single Event Upset
SOS Silicon On Sapphire
VCM Virtual Channel Multiplexer
VHDL VHSIC Hardware Description Language
VHSIC Very High Speed Integrated Circuits