WP110 (v1.1) February 10, 2000

Reed-Solomon Solutions with Spartan-II FPGAs

Author: Antolin Agatep

Summary

This paper explains the theory behind Reed-Solomon error correction, and discusses how a variety of practical Reed-Solomon encoding/decoding solutions can be implemented using Xilinx Spartan™-II family FPGAs.

Introduction

The Spartan-II family, combined with a vast soft IP portfolio, is the first programmable logic solution to effectively penetrate the ASSP marketplace.

Xilinx recently announced the availability of LogiCORE™ Reed-Solomon products. These cores are proven and optimized in their FPGA implementation. Smart-IP Technology is used, taking advantage of the relational placement constraint capabilities of Xilinx development software tools to leverage the distributed memory and segmented routing of the FPGAs. As a result, the user can easily customize the Reed-Solomon core and always achieve a predictable, highly-optimized implementation with the highest possible performance, which is unaffected by device size and surrounding user logic.

Xilinx also provides Reed-Solomon IP through AllianceCORE™ partners Memec Design Services (MDS) and ISS. The Reed-Solomon solutions from Xilinx on a Spartan-II device are good examples highlighting the concept of a programmable ASSP. Both MDS and ISS are members of the Xilinx AllianceCORE program and have a wealth of other IP cores that cater to a variety of applications.

About ISS

ISS is a leading supplier of application-specific virtual components (ASVCs) for multimedia and communications System-on-a-Chip (SOC) integrated circuits. Using proprietary techniques for direct-mapped implementations of digital signal processing functions and algorithms in hardware, ISS delivers solutions for wireless and wired communications, digital video, and digital imaging applications, realizing 10X to 1000X performance improvements compared to conventional implementations using software-programmable DSP microprocessors. For more information, visit the ISS website at http://www.iss-dsp.com.

About MDS

MDS is the design and engineering division of the Memec global semiconductor distribution group of companies. The MDS mission is to offer clients a complete suite of Programmable Logic engineering services, ranging from design evaluations all the way through to complete turnkey device solutions. Using MDS will reduce the time it takes to get new products to market, improving productivity and profitability for original equipment manufacturers in ever-increasing competitive business arenas. MDS is also a major developer of Intellectual Property for Xilinx FPGAs. For more information, visit the MDS website at http://www.memecdesign.com.

What is Reed-Solomon?

In real world communication, errors are introduced in messages sent from one point to another (Figure 1). Reed-Solomon is an error-correcting coding system that was devised to address the issue of correcting multiple errors—especially burst-type errors—in mass storage devices (hard disk drives, DVD, barcode tags), wireless and mobile communications units, satellite links, digital TV, digital video broadcasting (DVB), and modem technologies like xDSL ("x" refering to all the existing DSL solutions, whether ADSL, VDSL, SDSL, or HDSL).

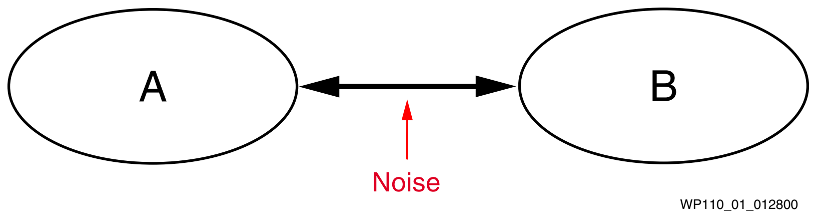

Figure 1: **Two Points Exchanging Information**

In order for the transmitted data to be corrected in the event that it acquires errors, it has to be encoded. The receiver uses the appended encoded bits to determine and correct the errors upon reception of the transmitted signal. The number and type of errors that are correctable depend on the specific Reed-Solomon coding scheme used.

A Reed-Solomon code is specified as $RS(n, k)$ with s-bit symbols, where n is the total number of bytes the code word contains and k is the number of data bytes. The number of parity bytes is equal to $n - k$, where n is 2 raised to the power of s minus one (2^{s-1}). A Reed-Solomon decoder can correct up to t number of bytes, where $2t = n - k$.

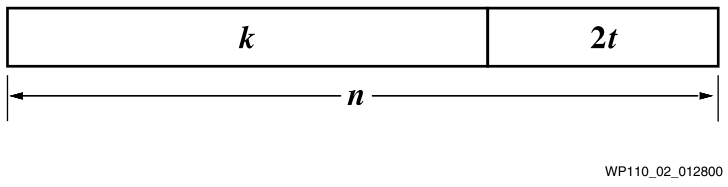

Figure 2: **Reed-Solomon Code Word**

Figure 2 shows a Reed-Solomon code word in which the data is left unaltered while the parity bits are suffixed to the data bits. This type of code is also known as a *systematic code*.

A well-known example of a Reed-Solomon code is $RS(255, 223)$ with 8-bit symbols. For this specific Reed-Solomon code, each code word has 255 total bytes, with 223 bytes of data and 32 bytes for parity. This code has: $n = 255,\ k = 223,\ s = 8,\ 2t = 32,\ t = 16$.

This means that the decoder can automatically correct 16 symbol errors—up to 16 bytes anywhere in the code word.

A Brief Summary of Reed-Solomon Terminology:

- *Symbol_Width* 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

Basic Features of the Reed-Solomon Encoder

A block diagram for a basic Reed-Solomon encoder is shown in Figure 3.

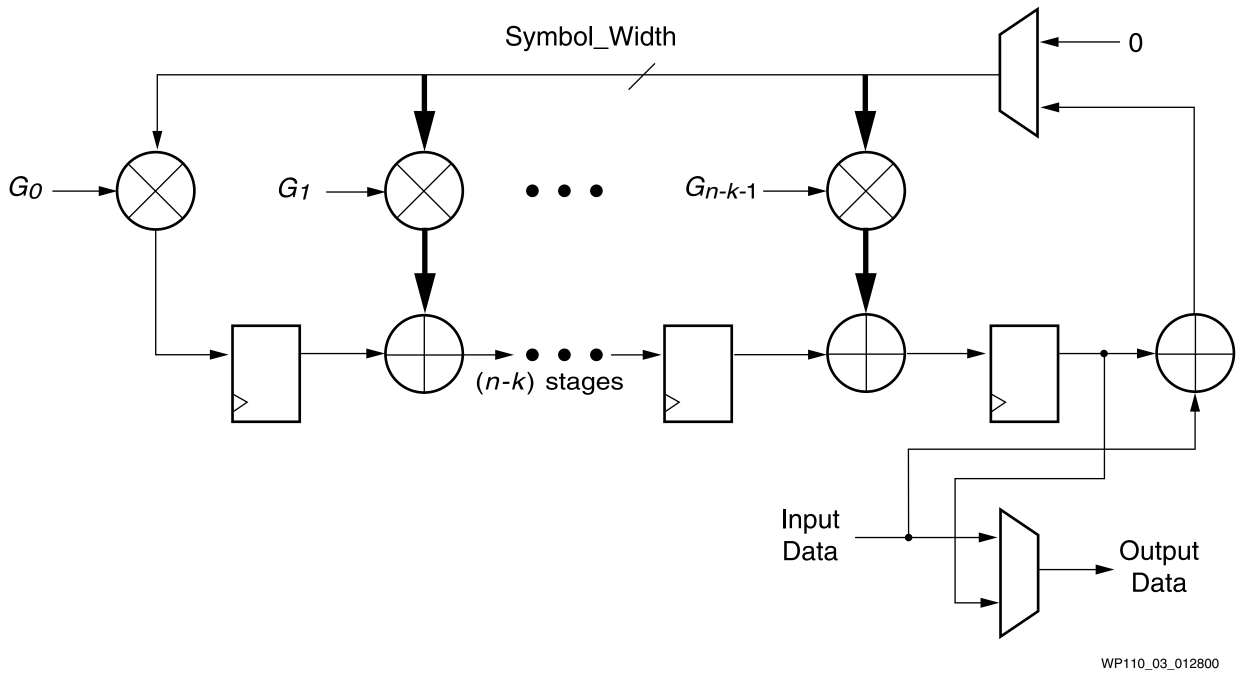

Figure 3: **Encoder Block Diagram**

Xilinx's Reed-Solomon encoder has a number of powerful features:

- Web-based configuration and downloading capability
- Fully synchronous, bit-parallel systematic encoder
- Customizable, supports several standards
- Optimized for Spartan-II FPGAs

Power Savings Resulting from "Coding Gain"

Besides correcting burst errors, the Reed-Solomon encoding-decoding solution offers another significant benefit for wireless communications systems: it allows for transmission at lower power levels. This power saving is called "coding gain," and it allows substitution of lower-cost, lower-power parts in transmitter electronics.

Reed-Solomon coding gain, in decibels (dB), is the difference between the measured power needed to transmit and receive error-free data without encoding and the measured power needed using the Reed-Solomon coding scheme. A graphical representation of the relationship for various encoding schemes is shown in Figure 4. Note that **a**, the non-encoded PSK, is 10.5 dB compared to 2.5 dB for **e** (Reed-Solomon coding)!

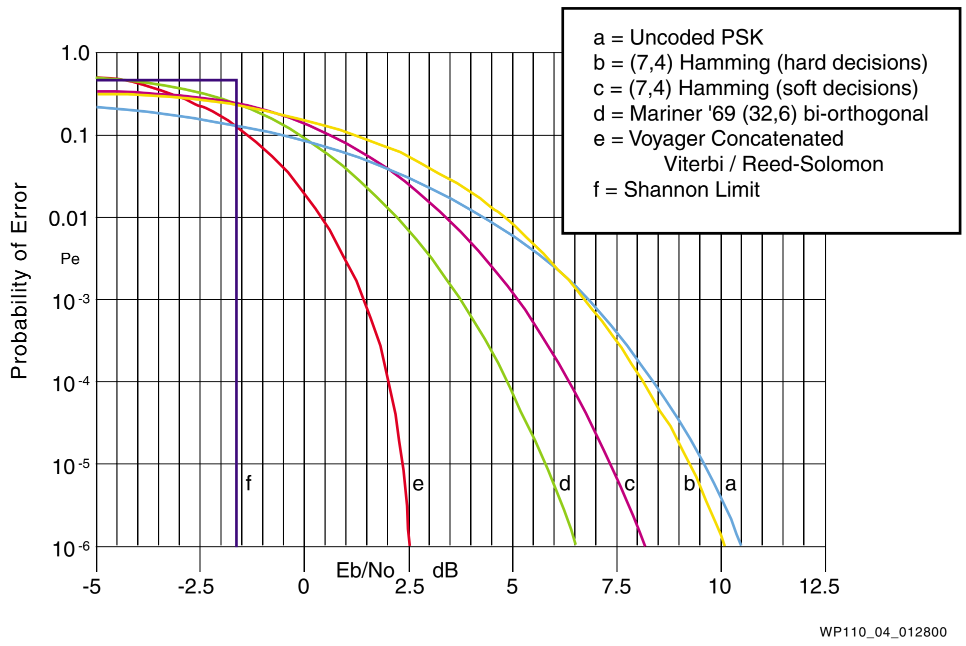

Figure 4: **Graph Showing Reed-Solomon's Coding Gain**

Calculating the Reed-Solomon Encoder Design Parameters

As seen in Figure 2, binary data is grouped with check symbols, $2t$, to form a code word. The check symbols are calculated by choosing a generator polynomial $g(x)$, then choosing the check symbols such that $c(x)$ is divisible by $g(x)$. Now, if $c(x)$ is a multiple of $g(x)$ and if $g(x)$ has roots, then $c(x)$ has the same roots as $g(x)$. This fact allows the validity of the code word to be tested.

All valid code words are exactly divisible by the generator polynomial. The general form of the generator polynomial is:

$$g(x) = (x-\alpha^{0})(x-\alpha^{1})\ldots(x-\alpha^{n-k-2})(x-\alpha^{n-k-1})$$

which has the product form of:

$$g(x) = \prod_{i=0}^{n-k-1}(x-\alpha^{h\times(Generator\_Start+i)})$$

and when expanded has the polynomial form of:

$$g(x) = G_{n-k-1}x^{n-k-1} + G_{n-k-2}x^{n-k-2} + \ldots + G_1x + G_0$$

The following is a worked-out example of calculating a generator polynomial by hand. In this example, $GF(16)$, $n = 15$, $k = 13$, $Generator\_start = 1$, $h = 1$.

From the example calculation, $G_0 = 8$, $G_1 = 6$, and $G_2 = 1$. Substitute these G_0, G_1, and G_2 values in the diagram of Figure 3 to create the specific Reed-Solomon encoder.

$$\begin{aligned} g(x) &= \prod_{i=0}^{1}\left(x-\alpha^{1\times(1+i)}\right) \\ &= (x-\alpha^1)(x-\alpha^2) \\ &= (x+\alpha^1)(x+\alpha^2) \\ &= x^2+(\alpha^1+\alpha^2)x+\alpha^1\alpha^2 \\ &= x^2+\alpha^5 x+\alpha^3 \\ &= \underset{G_2}{\underset{\uparrow}{1}}x^2+\underset{G_1}{\underset{\uparrow}{6}}x+\underset{G_0}{\underset{\uparrow}{8}} \end{aligned}$$

Reed-Solomon Decoders

Basic Features of the Reed-Solomon Decoder

Figure 5, a block diagram of the Reed-Solomon decoder, shows the functions necessary to determine if a received Reed-Solomon encoded signal has been corrupted by noise.

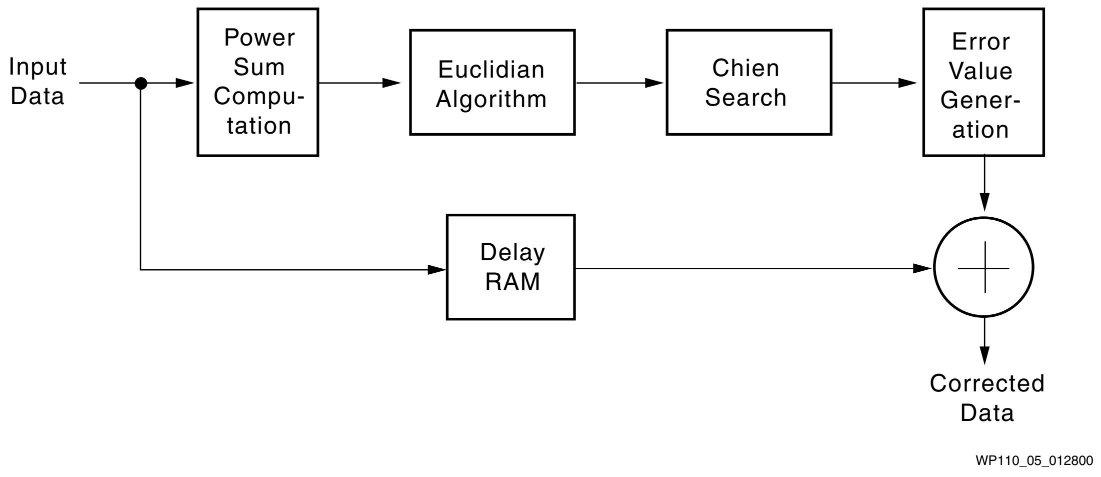

Figure 5: **Decoder Block Diagram**

Xilinx's Reed-Solomon decoder has a number of very useful features:

- Fully synchronous bit-parallel decoders
- Fully parameterizable
- Supports discrete or continuous input data
- Supports erasure decoding

- Optimized for Spartan-II

The received encoded information is decoded by *a*) calculating the syndromes, *b*) finding the error locator polynomial, *c*) finding the error evaluator polynomial, and finally *d*) calculating the error locations.

Reed-Solomon Decoder Applications

CD Player

A very popular consumer application for the Reed-Solomon decoder is the CD player. The Reed-Solomon encoding is literally burned into the CD's surface; the decoding happens when the CD is played back by the user. Reed-Solomon encoding is a very logical and effective solution for the problems inherent in CD storage because it makes the media very tolerant of scratches. Compare this to the old vinyl turntable records that were quite intolerant of dust and scratches.

The decoding occurs in the CD player itself. The laser diode and its associated optics read the "pits" from the CD's surface, passing the information to the Reed-Solomon decoder and then to the other CD player system blocks for further processing.

ADSL Modem

The transmitter and receiver of an ADSL modem implementation are shown in Figure 6. The circled areas highlight the Reed-Solomon encoding and decoding blocks.

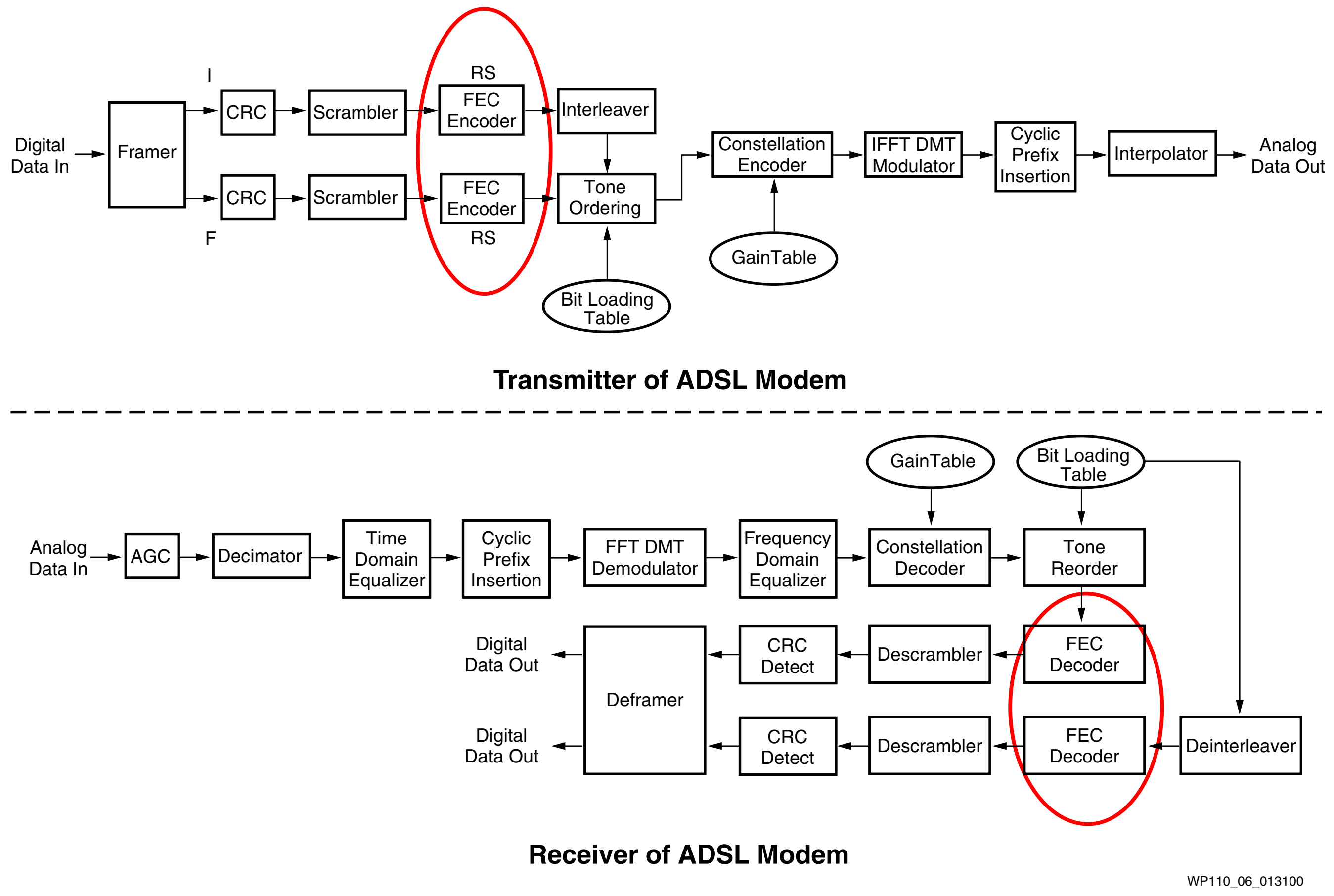

Figure 6: **ADSL Transmitter and Receiver Block Diagrams**

Digital Television (DTV)

One of the more popular consumer applications for the coming decade will be digital television (DTV). An implementation of a DTV Reed-Solomon decoder is shown in Figure 7 (Motorola MC92301).

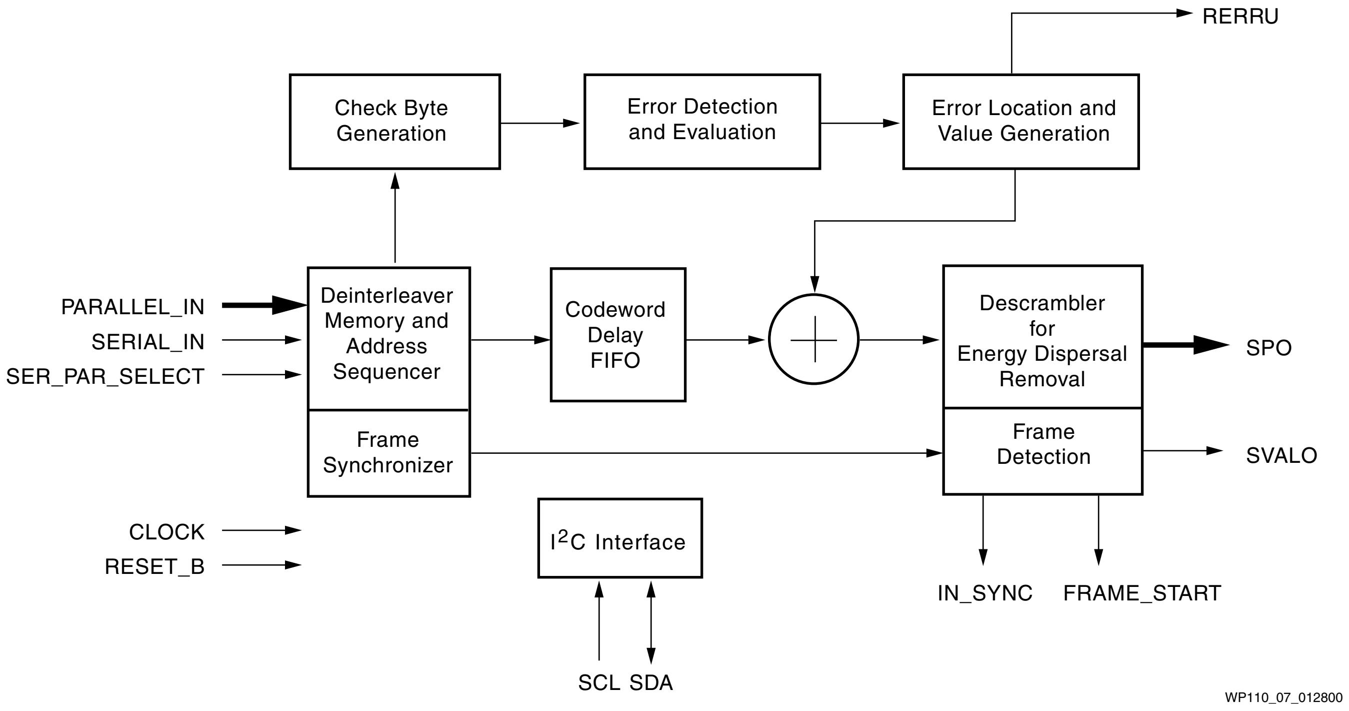

Figure 7: **DTV Reed-Solomon Decoder**

Many Other Popular Applications

The following is a list of popular applications that use the Reed-Solomon scheme:

- xDSL (ADSL, VDSL, HDSL, SDSL)
- CD players, DVD players
- DVB, DTV, ATSC
- Mobile systems geo-synchronous satellite communications links
- Hard disk drives, CD-ROMs
- Wireless communications (cell phones, base stations)
- Wireless-enabled PDAs
- RAID (Redundant Array of Inexpensive Disks) controllers with fault-tolerance
- Barcode tags and readers
- Deep-space probe missions (Voyager, Mariner)
- Interplanetary reconnaissance (Mars Lander)

Spartan-II Advantages Over Traditional ASSPs

Easily Debugged

Most stand-alone ASSPs never behave as expected, due to reasons like bugs in the silicon, system integration issues, software drivers, or even user error. Whatever the cause, verifying and identifying device problems can be very difficult with stand-alone ASSPs, but a lot easier with programmable ASSPs.

Created from the fabric of a proven FPGA technology using pre-verified silicon guaranteed to perform, the Spartan-II family narrows down potential problems to a software-only issue. And Xilinx provides powerful tools that tremendously improve the success of the final solution. With the help of HDL compilers, simulators, test benches, and run-time debugging tools like ChipScope, designers can easily identify debugging problems.

Because Spartan-II is inherently reprogrammable, fixing the problems is also very simple. This is a tremendous value-added feature stand-alone ASSPs cannot offer. It is much simpler and safer to integrate a reprogrammable Reed-Solomon solution than a hard-wired solution where the functionality of the device is not under your control.

Easily Updated

The Spartan-II family easily accommodates specification changes and can be used in high-volume production. Conflicting specifications and lack of a clear design direction—conditions often unavoidable in the early stages of product development—create the need for programmable ASSP solutions. For example, suppose a new Reed-Solomon encoding-decoding solution is found to provide a much better code gain than the version implemented in a released device. It would be cost-prohibitive for an ASSP vendor to cater to every new specification change—but at the same time, betting on the success of only a single product might preclude them from being successful in the marketplace.

These conditions create many opportunities for the Spartan-II family, the industry's first *programmable* ASSP. Process technology has allowed Xilinx to offer 100,000 gates for less than $10, and this in turn allows designers to easily continue using programmable ASSPs in volume production.

Through the Xilinx On-line program, the Spartan-II family allows designers to gain market share by bringing themselves to market much sooner than a stand-alone ASSP. Spartan-II FPGAs are based on SRAM technology and are customized by loading configuration data into internal memory cells and therefore are very easy to re-program in an unlimited number of times. Updating the functionality of an FPGA only requires that the designer include a mechanism for updating the configuration bitstream. Remotely updating software with any new enhancements and bug fixes, increases the life of the Reed-Solomon solution within any product. Designing systems that do remote upgrades can also provide new revenue opportunities. After the initial product is released, new hardware features can be developed, sold and distributed inexpensively to existing customers in much the same way as new versions of software can be distributed today. In addition, a standard "off-the-shelf" application can be developed so that features can be swapped in and out depending on what the end-user purchases or needs. The designer can hence take advantage of the fact that the solution now allows them to upgrade their hardware and stay in the market-place longer, thus maximizing profitability.

Spartan-II: The Clear Reed-Solomon Solution Winner

Significant features like value proposition against conventional standalone ASSPs, accommodation of specification changes, improved testing and verification, and field upgradeability as discussed above, show the advantages presented by Spartan-II devices (Table 1). The cost difference between a stand-alone ASSP and an equivalent Spartan-II solution is also considerable. Therefore, the Spartan-II family is a clear winner in not only the Reed-Solomon solution market, but also in a variety of other traditional standalone ASSP areas.

Table 1: **Spartan-II vs. Stand-alone Comparison**

Feature Comparison	Typical Reed-Solomon ASSPs	Reed-Solomon in Spartan-II
Polynomial	Fixed	Parameterizable
Symbol Width	Fixed	Parameterizable
Block Length	Programmable (3-255)	Parameterizable (3-4095)
Correctable Errors	Programmable (1-10)	Parameterizable (1-64)
Erasure Handling	Fixed	Parameterizable
Maximum Throughput	12.5 MBytes/sec	Decoder: 62 Mbytes/sec Encoder: 108 Mbytes/sec
Latency	1276 cycles	3 to 756 cycles
Cost (in 250k units)	$20	$9.95(1)

Notes:

1. With Reed-Solomon configuration comparable to listed typical ASSP example. The price is based on 250KU resale price for XC2S100.

Xilinx clearly stands out in the comparison above as a clear winning choice, both in price and in the ability to outperform the typical ASSP solution.

Figure 8 shows the cost savings realized by using the Spartan-II programmable ASSP.

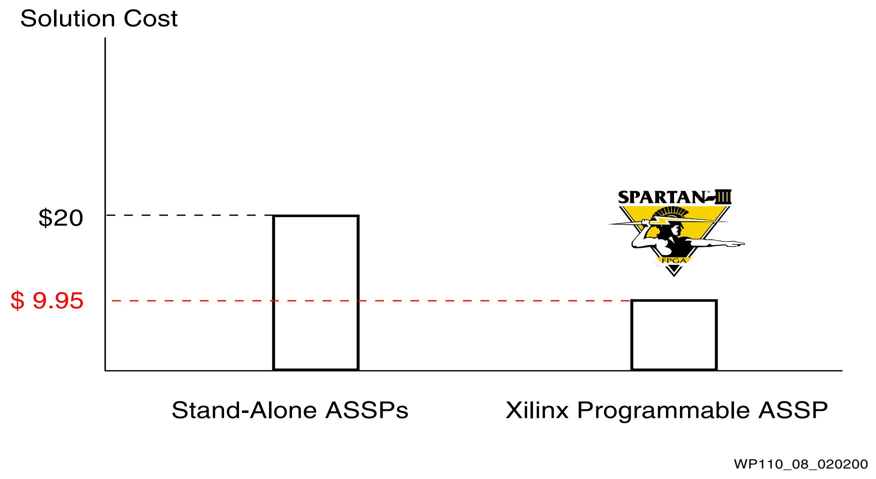

Figure 8: **Spartan-II vs Stand-alone ASSPs**

Conclusions

Reed-Solomon solutions are deployed in a vast number of different applications. Built on the capabilities of the Xilinx Virtex family of high-end FPGAs, powerful and cost-effective Spartan-II devices broaden the Spartan family's profile in competing against ASICs, and are uniquely poised to penetrate the ASSP marketplace. A Spartan-II FPGA-based Reed-Solomon solution with efficient partitioning of hardware and software functions provides the necessary scalability and flexibility to handle all types of applications.

References

The Spartan-II Family – The Complete Package by Krishna Rangasayee

Memec Design Services

Integrated Silicon Systems

Error Control Coding by Shu Lin and Daniel Costello

Error Control Systems for Digital Communication and Storage by Stephen Wicker

Reed-Solomon Code and the Exploration of the Solar System by Robert McEliece and Laif Swanson

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
02/02/00	1.0	Initial Xilinx release.
02/10/00	1.1	Updated first paragraph.