An Adaptive Reed-Solomon Errors-and-Erasures Decoder

Lilian Atieno, Jonathan Allen, Dennis Goeckel and Russell Tessier
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003
tessier@ecs.umass.edu

ABSTRACT

The development of Reed-Solomon (RS) codes has allowed for improved data transmission over a variety of communication media. Although Reed-Solomon decoding provides a powerful defense against burst data errors, the significant circuit area and power consumption of customized RS decoder hardware can be limiting for embedded computing environments. To support enhanced performance decoding with minimal power consumption, a dynamically-reconfigurable FPGA-based Reed-Solomon decoder has been developed. Our errors-and-erasures decoding system uses multiple erasure blocks to identify the location of likely corrupted data and multiple decoders to attempt error correction. The RS decoder design is implemented in reconfigurable hardware to leverage architectural parallelism and specialization. Run-time dynamic reconfiguration of the decoding system is used in response to variations in channel conditions to support the fastest possible data rate while, as a secondary metric, minimizing decoder power consumption. Algorithm parameters for the decoding system have been determined via simulation and the design has been implemented in Altera Stratix FPGAs. Through experimentation using an Altera 1S40 Stratix FPGA, we show that dynamic reconfiguration can result in an 14% performance improvement versus a non-reconfigurable decoder implementation. Comparisons with a Pentium IV microprocessor illustrate five orders of magnitude performance improvement.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]

General Terms

Design

Keywords

FPGA, power reduction, Reed-Solomon

FPGA'06, February 22–24, 2006, Monterey, California, USA.

1. INTRODUCTION

The use of forward error-correcting (FEC) codes in communication systems is an integral part of ensuring reliable communication [10] [16]. Although FEC-based Reed-Solomon (RS) decoding is effective in addressing burst errors, the significant circuit area of customized RS decoder hardware can be limiting for resource constrained implementation platforms. Most code-based communication systems are designed to meet certain reliability requirements in terms of codeword error rate (CER). Maintaining this desired transmission performance in the presence of significant variations in the quality of the channel requires a decoder implementation that exhibits flexibility. Our approach, hardware reconfiguration, permits run-time implementation changes in response to channel variation. Reconfigurable devices, such as field-programmable gate arrays (FPGAs), provide both the fine-grained parallelism and run-time reconfiguration capability needed to achieve desired performance and power levels for Reed-Solomon decoding.

Errors-and-erasures (e-and-E) RS decoders provide enhanced error recovery at the receiver by both identifying symbols likely to be in error (erasure generation) and correcting symbols [10]. To improve error correction capability, our decoding system employs an implementation enhancement to traditional errors-and-erasures RS decoding. Unlike previous errors and erasures decoders [13], our e-and-E system can employ *multiple* erasure generators and corresponding decoders in parallel. Each erasure generator is tuned to a specific error threshold level based on existent channel conditions. Appropriate erasure threshold levels for our e-and-E decoders have been determined via simulation. This approach provides for increased noise tolerance at constant codeword error rates (CER) versus previous single erasure generator designs, without the loss in raw decoder throughput of serial versions.

For significant run-time channel variations it is often difficult to maintain required CER for a fixed channel data rate, necessitating a rate adjustment. We address this issue by reconfiguring the FPGA-based design to include the decoder design which minimally meets the CER requirement while achieving the fastest possible data transmission rate. If less favorable channel conditions are detected, a more complex, reduced-rate decoder is swapped into the FPGA hardware to maintain the fixed CER. More favorable conditions result in the opposite effect. The possibility of dynamic reconfiguration based on channel variation is evaluated every few seconds to ensure that the decoder that best meets the required CER is present in the FPGA. To further promote

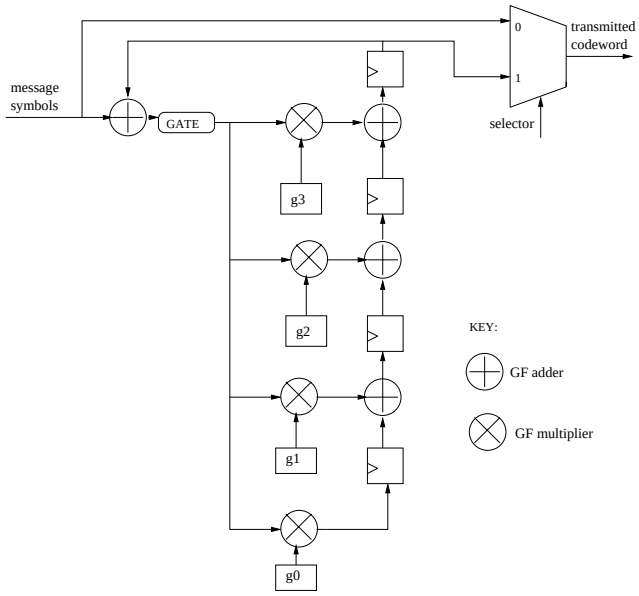

Figure 1: A Reed-Solomon encoder

design performance, FPGA block structures, such as dual-port memories and block multipliers are used in our adaptive e-and-E decoder design.

Following simulation to determine decoder parameters, a series of adaptive errors-and-erasures RS decoders were mapped to Altera Stratix FPGAs. It is shown that when dynamic reconfiguration is applied to our decoding system a decode rate performance improvement of 14% and a power savings of 24% is achieved versus a non-reconfigurable implementation. Additionally, the FPGA-based decoding system is also shown to provide improved performance (10000×) compared to a software implementation on a commercial Pentium IV microprocessor due primarily to design specialization and application-level parallelism.

The rest of the paper is organized as follows. Section 2 introduces Reed-Solomon codes and basic RS decoding techniques. Section 3 presents the architecture of our adaptive Reed-Solomon e-and-E decoder and the algorithm tradeoffs required for FPGA implementation. The experimental approach used for simulation and hardware test are described in Section 4 and experimental results and analysis are provided in Section 5. In Section 6, we contrast our approach to related work in the RS decoding area. Section 7 summarizes our efforts and offers directions for future work.

2. BACKGROUND

Reed-Solomon codes are non-binary linear block codes which are capable of correcting both random and burst errors [10]. RS codes are based on *Galois fields (GFs)* and operate on data symbols that consist of several bits [16]. Thus, in Reed-Solomon systems, information is transmitted in codewords which consist of n multi-bit symbols. This feature makes RS coding effective at correcting burst errors because error correction is performed at the symbol level. In each RS(n,k) codeword, a total of k symbols contain message symbols and $n-k$ symbols contain parity symbols. RS codes typically operate on GFs of $q = 2^m$ where m is the symbol size in bits and $n = 2^m - 1$ is the number of symbols per codeword. Galois fields support a variety of specialized arithmetic operations. The operations that are primarily used for RS coding over GF(256) include addition and subtraction on GF(2^8) (8 bit symbols) [10].

At the transmitter of the communication system, a Reed-Solomon *encoder* determines the value of the $n-k$ parity symbols that provide redundancy for the encoded message symbols. As seen in Figure 1, the value of the parity symbols are determined from the message symbols through a series of pipelined Galois Field multiplication and addition operations. These operations are performed using a GF polynomial ($g_0 \ldots g_3$), a predetermined characteristic function based on the number of required parity symbols. The complexity of both the RS encoder and decoder is a function of k. A smaller k value will require the processing of more parity symbols which results in more complex hardware. After parity generation, each codeword bit (both message and parity) is modulated to an analog format for channel transmission. Following channel transport (including possible fading and corruption by noise), analog values are demodulated back to digital format. A detailed example of RS encoding is provided in [3].

A Reed-Solomon decoder attempts to create an estimate of the transmitted data values from the demodulated version of the received waveform. In general, a RS decoder can correct any combination of errors and erasures as long as the number of erasures plus twice the number of remaining errors (after erasure) is less than $n-k$. A polynomial corresponding to the received codeword $r(x)$ can be represented as $u(x)+e(x)$ where $u(x)$ is the original transmission and $e(x)$ is the added error. For errors-and-erasures RS decoders, the determination of $e(x)$ takes place in two separate steps. Initially, in the erasure generator, sampled data symbols are compared against prespecified error thresholds using Baysian decision theory [4] [5]. The erasure threshold then generates a one-bit *erasure flag* per symbol to indicate to the subsequent *component decoder* if the symbol should be ignored. The component decoder determines the error vector $e(x)$ from the received symbols $u(x)$ and the erasure flags and adds the vector to the received symbols.

2.1 Errors-and-erasures Decoder Architecture

Figure 2 shows the functional blocks needed to implement an errors-and-erasures (e-and-E) RS decoder [10]. This system contains two distinct parts, an erasure generator and a component decoder. The erasure generator consists of six primary blocks. The **divide unit** determines the ratio of the noise standard deviation and channel fading. The two **estimation units** determine the most likely and second most likely symbols that were transmitted based on received channel bits and the **difference unit** evaluates the difference between the two. The **channel threshold unit** evaluates current noise and fading versus the preset threshold. Ultimately, the **decider** block uses the difference and channel threshold unit output to determine the symbol erasure flag.

As shown at the right in Figure 2, the determination of corrected codewords is performed in a sequence of eight subtasks [10]. The **syndrome generation block** determines if symbols are in error via GF operations and generates reliability output (a *syndrome polynomial*). The **erasure location extraction** block keeps track of which symbols have been flagged as erasures and the **syndrome expansion** block combines this information with the syndrome polyno-

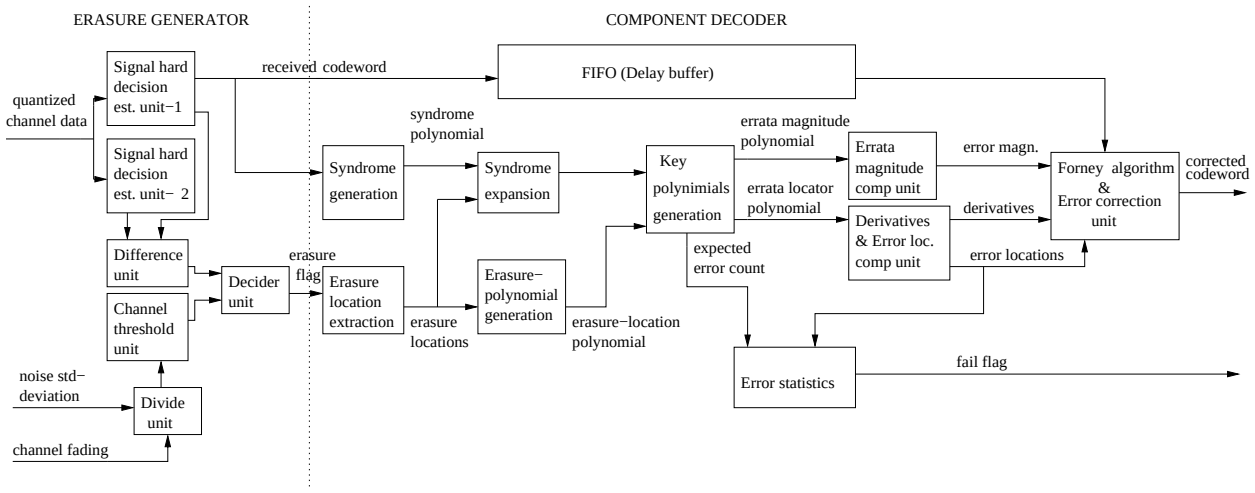

Figure 2: A general architecture of an errors-and-erasures Reed-Solomon decoder

mial. The erasure locations are represented in a form which can be used to determine error locations by the **erasure location polynomial generator**. The **key polynomial generation block** uses the syndrome polynomial to assist in determining the codeword location of the transmission errors and their associated magnitudes. The most common algorithm used to determine these values is the Modified Euclid Algorithm (MEA) [10]. The **errata magnitude** and **error location** units identify the specific erroneous symbols in the codeword and determine if the decoder will be unable to recover from the errors. The error magnitude vector to be added to the codeword is determined by the **Forney Algorithm and error correction block**, which adds the error vector $e(x)$ to the received codeword $r(x)$ to form the corrected codeword. It will be shown in Section 3 that when multiple erasure generators and component decoders are used per e-and-E decoder, many of these blocks can be shared without the need for replication.

3. ADAPTIVE ERRORS-AND-ERASURES ALGORITHM

Over time, the received signal power in communication systems can fluctuate due to external factors. For example, on wireless communication channels, the propagation distance, shadowing of the signal by large objects, and multipath fading all cause variations in the signal power. Although some of these impairments (e.g. multipath fading) often change too rapidly to allow for feasible system adaptation, others such as the propagation distance and shadowing are readily measured and can be made available to the transmitter and receiver [12]. Although operating parameters such as decode rate can be allowed to vary over time, decoding accuracy, in terms of codeword error rate, often must remain stable. If a fixed-k Reed-Solomon decoder is used (instead of a reconfigurable one), it must be designed to successfully achieve the desired CER even in cases of extreme signal loss. This fixed-decoder condition limits the data rate and often leads to increased power consumption.

Our adaptive e-and-E decoding approach uses two techniques to provide a flexible, constant CER Reed-Solomon decoding over time:

- To support small changes in channel signal-to-noise (SNR) while providing a consistent decode rate (fixed k), an RS decoding system is used which provides multiple errors-and-erasures component decoders operating in parallel. Each component decoder operates at a different erasure threshold level.
- For larger SNR variations, modifications in data rate are unavoidable. As a result, an encoder with modified k is employed at the transmitter, and a decoder with modified k is swapped into the FPGA hardware. This type of reconfiguration maintains the desired CER while maximizing the throughout rate of the system, and, as a secondary metric, minimizing the amount of power consumed by the erasure generation and decoding units. It is assumed a management channel exists to notify the transmitter (in addition to the receiver) of the SNR variation.

An *adaptive* errors-and-erasures decoder architecture which meets these requirements is shown in Figure 3. This architecture contains three main blocks: an erasure generation unit which contains multiple erasure generators, each with a different erasure threshold, a set of parallel Reed-Solomon component decoders, and a controller for dynamic reconfiguration. Each of the blocks require design at the algorithm and architecture level, which is described in subsequent subsections.

3.1 Multiple Erasure Generators and Component Decoders

To address minor signal power variations, we advocate the use of multiple erasure generator/component decoder pairs for a single channel, each tuned to a different erasure threshold. Since these D component decoders operate in parallel, decoding speed versus a single component decoder is not substantially affected. Since, for large symbol fields, Reed-Solomon decoders are able to tell reliably when they are unable to find the proper codeword, the Reed-Solomon decoding is successful if at least one decoder is able to produce the correct codeword.

A key part of using an RS decoding system with multiple erasure generators is the determination of appropriate

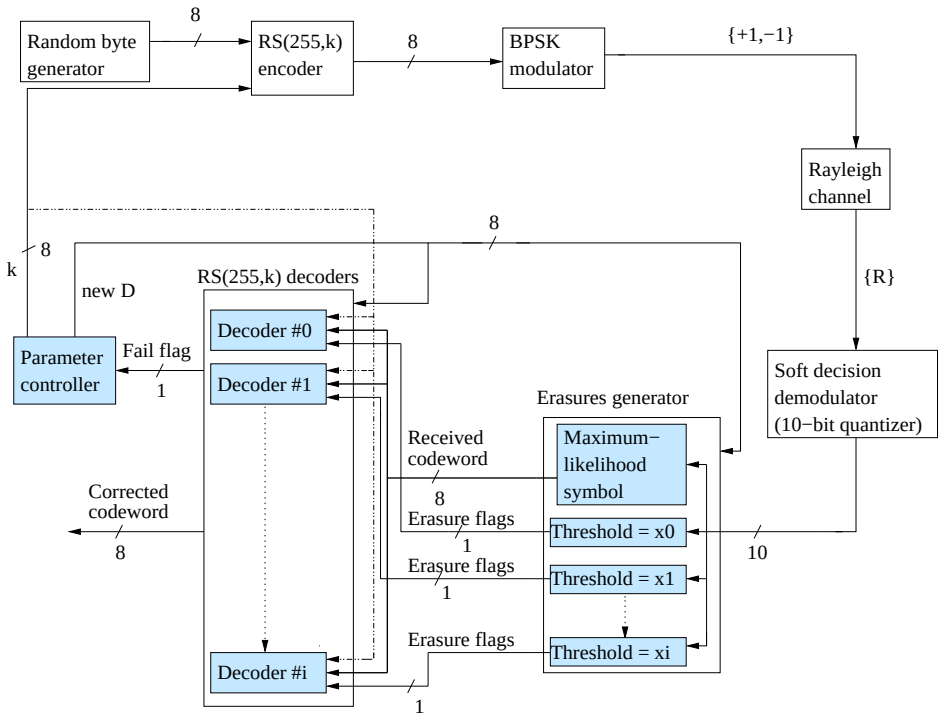

Figure 3: Adaptive Errors-and-Erasures Reed-Solomon Decoding System

erasure thresholds. As seen in Figure 3, each decoder receives the same stream of codewords from the demodulator, however, they receive different streams of erasure flags from the erasure generators. Threshold values for each erasure generator were determined through a series of steps.

Motivated by Baysian decision theory [5], the erasure flag for a given received symbol $\underline{r} = (r_0, r_1, \ldots, r_7)$ is asserted for a *single* component decoder if:

$$P(s_0|\underline{r}) \leq 1 - T \tag{1}$$

where $P(s_0|\underline{r})$ is defined as the probability that the most likely symbol s_0 is indeed correct given the received vector, and T is the threshold. Employing Bayes Rule and considering only the two most likely codewords in the denominator, which greatly simplifies the decoder with only minimal performance loss, yields:

$$P(s_0|\underline{r}) \simeq \frac{f(\underline{r}|s_0)}{\sum_{i=0}^{1} f(\underline{r}|s_i)} \tag{2}$$

where $f(.|.)$ represents the conditional probability density function of its first argument given its second. For fading channels with gain α, assumed to be measured perfectly at the receiver, and Gaussian noise of variance δ^2, it is straightforward to write (2) as:

$$\frac{e^{-\sum_{l=0}^{7}(r_l - \alpha * s_{0,l})^2/2\delta^2}}{\sum_{i=0}^{1} e^{-\sum_{l=0}^{7}(r_l - \alpha * s_{i,l})^2/2\delta^2}} \leq 1 - T \tag{3}$$

Equation 3 can be simplified to:

$$\sum_{l=0}^{l=7} r_l(s_{1,l} - s_{0,l}) \geq \frac{\delta^2}{\alpha} ln(\frac{1}{1-T} - 1) \tag{4}$$

No. decoders	Frac. of max. erasures generated
1	1.0
2	1.0, 0.75
3	1.0, 0.75, 0.5
4	1.0, 0.83, 0.67, 0.5

Table 1: Threshold fractions for multiple decoders per decoding system

where $s_{0,l}$ is the l^{th} value of the most likely received symbol and $s_{1,l}$ is the l^{th} value of 2^{nd} most likely received symbol. Flagging a byte that is truthfully correct with a low threshold T reduces the total errors a decoder can correct, while a high T allows corrupted symbols to pass undetected.

To allow for enhanced operation, threshold values for decoders were determined via simulation for a range of channel conditions. After single-decoder threshold values were determined, an iterative technique was used to determine the threshold levels for multiple decoders operating simultaneously. For a given δ and α, an initial T value is determined so that the maximum number of erasures is generated on average for sample data. The T values for *subsequent* erasure generators in the adaptive e-and-E decoder is based on the creation of a *fraction* of possible erasures. Fractional values are shown in Table 1 per the number of component decoders in each e-and-E decoder. After the determination of threshold values, the SNR coverage of each decoder was evaluated via simulation.

Figure 4 demonstrates the CER of RS(255,225) decoding systems with $D = 1$, 2, 3, and 4 component decoders. As shown in the figure, the more component decoders, the lower the CER for a given signal-to-noise ratio.

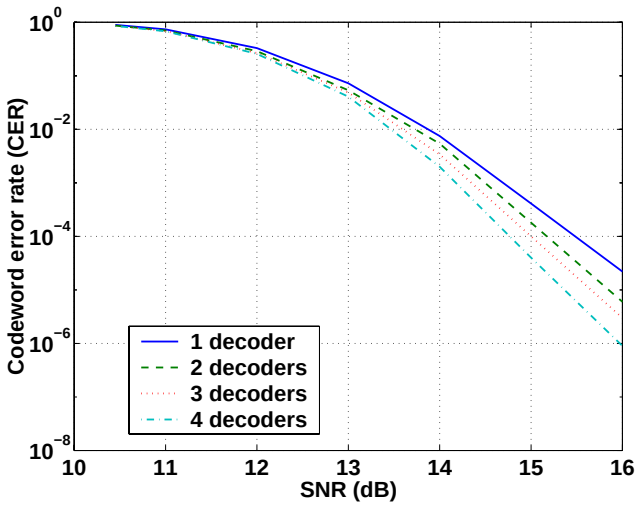

Figure 4: Performance of RS(255,225) decoders

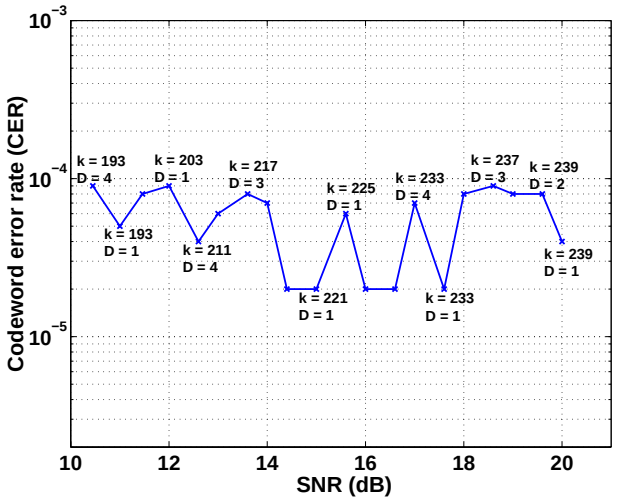

Figure 5: Variation of k parameter of RS code to achieve codeword error rate of 10^{-4}

3.2 FPGA-based Dynamic Reconfiguration

To allow for decoder hardware changes in response to variable channel noise conditions, dynamic reconfiguration is applied to our architecture. The run-time reconfiguration of our adaptive errors-and-erasures decoder allows for enhanced performance and reduced overall power consumption without compromising decode accuracy (CER). To amortize power and performance overhead, the channel SNR value is evaluated following the receipt of each 255,000,000 bit sequence (125,000 codewords). Based on this estimate, the adaptive errors-and-erasures decoder that has the highest possible data rate while achieving the required CER is used. As shown in Figure 5, a CER of 10^{-4} can be achieved using decoders with a range of k and D values. For small changes in SNR, k can remain fixed while the number of erasure/decoder pairs is updated in hardware. This change does not require changes to the encoder. For larger SNR variations, an encoder and decoding system with a different k must be used.

3.3 Decoding System Design

The implementation of multiple decoders for the same channel does not require complete hardware replication of all erasure generator and component decoder blocks. As shown in Figure 6, since the input to all erasure generators is the same, all hardware blocks can be shared across generators except for the **channel threshold units** and **decider units**, which generate the erasure flags. Since these flags are different across parallel RS decoders, fewer hardware blocks can be shared across component decoders.

Figure 6 shows that only the **erasure location extraction**, **syndrome generation**, and **FIFO** units can be shared across decoders. The selection of the appropriate corrected codewords can only occur after all iterative key polynomial generation employing the MEA (key polynomial generator) have completed. The number of iterations performed by the MEA is directly tied to the effectiveness of erasure generation. The **error statistics** block compares the expected and actual error count for each MEA to determine if the counts match. The decoder path with the exact match is subsequently corrected via the **errata magnitude computation** unit and **error correction** unit and a corrected codeword is produced.

The use of specialized FPGA resources, such as embedded multipliers and memory blocks, aids the efficient implementation of our adaptive errors-and-erasures decoder. In the erasure generator, embedded multipliers are used to scale threshold values by noise standard deviation and fading parameters in the **channel threshold unit**. FPGA dual-ported memories are used extensively throughout the design. DPRAMs are used to store codeword and erasure flags in the erasure generator. Intermediate error location and magnitude values are stored in DPRAMs in each decoder.

4. EXPERIMENTAL APPROACH

4.1 Test Platform

To evaluate the performance of our adaptive errors-and-erasures decoding system, a hardware implementation was tested as part of a multi-block model (Figure 3) of a communication system. A *random byte generator* creates a byte sequence to model transmitted data. A *Reed-Solomon encoder* (Figure 1) receives the message bytes and uses them to generate parity bytes and generate a codeword. The encoder is parameterized for RS(255,k). The encoder transmits codewords through a channel simulator that applies a Rayleigh fading variable to the signal power and then includes additive white Gaussian noise (AWGN); in other words, the standard Rayleigh fading channel model is employed. The simulator performs binary phase-shift keyed (BPSK) modulation which converts coded bits to analog values: 0 to 1, 1 to -1. Symbols obtained from the channel simulator are quantized to 10 bits before being sent to the *erasure generator unit(s)* and *component decoder(s)* as input. All software modeling of the communication system (except for the FPGA-based decoding system) was performed using a 1.6 GHz Pentium IV PC.

4.2 Hardware Implementation

Our adaptive errors-and-erasures decoder was mapped to Altera Stratix FPGAs. A sample decoder was mapped to a

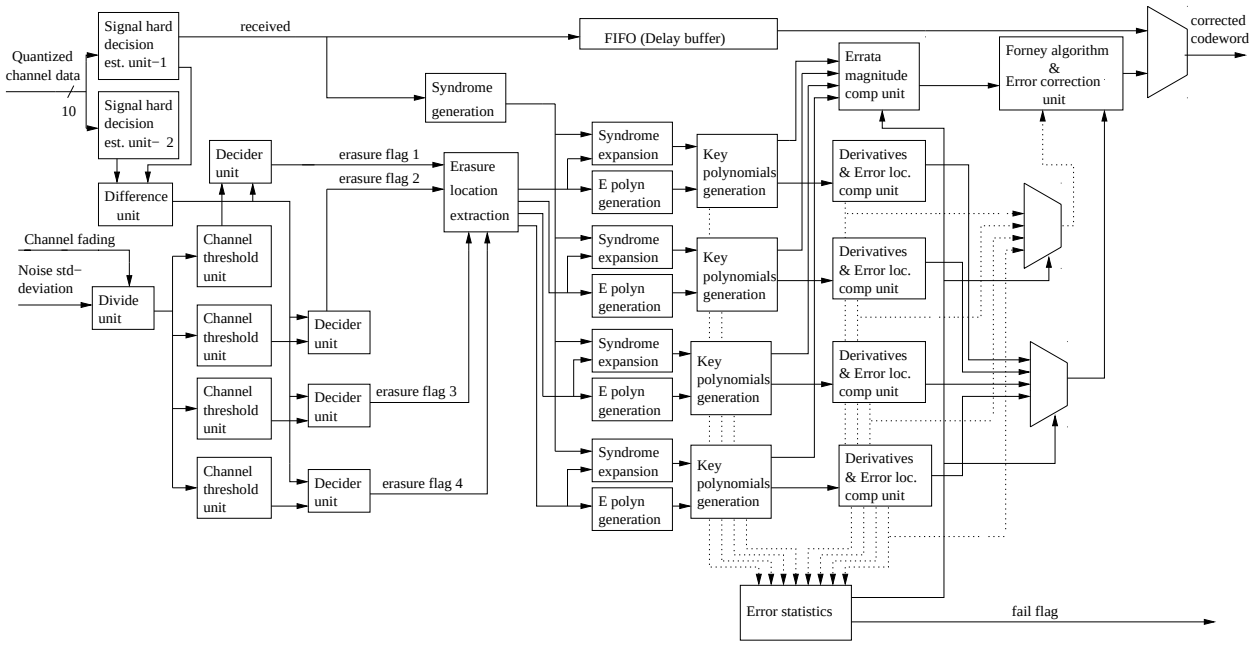

Figure 6: *Ch = 4* Parallel RS decoders and erasures generator system

k	D	T	SNR (dB)	LUTs	FFs
239	1	0.15	19.6-20.0	7,056	1,921
239	2	0.15, 0.28	19.4-19.6	11,644	2,925
239	3	0.15, 0.28, 0.32	19.2-19.4	16,459	3,922
237	1	0.20	19.0-19.2	7,608	2,207
237	2	0.08, 0.22	18.8-19.0	12,699	3,119
237	3	0.04, 0.22, 0.35	18.6-18.8	17,967	4,197
233	1	0.21	17.6-18.6	8,711	2,251
233	2	0.04, 0.21	17.4-17.6	14,553	3,503
233	3	0.04, 0.18, 0.26	17.2-17.4	20,544	4,744
229	1	0.21	16.4-17.2	9,710	2,475
229	2	0.21, 0.27	16.2-16.4	16,550	3,960
225	1	0.21	15.6-16.2	11,007	2,736
225	2	0.04, 0.22	15.4-15.6	18,580	4,344
225	3	0.04, 0.22, 0.30	15.2-15.4	26,652	5,935
221	1	0.21	14.8-15.2	12,102	2,962
221	2	0.04, 0.21	14.6-14.8	20,301	4,732
221	3	0.04, 0.21, 0.30	14.4-14.6	28,821	6,421
217	1	0.21	14.0-14.4	13,155	3,186
217	2	0.04, 0.21	13.8-14.0	22,175	5,116
217	3	0.10, 0.24, 0.34	13.6-13.8	31,567	6,965

Table 2: Decoder statistics for CER=10^{-4}

Stratix EP1S10 FPGA located on an Altera NIOS Development Board [1] to verify decoder functionality with a series of test vectors. Additionally, a range of decoders, described in Section 5, were mapped to a Stratix EP1S40. An RTL description of the adaptive errors-and-erasures decoder was written in Verilog and mapped to FPGAs. Verilog code was simulated using the Altera Quartus II simulator and all designs were synthesized and mapped using Quartus II with timing constraints.

Power consumption values for the decoders were determined using the Quartus II power analyzer. To account for power consumption during EP1S40 reconfiguration, the power associated with reading the configuration bitstream from SDRAM and storing it in the FPGA was calculated. It was determined that approximately 438 mW of power are needed during reconfiguration to read the 12,389,632 EP1S40 configuration bits from 4M×32 Micron SDRAM [11]. This value was determined by scaling the specified maximum power dissipation at 200 MHz by the required FPGA reconfiguration speed. The amount of power required to reconfigure the EP1S40 was approximated by assuming the use of an on-chip reconfiguration shift chain. The power dissipated by the shift chain was determined by calculating the energy dissipated by a single shift in 0.13 μm technology with SPICE. This shift chain energy value was scaled by the required 12,389,632 shifts and divided by configuration time to calculate FPGA reconfiguration power. It was calculated that 192.1 mW are required to reprogram the configuration bits of the EP1S40. Total EP1S40 reconfiguration time is 32 ms [2] at 50 MHz.

5. RESULTS

5.1 Parameter Evaluation

Prior to implementing the adaptive RS decoder in hardware, a set of simulations was performed to determine appropriate T values. Using the simulation technique mentioned in Section 3.1, T values for each target decoder (shown in Table 2) were determined for a fixed CER of 10^{-4}. The signal-to-noise ratio (SNR) range supported by each tested decoder is shown in Table 2. For each decoder a $(255,k)$ RS code was used. For decoding systems with multiple erasure generators, thresholds are listed in order of most erasures covered. As an example, Figure 7 shows the probabilities of incorrect decoding (CER) at $SNR = 15.8$ as T is varied

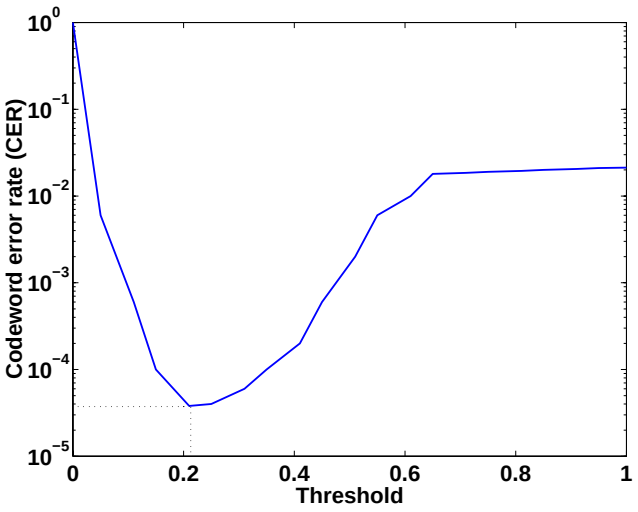

Figure 7: Variation of decoding error probability with respect to threshold, T for AWGN $\delta = 0.16$ (SNR$\simeq 15.8$ dB), $k = 225$ and $D = 1$

between 0 and 1. A RS(255,225) decoder was used for these simulations. The optimal value of $T = 0.21$ is shown with the dashed line.

5.2 Adaptive Decoder Implementation

To test the power consumption and decoding speed of our adaptive e-and-E Reed-Solomon decoders, a parameterizable decoder was written in Verilog. Decoders for a variety of k and D values were synthesized to Altera Stratix EP1S40 and EP1S10 FPGAs. In the following experiments, (255,k) decoders were tested.

Table 2 illustrates the hardware resource usage of the decoders. Table 3 shows the decode rate and power consumption of the decoders for a range of k and D values. All decoders were found to successfully operate at clock frequencies of between 64-68 MHz, although the statistics shown in the table are capped at 50 MHz, the FPGA operating frequency of the Altera NIOS development board used to verify our design. For each k value, multiple component decoders may be used. In general, for communication systems, it is desirable to maintain a constant decode rate across decoders with the same k so that a fixed *encoder* for each k is needed regardless of the number of component decoders. Since multiple-decoder systems require more clock cycles per decode, the clock rate of systems which contain fewer component decoders have been reduced incrementally from 50 MHz. The phase-locked loop (PLL) circuitry inside the Stratix FPGA can be used to customize the clock frequency.

For the decoding systems, it can be seen that decode rate and power consumption varies roughly with k and D, the number of decoders. For example, as k varies from 239 to 217, for $D = 1$, decode rate is reduced by 22% from 227 Mbps to 177 Mbps and power consumption increases by 32% from 197.0 mW to 259.4 mW. Due to increased decoding system size, power increases can also be seen as D increases for fixed k. For example, for a (255,239) decoding system power increases by 19% (197.0 mW to 234.2 mW) as D increases from 1 to 3. Other power increases with increased D are similar (between 19% to 25%). All designs listed in the table were targeted to a Stratix EP1S40-5.

k	D	Clk freq (MHz)	Decode rate (Mbps)	Power (mW)	Pentium IV Decode rate (Kbps)
239	1	43.4	227	197.0	4.2
239	2	47.9	227	209.6	2.1
239	3	50.0	227	234.2	1.4
237	1	45.6	225	204.0	3.8
237	2	47.8	225	216.7	1.9
237	3	50.0	225	245.4	1.3
233	1	46.0	219	205.3	3.1
233	2	48.7	219	238.6	1.6
233	3	50.0	219	263.6	1.1
229	1	44.5	212	223.0	2.4
229	2	50.0	212	246.7	1.2
225	1	46.7	205	237.9	2.0
225	2	47.5	205	258.5	1.0
225	3	50.0	205	300.1	0.7
221	1	42.5	184	252.1	1.8
221	2	47.2	184	271.3	0.9
221	3	50.0	184	316.8	0.6
217	1	44.0	177	259.4	1.6
217	2	47.7	177	283.5	0.8
217	3	50.0	177	326.4	0.5

Table 3: Decoder Performance on a Stratix EP1S40 FPGA for CER=10^{-4}

To test the functionality of the decoder architecture, a Stratix EP1S10-6 based NIOS Development Board was targeted. A (255,243) adaptive RS decoder was mapped using Quartus II tools and the resulting configuration was loaded onto the board. The results obtained by decoding a series of test codewords matched those achieved via simulation. A decode rate of 262 Mbps was achieved. To highlight the performance benefits of our adaptive Reed-Solomon decoder versus a software implementation, a C program with the same RS functionality was developed. Software results were determined using a 1.6 GHz Pentium IV PC (the host for the NIOS board). Decode rates are shown at the right in Table 3. These results indicate a performance improvement of nearly 10000× for the FPGA versus software implementations.

5.3 Dynamic Reconfiguration

A second set of experiments were used to determine power savings that could be achieved if the entire FPGA decoder was reconfigured at run time to support changes in channel SNR requirements. Periodically, the SNR is sampled to determined if decoder reconfiguration may be beneficial. Three reconfiguration scenarios are possible:

- If the sampled SNR falls within the acceptable SNR range of the current decoder no reconfiguration is necessary.
- If the sampled SNR falls within the range of a decoder with the same k but a different D as the current decoder, the new decoder is swapped into the FPGA. Since k, transmission and decode rate remain constant, no reconfiguration of the encoder is required. Depending on whether a larger or smaller D is used, the power

Reconfigs with constant k	2343
Overall reconfigs	8972
Average decode rate	202.0 Mbps
Average power	249.5 mW

Table 4: Dynamic reconfiguration statistics (for 10,000 potential reconfigurations)

consumed by the decoder may be either increased or decreased.

- If the sampled SNR falls outside the SNR range of the current decoder's k, a new decoder can be swapped into the FPGA and the k value of the encoder is updated. If noise increases, a smaller k decoder is needed, while the opposite is true for a noise increase.

Note that if reconfiguration was not possible, the channel decode rate would be limited to the rate associated with the worst possible SNR ($k = 217$) for the decoder.

A set of 10,000 SNR values were generated using a log-normal shadowing distribution [14] to test a total of 2.55 trillion bits with a desired CER of 10^{-4}. Based on the assumption that SNR can be sampled every 255M bits (once approximately every 1.5 seconds), the FPGA was periodically reconfigured during the transmission process. Table 4 shows the number of reconfigurations for fixed k and total reconfigurations out of 10,000 possible reconfigurations, the resulting average decode rate, and the average power dissipated. The average data rate across all decoders is 202.0 Mbps, a 14% improvement over a fixed $k = 217$ decoder. The average power consumption of 249.5 mW is 24% less than the power of the $k = 217$, $D = 3$ decoder. Power and decode rate numbers include the time and power needed for FPGA reconfiguration and the time and power needed to read associated configuration bits from SDRAM, as described in Section 4.

The circuitry required to determine SNR and associated k and D values is assumed to be located external to the decoder. Upon detection of an SNR change, this circuitry sends new k and D values to the decoder (and, if necessary, the encoder) to initiate reconfiguration.

6. RELATED WORK

Although adaptive Reed-Solomon coding has been explored [9], hardware implementations which can take advantage of parameter variations are limited. In Lee [8], a decoding system which provides multiple decoders for multiple channels is outlined. Since the basic MEA cell for this design is replicated, the MEA blocks consume 80% of decoder area. In a DVD-specific single-channel implementation [7], multiple MEA blocks are used to enhance decoding with an Altera Flex10K200 device. Our decoding system uses a single MEA processing cell, which is used recursively. Since our decoder has adaptive erasure capability, the additional MEA run-time overhead is minimal. An adaptive RS decoding technique, presented in [15], allows both variation in n and k at run-time. Since dynamic reconfiguration is not used, the largest decoder must always be present in the Altera APEX20KE hardware (17,608 LUTs used). This approach also does not support erasures. In Haase et al [6], FPGA dynamic reconfiguration is used to implement portions of a single RS decoder in an FPGA at different times. Given the time required for reconfiguration, it is impractical to reconfigure in this fashion for each received codeword. An FPGA-based errors-and-erasures decoder is outlined in [13]. In contrast to our technique, this decoder uses serial polynomial expansion and does not support run-time dynamic reconfiguration.

7. CONCLUSION

In this paper we have presented an adaptive errors-and-erasures Reed-Solomon decoder. The main algorithmic innovation is the development of a single channel decoding system which contains multiple erasure generators and component decoders. For successful data recovery, only one of the generator/decoder pairs must generate a successful result. Threshold parameters for the decoder have been determined via simulation. The key to improved performance is the use of dynamic reconfiguration based on the periodic sampling of channel noise conditions. Through experimentation it is shown that 14% performance improvement can be achieved by reconfiguring the decoder at run-time rather than requiring a static implementation of a higher-complexity, higher power decoder. Our decoder has been verified in hardware using an Altera NIOS Development Board containing a Stratix FPGA. In the future, we plan to consider approaches to make the adaptive RS decoder partially reconfigurable.

8. ACKNOWLEDGMENTS

The work was funded in part by National Science Foundation grants CCR-9875482, CCR-9988238, EIA-0080119 and ECS-0300130 and a grant from M/A-COM. The authors wish to thank Altera for the donation of the NIOS Development Board and Quartus II software.

9. REFERENCES

[1] Altera Corporation. *Nios Stratix Development Kit*, July 2003.

[2] Altera Corporation. *Stratix Data Sheet*, May 2003.

[3] L. Atieno. *Run-time Dynamically Reconfigurable Reed-Solomon Decoder System*. Master's thesis, Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, Feb. 2005.

[4] C. W. Baum and M. Pursley. Bayesian generation of dependent erasures for frequency-hop communications and fading channels. *IEEE Transactions on Communications*, 44(12):1720–1729, Dec. 1996.

[5] C. W. Baum and C. S. Wilkins. Erasure generation and interleaving for meteor-burst communications with fixed-rate and variable-rate coding. *IEEE Transactions on Communications*, 45(6):625–628, June 1997.

[6] A. Haase, M. Boden, and M. Langer. Design of a Reed Solomon Decoder Using Partial Dynamic Reconfiguration of XILINX VIRTEX FPGAs - A Case Study. In *Design, Automation and Test in Europe*, Mar. 2002.

[7] D. Lee, S. Lee, and J. Kim. A Reed-Solomon decoder with efficient recursive cell architecture for DVD application. In *IEEE International Conference on Consumer Electronics*, pages 184–185, 2001.

[8] H. Lee, M. Yu, and L. Song. High-speed VLSI architecture for parallel Reed-Solomon decoder. *IEEE Transactions on VLSI Systems*, 11(2):288–294, April 2003.

[9] S. Li, K. Pan, J. Yuan, A. Vigil, and A. Berg. Adaptive Reed-Solomon coding for wireless ATM communications. In *IEEE Southeastcon*, pages 27–30, Apr. 2000.

[10] S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1983.

[11] Micron Technology, Inc. *MT48LC4M32B2 SDRAM Data Sheet*, Apr. 2003.

[12] S. Nanda, K. Balachandran, and S. Kumar. Adaptation techniques in wireless packet data services. *IEEE Communications Magazine*, 38(1):54–64, Jan. 2000.

[13] K. Oh and W. Sung. An Efficient Reed-Solomon Decoder VLSI with Erasure Correction. In *IEEE Workshop on Signal Processing Systems*, pages 193–201, Nov. 1997.

[14] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, Upper Saddle River, NJ, 1996.

[15] M. K. Song, E. B. Kim, H. S. Won, and M. H. Kong. Architecture For Decoding Adaptive Reed-Solomon Codes with Variable Block Length. *IEEE Transactions on Consumer Electronics*, 48(3):631–637, Aug. 2002.

[16] S. B. Wicker and V. K. Bhargava. *Reed-Solomon Codes and Their Applications*. IEEE Press, Piscataway, NJ, 1994.