VLSI Design OF A Reconfigurable Multi-mode Reed-Solomon Codec for High-Speed Communication Systems

Huai-Yi Hsu
Graduate Institute of Electronics Engineering
National Taiwan University
Taipei 106, Taiwan, R.O.C.
yuki@access.ee.ntu.edu.tw

An-Yeu Wu
Graduate Institute of Electronics Engineering
National Taiwan University
Taipei 106, Taiwan, R.O.C.
andywu@cc.ee.ntu.edu.tw

ABSTRACT

This paper presents the VLSI design of a reconfigurable multi-mode *Reed Solomon* (*RS*) codec for various high-speed communication systems. Our decoder design is based on the Euclidean algorithm such that the datapath units are regular and simple. With its ability to support a variety of (n, k, t) RS specifications ($0 \le t \le 8$) and ($0 \le n \le 255$), this RS codec design is suitable for multi-mode systems such as the *xDSL* and the Cable Modem systems. The chip operations at a clock frequency of 100 MHz and has a data processing rate of 800 Mbits/s in 0.35um CMOS technology at the supply voltage of 3.3 V. The total gate count is 34,647 gates and the core size is only 1,578×1,560 um^2.

Keywords

Reed-Solomon codec, Irreducible Primitive, Filed Generator Polynomial, Code Generator Polynomial, Euclidean Algorithm, and multi-mode Chien's Search.

1. INTRODUCTION

Among various *Forward Error Correcting* (*FEC*) techniques [1], Reed Solomon codec is one of the most widely applied schemes for error correction. It provides excellent error correction capability for both random and bursty errors. Hence, RS has been employed in many practical applications, such as digital audio and video, magnetic and optical recording, computer memory, cable modem, xDSL [2] wireless and satellite communications systems.

Currently, Cable Modem and xDSL systems are the major broadband technologies in the market. For different specification of the transmission rates, various FEC specifications are specified to ensure the transmission quality. That is, various *RS*(n, k, t) specifications need to be performed in the transmission systems. Hence, in this paper, we propose the VLSI architecture of a reconfigurable multi-mode RS codec. It can be easily configured to perform specified (n, k, t) value in a unified VLSI architecture.

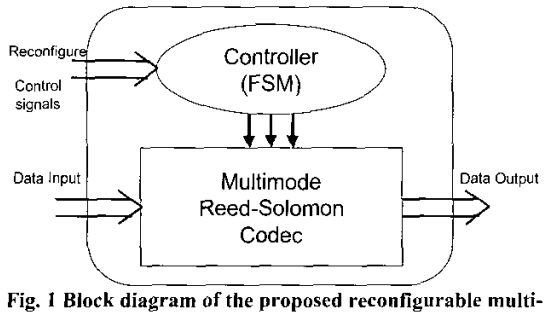

Fig. 1 Block diagram of the proposed reconfigurable multi-mode RS codec design.

The block diagram of our design is shown in Fig. 1. It consists of two parts, the *Softcore* and the *Hardcore*. The softcore is a configurable control unit. All the datapaths and dataflow of the hardcore will be controlled by the softcore. In our design, concurrent *Finite State Machines* (*FSM's*) are used to generate all control signals. We can input RS parameter (n, k, t) to reconfigure the states of the FSM's. The hardcore is a fixed operating datapath that is optimized for the arithmetic units in speed, area, and power. With such an arrangement, most performance indices are predicable in the layout-proven hardcore. On the other hand, the softcore can be run-time programmed to perform the desired RS specification. The proposed reconfigurable RS architecture is a scalable design, and can handle variable error correction capability ($0 \le t \le 8$) and variable codeword length ($0 \le n \le 255$).

2. DATAPATH DESIGN OF MULTI-MODE REED-SOLOMON CODEC

RS code is defined over the finite fields, or Galois fields GF(2^m). The elements of the filed are specified by an *Irreducible Primitive Polynomial* $p(x)$, called *Filed Generator Polynomial*. Each element in GF(2^m) can be represented by m-bits, or so-called *symbol*. An *RS*(n, k, t) code is a block code whose codeword are blocks of $n \le 2^m - 1$ symbols. Each codeword includes both k symbols of information and $r = n-k$ symbols of redundancy (or parity check). In order to correct $t = \lfloor r/2 \rfloor$ symbol errors, $2t$ parity symbols calculated and appended to a group of information symbols during the encoding process. The *Code Generator Polynomial* of the code is defined as

$$g(x) = \prod_{i=1}^{n-k} (x + \alpha^i), \qquad (1)$$

where α is a primitive element of the field GF(2^m), *i.e.*, it is a root of the filed generator polynomial.

2.1 The $a(x)$-based encoder design

In RS encoding, the n codeword symbols are generated form the k information symbols by using the code generator polynomial $g(x)$. A codeword can be obtained in a systematic form by adding $2t$ parity-check symbols. The codeword, $c(x)$, can be represented by a polynomial as follows:

$$c(x) = x^{2t}v(x) + r(x) = x^{2t}v(x) + x^{2t}v(x) \bmod g(x). \qquad (2)$$

The conventional RS encoder is a $2t$-stage *Linear Feedback Shift Register* (*LFSR*). The finite-field multiplier of the LFSR are set as the coefficients of the code generator polynomial $g(x)$. With such an approach, to deal with different the error correction capability t, the LFSR need to set/store different coefficient set of $g(x)$, which is not a scalable design. It transforms conventional encoder into a fractional form [4]. This architecture is designed with coefficient α^i, and we denote it as $a(x)$-basis encoder.

0-7803-7363-4/02/$17.00 ©2002 IEEE.

The $a(x)$-basis encoder has expandable property that is suitable for scalable design. The compares between $g(x)$-based and $a(x)$-based is shown the Table 1. The $a(x)$-based architecture has high regularity in the coefficients of the code generator polynomial. Therefore, it is very suitable to the hardware design concept of our reconfigurable circuit, as shown in Fig. 3. According to the different error correction capability has different definition in the code generator polynomial. We propose the scheme of the correction feedback selector, which can configure one suited feedback path for certain error correction capability to generate the corresponding codeword.

Table 1 The comparison of $g(x)$-based and $a(x)$-based.

		g(x) basis	a(x) basis
Area Cost	FFA	$2t$	$2t$
	FFM	$2t$	$4t$
	Reg.	$2t$	$2t$
Critical Path		1 x FFA 1 x FFM	$2t$ x FFA 1 x FFM
ROM Size		Larger	Smaller
Regularity		Low	High
Hardware Sharing		No	Yes (Syndrome)

2.2 Decoder design

The decoding procedure of RS code based on the syndrome-based architecture essentially consists of three modules, as shown in Fig. 2. In the first module, the syndrome polynomial of the received symbols is computed, which is used in the second module for solving the key equation. In the second component, we employ Euclidean GCD algorithm to solve the key equation. In the third module, the location and magnitude of a given error are calculated.

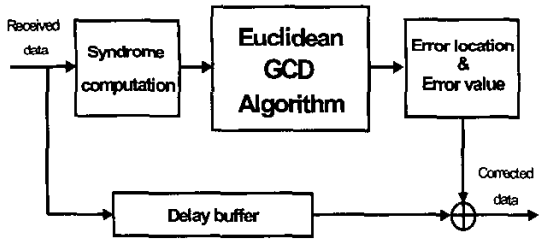

Fig. 2 Syndrome-based architecture of Reed-Solomon decoding.

2.2.1 Syndrome Calculation

Let polynomial $c(x)$ be a transmitted codeword. Then the received word $r(x)$ can be represented by

$$r(x) = c(x) + e(x), \tag{3}$$

where polynomial $e(x)$ denotes for error pattern whose coefficients are also from GF(2^m). Form Horner's rule [3], the coefficient of the syndrome polynomial, denoted by S_i, are obtained to evaluate the received polynomial $r(x)$. The equation can be written as

$$S_i = (\ldots(((r_{n-1}\alpha^i + r_{n-2})\alpha^i + r_{n-3})\ldots + r_1) + r_0. \tag{4}$$

Since the syndrome architecture is similar to the $a(x)$-basis encoding architecture. For our multi-mode RS codec, we combine both the encoder and the syndrome calculator in one hardware unit. One module include two elements has one error correction capability. We cascade t module to form the architecture is shown in Fig. 3.

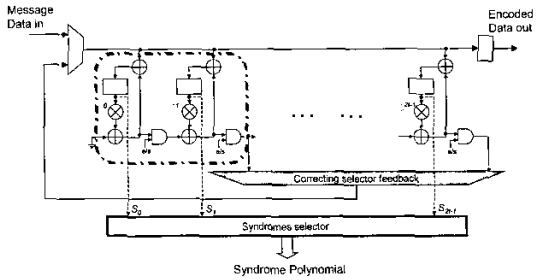

Fig. 3 Architecture for both encoding and syndrome calculation.

According to the different error correction capability, the RS specification has different definition in the syndrome polynomial. Hence, we propose the scheme of the syndromes selector, which can configure one syndrome polynomial according to the error correction capability. The output will enter to next step, which solve the key equation. The configurable datapaths is shown in Table 2.

Table 2 Truth table of the syndrome selector.

T_sel	So_0	So_1	So_2	So_3	So_4	So_5	So_6	So_7	So_8	So_9	So_{10}	So_{11}	So_{12}	So_{13}	So_{14}	So_{15}
$t=1$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Si_0	Si_1
$t=2$	0	0	0	0	0	0	0	0	0	0	0	0	Si_0	Si_1	Si_2	Si_3
$t=3$	0	0	0	0	0	0	0	0	0	0	Si_0	Si_1	Si_2	Si_3	Si_4	Si_5
$t=4$	0	0	0	0	0	0	0	0	Si_0	Si_1	Si_2	Si_3	Si_4	Si_5	Si_6	Si_7
$t=5$	0	0	0	0	0	0	Si_0	Si_1	Si_2	Si_3	Si_4	Si_5	Si_6	Si_7	Si_8	Si_9
$t=6$	0	0	0	0	Si_0	Si_1	Si_2	Si_3	Si_4	Si_5	Si_6	Si_7	Si_8	Si_9	Si_{10}	Si_{11}
$t=7$	0	0	Si_0	Si_1	Si_2	Si_3	Si_4	Si_5	Si_6	Si_7	Si_8	Si_9	Si_{10}	Si_{11}	Si_{12}	Si_{13}
$t=8$	Si_0	Si_1	Si_2	Si_3	Si_4	Si_5	Si_6	Si_7	Si_8	Si_9	Si_{10}	Si_{11}	Si_{12}	Si_{13}	Si_{14}	Si_{15}

2.2.2 Euclidean GCD Algorithm

Assume that ν symbols error occurred in data transmission; we can define the error location polynomial as

$$\sigma(x) = \prod_{l=1}^{\nu}(1 + xX_l) = \sigma_0 + \sigma_1 x^1 + \cdots + \sigma_l x^{\nu}. \tag{5}$$

Besides, we also can define the error magnitude polynomial, it denoted by $\omega(x)$, that can be written as

$$\omega(x) = \omega_0 + \omega_1 x^1 + \cdots + \omega_{l-1} x^{\nu-1}. \tag{6}$$

Furthermore, we define the syndrome polynomial, it denoted by $S(x)$, that can be written as

$$S(x) = \sum_{i=0}^{2t-1} S_i x^i = S_0 + S_1 x^1 + \cdots + S_{2t-1} x^{2t-1}. \tag{7}$$

We can obtain the error locator and error magnitude polynomials by solving the key equation, as shown in Eq. (8). The error locator and error magnitude polynomials can be obtained by using Euclidean GCD algorithm.

$$S(x)\sigma(x) = \omega(x) \bmod x^{2t}. \tag{8}$$

To solve the key equation, Euclidean GCD algorithm is an iteration procedure for finding the error location polynomial and the error magnitude polynomial [5]. The algorithm can be explained using the following iteration equations:

A. Initial conditions:

$$R_{-1} = x^{2t},\ R_0 = S(x),\quad B_{-1} = 0,\quad B_0 = 1. \tag{9}$$

B. In i-th iteration:

$$\begin{cases} R_i = R_{i-2} + R_{i-1}Q_{i-1} \\ B_i = B_{i-2} + B_{i-1}Q_{i-1} \end{cases}, \tag{10}$$

C. Stop condition:

$$\deg(R_i) < t. \tag{11}$$

The overall block diagram of RS decoder by used Euclidean GCD algorithm is shown in Fig. 4. The architecture consists of three parts, *Euclidean Divider Module (EDM)*, *Euclidean Multiply Module (EMM)*, and the *Error Magnitude Coefficient Selector (EMCS)*.

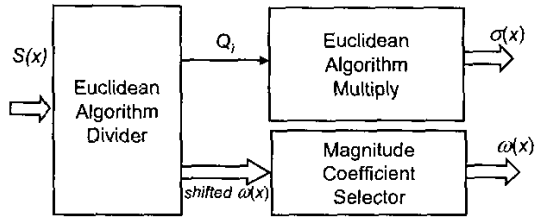

Fig. 4 Block diagram of the Euclidean architecture.

2.2.2.1 Euclidean Division Module

Euclidean division operation to compute Eq. (10) performs the division R_{i-2}/R_{i-1} in iterations. It generates the quotient Q_{i-1} and stores the new remainder R_i. The quotient Q_{i-1} is used in the Euclidean multiply module to compute Eq. (11). This polynomial division architecture includes two major components, the EAdivA and EAdivB modules, respectively. These architectures are showed in Fig. 5. The multi-mode Euclidean divider module architecture is shown in Fig. 6. For different error case, we can configure the EDM datapath to solve the key equation of error numbers equal to 1, 2,..., t, respectively, which are shown in Fig. 6(a)(b)(c)

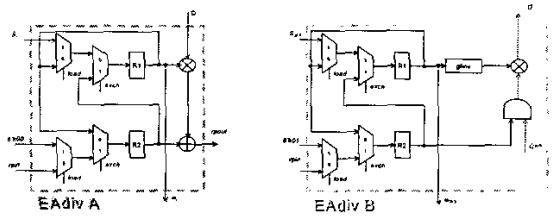

Fig. 5 EAdiv A and EAdiv B of the Euclidean divider module.

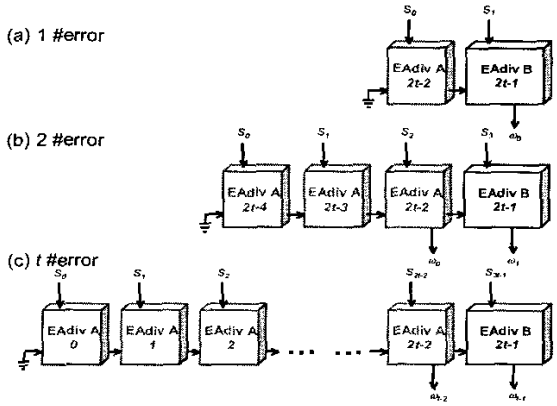

Fig. 6 The block diagram of the multi-mode EDM architecture.

2.2.2.2 Euclidean Multiply Module

Euclidean multiply operation to compute Eq. (11) performs the multiplication and accumulation in the polynomial domain. It is used to obtain error location polynomial $\sigma(x)$. The coefficients of the quotient $Q(x)$ are input sequentially from the highest coefficients. The polynomial multiply architecture includes one major component of the EAmulC module, as shown in Fig. 7. The multi-mode Euclidean multiply module is shown in Fig. 8. For different error case, we can configure the EMM datapath to solve the key equation of error numbers equal to 1, 2,..., t, respectively, which are shown in Fig. 8(a)(b)(c)

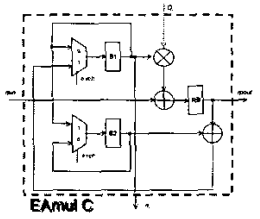

Fig. 7 Architecture of the EAmul C module in the Euclidean multiply operation.

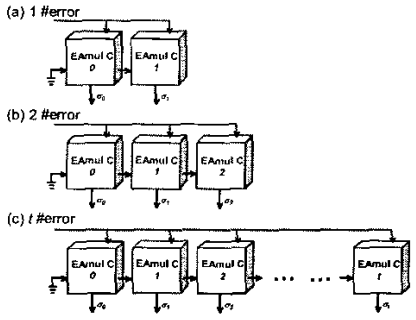

Fig. 8 The block diagram of the multi-mode EMM architecture.

2.2.2.3 Error Magnitude Coefficient Selector

We propose the scheme of error magnitude coefficient selector. This scheme can configure one error magnitude polynomial according to the error correction capability. The error magnitude polynomial enters the next stage to compute Chien search and Forney algorithm. The configurable datapath is shown in Table 3.

Table 3 Truth table of the error magnitude coefficient selector.

T_sel	ωo_0	ωo_0	ωo_0	ωo_0	ωo_0	ωo_0	ωo_0	ωo_0
$t=1$	ωi_{15}	0	0	0	0	0	0	0
$t=2$	ωi_{14}	ωi_{15}	0	0	0	0	0	0
$t=3$	ωi_{13}	ωi_{14}	ωi_{15}	0	0	0	0	0
$t=4$	ωi_{12}	ωi_{13}	ωi_{14}	ωi_{15}	0	0	0	0
$t=5$	ωi_{11}	ωi_{12}	ωi_{13}	ωi_{14}	ωi_{15}	0	0	0
$t=6$	ωi_{10}	ωi_{11}	ωi_{12}	ωi_{13}	ωi_{14}	ωi_{15}	0	0
$t=7$	ωi_9	ωi_{10}	ωi_{11}	ωi_{12}	ωi_{13}	ωi_{14}	ωi_{15}	0
$t=8$	ωi_8	ωi_9	ωi_{10}	ωi_{11}	ωi_{12}	ωi_{13}	ωi_{14}	ωi_{15}

2.2.3 Chien Search and Forney Algorithm

Chien search and Forney algorithm are used to calculate the error locations and magnitudes. The first step, the decoding procedure applies Chien search algorithm to find the roots of the error location polynomial $\sigma(x)$. The inverse of these roots is the error location of the received codeword. From Eq. (12),

we can deduct the cell module of Chien search as shown in Fig. 9(a).

$$\sigma(\alpha^l) = \sigma_0 + \sigma_1(\alpha^l) + \sigma_2(\alpha^l)^2 + \cdots + \sigma_v(\alpha^l)^v = \sigma_0 + \sigma_1(\alpha^1)^l + \sigma_2(\alpha^2)^l + \cdots + \sigma_v(\alpha^v)^l \quad (12)$$

In the shorten case ($n<2^8$-1), we introduced a *bias vector* (α^β) to improve the number of times of iteration of Chien search. Therefore, we can be reduced the latency. Moreover, this architecture of the bias vector can supply multi-mode selection after reconfiguring the datapath, as shown in Fig. 9(b). Simultaneously, we were computing the $\omega(\alpha^l)$ part. The overall architecture of multi-mode Chien search is shown in Fig. 10.

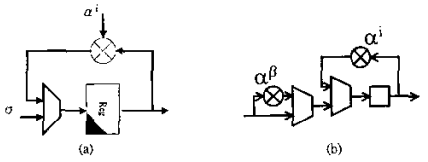

Fig. 9 (a) Cell module of Chien search. (b) Architecture of the multi-mode Chien search

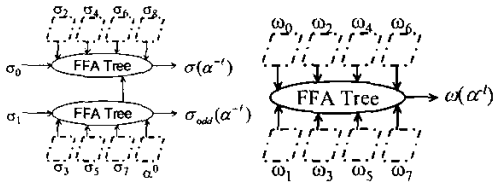

Fig. 10 The block diagrams of the multi-mode Chien search.

Forney's algorithm can be used to solve the error value, as follows:

$$e_l = \frac{\omega(\alpha^{-l})}{\sigma_{odd}(\alpha^{-l})}. \quad (13)$$

The corresponding Forney architecture is shown in Fig. 11.

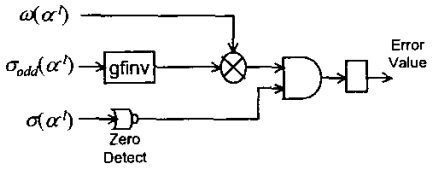

Fig. 11 Architecture of Forney algorithm.

Then we can add the error pattern and received codewords in the buffer memory to obtain the corrected codeword. Because our architecture of multi-mode RS codec is module-based design, this architecture can be very easily to be expanded for wide applications. Currently, our design has maximum error correction capability of 8 errors. Moreover, we support the shorten case for RS code, *i.e.*, codeword length $n<2^8-1$.

3. CHIP IMPLEMENTATION

We propose the VLSI architecture that has only 34,647 gate count and die size is only 2,653□2,653 um^2. From post-layout simulation, the chip can operate at a clock frequency of 100 MHz and has a data processing rate of 800 Mbps in 0.35um CMOS technology at 3.3 V. We make comparison between our design and other existing chip solutions as listed in Table 4. We can see from this comparison table that the speed performance of our multi-mode RS codec is the highest. For multi-mode system applications, our design can configure the error correction capability from zero to eight, and configure codeword length from zero to maximum of 255 values. The chip summary is shown in Fig. 12.

Table 4 The comparison table of hardware and performance.

	RS(255,239) [4]		RS(n, k, t) for ADSL 5]	Our Multi-mode RS(n, k, t) Codec
Gate count	122,630 Tr./4 ≅ 30,658 gate	220,841 tr./4 ≅ 55,210 gate	43,987 gate	34,647 gate (289x8 FIFO register)
Latency	287	321	$3mn+4mt+4m$ (Max: 7508)	$n+3t+10$ (Max: 289)
Clock Rate	41 MHz (0.25um)	75 MHz	48 MHz	100 MHz (0.35um)
Data Rate	330 Mbps	600Mbps	48 Mbps	800 Mbps
Reconfigurable Capability	n = 255 t = 8		$n = 2^m$-1 m □8, t□8 bit-serial	0 □n □255 0 □t□8

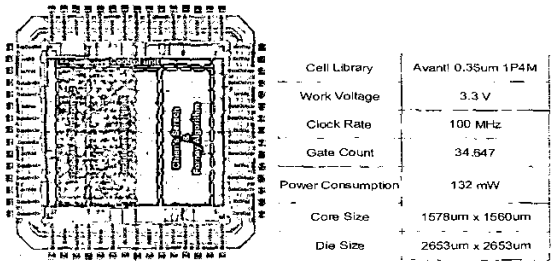

Fig. 12 The chip layout of the multi-mode Reed Solomon codec.

4. CONCLUSIONS

In this paper, we have developed VLSI architecture of the reconfigurable multi-mode Reed Solomon codec, which can be used in the high-speed communication systems and various specification of the applications, *i.e.*, *RS*(n, k, t). Our design consists of two components, the softcore and the hardcore. The softcore is a configurable control unit, and the hardcore is a fixed operating datapath. Hence, our multi-mode Reed Solomon Codec has many advantages of hard-IP, such as portability, reusability, predictability, and extensibility.

5. REFERENCES

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

[2] Dennis J. Rauschmayer, *ADSL/VDSL Principles: A Practical and Precise Study of Asymmetric Digital Subscriber Lines and Very High Speed Digital Subscriber Lines*, Macmillan Technical Publishing, Indianapolis, 1999.

[3] R. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley Co., 1983.

[4] G. Fettweis, M. Hassner, "A Combine Reed-Solomon Encoder and Syndrome Generator with Small Hardware Complexity," *Circuits and Systems, ISCAS 92 Proceedings*, vol. 4, pp. 1871-1874, 1992.

[5] H. Lee, M.L. Yu, and L. Song, "VLSI Design of Reed-Solomon Decoder Architecture," *ISCAS 2000 Proceedings Circuits and Systems*, pp. v-705-708, 2000.

[6] J. C. Huang, C. M. Wu, M. D. Shieh, and C. H Wu, "An Area-Efficient Versatile Reed-Solomon Decoder for ADSL," *ISCAS '99 Proceedings Circuits and Systems*, pp. 517-520, vol. 1, 1999.