A (204, 188) Reed-Solomon Decoder using Decomposed Euclidean Algorithm

Hsie-Chia Chang, Chih-Yu Cheng, Shu-Hui Tsai, and Chen-Yi Lee
Department of Electronics Engineering
National Chiao Tung University
Hsinchu, Taiwan, R.O.C

***Abstract*—A (204, 188) Reed-Solomon decoder for DVB application is presented. The RS decoder features an area-efficient Key Equation Solver using a novel *decomposed* Euclidean algorithm. We implement the RS decoder using 0.35μm CMOS 1P4M standard cells, where the total gate count is about 16K$\sim$ 17K. Test results show that the RS decoder chip can run up to 87MHz.**

I. Introduction

The DVB Project, which is used for broadcasting all kinds of data has very successfully developed a considerable list of specifications for Digital Video Broadcasting. In order to mitigate the errors, a (204, 188) Reed-Solomon Decoder is used in DVB for error correction. A (N, K) RS code contains K message symbols and $N-K$ parity checking symbols, and is capable of correcting up to $t = \lfloor \frac{N-K}{2} \rfloor$ symbol errors.

The most popular RS decoder architecture today, [1], [2], can be summarized into four steps: (1) calculating the *syndromes* from the received codeword, (2) computing the *error locator polynomial* and the *error evaluator polynomial*, (3) finding the error locations, and (4) computing error values. The second step in the four-step procedure involves solving the *key equation* [1], which is

$$S(x)\sigma(x) = \Omega(x) \bmod x^{N-K}, \tag{1}$$

where $S(x)$ is the syndrome polynomial, $\sigma(x)$ is the error locator polynomial and $\Omega(x)$ is the error evaluator polynomial.

Therefore, the RS decoder contains a Syndrome Calculator, a Key Equation Solver, a Chien Search, and an Error Value Evaluator as shown in Fig. 1. The Syndrome Calculator calculates a set of syndromes from the received codewords. From the syndromes, the Key Equation Solver produces the *error locator polynomial* $\sigma(x)$ and the *error evaluator polynomial* $\Omega(x)$, which are used by the Chien Search and the Error Value Evaluator to produce the error locations and error values, respectively.

This work was supported in part by the National Science Council in R.O.C, under Grant NSC89-2215-E-029-053.

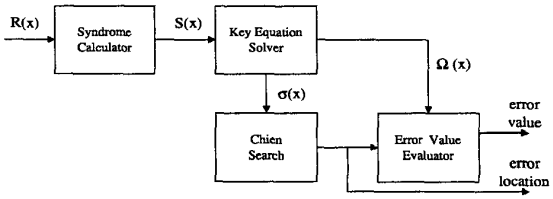


Fig. 1. Reed-Solomon decoding flowchart.

In this paper, we present a new design and implementation of a (204, 188) RS decoder for DVB application. For the Key Equation Solver, we developed a novel *decomposed* Euclidean algorithm which works with individual coefficients of the polynomial instead of the entire polynomial as a whole. The decomposed algorithm, which only needs *two* finite field multipliers (FFMs) is descirbed in Section II. Section III shows our proposed (204, 188) RS decoder for DVB application. Section IV describes our chip implementation and the conclusion is given in Section V.

II. The Decomposed Euclidean Algorithm

The techniques frequently used to solve the key equation include the Euclidean algorithm [2] and the Berlekamp-Massey algorithm [1]. In [3], a clever scheduling for the Berlekamp-Massey algorithm is proposed to reduce the circuit complexity. This serial architecture only needs three finite field multipliers (FFMs) to implement the Key Equation Solver. while other existing architectures, no matter for Berlekamp-Massey algorithm [4], [5], or for Euclidean algorithm [6], [7], they all require $2t \sim 3t$ FFMs where t is the number of correctable errors. Since the Key Equation Solver is the most complex part of the RS decoder, such an optimization procedure reaches 30% $\sim$ 40% hardware reduction for the entire RS decoder.

For illustrating our algorithm, firstly we rewrite (1) as

$$\Omega(x) = x^{2t}q(x) + S(x)\sigma(x) \tag{2}$$

where $q(x)$ is the quotient polynomial of x^{2t} and $S(x)\sigma(x)$. Hence, the determination of $\Omega(x)$ is equivalent to the process of computing the GCD polynomial of x^{2t} and $S(x)$

0-7803-6475-9/00/$10.00 ©IEEE 2000

by the Euclidean algorithm, which is shown as follows:

$$\begin{aligned} x^{2t} &= S(x)\cdot q^{(1)}(x)+\Omega^{(1)}(x) \\ S(x) &= \Omega^{(1)}(x)\cdot q^{(2)}(x)+\Omega^{(2)}(x) \\ &\vdots \\ \Omega^{(i-2)}(x) &= \Omega^{(i-1)}(x)\cdot q^{(i)}(x)+\Omega^{(i)}(x) \\ &\vdots \\ \Omega^{(n-2)}(x) &= \Omega^{(n-1)}(x)\cdot q^{(n)}(x)+\Omega(x) \\ \Omega^{(n-1)}(x) &= \Omega^{(n)}(x)\cdot \Omega(x) \end{aligned}$$

In errors-only RS code, $\deg(\Omega(x)) < \deg(\sigma(x)) \leq t$. Because the degree of $R^{(1)}(x)$ is less than $2t-1$ and each division eliminates degrees at least one, the total Euclidean algorithm requires at most t divisions.

Our proposed *decomposed* Euclidean algorithm works with individual coefficients of the polynomial instead of the entire polynomial as a whole. For simplicity to implement, we restrict every division to eliminate only one degree, the number of iterations to compute $\Omega(x)$ is t. Therefore, the quotient polynomial in each iteration must be $(q_1 x + q_0)$ and in the i-th iteration, it requires at most $2t-i$ cycles to compute all coeffients of $\Omega^{(i)}(x)$ because of $\deg(\Omega^{(i)}(x)) \leq 2t-1-i$. The calculating process for $\sigma(x)$ is similar to $\Omega(x)$ besides the minor difference of degrees [2]. The generalized formula for the Euclidean algorithm is shown as follws:

Initial condition:

$$\begin{aligned} \Omega^{(-1)}(x) &= x^{2t} \quad , \quad \Omega^{(0)}(x) = S(x) \\ \sigma^{(-1)}(x) &= 0 \quad , \quad \sigma^{(0)} = 1 \end{aligned}$$

for ($i = 1$ **to** t)

$$\begin{aligned} \Omega^{(i)}(x) &= \Omega^{(i-2)}(x)+\Omega^{(i-1)}q^{(i)}(x) \\ \sigma^{(i)}(x) &= \sigma^{(i-2)}(x)+\sigma^{(i-1)}q^{(i)}(x) \end{aligned}$$

where $q_i(x)$ is the i-th iteration quotient polynomial, $\Omega^{(i)}(x)$ and $\sigma^{(i)}(x)$ are the i-th iteration error evaluator polynomial and error locator polynomial, respectively. We define

$$\Omega_j^{(i)} = \Omega_j^{(i-2)} + \Omega_j^{(i-1)} q_0^{(i)} + \Omega_{j-1}^{(i-1)} q_1^{(i)} \quad (3)$$

$$\sigma_\lambda^{(i)} = \sigma_\lambda^{(i-2)} + \sigma_\lambda^{(i-1)} q_0^{(i)} + \sigma_{\lambda-1}^{(i-1)} q_1^{(i)} \quad (4)$$

for $0 \leq j \leq 2t-1-i$ and $0 \leq \lambda \leq i$, where $\Omega_j^{(i)}$, $\sigma_\lambda^{(i)}$ corresponds to the j-th, λ-th coefficient of $\Omega(x)$ and $\sigma(x)$ at the i-th iteration, respectively. Consequently, we decompose the Euclidean Algorithm and it only needs 2 FFMs to compute $\Omega_j^{(i)}$ or $\sigma_\lambda^{(i)}$.

III. The RS Decoder Architecture

We divide the decoding process into 4 steps. Fig. 2 illustrates a 3-stage pipelining used in our RS (204, 188) decoders, where the Chien Search and the Error Value Evaluator are both carried out at the third stage. Note that the pipelining speed is dominated by Syndrome Calculator [1].

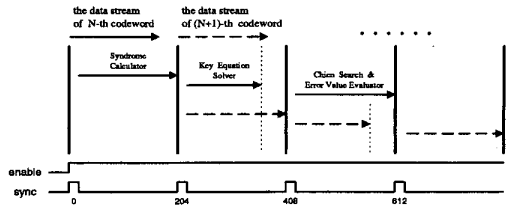


Fig. 2. Three-stage pipelining of the RS decoder.

A. Syndrome Calculator

By definition the syndrome polynomial is $S(x) = S_1 + S_2 x + \ldots + S_{2t} x^{2t-1}$, $S_i = R(\alpha^i)$, where $R(x) = R_0 + R_1 x + R_2 x^2 + \cdots + R_{N-1} x^{N-1}$ is a received polynomial and R_{N-1} is the first received symbol into a syndrome cell illustrated in Fig. 3.

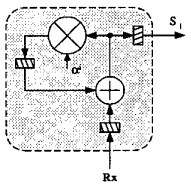


Fig. 3. The syndrome calculator cell.

As shown in Fig. 3, at each cycle, the partial syndrome is multiplied with α^i and accumulated with the received symbol. After all the received symbols are processed, the accumulated result is the i-th syndrome, S_i.

B. Key Equation Solver

It is found that our proposed decomposed Euclidean algorithm leads to a small number of FFMs, which has the most complexity among all RS decoder elements. Although more cycles are used in the decomposed algorithm, the code sizes of DVB standard are large enough that any speed penalty can be overcome. Compared to existing designs, our Key Equation Solver only needs 2 or 4 FFMs [2].

Fig. 4 illustrates the Key Equation Solver architecture, which needs 4 FFMs. The branch labeling in Fig. 4 corresponds to a particular time instance while computing

[1] After all symbols of an entire codeword are received, the set of syndromes is carried out and put into the next stage, Key Equation Solver.

[2] If we calculate $\sigma(x)$ and $\Omega(x)$ parallelly, it takes 4 FFMs. Otherwise, it only needs 2 FFMs to calculate $\sigma(x)$ and $\Omega(x)$ serially.

$\sigma(x)$ and $\Omega(x)$. This Key Equation Solver architecture can also be reconfigured to compute $\sigma(x)$ after $\Omega(x)$ is computed and the computation can be done within 204 cycles. Thus 2 FFMs required to compute $\sigma(x)$ can be saved but more storage registers and complex controlled signal are demanded.

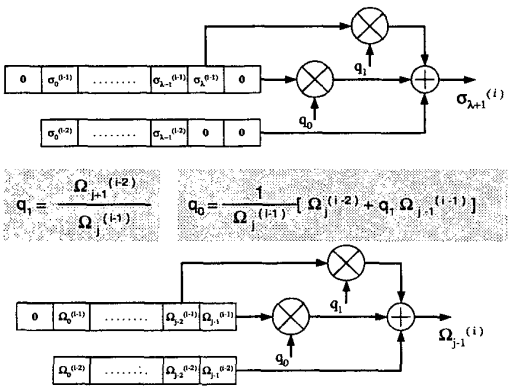
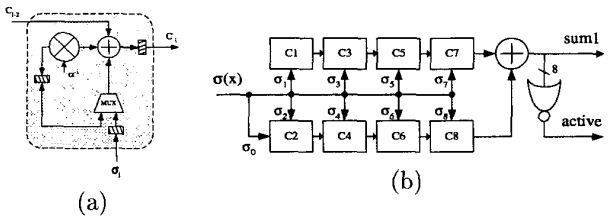


Fig. 4. Key Equation Solver architecture.

C. Chien Search

In RS decoding algorithm, Chien search is used to check whether the error locator polynomial $\sigma(x)$ equals zero or not while $x = \alpha^{-n}$, $n = 0, 1, \ldots, N-1$. If $\sigma(\alpha^{-n}) = 0$, it means there is an error at R_n, where the received polynomial is defined as $R(x) = R_0 + R_1 x + R_2 x^2 + \cdots + R_{N-1} x^{N-1}$ and R_{N-1} is the first received symbol.

Fig. 5(a) shows the circuit of the i-th Chien search cell. The right hand side of the Chien search cell accumulates the result of this and the previous cell, and sends the sum to the next cell. Fig. 5(b) shows the structure of the Chien search module with eight Chien searce cells.



(a) (b)

Fig. 5. (a)The Chien search cell: C_i. (b)The Chien search structure for $t = 8$.

D. Error Value Evaluator

In Forney algorithm, the error value becomes:

$$e_l = \frac{\Omega(\beta_l{}^{-1})}{\sigma'(\beta_l{}^{-1})} = \frac{\Omega(\beta_l{}^{-1}) \cdot \beta_l^{-1}}{\sigma_{odd}(\beta_l{}^{-1})} \qquad (5)$$

where $\sigma_{odd}(x) = \sigma_1 x + \sigma_3 x^3 + \ldots + \sigma_{odd} x^{t_{odd}}$, and β_l^{-1} indicates the root of $\sigma(x)$, for $l = 1, \ldots, t$. In Fig. 6, we calculate error values in parallel with the computation of Chien search

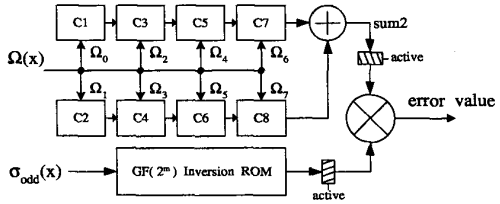


Fig. 6. Error value evaluator structure.

Note that cells C1 $\sim$ C8 in Fig. 6 are all the same as Chien search cells in Fig. 5(a). The only difference is the loaded coefficients are $\Omega_0 \sim \Omega_8$ instead of $\sigma_1 \sim \sigma_8$. While the computation of Chien search is into n-th iteration, the value of sum1 in Fig. 5(b) is $\sigma(\alpha^{-n})$, the value of sum2 in Fig. 6 is $\Omega_0\alpha^{-n} + \Omega_1\alpha^{-2n} + \ldots + \Omega_7\alpha^{-8n} = \Omega(\alpha^{-n}) \cdot \alpha^{-n}$. In other words, if $\sigma(\alpha^{-n}) = 0$, the *active* signal goes to high, and the output of FFM in Fig. 6 is the error value of the received codeword, R_n.

IV. Chip Implementation

Dual basis finite field arithmetic is adopted in the RS decoders for lower gate count. A dual-basis FFM takes one input in standard basis and the other input in dual basis to produce a dual-basis output. The design of a parallel-input parallel-output dual-basis FFM is shown in Fig. 7. The Reed-Muller Inner Product blocks implement $C_i = \sum_{k=0}^{7} \oplus\, a_k \cdot b_{k+i}$, $i = 0, 1, \ldots, 7$, where $\sum \oplus$ represents modulo-2 summation (XOR). The entire dual-basis FFM contains 73 XOR gates and 64 AND gates. Since the FFMs in the Syndrome Calculator and the Chien Search are all constant×variable multipliers, further simplification can be obtained. The Error Value Evaluator contains several constant×variable FFMs, one variable-variable FFM, and one finite field inversion(FFI) implemented by a look-up table.

We implement 2 different version of (204, 188) Reed-Solomon decoder. One is using 2 FFMs in the Key Equation Solver, whose gate count is about $16K$, and the other version of 4 FFMs also has $17K$ gate count. The chip was designed using the Standard cell library in a 0.35 μm 1P4M CMOS process. The chip size is $2.58mm \times 2.56mm$ with a core size of $1.52mm \times 1.50mm$, divided with $0.71mm \times 1.52mm$ of 2-FFM version and $0.81mm \times 1.52mm$ of 4-FFM version. The chip is packaged in a 68 LD PGA package. Fig. 8 shows the chip die

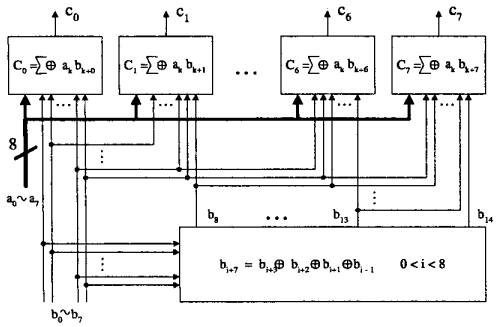


Fig. 7. Dual-basis FFM design.

photo. After fabrication, both of these two RS (204, 188) decoder chip can run at $87MHz$.

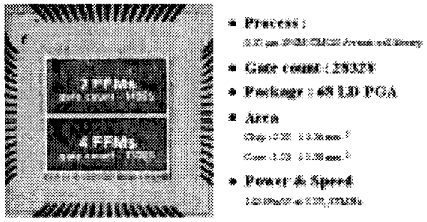


Fig. 8. The chip die photo.

V. Conclusion

In this paper we have presented the design and implementation of a RS (204, 188) decoder. Such a RS code is useful in the upcoming DVB applicaition. The novelty in our design is the decomposed Euclidean algorithm to implement the Key Equation Solver. Our technique can drastically reduce the required number of finite field multipliers (FFMs) from $2t \sim 3t$ to 2 or 4, while maintaining the decoding speed.

We have implemented our RS decoder by a 0.35μm CMOS 1P4M technology with an area of $6.6mm^2$ (core area $2.28mm^2$). The total gate count is about $16K \sim 17K$. After fabrication, both of these two RS (204, 188) decoders can run at 87MHz.

Acknowledgment

The authors would like to thank Dr. Meng-Lieh, Sheu, who support our chip testing work. The MPC support from the Chip Implementation Cneter is also acknowledged, too.

References

[1] E. Berlekamp, *Algebraic Coding Theory.* McGraw-Hill, New York, 1968.

[2] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A Method for Solving Key Equation for Decoding Goppa Codes," *Information and Control*, vol. 27, pp. 87–99, 1975.

[3] H. Chang and C. Chang, "New Serial Architecture for the Berlekamp-Massey Algorithm," *IEEE transactoin on Communications*, vol. 47, no. 4, pp. 481–483, 1999.

[4] K. Liu, "Architecture for VLSI Design of Reed-Solomon Decoders," *IEEE Transaction on Computers*, vol. c-33, no. 2, pp. 178–189, 1984.

[5] Y. Oh and D. Kim, *Method and Apparatus for Computing Error Locator Polynomial for Use in A Reed-Solomon Decoder.* U.S. Patent 4663470, 1996.

[6] H. M. Shao and I. S. Reed, "On the VLSI Design of a Pipeline Reed-Solomon Decoder Using Systolic Arrays," *IEEE Transaction on Computers*, vol. 37, no. 10, pp. 1273–1280, 1988.

[7] H. Chang and M. Sunwoo, "A low complexity Reed-Solomon architecture using the Euclid's algorithm," *Proc. of the IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 513–516, 1999.