Journal of Southeast University (English Edition) Vol. 22, No. 1, pp. 5 – 10 Mar. 2006 ISSN 1003—7985

Area optimization of parallel Chien search architecture for Reed-Solomon(255, 239) decoder

Hu Qingsheng Wang Zhigong Zhang Jun Xiao Jie

(Institute of RF- & OE-ICs, Southeast University, Nanjing 210096, China)

Abstract: A global optimization algorithm (GOA) for parallel Chien search circuit in Reed-Solomon (RS) (255, 239) decoder is presented. By finding out the common modulo 2 additions within groups of Galois field (GF) multipliers and pre-computing the common items, the GOA can reduce the number of XOR gates efficiently and thus reduce the circuit area. Different from other local optimization algorithms, the GOA is a global one. When there are more than one maximum matches at a time, the best match choice in the GOA has the least impact on the final result by only choosing the pair with the smallest relational value instead of choosing a pair randomly. The results show that the area of parallel Chien search circuits can be reduced by 51% compared to the direct implementation when the group-based GOA is used for GF multipliers and by 26% if applying the GOA to GF multipliers separately. This optimization scheme can be widely used in general parallel architecture in which many GF multipliers are involved.

Key words: RS decoder; Chien search circuit; area optimization; Galois field multiplier

Forward-error correction (FEC) codes used in long-distance optical communication systems should provide significant coding gains with a high code rate and moderate complexity. In ITU-T G. 975, the Reed-Solomon (RS) (255, 239) code has been standardized to resist burst errors for optical fiber submarine cable systems. With only about 7% overhead, this RS code can not only provide approximately a 6 dB coding gain at random errors correction, but also correct bursts of lengths up to 64 bits.

To increase the decoding throughput, various parallel decoders[1-6] are derived by developing parallel architecture for various building blocks. Among the three major building blocks of the RS decoder, i. e., syndrome generator unit, key equations solver and the Chien search block, the parallel Chien search block is the most area consuming unit[7]. It occupies more than 65% of the logic core for both 10 and 40 Gbit/s FEC devices. Therefore, how to develop a low complexity parallel Chien search circuit for high throughput RS decoders is of great interest and is considered in this paper.

1 Parallel Chien Search Architecture for RS Decoder

Consider an RS(n, k) decoder with a block length of n bits, k information bits. After syndrome calculation and solving the key equation, the error locator polynomial $\sigma(x)$ and the error value polynomial $\omega(x)$ can be found. Then determining the error locations from $\sigma(x)$ and the error values at these locations from $\omega(x)$ is the last step in decoding an RS code. The Chien search algorithm can effectively determine the error locations by finding out all the roots of the error locator polynomial $\sigma(x)$. Normally, the Chien search algorithm checks whether $\sigma(\alpha^i)=0$ for $1\leqslant i\leqslant n$. If $\sigma(\alpha^i)=0$, it means that an error is found at α^{-i}. Then, the corresponding error value e can be calculated using the Forney algorithm as

$$e=\frac{\omega(x)}{\sigma_{\text{odd}}(x)}\Bigg|_{x=\alpha^{-i}}$$

where σ_{odd} is the odd term of $\sigma(x)$.

A typical Chien search circuit is shown in Fig. 1. Since all the possible locations have to be evaluated for the $\sigma(x)$, normally, it takes n clock cycles to complete the Chien search process for an RS(n, k) decoder. For example, for the RS(255, 239) decoder, it will take 255 cycles to complete the Chien search.

To speedup this search process, the parallel Chien search architecture that evaluates several locations per clock cycle is essential. Supposing that p stands for the parallel factor, then a parallel Chien search architecture with a parallel factor p can reduce the searching cycles

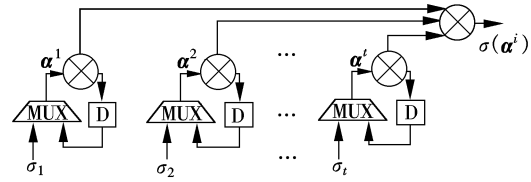

Fig. 1 Typical Chien search circuit

Received 2005-10-10.

Biography: Hu Qingsheng (1964—), female, doctor, associate professor, qshu@seu.edu.cn.

for error locations from n down to $\lfloor n/p \rfloor$. For example, a parallel Chien search architecture with $p=4$ can find four roots per cycle and thus can complete the total searching after $n/4$ cycles.

Fig. 2 is a parallel Chien search architecture for an RS(n, k) decoder, where parallel factor $p=4$. In addition to the four polynomials $\sigma(\alpha^{i+1})$, $\sigma(\alpha^{i+2})$, $\sigma(\alpha^{i+3})$ and $\sigma(\alpha^{i+4})$, four odd terms $\sigma_{\text{odd}}(\alpha^{i+1})$, $\sigma_{\text{odd}}(\alpha^{i+2})$, $\sigma_{\text{odd}}(\alpha^{i+3})$ and $\sigma_{\text{odd}}(\alpha^{i+4})$ which are needed in determining the error values with the Forney algorithm are computed simultaneously to facilitate the implementation of error correction.

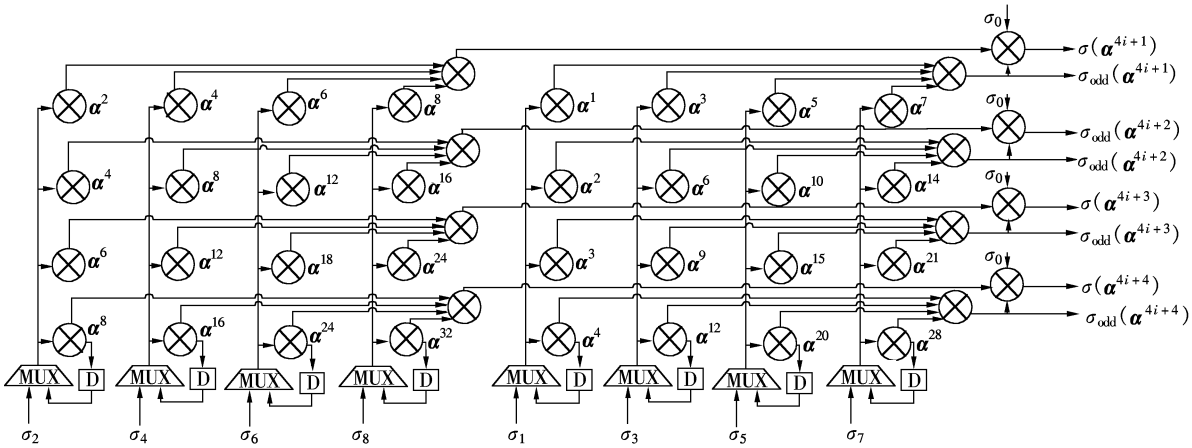

Fig. 2 Parallel Chien architecture for RS decoder

Obviously, the parallel Chien search architecture has a larger area compared to the conventional serial architecture. Supposing that the parallel factor is p, a parallel architecture for an RS(n, k) decoder has exactly $(p \times t)$ 8-bit constant Galois field (GF) multipliers, $3 \times p$ 8-bit GF adders, t 8-bit registers and t 8-bit multiplexes, where $2t = n - k$. Thus how to optimize the parallel Chien search architecture in the area reduction is important for the high-speed RS decoder. It has been found that the only operation required for constant multiplication is modulo 2 addition. Hence, we can define the computational complexity as the number of XOR gates included in the GF constant multipliers. In the next section, an optimization algorithm that can reduce the complexity of a GF constant multiplier is discussed in detail.

2 Complexity Reduction Scheme for GF Multiplier

In this section, a novel optimization algorithm for a GF constant multiplier is proposed. In Ref. [8], an algorithm has been proposed to optimize the area of an RS encoder by up to 35% compared to straightforward implementation. However, the algorithm gave only the local optimization and did not guarantee a globally optimum solution. In this paper, a global optimization algorithm (GOA) is proposed to reduce the complexity effectively. The same as the algorithm in Ref. [8], the GOA is also an iterative algorithm, in which the frequently appearing XOR-additions are found and precomputed. Different from Ref. [8], however, if there are more than one matching pairs with the maximum matching value, the pair with the smallest relational value is chosen as the best match. Below, we will give GOA in detail.

2.1 Global optimization algorithm

Consider a constant multiplication in GF(2^m), where $\boldsymbol{M}$ is the product of fixed element α^i with variable field element $\boldsymbol{B} = \{b_0, b_1, b_2, \ldots, b_7\}$. Without loss of generality, we assume $\alpha^i = \alpha^8 = (10111000)$, then we can describe $\boldsymbol{M}$ as

$$\boldsymbol{M} = \alpha^8 \boldsymbol{B} = \alpha^8 [b_7\alpha^7 + b_6\alpha^6 + \ldots + b_0] = \begin{Bmatrix} b_0 + b_4 + b_5 + b_6 \\ b_1 + b_5 + b_6 + b_7 \\ b_0 + b_2 + b_4 + b_5 + b_7 \\ b_0 + b_1 + b_3 + b_4 \\ b_0 + b_1 + b_2 + b_6 \\ b_1 + b_2 + b_3 + b_7 \\ b_2 + b_3 + b_4 \\ b_3 + b_4 + b_5 \end{Bmatrix} =$$

$$\begin{Bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{Bmatrix} = \boldsymbol{\alpha}_{\text{coeff}} \boldsymbol{b} \tag{1}$$

A straightforward realization of the above constant multiplier requires 23 XOR gates. However, there are

many redundant modulo 2 additions that allow a reduction in the number of operations. For instance, the sum $b_0 + b_4$ in Eq. (1) appears in rows 1, 3, and 4, respectively. A straightforward implementation would compute it three times. If the frequently appearing pairs can be pre-computed, the redundancies will be eliminated and the hardware area will be reduced.

Different from the local optimization algorithm in Ref. [8], the GOA is a global one that consists of the following four steps:

Step 1 For any column i and j in matrix $\boldsymbol{\alpha}_{\text{coeff}}$, find all the possible pairs with bit-wise matches greater than 1;

Step 2 Choose the pair (pairs) with the maximum bit-wise matches; if there are more than one such pairs, choose the one with the smallest relational value as the best match; if these maximum matches have the same relational value, choose a pair randomly;

Step 3 Eliminate the redundancy from the best match; append an additional column at the right of the matrix to hold the redundancy;

Step 4 Repeat steps 1 to 3 for all the columns in the matrix including the appended columns until no improvement is achieved; i. e., the best match is not greater than 1 bit.

After the 4-step optimization, the most frequently appearing XOR additions are chosen and pre-computed. This will reduce the number of XOR gates and decrease the complexity of the GF constant multiplier.

Below, we give the definition of the relational value for a maximum match. Suppose in an iteration, m pairs $b_{i1,j1}$, $b_{i2,j2}$, …, $b_{im,jm}$ are found with the same maximum bit-wise matches and the symbol $\text{col}_i \,\&\, \text{col}_j$ denotes the operation result of bit-wise matches for column i and j, then $R_{ir,jr}$, the relational value of maximum match $b_{ir,jr}$ is defined as

$$\begin{aligned} R_{ir,jr} = &(\text{col}_{ir} \,\&\, \text{col}_{jr} \,\&\, \text{col}_{i1} \,\&\, \text{col}_{j1}) + \ldots + \\ &(\text{col}_{ir} \,\&\, \text{col}_{jr} \,\&\, \text{col}_{iq} \,\&\, \text{col}_{jq}) + \ldots + \\ &(\text{col}_{ir} \,\&\, \text{col}_{jr} \,\&\, \text{col}_{im} \,\&\, \text{col}_{jm}) \end{aligned}$$

where $iq \neq ir, jq \neq jr$.

This means that if a matching pair has the minimum relational value and is chosen as the best match in the current iteration, it will have less impact on the next iteration optimization.

Below, we will take Eq. (1) as an example to explain the GOA. When applying the GOA in Eq. (1), in the first iteration, three matching pairs $b_{0,4} = b_0 + b_4$, $b_{3,4} = b_3 + b_4$, and $b_{4,5} = b_4 + b_5$ are found with the maximum matching bits of three, and their relational values are $R_{0,4} = 2$, $R_{3,4} = 2$, and $R_{4,5} = 2$. Since they have the same relational value, $b_{0,4}$ is selected randomly as the best match. Similarly, in the second iteration, 10 pairs $b_{1,2}$, $b_{1,3}$, $b_{1,6}$, $b_{1,7}$, $b_{2,3}$, $b_{2,7}$, $b_{3,4}$, $b_{5,6}$, $b_{5,7}$ and $b_{5,04}$ are found with maximum matching bits of two. Among them, $b_{3,4}$ has the smallest relational value of three, so it is chosen as the best match. In the next iteration, i. e. the third iteration, there are totally eight pairs $b_{1,2}$, $b_{1,3}$, $b_{1,6}$, $b_{1,7}$, $b_{2,7}$, $b_{5,6}$, $b_{5,7}$, and $b_{5,04}$ with maximum matching bits of two. Since $R_{1,3} = 3$ is the smallest relational value, $b_{1,3}$ is chosen as the result of this iteration. In the same way, $b_{1,6}$ and $b_{2,7}$ are chosen as the best matches in the next two iterations, respectively. In the last iteration, only one pair $b_{5,04}$ is found and the algorithm ends.

After six iteration steps, six common XOR additions are found: $b_0 + b_4$, $b_3 + b_4$, $b_1 + b_3$, $b_1 + b_6$, $b_2 + b_7$ and $b_5 + b_{04}$. By pre-computing them, the production matrix $\boldsymbol{M}$ can be written as

$$\boldsymbol{M} = \left\{ \begin{array}{c} ((b_0 + b_4) + b_5) + b_6 \\ (b_1 + b_6) + b_5 + b_7 \\ (b_2 + b_7) + (b_5 + (b_0 + b_4)) \\ (b_0 + b_4) + (b_1 + b_3) \\ (b_1 + b_6) + b_0 + b_2 \\ (b_1 + b_3) + (b_2 + b_7) \\ b_2 + (b_3 + b_4) \\ (b_3 + b_4) + b_5 \end{array} \right\} \tag{2}$$

It is easy to see that the number of XOR gates is reduced to 16 compared to the straightforward implementation that requires 23 XOR gates.

The advantage of the GOA is that when there are multiple maximum matches at a time, the best matching choice in each iteration has the least impact on the final result since the GOA only chooses the pair with the smallest relational value instead of choosing a pair randomly. Therefore, the GOA is a global optimization algorithm while the algorithm in Ref. [8] is a local optimization solution.

2.2 Group-based optimization

In fact, the GOA can be not only used in individual GF multipliers, but also used among g multipliers, where g (>1) is the group size and g multipliers share the same multiplicand. In this case, by searching frequently occurring XOR-additions among g coefficient matrices and pre-computing them, the number of XOR gates is reduced efficiently. Obviously, in most cases the results of optimizing multiple multipliers are better than those of optimizing individual multipliers separately. We call the optimization of multiple multipliers group-based optimization.

As shown in Fig. 2, the Chien search circuit is divided into eight groups of multipliers, each corresponding to a variable element σ_i ($1 \leqslant i \leqslant 8$). One of the groups with the multiplicands σ_8 is shown in Fig. 3, in

which four constant multipliers with constant multiplicands $\boldsymbol{\alpha}^8$, $\boldsymbol{\alpha}^{16}$, $\boldsymbol{\alpha}^{24}$ and $\boldsymbol{\alpha}^{32}$ are included. Since the four multipliers have the same multiplicand σ_8, we can apply the GOA to the multiplier group.

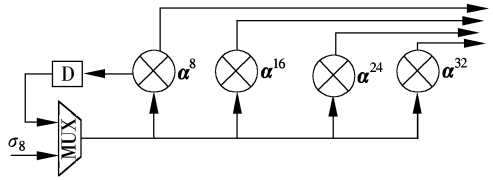

Fig.3 Multiplier group of $\boldsymbol{\alpha}^8$ in parallel Chien search architecture

As given in section 2, let $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3$ and $\boldsymbol{M}_4$ denote the products of the variable field element $\boldsymbol{B}$ with the constant elements $\boldsymbol{\alpha}^8, \boldsymbol{\alpha}^{16}, \boldsymbol{\alpha}^{24}, \boldsymbol{\alpha}^{32}$, respectively.

$$\boldsymbol{M}_1 = \boldsymbol{\alpha}^8 \boldsymbol{B} = \begin{Bmatrix} b_0 + b_4 + b_5 + b_6 \\ b_1 + b_5 + b_6 + b_7 \\ b_0 + b_2 + b_4 + b_5 + b_7 \\ b_0 + b_1 + b_3 + b_4 \\ b_0 + b_1 + b_2 + b_6 \\ b_1 + b_2 + b_3 + b_7 \\ b_2 + b_3 + b_4 \\ b_3 + b_4 + b_5 \end{Bmatrix}$$

$$\boldsymbol{M}_2 = \boldsymbol{\alpha}^{16} \boldsymbol{B} = \begin{Bmatrix} b_2 + b_5 + b_7 \\ b_3 + b_6 \\ b_0 + b_2 + b_4 + b_5 \\ b_0 + b_1 + b_2 + b_3 + b_6 + b_7 \\ b_1 + b_3 + b_4 + b_5 \\ b_2 + b_4 + b_5 + b_6 \\ b_0 + b_3 + b_5 + b_6 + b_7 \\ b_1 + b_4 + b_6 + b_7 \end{Bmatrix}$$

$$\boldsymbol{M}_3 = \boldsymbol{\alpha}^{24} \boldsymbol{B} = \begin{Bmatrix} b_0 + b_1 \\ b_0 + b_1 + b_2 \\ b_0 + b_2 + b_3 \\ b_0 + b_3 + b_4 \\ b_4 + b_5 \\ b_5 + b_6 \\ b_6 + b_7 \\ b_0 + b_7 \end{Bmatrix}$$

$$\boldsymbol{M}_4 = \boldsymbol{\alpha}^{32} \boldsymbol{B} = \begin{Bmatrix} b_0 + b_1 + b_4 + b_7 \\ b_1 + b_2 + b_5 \\ b_0 + b_1 + b_2 + b_3 + b_4 + b_6 + b_7 \\ b_0 + b_2 + b_3 + b_5 \\ b_0 + b_3 + b_6 + b_7 \\ b_1 + b_4 + b_7 \\ b_2 + b_5 \\ b_0 + b_3 + b_6 \end{Bmatrix}$$

It is easy to see that the original numbers of XOR gates of the four multipliers are 23, 24, 11 and 22, respectively.

First, the GOA is applied to the four multipliers separately. After optimization, four groups of common XOR additions (denoted as c_i) are obtained:

$$\begin{aligned}
\boldsymbol{M}_1 \quad & c_1 = b_0 + b_4,\ c_2 = b_3 + b_4,\ c_3 = b_1 + b_3, \\
& c_4 = b_1 + b_6,\ c_5 = b_2 + b_7,\ c_6 = b_5 + c_1 \\
\boldsymbol{M}_2 \quad & c_1 = b_2 + b_5,\ c_2 = b_3 + b_6,\ c_3 = b_0 + c_2, \\
& c_4 = b_4 + c_1,\ c_5 = b_1 + b_4,\ c_6 = b_7 + c_3 \\
\boldsymbol{M}_3 \quad & c_1 = b_0 + b_1,\ c_2 = b_0 + b_3 \\
\boldsymbol{M}_4 \quad & c_1 = b_0 + b_3,\ c_2 = b_2 + b_5,\ c_3 = b_1 + b_4, \\
& c_4 = b_1 + b_4 + b_7 = c_3 + b_7, \\
& c_5 = b_0 + b_3 + b_6 = c_1 + b_6
\end{aligned}$$

Consequently, $\boldsymbol{M}_1$ to $\boldsymbol{M}_4$ can be written as

$$\boldsymbol{M}_1 = \begin{Bmatrix} c_6 + b_6 \\ c_4 + b_5 + b_7 \\ c_5 + c_6 \\ c_1 + c_3 \\ c_4 + b_0 + b_2 \\ c_3 + c_5 \\ b_2 + c_2 \\ c_2 + b_5 \end{Bmatrix}, \quad \boldsymbol{M}_2 = \begin{Bmatrix} c_1 + b_7 \\ c_2 \\ b_0 + c_4 \\ b_1 + b_2 + c_6 \\ c_5 + b_3 + b_5 \\ c_4 + b_6 \\ b_5 + c_6 \\ c_5 + b_6 + b_7 \end{Bmatrix}$$

$$\boldsymbol{M}_3 = \begin{Bmatrix} c_1 \\ c_1 + b_2 \\ b_2 + c_2 \\ c_2 + b_4 \\ b_4 + b_5 \\ b_5 + b_6 \\ b_6 + b_7 \\ b_0 + b_7 \end{Bmatrix}, \quad \boldsymbol{M}_4 = \begin{Bmatrix} b_0 + c_4 \\ b_1 + c_2 \\ b_2 + c_4 + c_5 \\ c_1 + c_2 \\ c_5 + b_7 \\ c_4 \\ c_2 \\ c_5 \end{Bmatrix}$$

As a result, the required numbers of XOR gates for the four multipliers are reduced from 23 to 16, 24 to 16, 11 to 9 and 22 to 11, respectively.

Now, if group-based optimization is used to the multiplier group of σ_8, the GOA will search all the matches within the four matrices $\boldsymbol{M}_1$ to $\boldsymbol{M}_4$ and 16 common items are found:

$$\begin{gathered}
c_1 = b_0 + b_3,\ \ c_2 = b_1 + b_7,\ \ c_3 = b_2 + b_5 \\
c_4 = b_6 + c_1 = b_0 + b_3 + b_6,\ c_5 = b_4 + b_5 \\
c_6 = b_4 + c_2 = b_1 + b_4 + b_7,\ \ c_7 = b_0 + b_1 \\
c_8 = b_4 + c_3 = b_2 + b_4 + b_5,\ \ c_9 = b_2 + c_7 = b_0 + b_1 + b_2 \\
c_{10} = b_3 + c_5 = b_3 + b_4 + b_5,\ \ c_{11} = b_4 + c_1 = b_0 + b_3 + b_4 \\
c_{12} = b_5 + b_6,\ \ c_{13} = b_7 + c_4 = b_0 + b_3 + b_6 + b_7 \\
c_{14} = b_0 + b_7,\ c_{15} = b_2 + b_3 \\
c_{16} = b_2 + c_4 = b_0 + b_2 + b_3 + b_6
\end{gathered}$$

Thus, $\boldsymbol{M}_1$ to $\boldsymbol{M}_4$ can be expressed as

$$\boldsymbol{M}_1 = \begin{Bmatrix} b_0 + b_6 + c_5 \\ c_2 + c_{12} \\ c_8 + c_{14} \\ b_1 + c_{11} \\ b_6 + c_9 \\ c_2 + c_{15} \\ b_4 + c_{15} \\ c_{10} \end{Bmatrix}, \quad \boldsymbol{M}_2 = \begin{Bmatrix} b_7 + c_3 \\ b_3 + b_6 \\ b_0 + c_8 \\ c_2 + c_{16} \\ b_1 + c_{10} \\ b_6 + c_8 \\ b_5 + c_{13} \\ b_6 + c_6 \end{Bmatrix}$$

$$\boldsymbol{M}_3 = \begin{Bmatrix} c_7 \\ c_9 \\ b_2 + c_1 \\ c_{11} \\ c_5 \\ c_{12} \\ b_6 + b_7 \\ c_{14} \end{Bmatrix}, \quad \boldsymbol{M}_4 = \begin{Bmatrix} b_0 + c_6 \\ b_1 + c_3 \\ c_6 + c_{16} \\ c_1 + c_3 \\ c_{13} \\ c_6 \\ c_3 \\ c_4 \end{Bmatrix}$$

Obviously, by sharing these common items within the multiplier group, the number of XOR gates is decreased significantly. It is not difficult to see that the total number of XOR gates in the multiplier group of σ_8 is now 38 compared to 52 in individual optimization, both of which can save significant hardware area compared to the straightforward implementation that requires 80 XOR gates.

3 Optimization Results

By applying the group-based GOA to the parallel Chien search circuit in RS(255, 239) decoder, the optimized results in terms of XOR gates are listed in Tab. 1.

Tab. 1 Chien search complexity for RS(255, 239) decoder with parallel factor of four

Groups	Straight-forward implementation	Individual optimization		Group-based optimization	
		No. of gates	Improvement/%	No. of gates	Improvement/%
σ_1	30	26	13	14	53
σ_2	57	47	17	25	56
σ_3	72	54	25	30	58
σ_4	80	59	26	38	52
σ_5	81	57	30	41	49
σ_6	75	56	25	40	47
σ_7	78	57	27	43	45
σ_8	80	52	35	38	52
Total circuit	553	408	26	269	51

In Tab. 1, the number of XOR gates for eight multiplier groups in the parallel Chien search circuit is given separately. We can see that after group-based optimization, the complexity of the Chien search circuit is reduced greatly compared to the straightforward implementation. The total decrease of the Chien search circuit can be up to 51% while the decrease is different for each group. The maximum improvement of 58% is obtained in the group of σ_3. Generally the more complex the original circuit is, the more improvement can be obtained.

On the other hand, as described above, the group-based optimization can achieve better results compared to the individual optimization. In the former case, the improvement is about 51% while the reduction is only 26% in the latter.

4 Conclusion

In this paper, to reduce the hardware size of a high speed RS decoder, a novel global optimization algorithm is proposed. By applying a GOA to GF multiplier groups in parallel Chien search circuits, the result shows that the number of XOR gates of the circuit is reduced by 51% compared to the original design. The result is also better than that of individual multiplier optimization. It is worthy to point out that the decrease of area consumption also helps to shorten the critical path of the circuit. It is concluded that group-based optimization techniques can be not only used in Chien search architecture but also used in other parallel architectures in which GF multipliers are used.

References

[1] Lee Hanho. High-speed VLSI architecture for parallel Reed-Solomon decoder [J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2003, **11**(4): 288 - 294.

[2] Lee Hanho. An area-efficient Euclidean algorithm block for Reed-Solomon decoder [C]//*Proc of IEEE Computer Society Annual Symposium on VLSI*. Tampa, FL, 2003: 209 - 210.

[3] Chang Hsie-Chia, Lin Chien-Ching, Lee Chen-Yi. A low-power Reed-Solomon decoder for STM-16 optical communications [C]//*Proc of IEEE Asia-Pacific Conference on ASIC*. Taipei, 2002: 351 - 354.

[4] Song L, Parhi K K. Low complexity modified Mastrovito multipliers over finite fields GF (2^m) [C]//*Proc of IEEE ISCAS*. Orlando, FL, 1999: 508 - 512.

[5] Zhang T, Parhi K K. Systematic design of original and modified Mastrovito multipliers for general irreducible polynomials [J]. *IEEE Trans on Computers*, 2001, **50**(7): 734 - 749.

[6] Zhu Haikun, Shen Bo, Zhang Qianling. Design of a high performance Reed-Solomon decoder with switch box control logic [C]//*Proc of the 4th International Conference on ASIC*. Shanghai, China, 2001: 452 - 455.

[7] Chen Yanni, Parhi K K. Small area parallel Chien search

architectures for long BCH codes [J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2004, **12**(5): 545 - 549.

[8] Paar C. Optimized arithmetic for Reed-Solomon encoders [C]//*Proc of IEEE International Symposium on Information Theory*. Ulm, Germany, 1997: 250 - 255.

[9] Hu Qingsheng, Wang Zhigong, Zhang Jun, et al. Low complexity parallel Chien search architecture for RS decoder [C]//*Proc of IEEE ISCAS*. Kobe, Japan, 2005: 340 - 343.

RS(255,239) 解码器并行钱氏搜索电路的面积优化

胡庆生　王志功　张　军　肖　洁

(东南大学射频与光电集成电路研究所,南京 210096)

摘要:提出了一个全局优化算法(GOA)对 RS(255,239)解码器中的并行钱氏搜索电路进行面积优化. 通过查找钱氏搜索电路中 GF (Galois field)常数乘法器的公共模 2 加运算并进行预运算,GOA 能够有效地减少电路中异或门的数量,从而减少电路面积. 与原有局部优化方法不同,GOA 是一个全局优化算法. 当每次迭代中同时有多个最大匹配对时,GOA 通过选取与其他匹配对关系最小的一对作为最优匹配而不是随机地选择一对,使得当前结果对最终的优化结果影响最小. 进一步将基于组的 GOA 用于 GF 乘法器组的优化,结果显示相对于直接实现方法,可使并行钱氏搜索电路的面积减少 51%,而对 GF 乘法器的单独优化也能使电路面积减少 26%. 该优化方法可广泛地用于含有大量模 2 加运算的并行结构中.

关键词:RS 解码器;钱氏搜索电路;面积优化;GF 乘法器

中图分类号:TN722