The Design of IP Cores in Finite Field for Error Correction

Ming-Haw Jing, Jian-Hong Chen, Yan-Haw Chen,
Zih-Heng Chen and Yaotsu Chang
I-Shou University
Taiwan, R.O.C.

1. Introduction

In recent studies, the bandwidth of communication channel, the reliability of information transferring, and the performance of data storing devices become the major design factors in digital transmission /storage systems. In consideration of those factors, there are many algorithms to detect or remove the noisefrom the communication channel and storage media, such as cyclic redundancy check (CRC) and errorcorrecting code (Peterson & Weldon, 1972; Wicker, 1995). The former, a hush function proposed by Peterson and Brown (Peterson & Brown, 1961), is utilized applied in the hard disk and network for error detection; the later is a type of channel coding algorithms recover the original data from the corrupted data against various failures. Normally, the scheme adds redundant code(s) to the original data to provide reliability functions such as error detection or error correction. The background of this chapter involves the mathematics of algebra, coding theory, and so on.
In terms of the design of reliable components by hardware and / or software implementations, a large proportion of finite filed operations is used in most related applications. Moreover, the frequently used finite field operations are usually simplified and reconstructed into the hardware modules for high-speed and efficient features to replace the slow software modules or huge look-up tables (a fast software computation). Therefore, we will introduce those common operations and some techniques for circuit simplification in this chapter. Those finite field operations are additions, multiplications, inversions, and constant multiplications, and the techniques include circuit simplification, resource-sharing methods, etc. Furthermore, the designers may use mathematical techniques such as group isomorphism and basis transformation to yield the minimum hardware complexities of those operations. And, it takes a great deal of time and effort to search the optimal designs. To solve this problem, we propose the computer-aided functions which can be used to analyze the hardware speed/complexity and then provide the optimal parameters for the IP design.
This chapter is organized as follows: In Section 2, the mathematical background of finite field operations is presented. The VLSI implementation of those operations is described in Section 3. Section 4 provides some techniques for simplification of VLSI design. The use of

computer-aided functions in choosing the suitable parameters is introduced in Section 5. Finally, the result and conclusion are given.

2. The mathematic background of finite field

Elements of a finite field are often expressed as a polynomial form over $GF(q)$, the characteristic of the field. In most computer related applications, the Galois field with characteristic 2 is wildly used because its ground field, $GF(2)$, can be mapped into bit-0 and bit-1 for digital computing. For convenience, the value within two parenthesises indicates that the coefficients for a polynomial in descending order. For example, the polynomial, $x^6 + x^5 + x^3 + 1$, is represented by {1101001} in binary form or {69} in hexadecimal form. So does an element $\alpha \in GF(2^m)$ is presented as symbol based polynomial.

2.1 The common base representations

2.1.1 The standard basis

If an element $\alpha \in GF(2^m)$ is the root of a degree m irreducible polynomial $f(x)$, i.e., $f(\alpha) = 0$, then the set $\{1, \alpha^1, \alpha^2, \ldots, \alpha^{m-1}\}$ forms a basis, is called a standard basis, a polynomial basis or a canonical basis (Lidl & Niederreiter, 1986). For example, construct $E = GF(2^4)$ with the degree 4 irreducible polynomial $f(x) = x^4 + x + 1$, suppose $f(\alpha) = 0$, that is, $\alpha^4 = \alpha + 1$ and $E = <\alpha> \cup \{0\}$ as Table 1.

element	α^3	α^2	α^1	α^0	element	α^3	α^2	α^1	α^0
0	0	0	0	0	α^7	0	1	1	1
α^0	0	0	0	1	α^8	1	1	1	0
α^1	0	0	1	0	α^9	0	1	0	1
α^2	0	1	0	0	α^{10}	1	0	1	0
α^3	1	0	0	0	α^{11}	1	1	0	1
α^4	1	0	0	1	α^{12}	0	0	1	1
α^5	1	0	1	1	α^{13}	0	1	1	0
α^6	1	1	1	1	α^{14}	1	1	0	0

Table 1. The standard basis expression for all elements of $E = GF(2^4)$

2.1.2 The normal basis

For a given $GF(2^m)$, there exists a normal basis $\{\alpha, \alpha^2, \alpha^{2^2}, \ldots, \alpha^{2^{m-1}}\}$. Let $\beta = \sum_{i=0}^{m-1} b_i \alpha^{2^i}$ be represented in a normal basis, and the binary vector $(b_0, b_1, \ldots b_{m-1})$ is used to represent the coefficients of β, denoted by $\beta = (b_0, b_1, \ldots b_{m-1})$. Since $\alpha^{2^m} = 1 = \alpha^{2^0}$ by Fermat's little theorem (Wang et al., 1985), $\beta^2 = b_{m-1}\alpha^{2^0} + b_0\alpha^{2^1} + \cdots + b_{m-2}\alpha^{2^{m-1}} = (b_{m-1}, b_0, \ldots b_{m-2})$ or $\beta^{2^i} = (b_{m-i}, b_{m-i+i}, \ldots, b_{m-1}, b_0, b_1, \ldots, b_{m-i-1})$. That is, the squaring operations (2^ith power operations) can be constructed by cyclic rotations in software or by changing lines in hardware, which is with low complexity for practical applications (Fenn et al., 1996).

2.1.3 Composite field

For circuit design, using a composite field to execute some specific operations is an effective method, for example, the circuit of finite field inversion obtained in composite filed has the minimum complexity. The famous example is found in most hardware designs of AES VLSI (Hsiao et al., 2006; Jing et al., 2007), in which the S-box is a non-linear substitution for all elements in $GF(2^8)$ can be designed with a less area complexity by several isomorphism composite fields such as $GF((2^2)^4)$, $GF((2^4)^2)$, and $GF(((2^2)^2))$ (Morioka & Satoh, 2003). In this section, we introduce the process to construct a composite field and the basis transformation between a standard basis and a basis in composite field.

Let $GF(2^8)$ be represented in a standard basis with relation polynomial $f(x)=x^8+x^4+x^3+x^2+1$ ($f(x)$ is primitive) and $f(\alpha)=0$ such that $\langle\alpha\rangle=GF(2^8)$ and $\gamma=\alpha^r$ is a primitive element in the ground field $GF(2^4)$, where $r=(2^8-1)/(2^4-1)=17$. We construct the composite field $GF((2^4)^2)$ over the field $GF(2^4)$ using the irreducible polynomial $q(x)$ with degree 2 over $GF(2^4)$, which is given as follows

$$q(x)=(x+\alpha)(x+\alpha^{2^4})=x^2+(\alpha+\alpha^{2^4})x+\alpha^{17}=x^2+\alpha^{34}x+\alpha^{17}. \tag{1}$$

Such that $\gamma=\alpha^{17}$ is an element of $GF(2^2)$. In order to represent the elements of the ground field $GF(2^2)$, we use the term in $q(x)$ as the basis element, which is $\gamma=\alpha^{17}$. An element A is expressed in $GF((2^4)^2)$ as

$$A=a'_0+a'_1\alpha . \tag{2}$$

where $a'_j\in GF(2^4)$. We can express a'_j in $GF(2^4)$ using $\gamma=\alpha^{17}$ as the basis element

$$a'_j=\bar{a}_{j0}+\bar{a}_{j0}\gamma+\bar{a}_{j0}\gamma^2+\bar{a}_{j0}\gamma^3=\bar{a}_{j0}+\bar{a}_{j1}\alpha^{17}+\bar{a}_{j1}\alpha^{34}+\bar{a}_{j1}\alpha^{51} . \tag{3}$$

where $\bar{a}_{ji}\in GF(2)$ for $j=0,1$ and $i=0,1,2,3$. Therefore, the representation of A in the composite field is obtained as

$$A=a'_0+a'_1\alpha=(\bar{a}_{00}+\bar{a}_{01}\alpha^{17}+\bar{a}_{02}\alpha^{34}+\bar{a}_{03}\alpha^{51})+(\bar{a}_{10}\alpha+\bar{a}_{11}\alpha^{18}+\bar{a}_{12}\alpha^{35}+\bar{a}_{13}\alpha^{52}) . \tag{4}$$

Next, substitute the terms α^{17i+j} for $j=0,1$ and $i=0,1,2,3$ by the relation polynomial $f(x)=x^8+x^4+x^3+x^2+1$ as follows:

$$\begin{gathered}\alpha^{17}=\alpha^7+\alpha^4+\alpha^3,\ \ \alpha^{34}=\alpha^6+\alpha^3+\alpha^2+\alpha^1,\ \ \alpha^{51}=\alpha^3+\alpha^1,\\ \alpha^{18}=\alpha^5+\alpha^3+\alpha^2+1,\ \ \alpha^{35}=\alpha^7+\alpha^4+\alpha^3+\alpha^2,\ \ \alpha^{52}=\alpha^4+\alpha^2 .\end{gathered} \tag{5}$$

By substituting the above terms in expression Equation (4), we obtain the representation of

A in the standard basis $(1,\alpha,\alpha^1,\ldots\alpha^7)$ as

$$A=\alpha_0+a_1\alpha_1+a_2\alpha^2+a_3\alpha^3+a_4\alpha^4+a_5\alpha^5+a_6\alpha^6+a_7\alpha^7\ . \tag{6}$$

The relationship between the terms a_h for $h=0,1,\ldots,7$ and $\overline{a}_{ji}$ for $j=0,1$ and $i=0,1,2,3$ determines a 8 by 8 conversion matrix T (Sunar et al., 2003). The first row of the matrix T is obtained by gathering the constant terms in the right hand side of Equation (4) after the substitution, which gives the constant coefficients in the left hand side, i.e., the term a_0 . A simple inspection shows that $\alpha_0=\overline{a}_{00}+\overline{a}_{11}$. Therefore, we obtain the 8×8 matrix T and this matrix gives the representation of an element in the binary field $GF(2^8)$ given its representation in the composite field $GF((2^4)^2)$ as follows:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix}=\begin{bmatrix} 1&0&0&0&0&1&0&0 \\ 0&0&1&1&1&0&0&0 \\ 0&0&1&0&0&1&1&1 \\ 0&1&1&1&0&1&1&0 \\ 0&1&0&0&0&0&1&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&1&0 \end{bmatrix}\begin{bmatrix} \overline{a}_{00} \\ \overline{a}_{01} \\ \overline{a}_{02} \\ \overline{a}_{03} \\ \overline{a}_{10} \\ \overline{a}_{11} \\ \overline{a}_{12} \\ \overline{a}_{13} \end{bmatrix}. \tag{7}$$

The inverse transformation, i.e., the conversion from $GF(2^8)$ to $GF((2^4)^2)$, requires computing the T^{-1} matrix. We can use Gauss-Jordan Elimination to derive the T^{-1} matrix as follows:

$$\begin{bmatrix} \overline{a}_{00} \\ \overline{a}_{01} \\ \overline{a}_{02} \\ \overline{a}_{03} \\ \overline{a}_{10} \\ \overline{a}_{11} \\ \overline{a}_{12} \\ \overline{a}_{13} \end{bmatrix}=\begin{bmatrix} 1&0&0&0&0&1&0&0 \\ 0&0&1&0&1&1&1&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&1&0&1&1&1 \\ 0&1&0&1&0&1&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&1&0&1&1&1&1 \\ 0&0&0&0&1&0&0&1 \end{bmatrix}\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix}. \tag{8}$$

2.1.4 The basis transformation between standard basis and normal basis

The normal basis is with some good features in hardware, but the standard basis is used in popular designs. Finding the transformation between them is an important topic (Lu, 1997), we use $GF(2^4)$ as an example to illustrate that. Suppose $GF(2^4)$ is with the relation $p(x)=x^4+x^3+1$ which is a primitive polynomial. Let $p(\alpha)=0$ such that $B_1=(\alpha^0,\alpha^1,\alpha^2,\alpha^3)$ form a standard basis. Let $\gamma=\alpha^3$ and the set $\{\gamma^1,\gamma^2,\gamma^4,\gamma^8\}$ is linear independent such that $B_2 = (\gamma^1, \gamma^2, \gamma^4, \gamma^8)$ forms a normal basis. There exists a matrix T such that $B_2^T = T \times B_1^T$ and $B_1^T = T^{-1} \times B_2^T$. The matrixes T and T^{-1} are listed as follows.

$$T = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad T^{-1} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$
$$\begin{bmatrix} \gamma^8 \\ \gamma^4 \\ \gamma^2 \\ \gamma^1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha^3 \\ \alpha^2 \\ \alpha^1 \\ \alpha^0 \end{bmatrix}, \quad \begin{bmatrix} \alpha^3 \\ \alpha^2 \\ \alpha^1 \\ \alpha^0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \gamma^8 \\ \gamma^4 \\ \gamma^2 \\ \gamma^1 \end{bmatrix}. \tag{9}$$

2.2 The basic operation in finite field

2.2.1 Addition and subtraction

For a finite field with characteristic 2, addition and subtraction are performed by the bitwise XOR operator. For example, let $a(x) = x^4 + x^2 + x^1 + 1$, $b(x) = x^4 + x^3 + x^1 + 1$, and $c(x)$ be the summation of two polynomials, thus, $c(x) = a(x) + b(x) = 2x^4 + x^3 + x^2 + 2x^1 + 2 = x^3 + x^2$ or perform in binary form {10111} + {11011} = {01100} .

2.2.2 Multiplication and inversion

The multiplication in a finite field is performed by multiply two polynomials modulo a specific irreducible polynomial. For example, consider the finite field $E = GF(2^4)$ which is with the relation $p(x) = x^4 + x + 1$ and let $p(\alpha) = 0$ thus $(\alpha^0, \alpha^1, \alpha^2, \alpha^3)$ forms a standard basis. Suppose $a, b, c \in E$ and $a = \alpha^3 + 1$, $b = \alpha^2 + \alpha + 1$, and c is the product of them. Thus $c = a \times b = (\alpha^3 + 1) \times (\alpha^2 + \alpha + 1) = \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$, refer to Table 1, we have the product result as $c = (\alpha^2 + \alpha) + (\alpha + 1) + \alpha^3 + \alpha^2 + \alpha + 1 = \alpha^3 + \alpha$. For every nonzero element $\gamma \in E = GF(2^m)$, one has $\gamma^{2^m} = \gamma$ or $\gamma^{-1} = \gamma^{2^m - 2}$ equivalently (Dinh et al., 2001). Therefore, the division for finite field can be performed by the multiplicative inversion. For example, consider the inversion in $GF(2^8)$, $\gamma^{-1} = \gamma^{2^8 - 2}$, and one can obtain this as Fig. 1.

2.2.3 Square operation

Consider an element $A = a_0 + a_1 x^1 + \cdots + a_{m-1} x^{m-1} \in E$ where $a_i \in GF(2)$ for $0 \le i < m$, the square operation for the characteristic 2 finite field is: $A^2 = (a_0 + a_1 x^1 + \cdots + a_{m-1} x^{m-1})^2$. For $a_i \in GF(2)$, we have $a_i^{\ 2} = a_i$ and thus $A^2 = a_0 + a_1 x^2 + \cdots + a_{m-1} x^{2(m-1)}$. Besides, those items with power not less m can be expressed by standard basis. Thus, we can perform the square operation by some finite field additions, i.e., XOR gates. For instance, let $E = GF(2^4)$ constructed by $f(x) = x^4 + x + 1$, an element $A = a_0 + a_1 x^1 + a_2 x^2 + a_3 x^3 \in E$, $A^2 = a_0 + a_1 x^2 + a_2 x^4 + a_3 x^6$. Two terms x^4 and x^6 can be substituted by $x + 1$ and $x^3 + x$ according to Table 1. We have $A^2 = a_0 x^0 + a_1 x^2 + a_2 (x + 1) + a_3 (x^3 + x)$ or $A^2 = (a_0 + a_2) + (a_2 + a_3) x + a_1 x^2 + a_3 x^3$. The same property is also suitable for the power 2^i operation, such as $A^{2^2}, A^{2^3}, \ldots, A^{2^{m-1}}$.

3. The hardware designs for finite field operations

3.1 Multiplier

Finite field multiplier is the basic component for most applications. Many designers choose the one with standard basis for their applications, because the standard basis is easier to show the value by the bit-vector in digital computing. As follows, we introduce two most used types of finite field multipliers, one is the conventional multiplier and another is the bit-serial one.

3.1.1 Conventional multiplier

As the statement in Section 2.2.2, let $A, B, C \in GF(2^m)$ are represented with standard basis and $C = A \times B$, where $A = \sum_{i=0}^{m-1} a_i \alpha^i$, $B = \sum_{i=0}^{m-1} b_i \alpha^i$, and the product $P = \left(\sum_{i=0}^{m-1} a_i \alpha^i\right) \times \left(\sum_{i=0}^{m-1} b_i \alpha^i\right) = \sum_{i=0}^{2m-1} p_i \alpha^i$. Note that every element in $GF(2^m)$ is with the relation $f(x)$ described in Section 2.1.1, such that the terms with order greater than *m*, $\alpha^m, \alpha^{m+1}, \ldots, \alpha^{2m-1}$, can be substituted by the linear combination of standard basis $\{1, \alpha^1, \ldots, \alpha^{m-1}\}$. Thus, we can observe that there are m^2 and gate and about $m \times O(m)$ XOR gates in the substitution for high-order terms.

3.1.2 Massey-Omura multiplier

Here, we introduce the popular version named the bit-serial type of Massey-Omura multiplier. It is based on the normal basis, and the transformation between standard basis and normal basis is introduced in Section 2.1.4. Let $A, B, C \in GF(2^m)$ are represented with normal basis and $C = A \times B$, where $A = \sum_{i=0}^{m-1} a_i \alpha^{2^i}$, $B = \sum_{i=0}^{m-1} b_i \alpha^{2^i}$, and $C = \sum_{i=0}^{m-1} c_i \alpha^{2^i}$. Denote the coefficient-vector of A, B, and C by a, b, and c, and the notation $a^{(i)}$ means A^{2^i}, we have:

$$C = A \times B = [a_0, a_1, \ldots, a_{m-1}] \times \begin{bmatrix} \alpha^{2^0+2^0} & \alpha^{2^0+2^1} & \cdots & \alpha^{2^0+2^{m-1}} \\ \alpha^{2^1+2^0} & \alpha^{2^1+2^1} & \cdots & \alpha^{2^1+2^{m-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{2^{m-1}+2^0} & \alpha^{2^{m-1}+2^1} & \cdots & \alpha^{2^{m-1}+2^{m-1}} \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \end{bmatrix} = a \times M \times b^T, \tag{10}$$

where $M = M_0 \alpha + M_1 \alpha^2 + \cdots + M_{m-1} \alpha^{2^{m-1}}$, such that

$$c_{m-1-i} = a \times M_{m-1-i} \times b^T = a^{(i)} \times M_{m-1} \times (b^{(i)})^T. \tag{11}$$

Using Equation (11), the bit-serial Massey-Omura multiplier can be designed as following:

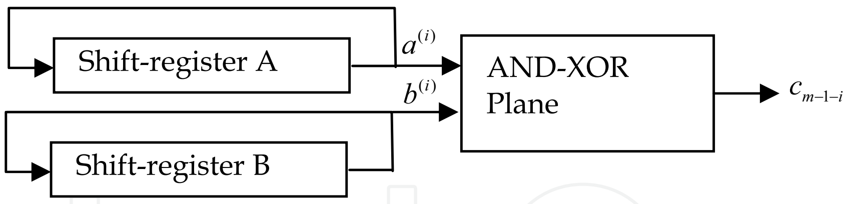


Fig. 1. The Massey-Omura bit-serial multiplier

In Fig. 1, the two shift-register perform the square operation in normal basis, and the complexity of and-xor plane is about $O(m)$ and relative to the number of nonzero element in M_{m-1-i}. Therefore, Massey-Omura multiplier is suitable to the design of area-limited circuits.

3.2 Inverse

In general the inverse circuit is usually with the biggest area and time complexity among other operations. There are two main methods to implement the finite field inverse, that is, multiplicative inversion and inversion based on composed field. The first method decomposes inversion by multiplier and squaring, and the optimal way for decomposing is proposed by Itoh and Tsujii (Itoh & Tsujii, 1988). The later one is based on the composed field and suited for area-limited circuits, which has been widely used in many applications.

3.2.1 Multiplicative inversion

From Fermat's theorem, for any nonzero element $\alpha \in GF(2^m)$ holds $\alpha^{2^m-1}=1$. Therefore, multiplicative inversion is equal to α^{2^m-2}. Based on this fact $\alpha^{-1}=\alpha^{2^m-2}=\prod_{i=1}^{m-1}\alpha^{2^i}$, Itoh and Tsujii reduced the number of required multiplications to $O(\log m)$, which is based on the decomposition of integer. Suppose $m-1=\sum_{n=0}^{b-1}a_n \times 2^n$, where $a_n \in GF(2)$ and $a_{b-1}=1$ denoted the decimal number $[1a_{b-2}\ldots a_1a_0]_2$, we have the following facts:

$$\begin{aligned}
2^{m-1}-1 &= (2^{2^{b-1}}-1)\cdot 2^{[a_{b-2}\ldots a_1a_0]_2}+2^{[a_{b-2}\ldots a_1a_0]_2}-1 \\
&= (2^{2^{b-1}}-1)(2^{2^{b-2}}+1)\cdot 2^{[a_{b-2}\ldots a_1a_0]_2}+2^{[a_{b-2}\ldots a_1a_0]_2}-1 \\
&= (2^{2^{b-2}}+1)\cdots(2^{2^1}+1)(2^{1^0}+1)\cdot 2^{[a_{b-2}\ldots a_1a_0]_2}+2^{[a_{b-2}\ldots a_1a_0]_2}-1
\end{aligned} \tag{12}$$

$$\begin{aligned}
2^{[a_{b-2}\ldots a_1a_0]}-1 &= a_{b-2}(2^{2^{b-2}}-1)2^{[a_{b-3}\ldots a_1a_0]_2}+2^{[a_{b-3}\ldots a_1a_0]_2}-1 \\
&= a_{b-2}(2^{2^{b-2}}-1)\cdots(2^{2^1}+1)(2^{1^0}+1)2^{[a_{b-3}\ldots a_1a_0]_2}+2^{[a_{b-3}\ldots a_1a_0]_2}-1
\end{aligned} \tag{13}$$

$$\begin{aligned}
2^{m-1}-1 &= (2^{2^{b-2}}+1)\cdots(2^{2^1}+1)(2^{1^0}+1)\cdot 2^{[a_{b-2}\ldots a_1a_0]_2}+2^{[a_{b-2}\ldots a_1a_0]_2}-1 \\
&= ((2^{2^{b-2}}+1)2^{2^{b-2}})(2^{2^{b-3}}+1)\cdots(2^{2^1}+1)(2^{2^0}+1)\cdot 2^{[a_{b-3}\ldots a_1a_0]_2}+ \\
&\quad a_{b-2}(2^{2^{b-3}}+1)\cdots(2^{2^1}+1)(2^{2^0}+1)2^{[a_{b-3}\ldots a_1a_0]_2}+2^{[a_{b-3}\ldots a_1a_0]_2}-1 \\
&= ((2^{2^{b-2}}+1)2^{2^{b-2}}+a_{b-2})(2^{2^{b-3}}+1)\cdots(2^{2^1}+1)(2^{2^0}+1)+2^{[a_{b-3}\ldots a_1a_0]_2}-1 \\
&= (((\cdots((2^{2^{b-2}}+1)2^{2^{b-2}}+a_{b-2})(2^{2^{b-3}}+1)2^{2^{b-3}}+a_{b-3})\cdots)(2^{2^2}+1)2^{2^2}+a_2) \\
&\quad (2^{2^1}+1)2^{2^1}+a_1)(2^{2^0}+1)2^{2^0}+a_0
\end{aligned} \tag{14}$$

This algorithm requires $N_M = len.(m-1) + wt.(m-1) - 2$ multipliers, and $N_P = len.(m-1) + wt.(m-1) - 1$ square circuits, where $len.(m-1)$ the length of binary representation of $m-1$ and $wt.(m-1)$ is the number of nonzero bit in the representation. For instance, if $m=8$ then $m-1=7$, $N_M = len.(7) + wt.(7) - 2 = 3+3-2 = 4$ and $N_P = len.(7) + wt.(7) - 1 = 3+3-1 = 5$. For the latency of circuit, it takes $(\lceil \log_2 (m-1) \rceil) T_M + (\lceil \log_2 (m-1) \rceil + 1) T_S$, where T_M (resp. T_S) is the latency of multiplier (resp. squaring circuit). We list some results of this algorithm as Table 2.

m	area	latency	*m*	area	latency
5	2 NM +3 NP	2 TM +3 TP	11	4 NM +5 NP	4 TM +5 TP
6	3 NM +4 NP	3 TM +4 TP	12	5 NM +6 NP	4 TM +5 TP
7	3 NM +4 NP	3 TM +4 TP	13	4 NM +5 NP	4 TM +5 TP
8	4 NM +5 NP	3 TM +4 TP	14	5 NM +6 NP	4 TM +5 TP
9	3 NM +4 NP	3 TM +4 TP	15	5 NM +6 NP	4 TM +5 TP
10	4 NM +5 NP	4 TM +5 TP	16	6 NM +7 NP	4 TM +5 TP

Table 2. The list of Itoh and Tsujii algorithm

3.2.2 Composite field inversion

The use of composite field provides an isomorphism for $GF(2^m)$, while *m* is not prime. Especially, if *m* is even, then inverse using composite field is with very low complexity. Consider the inverse in $GF((2^{m/2})^2)$ where *m* is even. Suppose $A, B \in GF((2^{m/2})^2)$ constructed by an irreducible polynomial $P(x) = p_1 x + p_0$, where $p_0, p_1 \in GF(2^{m/2})$. Let $A = a_1 x + a_0$ and $B = b_1 x + b_0$, where $a_1, a_0, b_1, b_0, p_1, p_0 \in GF(2^{m/2})$. Assume that *B* is the inverse of *A*, thus $A \times B = 1$ or $(a_1 x + a_0) \times (b_1 x + b_0) \equiv 1$ modulo $P(x)$. After the distribution, one has $A \times B = (a_1 b_1 p_1 + a_1 b_0 + a_0 b_1) x + (a_1 b_1 p_0 + a_0 b_0) = 1$. Therefore, $a_1 b_1 p_1 + a_1 b_0 + a_0 b_1 = 0$ and $a_1 b_1 p_0 + a_0 b_0 = 1$. Let $\Delta = (a_0{}^2 + a_0 a_1 p_1 + p_0 a_1{}^2)$, one has $b_1 = a_1 \Delta^{-1}$ and $b_0 = (a_0 + a_1 p_1) \Delta^{-1}$, which is design as Fig. 2. Obviously, one can observe the inversion in $GF(2^m)$ is executed by several operations which are all in $GF((2^{m/2})^2)$, thus the total gated count used can be reduced.

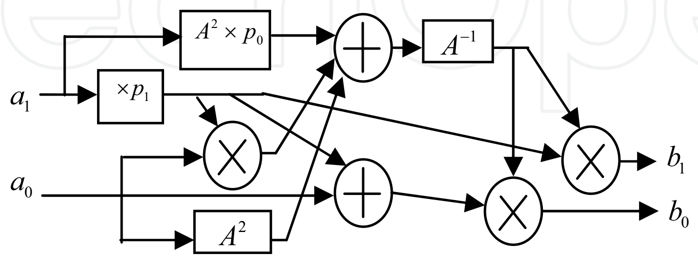


Fig. 2. The circuit for composite field inversion

4. Some techniques for simplification of VLSI

4.1 Finding common sharing resource in various design levels

Sharing resource is a common method to reduce the area cost. This skill can be used in different design stages. For example, consider the basis transformation in Section 2.1.4, the element of normal basis is obtained by the linear combination of standard basis as follows:

$$\gamma^8 = \alpha^1 + \alpha^0,\ \gamma^4 = \alpha^2 + \alpha^0,\ \gamma^2 = \alpha^2 + \alpha^1 + \alpha^0,\ \gamma^1 = \alpha^3 + \alpha^2 + \alpha^1 + \alpha^0. \tag{15}$$

It takes 7 XOR gates for the straightforward implementation. However, if one calculate the summation $t = \alpha^2 + \alpha^1 + \alpha^0$ firstly, then $\gamma^2 = t$ and $\gamma^1 = \alpha^3 + t$. Therefore, the number of XOR gates is reduced to 5. Although it is effective in the bit-level, this idea is also effective in other design stages. Consider another example in previous section, when we form those components $\Delta = (a_0{}^2 + a_0 a_1 p_1 + p_0 a_1{}^2)$ and $b_0 = (a_0 + a_1 p_1)\Delta^{-1}$, it takes 3 2-input adders in two expressions. Suppose we form the component $a_0 + a_1 p_1$ firstly, thus the number of 2-input adder is reduced from 3 to 2 ($\Delta = (a_0(a_0 + a_1 p_1) + p_0 a_1{}^2)$). Therefore, the resource-sharing idea is suitable to different design stages.

4.2 Finding the optimal parameters of components

Another technique used to simplify circuits for finite field operations is change the original field to another isomorphism. Although these methods are equal in mathematics, it provides different outcomes in VLSI designs. There are two main methods to be realized.

4.2.1 Change the relation polynomial

Consider the implementations of hardware multiplier/inverse in $GF(2^8)$ using FPGA, we gather area statistics of multiplier/inverse by using different irreducible polynomials ($f(x)$) and draw the line chart as Fig. 3 and Fig. 4, where the X axis indicates various irreducible polynomials in decimal representation and the Y axis is the number of needed XOR gates. In Fig. 3, one can observe the lowest complexity of area and delay is with $f(x)$ is 45. The maximum difference of XOR number (resp. delay) between two polynomials is 50 (resp. 2). Therefore, choosing the optimal parameters has great influence in complexity in VLSI. The same phenomenon is also been observed in Fig. 4, the maximum difference is 196 XOR gates.

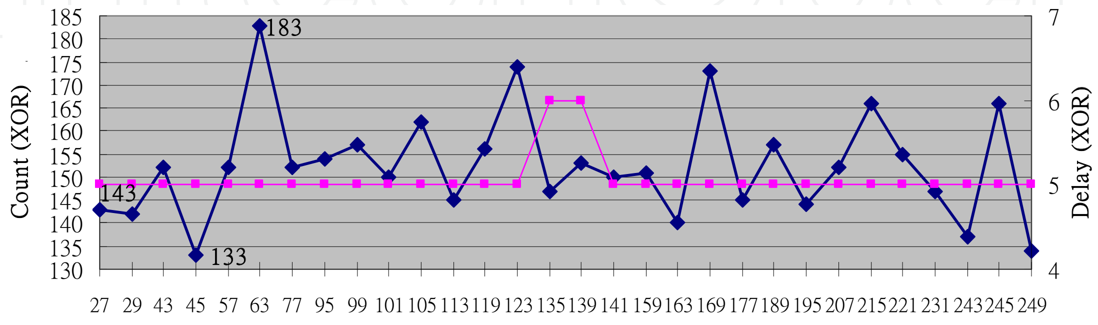


Fig. 3. The statistic of area for multiplier v.s. $f(x)$

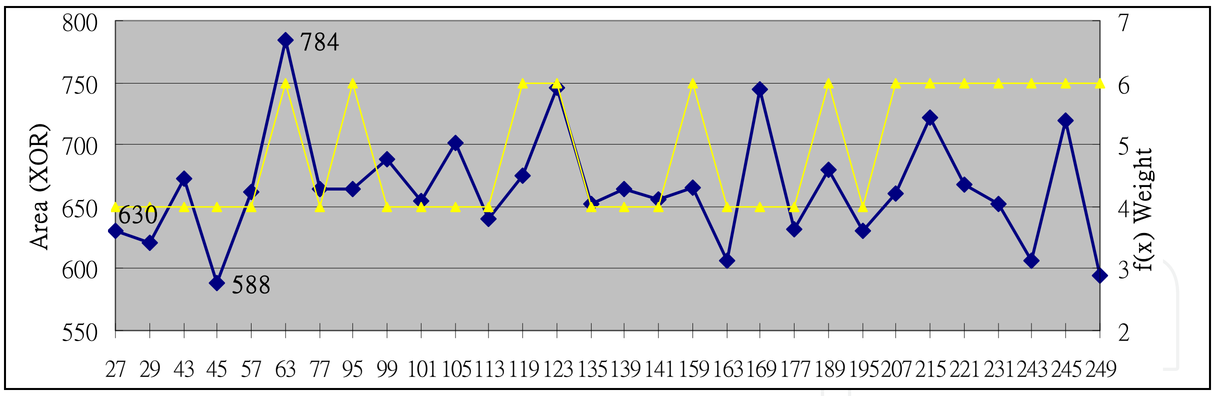


Fig. 4. The statistic of area for inverse v.s. $f(x)$

4.2.2 Using composite field

In Section 2.1.3, we illustrate the transformation between a finite field represented by standard basis and a composite field. The most applications for composite field are to design the inverse, for instance, the S-box in AES algorithm (Morioka & Satoh, 2003). As we know, the main component in S-box is the finite field inverse of $GF(2^8)$. Here, we implement the S-box by the multiplicative inversion described in Section 3.2.1 and by using composite field $GF((2^4)^2)$ described in 3.2.2 as Table 3 by using the Altera FPGA Stratix 2S1020C4 device. Obviously, the later method is with more advantages for both area and time complexity than that of previous one.

Method	LE/ALUT	Delay (ns)	CLK (MHz)	Throughput (MHz)
mult. inverse	210	23.240	43.029	344.232
composite field	82	20.219	49.458	395.664

Table 3. The results for S-box using multiplicative inversion and using composite field

5. Using computer-aided functions to choose suitable parameters

According to the explanations in Section 4, we can realize the related VLSI IPs using various parameters to bring the benefits for lower area or time complexity. However, there exist so many isomorphisms in using finite filed, it seems that there are so many procedures and variations to choose the parameters and hard to find a better ones. As a result, our group developed a software tools which is the computer-aided design (CAD) to help engineers to do the tedious analysis and search. This section will introduce the methods to apply the isomorphism transformations between $GF(2^8)$ and $GF((2^4)^2)$ illustrated in Section 4.1 and 4.2 step by step.

Firstly, list all irreducible and primitive polynomials in two fields as shown in Table 4 and 5, respectively. In this table, all irreducible and primitive polynomials are represented in hexadecimal form and we omit the most significant bit. For example, in Table 4, one chooses 1B that means $(00011011)_2$ or $x^8+x^4+x^3+x+1$; in Table 5, suppose the primitive element $\omega \in GF(2^4)$, one chooses 18 that means $(00011000)_2$ or $x^2+1x+\omega^3$.

$GF(2^8)$	irreducible polynomials															
#=30	1B	1D	2B	2D	39	3F	4D	5F	63	65	69	71	77	7B	87	8B
	8D	9F	A3	A9	B1	BD	C3	CF	D7	DD	E7	F3	F5	F9		
	primitive polynomials															
#=16	1D	2B	2D	4D	5F	63	65	69	71	87	8D	A9	C3	CF	E7	F5

Table 4. The irreducible and primitive polynomials in $GF(2^8)$

$GF((2^4)^2)$	irreducible polynomials															
#=120	18	19	1A	1B	1C	1D	1E	1F	21	22	25	26	29	2A	2D	2E
	31	33	34	36	39	3B	3C	3E	41	42	44	47	48	4B	4D	4E
	51	53	55	57	59	5B	5D	5F	62	63	64	65	6A	6B	6C	6D
	72	73	74	75	78	79	7E	7F	81	83	84	86	88	8A	8D	8F
	92	93	96	97	9A	9B	9E	9F	A1	A2	A4	A7	A9	AA	AC	AF
	B4	B5	B6	B7	BC	BD	BE	BF	C1	C3	C5	C7	C8	CA	CC	CE
	D4	D5	D6	D7	D8	D9	DA	DB	E2	E3	E6	E7	E8	E9	EC	ED
	F1	F2	F5	F6	F8	FB	FC	FF								
	primitive polynomials															
#=60	19	1B	1D	1E	22	25	29	2D	2E	33	34	39	3B	3E	42	44
	55	59	5B	5D	62	63	64	65	6B	6D	72	73	74	75	79	7E
	83	84	8D	92	93	9B	9E	A2	A4	A9	B4	B5	BD	BE	C3	C5
	CE	D4	D5	D9	DB	E2	E3	E9	ED	F2	F5	FB				

Table 5. The irreducible and primitive polynomials in $GF((2^4)^2)$

Secondly, the CAD searches for all possible combinations by the proposed algorithm as shown in Table 6. This algorithm regards as a function used to find transformation matrices as shown in Table 7. After we gather all results, we can choose the better parameters from the list of analyzed results for hardware design of new IP.

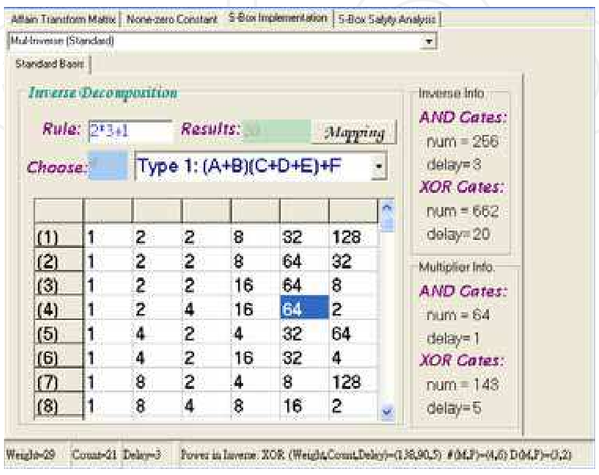
Chose relation polynomial 1D for $GF(2^8)$ ➔ $p(x) = x^8 + x^4 + x^3 + x^2 + 1$. Let $p(\alpha) = 0$, such that $\forall \beta \in GF(2^8)$ can be expressed by binary form as $(\alpha_7, \alpha_6, \alpha_5, \alpha_4, \alpha_3, \alpha_2, \alpha_1, \alpha_0)$	
Step 1:	Find a $GF((2^4)^2)$ irreducible polynomial. Select an irreducible polynomial in ground field $GF(2^4)$ is $f_1(x) = x^4 + x + 1$. Let ω be the root of $f_1(x)$, thus $f_1(\omega) = \omega^4 + \omega + 1 = 0$. Select an irreducible polynomial in $GF((2^4)^2)$ is 18➔ $f_2(x) = x^2 + x + \omega^3$ and let γ be the root of $f_2(x)$, $f_2(\gamma) = \gamma^2 + \gamma + \omega^3 = 0$.
Step 2:	Assume a generator σ in $GF((2^4)^2)$ and generate all none-zero elements of $GF((2^4)^2)$. For any element in $GF((2^4)^2)$ can be expressed in binary form as $(\gamma_{13}, \gamma_{12}, \gamma_{11}, \gamma_{10}, \gamma_{03}, \gamma_{01}, \gamma_{02}, \gamma_{03})$. Assume the T matrix = $\left[(\sigma^7)^T \quad (\sigma^6)^T \quad \dots \quad (\sigma^0)^T\right]_{8\times 8}$, we have

	$\begin{bmatrix}\gamma_{13}\\ \gamma_{12}\\ \gamma_{11}\\ \gamma_{10}\\ \gamma_{03}\\ \gamma_{02}\\ \gamma_{01}\\ \gamma_{00}\end{bmatrix}_{8\times1} = \left[(\sigma^7)^T \quad (\sigma^6)^T \quad \dots \quad (\sigma^0)^T\right]_{8\times8} \begin{bmatrix}\alpha_7\\ \alpha_6\\ \alpha_5\\ \alpha_4\\ \alpha_3\\ \alpha_2\\ \alpha_1\\ \alpha_0\end{bmatrix}_{8\times1}$
Step 3:	Compute the T^{-1} matrix= $\left[(\sigma^7)^T \quad (\sigma^6)^T \quad \dots \quad (\sigma^0)^T\right]^{-1}_{8\times8}$. Put all none-zero elements $GF((2^4)^2)$ in the following equation and check if they all hold, i.e., $[\alpha^i]^T = T^{-1}[\sigma^i]^T$, *where* $0 \le i \le 254$.
Step 4:	If Step 3 does not hold, return to Step 2 and choose another generator; If Step 3 holds, then the T and T^{-1} matrices are found.

Table 6. The proposed algorithm for searching transformation matrices

input	$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \qquad p(x) \Rightarrow GF(2^8)$ $f_1(x) = x^4 + x + 1 \qquad f_1(\omega) = \omega^4 + \omega + 1 = 0$ $f_2(x) = x^2 + x + \omega^3 \qquad f_1(x), f_2(x) \Rightarrow GF((2^4)^2)$
output	$T = \begin{bmatrix}0&0&1&0&0&0&0&0\\0&1&1&0&0&1&0&0\\0&0&0&1&1&0&1&0\\1&0&0&1&0&0&0&0\\0&1&1&1&1&0&0&0\\0&1&0&1&0&1&0&0\\1&1&0&0&1&1&0&0\\1&1&0&1&1&1&1&1\end{bmatrix}_{8\times8}, T^{-1} = \begin{bmatrix}1&1&0&1&0&1&0&0\\0&1&0&1&1&0&1&0\\1&0&0&0&0&0&0&0\\1&1&0&0&0&1&0&0\\0&0&0&1&0&1&1&0\\1&0&0&1&1&0&1&0\\1&1&1&1&0&0&1&0\\0&0&1&1&0&1&0&1\end{bmatrix}_{8\times8}$

Table 7. The result of transformation matrices between $GF(2^8)$ and $GF((2^4)^2)$.



Rule: 2*3+1 Results:

Choose: Type 1: (A+B)(C+D+E)+F

(1)	1	2	2	8	32	128
(2)	1	2	2	8	64	32
(3)	1	2	2	16	64	8
(4)	1	2	4	16	64	2
(5)	1	4	2	4	32	64
(6)	1	4	2	16	32	4
(7)	1	8	2	4	8	128
(8)	1	8	4	8	16	2

Inverse Info
AND Gates: num = 256 delay=3
XOR Gates: num = 662 delay=20

Multiplier Info
AND Gates: num = 64 delay=1
XOR Gates: num = 143 delay=5

Weight=29 Count=21 Delay=3 Power in Inverse: XOR (Weight,Count,Delay)=(136,90,5) #(M,F)=(4,6) D(M,F)=(3,2)

Fig. 5. The interface of CAD tool

Because various parameters provide VLSI's outputs with huge variation and it's seem impossible to run all parameters, we should provide engineers a CAD tool to obtain the analyzed algorithm and results. In Fig. 5, a designer can use a CAD with Windows interface to find better parameters of S-box. In this CAD, it provides the complexity information of the multipliers or the inverse in $GF(2^8)$. In this figure, designer chooses the fourth result, and the estimative complexity of inverse is shown in the top right of the figure; that the choice of multiplier is shown under the inverse information. Therefore, the CAD tool helps designer to choose the better parameters efficiently.

6. Summary

In this chapter, we introduce the common concepts of finite field regarding its applications in error correcting coding, cryptography, and others, including the mathematical background, some important designs for multiplier and inversion, and the idea to utilize the computer-aided functions to find better parameters for IP or system design. The summary of this chapter is as follows:

1. Introducing the basic finite field operations and their hardware designs: Those common operations include addition, multiplication, squaring, inversion, basis transformation, and so on. The VLSI designs of those operations may be understood through the mathematical background provided in Section 2. From the mathematical background, one should realize the benefits and the processes of the transformation between two isomorphic finite fields.
2. Using some techniques to simplify the circuits: We have introduced some useful techniques to reduce the area cost in VLSI design, such as the resource-sharing method, utilization of different parameters, or use some isomorphic field to substitute the used field. The first technique is widely used in various design stages. The later two techniques depend on the parameters used. Different parameters lead to different hardware implementation results. However, it seems infeasible to analyze all possible parameters manually.
3. Using the composite field inversion: Composite field inversion is used in the finite field inversion due to its superiority in hardware implementation. The main idea is to consider the use of intermediate fields and decompose the inversion on the original field into several operations on smaller fields. This method has been used in the AES S-box design to minimize the area cost.
4. Calculating the transformation matrices between isomorphic finite fields. It is well known that finite fields of the same order are isomorphic, and this implies the existence of transformation matrices. Finding the optimal one is important in the investigation of the VLSI designs. Two methods are presented; one is to change the relation polynomial, and the other is to use the composite field. An algorithm to calculate the transformation matrices is provided in Section 5, and it can be used to find the optimal one.
5. Using the computer-aided design to search for better parameters: A good hardware CAD tool provides fast search and enough information for designer, because it brings fast and accurate designs. In Section 5, the computer-aided function based on the proposed algorithms is one of the examples. When the order of the finite field gets large, the number of isomorphic field increases rapidly. This makes it almost impossible to do the exhausting search, and the proposed CAD can be used to support engineers to get the best choices.

7. Conclusion

In this chapter, we use the concept of composite fields for the CAD designs, which can support the VLSI designer to calculate the optimal parameters for finite field inversion.

8. Acknowledgments

This work is supported in part by the Nation Science Council, Taiwan, under grant NSC 96-2623-7-214-001.

9. References

Dinh, A.V.; Palmer, R.J.; Bolton, R.J. & Mason, R. (2001). A low latency architecture for computing multiplicative inverses and divisions in GF(2^m). *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 48, No. 8, pp. 789-793, ISSN: 1057-7130

Fenn, S.T.J.; Benaissa, M. & Taylor, D. (1996). Fast normal basis inversion in GF(2^m). *Electronics Letters*, Vol. 32, No. 17, pp. 1566-1567, ISSN: 0013-5194

Hsiao, S.-F.; Chen, M.-C. Chen & Tu, C.-S. (2006). Memory-free low-cost designs of advanced encryption standard using common subexpression elimination for subfunctions in transformations. *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 53, No. 3, pp. 615–626, ISSN: 1549-8328

Itoh, T. & Tsujii, S. (1988). A fast algorithm for computing multiplicative inverses in GF(2^m) using normal basis. *Information and Computing*, Vol. 78, No. 3, pp. 171-177, ISSN: 0890-5401

Jing, M.-H.; Chen, Z.-H.; Chen, J.-H. & Chen, Y.-H. (2007). Reconfigurable system for high-speed and diversified AES using FPGA. *Microprocessors and Microsystems*, Vol. 31, No. 2, pp. 94-102, ISSN: 0141-9331

Lidl, R. & Niederreiter, H. (1986). *Introduction to finite fields and their applications*, Cambridge University Press, ISBN: 9780521460941

Lu, C.-C. (1997). A search of minimal key functions for normal basis multipliers. *IEEE Transactions on Computers*, Vol. 46, No. 5, pp.588–592, ISSN: 0018-9340

Morioka, S. & Satoh, A. (2003). An optimized S-box circuit architecture for low power AES design. *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science*, Vol. 2523, pp. 172–186, ISBN: 3-540-00409-2, August, 2002, Redwood Shores, California, USA

Peterson, W.W. & Brown, D.T. (1961). Cyclic Codes for Error Detection, *Proceedings of the IRE,* Vol. 49, No. 1, pp. 228-235, ISSN: 0096-8390

Peterson, W.W. & Weldon, E.J. (1972). *Error-Correcting Codes*, The MIT Press, 2 edition, Cambridge, MA, ISBN: 3540004092

Sunar, B.; Savas, E. & Koc, C.K., (2003). Constructing composite field representations for efficient conversion. *IEEE Transactions on Computer*, Vol. 52, No. 11, pp. 1391-1398, ISSN: 0018-9340

Wang, C.C.; Truong, T.K.; Shao, H.M.; Deutsch, L.J.; Omura, J.K.; & Reed, I.S. (1985). VLSI architecture for computing multiplications and inverses in GF(2^m). *IEEE Transactions on Computers*, Vol. 34, No. 8, pp. 709-716, ISSN: 0018-9340

Wicker, S.B. (1995). *Error Control Systems for Digital Communication and Storage*, Prentice Hall, ISBN: 0-13-308941-X, US

VLSI
Edited by Zhongfeng Wang

ISBN 978-953-307-049-0
Hard cover, 456 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

The process of Integrated Circuits (IC) started its era of VLSI (Very Large Scale Integration) in 1970’s when thousands of transistors were integrated into one single chip. Nowadays we are able to integrate more than a billion transistors on a single chip. However, the term “VLSI” is still being used, though there was some effort to coin a new term ULSI (Ultra-Large Scale Integration) for fine distinctions many years ago. VLSI technology has brought tremendous benefits to our everyday life since its occurrence. VLSI circuits are used everywhere, real applications include microprocessors in a personal computer or workstation, chips in a graphic card, digital camera or camcorder, chips in a cell phone or a portable computing device, and embedded processors in an automobile, et al. VLSI covers many phases of design and fabrication of integrated circuits. For a commercial chip design, it involves system definition, VLSI architecture design and optimization, RTL (register transfer language) coding, (pre- and post-synthesis) simulation and verification, synthesis, place and route, timing analyses and timing closure, and multi-step semiconductor device fabrication including wafer processing, die preparation, IC packaging and testing, et al. As the process technology scales down, hundreds or even thousands of millions of transistors are integrated into one single chip. Hence, more and more complicated systems can be integrated into a single chip, the so-called System-on-chip (SoC), which brings to VLSI engineers ever increasingly challenges to master techniques in various phases of VLSI design. For modern SoC design, practical applications are usually speed hungry. For instance, Ethernet standard has evolved from 10Mbps to 10Gbps. Now the specification for 100Mbps Ethernet is on the way. On the other hand, with the popularity of wireless and portable computing devices, low power consumption has become extremely critical. To meet these contradicting requirements, VLSI designers have to perform optimizations at all levels of design. This book is intended to cover a wide range of VLSI design topics. The book can be roughly partitioned into four parts. Part I is mainly focused on algorithmic level and architectural level VLSI design and optimization for image and video signal processing systems. Part II addresses VLSI design optimizations for cryptography and error correction coding. Part III discusses general SoC design techniques as well as other application-specific VLSI design optimizations. The last part will cover generic nano-scale circuit-level design techniques.