FAST MATRIX SOLVER in GF(2)

Yoshiyasu TAKEFUJI, Takakazu KUROKAWA and Hideo AISO

Department of Electrical Engineering
KEIO University
3-14-1 Hiyoshi, Yokohama 223, JAPAN

Abstract

In this paper a parallel and pipelined fast matrix equation solver in GF(2) is proposed where the elements are 0s or 1s. The solver employing the iterative logic circuits which are suitable for VLSI implementation can be realized by the conventional Gauss Jordan Elimination Method. O(n) gate stages in the pipeline and $O(n^3)$ total gates are required for solving $A\mathbf{X} = \mathbf{b}$ where A is a matrix of n*n, $\mathbf{X}$ and $\mathbf{b}$ are vectors respectively. The organization of the solver is discussed in this paper.

1. Introduction

This paper presents an ultra high speed solver of the regular matrix equations in GF(2)* where the elements are 0s or 1s. The parallel and pipelined solver composed of the iterative logic circuits which are suitable for VLSI implementation can be realized by employing the Gauss Jordan Elimination Method (GJEM). The iteration of the GJEM for solving a matrix equation is shown in Fig.1. The high speed matrix solver in GF(2) is required in various application fields of real-time systems. Concretely the proposed solver is applied to the decryption of an encrypted code as a real-time application.

The pipelined solver consists of n-stage pipes for obtaining $\mathbf{X}$ of

$$A\mathbf{X} = \mathbf{b} \quad \text{in GF(2)}$$

where $\mathbf{X}$ is a vector

$$\mathbf{X} = (x_1 x_2 \cdots x_n)^t,$$

A is a regular matrix of n*n, and $\mathbf{b}$ is also a vector

$$\mathbf{b} = (b_1 b_2 \cdots b_n)^t .$$

The organization of the pipelined solver is outlined in Fig.1. A single pipe described in Fig.2 is composed of a panel of D Flip Flops, column exchangers, a shuffle memory system, and logic units. $O(n^3)$ gates required for solving n*n matrix equations are employed in the solver. Every solution of 100*100 pipelined matrix equations can be obtained in 10 nsec when assuming that the propagation delay time of a single gate is 1 nsec.

* Note that GF(2) means Galois Field 2.

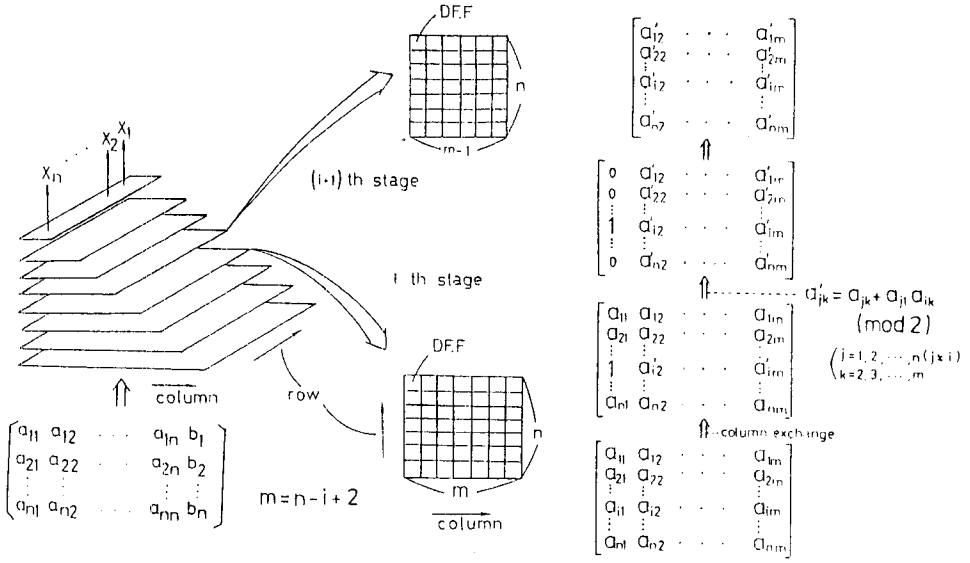


Fig.1 The Gauss Jordan Elimination Method

2. Gauss Jordan Elimination Method (GJEM)

The GJEM composed of two processing stages is discussed in this section. The one is a column exchange stage. The other is a logic stage. The number of iterations in the GJEM shown in Fig.1 for solving a regular matrix equation

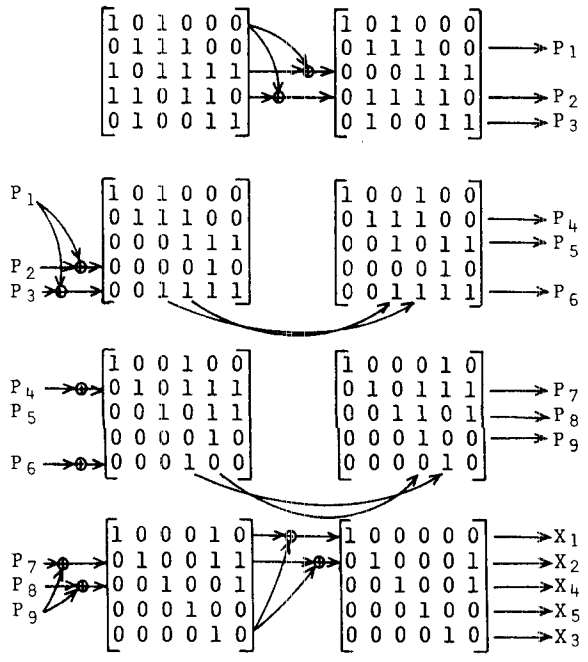
$$A\,X = b \qquad \text{in } GF(2)$$

where A is a regular matrix of $n \times n$ becomes n. Hence, the size of a matrix in the ith stage is $n \times (n-i+2)$. When the leftmost element of the $(n-i+2)$th row in the ith stage is 0, the leftmost column should be exchanged through a column exchanger described in section 3 with another column where the element of the $(n-i+2)$th row is 1.

Let us show you the GJEM in GF(2) in a simple example. Solve $(x_1\ x_2\ x_3\ x_4\ x_5)^t$ in a regular matrix equation

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Solution:



⊕ : Exclusive OR

$x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$, and $x_5 = 0$ are obtained by solving the equation.

3. Pipelined Matrix Solver

The organization of the proposed matrix solver for realizing the GJEM is discussed in this section. In GF(2) the addition and subtraction corresponds to the function of Exclusive OR logic and the multiplication to that of AND logic. The division corresponds to no-operation when the divisor is 1. The organization of the pipelined solver is described in Fig.1 and Fig.2. A single pipe of the ith stage is composed of a panel of $n \times (n-i+2)$ D Flip Flops, $n \times (n-i+2)$ column exchangers, a shuffle memory system, and $n \times (i-1)$ logic units. A panel of D Flip Flops, a 1-detector, a column exchanger, a shuffule memory system, and a logic unit of the ith stage are shown in Fig.3, Fig.4, Fig.5, Fig.6-(a), Fig.6(b), and Fig.7 respectively. The 1-detector in the ith stage detects the leftmost element to be 1 and produces the row number of the element for exchanging the leftmost column with the column of the row number. The row number corresponds to the output of case i in Fig.4. The column exchanger exchanges the required two columns. The shuffle memory system in the ith stage shown in Fig.6-(a) and Fig.6-(b) has the information on the names of columns to be exchanged by the 1-detector. The shuffle memory system is composed of a shuffle memory and a shuffle circuit described in Fig.6-(a) and Fig.6-(b). The logic unit described in Fig.7 calculates the arithmetic operation in GF(2). In order to realize an easy VLSI implementation, the following items should be discussed;

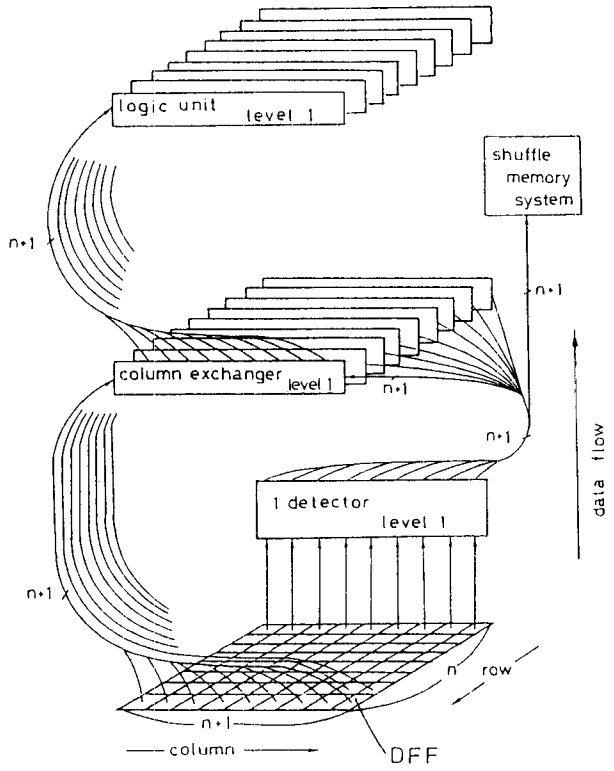


Fig.2 A single pipe of GJEM

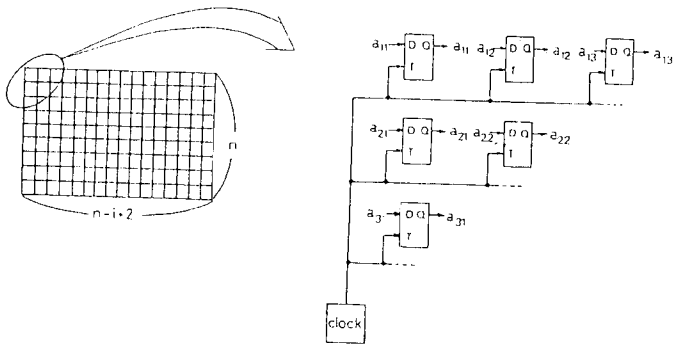


Fig.3 A panel of D Flip Flops of the ith stage

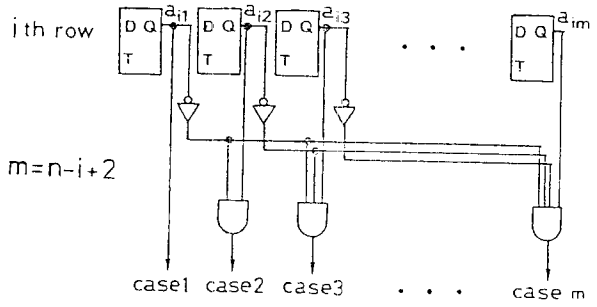
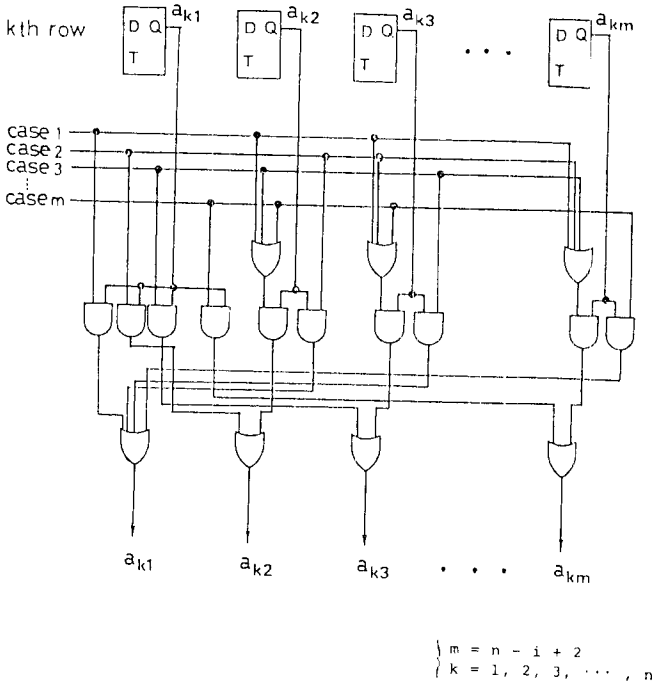


Fig.4 A 1-detector of the ith stage



$$\begin{cases} m = n - i + 2 \\ k = 1, 2, 3, \cdots, n \end{cases}$$

Fig.5 A column exchanger of the ith stage

1. a high regularity for shortening the required period of the VLSI design,

and 2. a flexible expandability toward the increase of the matrix size.

The good VLSI algorithms should satisfy the items.

The proposed solver which is suitable for VLSI implementation can be easily realized by the high regularity of the iterative logic circuits. However the organization of the solver does not have the flexible expandability as a drawback. The number of required gates for organizing a solver vs. the size of a matrix and the number of gate stages in the pipeline vs. the size of a matrix are listed in Fig.8 and calculated in Table 1 A pipelined matrix solver of 3*3 as a simple example is shown in Fig. 9 -20.

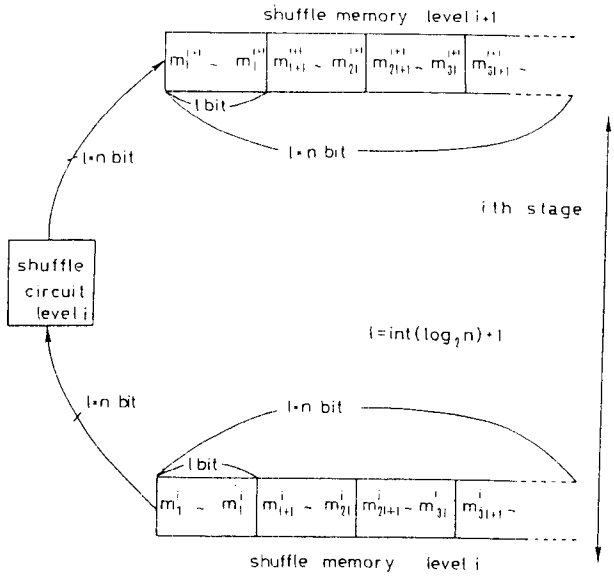
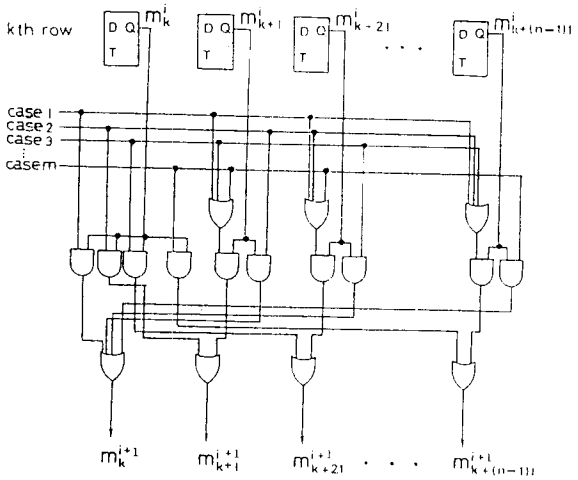


Fig.6-(a) A shuffle memory system of the ith stage



$$\begin{cases} m = n - i + 2 \\ l = \text{int}(\log_2 n) + 1 \\ k = 1, 2, 3, \cdots, l \end{cases}$$

Fig.6-(b) A shuffle circuit of the ith stage

4. Application of Matrix Solver

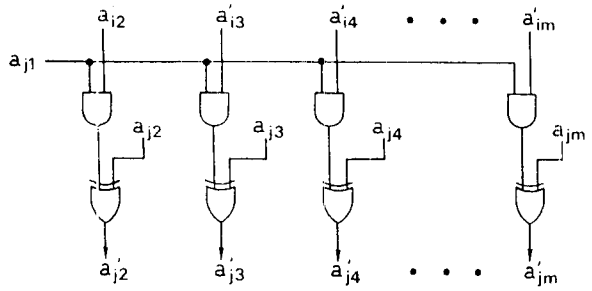
The proposed solver can be applied to the decryption of an encrypted code. Consider multiresidue codes in a polynomial ring as a cipher code. Conversion from multiresidue code to a polynomial corresponds to the function of the decryption. In a polynomial ring a new conversion scheme has been developed in our former paper [1],[3] which corresponds to the conventional Chinese Remainder Theorem [2],[4] or the Mixed Radix Conversion scheme in a number ring [2],[4].

Let us show you our scheme in a simple example. Find a polynomial f(x)

$$f(x) = t_7x^7 + t_6x^6 + t_5x^5 + t_4x^4 + t_3x^3 + t_2x^2 + t_1x + t_0$$

for

$$f_1(x) = x^2 + x + 1 \bmod (x^4+x+1)$$
$$f_2(x) = x^3 + 1 \bmod (x^4+x^3+1).$$



$$\left\{\begin{array}{l} j = 1, 2, \cdots, n \quad (j \neq i) \\ k = 2, 3, \cdots, m \\ m = n - i + 2 \end{array}\right.$$

Fig.7 A logic unit of the ith stage

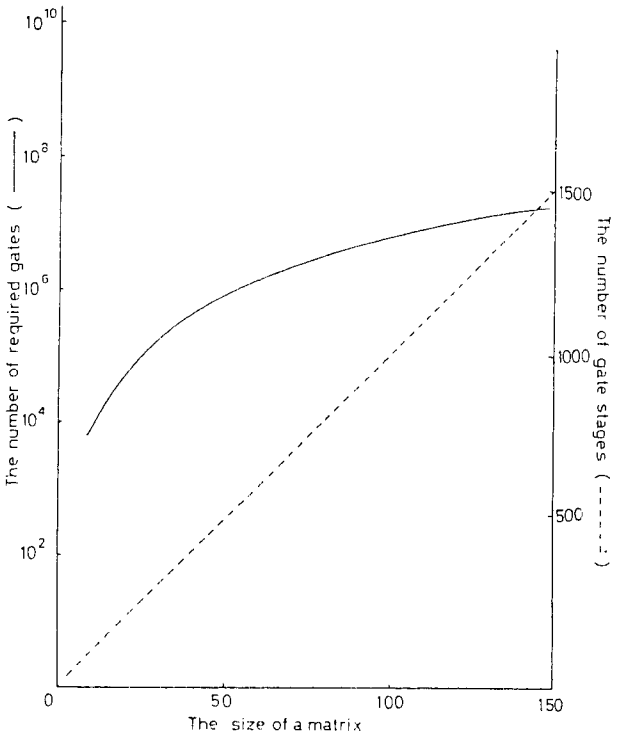


Fig. 8 Evaluation of the matrix solver

Table 1 The number of required gates of the matrix solver

	AND	OR	NOT	XOR	D F.F
Panel of D F.F	0	0	0	0	ni
1-detector	i - 1	0	i - 1	0	0
Column exchanger	3ni - 2n	2ni - n	0	0	0
Shuffle memory system	(3n-2)* int($\log_2$n)	(2n-1)* int($\log_2$n)	0	0	n*int($\log_2$n)
Logic unit	(n-1)(i-1)	0	0	(n-1)(i-1)	0
Total	(4ni-3n) +(3n-2)* int($\log_2$n)	(2ni-n) +(2n-1)* int($\log_2$n)	i - 1	(n-1)i-(n-1)	ni+ n*int($\log_2$n)

The number of required gates = $\frac{13}{2}n^3 + \frac{23}{2}n^2 + \frac{15}{2}n + 1$

$+n(6n-3)*\text{int}(\log_2 n)$

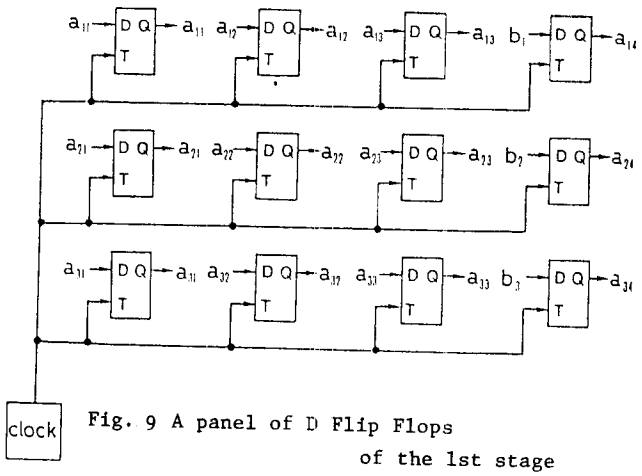


Fig. 9 A panel of D Flip Flops of the 1st stage

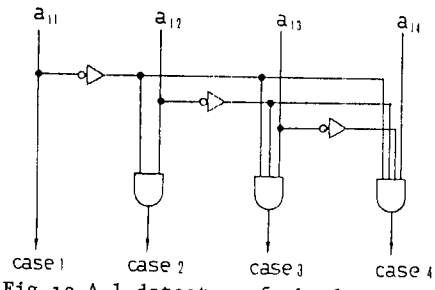


Fig.10 A 1-detector of the 1st stage

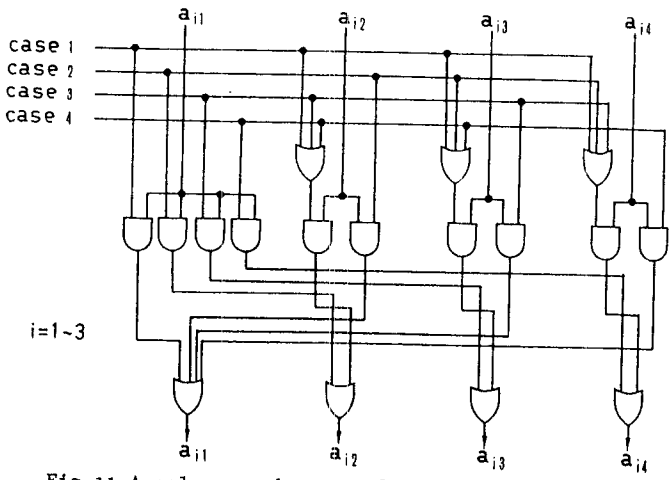


Fig.11 A column exchanger of the 1st stage

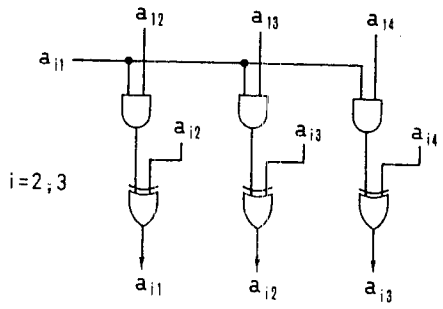


Fig. 12 A logic unit of the 1st stage

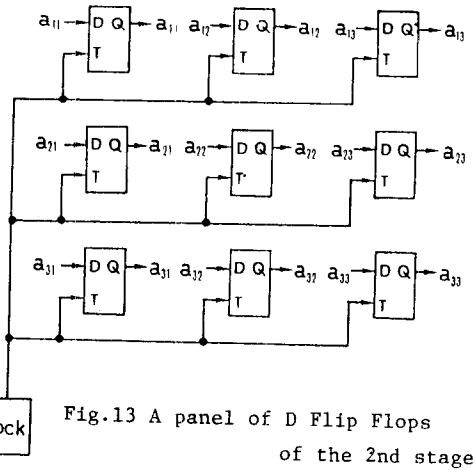


Fig.13 A panel of D Flip Flops of the 2nd stage

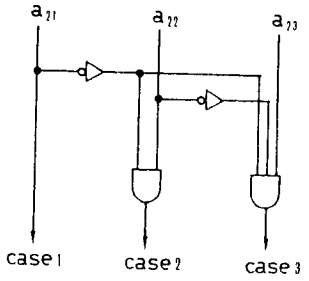


Fig.14 A 1-detector of the 2nd stage

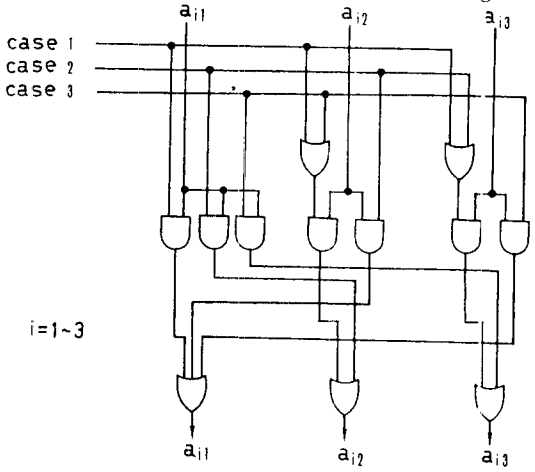


Fig. 15 A column exchanger of the 2nd stage

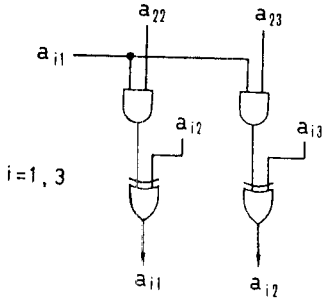


Fig.16 A logic unit of the 2nd stage

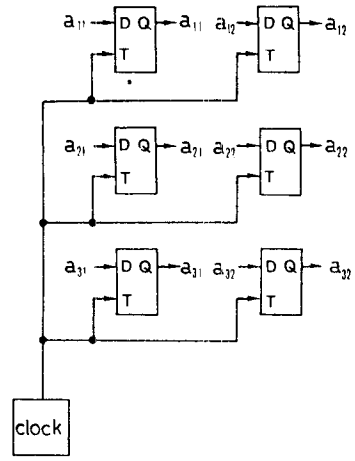


Fig.17 A panel of D Flip Flops of the 3rd stage

Solution :

$$f(x) = f_1(x) = f_2(x)$$
$$f_1(x) = Q_1(x)\cdot(x^4+x+1) + (x^2+x+1)$$
$$f_2(x) = Q_2(x)\cdot(x^4+x^3+1) + (x^3+1)$$

where $Q_1(x) = a_1x^3 + b_1x^2 + c_1x + d_1$
$Q_2(x) = a_2x^3 + b_2x^2 + c_2x + d_2$

In order to find f(x), we should determine $Q_1(x)$ or $Q_2(x)$. To determine $Q_1(x)$ or $Q_2(x)$, the equation

$$f_1(x) + f_2(x) = 0 \quad \text{in GF(2)} \qquad (1)$$

is employed where the equation is always satisfied in GF(2). Hence, equation (1) presents the following matrix equation:

$$\begin{bmatrix} 1&0&0&0&1&0&0&0 \\ 0&1&0&0&1&1&0&0 \\ 0&0&1&0&0&1&1&0 \\ 1&0&0&1&0&0&1&1 \\ 1&1&0&0&1&0&0&1 \\ 0&1&1&0&0&1&0&0 \\ 0&0&1&1&0&0&1&0 \\ 0&0&0&1&0&0&0&1 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \\ a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$a_1 = 0$, $b_1 = 1$, $c_1 = 1$, $d_1 = 0$
$a_2 = 0$, $b_2 = 1$, $c_2 = 0$, and $d_2 = 0$ are obtained by solving Eq.(2). We can obtain f(x);

$$f(x) = (x^2 + x)\cdot(x^4 + x + 1) + (x^2 + x + 1) = x^6 + x^5 + x^3 + x^2 + 1 .$$

5. Conclusion

A fast matrix solver for solving A X = b in GF(2) is discussed from the viewpoints of the speed of the solution and the number of required gates through considering the parallel and pipelined organization.

O(n) gate stages in the pipeline and $O(n^3)$ total gates are required for solving A X = b where A is a regular matrix of n n. The decryption of an encrypted code as an application of the proposed solver is shown in this paper.

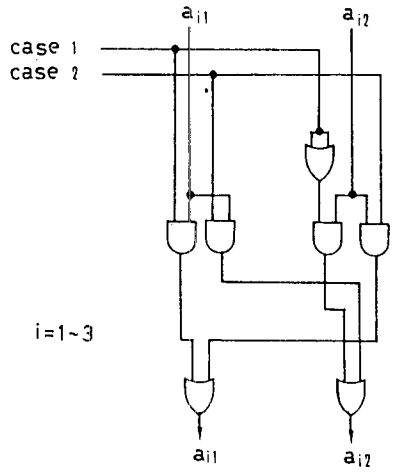


Fig.19 A column exchanger of the 3rd stage

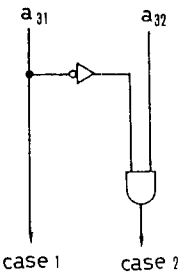


Fig.18 A 1-detector of the 3rd stage

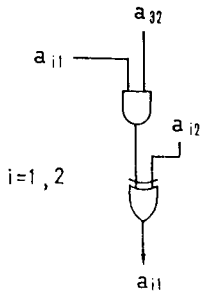


Fig.20 A logic unit of the 3rd stage

References

[1] Takakazu Kurokawa, BS paper "A proposal of multiresidue codes in polynomial ring applied to cipher codes," (1983).

[2] Nicholas S. Szabo, Richard Tanaka, "Residue Arithmatic and its Applications to computer technology," McGraw Hill (1967).

[3] Yoshiyasu Takefuji, Dissertation, "A STUDY OF FAULT-TOLERANT HARDWARE," (1983).

[4] Yoshiyasu Takefuji, Koichiro Tsujino, Mari Ibuki and Hideo Aiso, "A NOVEL APPROACH TO PARALLEL PROCESSING CRYPTOSYSTEM," Proc. of ICPP (1982).