

A New VLSI Complex Integer Multiplier Which Uses a Quadratic-Polynomial Residue System With Fermat Numbers

T. K. Truong and I. S. Hsu

Communications Systems Research Section

J. J. Chang

Spacecraft Data Systems Section

H. C. Shyu and I. S. Reed

Department of Electrical Engineering
University of Southern California

A quadratic-polynomial Fermat residue number system (QFNS) has been used to compute complex integer multiplications. The advantage of such a QFNS is that a complex integer multiplication requires only two integer multiplications. In this article, a new type Fermat number multiplier is developed which eliminates the initialization condition of the previous method. It is shown that the new complex multiplier can be implemented on a single VLSI chip. Such a chip is designed and fabricated in CMOS-pw technology.

I. Introduction

The era of very large scale integrated (VLSI) circuits has arrived. VLSI systems have the characteristic of being compact, high speed and of low power consumption. Therefore, a large system can be integrated into a VLSI chip. Many systems which were realized in discrete components can be improved dramatically by taking advantage of VLSI technology.

Since the available area on a chip is limited, a residue number system can be introduced in order to reduce the computing complexity. The ring of integers modulo the Fermat number $F_n = 2^{2^n} + 1$ has some special simplifying characteristics for residue number systems. Recently, Leibowitz (Ref. 2) developed a binary arithmetic for implementing the Fermat number transform (FNT). In this development a special representation of binary numbers, the diminished-1

representation, was introduced. Arithmetic operations using this representation were developed also in Ref. 2. The FNT has been widely discussed in many papers (Refs. 3-5). Recently, the authors (Ref. 6) developed a VLSI architecture for a modified Leibowitz multiplier of integers modulo a Fermat number. This bit-modulo multiplier uses only addition and cyclic shifts. With this architecture a single chip integer multiplier was designed, fabricated, and proved to work well.

Recently, the authors (Ref. 1) also developed an algorithm to compute the DFT using the residue Fermat number systems. In this algorithm, a complex multiplier was developed which used the direct sum of two copies of the residue ring of integers modulo F_n . The advantage of this approach is that the operations for computing the complex multiplier need only two integer multiplications in Z_{F_n} , the residue ring of integers modulo F_n . Hence the number of multiplications required for

computing a systolic array of the DFT can be reduced substantially over previous approaches. The basic unit of such a complex multiplier is the integer multiplier over Z_{F_n} . In this article, the integer multiplier in Ref. 6 is modified. This new multiplier uses the diminished-1 representation for both input numbers. The second improvement is that the new design does not require the computation of an initial value. Hence considerable computation time and hardware can be saved. Thus, the new integer multiplier is easier to connect to a quadratic-polynomial residue Fermat number system for computing complex multiplications. As a consequence the new complex integer multiplier unit is readily implemented on a single VLSI chip.

II. The Integer Modulo F_n Multiplier

Integer multiplication with small dynamic range is often implemented by look-up tables. When the dynamic range is large, however, this method is undesirable (Ref. 13). Hence an arithmetic algorithmic solution for implementing the multiplier is needed. In Leibowitz's paper (Refs. 1 and 2), general multiplication has the form

$$(A \cdot B - 1) = (A - 1) \cdot (B - 1) + (A + B - 1) - 1 \quad (1)$$

in the diminished-1 representation.

In the algorithm developed in Ref. 2 the binary multiplication of $(A - 1) \cdot (B - 1)$ was computed first. Then the term $(A + B - 1)$ was added to the result in the diminished-1 representation. In Ref. 6, a new multiplier was designed which uses the diminished-1 representation of numbers. The product of two integers, A and B , was obtained as

$$(A \cdot B - 1) = (A - 1) \cdot B + \bar{D} + 1 \quad (2)$$

where \bar{D} is an initial value calculated from the number of ones in B , and where $(x - 1)$ denotes the diminished-1 representation of the number x . The main disadvantage of the multiplier in Ref. 6 is that the initial value \bar{D} must be calculated before the process starts. The use of different representation of numbers, i.e., the diminished-1 representation of A along with the ordinary binary representation of B , leads to some confusion.

A new modulo F_n multiplier is now derived. Let $F_n = 2^n + 1$ be a Fermat number where n is a positive integer. Also let $+$ and \cdot be the symbols for addition and multiplication and \oplus and $*$ be the symbols for the diminished-1 addition and multiplication. Also let \sum denote the diminished-1 summation (i.e., consecutive diminished-1 additions). Then the following theorem holds.

Theorem 1: Let A and B be two elements of the finite residue ring of integers modulo F_n , i.e., Z_{F_n} . Then the diminished-1 multiplication can be calculated within Z_{F_n} as follows:

$$(A - 1) \cdot (B - 1) = (A \cdot B - 1) = \sum_{i=0}^{2^n} (b_i 2^i A - 1) + (A - 1) \quad (3)$$

where

$$(B - 1) = \sum_{i=0}^{2^n} b_i 2^i, \quad b_i = 0 \text{ or } 1$$

is the binary representation of $B - 1$.

Proof: From Ref. 2, it is known that

$$(A \cdot B - 1) = (A - 1) \cdot (B - 1) + (B - 1) + (A - 1)$$

But

$$B - 1 = \sum_{i=0}^{2^n} b_i 2^i$$

Thus,

$$\begin{aligned} (A \cdot B - 1) &= \sum_{i=0}^{2^n} (b_i 2^i A - 1) - \sum_{i=0}^{2^n} b_i 2^i + \sum_{i=0}^{2^n} 1 \\ &\quad + (A - 1) + \sum_{i=0}^{2^n} b_i 2^i \\ &= \sum_{i=0}^{2^n} (b_i 2^i A - 1) + 2^n + 1 + (A - 1) \end{aligned}$$

Hence,

$$(A \cdot B - 1) = \sum_{i=0}^{2^n} b_i 2^i (A - 1) + (A - 1) \quad (4)$$

Note that the diminished-1 addition can be performed by an adder of the type used in Ref. 6.

The recursive architecture for computing developed in Ref. 6 can be modified to compute Eq. (4). To see this, let the initial value be $C_0 = A - 1$. Then the multiplication algorithm in Eq. (4) can be put into the following recursive form:

$$C_{i+1} = C_i + (b_i 2^i A - 1) + 1 = C_i \oplus (b_i 2^i A - 1) \quad (5a)$$

if one successively computes C_{i+1} in Eq. (5a) for $0 \leq i \leq 2^n$, the required result is obtained as follows:

$$C = A \cdot B - 1 = C_{2^{n+1}} = C_{2^n} \oplus (b_{2^n} 2^{2^n} \cdot A - 1) \quad (5b)$$

In Eq. (5a), one observes that if $b_i = 1$, then $C_{i+1} = C_i \oplus (2^i A - 1)$ and if $b_i = 0$, then $C_{i+1} = C_i - 1 + 1 = C_i$. In other words, $(2^i \cdot A - 1)$ is added into C_i for $b_i = 1$ and no operation is needed for $b_i = 0$.

This new algorithm does not need a calculation of the initial value and to transform $B - 1$ to B . Now let the new algorithm be illustrated by an example for $F_2 = 2^4 + 1$. The same structure clearly extends to more general multiply algorithms over Z_{F_n} .

Consider an example in F_2 . The elements in $GF(2^4 + 1)$ with their decimal equivalents in a normal binary representation along with their values in the diminished-1 representation are shown in Table 1.

Example 1: Let $A - 1 = 01010$ and $B - 1 = 00101$. Compute $C = (A \cdot B - 1) = 01010 \times 00101$ modulo $2^4 + 1$.

To compute C , let $C_0 = A - 1 = 01010$ and $B - 1 = 00101 = b_4 b_3 b_2 b_1 b_0$. The sequence of computation for 01010×00101 is then as follows:

$\begin{array}{r} 01010 \\ +01010 \\ \hline 10100 \\ + \\ \hline 00100 \\ +10000 \\ \hline 10100 \\ + \\ \hline 00100 \\ +01001 \\ \hline 01101 \\ + \\ \hline 01110 \\ +10000 \\ \hline 11110 \\ + \\ \hline 01110 \\ +10000 \\ \hline 11110 \\ + \\ \hline 01110 \end{array}$	$C_0 = A - 1 = 01010$ $b_0 2^0 A - 1 = 01010, \quad 2A - 1 = 00100$ $C_1 = C_0 \oplus (b_0 2^0 A - 1)$ $b_1 2A - 1 = -1, \quad 2^2 A - 1 = 01001$ $C_2 = C_1 - 1 + 1 = C_1$ $b_2 2^2 \cdot A - 1 = 01001, \quad 2^3 A - 1 = 00010$ $C_3 = C_2 \oplus (b_2 2^2 A - 1)$ $b_3 2^3 A - 1 = -1, \quad 2^4 A - 1 = 00101$ $C_4 = C_3 - 1 + 1 = C_3$ $b_4 2^4 \cdot A - 1 = -1, \quad 2^5 A - 1 = 01011$ $C_5 = C_4 - 1 + 1 = C_4$
---	---

Thus, $C = C_5 = 01110$ is the desired result of 01010 times 00101 , modulo $2^4 + 1$ in diminished-1 notation.

In Example 1, one observes that no operation is needed for $b_i = 0$, that is, $C_{i+1} = C_i$ for $b_i = 0$. This example can be simplified as follows:

$\begin{array}{r} 01010 \\ +01010 \\ \hline 10100 \\ + \\ \hline 00100 \\ +01001 \\ \hline 01101 \\ + \\ \hline 01110 \end{array}$	$C_0 = A - 1 = 01010$ $b_0 2^0 A - 1 = 01010$ $C_1 = C_0 \oplus (b_0 2^0 A - 1) = C_2$ $b_2 2^2 A - 1$ $C_3 = C_2 \oplus (b_2 2^2 A - 1) = C_4 = C_5 = C$
--	---

Example 1 shows that diminished additions require the addition of the complement of an end-around carry to its sum. It was shown (Ref. 6) that a considerable speed improvement can be obtained by performing this operation simultaneously with the summation. A modified algorithm with this simultaneous addition is given for the previous example as follows:

Example 2:

$\begin{array}{r} 11010 \\ 01010 \\ + \\ \hline 10100 \\ 01001 \\ + \\ \hline 01101 \\ 00000 \\ + \\ \hline 01110 \end{array}$	$\bar{C}_0 = 10000 + (A - 1)$ $b_0 2^0 A - 1$ $\bar{C}_1 = \bar{C}_0 \oplus (b_0 2^0 A - 1)$ $b_2 2^2 A - 1 = 01001$ $\bar{C}_3 = C_2 + (b_2 2^2 A - 1), C_2 = C_1 = \bar{C}_1 + 1$ $C_3 = \bar{C}_3 + 1 = C_4 = C_5 = C$
--	--

III. A VLSI Structure for Implementing an Integer Multiplication Modulo F_n

In Fig. 1, A, B, C, and D are 5-bit, 6-bit, 5-bit, and 6-bit registers, respectively. Initially registers A, B, C, and D contain the multiplicand $A - 1$, the multiplier $B - 1$, $2^4 + (B - 1)$, and $2^5 - 1$, respectively. The content in register D is used to add 00000 into $\bar{C}_5 = C_4 + (b_4 \cdot 2^4 \cdot A - 1)$. That is, $C_5 = \bar{C}_5 + 1$. The content in register B is used to control whether $C_{i+1} = C_i$ for $b_i = 0$ or $b_i 2^i + 1$ is added into register C for $b_i = 1$. At the very same moment $\bar{C}_{i+1} = (\bar{C}_i + 1) + (b_i 2^i A - 1) = C_i + (b_i 2^i A - 1)$ is computed and loaded into the register C for

$0 \leq i \leq 4$. The diminished-1 multiplication of $(A - 1)$ by 2 is performed by a left cyclic shift of the four least significant bits of register A with the A_3 bit circulated into the first significant bit and complemented. Also at the same time registers B and D are shifted right by one bit. These operations are continued repetitively until the MSB of registers B and D are shifted out. The desired final result 0 1 1 1 0 is obtained in register C after six iterations.

The layout of the structure in Fig. 2 has been completed by the use of the CAESER design tool (Ref. 7). The final layout of the multiplication chip is shown in Fig. 2. Both logic and circuit level simulations were performed using the logic simulator "Esim" (Ref. 8) and circuit simulator "Spice" (Ref. 9). A timing analysis was done using the timing simulator "Crystal" (Ref. 10). The VLSI chip is being fabricated. The operating frequency is estimated at around 5 MHz with 3 μ m CMOS technology. The total number of transistors in this chip is about 480. The chip for multiplication modulo F_2 in Ref. 6 requires 1000 transistors. Thus, this new multiplication algorithm requires only 50% of the transistors than the one in Ref. 6. The area of this chip with pads is estimated to be about 0.28 cm \times 0.28 cm (110 mil \times 110 mil).

IV. The Complex Modulo F_n Multiplier

Let $a + ib$ and $c + id$ be two complex numbers where $a, b, c,$ and d are integers and $i^2 = -1$. The general complex multiplication of $(a + ib)$ and $(c + id)$ is $(a + ib) \cdot (c + id) = (ac - bd) + i(bc + ad)$ which needs four integer multiplications and two integer additions. An algorithm which can perform a complex multiplication by only two integer multiplications is introduced in Refs. 11 and 12. A special case of this algorithm for Fermat number F_n is introduced in Ref. 1. It is shown in Ref. 1 that $Z_{F_n}(i)$ is isomorphic to the direct sum of two copies of Z_{F_n} , i.e., $S_{F_n} = Z_{F_n} + Z_{F_n}$, integers modulo F_n . In Ref. 1, let s be the solution of $x^2 \equiv -1 \pmod{F_n}$. For $a + ib \in Z_{F_n}(i)$, the following mapping,

$$f : a + ib \rightarrow ((a + sb)_{F_n}, (a - sb)_{F_n}) = (\alpha, \bar{\alpha}) \quad (6)$$

is an isomorphism of $Z_{F_n}(i)$ onto S_{F_n} where addition and multiplication in S_{F_n} are defined by

$$(\alpha, \bar{\alpha}) + (\beta, \bar{\beta}) = (\alpha + \beta, \bar{\alpha} + \bar{\beta}) \quad (7a)$$

and

$$(\alpha, \bar{\alpha}) \cdot (\beta, \bar{\beta}) = (\alpha \cdot \beta, \bar{\alpha} \cdot \bar{\beta}) \quad (7b)$$

In the set of F_n , s can be found as $s = \pm 2^{2^{n-1}}$. Thus the forward mapping from $a + ib$ to $(\alpha, \bar{\alpha})$ requires cyclic shifts

and additions only. The inverse mapping of f, f^{-1} , is also simple. From Ref. 1,

$$a \equiv -2^{2^{n-1}} (\alpha + \bar{\alpha}) \pmod{F_n} \quad (8a)$$

and

$$b \equiv -2^{2^{n-1}-1} (\alpha + \bar{\alpha}) \pmod{F_n} \quad (8b)$$

Note that in Ref. 2 the negative number is the complement of the 2^n least significant bits of its positive counterpart. Hence the inverse mapping requires cyclic shifts, complements, and additions only. This complex integer multiplier algorithm is illustrated in the following example.

Example 3: Compute $(0 0 1 1 0 + i 0 0 0 1 0) \cdot (0 0 1 0 0 + i 0 0 0 1 1)$ modulo $F_2 = 2^4 + 1$.

In residue rings of $F_2 = 17$, one obtains $s = \pm 2^2$. By Eq. (6), the forward mapping of $(0 0 1 1 0 + i 0 0 0 1 0)$ is

$$(0 0 1 1 0 + i 0 0 0 1 0) \rightarrow ((0 0 1 1 0 + 2^2 (0 0 0 1 0)),$$

$$(0 0 1 1 0 - 2^2 (0 0 0 1 0))$$

$$= ((0 0 1 1 0 + 0 1 0 1 1), (0 0 1 1 0 - 0 1 0 1 1))$$

$$= (0 0 0 0 1, (0 0 1 1 0 + 0 0 1 0 0))$$

$$= (0 0 0 0 1, 0 1 0 1 1)$$

Similarly

$$(0 0 1 0 0 + i 0 0 0 1 1) \rightarrow (0 0 0 1 1, 0 0 1 0 1)$$

Thus, the multiplication over Z_{F_2} is

$$(0 0 1 1 0 + i 0 0 0 1 0) \cdot (0 0 1 0 0 + i 0 0 0 1 1)$$

$$= (0 0 0 0 1, 0 1 0 1 1) \cdot (0 0 0 1 1, 0 0 1 0 1)$$

$$= (0 0 0 0 1 \cdot 0 0 0 1 1, 0 1 0 1 1 \cdot 0 0 1 0 1)$$

Using the integer multiplier modulo F_2 developed in the previous section, one obtains $(0 0 1 1 0 + i 0 0 0 1 0) \cdot (0 0 1 0 0 + i 0 0 0 1 1) = (0 0 1 1 1, 0 0 0 1 1)$. By Eqs. (8a) and (8b), the inverse mapping of $(0 0 1 1 1, 0 0 0 1 1)$ are

$$e = -2^3 (0 0 1 1 1 + 0 0 0 1 1) = -2^3 (0 1 0 1 1)$$

$$= -0 1 0 1 0 = 0 0 1 0 1$$

and

$$\begin{aligned} f &= -2(00111 - 00011) = -2(00111 + 01100) \\ &= -2(00011) = -00111 = 01000 \end{aligned}$$

Thus, $00101 + i01000$ is the desired result of $(00110 + i00010) \cdot (00100 + i00011)$ modulo $2^4 + 1$ in diminished-1 notation.

V. A VLSI Structure for Implementing a Complex Integer Multiplication Modulo F_n

In most digital signal processing applications, the multipliers are usually known. Thus, this multiplier can be precomputed to be $(\beta, \bar{\beta})$. Figure 3 shows the architecture for implementing Example 3. The same structure clearly extends to more general multiply algorithms over Z_{F_n} .

The layout of this multiplication chip is shown in Fig. 4. The logic, circuit, and timing simulations are performed. The chip of a complex integer multiplication circuit for $F_2 = 17$ is being fabricated. The operating frequency is around 5 MHz with $3 \mu\text{m}$ CMOS technology. The total number of transistors

in this chip is about 2300. The area of the chip with pads is estimated to be about $0.41 \text{ cm} \times 0.41 \text{ cm}$ ($162 \text{ mil} \times 162 \text{ mil}$).

VI. Conclusion

A new Fermat number integer multiplier is described in this article. This new Fermat integer multiplier does not need the initialization procedure as the one developed previously. Both the area and operating speed of the chip are greatly enhanced with this modification. In Ref. 1, a quadratic-polynomial Fermat number system was used to compute complex integer multiplications. The advantage of such a system is that a complex integer multiplication requires only two integer multiplications while the conventional complex integer multiplier needs four integer multipliers.

In this article, the new Fermat number integer multiplier is used as an integer multiplier in the quadratic-polynomial Fermat number system to compute complex number multiplications. A VLSI architecture for this new complex integer multiplier is developed and it is demonstrated in this article that this new complex integer multiplier, which uses the Fermat number F_2 , can be implemented on a single VLSI chip. Such a chip would be designed and fabricated in CMOS-pw technology.

References

1. Truong, T. K., Chang, J. J., Hsu, I. S., Pei, D. Y., and Reed, I. S., "Techniques for Computing the DFT Using the Residue Fermat Number System and VLSI," *GOMAC-85 Advance Program*, Orlando, FL, Nov. 5-7, 1985.
2. Leibowitz, L. M., "A Simplified Binary Arithmetic for the Fermat Number," *IEEE Trans. Acoustic, Speech and Signal Processing*, Vol. ASSP-24, No. 5, pp. 356-359, Oct. 1976.
3. Rader, C. M., "Discrete Convolution via Mersenne Transforms," *IEEE Trans. Comput.*, Vol. C-21, No. 12, pp. 1269-1273, Dec. 1972.
4. Agarwal, R. C., and Burns, C. S., "Fast Convolution Using Fermat Number Transform with Application to Digital Filtering," *IEEE Trans. Acoustic, Speech and Signal Processing*, Vol. ASSP-22, No. 2, pp. 89-97, Apr 1974.
5. Reed, I. S., Truong, T. K., and Welch, L. R., "The Fast Decoding of Reed-Solomon Code Using Fermat Number Transforms," *IEEE Trans. Information Theory*, Vol. IT-24, No. 4, pp. 497-499, July 1978.
6. Chang, J. J., Truong, T. K., Reed, I. S., and Hsu, I. S., "VLSI Architecture for the Multiplication of Integers Modulo a Fermat Number," *IEEE Trans. on Acoustic Speech, and Signal Processing*, Vol. ASSP-35, No. 6, pp. 1599-1602, Dec. 1985.
7. Ousterhout, J., *Editing VLSI Circuits with Caesar*, Computer Science Division, Electrical Engineering and Computer Sciences, University of California, Berkeley, Apr. 21, 1982.
8. Tevman, C., *Esim-Event Driven Switch Level Simulator*, Dept. of Electrical Engineering, MIT, 1977.
9. Negal, L. W., and Pederson, D. O., *SPICE-Simulation Program with Integrated Circuit Emphasis*, Memorandum No. ERL-M382, Electronics Research Laboratory, 1976.
10. Ousterhout, J., *Using Crystal for Timing Analysis*, Computer Science Division, Electrical Engineering and Computer Sciences, University of California, Berkeley, Mar. 1983.
11. Despain, A. M., Peterson, A. M., Rothaus, O. S., and Wold, E., "Fast Fourier Transform Processors Using Complex Residue Arithmetic," to appear in *The Journal of Parallel and Distributed Processing*.
12. Cozzens, J. H., and Finkelstein, L. A., "Computing the Discrete Fourier Transform Using Residue Number Systems in a Ring of Algebraic Integers," *IEEE Trans. on Information Theory*, Vol. IT-31, No. 5, pp. 580-588, Sept. 1985.
13. Shyu, H. C., Truong, T. K., and Reed, I. S., "A Complex Integer Multiplier Using the Quadratic Polynomial Residue Number System with Numbers of Form $2^{2^n} + 1$," *International Workshop on Systolic Array*, University of Oxford, England, July 2-4, 1986.

Table 1. The correspondence among decimal numbers, their values in the normal binary representation, and in the diminished-1 representation

Decimal number	Normal binary representation	Diminished-1 representation
0	0 0 0 0 0	1
1	0 0 0 0 1	2
2	0 0 0 1 0	3
3	0 0 0 1 1	4
4	0 0 1 0 0	5
5	0 0 1 0 1	6
6	0 0 1 1 0	7
7	0 0 1 1 1	8
8	0 1 0 0 0	9 (-8)
9 (-8)	0 1 0 0 1	10 (-7)
10 (-7)	0 1 0 1 0	11 (-6)
11 (-6)	0 1 0 1 1	12 (-5)
12 (-5)	0 1 1 0 0	13 (-4)
13 (-4)	0 1 1 0 1	14 (-3)
14 (-3)	0 1 1 1 0	15 (-2)
15 (-2)	0 1 1 1 1	16 (-1)
16 (-1)	1 0 0 0 0	0

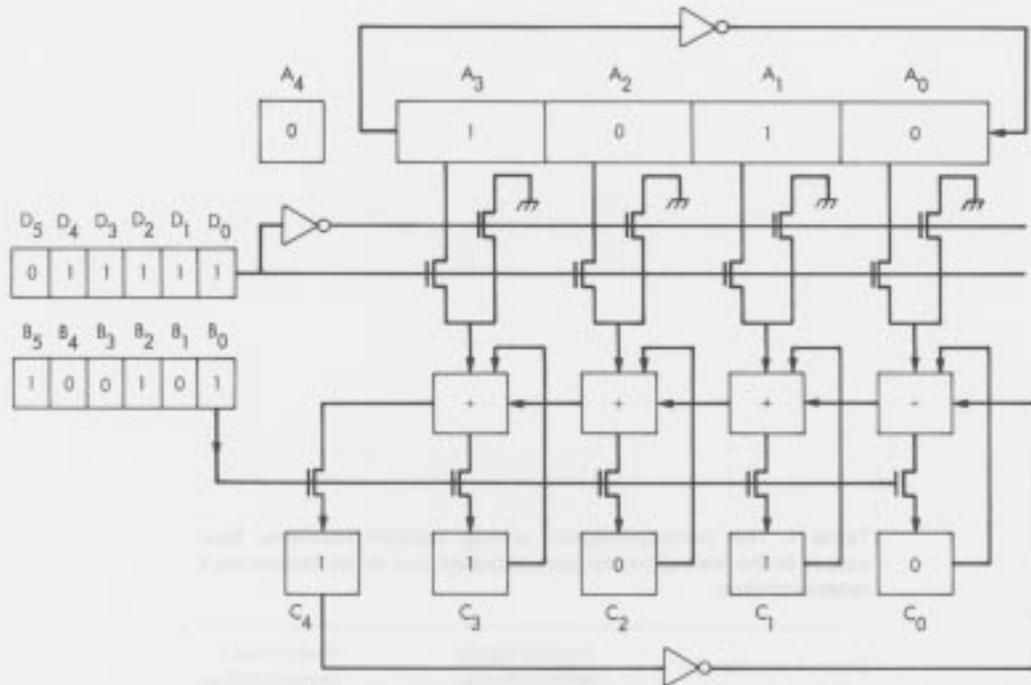


Fig. 1. The pipeline architecture for the implementation of multiplication modulo the Fermat number $2^4 + 1$ using diminished-1 number presentation

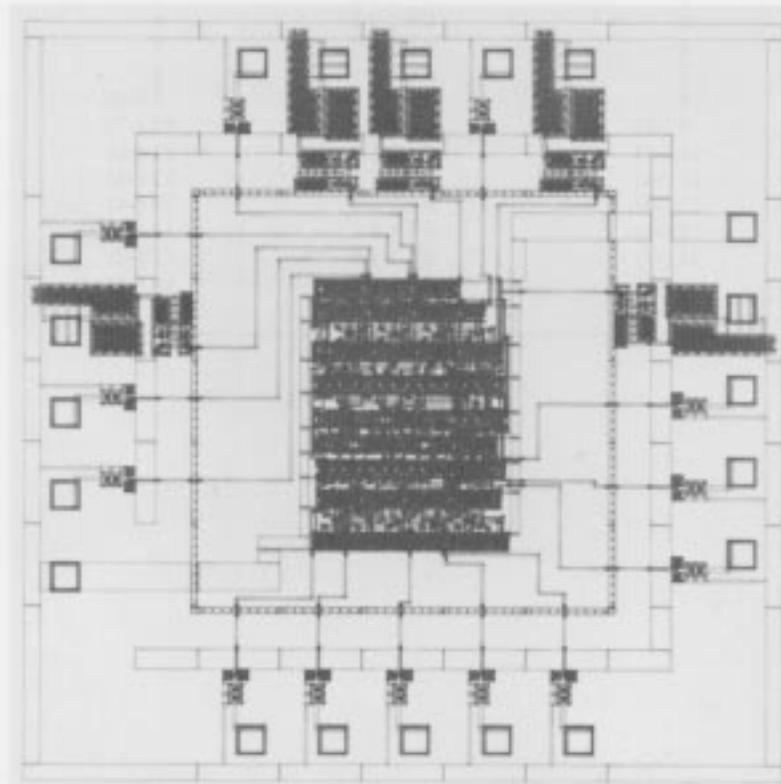


Fig. 2. VLSI layout of an integer multiplication circuit for $F_2 = 2^4 + 1$

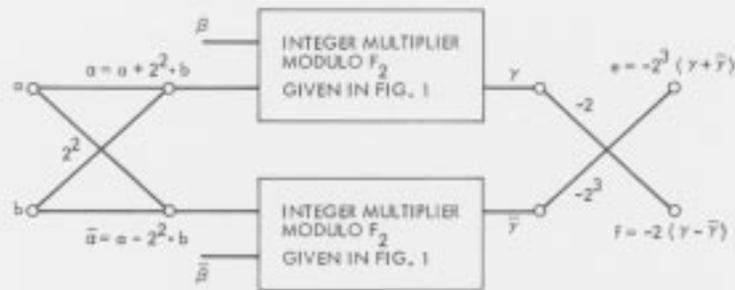


Fig. 3. The architecture for implementation of complex integer multiplication modulo $F_2 = 17$ using diminished-1 number representation

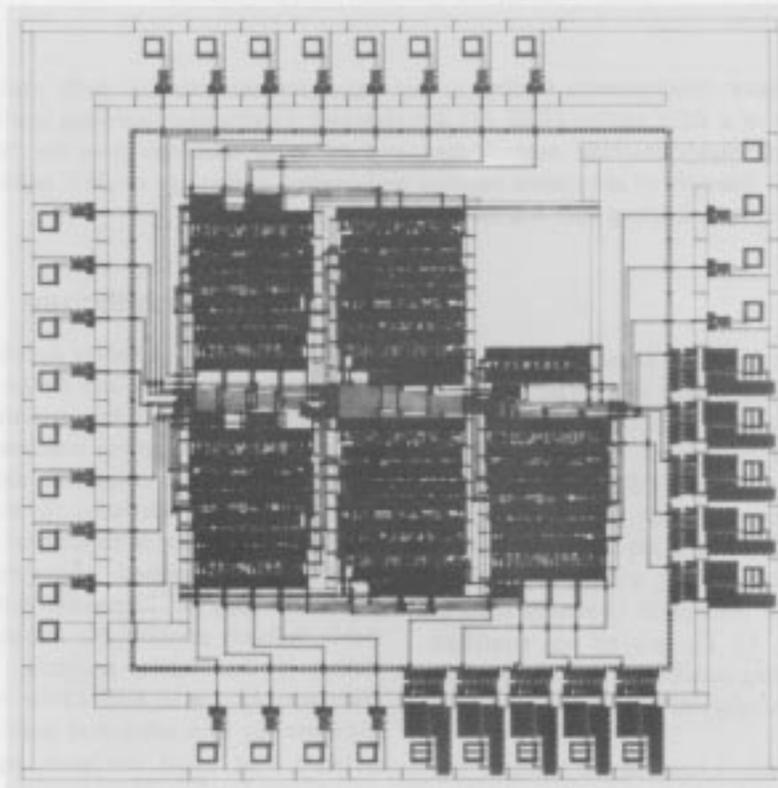


Fig. 4. VLSI layout of a complex integer multiplication circuit for $F_2 = 17$