tient. If m bits are used from the reciprocal the table requires 2^m entries and establishes m bits of quotient.

Since the table doubles in size for each additional bit of accuracy required in the initial quotient guess, small changes in the convergence rate per iteration may reflect substantial changes in the size of the starting table. Notice in almost all schemes the error is biased, hence it (or part of it) can be subtracted from the quotient, slightly reducing the average error. Referring back to Fig. 1, since we are approaching the root from the left side uniformly, we may predict ahead part of the distance for the next iteration. While this is attractive, the error bias may serve a useful function when left in the iterant. In certain cases it will serve to protect the integrity of integers (i.e., integer quotients will be preserved in their usual representation).

Conclusion

The problem of finding complexity or efficiency bounds for division is much more difficult than for add or multiply because of the multiplicity of approaches. The best known techniques require two basic arithmetic operations (add or multiply) to double the precision of the quotient. Even relatively small improvements in the convergence rate of a scheme can result in considerable hardware savings in the area of a starting table. The development of these techniques remains an open problem as does the application of non-Newtonian higher order iterations.

References

[1] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electronic Computers*, vol. EC-13, pp. 14–17, February 1964.
[2] S. F. Anderson, J. G. Earle, R. E. Goldschmidt, and D. M. Powers, "IBM System/360 Model 91: floating-point execution unit," *IBM J. Res. Develop.*, vol. 11, pp. 34–53, January 1967.
[3] R. E. Goldschmidt, "Applications of division by convergence," M.S. thesis, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Mass., June 1964.
[4] M. J. Flynn, "Very high-speed computing systems," *Proc. IEEE*, vol. 54, pp. 1901–1909, December 1966.
[5] H. H. Laughlin, "Large-number division by calculating machine," *Am. Math. Monthly*, vol. 37, pp. 287–293, 1930.
[6] D. E. Knuth, "Seminumerical algorithms," in *The Art of Computer Programming*, vol. 2. Reading, Mass.: Addison-Wesley, 1969, p. 215.
[7] M. Lehman, D. Senzig, and J. Lee, "Serial arithmetic techniques," *1965 Fall Joint Computer Conf., AFIPS Proc.*, vol. 27. Washington, D. C.: Spartan, 1965, pp. 715–725.
[8] E. V. Krishnamurthy, "On optimal iterative schemes for high-speed division," *IEEE Trans. Computers*, vol. C-19, p. 227–231, March 1970.
[9] J. F. Traub, *Iterative Methods for the Solution of Equations.* Englewood Cliffs, N. J.: Prentice-Hall, 1964.
[10] R. K. Richards, *Arithmetic Operations in Digital Computers.* New York: Van Nostrand Rheinhold, 1955.
[11] M. V. Wilkes, D. J. Wheeler, and S. Gill, *The Preparation of Programs for an Electronic Digital Computer.* Cambridge, Mass.: Addison-Wesley, 1951.
[12] R. J. Fieg, "Analysis of a computer divide algorithm," unpublished communication.
[13] P. Rabinowitz, "Multiple precision division," *Commun. ACM*, vol. 4, p. 98, February 1961.
[14] K. Kunz, *Numerical Analysis.* New York: McGraw-Hill, 1957.
[15] D. Ferrari, "A division method using a parallel multiplier," *IEEE Trans. Electronic Computers*, vol. EC-16, pp. 224–226, April 1967.

High-Speed Computer Multiplication Using a Multiple-Bit Decoding Algorithm

H. LING, MEMBER, IEEE

Abstract—This paper presents a method of performing the binary multiplication beyond the scheme of multiple ADD and SHIFT. The binary multiplication algorithm will be discussed first, followed by block decoding method, logic implementation, hardware consideration, and two examples which are at the end of the discussion.

Index Terms—Block decoding technique, fast multiplication, high-speed computer logic, high-speed multiplication, parallel multiplication.

Manuscript received May 20, 1969; revised December 12, 1969, and February 22, 1970.

The author is with the Information Sciences Department, IBM Research Laboratory, San Jose, Calif.

Introduction

One problem which the computer field has been concerned with for many years is how to improve the process of binary multiplication beyond the technique of repetitive ADD and SHIFT.

Some methods have been proposed, all of which have some disadvantages. It was pointed out by Lamdan and Aspinall [1], for example, that the realization of simultaneous multipliers necessitates a large number of components. Recently, carry save adders have generally been used to increase the speed of multiplication. However, due to the requirement of a large amount of hardware support [2], it is applicable only to a larger machine such as the 360/91.

The author [3] proposed a decomposition scheme to perform multiplication in 1966. This paper presents the detailed binary multiplication algorithm based upon a multiple-bit decoding technique. The sum (I) and difference (J) of the factors are assumed to have been computed in advance. A combinational logic module S is defined which has the transfer function

$$S(x) = x - \frac{x^2}{2},$$

if the set of n inputs is interpreted as a binary fraction x. If n is large enough to accommodate the entire length of I (or J), the product is obtained in three additions, as demonstrated by (20). If the number of bits exceeds n, then copies of S may be applied to segments of I and J in parallel and the results summed to form the complete product. In particular, if the total length of the product is $n \cdot 2^k$ bits, then a total of $1+2k$ additions are required.

Theory

Let A and B be two fractions whose product is being sought (we note that with proper scaling A, B can be integers or even general floating-point numbers).

Then

$$AB = 2[f(I) - f(J) - (I - J)/2] \tag{1}$$

with

$$\begin{aligned} I &= (A + B)/2 \\ J &= (A - B)/2 \\ f(x) &= (1/2)x(x + 1). \end{aligned}$$

It should be noted that the time required to square a number will be equal to half of that required in the general multiplication case due to $J = 0$. The purpose of this section is to decompose $f(I)$, $f(J)$ into efficiently manageable form. The following lemmas are presented:

Lemma 1: For $\delta = 0$ or 1, $f(\delta) = \delta$. (2)

Lemma 2: $f(x + y) = f(x) + f(y) + xy$. (3)

Lemma 3: $f(ny) = n^2 f(y) - y f(n - 1)$. (4)

From (4) $f(y)$ can be rewritten as

$$f(y) = (1/4)y + (1/4)f(2y). \tag{5}$$

Taking a binary fraction

$$x = \sum_{n=1}^{k} x_n 2^{-n},$$

where $x_n = 0$ or 1, applying (2), (3), and (5) systematically, we have

$$\begin{aligned} f(x) &= f(0.x_1 x_2 x_3 \cdots x_n) \\ &= \sum_{k=1}^{n} (1/4^k)(1 + 2x_k) \sum_{l=1}^{n-k+1} x_l + k - 1^{2^{-l}} \end{aligned} \tag{6}$$

where $1 > x \geq (1/2)$, $x_1 = 1$.

Equation (6) can be rewritten as follows:

$$\begin{aligned} f(x) &= D(x) + 2L(x) && (7) \\ &= 3D(x) - 2O(x). && (8) \end{aligned}$$

where[1]

$$D(x) = (1/2)x - B(x) \tag{9}$$

$$\begin{array}{rllllllll} L(x) = & 0.0 & 0 & (x_1x_1) & (x_1x_2) & (x_1x_3) & (x_1x_4) \cdots & (x_1x_n) & \\ & +0.0 & 0 & 0 & 0 & (x_2x_2) & (x_2x_3) \quad \cdots & (x_2x_n) & (10) \\ & +0.0 & 0 & 0 & 0 & & \cdots & (x_nx_n). & \end{array}$$

$$\begin{array}{rllllllll} O(x) = & 0.0 & 0 & (x'_1x_1) & (x'_1x_2) & (x'_1x_3) & (x'_1x_4) \cdots & (x'_1x_n) & \\ & +0.0 & 0 & 0 & 0 & (x'_2x_2) & (x'_2x_3) \quad \cdots & (x'_2x_n) & \\ & + \cdots + & & & & & & & (11) \\ & +0.0 & 0 & 0 & 0 & & \cdots + & (x'_{n-1}x_n). & \end{array}$$

$$B(x) = 0.0 \; 0x_1 0x_2 0x_3 0x_4 \cdots 0x_{n-1} 0x_n. \tag{12}$$

If both I and J are binary integers,

$$I = 2^n(0 \cdot i_1 i_2 i_3 \cdots i_n) = 2^n i \tag{13}$$

$$J = 2^m(0 \cdot j_1 j_2 j_3 \cdots j) = 2^m j, \tag{14}$$

the product AB can be rewritten as

$$AB = 2\{f[2^n i] - f[2^m j] - (I - J)/2\}. \tag{15}$$

By substituting (4) into (15), the product AB becomes

[1] Parenthesized terms represent bits expressed as logical AND functions, with x' representing logical complement of x.

$$AB = 2\left\{2^{2n}f(i) - (2^n - 1)\frac{I}{2} - 2^{2m}f(j) - (2^m - 1)\frac{J}{2} - \frac{(I - J)}{2}\right\}, \tag{16}$$

or simply

$$AB = 2\left\{2^{2n}f(i) - 2^{2m}f(j) - 2^n\left(\frac{I}{2}\right) + 2^m\left(\frac{J}{2}\right)\right\}. \tag{17}$$

By substituting (8) into (17), the product AB becomes

$$AB = 2\left\{2^{2n}[3D(i) - 2O(i)] - 2^{2m}[3D(j) - 2O(j)] - I\left(\frac{2^n}{2}\right) + J\left(\frac{2^m}{2}\right)\right\}. \tag{18}$$

Using (9) and regrouping AB, we finally obtain

$$AB = 2\{2^nI - 2^mJ - 2^{2n}[3B(i) + 2O(i)] + 2^{2m}[3B(j) + 2O(j)]\}. \tag{19}$$

Let $S(x) = 3B(x) + 2O(x)$; (19) can be rewritten as

$$AB = 2\{2^nI - 2^mJ - 2^{2n}[S(i)] + 2^{2m}[S(j)]\}. \tag{20}$$

If both A and B are n digit numbers, it is obvious that $S(i)$ will be $2n$ digits in length. The number of terms collected from $S(i)$ and $S(j)$ will affect the accuracy of the multiplication, and how to collect all the terms from $S(i)$ and $S(j)$ will depend on the choice of multiple-bit decoding rules.

Logic Implementation

As discussed above, n is an arbitrary integer. In order to consider the multiple-bit decoding technique, n has to be a reasonable number. Let $n=8$ (the reason will be discussed in a later section). The state table of S is generated by evaluating $S(x) = x - x^2/2$ for all 128 8-bit positive normalized binary fractions. If inputs are $0 \cdot i_1 i_2 i_3 \cdots i_8$ and outputs are $0 \cdot 0S_0S_1S_2 \cdots S_{15}$, the logical relations may be readily determined. For example,

$$S_0 = i_1, S_1 = i_1,$$
$$S'_2 = i_1i'_2i'_3 + i_1i'_2i'_4i'_5i'_7i'_8 + i_1i'_2i'_4i'_5i'_6i'_8 + i_1i'_2i'_4i'_5i'_6i'_7 + i_1i'_2i'_4i'_5i'_6i_8.$$

(+ means logic OR.) The completed decision is listed in Table I. Equation (20) and Table I show that the multiplication of two 8-bit numbers can be completed with three consecutive additions with some hardware support (the implementation of s). Any number larger than 8 bits, say 16 bits, can be decomposed into two 8-bit blocks; the procedure then follows in an obvious manner.

Hardware Consideration

The last phase of design is to implement the final solution as obtained by the decision maker as listed in Table I.

Fan-in and fan-out are always the basic problems facing switching circuit designers (especially when passive elements are used such as diodes). Large numbers of fan-in and

TABLE I
THE LOGIC IMPLEMENTATION OF S

S_0	S_1	S'_2	S'_3	S'_4
i_1	i_1	$i_1i_2i'_3$	$i_1i'_2i_3i_4$	$i_1i_2i'_3i'_4$
		$i_1i'_2i'_4i'_5i'_7i'_8$	$i_1i'_2i'_3i'_4i_8$	$i_1i'_2i_3i'_4i_5$
		$i_1i'_2i'_4i'_5i'_6i'_8$	$i_1i'_2i'_3i'_4i_5$	$i_1i'_3i'_4i'_6i'_7i'_8$
		$i_1i'_2i'_4i'_5i_6i'_7$	$i_1i'_2i'_4i_5i'_8$	$i_1i_2i'_3i'_5i'_6i'_8$
		$i_1i'_2i'_4i'_5i'_6i_8$	$i_1i'_2i_3i_5i_8$	$i_1i'_3i_4i_5i'_6i'_8$
			$i_1i'_2i'_4i'_5i_6i_7$	$i_1i'_2i'_3i'_5i_6i'_8$
			$i_1i_2i'_3i'_5i'_6i'_7$	$i_1i'_2i'_3i'_5i'_6i'_7i'_8$
				$i_1i'_3i'_4i'_5i_8$
				$i_1i'_2i'_3i_4i'_5i_7i_8$
				$i_1i'_2i'_3i'_5i_6i'_7i_8$
				$i_1i'_2i'_3i_4i_5i'_6i'_7$
				$i_1i'_2i_3i'_4i_6i_7$
				$i_1i'_2i_3i_4i'_5i'_6i_7$
				$i_1i_2i'_3i'_5i'_6i'_7i_8$

S'_5	S'_6	S'_7	S'_8
$i_1i'_3i'_4i'_5i'_7i'_8$	$i_1i'_4i'_5i'_6i'_7i'_8$	$i_1i'_4i'_5i'_6i'_7i'_8$	$i_1i'_5i'_6i'_7i'_8$
$i_1i'_2i_3i'_4i_5i'_6$	$i_1i_2i'_3i_4i'_5i_6$	$i_1i'_3i_4i_6i'_7i'_8$	$i_1i'_3i'_4i_5i_6i'_8$
$i_1i'_2i_4i'_5i_6i'_7$	$i_1i'_2i'_3i'_4i_7i'_8$	$i_1i_3i'_4i_6i'_7i'_8$	$i_1i'_3i_4i'_5i_6i'_8$
$i_1i'_2i_3i'_5i_6i_7$	$i_1i_3i'_4i'_5i_7i_8$	$i_1i'_2i_4i_6i_7i'_8$	$i_1i'_3i_4i_5i_7$
$i_1i'_2i'_3i_4i_5i_6$	$i_1i'_2i_3i_4i'_6i_7$	$i_1i_2i'_4i'_6i_7i_8$	$i_1i_3i'_4i'_5i_7i'_8$
$i_1i_2i'_3i_4i_6i'_8$	$i_1i'_3i'_4i_5i'_7i_8$	$i_1i_2i'_3i'_4i_7i'_8$	$i_1i_3i'_4i_5i_6i'_7i'_8$
$i_1i'_2i_4i'_5i'_6i_7$	$i_1i'_3i_4i'_5i_7i_8$	$i_1i'_2i'_3i'_4i_8$	$i_1i_3i_4i'_5i_6i'_7i'_8$
$i_1i'_2i'_3i'_4i'_6i_8$	$i_1i_2i'_3i'_4i'_6i_8$	$i_1i'_2i_4i'_5i'_6i_7i'_8$	$i_1i_3i_4i'_5i'_6i_7i'_8$
$i_1i'_3i_4i'_6i_7i_8$	$i_1i_2i_3i'_4i'_5i_8$	$i_1i'_2i_3i'_5i_6i_7i_8$	$i_1i'_2i_3i_6i_8$
$i_1i_2i'_3i'_4i'_5i_8$	$i_1i'_3i_4i_5i'_6i'_7i'_8$	$i_1i'_2i_3i_4i_5i_7i'_8$	$i_1i'_2i_3i_5i'_6i_8$
$i_1i_2i'_3i_4i_5i_8$	$i_1i_3i'_4i_5i'_6i'_7i'_8$	$i_1i_2i'_3i_5i'_6i_7i'_8$	$i_1i_2i'_3i'_5i'_6i_8$
$i_1i'_2i_3i_5i'_6i'_7i'_8$	$i_1i'_2i_3i'_4i_5i_6i'_8$	$i_1i_2i_3i'_4i_5i_7i'_8$	$i_1i'_2i'_3i'_4i_5i_6i_8$
$i_1i_2i'_3i_5i'_6i'_7i'_8$	$i_1i'_2i_3i_4i_6i'_7i'_8$	$i_1i'_2i'_3i'_5i_6i_7i_8$	$i_1i'_2i_4i'_5i_6i'_7i_8$
$i_1i'_2i'_3i_5i_6i'_7i'_8$	$i_1i_2i'_3i'_4i_6i'_7i_8$	$i_1i'_2i'_3i_4i_5i'_6i_8$	$i_1i'_2i_3i'_4i'_5i_7i_8$
$i_1i'_2i'_3i'_4i'_6i_7i'_8$	$i_1i_2i'_4i'_5i_6i'_7i'_8$	$i_1i'_2i_3i'_4i'_5i'_6i_8$	$i_1i_2i'_3i_4i_5i'_7i'_8$
$i_1i_2i'_3i'_4i'_5i_7i'_8$	$i_1i'_2i'_3i_4i'_5i_7i'_8$	$i_1i'_2i_3i_5i'_6i'_7i_8$	$i_1i_2i_3i'_4i'_5i'_7i_8$
$i_1i_2i'_3i_4i_5i_7i'_8$	$i_1i_2i'_3i'_4i'_6i_7i'_8$	$i_1i'_2i'_4i_5i_6i_7i_8$	$i_1i_3i'_4i_5i'_6i_7$
$i_1i'_2i_3i_4i'_5i_6i_8$	$i_1i'_2i'_3i'_4i'_5i'_7i_8$	$i_1i'_2i_3i'_5i'_6i_7i_8$	$i_1i_2i_3i_4i'_5i'_6i_8$
$i_1i_2i'_3i_4i'_5i_6i_8$	$i_1i'_2i'_3i_4i'_6i_7i_8$	$i_1i'_2i_3i_4i_6i'_7i_8$	
	$i_1i'_2i'_4i'_5i'_6i_7i_8$	$i_1i_2i'_4i'_5i'_6i'_7i_8$	
	$i_1i_3i'_4i'_5i_6i_7i_8$	$i_1i_2i'_3i_5i_6i'_7i_8$	
	$i_1i'_2i'_4i_5i_6i'_7i_8$	$i_1i_2i'_3i_5i'_6i'_7i_8$	
	$i_1i'_2i_3i_4i_5i'_6i_8$	$i_1i_2i'_3i_5i_6i_7i_8$	
	$i_1i_2i_3i'_4i'_6i'_7i_8$	$i_1i_2i'_3i_4i'_6i_7i_8$	
	$i_1i'_2i'_3i_4i_5i_6i'_7$	$i_1i_2i_3i'_4i'_6i_7i_8$	
		$i_1i_2i_3i'_4i_5i_6i_8$	

S'_9	S'_{10}	S'_{11}	S'_{12}
$i_1i'_5i'_6i'_7i'_8$	$i_1i'_6i'_7i'_8$	$i_1i'_6i'_7i'_8$	$i_1i'_7i'_8$
$i_1i'_4i_5i_7i'_8$	$i_1i'_5i_6i_7i'_8$	$i_1i'_5i_6i_8$	$i_1i'_6i_7i_8$
$i_1i_4i'_5i_7i'_8$	$i_1i_5i'_6i_7i'_8$	$i_1i_5i'_6i_8$	$i_1i_6i'_7$
$i_1i'_3i'_4i_5i'_6i_8$	$i_1i'_4i'_5i_6i_7$		
$i_1i'_3i_4i_5i_8$	$i_1i'_4i_5i'_6i_7$		
$i_1i_3i'_4i'_5i'_6i_8$	$i_1i'_4i_5i_6i_8$		
$i_1i_3i_4i'_5i'_7i_8$	$i_1i_4i'_5i'_6i_8$		
$i_1i'_3i'_4i_5i_6i_7i_8$	$i_1i_4i'_5i'_7i_8$		
$i_1i'_3i_4i'_5i'_6i_7i_8$	$i_1i_4i_5i'_6i'_7$		
$i_1i_3i'_4i_5i_6i'_7i_8$			
$i_1i_3i'_5i_6i_8$			

S'_{13}	S'_{14}	S'_{15}
$i_1i'_7i'_8$	$i_1i'_8$	$i_1i'_8$

fan-out not only deteriorate the input waveform, but also affect the circuit delay. Since the advent of integrated circuits, an active device is no more costly than a passive one. The waveform deterioration has been removed somewhat, but the overall rise time and delay time per stage still affects the choice of the maximum number of fan-in and fan-out elements. The author has obtained 25 ns per stage using 2N976 with 8 fan-in and 3 fan-out operating at current switching mode. Today, a 1-ns per stage integrated chip is available and 0.5 ns per stage is obtainable in laboratory scale. A further increase in the number of fan-in and fan-out elements is possible. Of course, the limitation of the state-of-the-art plays an important role in deciding the maximum number of fan-in and fan-out elements.

which is 37513 in decimal.

Example 2: Let $A=59881$, $B=41377$. These numbers are contained in registers A and B; in binary, they show as follows:

$$\text{Register } A = 1110100111101001$$
$$\text{Register } B = 1010000110100001.$$

In order to hold the product of 16 bits by 16 bits, the length of the accumulator should be 32 bits. After completing step 2 (three consecutive additions) the accumulator holds the product of A_{1-8} B_{1-8} and A_{9-16} B_{9-16}. The register A and register B hold the product of A_{1-8} B_{9-16} and A_{9-16} B_{1-16}. These now show as

$$\text{Accumulator} \quad \overbrace{1\,0\,0\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,1}^{\longleftarrow A_{1-8}\ B_{1-8} \longrightarrow}\ \overbrace{1\,0\,0\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,1}^{\longleftarrow A_{9-16}\ B_{9-16} \longrightarrow}$$

$$\text{Register } A \quad \overbrace{1\,0\,0\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,1}^{\longleftarrow A_{1-8}\ B_{9-16} \longrightarrow}$$

$$\text{Register } B \quad \overbrace{1\,0\,0\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,1}^{\longleftarrow A_{9-16}\ B_{1-8} \longrightarrow}.$$

The product of 16 bits by 16 bits can now be obtained with a total of five additions. The product is

$$1\,0\,0\,1\,0\,0\,1\,1\,1\,0\,1\,0\,1\,1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1$$

which is 2477696137 in decimal.

Description

In order to explain the operating procedure step by step, an example is given.

Example 1: Let $A=233$, $B=161$. In binary, these numbers are shown as

$$A = 11101001$$
$$B = 10100001.$$

From (1), (13), and (14), I and J assume the following values:

$$I = 2^8(0.11000101)$$
$$J = 2^6(0.1001).$$

Step 1: Substituting I and J into Table I, $S_0, S_1, S_2, \cdots, S_{15}$ are logically formed. $S(i)$ and $S(j)$ are shown as follows:

$$S(i) = 0.0111100100110011\,1$$
$$S(j) = 0.01100111100000000.$$

Step 2: Complete the multiplication (8 bits by 8 bits) with three consecutive additions:

$$\begin{aligned} AB &= 2[1100010100000000 \\ &\quad - 100100000000 \\ &\quad - 111100100110011.1 \\ &\quad + 11001111000.0] \\ &= 1001001010001001 \end{aligned}$$

Conclusion

Using this algorithm to perform the multiplication requires minimum circuit delay (only one shift operation in forming I and J). No arithmetic operation is needed to obtain the bit pattern of $S(i)$ and $S(j)$. The logic equations (all the S's listed in Table I) are not in the most simple form because the existence of redundant elements in S's may reduce the total number of required chips. Factoring out the term $i_1 i_2'$, common to some S's, will eliminate the number of fan-ins, but one additional level is created.

The use of this method to perform multiplication for any 8-bit machine requires three additions, a 16-bit machine requires five additions, and a 32-bit machine requires seven additions.

Acknowledgment

The author wishes to thank Dr. T. C. Chen for his continued helpful criticism and R. Shively for his revisions and suggestions.

References

[1] T. Lamdan and D. Aspinall, "Some aspects of the design of a simultaneous multiplier for a parallel binary digital computer," *1965 Proc. IFIP Cong.*, vol. 2. Washington, D. C.: Spartan, 1966, pp. 440–446.
[2] S. F. Anderson, J. G. Earle, R. E. Goldschmidt, and D. M. Powers, "Floating-point execution unit," *IBM J. Res. Develop.*, vol. 11, no. 1, pp. 35–53, 1967.
[3] H. Ling, "A short note on binary multiplication," IBM Res. Note NC 626, May 1966.