the corresponding minimum-row *LA* table. For example, in Fig. 10(d) table T_{min} obtained by reducing table T has fewer rows than the irreducible table L.

The extra minimization we may achieve in performing the *LA* to *PA* table transformation is useful when the universal STT state assignment method mentioned in Section III and described in [12] is used. To show this, we apply this method to tables T_{min} and T_r, the first obtained by an *LA* to *PA* table transformation and the second by an *LA* to *PF* table transformation.

Note that table T_{min} has fewer end-states than table T_r for two reaons: T_{min} has fewer internal states than T_r; all the internal states of T_r must be end-states, whereas at least one start-state must be present in T_{min}. By applying the mentioned state assignment method, which leads to a number of internal variables depending on the number of the end-states, six internal variables are needed for the circuit corresponding to T_r and two internal variables for the circuit corresponding to T_{min}. Thus, the number of internal variables can be considerably reduced by means of the extra minimization.

VII. Conclusions

In this note we have studied the general properties and the synthesis of a class of asynchronous circuits called *PA* circuits. It has been shown that *PA* circuits are a general type of sequential circuits, since for every level-input asynchronous circuit (*LA* circuit) and for every synchronous circuit of the Moore or Mealy type (*PS* circuit) there exists an equivalent *PA* circuit. Methods for *PS*, or *LA*, to *PA* circuit transformation have been described. Specific methods for state reduction and state assignment are described in other papers [11], [12].

Since *PA* circuits are operating in fundamental mode, the maximum length of input pulses need not be controlled. In realizing sequential machines, *PA* circuits are therefore preferred to *PS* circuits whenever it is difficult to control the maximum duration of input pulses. Further, *PA* circuits may be preferred to *LA* circuits in realizing machines having like successive inputs, as any two successive inputs can easily be distinguished by a clock-pulse input.

A way to realize any normal mode *LA* circuit by an equivalent *PA* circuit has been discussed. It has been shown that the minimum-state *PA* circuit has a number of internal states equal to, or less than, the equivalent minimum-state *LA* circuit.

The known STT state assignment methods for normal mode asynchronous circuits have an upper bound on the number of internal variables much greater than the number of variables resulting from applying the specific method for *PA* circuits mentioned in Section III and described in [12]. This may lead to reduction of the number of the internal variables of an *LA* circuit when it is realized by a *PA* circuit to which the specific STT state assignment method is applied. As shown in Section VI, the number of internal variables is further reduced if the minimum-state *PA* circuit has fewer internal states than the equivalent minimum-state *LA* circuit.

References

[1] E. F. Moore, "Gedanken-experiments on sequential machines," in *Automata Studies*, C. Shannon and J. McCarthy, Eds. Princeton, N. J.: Princeton Univ. Press, 1956.

[2] G. H. Mealy, "A method for synthesizing sequential circuits," *Bell Sys. Tech. J.*, vol. 34, Sept. 1955, pp. 1045–1079.

[3] D. A. Huffman, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, Mar. and Apr., pp. 161–190 and 275–303.

[4] E. J. McCluskey, *Introduction to the Theory of Switching Circuits*. New York: McGraw-Hill, 1965.

[5] G. B. Gerace, "Sequential circuit realizations with pulse input fundamental mode sequential circuits," *Calcolo*, vol. 3, Oct–Dec. 1966, pp. 493–539.

[6] M. C. Paull and S. H. Unger, "Minimizing the number of states in incompletely specified sequential switching functions," *IRE Trans. Electronic Computers*, vol. EC-8, Sept. 1959, pp. 356–367.

[7] A. Grasselli and F. Luccio, "A method for minimizing the number of internal states in incompletely specified sequential networks," *IEEE Trans. Electronic Computers*, vol. EC-14, June 1965, pp.350–359.

[8] D. A. Huffman, "A study on the memory requirements of sequential switching circuits," M.I.T. Res. Lab. of Electronics, Cambridge, Mass., Rept. 293, March 1955.

[9] C. N. Liu, "A state variable assignment method for asynchronous sequential switching circuits," *J. ACM*, vol. 10, Apr. 1963, pp. 209–216.

[10] J. H. Tracey, "Internal state assignment for asynchronous sequential machines," *IEEE Trans. Electronic Computers*, vol. EC-15, Aug. 1966, pp. 551–560.

[11] G. Frosini and G. B. Gerace, "A simplified method for minimizing the number of internal states in incompletely specified pulse input asynchronous sequential circuits," Istituto di Elaborazione dell'Informazione, Pisa, Italy, Internal Rept. B69/2, Jan. 1969.

[12] ——, "A universal STT state assignment method for pulse input asynchronous sequential circuits," submitted to *IEEE Trans. Computers*.

[13] W. J. Cadden, "Equivalent sequential circuits," *IRE Trans. Circuit Theory*, vol. CT-6, Mar. 1959, pp. 30–34.

[14] G. B. Gerace, "The synthesis of pulse input sequential circuits self-concurrently operating," in *Network and Switching Theory*, G. Biorci, Ed. New York: Academic Press, Inc., 1968, pp. 593–634.

[15] S. H. Unger, "Hazards and delays in asynchronous sequential switching circuits," *IRE Trans. Circuit Theory*, vol. CT-6, Mar. 1959, pp. 12–25.

[16] A. D. Friedman and P. R. Menon, "Synthesis of synchronous sequential circuits with multiple-input changes," *IEEE Trans. Computers*, vol. C-17, June 1968, pp. 559–566.

A 40-ns 17-Bit by 17-Bit Array Multiplier

STYLIANOS D. PEZARIS, MEMBER, IEEE

Abstract—A high-speed array multiplier generating the full 34-bit product of two 17-bit signed (2's complement) numbers in 40 ns is described. The multiplier uses a special 2-bit gated adder circuit with anticipated carry. Negative numbers are handled by considering their highest order bit as negative, all other bits as positive, and adding negative partial products directly through appropriate circuits. The propagation of sum and carry signals is such that sum delays do not significantly contribute to the overall multiplier delay.

Index Terms—Array multiplier, Dadda's multiplier, digital multiplier, fast multiplier, parallel multiplier, Wallace's multiplier.

I. Introduction

High-speed multipliers are useful for many digital filtering applications. To obtain highest possible speed, array multipliers, rather than conventional sequential add-and-

Manuscript received June 9, 1970. This work was sponsored by the Department of the Air Force.

The author is with the M.I.T. Lincoln Laboratory, Lexington, Mass. 02173.

shift multipliers, must be used. The Lincoln Laboratory array multiplier described in this paper was designed with emphasis on speed, with the restriction, however, that speed should not be pushed so far that state of the art problems are encountered. Thus, the circuits used are current-steering MECL-like with speed intermediate between the MECL II and MECL III lines. They are packaged in conventional dual-in-line 14-pin packages. Power dissipation per package is a reasonable 350 mW. Interconnections between packages are made through a conventional 4-layer printed circuit board.

In spite of these restrictions, the multiplier tested yields the full 34-bit product of two 17-bit positive or negative numbers in just 40 ns. This high performance was achieved primarily by the following means.

1) A special 2-bit gated adder circuit with 2-bit anticipated carry is used.

2) Negative (2's complement) numbers are handled by treating their highest order bit as negative, all other bits as positive, and adding negative partial products directly by two variants of the basic 2-bit gated adder circuit.

3) The propagation paths for sum and carry signals between adders are such that sum delays do not significantly contribute to the overall multiplier delay.

These three main characteristics of the design are discussed in the following sections.

II. The L101, A 2-Bit Gated Adder with Anticipated Carry

The L101 is a Lincoln Laboratory designed circuit that forms the heart of the multiplier. It is a 2-bit adder with gated inputs and 2-bit anticipated carry. The gated inputs are used for forming partial products. Carry delay is 1.6 ns. Sum delay is 2.8 ns. The circuitry used is ECL, similar to the Motorola MECL line.

The basic characteristics of the L101 are shown in Fig. 1. It consists of four EXCLUSIVE-OR gates and two carry circuits. The EXCLUSIVE-OR gates use two-level series gating (they are similar to the MECL MC1030 gates). The partial products, designated as A_0, B_0, A_1, and B_1 in the figure, are generated by duplicate transistors at the inputs of the corresponding EXCLUSIVE-OR gates. This simple way of generating partial products requires *negative* logic (logical ONE $= -1.6$ V, logical ZERO $= -0.8$ V); except for this input gating the adder would of course operate equally well with either negative or positive logic.

The two carry circuits generate the carry C_1 into the second stage, and the output carry C_2, according to the equations (C_0 is the input carry):

$$C_1 = C_0(A_0 \oplus B_0) + A_0B_0$$
$$C_2 = C_0(A_0 \oplus B_0)(A_1 \oplus B_1) + A_0B_0(A_1 \oplus B_1) + A_1B_1.$$

The signals $A_0 \oplus B_0$, $A_1 \oplus B_1$, A_0B_0, and A_1B_1 are generated by the two input EXCLUSIVE-OR gates. To evaluate C_1 and C_2 in terms of C_0 and the $A_0 \oplus B_0$, $A_1 \oplus B_1$, A_0B_0, A_1B_1 we must evaluate the above AND-OR functions. In general, evaluation of an AND-OR function requires two

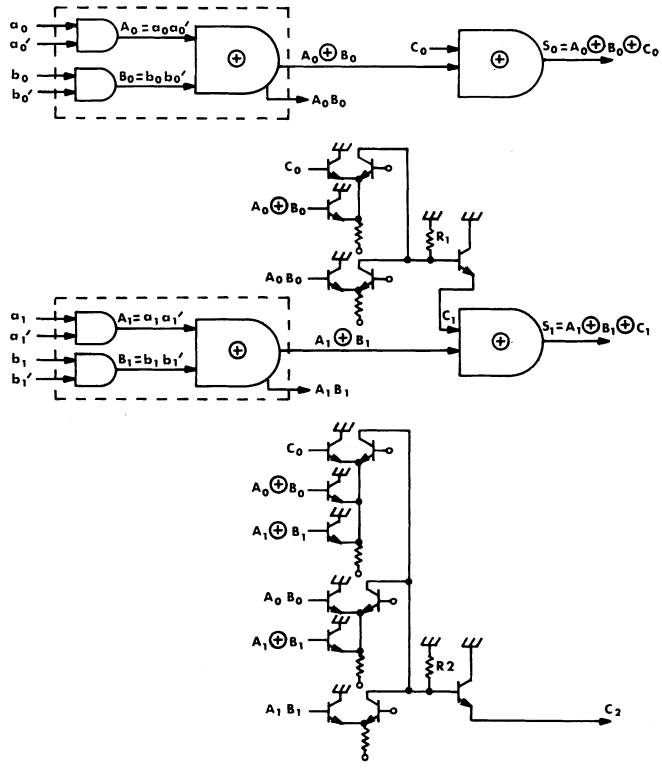

Fig. 1. Basic characteristics of the L101 2-bit adder.

gate delays. In this case, however, the terms to be ORed are mutually exclusive, and therefore the OR function can be evaluated by simply adding the output currents of the AND stages by a resistor (R_1, R_2 in Fig. 1), thus obtaining C_1 and C_2 with (almost) a *single* gate delay.

In addition to the L101, two variants of it, the L102 and L103, that add negative partial products are also used in the multiplier. With a view towards facilitating the discussion of these circuits in the next section, we introduce here the following notation.

L101 has five input variables A_1, B_1, A_0, B_0, C_0, and three output variables C_2, S_1, S_0. If we think of these variables as binary *arithmetic*, rather than binary *logical* variables, then, given any inputs A_1, B_1, A_0, B_0, C_0, the L101 produces outputs C_2, S_1, S_0 satisfying

$$A_1 + B_1 + A_0 + B_0 + C_0 = C_2 + S_1 + S_0.$$

In this equation, of course, A_1 can have any one of two allowable values, B_1 similarly, A_0 similarly, etc. To indicate that a binary arithmetic variable P has the possible values u and v, where, in general, $u \gtreqless 0$ and $v \gtreqless 0$, we use the notation

$$P = (u, v)$$

with the first value u corresponding to the logical ZERO state (-0.8V), and the second value v to the logical ONE state (-1.6V). A choice of values for the L101 variables is

$$A_1 = (0, 2),\ B_1 = (0, 2),\ A_0 = (0, 1),\ B_0 = (0, 1),$$
$$C_0 = (0, 1),\ C_2 = (0, 4),\ S_1 = (0, 2),\ S_0 = (0, 1).$$

This is clearly not the only possible choice. For example,

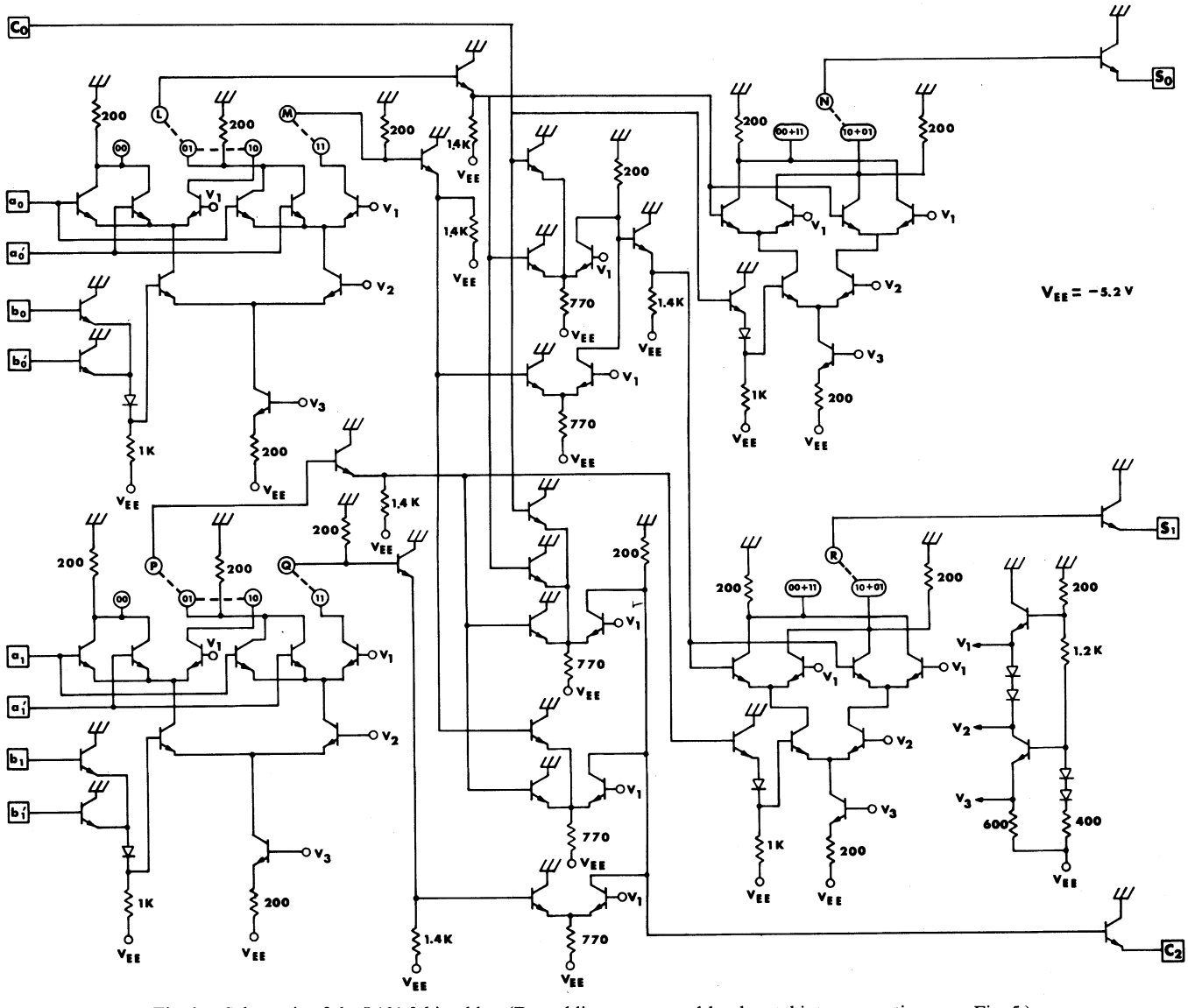

Fig. 2. Schematic of the L101 2-bit adder. (Dotted lines are second-level metal interconnections, see Fig. 5.)

since the circuit performs a linear arithmetic operation, multiplication of all these values by the same arbitrary constant would give us another possible choice. Multiplying by -1 we obtain the useful choice

$$(0, -2), (0, -2), (0, -1), (0, -1), (0, -1), (0, -4), (0, -2), (0, -1).$$

To indicate the arithmetic operation that the adder can perform, and a particular choice of values for its variables we use the notation

$$(0, 2) + (0, 2) + (0, 1) + (0, 1) + (0, 1) = (0, 4) + (0, 2) + (0, 1).$$

This notation proves to be very convenient for discussing adders such as the L102 and L103, whose input values are not all of the same sign.

The L101 circuit schematic is shown in Fig. 2. Typical output pull-down resistors used are 1K. The dotted lines connecting the points L to 01 and 10, M to 11, N to 10+01, P to 01 and 10, Q to 11, and R to 10+01 are second-level metal connections that make the circuit be an L101 rather than an L102 or L103. The connections required for the L102 and L103 are described in the next section.

Fig. 3 shows the circuit chip. The chip's size is 35-by 45-mil. Two levels of metallization are used. The masks for the circuit were designed at Lincoln Laboratory and the circuits were produced by the Microelectronics Division of Philco-Ford, Blue Bell, Pa.

III. Array Multiplication of 2's Complement Numbers

Given $N+1$ binary arithmetic variables $X_0, X_1, \cdots, X_N$, with $X_0, \cdots, X_{N-1}$ having the possible values (0, 1) and X_N the possible values (0, -1), any integer X within the range

$$-2^N \leqq X \leqq 2^N - 1$$

has a unique expansion

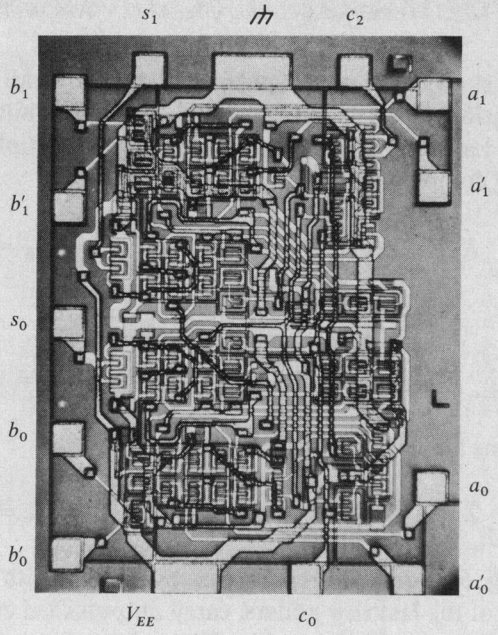

Fig. 3. The L101 2-bit adder chip.

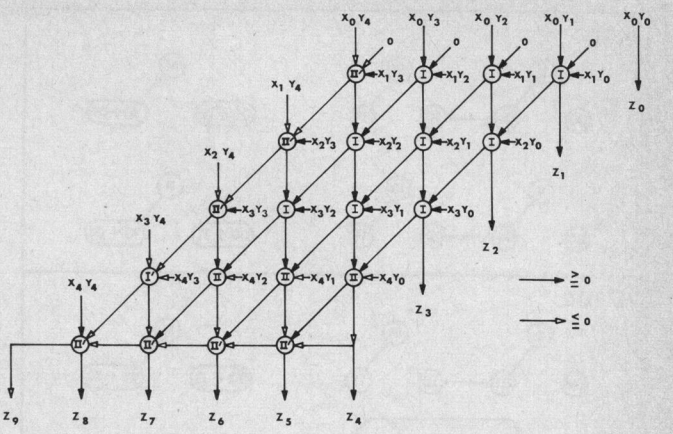

Fig. 4. A 5-bit array multiplier that multiplies signed (2's complement) numbers by adding negative partial products directly.

bers $\geqq 0$ are identified by black arrows, signals representing numbers $\leqq 0$ by hollow arrows. All $X_i Y_k$ are $\geqq 0$ except the X_0Y_4, X_1Y_4, X_2Y_4, X_3Y_4, X_4Y_0, X_4Y_1, X_4Y_2, X_4Y_3 that are $\leqq 0$. The figure makes evident that we need the following types of adders.

Type I (all positive inputs), performing	$(0, 1) + (0, 2) + (0, 1) = (0, 2) + (0, 1)$
Type II (one negative input), performing	$(0, -1) + (0, 1) + (0, 1) = (0, 2) + (0, -1)$
Type II′ (two negative inputs), performing	$(0, 1) + (0, -1) + (0, -1) = (0, -2) + (0, 1)$
Type I′ (all negative inputs), performing	$(0, -1) + (0, -1) + (0, -1) = (0, -2) + (0, -1)$

$$X = \sum_{n=0}^{N} X_n 2^n$$

in terms of the $X_0, \cdots, X_N$. This expansion is known as the 2's complement representation of X and numbers given by this representation are called 2's complement numbers.

The product of two 2's complement numbers X, Y

$$X = \sum_{n=0}^{N} X_n 2^n, \qquad X_0, X_1, \cdots, X_{N-1} = 0, 1, \qquad X_N = 0, -1$$

$$Y = \sum_{n=0}^{N} Y_n 2^n, \qquad Y_0, Y_1, \cdots, Y_{N-1} = 0, 1, \qquad Y_N = 0, -1$$

can of course be found by first converting them to positive numbers, then multiplying these positive numbers, and finally converting the product to a negative number if $X \cdot Y < 0$. This, however, is not the fastest method, especially so if the numbers X and Y are being calculated by adders, so that their highest order bits become available last. In the Lincoln multiplier the numbers X and Y are multiplied directly, by adding the partial products $X_i Y_k$ *with their signs* through appropriate circuits.

Fig. 4 illustrates this method by showing a 5-bit by 5-bit carry-save multiplier using single-bit adders. In the figure, adders are represented by circles. Signals representing numbers $\geqq 0$ are identified by black arrows, signals representing numbers $\leqq 0$ by hollow arrows. All $X_i Y_k$ are $\geqq 0$ except the X_0Y_4, X_1Y_4, X_2Y_4, X_3Y_4, X_4Y_0, X_4Y_1, X_4Y_2, X_4Y_3 that are $\leqq 0$. The figure makes evident that we need the following types of adders.

Type I (all positive inputs), performing	$(0, 1) + (0, 2) + (0, 1) = (0, 2) + (0, 1)$
Type II (one negative input), performing	$(0, -1) + (0, 1) + (0, 1) = (0, 2) + (0, -1)$
Type II′ (two negative inputs), performing	$(0, 1) + (0, -1) + (0, -1) = (0, -2) + (0, 1)$
Type I′ (all negative inputs), performing	$(0, -1) + (0, -1) + (0, -1) = (0, -2) + (0, -1)$

Types I and I′ correspond to identical truth tables (because if $x+y+z=u+v$, then $-x-y-z=-u-v$) and therefore to identical circuits. Similarly, Types II and II′ correspond to identical circuits. Thus, *two* types of circuits are required for this multiplier. (Note, however, that we might require more than one type of "Type II" circuits if the electrical characteristics of the three inputs are not identical. For example, if the last input is the fastest, we might require a $(0, -1) + (0, 1) + (0, 1)$ *and* a $(0, 1) + (0, 1) + (0, -1)$ circuit.)

For implementing this method with 2-bit adders *three* types of circuits (the L101, L102, L103) are required, as seen in the next section. The arithmetic operations performed by these three types are given below, in the notation introduced in Section II.

	A_1	B_1	A_0	B_0	C_0	C_2	S_1	S_0
L101:	$(0, 2) + (0,$	$2) + (0, 1) + (0,$		$1) + (0, 1) =$	$(0, 4) + (0,$		$2) + (0,$	$1)$
L102:	$(0, 2) + (0,$	$-2) + (0, 1) + (0,$		$1) + (0, 1) =$	$(0, 4) + (0,$	$-2) + (0,$		$1)$
L103:	$(0, 2) + (0,$	$-2) + (0, 1) + (0,$	$-1) + (0, 1) =$		$(0, 4) + (0,$	$-2) + (0,$		$-1)$

Thus, L102 evaluates

$$0 - 2 + 1 + 1 + 1 \quad \text{as} \quad 0 + 0 + 1,$$

$$2 + 0 + 0 + 0 + 0 \quad \text{as} \quad 4 - 2 + 0, \quad \text{etc.}$$

and L103 evaluates

$$2 - 2 + 1 - 1 + 1 \quad \text{as} \quad 4 - 2 - 1, \quad \text{etc.}$$

By writing out the truth table for the L102 we can verify that the logic performed by it is correctly expressed by the equations

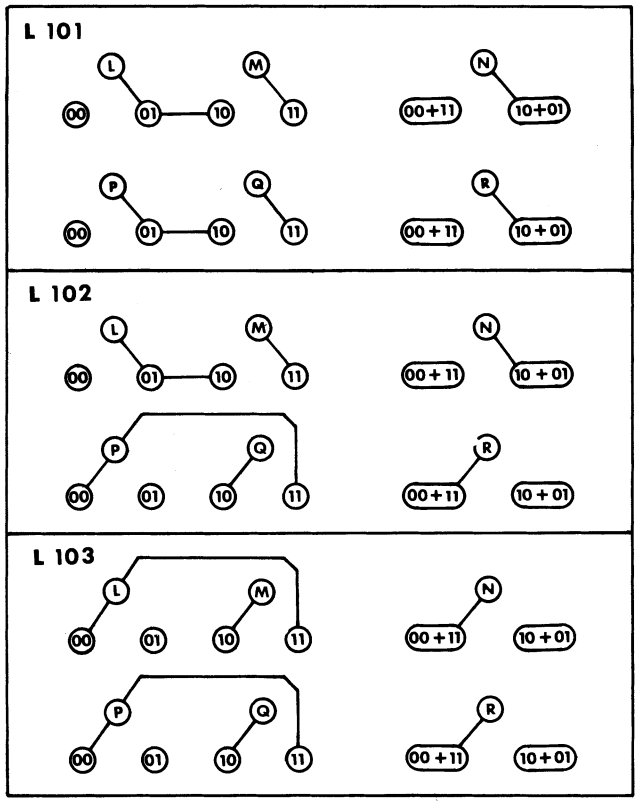

Fig. 5. Second-level metal interconnections that cause the basic 2-bit adder chip to become an L101, L102, or L103 adder.

$$S_0 = A_0 \oplus B_0 \oplus C_0$$
$$C_1 = C_0(A_0 \oplus B_0) + A_0B_0$$
$$S_1 = C_1(\bar{A}_1\bar{B}_1 + A_1B_1) + \bar{C}_1(\bar{A}_1B_1 + A_1\bar{B}_1)$$
$$C_2 = C_0(A_0 \oplus B_0)(\bar{A}_1\bar{B}_1 + A_1B_1) + A_0B_0(\bar{A}_1\bar{B}_1 + A_1B_1) + A_1\bar{B}_1$$

and for the L103 by the equations

$$S_0 = C_0(\bar{A}_0\bar{B}_0 + A_0B_0) + \bar{C}_0(\bar{A}_0B_0 + A_0\bar{B}_0)$$
$$C_1 = C_0(\bar{A}_0\bar{B}_0 + A_0B_0) + A_0\bar{B}_0$$
$$S_1 = C_1(\bar{A}_1\bar{B}_1 + A_1B_1) + \bar{C}_1(\bar{A}_1B_1 + A_1\bar{B}_1)$$
$$C_2 = C_0(\bar{A}_0\bar{B}_0 + A_0B_0)(\bar{A}_1\bar{B}_1 + A_1B_1) + A_0\bar{B}_0(\bar{A}_1\bar{B}_1 + A_1B_1) + A_1\bar{B}_1.$$

These logic functions[1] can be evaluated with a basic L101 chip by modifying some second level metal interconnections that select outputs from the EXCLUSIVE-OR gates. The interconnections required are shown in Fig. 5.[2]

[1] The sum functions are identical for all three (L101, L102, L103) circuits; they are written above in different forms to indicate the way in which each circuit evaluates them.

[2] One might ask if the same variation of second-level metal technique can be used to produce, from an L101, adders such as

A_1 B_1 A_0 B_0 C_0 C_2 S_1 S_0

$$(0, 2) + (0, 2) + (0, 1) + (0, 1) + (0, -1) = (0, 4) + (0, 2) + (0, -1).$$

The answer to this question is no. If the input carry is of opposite sign to both other first stage inputs, or if the carry to the second stage is of opposite sign to both other second stage inputs, the scheme does not work. Fortunately, the design of the multiplier does not require such adders.

IV. The Overall Logical Organization of the 40-ns Multiplier

Fig. 6 shows a 9-bit multiplier of identical overall logical organization to the 17-bit by 17-bit multiplier built at the Lincoln Laboratory. The multiplier of Fig. 6 multiplies two 9-bit 2's complement numbers

$$X = \sum_{n=0}^{8} X_n 2^n, \qquad X_0, X_1, \cdots, X_7 = 0, 1, \qquad X_8 = 0, -1$$

and

$$Y = \sum_{n=0}^{8} Y_n 2^n, \qquad Y_0, Y_1, \cdots, Y_7 = 0, 1, \qquad Y_8 = 0, -1$$

and forms an 18-bit 2's complement product

$$Z = \sum_{n=0}^{17} Z_n 2^n, \qquad Z_0, Z_1, \cdots, Z_{16} = 0, 1, \qquad Z_{17} = 0, -1.$$

In Fig. 6, adders are represented by circles. With the exception of the last row adders, carry outputs feed carry inputs and propagate *diagonally*. Throughout the multiplier, sum outputs feed sum inputs and propagate *vertically*. In the last row of adders, carries originating within the row feed carry inputs and propagate horizontally, and carries from the previous row feed sum inputs and propagate diagonally.

The logical organization is basically of the carry-save type,[3] modified by a "sum-skip" arrangement to speed-up vertical propagation of signals. Sum signals jump after every three adders (every four adders in the 17-bit multiplier). The longest path using as many carries as possible is the path along the diagonal from X_1Y_0 to Z_9 and then along the horizontal from Z_9 to Z_{16}. The longest path using as many sums as possible is the path along the vertical from X_0Y_8 to Z_8 and then along the horizontal from Z_8 to Z_{16}. In the 17-bit by 17-bit multiplier these paths are 4 sum + 12 carry delays long, or about 30 ns, and 9 sum + 6 carry delays long, or about 35 ns. Thus, these two extreme paths have reasonably equal delays.

In Fig. 6, signals representing arithmetic values $\geqq 0$ are identified by black arrows, signals representing arithmetic values $\leqq 0$ by hollow arrows. The numbers 1, 2, or 3, drawn next to carry lines joining two adders, identify these two adders as parts of an L101, L102, or L103, respectively. The figure also shows the basic arithmetic operations performed by each of these circuits, in the notation of Section II. As mentioned previously, each adder can also perform the operations resulting from these basic operations by changing the signs of all terms; for example the L101 and L103 adders in the lower left-hand corner of the multiplier are used in this fashion.

A point of interest is that Z_8 comes out of an adder as a negative bit; since we want Z_8 to be positive we need a circuit capable of performing the operation

$$(0, -1) = (0, 1) + (0, -2).$$

[3] A. Habibi and P. A. Wintz, "Fast multipliers," *IEEE Trans. Computers* (Short Notes), vol. C-19, Feb. 1970, pp. 153–157.

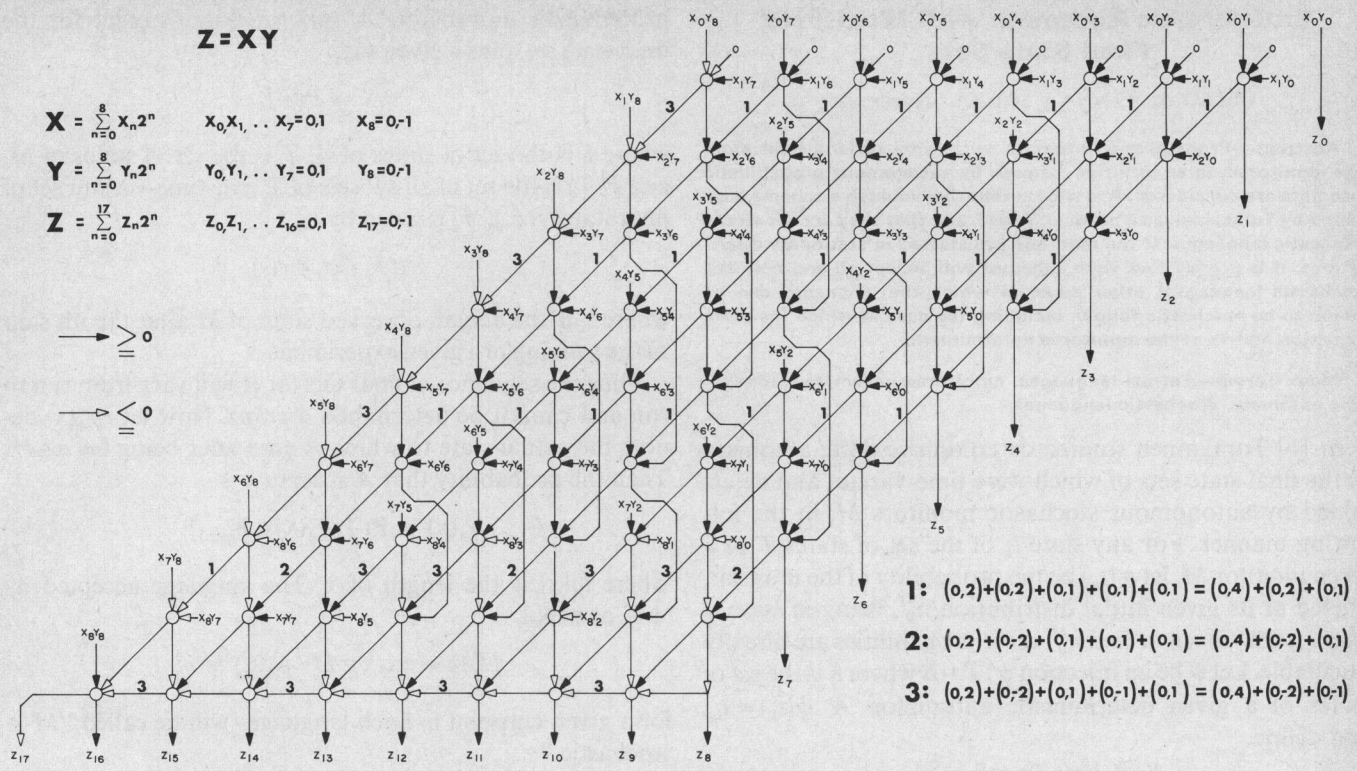

Fig. 6. A 9-bit by 9-bit multiplier of identical overall logical organization to the 17-bit by 17-bit Lincoln multiplier.

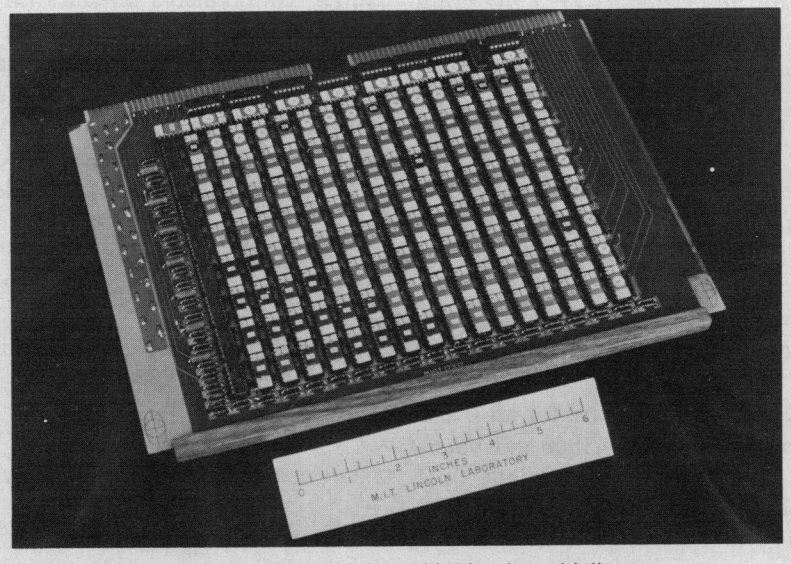

Fig. 7. The 17-bit by 17-bit Lincoln multiplier.

As easily seen, the circuit required is nothing more than a wired connection, feeding a $(0, -2)$ carry to the next higher bit position.

Fig. 7 is a photograph of the 17-bit by 17-bit multiplier. The card's size is 9- by 7-in. Four-layer interconnections are used (one ground plane, one voltage plane, and two signal planes).

V. Conclusion

The main design characteristics of the Lincoln Laboratory multiplier were discussed. The multiplier's speed, although rather high, can certainly be significantly improved by using state of the art methods in the fabrication of the circuits and in the interconnections between the circuits. Even further speed improvements are probably possible by extending the anticipated carry to 3 bits and providing more elaborate sum propagation paths.

Acknowledgment

The author would like to thank S. A. Idsik of Philco-Ford, Blue Bell, Pa., for his efforts, which were instrumental for the successful fabrication of the adder circuits.