ARITHMETIC OPERATIONS WITH THE HYDAC* DIGITAL OPERATIONS SYSTEM

ABSTRACT: This paper describes the use of the general purpose components of the Digital Operations System (DOS) of the EAI HYDAC Computer in the performance of basic digital arithmetic operations. The intent is to bring the attention of those who are unfamiliar with logic design, the manner in which the DOS components are combined to perform the binary arithmetic operations of addition, subtraction, and multiplication. Representative programs are presented which serve to indicate the manner in which logic building blocks are interconnected to perform specific operations, and also the approximate number ot components required. In the interest of clarity in emphasizing the concept of utilizing general purpose digital building blocks to perform arithmetic operations, the operation of these circuits are not explained fully, although their general features are described. References are suggested for those interested in further details concerning the fundamentals of these operations.

GENERAL

With a sufficient number of HYDAC logic building blocks available in the Digital Operations System (DOS), most of the arithmetic and logical functions of a digital computer can be programmed. Indeed, implementation of a complete stored program computer is possible. Such an undertaking would require very many components, however. It should be noted, though, that many useful arithmetic functions can be programmed with the DOS using only a few logic components and a minimum of patching.

The fundamentals involved in the basic digital arithmetic operations of binary addition, subtraction, and multiplication are presented here in sufficient detail for an understanding of the DOS programs presented. Division is somewhat more complex than multiplication and is not so frequently required. For this reason, and since the choice of a divider circuit depends greatly upon its use and the requirements of the problem, it has not been included in this study. The more complex operations of integration, polar-to-cartesion, cartesion-to-polar resolution, vector rotation, square root, logarithm, exponentiation, sine, cosine, and hyperbolic sine and cosine with the DOS will be described in subsequent papers.

For each of the operations discussed, a representative program is presented. No serious attempt has been made to minimize the number of components employed or to optimize the performance. Speed, precision, and amount of equipment can usually be traded to improve one feature at the expense of another. Hence, refinements may be possible. The control logic for input, output, and timing operations has been left at a minimum for these illustrative programs, since it depends to a large extent upon the use of the program.

HALF ADDER

Binary addition is performed in the same manner as decimal addition and is clearly much simpler; the complete table for binary addition is as follows:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$ plus a *carry* of 1

"Carry's" are utilized in the same manner as in decimal arithmetic. Since 1 is the largest digit in the binary system, any sum greater than 1 requires that a digit be carried over. For instance, taking the sum of the two binary numbers 10 plus 10 (2 + 2 in decimal) requires the addition of the two ones in the second position to the left Since $1+1=0$

*Trademark

Printed in U.S.A. 364

Bulletin Number ALHC 6388-1

plus a carry of 1, the sum of 10 and 10 is 100 (decimal 4). Two additional examples of binary addition are as follows:

Decimal	Binary	Decimal	Binary
3	0011	10	1010
+ 5	0101	+ 12	1100
8	1000	22	10110

Simple binary addition as expressed by the above table is performed by a device known as a *half adder.* This device is capable of accepting two signals, representing the *augend* and *addend* digits, and producing output signals representing the *sum* and *carry.*

The term half adder arises from the fact that this elementary device has no provisions for adding the carry from the lower order to obtain the correct sum digit. Thus, a half adder can only sum two binary bits and, in the addition of multi-digit numbers, is useful only for adding the least significant or right hand digits.

From the half adder *truth table,* shown in Figure 1, it can be deduced that the logic operations required for performing simple binary addition are expressed with Boolean functions as

$$\text{Sum} = S = X\bar{Y} + \bar{X}Y$$
$$\text{Carry} = C = XY$$

INPUT		OUTPUT	
X	Y	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	1	1

Figure 1. Half Adder Truth Table

X and Y are the input signals of a given order in the *augend* and *addend*. This expression reads "the *sum* is 1 when X AND the complement* of Y are 1, OR when Y and the complement of X are 1".

Utilizing the rules of Boolean algebra, the expressions for the sum and carry can be factored and rearranged in a number of ways, and each different expression represents a physically different way of instrumenting a half adder. The logic expressions deemed most suitable for implementation with DOS components are

$$S = (X+Y)\,\bar{C}$$
$$C = XY$$

which is shown in the circuit of Figure 2.

*Since there are only two possible values a binary bit may take, 0 or 1, the complement of a 1 is 0; of a 0, 1. The complement of a binary signal is formed by an "inverter" and is represented in the notation by a bar over the signal. For example, $\bar{Y}$ is read as the complement of Y.

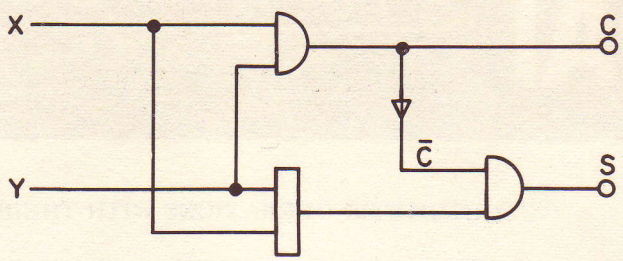


Figure 2. DOS Program for Half Adder.

FULL ADDER

It has been shown that when two binary numbers are added, there is a need to add not only the corresponding bits of the numbers, but also the carry bit from the lower order. To perform this operation, a three input device known as a *full adder* is required.

The following truth table is used to describe the transfer characteristics of the *full adder*

INPUT			OUTPUT	
X	Y	Z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 3. Full Adder Truth Adder

By noting the states on X, Y, and Z for which the sum and carry are 1, the following logic expressions can be written:

$$S = X\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z + XYZ$$
$$C = XY\bar{Z} + X\bar{Y}Z + \bar{X}YZ + XYZ$$

These can be factored and rearranged to a more convenient form:

$$S' = (X + Y)\,\bar{C'}$$
$$C' = XY$$
$$S = (S' + Z)\,\overline{S'Z}$$
$$C = C' + S'Z$$

The first two expressions are recognized as the logic expressions for a half adder. The simplest DOS circuit for implementing the full adder is shown in Figure 4 which, it is seen, consists of two half adder circuits and an "OR" gate for forming the carry.

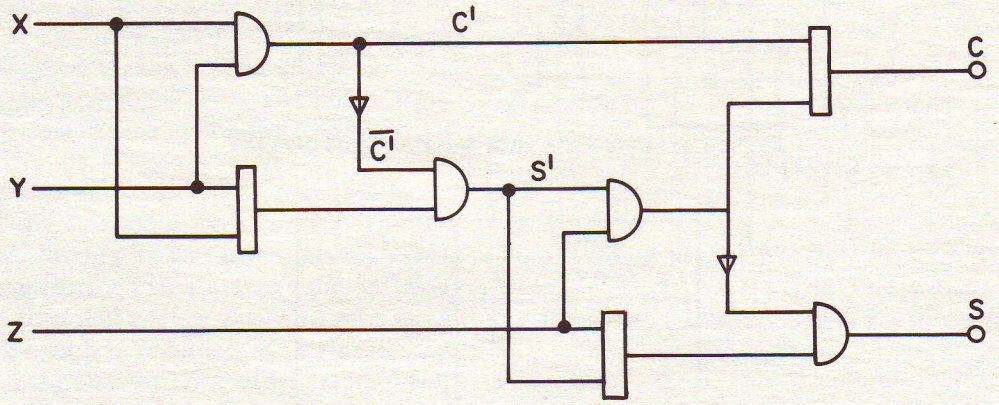


Figure 4. DOS Program for Full Adder

HALF SUBTRACTOR

Direct binary subtraction* is the inverse operation of addition. As such the binary subtraction table becomes

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1 with a *borrow* of 1

It can be seen that this procedure follows the pencil-and-paper method of decimal subtraction in that when a larger digit is subtracted from a smaller one it is necessary to "borrow" from the next column to the left. Thus when a 1 is subtracted from a 0 the remainder is 1, but it is necessary to borrow 1 from the next higher order. Two examples illustrate the procedures involved.

Decimal	Binary	Decimal	Binary
10	1010	28	11100
- 4	-0100	-7	-00111
6	0110	21	10101

Simple binary subtraction is performed by an elementary logic device known as the *halfsubtractor.* This device accepts two signals representing the *minuend* and *subtrahend* digits to produce a *difference* and a *borrow.* As above, the term half subtraction arises from the restriction of applying only two inputs to the device. Thus, it is subject to the same limitations as the half adder in that it is only capable of forming the difference for a single order of the binary numbers representing the minuend and subtrahend. By noting the states on the inputs in the truth table for which the difference and borrow are 1, one can write the following logic expressions for direct binary subtraction:

$$\text{Difference} = D = B + X\overline{Y}$$
$$\text{Borrow} = B = \overline{X} + \overline{Y}$$

INPUT		OUTPUT	
X	Y	D	B
0	0	0	0
1	0	1	0
0	1	1	1
1	1	0	0

Figure 5. Half Subtractor Truth Table

where

X = Minuend and Y = Subtrahend

These expressions are implemented by the DOS program shown in Figure 6. This form is chosen to emphasize the similarity of the adder and subtractor circuits (compare Figures 2 and 6).

FULL SUBTRACTOR

As in the case of binary addition, it follows that the subtraction of two binary numbers also requires a three input device known as a *full subtractor.* The binary subtraction rules for a full subtractor are presented in the following truth table.

Note that the X input of a subtractor is not interchangeable with the Y and Z inputs. This is in contrast to the adder where all three inputs are interchangeable. From this table the following logic

*In contrast to subtraction by means of complements.

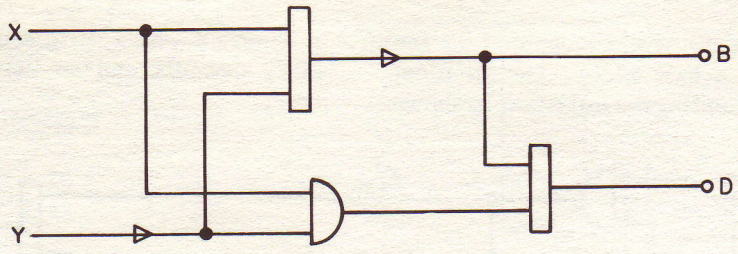


Figure 6. DOS Program for Half Subtractor

INPUT			OUTPUT	
X	Y	Z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Figure 7. Full Subtractor Truth Table

expressions for difference and borrow can be obtained:

$$D = X\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z + XYZ$$
$$B = \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z + \bar{X}YZ + XYZ$$

Once again, these expressions may be factored and rearranged to a more convenient form, or

$$\left.\begin{aligned} D' &= B' + X\bar{Y} \\ B' &= \overline{X + \bar{Y}} \end{aligned}\right\} \text{ outputs of half subtractor}$$
$$D = \overline{D' + \bar{Z}} + D'\bar{Z}$$
$$B = B' + \overline{D' + \bar{Z}}$$

The DOS circuit for implementing the full subtractor is shown in Figure 8. Note the similarity between Figures 4 and 8.

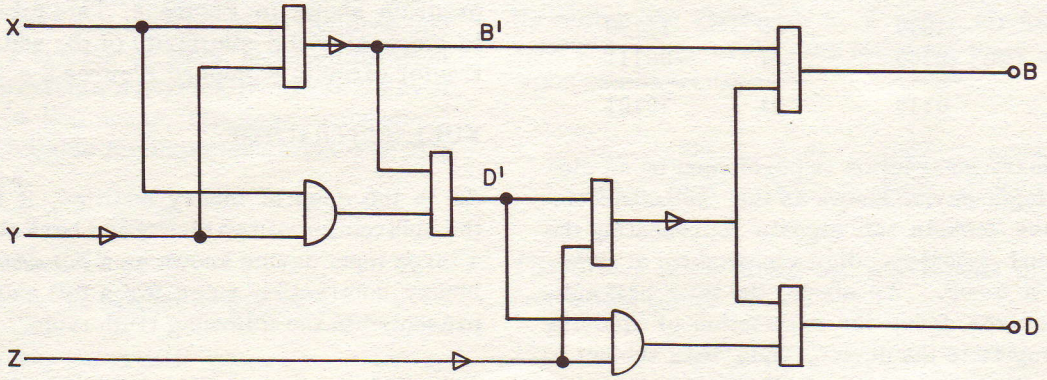


Figure 8. DOS Program for Full Subtractor

APPLICATIONS OF BASIC ARITHMETIC OPERATIONS

The preceding sections have established the methods whereby the basic operations of binary addition and subtraction are performed with DOS components. This section will show briefly how these basic circuits may be combined with additional logic operations to perform more complex functions.

SERIAL ADDERS AND SUBTRACTORS: One basic application of the full adder (or subtractor) is the addition (or subtraction) of two serial binary numbers. This is performed by starting with the least significant bit first and adding, successively, each pair of bits. If a sum is obtained, it is stored as a bit of the resulting number. If a carry is obtained, it is stored in a flip-flop to be added to the bits of the next higher significance. This addition can produce a sum, a carry, or a sum plus a carry. The carry is again stored for a *bit time,* or the time interval between the addition of successive digits. If this process is continued for the entire serial binary word, the serial binary number available sequentially from the sum output will equal the arithmetic sum (or difference) of the two input numbers.

One DOS program for the Serial Adder is shown in Figure 9 employing the previous full adder circuit and a General Purpose Flip-Flop to store the carry for one bit time.

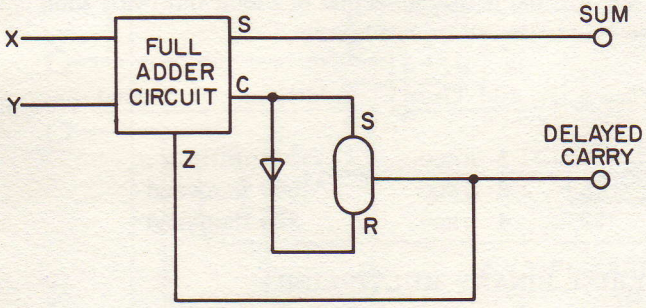


Figure 9. DOS Program for Serial Adder

A more efficient serial adder program[1] (one less OR gate is required for instance), assuming both X and Y and their complements* to be available, is shown in the circuit of Figure 10.

Since the basic arithmetic operations of addition and subtraction are employed extensively in the implementation of more complex digital arithmetic functions, the Adder-Subtractor Group is made

*All logic components in the DOS have both their true and complement outputs terminated on the patch panel.

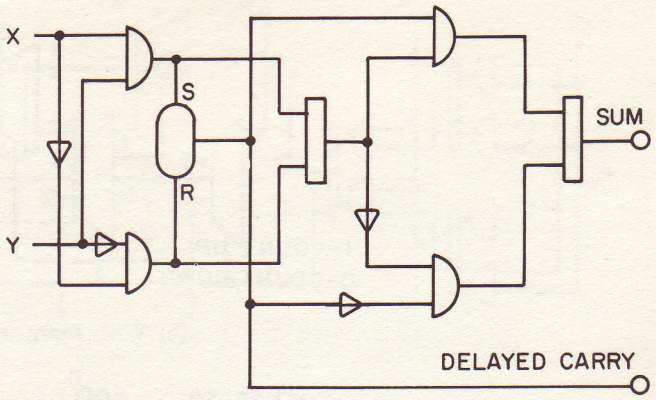


Figure 10. Alternate DOS Program for Serial Adder

available as a Logic Building Block in the DOS. This general purpose component performs addition and subtraction and alleviates the need to patch adder and/or subtractor circuits when many arithmetic operations are required on the DOS. The general characteristics of this group are outlined in Appendix I.

SERIAL NUMERICAL ADDER: When two binary numbers are to be added and the results need to be retained in a register as a whole binary number the circuit of Figure 11 may be used. If the output of the register is fed to one of the inputs (as in Figure 10), the program becomes a *serial accumulator.*

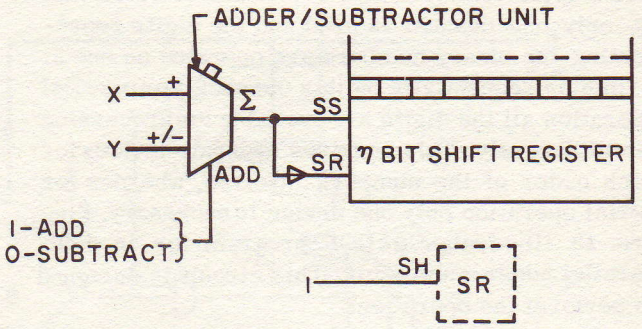
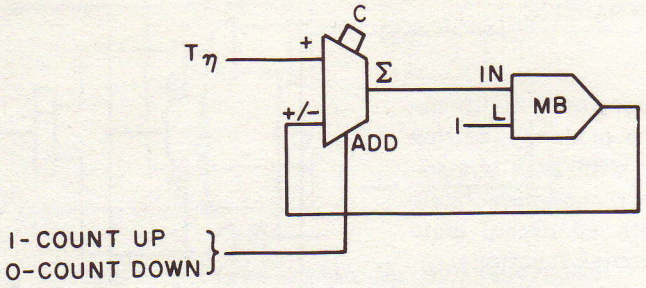
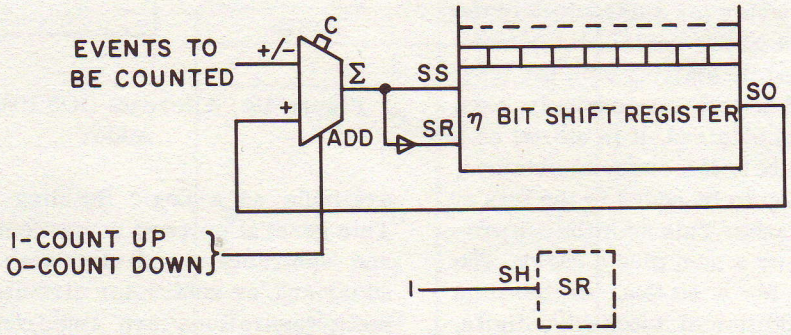


Figure 11. DOS Program for a Serial Numerical Adder

SERIAL BINARY COUNTER: Another simple application of serial adder and a shift register (or Memory Buffer) to provide a serial binary counter for counting up and/or down. The only limitation of this form of counter is that the input to be counted must not have a repetition period less than the cycle time of the shift register employed. Two DOS circuits for a serial binary *up-down counter* employing the Adder-Subtractor component are shown in Figure 12.



(a) With Memory Buffer (MB)



(b) With Shift Register

Figure 12. DOS Programs for Serial Binary Counter

If the addition operation is to be programmed with general purpose DOS logic components then only a serial half adder and/or subtractor is required for binary counting.

PARALLEL ADDER SUBTRACTOR: All DOS circuits described to this point are designed for arithmetic operations executed in a serial fashion. That is, only one channel is used and the digits representing the binary numbers are operated on one at a time or sequentially by this channel. For parallel operation all the digits are available for processing simultaneously. This requires separate devices for each order of the numbers involved, whereas for serial operation only one device is necessary. Figure 13 illustrates a DOS program for an n-bit parallel adder-subtractor. This circuit is designed to perform the operations

$$A + B = C$$
$$A - B = C, \text{ provided } |A| \geq |B|$$

where the numbers A, B, and C are defined by:

$$A = A_s\left[A_1 2^{-1} + A_2 2^{-2} + \ldots + A_n 2^{-n}\right] \quad A_s = \text{sign of A}$$
$$B = B_s\left[B_1 2^{-1} + B_2 2^{-2} + \ldots + B_n 2^{-n}\right] \quad B_s = \text{sign of B}$$
$$C = C_s\left[C_1 2^{-1} + C_2 2^{-2} + \ldots + C_n 2^{-n}\right] \quad C_s = \text{sign of C}$$

For addition, the control line marked ± is to be a ONE; for subtraction, a ZERO. A, B, and C are signed n-bit numbers, in sign-magnitude form. If A and B are not scaled properly an overflow warning can occur. Addition and subtraction is accomplished in a single clock interval. However there are practical limits to the number of bits, n, for a particular clock frequency. A rough guide is that n = 8 for each microsecond of clock interval; thus the "add times" possible are

n	Add Time	Additions per Second
8	1 μsec	1 million
16	2 μsec	500 thousand
32	4 μsec	250 thousand

SERIAL BINARY MULTIPLIER

A further arithmetic operation that can be performed with the general purpose logic on the DOS is binary multiplication. The multiplier program shown in Figure 14 accepts two serial binary numbers, each of 15 bits and sign, in 2's complement form, and provides the 2's complement binary product as one serial number of 15 bits and sign for single precision (SP OUT) or 30 bits and sign for the double length product (DP OUT). The product is also available in parallel for as many bits as desired. The multiplication process requires 16 word times of approximately 8 μsec for a total of approximately 128 μsec. The algorithm for the multiplier is described in Reference 2. The method employs a process of adding or subtracting the

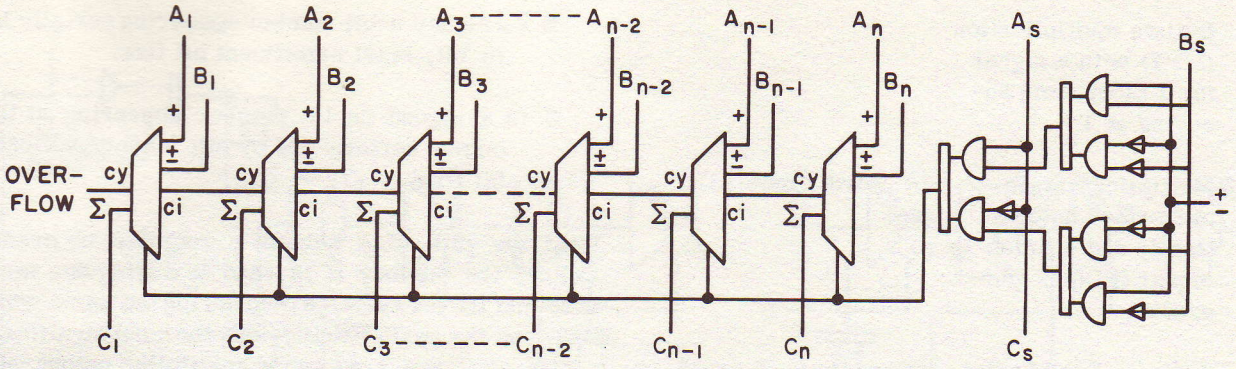


Figure 13. DOS Program for Parallel Adder-Subtractor

multiplicand or adding zero according to the state of the particular multiplier bit being considered as well as the next lower bit. A 15 bit register (Product Register) stores the intermediate results so that the multiplicand precesses to the left with respect to the partial products. As the least significant bits of the product are obtained, they are entered into the multiplier register each time it is shifted. When the multiplication is completed, the least significant bit of the register holding the single precision product is identical to the most significant bit of the lower portion of the double precision product.

The procedure for using the program is as follows:

	Operation Times
a. Load X (activate LX)	1 word time--8 μsec
b. Load Y (activate LY)	1 word time (can be same time as X if available)--8 μsec

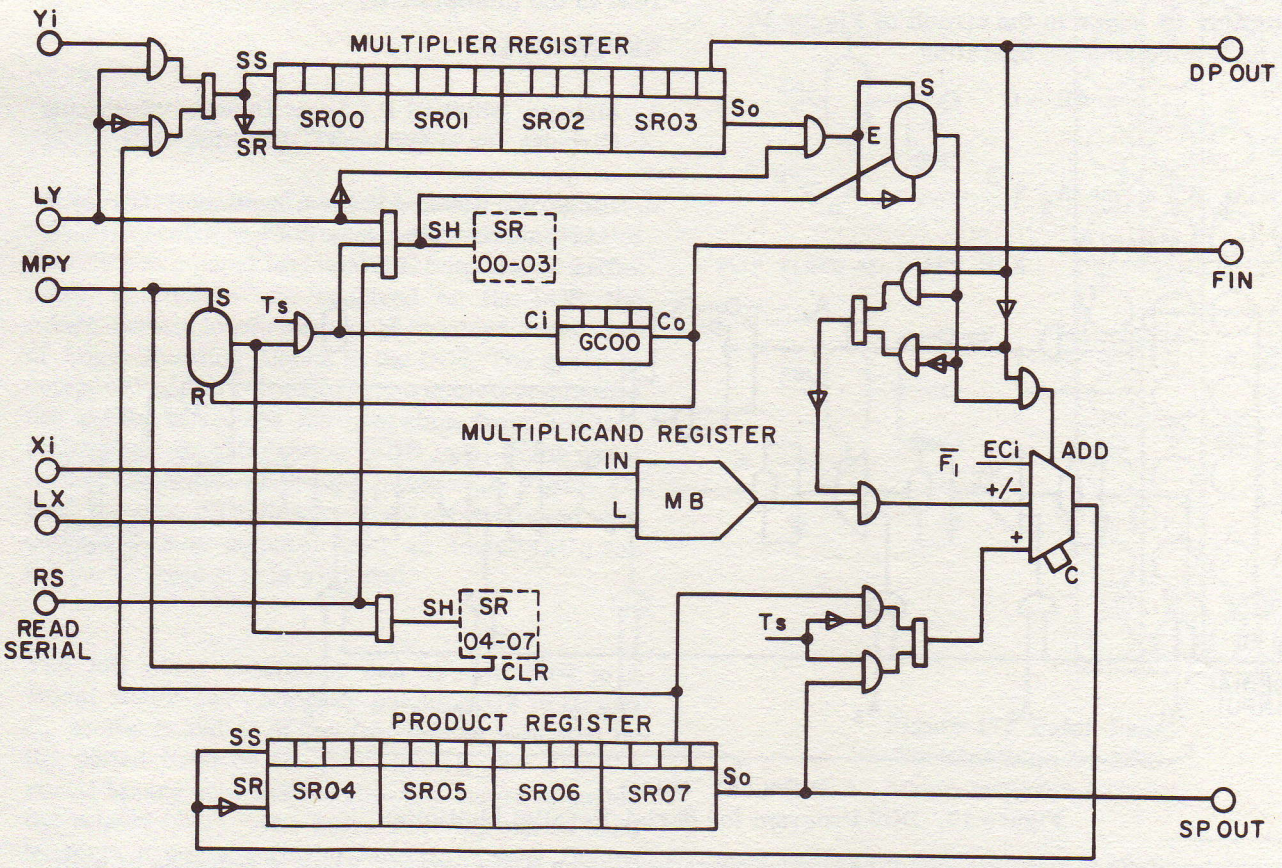


Figure 14. DOS Program for Serial Binary Multiplier

c. Initiate multiplication (MPY) with a signal for 1 clock time occuring at T_s

d. Multiplication is performed automatically and a finish signal (FIN) is generated. — 16 word times--128 μsec

e. Activate Read Serial (RS) and obtain single precision product at SP OUT — 1 word time--8 μsec*

or

Obtain the double length product (31 bits) at DP OUT — 2 word times--16 μsec*

SERIAL-PARALLEL MULTIPLICATION: At the expense of somewhat more equipment, faster multiplication can be obtained by holding the multiplicand in a static register and adding its bits in parallel with a group of adders according to the bits of the multiplier, which appear serially. This type of multiplier is shown in the circuit of Figure 15. This circuit performs the operation

$$AB = C$$

where

$$A = A_s\, A_1\, 2^{-1} + A_2\, 2^{-2} + \ldots + A_n\, 2^{-n},$$

A_s = sign of A

B is a signed n-bit number appearing serially bit by bit, least significant bit first

C is a signed 2n bit number appearing at the output serially bit by bit, least significant bit first.

With the bits of A held in a register, or preset switch, the number B is read in during one word time and the product C is read during the same word time and the next. Usually, only the most significant n-bits of C are used as the multiplier output, and these appear during the second word time. For two 16-bit numbers the multiplier is read serially in 8 microseconds and the lower significant n-bits of the 2n-bit product are read from the output during the same word time. The 16 most significant bits of the product are read at the output, in serial form, during the following word time. The maximum multiplication rate is about 65,000 products per second.

Multiplication speed can be doubled with the DOS circuit shown in Figure 16 where the most significant bits are read out in parallel after the first word time. For n=15, the product C appears in the flip-flops after 8 microseconds, which is the time to read in the multiplier B.

REFERENCES

1. Marcus, Mitchell P.; Logic Design; "Switching Circuits for Engineers", Prentice-Hall, 1962.

2. Richards, Richard Kohler; Arithmetic Circuits: "Arithmetic Operations in Digital Computers", Van Nostrand, 1957.

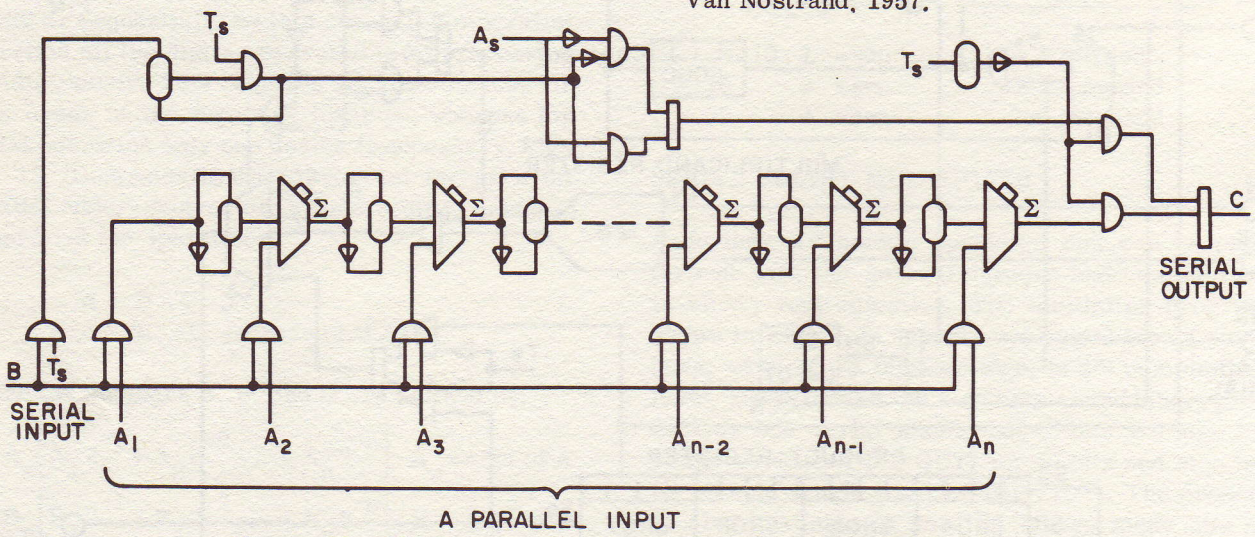


Figure 15. DOS Program for Serial-Parallel Multiplier

*New values of X+Y can be loaded for the next operation during these periods. Thus, a sequence of multiplications of pairs of numbers can be performed at the rate of 6950 products/sec (144 μsec each), if not at the full rate of 7800 products/sec (128 μsec each).

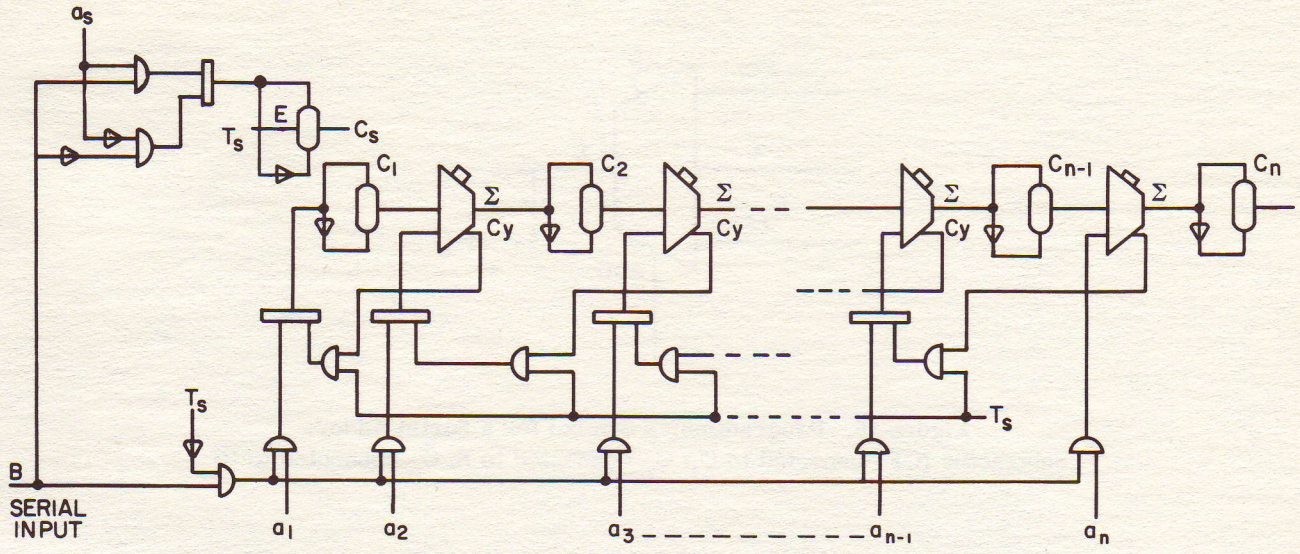


Figure 16. DOS Program for High Speed Serial-Parallel Multiplier

3. Ledley, Robert Steven; Arithmetic Circuits: "Digital Computers and Control Engineering", McGraw-Hill, 1960.

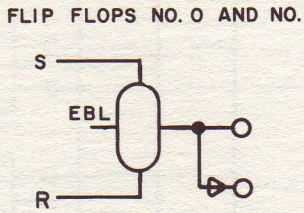
4. Huskey, Harry D.; and Korn, Granino A.; "Computer Handbook", McGraw-Hill, 1962.

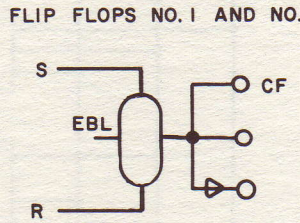
APPENDIX I

DOS ADDER-SUBTRACTOR GROUP

To alleviate the need to patch adder and/or subtractor circuits as well as to reduce the number of wasted patch panel terminations when many arithmetic operations are required on the DOS, the Adder-Subtractor Group has been added to the line of Logic Building Blocks in the DOS. The group is composed of two full adder-subtractor circuits with the ability to change the operation from addition (ADD high) to subtraction (ADD low). Patch panel terminations for the group are shown in Figure 17. Both the sum (Σ) and carry (C_O) are terminated with complementary outputs. Also, an Enable (EC_i) for the carry input (C_i) is provided.

The full adder-subtractor can be converted to a serial adder-subtractor by patching CF to C_i and C_o to the S and R of the flip-flop as indicated by the dotted lines on the patch panel. The Σ is the serial binary sum and the flip-flop output is called the *delayed carry*. The extra flip-flop can be used to form a serial-parallel multiplier (see Figure 15). The flip-flop outputs are displayed and controlled by appropriate indicator lights and push-buttons on the DOS control panels.





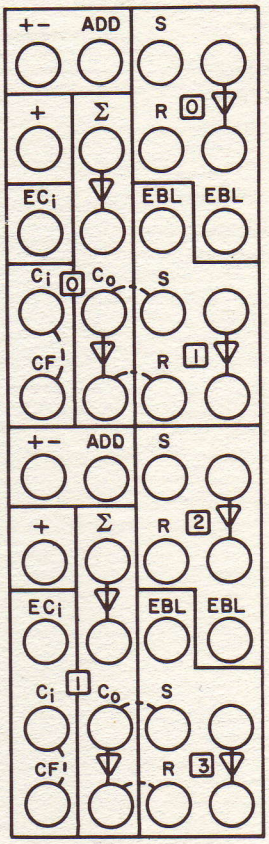


Figure 17. Patch Panel Terminations for DOS Adder-Subtractor Group

The programmer's symbol for a Serial Adder/Subtractor is shown in Figure 16. A suggested program for an equivalent adder-subtractor patched from DOS components is shown in Figure 17.

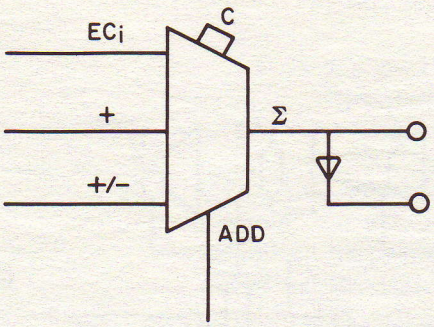


Figure 18. Programmer's Symbol for a Serial Adder-Subtractor (CF connected to C_i; C_o connected to S; $\overline{C}_o$ connected to R)

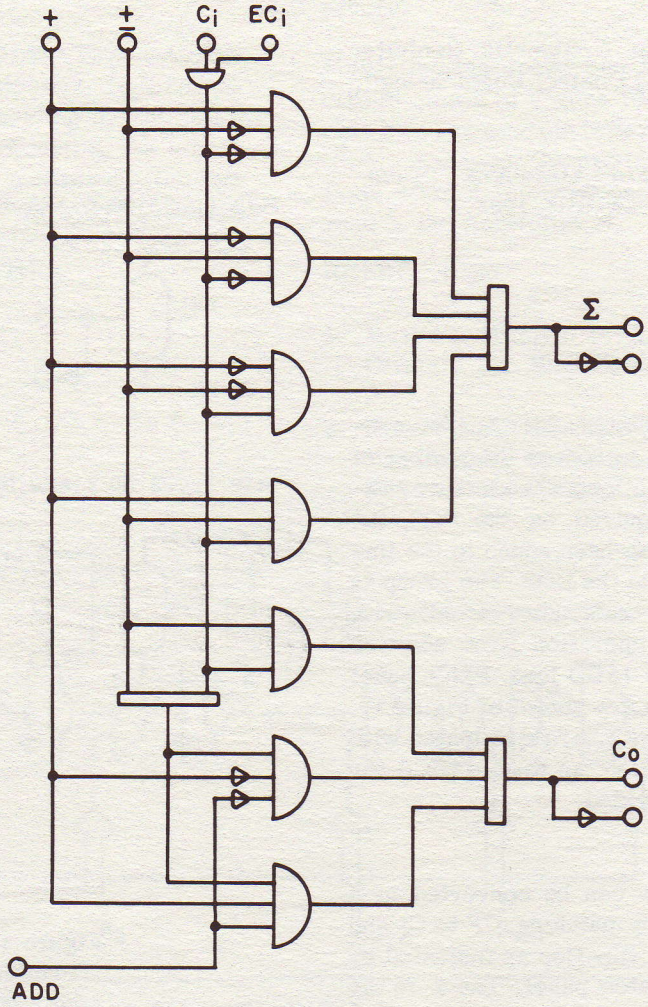


Figure 19. DOS Patched Program for Adder-Subtractor