NBS CIRCULAR 591

SECTION 1

System Design of Digital Computers at the National Bureau of Standards:

Methods for High-Speed Addition and Multiplication

UNITED STATES DEPARTMENT OF COMMERCE

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards

Functions and Activities

The functions of the National Bureau of Standards are set forth in the Act of Congress, March 3, 1901, as amended by Congress in Public Law 619, 1950. These include the development and maintenance of the national standards of measurement and the provision of means and methods for making measurements consistent with these standards; the determination of physical constants and properties of materials; the development of methods and instruments for testing materials, devices, and structures; advisory services to Government Agencies on scientific and technical problems; invention and development of devices to serve special needs of the Government; and the development of standard practices, codes, and specifications. The work includes basic and applied research, development, engineering, instrumentation, testing, evaluation, calibration services, and various consultation and information services. A major portion of the Bureau's work is performed for other Government Agencies, particularly the Department of Defense and the Atomic Energy Commission. The scope of activities is suggested by the listing of divisions and sections on the inside of the back cover.

Publications

The results of the Bureau's work take the form of either actual equipment and devices or published papers. These papers appear either in the Bureau's own series of publications or in the journals of professional and scientific societies. The Bureau itself publishes three monthly periodicals, available from the Government Printing Office: The Journal of Research, which presents complete papers reporting technical investigations; the Technical News Bulletin, which presents summary and preliminary reports on work in progress; and Basic Radio Propagation Predictions, which provides data for determining the best frequencies to use for radio communications throughout the world. There are also five series of nonperiodical publications: The Applied Mathematics Series, Circulars, Handbooks, Building Materials and Structures Reports, and Miscellaneous Publications.

Information on the Bureau's publications can be found in NBS Circular 460, Publications of the National Bureau of Standards ($1.25) and its Supplement ($0.75), available from the Superintendent of Documents, Government Printing Office, Washington 25, D. C.

UNITED STATES DEPARTMENT OF COMMERCE • Sinclair Weeks, *Secretary*
NATIONAL BUREAU OF STANDARDS • A. V. Astin, *Director*

System Design of Digital Computer at the National Bureau of Standards:

Methods for High-Speed Addition and Multiplication

National Bureau of Standards Circular 591, Section 1
Issued February 14, 1958

For sale by the Superintendent of Documents, U. S. Government Printing Office ,Washington 25, D. C.
Price 20 cents

Contents

Introduction

A. L. Leiner

In the seven years since SEAC first began regular operation (April 1950), the feasibility of applying general-purpose digital computing machines to a broad range of new problems has become well established. During this period, digital computers have been successfully put to work on a wide variety of scientific and engineering calculations, clerical and technical data-processing jobs, and real-time control operations.

For some important applications, however, the opportunity of using automatic computers is still severely restricted by the limited speed at which available digital machines can carry out the basic arithmetical operations. For this reason a considerable development effort is still being directed toward the design of faster digital arithmetic devices.

In attacking the general problem of how to increase arithmetic computation speeds, two distinct paths are open to the computer designer. One approach consists in trying to develop electronic components and circuitry capable of amplifying, storing and switching signals at higher basic repetition rates. The other approach attempts to gain over-all speed by finding methods for organizing slower electronic elements into more efficient over-all systems. In following the latter course, the increased problem-solving speed is obtained by harnessing the same simple computing elements together in more complex logical combinations. A typical example of the latter method, for instance, would be to organize the basic computing elements in such a way that several separate steps of a given over-all operation could be carried out simultaneously by different units in the same machine. A straightforward application of this principle, however, would often entail the use of excessively large amounts of additional equipment. As a rule, therefore, methods based on the use of duplicated equipment must also be coupled with the use of more subtle algorithms for carrying out the basic processes and the use of processing logics specially adapted to the physical peculiarities of the available computing components.

The two papers that follow represent attempts to produce faster arithmetic processing units by increasing the complexity of their logical structure without at the same time materially increasing the number of components used. In the first paper, which is concerned with addition and subtraction operations, the increased speed is obtained chiefly by using the early phases of the operation for generating certain special auxiliary functions of the digits in the numbers being added together. These auxiliary functions are then used later on in the process to facilitate the determination of the conventional "carry" signals that are required in the last stages of the operation. The adder designs illustrated are particularly suited to the special type of dynamic pulse circuitry developed at NBS during the course of the SEAC and DYSEAC programs. When associated with a suitably fast storage device, these adder designs yield over-all arithmetic processing units that perform up to 150 times faster than the earlier machines. With appropriate modifications, the same principles could be applied to other types of electronic circuitry.

The second paper is concerned with the more general problem of how multiplication speeds can be improved by omitting certain time-consuming steps that are shown to be not strictly necessary. These steps are ordinarily included when the elementary definition of multiplication as a simple sequence of repeated addition operations is applied in too straightforward a fashion. The authors show how, by making a simple preliminary inspection of one of the numbers to be multiplied, up to two-thirds of the usual addition steps can sometimes be skipped. In this way speed increases of of 2.5 to 3 over standard methods can generally be obtained. This shortcut multiplication method, unlike the previously described addition method, is of general applicability and not necessarily associated with any particular class of computing component or circuitry. In terms of equipment needs it compares most favorably with other proposed methods that involve the use of several separate addition devices.

The present Circular contains the first part of a group of papers to be published on the logical design work carried out at NBS since 1948 on digital-computing, data-processing, and control systems. The second section, in a forthcoming Circular, will describe several digital systems other than the SEAC and DYSEAC that have been designed for various general-purpose and special-purpose applications.

1. A Logic for High-Speed Addition

A. Weinberger and J. L. Smith

1. Introduction

The development at the National Bureau of Standards of the diode capacitor memory [1,2],[1] which is capable of being read or written into randomly at the rate of one word per microsecond, has made it worth while to build devices capable of processing information at comparable rates. Since the basic microo-peration common to most arithmetic processes is the adding of two numbers, it seemed desirable to design an adder having a cycle time no greater than 1 μsec.

The major speed limitation in any adder is in the production of carries, and in this paper the problem is attacked from the standpoint of logical organization. Although work is being done elsewhere on this subject, using newer and faster basic circuit elements, the analyses to be described show that it is both feasible and economical to achieve 1-μsec addition times for 53-bit words, using the 1-Mc circuitry that has been successfully utilized in SEAC [3] and DYSEAC [4,5].

The increased complexity of the logic of this adder necessitated the extensive use of Boolean algebra in arriving at the design itself. Because the procedure used in developing the final design is an interesting example of the practical application of Boolean algebra, the actual logic of the design process is described in considerable detail.

Before discussing the adder, a brief description of the logical capabilities of the SEAC circuitry [6] will be presented. As shown in figure 1.1, the basic electronic unit consists essentially of three levels of diode gates in an OR–AND–OR logical array followed by a transformer-coupled pulse amplifier. The rate at which successive pulses pass through such a stage is determined by the clock frequency, which is, in this case, 1 Mc/sec. The transit time of a pulse through a stage, however, is much less than 1 μsec. For this reason, the clock pulses are made available in several phases. The way in which different stages are controlled by clock pulses of different phases is illustrated in figure 1.2. In SEAC, for example, 1-Mc clock pulses are available in 3 phases, $\frac{1}{3}$ μsec apart. In DYSEAC, 4-phase clock pulses are used, whereas for reasons that will be developed later, in the adder to be described a 5-phase clock is used. Figure 1.3 shows graphically these timing relationships for SEAC. Signals emitted from different stages clocked at different times must be synchronized by means of electric delay lines before they are gated in a common stage, as shown in figure 1.4. Both positive and negative signals are available from a stage, the negative signals being used for inhibiting (see fig. 1.5).

The logical gating required in any stage of the adder to be described is essentially of the same complexity as that required in the packaged building blocks used in constructing DYSEAC, and in the OR–AND–OR gating configuration of a stage up to 4 AND-gates and up to 6 inputs in the largest AND-gate are permitted.

Boolean notation of the sort described by Richards [7] will be used hereafter to describe the gating configurations. In figure 1.6 are shown a typical gating stage and the corresponding Boolean expression for the output in terms of the inputs. There are three terms in the expression, each one corresponding to an AND-gate; the first term, $(A+B)\overline{C}\overline{D}EF$, corresponds to the top AND-gate; the second term, $(G+H)I$, corresponds to the middle AND-gate; and the last term, $J(K+L+M)\overline{N}$, corresponds to the bottom AND-gate. The factors of a term represent the inputs to the corresponding AND-gate. For example, the five factors of the first term, $(A+B)$, $\overline{C}$, $\overline{D}$, E, and F, correspond to the five inputs to the top AND-gate Whenever a factor consists of more than one term, it is represented by an OR-gate. For example, the factor $(A+B)$ of the first term corresponds to the 2-input OR-gate of the top AND-gate. A factor could also be a negative or inhibit signal, and in this case it is denoted by a bar on top; e. g., $\overline{C}$ and $\overline{D}$ are two factors of the first term corresponding to the two negative signals, which may inhibit the top AND-gate. For the sake of simplicity in the discussion of the Boolean expressions that follow, no distinction is made between delayed and undelayed signals.

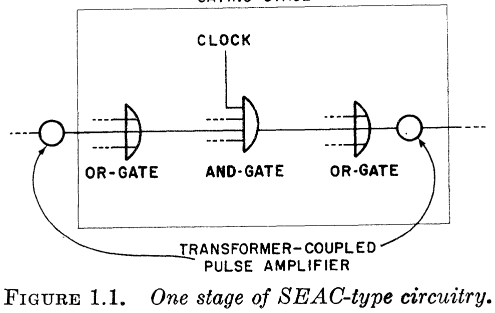

FIGURE 1.1. *One stage of SEAC-type circuitry.*

[1] Figures in brackets indicate the literature references on page 12.

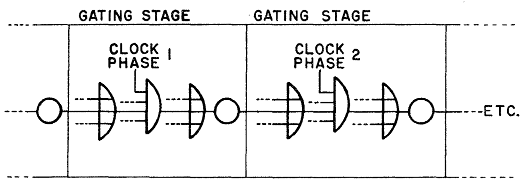

FIGURE 1.2. *Gating stages with different clock phases.*

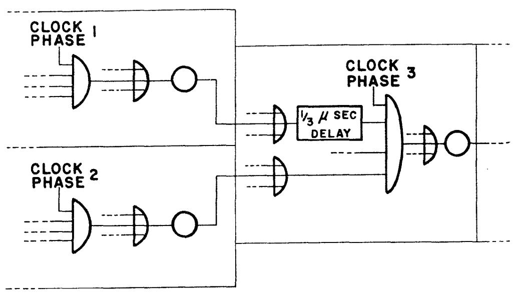

FIGURE 1.4. *Synchronizing by means of electric delay lines.*

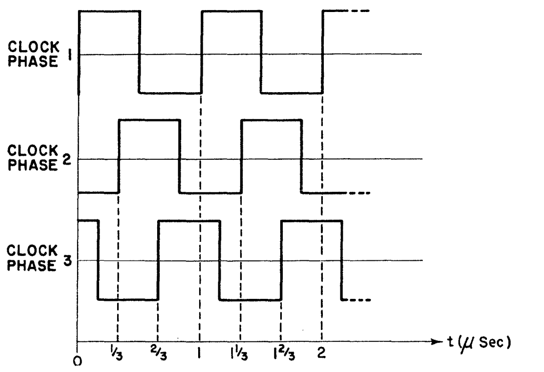

FIGURE 1.3. *Time relationships among SEAC clock phases.*

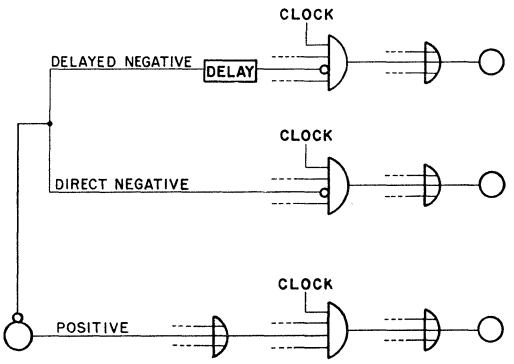

FIGURE 1.5. *Use of negative signals for inhibiting.*

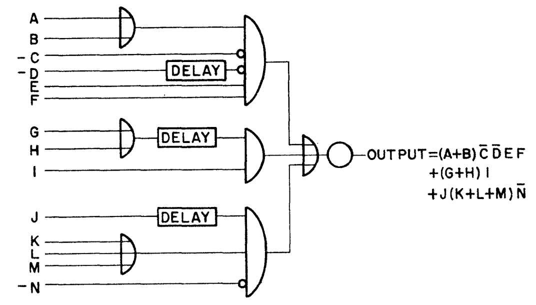

FIGURE 1.6. *Typical gating stage and corresponding Boolean expression.*

2. Sequential Carry Generation

The analysis leading to the design of the parallel adder will now be described in detail.

Let

$$A = \text{augend} = A_k 2^{k-1} + A_{k-1} 2^{k-2} + \ . \ . \ . \ + A_1 2^0,$$

$$B = \text{addend} = B_k 2^{k-1} + B_{k-1} 2^{k-2} + \ . \ . \ . \ + B_1 2^0,$$

$$S = \text{sum} \quad = S_k 2^{k-1} + S_{k-1} 2^{k-2} + \quad . \ + S_1 2^0,$$

C_k = the carry resulting from the addition in the kth digit position.

The well-known rules for binary addition are given in the form of a function table (table 1.1). From these, the binary sum and carry can be expressed in Boolean notation as follows:

$$S_k = \bar{A}_k \bar{B}_k C_{k-1} + \bar{A}_k B_k \bar{C}_{k-1} + A_k \bar{B}_k \bar{C}_{k-1} + A_k B_k C_{k-1}. \tag{1}$$

TABLE 1.1. *Function table for binary addition*

Augend----	A_k	0	0	0	0	1	1	1	1
Addend----	B_k	0	0	1	1	0	0	1	1
Previous carry.	C_{k-1}	0	1	0	1	0	1	0	1
Sum-------	S_k	0	1	1	0	1	0	0	1
Carry------	C_k	0	0	0	1	0	1	1	1

$$\begin{aligned} C_k &= \overline{A}_k B_k C_{k-1} + A_k \overline{B}_k C_{k-1} + A_k B_k \overline{C}_{k-1} + A_k B_k C_{k-1} \\ &= A_k B_k + A_k C_{k-1} + B_k C_{k-1} \\ &= (A_k + B_k)(A_k + C_{k-1})(B_k + C_{k-1}) \qquad (2) \\ &= A_k B_k + (A_k + B_k) C_{k-1}. \end{aligned}$$

The carry function, C_k, has been reduced from 4 terms of 3 factors each (corresponding to 4 AND-gates with 3 inputs each), as shown in the top line of eq (2), to 3 alternative forms, each involving fewer terms and factors.

Since the expression for S_k in eq (1) can be implemented in one gating stage, any sum digit can be made available during the clock phase immediately following the formation of its corresponding carry, C_{k-1}. However, if the carries are generated according to eq (2), each carry digit would have to await the formation of the next lower-order carry. As a result, the sum digits could be obtained at the rate of only one per clock phase, for if C_1 is available during the first clock phase, C_2 could be generated during the second clock phase, C_3 during the third clock phase, etc. For numbers having n binary digits, $n-1$ possible carries would have to be provided for, requiring $n-1$ clock phases for their complete determination. If a 4-phase, 1-Mc clock were used, 4 successive sum digits could be obtained during 1 μsec. Such an arrangement, using *sequential* carry generation, would provide an increase in speed of a factor of only four over the addition speed of a completely serial adder.

3. Simultaneous Carry Generation

The limitation on the sequential method of forming the carries stems from the use of eq (2), which specify C_k as an explicit function of C_{k-1}. It can be shown that a carry need not depend explicitly on the preceding one, but can be expressed as a function of only the relevant augend and addend digits and some lower-order carry. A considerable gain in speed may be obtained as a result of this.

Using the functional form given by the last equation in (2), successive carries are shown to be expressible in terms of the same lower-order carry by a method of substitution.

$$\begin{aligned}
C_1 = {} & A_1B_1 \\
& + (A_1+B_1)C_0 \\[1em]
C_2 = {} & A_2B_2 \qquad = A_2B_2 \\
& + (A_2+B_2)C_1 \qquad + (A_2+B_2)A_1B_1 \\
& \qquad\qquad + (A_2+B_2)(A_1+B_1)C_0 \\[1em]
C_3 = {} & A_3B_3 \qquad = A_3B_3 \\
& + (A_3+B_3)C_2 \qquad + (A_3+B_3)A_2B_2 \\
& \qquad\qquad + (A_3+B_3)(A_2+B_2)A_1B_1 \\
& \qquad\qquad + (A_3+B_3)(A_2+B_2)(A_1+B_1)C_0 \qquad (3) \\[1em]
C_4 = {} & A_4B_4 \qquad = A_4B_4 \\
& + (A_4+B_4)C_3 \qquad + (A_4+B_4)A_3B_3 \\
& \qquad\qquad + (A_4+B_4)(A_3+B_3)A_2B_2 \\
& \qquad\qquad + (A_4+B_4)(A_3+B_3)(A_2+B_2)A_1B_1 \\
& \qquad\qquad + (A_4+B_4)(A_3+B_3)(A_2+B_2)(A_1+B_1)C_0 \\
& \qquad = A_4B_4 \\
& \qquad\qquad + (A_4+B_4)A_3B_3 \\
& \qquad\qquad + (A_4+B_4)(A_3+B_3)A_2B_2 \\
& \qquad\qquad + (A_4+B_4)(A_3+B_3)(A_2+B_2)(A_1+B_1)(A_1+C_0)(B_1+C_0).
\end{aligned}$$

Equations (3) show how as many as 4 successive carries can be expressed as functions of the same carry, with all expressions consisting of no more than 4 terms and with the largest term consisting of no more than 6 factors. These 4 carries can therefore be generated simultaneously by means of only 4 gating stages.

Similarly, the next more significant four carries, C_5 through C_8, can be formed simultaneously during the next clock phase as functions of the appropriate augend and addend digits and C_4. In short, four successive carry digits can be formed simultaneously every clock phase. One gating stage per carry is required.

To summarize, if C_0 is available in the first clock phase, C_1 through C_4 can be generated during the second clock phase, C_5 through C_8 during the third clock phase, etc. Each group of sum digits can be obtained one clock phase after the corresponding group of carries has been formed. Figure 1.7 illustrates in block-diagram form an adder utilizing this principle of simultaneous carry generation.

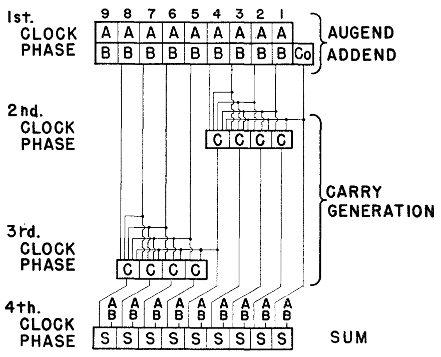

FIGURE 1.7. *Nine-bit parallel binary adder.*

4. Use of Auxiliary Carry Functions

Of signal importance is the use that can be made of the second clock phase to further speed up the addition process. This time can be utilized to form certain *auxiliary carry functions*, which enable additional carries to be generated during the third clock phase simultaneously with the carries C_5 through C_8. More specifically, C_9, C_{10}, etc., can be formed during the third clock phase as functions of C_4 if some of the terms involving only the augend and addend digits in the expanded relations for C_9, C_{10}, etc., are combined as auxiliary carry functions in separate stages during the intervening clock phase.

For example, the expression for C_9 is shown in the first equation in (4) expanded as a function of C_4. Because of limitations on the gating complexity, it is not possible to form C_9 directly even if it were reduced to four terms. Instead, the function is implemented by parts.

$$\begin{aligned}
C_9 = {} & A_9B_9 \\
&+(A_9+B_9)A_8B_8 \\
&+(A_9+B_9)(A_8+B_8)A_7B_7 \\
&+(A_9+B_9)(A_8+B_8)(A_7+B_7)A_6B_6 \\
&+(A_9+B_9)(A_8+B_8)(A_7+B_7)(A_6+B_6)A_5B_5 \\
&+\boxed{(A_9+B_9)(A_8+B_8)(A_7+B_7)(A_6+B_6)(A_5+B_5)}\,C_4
\end{aligned} \tag{4}$$

$$\begin{aligned}
C_9 = {} & X_9 \\
&+\boxed{Y_9}\,C_4
\end{aligned}$$

(The outlines drawn around the various parts of eq (4) serve merely to correlate the corresponding parts of the two equations.) The 5 terms enclosed within the triangle can be reduced to 4 terms by combining the first 2 terms. This reduced 4-term expression can then be implemented in 1 gating stage during the second clock phase, and it is then designated by X_9. The single factor enclosed within the rectangle can also be implemented during the second clock phase in one gating stage. It is designated by Y_9. By means of these 2 auxiliary carry functions, X_9 and Y_9, the actual carry C_9 can be formed quite handily in 1 gating stage during the third clock phase, according to the second equation in (4).

The next 4 carries, C_{10} through C_{13}, can also be formed during the third clock phase by utilizing these same auxiliary carry functions. The most complicated of these expressions, the one for C_{13}, is given in eq (5), where further combinations are made to reduce the number of terms to four.

$$\begin{aligned}
C_{13} = {} & A_{13}B_{13} \\
&+(A_{13}+B_{13})A_{12}B_{12} \\
&+(A_{13}+B_{13})(A_{12}+B_{12})A_{11}B_{11} \\
&+(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})A_{10}B_{10} \\
&+(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})(A_{10}+B_{10})X_9 \\
&+(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})(A_{10}+B_{10})Y_9C_4
\end{aligned} \tag{5}$$

$$\begin{aligned}
C_{13} = {} & (A_{13}+B_{13})(A_{13}+A_{12})(A_{13}+B_{12})(B_{13}+A_{12}) \\
&\quad (B_{13}+B_{12}) \\
&+(A_{13}+B_{13})(A_{12}+B_{12})A_{11}B_{11} \\
&+(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})A_{10}B_{10} \\
&+(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})(A_{10}+B_{10}) \\
&\quad (X_9+Y_9)(X_9+C_4).
\end{aligned}$$

Figure 1.8 illustrates a parallel adder that will complete an addition on 14 binary digits in 4 clock phases, using 1 pair of auxiliary carry functions.

By means of additional auxiliary carry functions it is possible to extend still further the sequence of carries that can be formed in the same clock phase. For example, as shown in eq (6), C_{14} can be expressed as a simple function involving C_4 and another pair of auxiliary carry functions, X_{14} and Y_{14}, which are defined implicitly in eq (6).

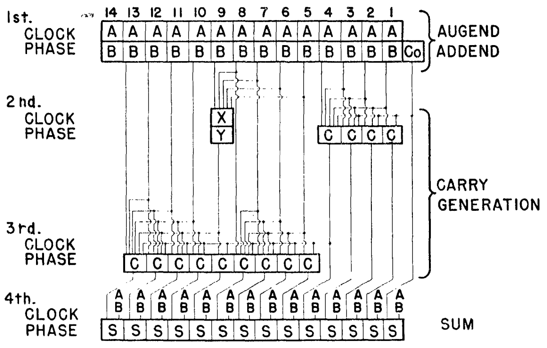

FIGURE 1.8. *Fourteen-bit parallel binary adder.*

$$
\begin{aligned}
C_{14} = & A_{14}B_{14} \\
& +(A_{14}+B_{14})A_{13}B_{13} \\
& +(A_{14}+B_{14})(A_{13}+B_{13})A_{12}B_{12} \\
& +(A_{14}+B_{14})(A_{13}+B_{13})(A_{12}+B_{12})A_{11}B_{11} \\
& +(A_{14}+B_{14})(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})A_{10}B_{10} \\
& +(A_{14}+B_{14})(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})(A_{10}+B_{10})X_9 \\
& +(A_{14}+B_{14})(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})(A_{10}+B_{10})Y_9C_4
\end{aligned}
\qquad (6)
$$

$$
\begin{aligned}
C_{14} = & X_{14} \\
& +Y_{14}X_9 \\
& +Y_{14}Y_9C_4
\end{aligned}
$$

C_{15}, C_{16}, and C_{17} can also be implemented in single stages as functions of C_4 by using the same two pairs of auxiliary carry functions. C_{18}, C_{19}, and C_{20} require still a third pair of auxiliary carry functions in order that they be generated during the same clock phase as functions of C_4.

If it were desired, a total of 25 carries could be generated simultaneously as functions of C_4 during the third clock phase without exceeding the limitations on gating complexity. However, if the number of simultaneous carries is limited to 16, only 3 pairs of auxiliary carry functions are required. Figure 1.9 illustrates a parallel adder that can add numbers of 21 binary digits in 4 clock phases, utilizing this scheme.

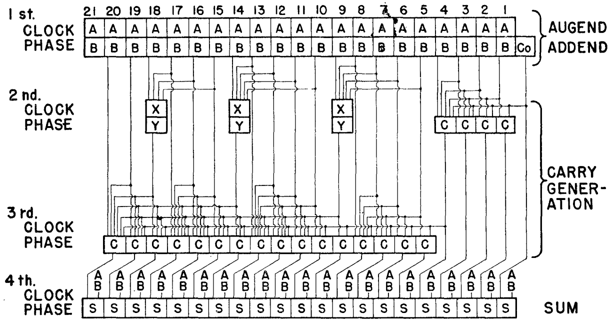

FIGURE 1.9. *Twenty-one-bit parallel binary adder.*

5. Two Levels of Auxiliary Carry Functions

To extend the parallel adder to accommodate 53 binary digits, it will be shown that only 1 additional clock phase is necessary, and that during the fourth clock phase the carries C_{21} through C_{52} can all be generated as functions of C_{20}. The entire parallel array of sum digits, S_1 through S_{53}, can then be formed during the fifth clock phase.

The ability to generate all of the carries C_{21} through C_{52} during the fourth clock phase stems from the fact that two clock phases are available between these carries and the input digits. This permits the formation of *two levels* of auxiliary functions. The first level consists of sets of X's and Y's, which are functions of the relevant augend and addend digits only, as was the case previously. The second-level auxiliary carry functions are generated by sets of stages labeled Z and W and are functions of certain first-level functions only.

Figure 1.10 illustrates in block-diagram form the complete 53-bit adder, which makes use of first-level and second-level auxiliary carry stages. As in the case of the preceding carries, C_{21} through C_{32} are generated as functions of the appropriate augend and addend digits, some of the first-level auxiliary carry stages, and C_{20}. For example, the most complicated of these, C_{32}, is shown in eq (7) to be reducible to four terms.

$$\begin{aligned}
C_{32}=\;&A_{32}B_{32}\\
&+(A_{32}+B_{32})A_{31}B_{31}\\
&+(A_{32}+B_{32})(A_{31}+B_{31})A_{30}B_{30}\\
&+(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})X_{29}\\
&+(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})Y_{29}X_{25}\\
&+(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})Y_{29}Y_{25}C_{20}
\end{aligned}\tag{7}$$

$$\begin{aligned}
C_{32}=\;&(A_{32}+B_{32})(A_{32}+A_{31})(A_{32}+B_{31})(B_{32}+A_{31})(B_{32}+B_{31})\\
&+(A_{32}+B_{32})(A_{31}+B_{31})A_{30}B_{30}\\
&+(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})(X_{29}+Y_{29})(X_{29}+X_{25})\\
&+(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})Y_{29}Y_{25}C_{20}.
\end{aligned}$$

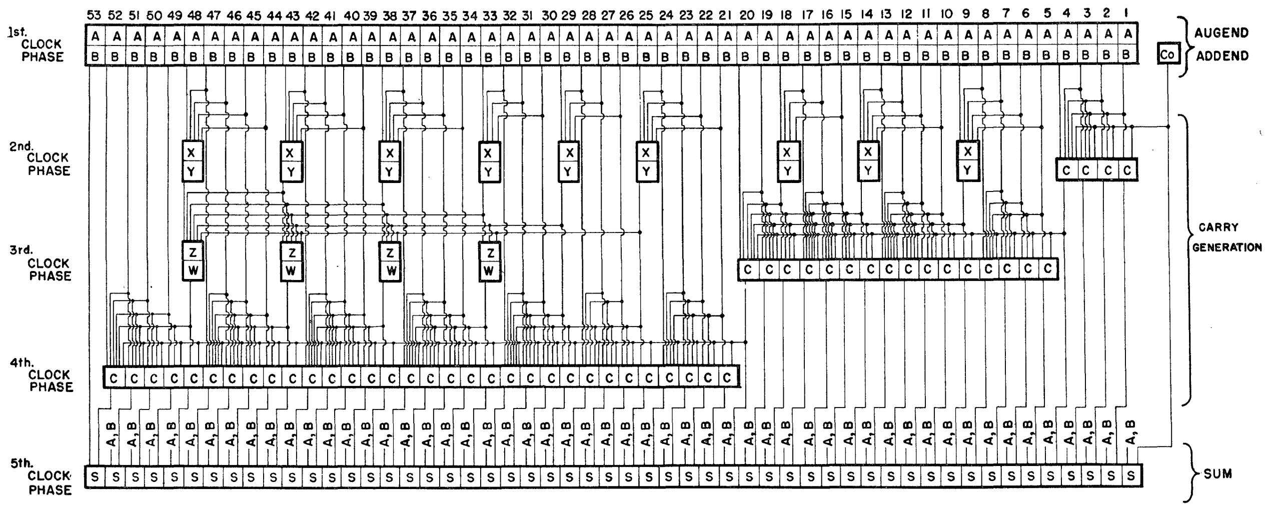

FIGURE 1.10. *Fifty-three-bit parallel binary adder.*

The next higher-order carry, C_{33}, requires a third pair of auxiliary carry functions, X_{33} and Y_{33}, as shown in eq (8). Also, at this point it becomes economical to form a pair of second-level auxiliary carry functions, Z_{33}, consisting of the terms within the solid-line triangle, and W_{33}, consisting of the factors within the solid-line rectangle. C_{33} can then be simply generated by means of Z_{33} and W_{33}, as shown in the last of eq (8).

The subsequent carries, C_{34}, C_{35}, etc., are similarly generated by means of these and, when necessary, other second-level auxiliary carry functions. For example, for the carries up to C_{37}, the same pair of second-level functions, W_{33} and Z_{33}, is sufficient, whereas C_{38} requires the use of another pair of first-level and second-level auxiliary carry functions in a manner exactly analogous to the formation of C_{33}.

$$\begin{aligned}
C_{33} = {} & A_{33}B_{33} \\
& + (A_{33}+B_{33})A_{32}B_{32} \\
& + (A_{33}+B_{33})(A_{32}+B_{32})A_{31}B_{31} \\
& + (A_{33}+B_{33})(A_{32}+B_{32})(A_{31}+B_{31})A_{30}B_{30} \\
& + (A_{33}+B_{33})(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})X_{29} \\
& + (A_{33}+B_{33})(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})Y_{29}X_{25} \\
& + (A_{33}+B_{33})(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})Y_{29}Y_{25}C_{20} \\[1em]
C_{33} = {} & X_{33} \\
& + Y_{33}X_{29} \\
& + Y_{33}Y_{29}X_{25} \\
& + Y_{33}Y_{29}Y_{25}C_{20} \\[1em]
C_{33} = {} & Z_{33} \\
& + W_{33}C_{20}
\end{aligned} \tag{8}$$

The last digit position where auxiliary carry functions are introduced is at 48. The carry at this position, C_{48}, is shown in eq (9) to be a simple function of the last pair of second-level auxiliary carry functions.

$$
\begin{array}{rl}
C_{48}= & X_{48} \\
+ & Y_{48}X_{43} \\
+ & Y_{48}Y_{43}X_{38} \\
+ & Y_{48}Y_{43}Y_{38}X_{33} \\
+ & Y_{48}Y_{43}Y_{38}Y_{33}X_{29} \\
+ & Y_{48}Y_{43}Y_{38}Y_{33}Y_{29}X_{25} \\
\hline
+ & Y_{48}Y_{43}Y_{38}Y_{33}Y_{29}Y_{25}\,\big|\,C_{20}
\end{array}
\qquad (9)
$$

$$
\begin{array}{rl}
C_{48}= & X_{48} \\
+ & Y_{48}X_{43} \\
+ & Y_{48}Y_{43}(X_{38}+Y_{38})(X_{38}+X_{33}) \\
+ & Y_{48}Y_{43}Y_{38}Y_{33}(X_{29}+Y_{29})(X_{29}+X_{25}) \\
\hline
+ & Y_{48}Y_{43}Y_{38}Y_{33}Y_{29}Y_{25}\,\big|\,C_{20}
\end{array}
$$

$$
\begin{array}{rl}
C_{48}= & Z_{48} \\
\hline
+ & W_{48}\,\big|\,C_{20}
\end{array}
$$

The collected Boolean expressions for the auxiliary carry functions and for the carry functions themselves for this particular 53-bit adder are given in tables 1.2 and 1.3.

The number of gating stages required to implement this design can be seen by examining figure 1.10. Each square box in the diagram represents one gating stage. Of the 238 stages used in the whole adder register, note that *only 26* are used to create the auxiliary carry functions. The other 212 stages are needed irrespective of how the carry digits are formed, because they comprise 53 sets of 4 stages for the augend, addend, carry, and sum digits.

TABLE 1.2. *Auxiliary carry functions*

F_k represents $(A_k+B_k)(A_k+A_{k-1})(A_k+B_{k-1})(B_k+A_{k-1})(B_k+B_{k-1})$.
D_k represents A_kB_k. R_k represents (A_k+B_k).

$$X_9 = F_9+R_9R_8D_7+R_9R_8R_7D_6 + R_9R_8R_7R_6D_5$$

$$X_{14} = F_{14}+R_{14}R_{13}D_{12}+R_{14}R_{13}R_{12}D_{11} + R_{14}R_{13}R_{12}R_{11}D_{10}$$

$$X_{18} = D_{18}+R_{18}D_{17}+R_{18}R_{17}D_{16} + R_{18}R_{17}R_{16}D_{15}$$

$$X_{25} = F_{25}+R_{25}R_{24}D_{23}+R_{25}R_{24}R_{23}D_{22} + R_{25}R_{24}R_{23}R_{22}D_{21}$$

$$X_{29} = D_{29}+R_{29}D_{28}+R_{29}R_{28}D_{27} + R_{29}R_{28}R_{27}D_{26}$$

$$X_{33} = D_{33}+R_{33}D_{32}+R_{33}R_{32}D_{31} + R_{33}R_{32}R_{31}D_{30}$$

$$X_{38} = F_{38}+R_{38}R_{37}D_{36}+R_{38}R_{37}R_{36}D_{35} + R_{38}R_{37}R_{36}R_{35}D_{34}$$

$$X_{43} = F_{43}+R_{43}R_{42}D_{41}+R_{43}R_{42}R_{41}D_{40} + R_{43}R_{42}R_{41}R_{40}D_{39}$$

$$X_{48} = F_{48}+R_{48}R_{47}D_{46}+R_{48}R_{47}R_{46}D_{45} + R_{48}R_{47}R_{46}R_{45}D_{44}$$

$$Z_{33} = X_{33}+Y_{33}X_{29}+Y_{33}Y_{29}X_{25}$$

$$Z_{38} = X_{38}+Y_{38}X_{33}+Y_{38}Y_{33}X_{29}+Y_{38}Y_{33}Y_{29}X_{25}$$

$$Z_{43} = X_{43}+Y_{43}X_{38}+Y_{43}Y_{38}X_{33} + Y_{43}Y_{38}Y_{33}(X_{29}+Y_{29})(X_{29}+X_{25})$$

$$Z_{48} = X_{48}+Y_{48}X_{43} + Y_{48}Y_{43}(X_{38}+Y_{38})(X_{38}+X_{33}) + Y_{48}Y_{43}Y_{38}Y_{33}(X_{29}+Y_{29})(X_{29}+X_{25})$$

$$Y_9 = R_9R_8R_7R_6R_5$$

$$Y_{14} = R_{14}R_{13}R_{12}R_{11}R_{10}$$

$$Y_{18} = R_{18}R_{17}R_{16}R_{15}$$

$$Y_{25} = R_{25}R_{24}R_{23}R_{22}R_{21}$$

$$Y_{29} = R_{29}R_{28}R_{27}R_{26}$$

$$Y_{33} = R_{33}R_{32}R_{31}R_{30}$$

$$Y_{38} = R_{38}R_{37}R_{36}R_{35}R_{34}$$

$$Y_{43} = R_{43}R_{42}R_{41}R_{40}R_{39}$$

$$Y_{48} = R_{48}R_{47}R_{46}R_{45}R_{44}$$

$$W_{33} = Y_{33}Y_{29}Y_{25}$$

$$W_{38} = Y_{38}Y_{33}Y_{29}Y_{25}$$

$$W_{43} = Y_{43}Y_{38}Y_{33}Y_{29}Y_{25}$$

$$W_{48} = Y_{48}Y_{43}Y_{38}Y_{33}Y_{29}Y_{25}$$

TABLE 1.3. *Carry functions*

F_k represents $(A_k+B_k)(A_k+A_{k-1})(A_k+B_{k-1})(B_k+A_{k-1})(B_k+B_{k-1})$. D_k represents A_kB_k. R_k represents (A_k+B_k).

$$C_1 = D_1 + R_1C_0$$
$$C_2 = D_2 + R_2D_1 + R_2R_1C_0$$
$$C_3 = D_3 + R_3D_2 + R_3R_2D_1 + R_3R_2R_1C_0$$
$$C_4 = D_4 + R_4D_3 + R_4R_3D_2 + R_4R_3R_2R_1(A_1 + C_0)(B_1 + C_0)$$
$$C_5 = D_5 + R_5C_4$$
$$C_6 = D_6 + R_6D_5 + R_6R_5C_4$$
$$C_7 = D_7 + R_7D_6 + R_7R_6D_5 + R_7R_6R_5C_4$$
$$C_8 = F_8 + R_8R_7D_6 + R_8R_7R_6D_5 + R_8R_7R_6R_5C_4$$
$$C_9 = X_9 + Y_9C_4$$
$$C_{10} = D_{10} + R_{10}(X_9 + Y_9)(X_9 + C_4)$$
$$C_{11} = D_{11} + R_{11}D_{10} + R_{11}R_{10}(X_9 + Y_9)(X_9 + C_4)$$
$$C_{12} = D_{12} + R_{12}D_{11} + R_{12}R_{11}D_{10} + R_{12}R_{11}R_{10}(X_9 + Y_9)(X_9 + C_4)$$
$$C_{13} = F_{13} + R_{13}R_{12}D_{11} + R_{13}R_{12}R_{11}D_{10} + R_{13}R_{12}R_{11}R_{10}(X_9 + Y_9)(X_9 + C_4)$$
$$C_{14} = X_{14} + Y_{14}X_9 + Y_{14}Y_9C_4$$
$$C_{15} = D_{15} + R_{15}X_{14} + R_{15}Y_{14}(X_9 + Y_9)(X_9 + C_4)$$
$$C_{16} = D_{16} + R_{16}D_{15} + R_{16}R_{15}X_{14} + R_{16}R_{15}Y_{14}(X_9 + Y_9)(X_9 + C_4)$$
$$C_{17} = F_{17} + R_{17}R_{16}D_{15} + R_{17}R_{16}R_{15}X_{14} + R_{17}R_{16}R_{15}Y_{14}(X_9 + Y_9)(X_9 + C_4)$$
$$C_{18} = X_{18} + Y_{18}X_{14} + Y_{18}Y_{14}X_9 + Y_{18}Y_{14}Y_9C_4$$
$$C_{19} = D_{19} + R_{19}(X_{18} + Y_{18})(X_{18} + X_{14}) + R_{19}Y_{18}Y_{14}(X_9 + Y_9)(X_9 + C_4)$$
$$C_{20} = D_{20} + R_{20}D_{19} + R_{20}R_{19}(X_{18} + Y_{18})(X_{18} + X_{14}) + R_{20}R_{19}Y_{18}Y_{14}(X_9 + Y_9)(X_9 + C_4)$$
$$C_{21} = D_{21} + R_{21}C_{20}$$
$$C_{22} = D_{22} + R_{22}D_{21} + R_{22}R_{21}C_{20}$$
$$C_{23} = D_{23} + R_{23}D_{22} + R_{23}R_{22}D_{21} + R_{23}R_{22}R_{21}C_{20}$$
$$C_{24} = F_{24} + R_{24}R_{23}D_{22} + R_{24}R_{23}R_{22}D_{21} + R_{24}R_{23}R_{22}R_{21}C_{20}$$
$$C_{25} = X_{25} + Y_{25}C_{20}$$
$$C_{26} = D_{26} + R_{26}X_{25} + R_{26}Y_{25}C_{20}$$
$$C_{27} = D_{27} + R_{27}D_{26} + R_{27}R_{26}X_{25} + R_{27}R_{26}Y_{25}C_{20}$$
$$C_{28} = F_{28} + R_{28}R_{27}D_{26} + R_{28}R_{27}R_{26}X_{25} + R_{28}R_{27}R_{26}Y_{25}C_{20}$$
$$C_{29} = X_{29} + Y_{29}X_{25} + Y_{29}Y_{25}C_{20}$$
$$C_{30} = D_{30} + R_{30}(X_{29} + Y_{29})(X_{29} + X_{25}) + R_{30}Y_{29}Y_{25}C_{20}$$
$$C_{31} = D_{31} + R_{31}D_{30} + R_{31}R_{30}(X_{29} + Y_{29})(X_{29} + X_{25}) + R_{31}R_{30}Y_{29}Y_{25}C_{20}$$
$$C_{32} = F_{32} + R_{32}R_{31}D_{30} + R_{32}R_{31}R_{30}(X_{29} + Y_{29})(X_{29} + X_{25}) + R_{32}R_{31}R_{30}Y_{29}Y_{25}C_{20}$$
$$C_{33} = Z_{33} + W_{33}C_{20}$$
$$C_{34} = D_{34} + R_{34}(Z_{33} + W_{33})(Z_{33} + C_{20})$$
$$C_{35} = D_{35} + R_{35}D_{34} + R_{35}R_{34}(Z_{33} + W_{33})(Z_{33} + C_{20})$$
$$C_{36} = D_{36} + R_{36}D_{35} + R_{36}R_{35}D_{34} + R_{36}R_{35}R_{34}(Z_{33} + W_{33})(Z_{33} + C_{20})$$
$$C_{37} = F_{37} + R_{37}R_{36}D_{35} + R_{37}R_{36}R_{35}D_{34} + R_{37}R_{36}R_{35}R_{34}(Z_{33} + W_{33})(Z_{33} + C_{20})$$
$$C_{38} = Z_{38} + W_{38}C_{20}$$
$$C_{39} = D_{39} + R_{39}(Z_{38} + W_{38})(Z_{38} + C_{20})$$
$$C_{40} = D_{40} + R_{40}D_{39} + R_{40}R_{39}(Z_{38} + W_{38})(Z_{38} + C_{20})$$
$$C_{41} = D_{41} + R_{41}D_{40} + R_{41}R_{40}D_{39} + R_{41}R_{40}R_{39}(Z_{38} + W_{38})(Z_{38} + C_{20})$$
$$C_{42} = F_{42} + R_{42}R_{41}D_{40} + R_{42}R_{41}R_{40}D_{39} + R_{42}R_{41}R_{40}R_{39}(Z_{38} + W_{38})(Z_{38} + C_{20})$$
$$C_{43} = Z_{43} + W_{43}C_{20}$$
$$C_{44} = D_{44} + R_{44}(Z_{43} + W_{43})(Z_{43} + C_{20})$$
$$C_{45} = D_{45} + R_{45}D_{44} + R_{45}R_{44}(Z_{43} + W_{43})(Z_{43} + C_{20})$$
$$C_{46} = D_{46} + R_{46}D_{45} + R_{46}R_{45}D_{44} + R_{46}R_{45}R_{44}(Z_{43} + W_{43})(Z_{43} + C_{20})$$
$$C_{47} = F_{47} + R_{47}R_{46}D_{45} + R_{47}R_{46}R_{45}D_{44} + R_{47}R_{46}R_{45}R_{44}(Z_{43} + W_{43})(Z_{43} + C_{20})$$
$$C_{48} = Z_{48} + W_{48}C_{20}$$
$$C_{49} = D_{49} + R_{49}(Z_{48} + W_{48})(Z_{48} + C_{20})$$
$$C_{50} = D_{50} + R_{50}D_{49} + R_{50}R_{49}(Z_{48} + W_{48})(Z_{48} + C_{20})$$
$$C_{51} = D_{51} + R_{51}D_{50} + R_{51}R_{50}D_{49} + R_{51}R_{50}R_{49}(Z_{48} + W_{48})(Z_{48} + C_{20})$$
$$C_{52} = F_{52} + R_{52}R_{51}D_{50} + R_{52}R_{51}R_{50}D_{49} + R_{52}R_{51}R_{50}R_{49}(Z_{48} + W_{48})(Z_{48} + C_{20})$$

As indicated previously, five clock phases are occupied, starting with the input digits and ending with the sum digits. As the adder is to be used for multiplications and divisions in a repetitive fashion requiring the recirculation of the sum digits back into one of the inputs with appropriate shifts, the clock pulses must occur in five phases to allow an addition cycle to be completed in 1 μsec.

The top line of table 1.4 gives some statistics on the number of components required for the adder represented in figure 1.10. Two other slightly different versions have been worked out in which fewer gates need to be driven by the most heavily loaded tube. As the table shows, these variations also require different proportions of components. Approximately 10,000 germanium diodes are required in each of these versions.

TABLE 1.4. *Number of components required*

Maximum load [1]	Number of stages	Tubes	Delay lines
			μsec
25	238	238	300
19	253	253	250
14	285	285	150

[1] Unit of load=one gate-load.

6. References

[1] A. W. Holt, An experimental rapid access memory using diodes and capacitors, Proc. Assoc. for Computing Machinery Conference, p. 133–142 (December 1952).

[2] R. J. Slutz, A. W. Holt, R. P. Witt, D. C. Friedman, High-speed memory developments at the National Bureau of Standards, NBS Circ. 551, p. 93–108 (January 1955).

[3] S. Greenwald, R. D. Haueter, S. N. Alexander, SEAC, Proc. Inst. Radio Engrs. **41,** p. 1300–1313 (October 1953). See also, NBS Circ. 551, p. 5–26 (January 1955).

[4] A. L. Leiner, S. N. Alexander, System organization of the DYSEAC, Trans. Inst. Radio Engrs.-Prof. Group Electron. Computers, **EC–3,** No. 1 (March 1954).

[5] A. L. Leiner, W. A. Notz, J. L. Smith, A. Weinberger, System design of the SEAC and DYSEAC, Trans. Inst. Radio Engrs.-Prof. Group Electron. Computers, **EC–3,** No. 2 (June 1954).

[6] R. D. Elbourn, R. P. Witt, Dynamic circuit techniques used in SEAC and DYSEAC, Proc. Inst. Radio Engrs. **41,** p. 1380–1387 (October 1953). See also, Trans. Inst. Radio Engrs.-Prof. Group Electron. Computers, **EC–2,** No. 1 (March 1953), and NBS Circ. 551, p. 27–38 (January 1955).

[7] R. K. Richards, Arithmetic operations in digital computers, p. 26–50 (D. Van Nostrand Co., Inc., New York, N. Y., 1955).

2. Shortcut Multiplication for Binary Digital Computers

J. L. Smith and A. Weinberger

1. Introduction

The usual method of accomplishing automatic multiplication in binary digital computers consists of repeated additions of the multiplicand (the partial product) to the partial-product sum under the control of successive single digits of the multiplier beginning at the least-significant end. A one-position shift accompanies each addition cycle, either the partial-product sum to the right, or the multiplicand to the left. The process is somewhat analogous to the pencil-and-paper method of performing decimal multiplication, except that instead of storing all the partial products separately and adding them together as the final step, a running sum of the partial products is maintained in an accumulator. This method requires as many add-and-shift cycles as there are multiplier digits. The accumulator is cleared initially. During the first addition cycle, if the least-significant digit of the multiplier is a 1, the multiplicand is added to the contents of the accumulator, whereas if it is a 0, nothing is added. In either case the contents of the accumulator and of the multiplier register are shifted one position to the right. Similarly, during the second addition cycle, the next more significant digit of the multiplier determines whether the multiplicand is or is not added to the contents of the accumulator. Again, in either case, the contents of the accumulator and of the multiplier register are shifted one position to the right. After each multiplier digit in turn has controlled the addition of the multiplicand to the contents of the accumulator (with the one-position shift for each cycle), the multiplication operation is complete, and the final product is available in the accumulator.

To complete the description of the multiplication process, it should be pointed out that the length of the final product (assuming full significance) is generally equal to the sum of the lengths of the multiplicand and multiplier. The accumulator must therefore be of that length if it is to store the complete product. Actually, the product can be accumulated partly in the accumulator (major product) and partly in the multiplier register (minor product). Each digit of the minor product (the least significant one in the accumulator) obtained during any addition cycle is stored in the extreme left-hand position of the multiplier register, which is vacated during each shift of the multiplier-register contents.

2. Basis for Shortcut Multiplication

Whenever any multiplier digit is a 0 and nothing is to be added to the partial-product sum, it is obvious that that addition cycle could be omitted entirely, and that only a shift need be performed. A sequence of zeros would allow omitting as many addition cycles as there were zeros. The idea will now be developed that if a sequence of 1's occurs in the multiplier, cycles corresponding to these 1's may also be omitted in much the same way, if subtraction of the multiplicand from the partial-product sum is provided for.

Consider a string of n consecutive 1's in the multiplier. The placement of the binary point is arbitrary so far as this argument is concerned, and, for the sake of simplicity, let it be located to the right of the right-hand 1. Then the numerical value of just this string of n 1's is

$$2^{n-1}+2^{n-2}+2^{n-3}+\dots+2^1+2^0.$$

It is quite easily shown that this expression is equal to

$$2^n-2^0.$$

Therefore, in computing the partial products corresponding to this string of n 1's, one subtraction of the multiplicand for the right-hand 1 and one addition of the multiplicand for the 0 to the left of the string are sufficient. Such a sequence of 1's begun by a subtract cycle is referred to hereafter as a *subtract sequence*.

The fact that it is possible to compute a product correctly by omitting cycles that correspond to strings of 0's and to strings of 1's is the basis for this method of shortcut multiplication.

During the multiplication process, it is a fundamental requirement that the contents of the accumulator (and of the multiplier register) be shifted one digit position to the right for every digit of the multiplier, regardless of whether or not any of the cycles are omitted. Therefore, in order that the multiplication time be shortened to the fullest extent by virtue of omitting cycles of the operation, arrangements should be made for accomplishing multiple-position shifts as rapidly as possible. Ideally, the shifts that correspond to cycles to be omitted should be combined with the shift that corresponds to the last cycle actually performed. Thus, for example, if 3 cycles are to be omitted, 1 shift of 4 digit positions should be executed instead of 4 shifts of 1 digit position each.

From considerations of the logical gating required to accomplish any shift, it is evident that the number of different shift paths must be limited to only a few. But it will become evident that a point of diminishing return exists such that there is little need for exceeding the provision to omit either 1, 2, or 3 cycles at any one time.

There are two important features that must be incorporated in a device using a multiplication scheme such as this. These are illustrated in figure 2.1 where shifts of 1, 2, 3, and 4 positions are provided for (corresponding to the omission of 0, 1, 2, and 3 cycles). First, shift paths of the same lengths as are provided for the accumulator must also be provided for the multiplier register in order to have the proper multiplier digits at hand for controlling the next cycle. Second, provision must be made for examining a number of succeeding multiplier digits at once in order to determine how many of the succeeding cycles are to be omitted, and whether the multiplicand should be added or subtracted (or neither) during the current cycle.

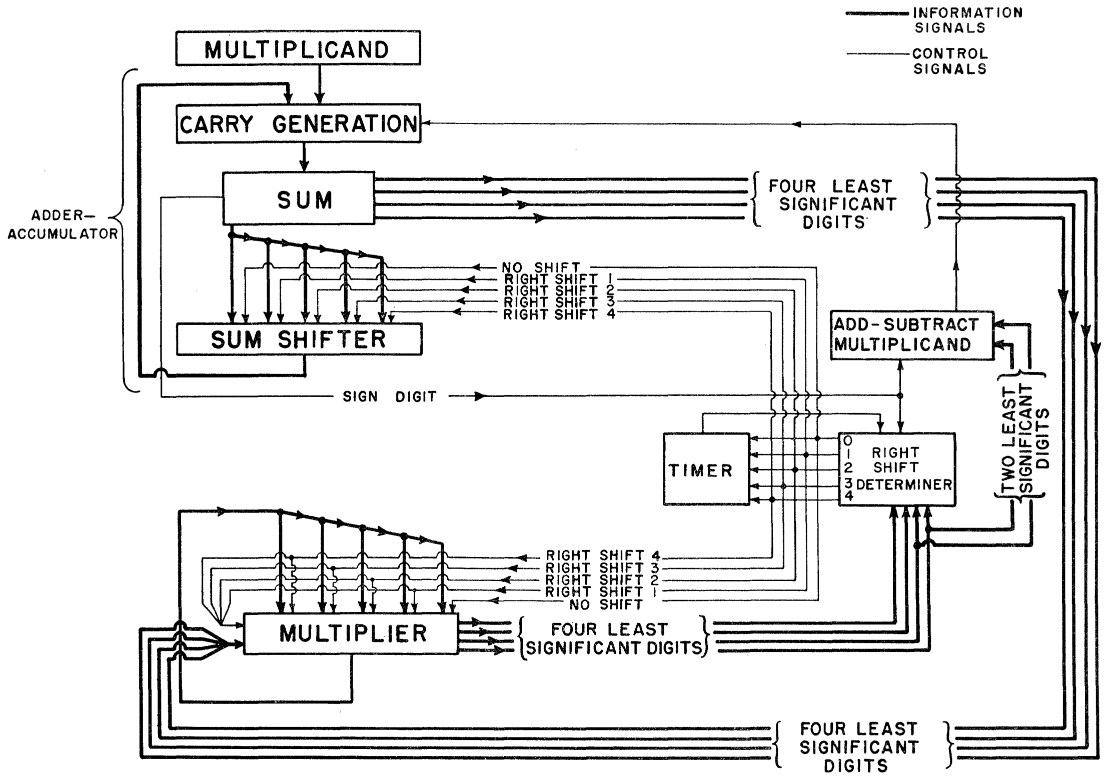

FIGURE 2.1. *Schematic diagram for shortcut multiplier.*

3. Rules for Governing Shortcut Multiplication

The previous paragraphs have outlined the basis for speeding up the multiplication process by omitting cycles of the operation. The means for governing such a procedure, in order to be mechanized, must be reduced to a set of well-defined rules.

Before the rules are set down in detail, an additional fact relating to a subtract sequence will be explained. For example, if the sequence is

$$11101011,$$

then its numerical value (assuming the sequence to be an integer) is

$$2^7+2^6+2^5+2^3+2^1+2^0,$$

which can also be expressed as

$$2^8-2^4-2^2-2^0.$$

This means that if a sequence of 1's in the multiplier is begun by subtracting for the right-hand 1, cycles for the rest of the 1's can be omitted, but cycles for the 0's become subtractions. Note that in the example the two 0's correspond in value to 2^2 and 2^4, and that these appear negatively in the last expression.

In the succeeding discussions, it is assumed that any number of cycles from 0 to n may be omitted at any one time, i. e., that the accumulator and multiplier registers can shift any number of positions from 1 to $n+1$. For the sake of brevity in the discussions to follow, an omitted or nonomitted cycle corresponding to any multiplier digit will be referred to simply as an omitted or nonomitted multiplier digit. The action proceeds from right to left through the sequence of multiplier digits.

Developing the rules for governing this method of shortcut multiplication falls naturally into two parts. First are the rules for determining whether to add or subtract the multiplicand or to do neither. This latter case, which constitutes a trivial cycle, is required only because the number of different shift paths, and hence the maximum amount of shift, is restricted for reasons of equipment economy; it is not always possible to omit as many cycles as could be omitted if there were no restriction on the maximum amount of shift. Second are the rules that determine how many digit positions the accumulator and multiplier registers are to be shifted. As stated before, the number of positions ranges from 1 to $n+1$.

Both sets of rules depend on (a) the current right-most digits of the multiplier after the shift for the previous cycle has been accomplished, and (b) the sign of the partial-product sum, i. e., whether the last nontrivial cycle was an add cycle or a subtract cycle.

3.1. Rules for Adding or Subtracting the Multiplicand

Case I. Partial-Product Sum Not Negative (Add Sequence):

If the partial-product sum is not negative, as it is in the initial state or as the result of adding for the last nontrivial cycle, the multiplicand is to be added if the current right-most digits of the multiplier are XX01. (Unspecified digits are immaterial.) It is not advantageous to begin a subtract sequence for a single 1 because an addition would be required during the very next cycle for the indicated 0. But for two or more 1's it is advantageous, so that if the current right-most digits are XX11, the multiplicand is to be subtracted from the partial-product sum.

If during an add sequence the current right-most multiplier digits are XXX0 (either an initial condition or a condition that might arise because a shift of more than $n+1$ digit positions is not permitted), the multiplicand is to be neither added nor subtracted, and the current cycle is a trivial one.

Case II. Partial-Product Sum Negative (Subtract Sequence):

If the partial-product sum is negative, as the result of subtracting for the last nontrivial cycle, the multiplicand is to be subtracted if the current right-most digits of the multiplier are XX10. In this instance, the 0 is treated like those in the subtract sequence described earlier in this section. It turns out to be advantageous to maintain a subtract sequence for isolated 0's. But for two or more consecutive 0's it is advantageous to terminate the subtract sequence, so that if the current right-most digits are XX00, the multiplicand is to be added to the partial-product sum.

If during a subtract sequence the current right-most multiplier digits are XXX1, a condition that might arise because a shift of more than $n+1$ digit positions is not permitted, the current cycle is a trivial one, and the multiplicand is to be neither added nor subtracted.

In table 2.1 are given practical add-subtract rules in a form that can easily be mechanized.

TABLE 2.1. *Rules for adding or subtracting*

Action	Required conditions [1]
Add multiplicand	01/+ or 00/−
Subtract multiplicand	11/+ or 10/−

[1] The digits shown are the current right-most digits of the multiplier. The sign indicates whether the partial-product sum for the last nontrivial cycle is positive or negative, i. e., whether the last nontrivial cycle was add or subtract.

3.2. Rules for Shifting

The general rules for omitting cycles, which follow easily from the precepts given in section 2, are as follows.

1. If an add sequence is initiated or maintained, i. e., if the cycle to be executed is an add cycle (or, in case it is trivial, if the last nontrivial one was an add cycle), the rule is: *consecutive 0's* in number from one to n immediately to the left of the right-most multiplier digit are to be omitted.

2. If a subtract sequence is initiated or maintained, i. e., if the cycle to be executed is a subtract cycle (or, in case it is trivial, if the last nontrivial one was a subtract cycle), the rule is: *consecutive 1's* in number from one to n immediately to the left of the right-most multiplier digit are to be omitted.

It should be clear that inasmuch as cycles are being executed continually during the multiplication process, the current right-most multiplier digit is never omitted.

As the rules for shifting follow directly from the rules for omitting cycles, the rules for shifting have as their criteria:

1. whether the current cycle initiates or maintains an add sequence, and
2. the number of consecutive 0's or 1's to the left of the current right-most multiplier digit.

If an add sequence is initiated or maintained, the number of consecutive 0's determines the shift; if a subtract sequence is initiated or maintained, the number of consecutive 1's determines the shift. In any event, only the sign of the partial-product sum resulting from the last nontrivial cycle and the current right-most digits of the multiplier are required as criteria for the rules, which are given in table 2.2. By comparing tables 2.1 and 2.2 it can be verified that the specific rules given in table 2.2 contain implicitly the conditions of initiating or maintaining an add or a subtract sequence.

As mentioned previously, it will be shown that there is little need for omitting more than three multiplier digits at any one time, and hence in the practical rules given, n is equal to 3. For n greater than 3, the rules given in table 2.2 can be extended in a manner that should be fairly obvious.

TABLE 2.2. *Rules for shifting*

Action	Required conditions[1]
Shift 1 digit position	10/+ or 01/−
Shift 2 digit positions	101/+ or 010/− or 100/± or 011/±
Shift 3 digit positions	1001/+ or 0110/− or 1000/± or 0111/±
Shift 4 digit positions	0001/+ or 1110/− or 0000/± or 1111/±

[1] The digits shown are the current right-most digits of the multiplier. The sign indicates whether the partial-product sum for the last nontrivial cycle is positive or negative, i. e., whether the last nontrivial cycle was add or subtract.

4. Effectiveness of Shortcut Multiplication

The effectiveness of this method of multiplication can be expressed in terms of percentages or ratios of the number of cycles actually required to the total number required without shortcutting. The smaller this ratio, the more effective is the method.

For the sake of clarity in the examples to be given, certain symbols to be placed atop each multiplier digit are defined as follows:

• = a multiplier digit for which nothing is added to or subtracted from the partial-product sum; a trivial cycle.
+ = a multiplier digit for which the multiplicand is added to the partial-product sum.
— = a multiplier digit for which the multiplicand is subtracted from the partial-product sum.
* = a multiplier digit corresponding to an omitted cycle.

The lower limit of effectiveness can be determined by finding repetitive sequences of multiplier digits such that the least number of digits can be omitted, obeying the general rules given in the last section. Thus, it can easily be verified that for $n=1$ an example of sequences that result in omitting the least number of cycles is

```
    ·*+·*-·*+·*-·*+·*-·*+·*-
. . . 000111000111000111000111,
```

and that the number of cycles actually required is two-thirds of the number of digits. Likewise, it can be seen that for $n>1$, examples of sequences that result in omitting the least number of cycles are

```
    *+*-*+*-*+*-*+*-*+*-
. . . 00110011001100110011,
```

and

```
    *+*+*+*+*+*+*+*+*+*+
. . . 01010101010101010101.
```

In these cases the number of cycles required is one-half of the number of digits. These sequences represent the worst possible conditions from the standpoint of speed. It is unlikely that they would occur very often in practice, although they do occur in the binary representation of certain decimal fractions, such as $\frac{1}{9}$, $\frac{1}{5}$, $\frac{1}{10}$, $\frac{1}{3}$, etc.

The upper limit of effectiveness is attained with strings of similar digits whose lengths are multiples of $n+1$. It is easy to show that the number of cycles actually required for this kind of sequence is $1/(n+1)$ times the number of digits. For $n=3$, an example of such a sequence is

```
    ***·***·***+***·***-***·
. . . 000000000000111111110000.
```

In this case, which represents the best possible conditions from the standpoint of speed, the number of cycles required is only one-fourth of the number of multiplier digits. There are, of course, many such "best" sequences, and it is likely that this kind of sequence, at least for parts of words, would occur rather often in practice.

From the practical standpoint, perhaps the most meaningful determination of effectiveness is the one for a purely random sequence of multiplier digits. A mathematical analysis [1] shows that the limiting value of the ratio of the number of cycles required (M) divided by the number of multiplier digits (m), as m increases without bound, is

$$\lim_{m=\infty} \frac{M}{m} = \frac{1}{3} \frac{1+2^{-(n+1)}}{1-2^{-(n+1)}}.$$

It is obvious that if n, the maximum number of cycles that may be omitted at once, also increases without bound, the limit of this ratio is one-third. For a practical value of m, say, 64, numerical calculation based on the results of a general mathematical treatment [2] shows that the ratio is greater than the limiting value by only about 0.01. The results of such numerical calculations are summarized in figure 2.2.

Figure 2.2 shows graphically for various values of n the ratio of the number of cycles actually required to compute a product to the number of digits in the multiplier. Besides the results for random sequences, there are also shown the results for best sequences and worst sequences. Although the random-digits points are constructed from computed values for multipliers of a particular length, namely, 64 binary digits, these results do not differ significantly from the results for somewhat shorter or longer words.

Figure 2.2 shows that if the maximum number of cycles that can be omitted at any one time is three, the average time to compute a product, using a large number of random multipliers, is about 40 percent of the usual multiplication time, all other things being equal, or that the speed factor due to shortcutting is about 2.5.

In practice, the multiplier digits are often not random, nor even apparently so. However, experience seems to indicate that departures from randomness will usually favor the sequences of similar digits that require fewer cycles rather than those special sequences that require more cycles. As a result, the average number of cycles required for multiplications will generally turn out in practice to be less than the theoretical number for random sequences obtained from figure 2.2. If the curve were plotted for the average number of cycles required for multiplier sequences actually occurring in practice, consisting of a mixture of random or pseudo-random digits and strings of similar digits, it would be found that this curve would lie between the curve for random sequences and the curve for the best sequences.

It is evident from an examination of figure 2.2 that, so far as random sequences are concerned, the number of cycles required to compute a product decreases but little as n is increased beyond three. This means that for random sequences there is little need for providing the means to omit more than three cycles at once. However, for nonrandom sequences such that the curve for the best sequences is applicable, it can be seen that making $n=7$ would halve the number of cycles required for $n=3$. Thus, for sequences consisting of a mixture of random digits and strings of similar digits, only the nonrandom portions would be significantly affected by increasing n.

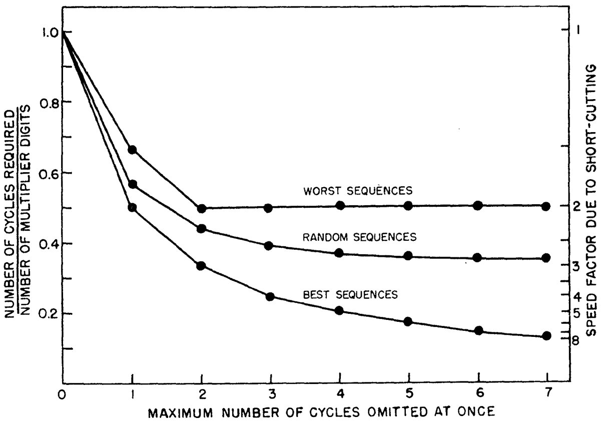

FIGURE 2.2. *Effectiveness of shortcut multiplication.*

5. Appendix 1. Speed for Infinite-Length Multiplier

The average rate at which digits of an infinitely long multiplier sequence are disposed of in digits per cycle is to be determined, under the assumptions that the multiplier sequence is purely random, and that the add-subtract and shift rules stated in the body of the paper apply. This average rate is a fairly good approximation to the speed factor achieved for random sequences of practical length.

[1] See Appendix 1.
[2] See Appendix 2.

The probabilities of shifting by various amounts from a nonomitted digit will now be derived for specific conditions. These probabilities follow directly from the shift rules stated in table 2.2. All the conditions, stipulated by the sign of the partial-product sum and the current nonomitted (right-most) digit of the sequence, result in only two sets of shift probabilities. These conditions are grouped according to the resulting shift probabilities and are designated as case 1 and case 2.

Case 1:

The pertinent specific conditions are (a) partial-product sum not negative and current right-hand (nonomitted) digit 1, or (b) partial-product sum negative and current right-hand digit 0.

Table 2.2 shows that under these conditions there can never be a shift of just *one* position. In order for there to be a shift of two positions, it is required that the table 2.2 states be 101/+, 011/+, 100/−, or 010/−. The number of different states containing each of conditions (a) and (b), namely two, is just one-half of the four pertinent combinations ending in 1/+ and 0/−, respectively.[3] Therefore, because random sequences are being considered, the probability of shifting two positions from a nonomitted digit is $\frac{1}{2}$ for case 1 conditions. By similar reasoning it can be seen that the probability of shifting three positions is $\frac{1}{4}$. Then by extension, and as the rules are similar for a shift of n and for a shift of $n+1$ positions, the following table (2.3) of shift probabilities for case 1, designated by V^j can be compiled. It should be borne in mind that n has the same definition as in the body of the paper, i. e., n is the maximum number of digits omitted at once. The amount of shift ranges from 1 to $n+1$.

TABLE 2.3. *Case 1 shift probabilities*

j	1	2	3	. . .	i	. . .	n	$n+1$
V^j	0	$\frac{1}{2}$	$\frac{1}{4}$	. . .	$2^{-(i-1)}$	. . .	$2^{-(n-1)}$	$2^{-(n-1)}$

Case 2:

The pertinent specific conditions are (a) partial-product sum not negative and current right-hand digit 0, or (b) partial-product sum negative and current right-hand digit 1.

Table 2.2 shows that in order for there to be a shift of one position, it is required that there be 10/+ or 01/−. But the number of these is just one-half of the pertinent combinations ending in either 0/+ or 1/−. Hence the probability of shifting one position is $\frac{1}{2}$ for case 2 conditions. By similar reasoning it can be seen that the probability of shifting two positions is $\frac{1}{4}$, and so on. Table 2.4, showing the case 2 shift probabilities, designated by W^j, is constructed in the same manner as table 2.3.

TABLE 2.4. *Case 2 shift probabilities*

j	1	2	3	. . .	i	. . .	n	$n+1$
Wj	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	. . .	2^{-i}	. . .	2^{-n}	2^{-n}

[3] For example, the four combinations ending in 1/+ are 001/+, 011/+, 101/+, and 111/+.

A fact regarding the combined effects of the add-substract rules of table 2.1 and the shift rules of table 2.2 that can easily be verified is that a shift of any amount less than $n+1$ *always* results in case 1 conditions, and that a shift of just $n+1$ positions results in case 1 conditions and case 2 conditions with equal probability. This latter statement is true because, so far as the rules are concerned, the right-hand digit after a shift of $n+1$ positions is completely unknown prior to the shift, and it can be a 1 or 0 with equal probability. It is also a fact, evident from an inspection of table 2.1, that the so-called trivial cycles, for which neither addition nor subtraction is done, are the very ones for which the case 2 conditions apply; conversely, cycles for which case 2 conditions apply are all trivial cycles.

The following argument, which evaluates the probabilities of occurrence of case 1 and case 2 conditions, is based on the assumption that the multiplier sequence is sufficiently long so that the probabilities for one cycle are identical to the probabilities for the preceding cycle.

Let Q be defined as the probability that case 2 conditions apply for any cycle. Then, clearly, $1-Q$ is the probability that case 1 conditions apply. Since it is noted in the preceding paragraph that case 2 results from a shift of $n+1$ positions with a probability of one-half, it follows that the probability of having had a shift of $n+1$ positions is $2Q$.

Now, from tables 2.3 and 2.4, it can be seen that the probability of shifting $n+1$ positions for case 1 is $2^{-(n-1)}$, and for case 2, 2^{-n}. Then $2Q$ can be evaluated by taking the weighted mean of the shift probabilities for the two cases.

$$2Q=(1-Q)2^{-(n-1)}+Q2^{-n}. \tag{1}$$

After some reduction, there is obtained

$$Q=\frac{2^{-(n-1)}}{2+2^{-n}}, \tag{2}$$

and

$$1-Q=\frac{2-2^{-n}}{2+2^{-n}}, \quad \text{for } n\geq 1. \tag{3}$$

Equations (2) and (3) are the expressions for the probabilities of having case 2 and case 1 conditions, respectively, after an infinite number of cycles.

The next step is to determine the average shift for case 1 and for case 2. This can be done from the information contained in tables 2.3 and 2.4. The average shift is simply the weighted mean of all the possible shifts for each case. Thus, if A is defined to be the average shift for case 1, we have

$$A=\sum_{i=2}^{n} i2^{-(i-1)}+(n+1)2^{-(n-1)}, \tag{4}$$

which by well-known methods is reduced to

$$A=3-2^{-(n-1)}, \quad \text{for } n \geq 1. \qquad (5)$$

Similarly, if B is defined to be the average shift for case 2, we have

$$B=\sum_{i=1}^{n} i2^{-i}+(n+1)2^{-n} \qquad (6)$$

$$=2-2^{-n}, \quad \text{for } n \geq 0. \qquad (7)$$

The average rate of shifting is the weighted mean of A and B, using for the weights $1-Q$ and Q, respectively. Thus the average rate of shifting is

$$L=(1-Q)A+QB, \qquad (8)$$

which, through the use of eq (2), (3), (5), and (7), becomes

$$L=3\frac{1-2^{-(n+1)}}{1+2^{-(n+1)}}, \quad \text{for } n \geq 1. \qquad (9)$$

The quantity L is the average amount of shift per cycle, i. e., the average number of digits disposed of in each cycle. The reciprocal of this is the average number of cycles required per digit. If the total number of cycles required to compute a product is M, and if the number of digits in the multiplier is m, then we have

$$\frac{1}{L}=\lim_{m=\infty}\frac{M}{m}=\frac{1}{3}\frac{1+2^{-(n+1)}}{1-2^{-(n+1)}}, \quad \text{for } n \geq 1. \qquad (10)$$

In table 2.5 are given the values of $1/L$ for various values of n. For the sake of completeness, the value for $n=0$ (the usual multiplication method) is included, which is unity.

TABLE 2.5. *Relative multiplication times*

n	0	1	2	3	4	5	6	. . .	∞
$1/L$	1	$\frac{5}{9}$	$\frac{3}{7}$	$\frac{17}{45}$	$\frac{11}{31}$	$\frac{65}{189}$	$\frac{43}{127}$	. . .	$\frac{1}{3}$

6. Appendix 2. Speed for Finite-Length Multiplier

The average number of cycles required to generate a product is to be determined under the assumptions that the multiplier is a purely random number of finite length and that the add-subtract rules and the shift rules stated in the body of this paper apply.

To anticipate the discussion, a quantity X_k is defined to be the probability that the kth digit of a sequence is not omitted—i. e., the probability that one cycle will be expended in passing over the kth digit. Then the total number of cycles expended in passing over m digits of an m-digit sequence is $\sum_{k=1}^{m} X_k$. It is the purpose in this appendix to derive an expression for this quantity in terms of known constants and parameters.

To begin the discussion, the definitions of certain terms are given.

R_k = the probability of omitting the kth digit.

X_k = the probability of not omitting the kth digit.

P_k = the probability that the kth digit corresponds to an add or a subtract cycle.

Q_k = the probability that the kth digit corresponds to a trivial cycle.

S_k^j = the probability of shifting j places from the kth digit, provided the kth digit is not omitted.

$X_kS_k^j$ = the actual probability of shifting j places from the kth digit.

n = the maximum number of digits that may be omitted at once, and the amount of shift may range from 1 to $n+1$.

Note that the superscript on S is not to be interpreted as an exponent.

From these definitions some rather obvious relations can be derived.

$$X_k+R_k=1 \qquad (1)$$

$$P_k+Q_k=X_k \qquad (2)$$

$$\sum_{j=1}^{n+1} S_k^j=1, \qquad (3)$$

where $n+1$ is the maximum amount of shift.

If the digit k is not omitted, then during the previous cycle when the nonomitted digit was, say, $k-p$, there must have been a shift of exactly p positions. The probability that the kth digit is not omitted is therefore given by

$$X_k=\sum_{p=1}^{n+1} X_{k-p}S_{k-p}^p. \qquad (4)$$

In appendix 1 it is shown that under all circumstances there are only two sets of probabilities for shifting various amounts, designated case 1 and case 2. These shift probabilities are given in tables 2.3 and 2.4 and are labeled V^j and W^j, respectively. For case 1, V^j is the probability that a shift of j places will take place, and for case 2, W^j has a similar definition. Note that the superscripts on V and W are not to be interpreted as exponents.

It is also brought out in appendix 1 that the trivial cycles are the ones for which the case 2 shift probabilities apply, and conversely. This means then that the probability that case 2 conditions apply for any digit k equals Q_k, and that the probability that case 1 conditions apply equals P_k. Therefore, the total probability of shifting j places from the kth digit is

$$X_k S_k^j = V^j P_k + W^j Q_k. \tag{5}$$

Now, the probability of occurrence of case 2 conditions, i. e., Q_k, is one-half the probability that there was a shift of $n+1$ positions during the previous cycle (noted in appendix 1):

$$Q_k = \tfrac{1}{2} X_{k-(n+1)} S_{k-(n+1)}^{n+1}. \tag{6}$$

Then, through the use of eq (5) and (6) and the values of V^{n+1} and W^{n+1} from tables 2.3 and 2.4, there is obtained

$$Q_k = \tfrac{1}{2}[Q_{k-(n+1)} \cdot 2^{-n} + P_{k-(n+1)} \cdot 2^{-(n-1)}], \tag{7}$$

which becomes

$$Q_k = 2^{-n} P_{k-(n+1)} + 2^{-(n+1)} Q_{k-(n+1)}. \tag{8}$$

If the digit k is omitted during the cycle corresponding to the nonomitted digit $k-p$, then there must be a shift of any amount greater than p during this cycle. For example, if the $(k-4)$th digit is the nonomitted digit, then the shift must be either 5, 6, 7, . . ., or $n+1$, and in this instance the probability of omitting the kth digit is

$$X_{k-4} \sum_{i=5}^{n+1} S_{k-4}^i.$$

If, for the omitted digit k, all possible nonomitted digits to the right, $k-p$, are considered, the probability of omitting this kth digit is the sum of such expressions for all possible p.

$$R_k = \sum_{p=1}^{n} \left[X_{k-p} \sum_{j=p+1}^{n+1} S_{k-p}^j \right]. \tag{9}$$

If use is made of

$$\sum_{j=1}^{n+1} S_k^j = 1, \tag{3}$$

and if the term for $p=1$ is separated, eq (9) becomes

$$R_k = X_{k-1}(1 - S_{k-1}^1) + \sum_{p=2}^{n} \left[X_{k-p} \sum_{j=p+1}^{n+1} S_{k-p}^j \right]. \tag{10}$$

From eq (5) can be derived

$$S_k^j = \frac{P_k}{X_k} V^j + \frac{Q_k}{X_k} W^j, \tag{11}$$

from which is easily obtained

$$\sum_{j=p+1}^{n+1} S_k^j = \frac{P_k}{X_k} \sum_{j=p+1}^{n+1} V^j + \frac{Q_k}{X_k} \sum_{j=p+1}^{n+1} W^j, \tag{12}$$

which by virtue of the character of V and W (see tables 2.3 and 2.4) is

$$\sum_{j=p+1}^{n+1} S_k^j = \frac{P_k}{X_k} V^p + \frac{Q_k}{X_k} W^p \tag{13}$$

$$= S_k^p, \qquad \text{for } 2 \leq p \leq n. \tag{14}$$

Therefore, eq (10) becomes

$$R_k = X_{k-1}(1 - S_{k-1}^1) + \sum_{p=2}^{n} X_{k-p} S_{k-p}^p, \tag{15}$$

and through the use of eq (4) there is obtained

$$R_k = X_{k-1} - X_{k-1} S_{k-1}^1 + X_k - X_{k-1} S_{k-1}^1 - X_{k-(n+1)} S_{k-(n+1)}^{n+1}. \tag{16}$$

After some reductions, using eq (5) and (6), eq (16) becomes

$$R_k = X_k - Q_k + X_{k-1} - Q_{k-1} - Q_k, \tag{17}$$

which, by virtue of eq (2), is

$$R_k = P_k + P_{k-1} - Q_k. \tag{18}$$

But because of

$$X_k + R_k = 1, \tag{1}$$

eq (18) is seen to be

$$1 - X_k = P_k + P_{k-1} - Q_k, \tag{19}$$

which after further reductions, using eq (2), becomes

$$P_k = \tfrac{1}{2}(1 - P_{k-1}). \tag{20}$$

Equation (2), with the aid of eq (20) and (8) is now to be summed to yield a closed expression in terms of known constants and parameters. That is to say, it is desired to find an expression for

$$\sum_{k=1}^{m} X_k = \sum_{k=1}^{m} P_k + \sum_{k=1}^{m} Q_k. \tag{21}$$

The summation of P_k can be determined readily through the use of eq (20) if the initial condition is stipulated. The initial condition is that the first digit is never omitted; i. e., $X_1 = 1$. Now as the zero-th digit is nonexistent, P_0 is not defined.

Therefore, eq (22) follows readily from eq (20) and (2).

$$P_1 = Q_1 = \tfrac{1}{2}. \tag{22}$$

If eq (20) is expanded until P_1 is reached, there is obtained

$$P_k = 2^{-1} - 2^{-2} + 2^{-3} - \ldots - (-2)^{-i} - \ldots - (-2)^{-k}, \tag{23}$$

from which by well-known techniques is obtained

$$P_k = \tfrac{1}{3}(1-(-2)^{-k}), \qquad \text{for } k \geq 1. \tag{24}$$

The summation of P_k is then

$$\begin{aligned}\sum_{k=1}^{m} P_k &= \tfrac{1}{3}m - \tfrac{1}{3}\sum_{k=1}^{m}(-2)^k \\ &= \tfrac{1}{3}m + \tfrac{1}{3}[2^{-1} - 2^{-2} + \ldots - (-2)^m] \\ &= \tfrac{1}{9}[3m+1-(-2)^{-m}], \qquad \text{for } m \geq 1.\end{aligned} \tag{25}$$

To obtain the summation of Q_k, eq (8) will be expanded in a straightforward manner until the first Q_z is reached for which $z \leq n+1$.

$$\begin{aligned}Q_k = {} & 2^{-n} P_{k-(n+1)} + 2^{-(n+1)} \cdot 2^{-n} P_{k-2(n+1)} + \ldots \\ & + 2^{-(i-1)(n+1)} \cdot 2^{-n} P_{k-i(n+1)} + \ldots \\ & + 2^{-(s-1)(n+1)} \cdot 2^{-n} P_{k-s(n+1)} + 2^{-s(n+1)} Q_{k-s(n+1)},\end{aligned} \tag{26}$$

where s is the integral number of times $k-1$ is divisible by $n+1$. It is important to note that $Q_z = 0$ for $2 \leq z \leq n+1$. This is true because Q_z is one-half the probability of having shifted $n+1$ positions during the previous cycle, which is clearly impossible for $z \leq n+1$. As the value of Q_1 is not 0, (eq (22)), the expansion of Q_k may have only Q_1 in the Q-term. The summation of Q_k can now be written as

$$\begin{aligned}\sum_{k=1}^{m} Q_k = {} & 2^{-n} \sum_{k=1}^{m-(n+1)} P_k + 2^{-(n+1)} \cdot 2^{-n} \sum_{k=1}^{m-2(n+1)} P_k + \ldots \\ & + 2^{-(i-1)(n+1)} \cdot 2^{-n} \sum_{k=1}^{m-i(n+1)} P_k + \ldots \\ & + 2^{-(t-1)(n+1)} \cdot 2^{-n} \sum_{k=1}^{m-t(n+1)} P_k \\ & + Q_1 \sum_{i=0}^{t} 2^{-i(n+1)},\end{aligned} \tag{27}$$

where t is the integral number of times $m-1$ is divisible by $n+1$. This in turn can be written as

$$\sum_{k=1}^{m} Q_k = 2 \sum_{i=1}^{t} \left[2^{-i(n+1)} \sum_{k=1}^{m-i(n+1)} P_k \right] + 2^{-1} \sum_{i=0}^{t} 2^{-i(n+1)}. \tag{28}$$

Substituting into this the expression for $\sum P_k$, eq (25), and using eq (21), there is obtained

$$\begin{aligned}\sum_{k=1}^{m} X_k = {} & \tfrac{1}{9}[3m+1-(-2)^{-m}] + \tfrac{2}{9}\sum_{i=1}^{t} 2^{-i(n+1)} \\ & \times [3m - 3i(n+1) + 1 - (-2)^{i(n+1)-m}] \\ & + \tfrac{1}{2}\sum_{i=0}^{t} 2^{-i(n+1)}, \qquad \text{for } m \geq 1 \text{ and } n \geq 0,\end{aligned} \tag{29}$$

$$\begin{aligned}\sum_{k=1}^{m} X_k = {} & \frac{1}{9}[3m+1-(-2)^{-m}] \\ & + \frac{2}{9}(3m+1)\frac{2^{-(n+1)} - 2^{-(t+1)(n+1)}}{1-2^{-(n+1)}} \\ & - \frac{2}{3}(n+1)\frac{2^{-(n+1)}(1-2^{-t(n+1)})}{(1-2^{-(n+1)})^2} \\ & + \frac{2}{3}(n+1)t\,\frac{2^{-(t+1)(n+1)}}{1-2^{-(n+1)}} \\ & - \frac{2}{9}2^{-m}\sum_{i=1}^{t}(-1)^{m-i(n+1)} + \frac{1-2^{-(t+1)(n+1)}}{2\,(1-2^{-(n+1)})},\end{aligned} \tag{30}$$

where $m \geq 1$, $n \geq 0$, and where t is the integral number of times $m-1$ is divisible by $n+1$.

The formula for the number of cycles required to generate a product is complete except for the terminal effect. An extra cycle is needed at the end to terminate if the last nontrivial cycle was a subtract and if the last digit is a 1. According to the rules, a subtract cycle is initiated by 11, and it is terminated by 00. The probability that the last nontrivial cycle is a subtract, to be called Z, is the probability that there is 11 somewhere in the sequence that is not followed by 00. Some examples follow, in which only the left-hand portions of the sequences are shown.

(a) 11. . . . . .

(b) 1011. . . . .

(c) 101011. . . .

(d) 10101011. . .

.
.
.

It should be noted that these sequences are mutually exclusive. The probability of occurrence of (a) is 2^{-2}; of (b), 2^{-4}; of (c), 2^{-6}; of (d) 2^{-8}; etc. If the word is m digits long, the last sequence of this kind has a probability of occurrence 2^{-m} if m is even, or $2^{-(m-1)}$ if m is odd. This latter fact results from the requirement that all the sequences be mutually exclusive. Now Z is the sum of all these probabilities.

$$Z=\sum_{k=1}^{\frac{1}{2}m} 2^{-2k} \qquad \text{if } m \text{ is even, or} \qquad (31)$$

$$Z=\sum_{k=1}^{\frac{1}{2}(m-1)} 2^{-2k} \qquad \text{if } m \text{ is odd.} \qquad (32)$$

These reduce to

$$Z=\tfrac{1}{3}(1-2^{-m}) \qquad \text{if } m \text{ is even, or} \qquad (33)$$

$$Z=\tfrac{1}{3}(1-2^{1-m}) \qquad \text{if } m \text{ is odd.} \qquad (34)$$

The total number of cycles required, to be called M, is therefore

$$M=\sum_{k=1}^{m} X_k+Z \qquad (35)$$

Figure 2.2 contains a plot of the values of M/m for $m=64$ and for various values of n. A discussion of figure 2.2 is in the body of the paper.

U.S. GOVERNMENT PRINTING OFFICE: 1960

DEPARTMENT OF COMMERCE
UNITED STATES OF AMERICA