TN 00.468

SUMMARY OF BINARY ADDERS

Frank B. Hartman

IBM Technical Note

October 20, 1960 TN 00.468

SUMMARY OF BINARY ADDERS

by

Frank B. Hartman

This note consists of a section from the Final Report for contract No. AF 19(604) - 4152.

NOT TO BE DISTRIBUTED TO IBM LOCATIONS OUTSIDE CONTINENTAL USA.

ABSTRACT

Drawing upon several sources, a number of binary adder structures are described. These include serial adder; carry-save adder; asynchronous adder; linear propagate adder; and simultaneous, linear-grouped, grouped-simultaneous, simultaneous-grouped, simultaneous-simultaneous, etc. structures, together with estimates of the time delays involved. These time delays are based on the assumptions that (1) all combinational blocks have the same delay, (2) all flip-flops have the same resolution time, and (3) the two possible values of each operand bit are equally probable.

Product Development Laboratory, Data Systems Division
International Business Machines Corporation, Poughkeepsie, New York

SUMMARY OF BINARY ADDERS

by
F. B. Hartman

The circuit technology used in the construction of a data processing system forms an important factor in determining its operating characteristics. The set of basic building blocks provided can be arranged in many forms to realize the functional requirements of the system. In designing for high speed operation, it is the duty of the circuit designer to provide fast circuits. This will guarantee a generally fast system. On the other hand, it is the duty of the logic designer to specify from among numerous alternatives those arrangements of the circuits that will facilitate the realization of high system speed consistent with reasonable cost.

System considerations often dictate the use of binary arithmetic, particularly when the system has to solve scientific problems. Speed and accuracy considerations dictate that the binary digits be handled in parallel as much as possible and with a sufficiently long word length. Since all arithmetic operations reduce ultimately to addition, the design of a fast parallel binary adder has a particularly important bearing on achieving high speed in the system containing it.

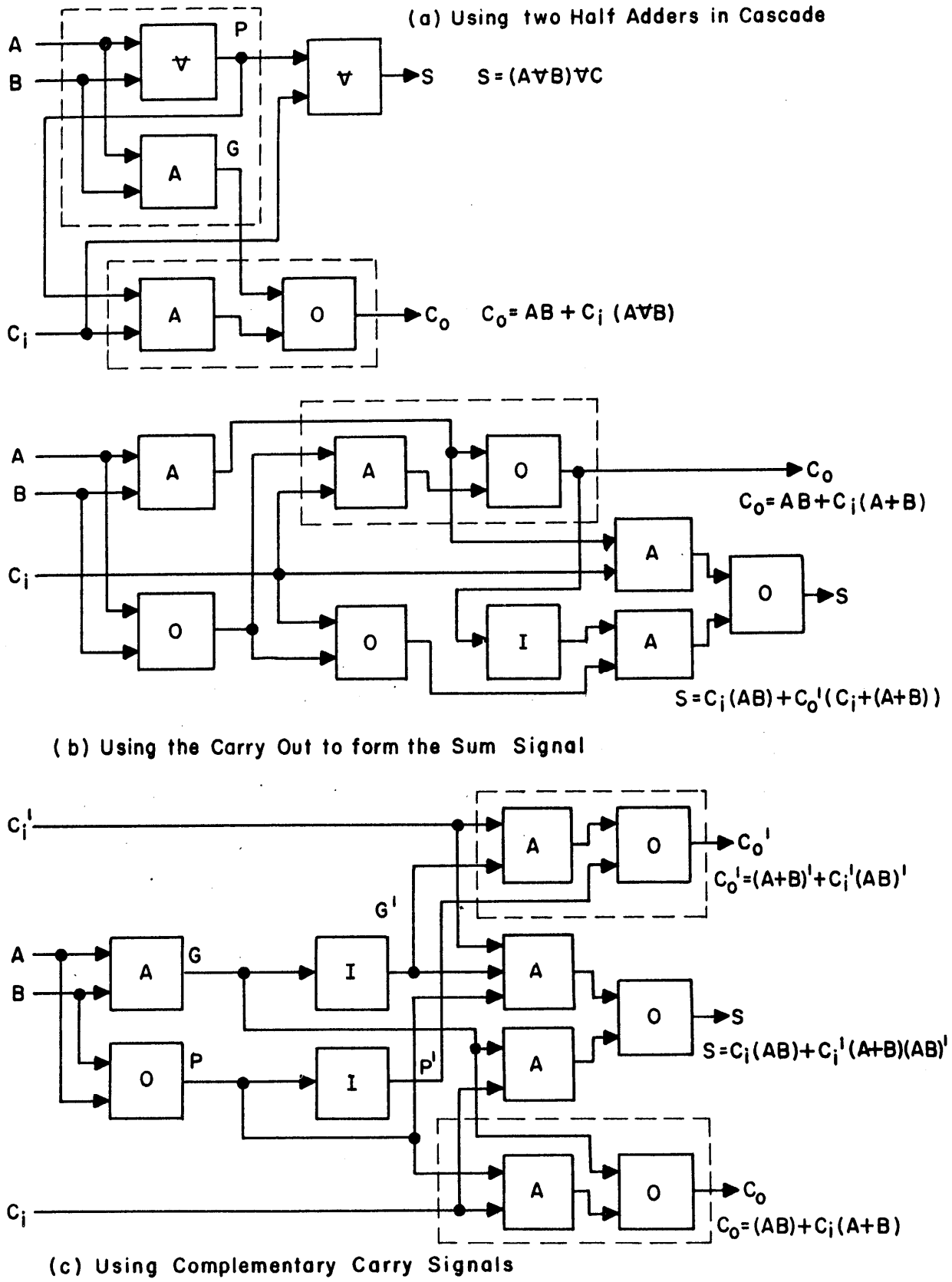
Analyses of binary adders have appeared extensively in the literature. Richard's book[1] contains a general survey. Campbell and Rosser[2] discuss the carry transmission problem and means for speeding up adders by various strategies. Gilchrist, Pomerene and Wong[3] discuss details of an asynchronous adder. Mercer[4] discusses a form of "carry save" adder in conjunction with a micro-program computer structure.

The purpose of this section consists in discussing the various forms of binary adders from a general viewpoint, including a summary of the data from the above sources.

Forms of Binary Full Adders - As a base, the design of any binary adder begins with a consideration of the single-position binary full adder. The actual form used will depend upon the circuit technology involved, but in Figure 1, we have shown several representative forms.

The full adder of Figure 1a uses the exclusive OR block and can be considered as two half adders in cascade. A single half adder (shown within the upper rectangle) generates the sum and carry of the operand bits, A and B. The sum line (P) from this circuit is added to the "carry in" signal C_i in the other half adder. A carry out of either half adder or both will generate a "carry out" signal (C_o) for the full addition in the inclusive OR circuit. Attention is directed

Figure 1 FORMS OF BINARY FULL ADDERS



to the circuitry in the lower dotted rectangle which shows the method of generating the "carry out" signal. A "carry out" of unity can occur in two distinguishable ways. In the first case, if both operand bits A and B are unity, the "generate" line (G) is unity, which forces a "carry out" through the OR circuit. In this case, the full adder is said to "generate" a carry. In the second case, if A and B have opposite values, the lower AND circuit will be conditioned by the "propagate" line (P) to transfer the value of the "carry in" line directly to the output, since at this time the "generate" line (G) is down and the OR circuit looks like a direct connection to the other input. In this case the adder is said to "propagate" the "carry in." On the other hand, if both A and B have a zero value, the stage neither generates nor propagates a carry.

Analysis will show that, insofar as the generation of the "carry out" is concerned, the "propagate" signal can be taken as the inclusive OR rather than the exclusive OR. This arrangement is shown in Figure 1b, which shows also the generation of the sum in terms of the "carry out." This form has been used in commercial equipment and has the advantage that complementary input signals are not needed and but one inverter is used.

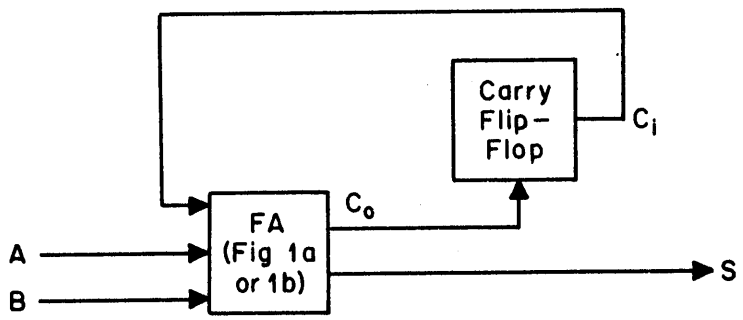
Figure 1c shows a special form of full adder in which the complements of the "propagate" and "generate" signals are generated internally, and used in conjunction with complementary "carry in" signals to generate the sum and complementary "carry out" signals. (If complementary "carry in" signals are used, complementary "carry out" signals will be needed in some applications).

Many other forms can be imagined, but unless complementary inputs for each of the three bits is provided, some form of inversion action will be required internally, and this implies some form of amplifier.

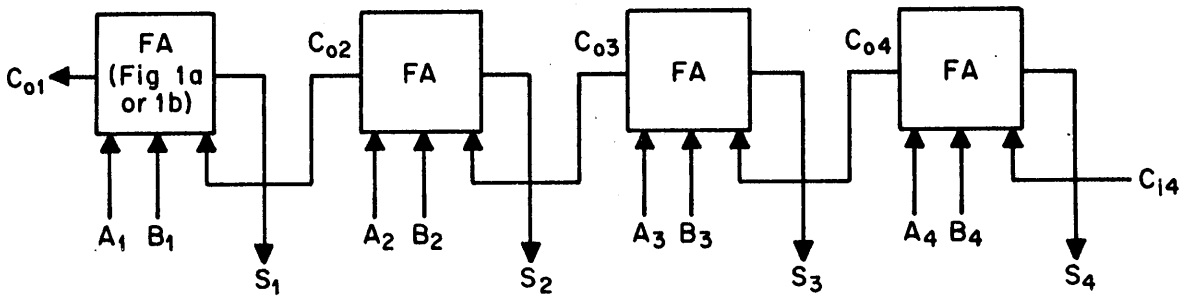
Basic Forms of Binary Adders - If the binary words consist of single digits, the problem reduces to the design of the binary full adder discussed above. For longer word lengths, several distinct types of adder have appeared. Figure 2a shows a serial binary adder which can add two binary words of any length one position at a time. Its distinguishing feature consists of the carry flip-flop which serves to store the "carry out" from one cycle of the addition in order to present it as the "carry in" of the next cycle. The operand bits must be presented serially, one lined-up pair per cycle, starting with the lowest-weight position. The sum bits are carried off in similar fashion to some storage device. Letting n represent the length of the operand in bits, t_1 the delay of the binary full adder in generating the sum or "carry out," whichever appears later, and t_3 the resolution time of the flip-flop (time required for entering the "carry out"), then the shortest cycle possible equals $t_1 + t_3$, and for the time T to add any pair of words we have

$$T = n(t_1 + t_3) \quad \text{(Serial Adder)} \qquad (1)$$

Figure 2 TWO BASIC FORMS OF ADDERS



(a) Serial Adder



(b) Linear-Propagate Adder

Figure 2b shows a parallel linear-propagate adder for adding four-bit words. Its characteristic feature consists in the connection of the "carry out" of one binary full adder to the "carry in" terminal of the next one in order. The operand bits are presented simultaneously to each position and the sum bits removed simultaneously. Because of the random nature of the bits of the operands, the actual delay will be random and will depend upon the maximum length that carries have to propagate.

A carry sequence of length j will be said to be present, if a carry is generated in some stage σ (by convention, $\sigma = 1$ represents the highest-weight position and $\sigma = n$ the lowest-weight position), the next j-1 positions to the left are set to propagate, and position $\sigma - j$ is not set to propagate. (Here we take "propagate" to mean the exclusive OR of the operand inputs rather than the inclusive OR). For each stage that generates a carry, there will be a corresponding carry sequence of length greater than zero. In linear-propagate adders, enough time has to be allowed for the maximum possible length of carry sequence, which is seen to involve the generation of a carry in the lowest-order position, with all other positions set to propagate. Thus the worst case involves a carry sequence of length n, the length of the operands. Letting t_1 represent the time for the "propagate" and "generate" signals to settle in all orders plus the time needed to generate the sum (given the carry in); letting t_2 represent the time to propagate a carry through a stage (given the "generate" and "propagate" signals), and letting t_3 represent the resolution time of the flip-flops which constitute the sum register, the complete period needed to form the sum and place it in the sum register under worst-case conditions is given by

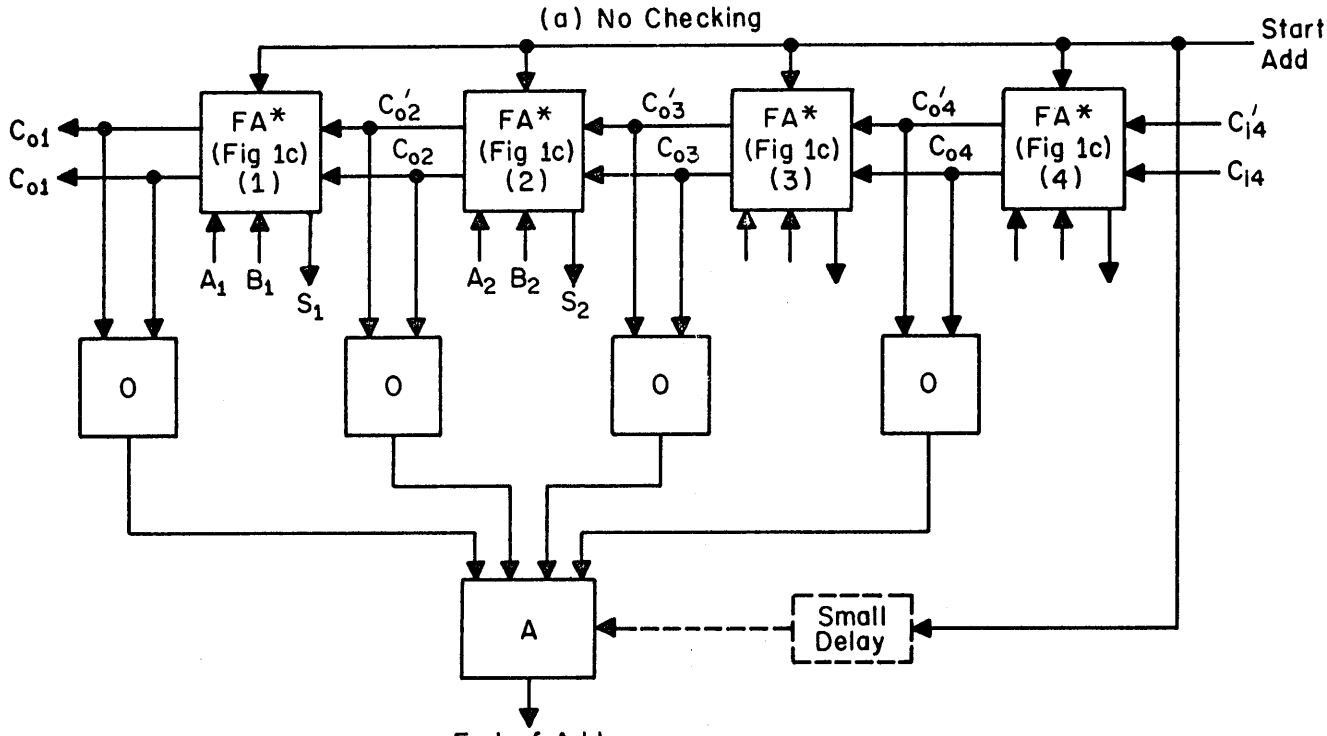
$$T = t_1 + n t_2 + t_3 \quad \text{(Linear-Propagate Adder)} \qquad (2)$$

The forms of the serial adder and the parallel linear-propagate adder are such that additions proceed at a rate of 1/T, where T is given by equations (1) and (2) respectively.

Two forms of variable-speed adders, called asynchronous adders, are shown in Figure 3. In both cases, the adder must be in a reference condition prior to the beginning of the addition, such that the lines C_o and C_o' in each position are both zero. In the adder of Figure 3a, this is performed by holding down the Start Add line for a time equal roughly to the time needed to propagate a carry through one stage, say t_2. With the adder in its reference condition, the addition is started by setting Start Add to unity. After a certain time, depending upon the operand magnitudes, one or the other (but not both) of the lines C_o and C_o' will be at unity for each position of the adder. The inclusive OR of these lines is formed for each position and connected to an n-way AND circuit, the output of which will be unity when the add is complete. This line can be used to cause the transfer of the sum to the sum register and after this is done, the adder can be reset to its reference condition for the next add operation. Internal race conditions leading to transient

Figure 3 ASYNCHRONOUS ADDERS

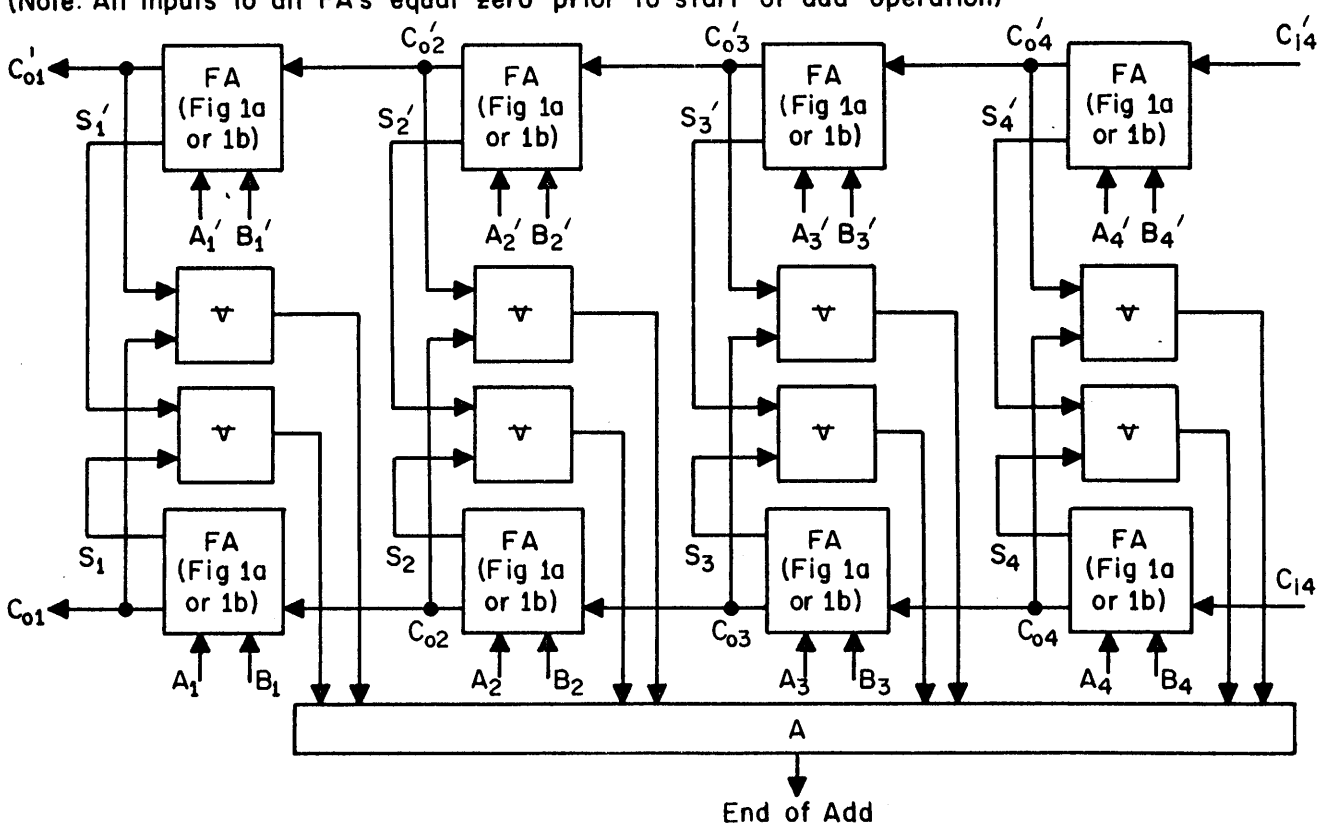
(a) No Checking



*Note that Full Adders are modified from Fig. 1c so that if Start Add line equals zero, then $C_{o1}=C'_{o1}=C_{o2}=C'_{o2}=C_{o3}=C'_{o3}=C_{o4}=C'_{o4}=0$.

(b) With Checking

(Note: All inputs to all FA's equal zero prior to start of add operation)



hazards on the carry lines may exist at the start of the add operation, but so long as a minimum time delay is enforced for the operation (as for example by means of the delay line shown) these should not cause error. After this initial delay, the appearance of a one on either the C_o line or the C_o' line will indicate that the "carry in" for that position has been received, or else internally generated or inhibited (both operand bits at zero). By the time the End of Add signal comes up, the sum lines should be settled to their final value. Thus, letting t_1 represent the minimum time allowed for the add operation (no carries propagated), assuming the time needed to reset the adder equals t_1, and letting t_3 represent the resolution time of the flip-flops of the sum register, we find that the execution time T of this adder is bounded as follows:

$$2t_1 + t_3 \leqq T \leqq 2t_1 + nt_2 + t_3, \text{ (Asynchronous Adder)} \quad (3)$$

where n is the length of the operands.

The actual value of T will depend upon the distribution of bits in the operands, hence the term "self-timed" is used to refer to asynchronous adders. The operands are random and some assumption as to the probability distributions has to be made in order to calculate the probability distribution of the maximum length carry sequence. For this type of adder, one must consider the propagation of zero carries in addition to the propagation of one carries in determining the maximum length carry sequence. Assuming a uniform distribution of operand bits (a one or zero in each position is equally probable), Gilchrist, Pomerene and Wong[3] show that the mean of the maximum-length carry sequence in adding 40-bit words equals 5.6.

In the Appendix of this section, we show that the expected maximum length carry sequence has an upper bound of $[\log_2 n] + 2$, where the brackets mean to take the integral portion of the expression enclosed, and n is the number of bits in the operands. Using this expression, the average add time for the asynchronous adder is bounded as follows:

$$T_{avg} \leqq 2t_1 + \left\langle \left[\log_2 n\right] + 2 \right\rangle t_2 + t_3 . \quad (4)$$

(Asynchronous Adder)

Figure 3b shows an adder similar to the one in Figure 3a except for the use of a pair of binary full adders in each position and the use of exclusive OR circuits. The exclusive OR circuits are used not only for the sensing of carry completion, but also for sensing of sum completion and error detection. If any sum or "carry out" is in error, the "End of Add" signal cannot come up. The same relations govern the time of executions as hold in the case already considered.

Another form of variable delay adder is shown in Figure 4. The distinguishing features of this adder are (1) each stage involves only a half adder, and (2) the "carry out" from each order is stored (or "saved") in a flip-flop and introduced as a "carry in" to the next order in the next cycle. To start, the two operands are added and the sum and carry words both stored in registers. In the next cycle, the numbers in the sum and carry registers are added and the revised sum and carry words formed. This is continued until the carry register contains all zeros. The operation is essentially synchronous, but a variable number of cycles is needed in order to propagate the carries. Letting t_1 represent the time needed to gate in operands and to form the sum or carry, whichever appears later; letting t_3 represent the resolution time of the flip-flops of the sum and carry registers, and allowing one cycle for the recognition of the "End of Add" signal, the execution time is given by

$$2(t_1 + t_3) \leqq T \leqq (n + 2)\ (t_1 + t_3) \quad \text{(Carry Save Adder)} \qquad (5)$$

The actual execution time will depend upon the distribution of one's and zero's in the operand words. In this case, only the propagation of one's needs to be considered. The Appendix shows that the average of the maximum length of carry sequence is bounded by $[\log_2 n] + 1$. Hence, the average execution time is bounded as follows:

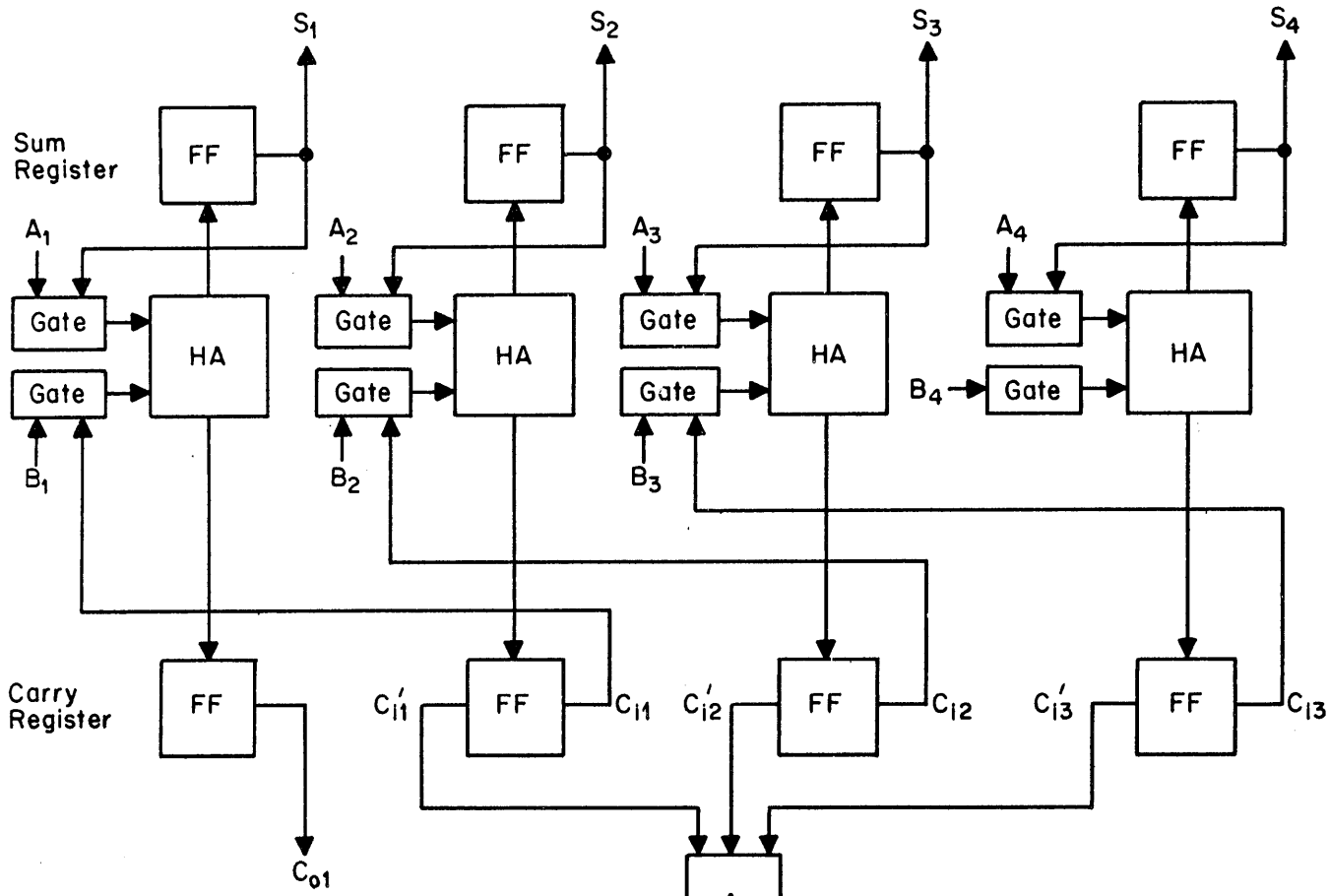
$$T_{avg} \leqq \left[\log_2 n + 3\right]\ (t_1 + t_3). \quad \text{(Carry Save Adder)} \qquad (6)$$

Carry Transformations - The length of the random maximum-length carry sequence forms a principal determinant of the speed of the adders considered above. Except for the serial and linear-propagate adders, each of these could take advantage of the relatively low average of the maximum-length carry sequence. In the linear-propagate adder, enough time has to be allowed for the maximum possible length of carry sequence. This situation has led to numerous strategies for speeding up the carry propagation by transforming those portions of the adder which have the specific function of transmitting the carry to the next position. For the individual stage, these are shown in dotted rectangles in Figure 1 for the various forms of binary adder considered.

Consider a linear-propagate adder with n stages, numbered from 1 to n from left to right (high order to low order). Letting C_{oj}, G_j and P_j represent the "carry out", "generate" and "propagate" signals, respectively, of the j-th stage, letting $C_o\ (n + 1)$ represent the carry into the adder $\langle C_o\ (n+1) = C_{i,n} \rangle$; then the following set of equations represent the carry propagation circuitry:

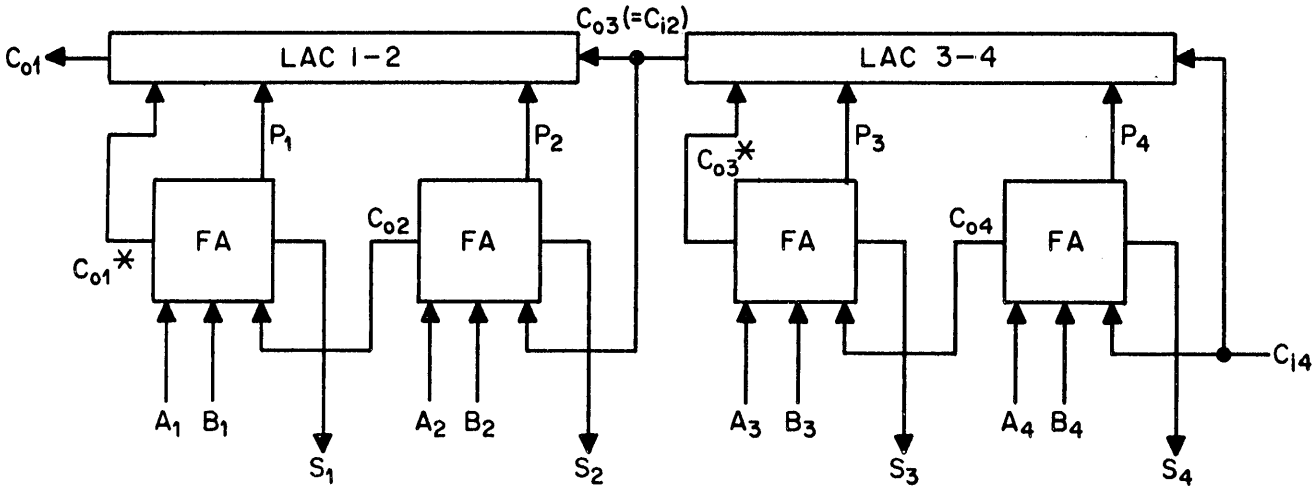
$$C_{oj} = G_j + P_j\ C_{o\ (j+1)}; (j = 1, 2, \ldots, n) \qquad (7)$$

Figure 4 CARRY SAVE ADDER



NOTE: On first cycle, gate oper and bits to adder and set Registers; on other cycles gate registers to adder and set Registers. Continue until End of Add equals Unity.

Figure 5 LINEAR GROUPED ADDER



In words, the "carry out" of the j-th stage will occur if a carry is "generated" within the j-th stage or if a "carry out" appears from the (j + 1)-th stage and the j-th stage is set to "propagate". Considering only the first four stages, the first four of the above set of equations can be combined to form

$$C_{o1} = G_1 + P_1G_2 + P_1P_2G_3 + P_1P_2P_3G_4 + P_1P_2P_3P_4C_{o5}. \quad (8)$$

Circuitry to realize this equation in two levels of logic will be called a Simultaneous Look Ahead Carry over stages 1 through 4, abbreviated SLAC 1-4. It is used in many of the adder forms to be discussed presently.

Another form of carry transformation consists in keeping the carry propagation circuitry shown in Figure 1 for the individual stages, but adding circuitry to speed up the propagation over a group of stages. Considering again stages 1-4, the equation corresponding to the added circuitry has the form

$$C_{o1} = C^*_{o1} + P_1P_2P_3P_4C_{o5}, \quad (9)$$

where C^*_{o1} represents the "carry out" generated by the stage itself. In applying this transformation, the actual carry out for the group is taken from the added circuit. This form of circuit will be called an ordinary carry look ahead over stages 1 through 4, abbreviated as LAC 1-4. It is assumed to be formed in two levels of logic.

The forms of circuitry represented by equations (8) and (9) can be formed for any group of adjacent stages within the adder. Upon choosing the groups and the tupe of lookahead to be used for each group, one establishes a carry transformation, by which a carry sequence of length j originating at some position of the adder is executed in a time less than or equal to that required for the same carry sequence in the linear-propagate adder. In order to correlate the present discussion with the analysis of Campbell and Rosser[2], we shall assume that the delay through a lookahead circuit is equal to the time required to propagate a carry through one stage of the linear-propagate adder, regardless of the number of stages involved in the lookahead. In some technologies, the actual delays will be somewhat longer because the lookahead circuit has more inputs, so that the assumption may not strictly hold. With this assumption, the delay through a lookahead circuit equals the previously-defined t_2.

A simple form of carry transformation, resulting in what we shall call the "linear-grouped" adder is shown in Figure 5. In this transformation, the entire adder is divided into groups, at least one of which contains more than one stage. We have shown a four-bit adder divided into two groups of

two stages each. Carries are propagated linearly within a group, but are transferred by carry lookahead around the group, should the need arise. An analysis of the time delay involved is facilitated if the adder retains an iterative structure, as for example by specifying equal-sized groups. Because of the assumption on the delay of the lookahead circuitry, a carry will propagate across a group in the time it takes to propagate over one stage within a group. Letting j represent the number of units of time for propagating a carry sequence of length j in the linear-propagate adder, letting R represent the number of units of time for propagating the same carry sequence through the linear-grouped adder, letting m denote the number of stages in one group, and letting σ represent the position within a group at which the carry originates (σ = 1, 2, ..., m; counting from left to right in a group), we see that R depends upon the length of the carry sequence involved and the position within a group at which the carry sequence originates. Thus, we write

$$R = f(j, \sigma). \tag{10}$$

The function f can be determined by inspection of the circuitry in each case. In doing this for the example of Figure 5, assume that the "carry out" of the adder is connected to the "carry in" of the adder, which is the actual situation which arises in subtraction. With this symmetry, it is easy to verify that the function f for this case is as given in Table I, (p. 16). The maximum number of units of time required is seen to be 3. This is an improvement over the linear-propagate adder which requires 4 units in the worst case.

Campbell and Rosser[2] contains an analysis of the general case which shows that with $m > 2$, the number of units of time needed in the worst case is given by

$$R_{max} = n/m + 2m - 3; \quad (m > 2), \tag{11}$$

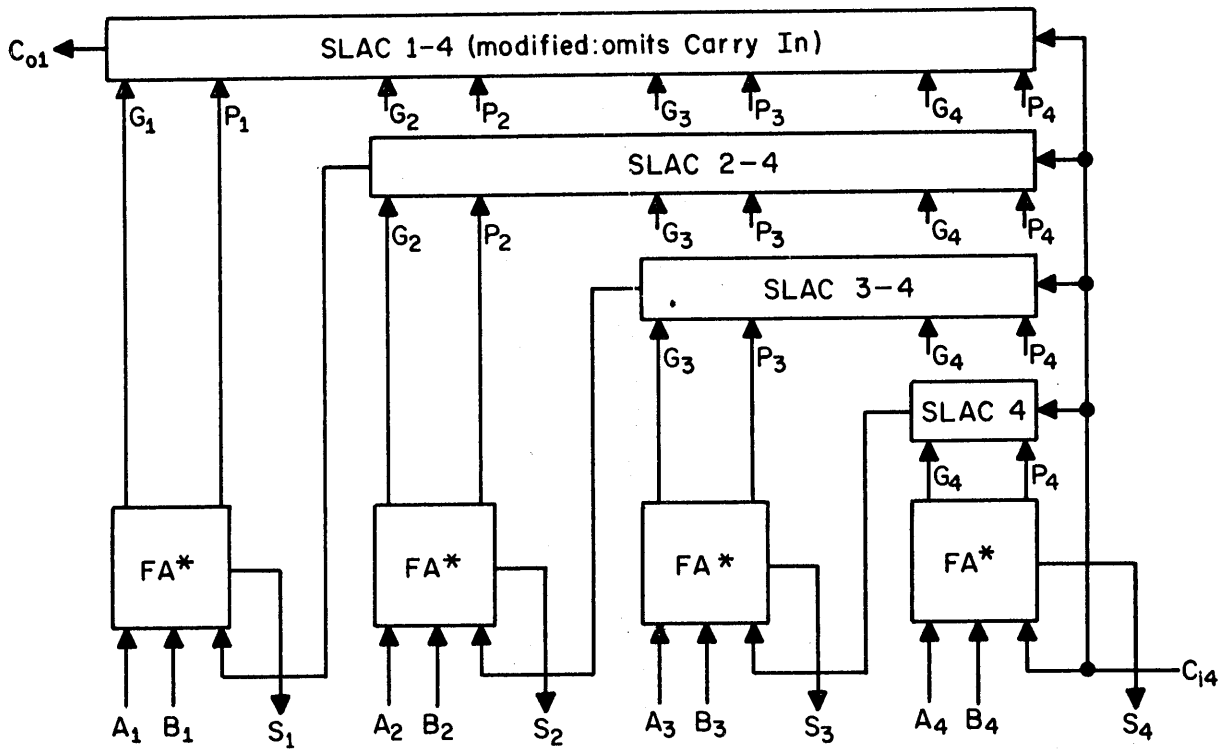
which arises when $j = n - 1$ and $\sigma = m$ (n/m equals the number of groups in the n-bit adder). Thus the linear-grouped adder has a worst-case execution time of

$$R = t_1 + (n/m + 2m - 3)\ t_2 + t_3 \quad \text{(Linear-Grouped Adder)} \tag{12}$$

<u>Simultaneous Adder</u> - A different sort of carry transformation is used in the adder as shown in Figure 6. Once the "generate" and "propagate" signals have settled for all stages, the "carry in" to each stage and the "carry out" of the adder as a whole is formed in the time required for the linear-propagate adder to transmit the carry over one stage. The full adders (denoted by FA) do not need the internal carry generating circuit, and this is indicated by the asterisk. Thus, the simultaneous adder has an execution time given by

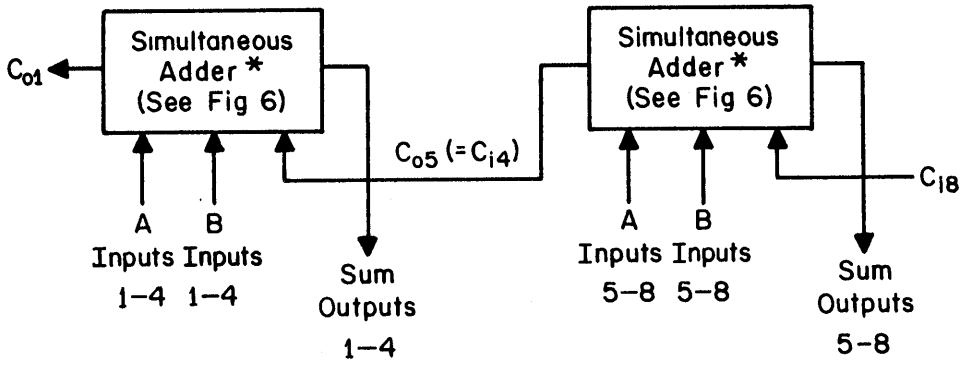
$$T = t_1 + t_2 + t_3; \quad \text{(Simultaneous Adder)} \tag{13}$$

Figure 6 SIMULTANEOUS ADDER



*Individual FA's have no means for direct carry propagation.

Figure 7 LINEAR SIMULTANEOUS ADDER



* Note that the longest SLAC circuit is not modified.

A design strategy related to the use of a simultaneous adder consists in dividing an n-bit adder into groups of simultaneous adders (m stages in each group) linearly connected. This scheme is illustrated in Figure 7. An analysis in Campbell and Rosser[2] for equal size sub-adders shows that N, the number of units of time needed to propagate a carry sequence depends upon j and σ. For $m < j \leq n$, N is given by

$$N = 3 - \delta_{\sigma,1} + \left[(j - \sigma - 1)/m \right], \qquad (14)$$

where $\delta_{\sigma,1}$ represents the Kronecker delta ($\delta_{\sigma,1} = 1$ if $\sigma = 1$, otherwise zero), and the brackets mean to take the integer part only. Taking again the case of $m > 2$, we see that N_{max} is given by $(n/m) + 2$. Thus, for execution time, we have for the worst case

$$T = t_1 + ((n/m) + 2)\, t_2 + t_3 \text{ (Linear Simultaneous Adder)} \qquad (15)$$

More General Forms of Carry Transformation - By considering each of the above forms of adder as simply a single group in a larger adder, we arrive at more general carry transformations. Thus, in Figure 8 we show a so-called "grouped-grouped" adder; in Figure 9, a "grouped-simultaneous" adder; in Figure 10, a "simultaneous-grouped" adder; and in Figure 11, a "simultaneous-simultaneous" adder. In each case, the first part of the name designates the manner of carry propagation involved in connecting the main groups. The second part designates the type of adder which forms one group.

A means for saving equipment in the Simultaneous-Grouped Adder is shown in Figure 10. Rather than have the "generate" and "propagate" signals feed from each stage to the SLAC circuits, a "generate" and a "propagate" signal is first formed for each of the linear-grouped adders as a whole, in the circuits labelled "G and P". Considering stages one through four, we have the equations

$$G_{1-4} = G_1 + P_1 G_2 + P_1 P_2 G_3 + P_1 P_2 P_3 G_4, \qquad (16)$$

$$P_{1-4} = P_1 P_2 P_3 P_4, \qquad (17)$$

which show the structure of the circuit. This strategy keeps the number of inputs to the largest SLAC circuit low at the expense of some additional time delay in the "G and P" circuits.

The number of levels of grouping can be extended to any desirable number of levels. A 100-bit adder involving three levels of grouping is shown in Figure 12. This could be called a "Simultaneous-Simultaneous-Simultaneous" Adder, but at these levels of complexity, the names become cumbersome. This form

Figure 8 GROUPED GROUPED ADDER

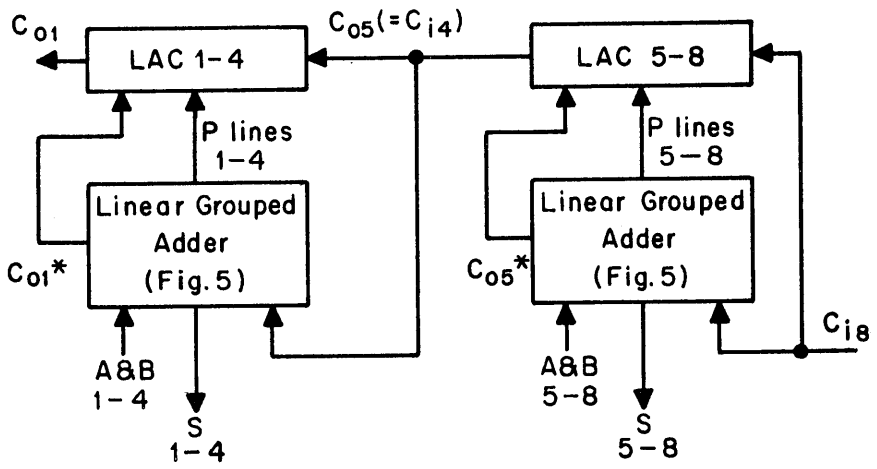
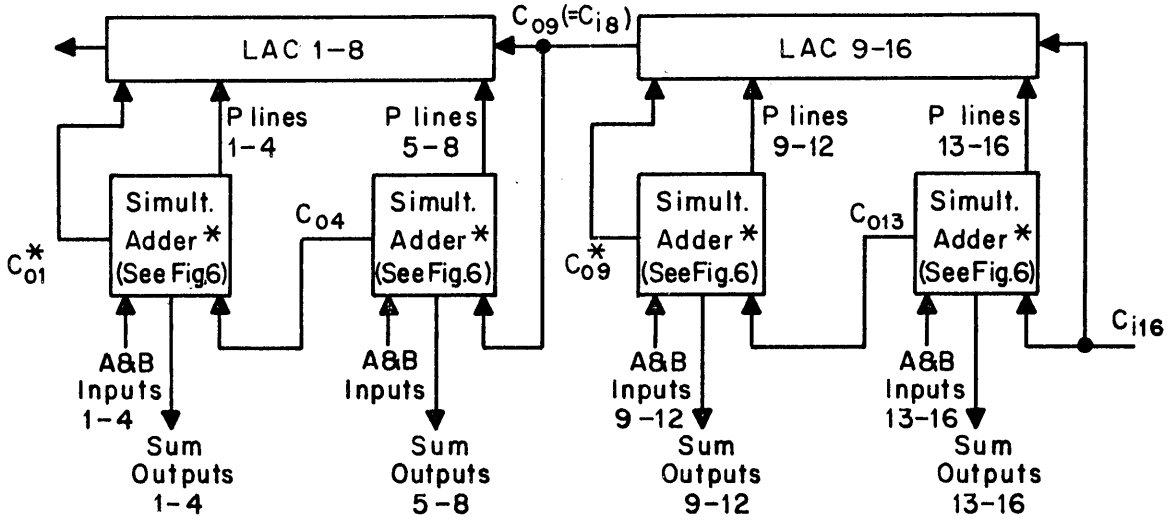


Figure 9 GROUPED SIMULTANEOUS ADDER



* SLAC circuits are not modified.

Figure 10 SIMULTANEOUS GROUPED ADDER

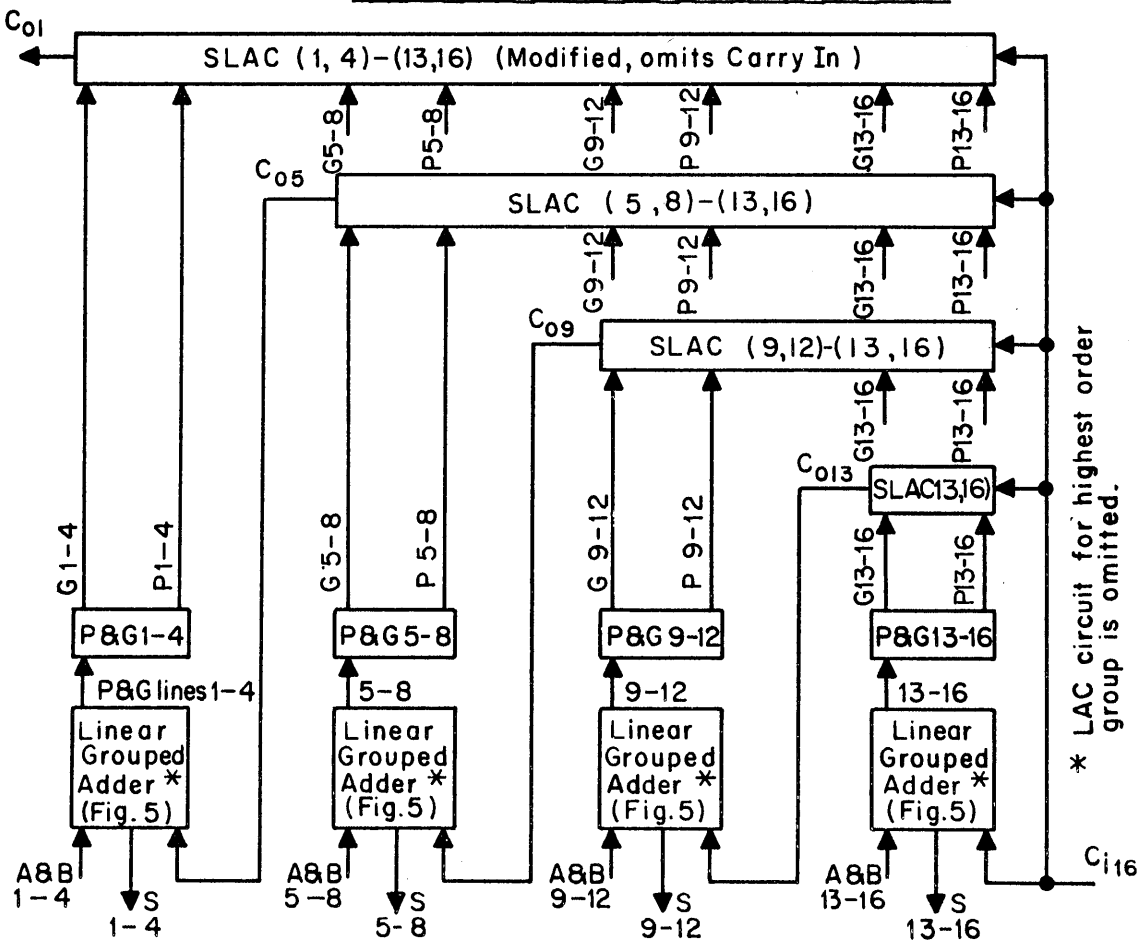
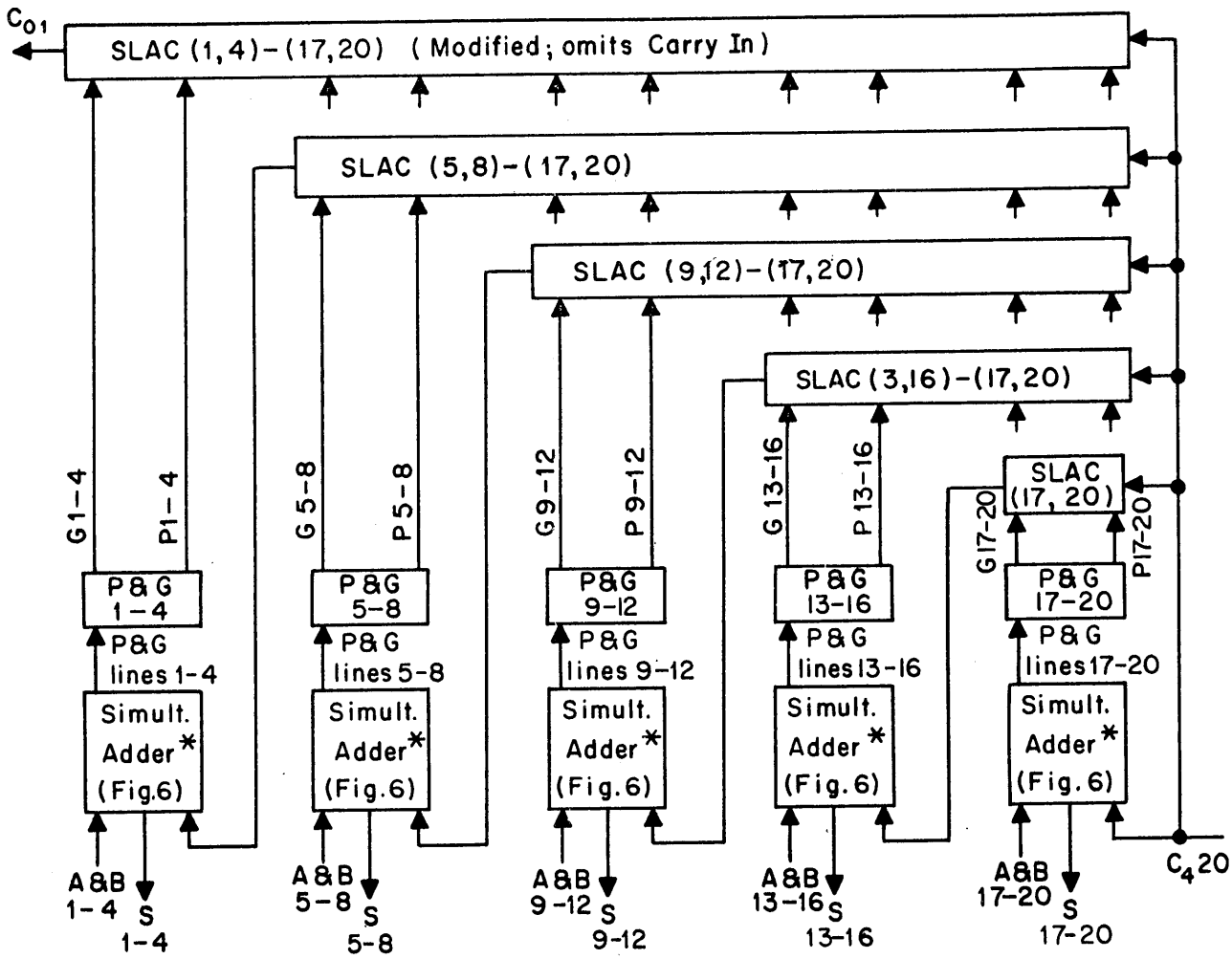


Figure 11 SIMULTANEOUS SIMULTANEOUS ADDER

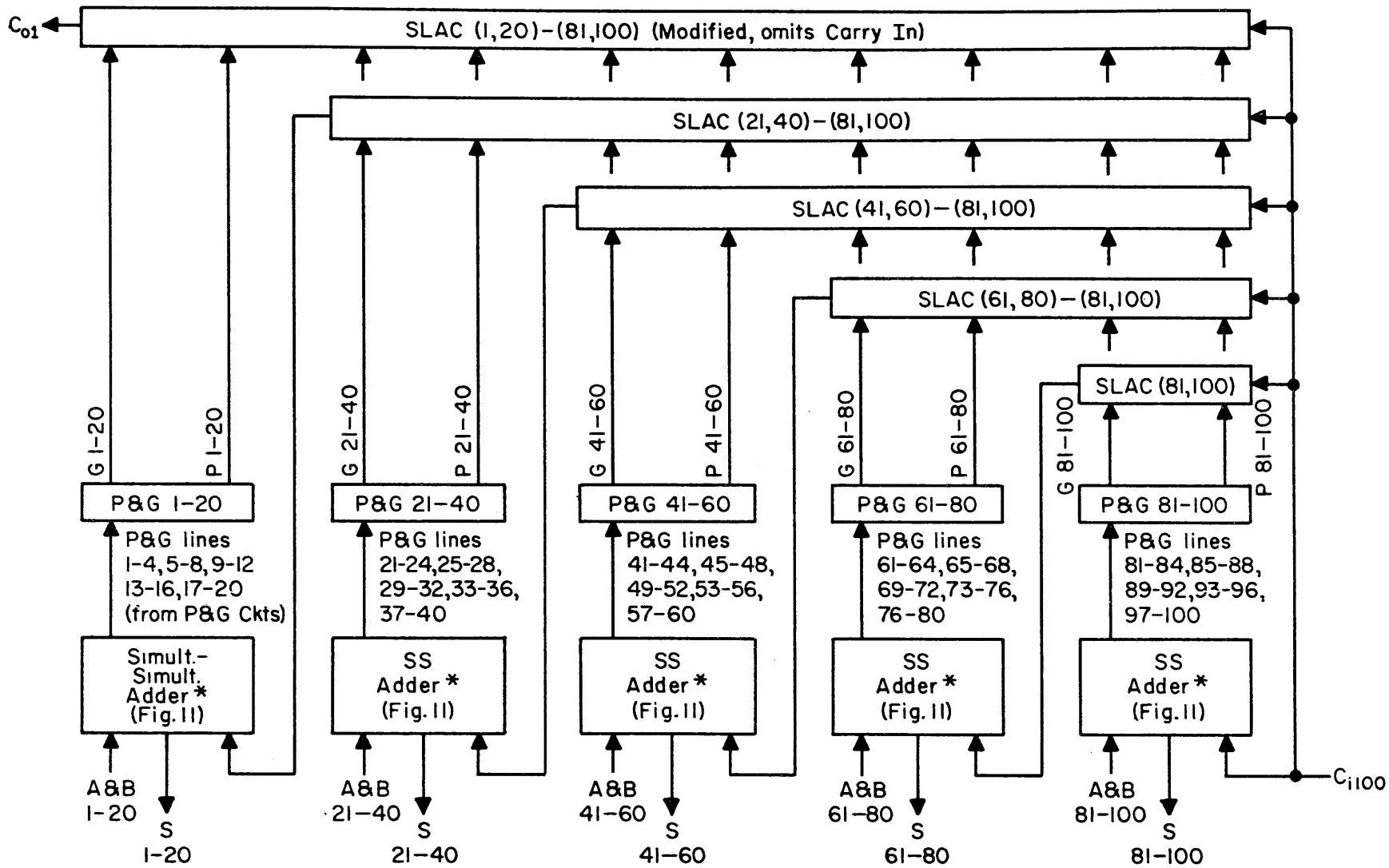


* In the Simultaneous 4-bit adders, the longest SLAC circuit is omitted. Note that outputs of G&P circuits drive all inputs vertically above points of generation.

Table I Carry Transformation of Linear Grouped Adder of Figure 5

Length of Carry Sequence j	Resultant Length for $\sigma = 1$	$\sigma = 2$
0	R = 0	R = 0
1	R = 1	R = 1
2	R = 2	R = 2
3	R = 2	R = 3
4	R = 3	R = 3

Figure 12 SIGMA ADDER



* Longest SLAC circuits of the 20-bit Simultaneous Simultaneous Adders are omitted.
Note that outputs of G&P circuits drive all inputs vertically, above points of generation.

of adder is used in the Sigma Computing System.

Evidently many possible carry transformations exist. In any specific case, the method of analysis set forther in Campbell and Rosser[2] should enable the logic designer to derive the carry transformation involved, and thus the worst case delay.

Other Possibilities - Each of the carry transformations considered above can be considered as variations on the linear-propagate adder. As such, the rate of operation of any of these adders can be no higher than that determined by the worst case carry situation. This contrasts sharply with the variable-speed adders such as the asynchronous and carry save adder schemes discussed previously, in that those of the latter group have an average speed which depends upon the average carry situation. We wish to point out, however, that the various basic methods of parallel addition can be combined in numerous ways. One could, for example, form a "carry-save grouped" adder in which the carries are propagated by some form of carry propagation within a group, but the "carries out" of a group are saved in a carry register for presentation as "carries in" during the next cycle. One could on the other hand have a "grouped-carry-save" adder in which each individual group forms a carry-save adder, but in which a look ahead carry circuit passes the carry around the group should the need arise. In this case, once a carry has been propagated across a group, means must be provided to prevent the "carry out" of the highest order stage from developing a group carry later on when the carry being propagated reaches it through the carry save feature.

As another possibility, any of several of the carry transformations discussed above can be combined with either form of asynchronous adder, by providing similar carry transformations on the zero carry lines. One could have a "linear-grouped asynchronous" adder, "linear-simultaneous asynchronous" adder, etc. Note, however, that it would be fruitless to specify a "simultaneous-asychronous" adder, since by equation (13) the speed of the adder will be as given if any carry is generated, and will be less by the amount t_2 only if no carries are generated. Thus, the delay introduced by the asynchronous connections would probably exceed t_2, which is the only time interval that could possibly be reduced.

As a final example of other possibilities, consider the following design strategy. First, determine from probability considerations that maximum length of carry sequence which will be exceeded in a small fraction of the additions, say 10%. Design a linear-simultaneous adder with this size grouping, and let its execution time equal t_1 at most for the majority (90%) of the cases. Provide additional carry transformations (say by transforming to a synchronous-synchronous adder) such that the worst of all possible carry sequences gives an execution time not exceeding $2t_1$. Finally, provide circuits

to recognize whether or not the carry sequence exceeds the size of the basic group, and design it fast enough to operate in a time t_1. Then the resulting adder can be made variable speed. In the majority of cases (90%) the execution time will equal t_1. In the other 10% of the cases, the execution time will equal $2t_1$. Whether to take one or two cycles will be decided by the recognition circuitry, which will operate fast enough. The average time will equal $.9\ (t_1) + .1\ (2t_1)$ or $1.1t_1$. The main feature of this arrangement consists of the recognition circuitry which is apt to involve considerable equipment, but with it one may almost double the speed attainable.

Summary - We have shown several basic forms of binary adders, several variations of the linear-propagate adder obtained by carry transformations, and mentioned a number of mixed forms. The logic designer has here a catalog of many possible forms. In deciding which to use in a particular application, his choice may be determined in part by considerations of the circuit technology involved. For example, in some technologies the delay through an AND-OR stage of logic may increase with the number of inputs, so that the LAC, SLAC and "P and G" circuits may not have the speeds assumed in this paper. Under such conditions, the increase in speed obtainable by carry transformations may not be as large as the theoretical figures show. Detailed considerations of this kind as applied to adders are beyond the scope of the present section. However, the reader should remember that important circuit factors enter into the consideration of each case, and so there is more to designing an adder than just picking one of the arrangements illustrated here.

REFERENCES TO SUMMARY OF BINARY ADDERS

1. R.K. Richards, "Arithmetic Operations in Digital Computers," D. Van Nostrand Co., New York, 1955.

2. S.G. Campbell and G.H. Rosser, Jr., "An Analysis of Carry Transmission in Computer Addition," AFOSR-TN-57-707, AD No. 136701, September, 1957. (A summary of this paper appears in an IBM Technical Note of the same title and by the same authors, TN 00.01011.276, May 19, 1958.)

3. B. Gilchrist, J.H. Pomerene and S.Y. Wong, "Fast Carry Logic for Digital Computers," IRE Transactions on Electronic Computers, Vol. EC-4, December, 1955, pp. 133-136.

4. R.J. Mercer, "Micro-Programming," Journal of the Association for Computing Machinery, Vol. 4, No. 2, April, 1957, pp. 157-171.

APPENDIX

Upper Bounds to the Average of the Longest Carry Sequence

Starting from simple assumptions upon the probability distributions of the two n-bit binary numbers that are to be added, we derive a recursive expression for the probability distribution function of the longest carry sequence. Neglecting certain terms in this expression that tend to reduce the values of the probabilities and manipulating the expression, we convert it to an inequality. Using this inequality in the formula for the "mean of the longest carry sequence," we derive an upper bound. Two cases are considered. In the first case, which applies to asynchronous adders, the propagation of "ones" and "zeros" is considered, and a value of $\left[\log_2 n + 2\right]$ is derived as an upper bound to the average of the longest carry sequence. In the second case, which applies to carry save adders, the propagation of "ones" only is considered, and an upper bound of $\left[\log_2 n + 1\right]$ is derived for the average. (Note that the brackets indicate to take the integer portion only of these expressions).

We assume that each bit in each word has a value independent of the values of all other bits and for each bit, a "one" or a "zero" value is equally likely and occurs with a probability of 1/2.

Considering a pair of operand bits in any position, we find four possible pairs of values: 00, 01, 10, and 11, each occurring with a probability of 1/4.

We distinguish two types of carry propagation: (1) Both "ones" and "zeros" are propagated and (2) only "ones" are propagated. In the first case, which applies to asynchronous adders, a "one" carry will be "generated" if the combination 11 occurs, and a "zero" carry will be "generated" if the combination 00 occurs. Thus the probability of "generating" a carry of either kind equals 1/2. In the second case, which applies to "carry save" adders, a "carry" will be generated if and only if the combination 11 appears, and this occurs with a probability of 1/4. In either case, if the two operand bits have opposite values (combinations 10 or 01), then any carry will be "propagated" through that stage and the probability of "propagation" equals 1/2.

Let C_{1j} and C_{2j} represent the event that a type (1) or type (2) carry respectively, is generated in some stage of the addition, is propagated through the next (j-1) positions, and enters the next stage thereafter (which may or may not be set to propagate). We shall call such an event a carry sequence of length j (of types (1) and (2), respectively).

The probability of each of these events consists of the product of the subevents which by conjunction make up the event, since the bit values for the individual stages are independent.

Hence we have

$$\text{Prob}\,(C_{1j}) = \text{Prob}\,(E_1 \text{ and } F_1 \text{ and } \cdots \text{and } F_{j-1}) = \text{Prob}\,(E_1) \cdot \text{Prob}\,(F_1) \cdot \cdots \cdot \text{Prob}\,(F_{j-1}) \qquad (1)$$

where E_1 represents the event of generating a type 1 carry in some position, and where $F_1, F_2 \cdots, F_{j-1}$ represent the events of propagating a carry through the first, second $\cdots$, and (j-1)-th positions, respectively, to the left of the one in which the carry is generated. Substituting the known values for the probabilities of these events, we find

$$\text{Prob}\,(C_{1j}) = 1/2^j \qquad (2)$$

In a similar manner we find

$$\text{Prob}\,(C_{2j}) = 1/2^{j+1} \qquad (3)$$

These probabilities enter into the formulas for expressing the probability distributions of the longest carry sequence for the two types of propagation.

Let $P_n(v)$ represent (for either type 1 or type 2 propagation) the probability that the longest carry sequence (l. c. s.) which occurs during the addition of two n-bit numbers, will equal or exceed the given amount, v. This event will be represented by (l. c. s. $\geqq$ v) or by $E_{n,v}$. We can show that this event can occur in two mutually exclusive ways. In the first case, the lowest order (n-1) positions will already contain a longest carry sequence that equals or exceeds v. This event is symbolized by (l. c. s. $_{(n-1)} \geqq$ v) or by $E_{n-1,v}$. In the alternative case, the highest-order position is set to propagate, the adjacent v-1 positions to the right contain a carry sequence of exactly (v-1), and the lowest-order (n-v) positions do not contain a longest carry sequence which exceeds or equals v. The alternative case is thus the conjunction of three events, symbolized by F_1, $C_{(v-1)}$, and $E'_{(n-v)}$, v, where F_1 is the event "first stage propagates," $C_{(v-1)}$ the event "carry sequence of length (v-1) occurs," and the prime indicates the absence of the event $E_{n-v,v}$.

With this analysis we see that

$$\text{Prob}\,(E_{n,v}) = \text{Prob}\left[(E_{n-1,\,v}) \text{ or } \left\langle (F_1) \text{ and } (C_{v-1}) \text{ and } E_{(n-v),\,v}\right\rangle\right] \qquad (4)$$

where the "or" is to be taken in the exclusive sense. From probability theory, we know that the probability of an exclusive "sum" of events is equal to the sum of the probabilities of the events. For a conjunction of independent events, the probability equals the products of the probabilities of the individual events. Finally, the probability of the negation of the event

is equal to (the probability of that event) subtracted from (unity). Applying these values to equation (4), we obtain

$$\text{Prob}\,(E_{n,v}) = \text{Prob}\,(E_{n-1,v}) + \text{Prob}\,(F_1) \cdot \text{Prob}\,(E_{v-1}) \left[1 - \text{Prob}\,(E_{n-v,v})\right] \tag{5}$$

For type 1 carry propagation, C_{v-1} reduces to $C_{1,\,(v-1)}$ whose probability is given by equation (2) with $j = v-1$. Since F_1 has a probability of 1/2 and the other symbols have been defined, for this type of propagation we find

$$P_n\,(v) = P_{n-1,v} + \left(\frac{1}{2}\right) \cdot \left(\frac{1}{2^{v-1}}\right) \left(1 - Pn - v\,(v)\right) \tag{6}$$

For type 2 carry propagation C_{v-1} reduces to $C_{2,\,v-1}$ whose probability is given by equation (3) with $j = v-1$. Thus we obtain for this type of propagation,

$$P_n(v) = P_{n-1}(v) + \left(\frac{1}{2}\right)\left(\frac{1}{2^v}\right)\left(1 - P_{n-v}\,(v)\right) \tag{7}$$

These are recursive relations for the probability distributions sought.

Considering now type 1 propagation we can transform equation (6) into an inequality by writing

$$P_n\,(v) - P_{n-1}\,(v) \leqq \frac{1}{2^v} \tag{8}$$

Suppose we form

$$\sum_{i=v}^{n} \left[P_i\,(v) - P_{i-1}\,(v)\right] \tag{9}$$

Since all probabilities except $P_n\,(v)$ and $P_{v-1}\,(v)$ will cancel out of the sum and since $P_{v-1}\,(v)$ equals zero (a carry sequence greater than the length of the adder cannot occur), we see that the entire sum equals $P_n(v)$. By changing the index n of equation (8) to index i and performing the sum over i from v to n we obtain expression (9) on the left (equal to $P_n\,(v)$) and on the right a total of $(n-v+1)$ terms each equal to $1/2^v$. Hence

$$P_n\,(v) \leqq \frac{n - v + 1}{2^v} \tag{10}$$

Recognizing that $P_n(v)$ must be a probability and is bounded by $0 \leqq P_n(v) \leqq 1$, we can write

$$P_n\,(v) \leqq \min\left[1,\ \frac{n-v+1}{2^v}\right] \tag{11}$$

where the right member indicates to take either 1 or $(n-v+1)/2^v$, whichever is less. This inequality will be used in determining the bound on the mean of the longest carry sequence.

If $P_n(v)$ is the probability that the longest carry sequence equals or exceeds v, then the probability that the longest carry sequence is exactly equal to v is the probability that the longest carry sequence equals or exceeds v but does not equal or exceed v + 1. Thus we write

$$\text{Prob}\,(\text{l. c. s.}_n = v) = P_n(v) - P_n(v+1) \tag{12}$$

The mean or average of a discrete random variable X that can take any values from 1 through n, is defined as

$$X_{avg} = \sum_{i=1}^{n} i \text{ Prob}\,(X = i) \tag{13}$$

Using this definition and letting M_1 represent the mean of the longest carry sequence we find from equation (12) that

$$M_1 = \sum_{i=1}^{n} v \cdot \text{Prob}\,(\text{l. c. s.} = v) = \sum_{v=1}^{n} v \cdot \left[P_n(v) - P_n(v+1)\right] \tag{14}$$

By writing a few terms of the sum as follows:

$$\begin{aligned} & \left[1 \cdot P_n(1) - 1 \cdot P_n(2)\right] \\ + & \left[2 \cdot P_n(2) - 2 \cdot P_n(3)\right] \\ + & \left[3 \cdot P_n(3) - 3 \cdot P_n(4)\right] \\ & \quad \vdots \\ + & \left[n \cdot P_n(n) - n\, P_n(n+1)\right] \end{aligned} \tag{15}$$

the reader will note that terms partially cancel. Using this fact and the fact that $P_n(n+1) = 0$, we can write

$$M_1 = \sum_{v=1}^{n} P_n(v) \tag{16}$$

Now substituting inequality (11) into equation (16) we find

$$M_1 \leqq \sum_{v=1}^{n} \text{Min} \left\langle 1, \frac{n-v+1}{2^v} \right\rangle \tag{17}$$

The rest of the derivation consists in finding an upper bound for this series which can be taken as an upper bound for the longest carry sequence.

In forming this sum we see that the minimum of the pair of magnitudes will equal unity for values up to some value of v, say v_1. For $v \geqq v_1$, the minimum will be less than unity. . To find v_1, find bounds on v such that

$$\frac{n - v + 1}{2^v} \leqq 1 \tag{18}$$

$$\text{Thus } 2^v \geqq n - v + 1 \tag{19}$$

This will be assured if

$$2^v \geqq n \geqq 2 \tag{20}$$

$$\text{or if} \quad v \geqq \log_2 n \geqq 1 \tag{21}$$

Letting $\log_2 n = a + e$, where a is an integer and $0 \leqq e < 1$, we see that with $v = \left[\log_2 n + 1\right]$ (brackets mean to take the integer part only), we have

$$\frac{n - v + 1}{2^v} = \frac{2^{a+e} - (a+1) + 1}{2^{a+1}} = 2^{(e-1)} - \frac{a}{2^a} \tag{22}$$

Restricting the discussion to $n \geqq 2$ and $a \geqq 1$, we see that the expression must be less than unity, although it may be almost unity for some large values of n. On the other hand, letting $v = \left[\log_2 n\right] = a$, we see that

$$\frac{n - v + 1}{2^v} = \frac{2^{a+e} - (a) + 1}{2^a} = 2^e - \frac{(a-1)}{2^a} \tag{23}$$

and for $n \geqq 2$, $a \geqq 1$, this expression can exceed unity.

Hence we may choose $v_1 = \left[\log_2 n + 1\right]$

$$\sum_{v = \left[\log_2 n + 1\right]}^{n}$$

Then we have

$$M_1 \leqq \left[\log_2 n\right] + \left(\frac{n - v + 1}{2^v}\right) \tag{24}$$

Consider the summation as proceeding to infinity. (This only enlarges the right-hand member). Since the first term is less than unity but may be very close to unity for some values of n; and since the ratio of successive terms is less than 1/2; we see that the entire sum is less than the infinite series

$$1 + \frac{1}{2} + \frac{1}{4} + \ldots$$

which has a sum equal to 2. Thus we see

$$M_1 \leqq \left[\log_2 n\right] + 2 \tag{25}$$

Considering now type 2 carry propagation, similar arguments apply to derive the mean of the longest carry sequence, but wherever one had the factor 2^v, one replaces it with 2^{v+1}, this being the only operation needed to convert equation (6) into equation (7), which is the foundation of the analysis. Thus one can show that M_2, defined as the mean of the longest carry sequence, is bounded as follows:

$$M_2 \leqq \sum_{v=1}^{n} \min\left[1, \frac{n - v + 1}{2^{v+1}}\right] \tag{26}$$

For this case we seek bounds on v for which

$$\frac{n - v + 1}{2^{v+1}} \leqq 1 \tag{27}$$

But this requires

$$2^{v+1} \geqq n - v + 1 \tag{28}$$

This will be assured if

$$2^{v+1} \geqq n \geqq 4 \tag{29}$$

or if $v \geqq (\log_2 n) - 1 \geqq 1$ (30)

Thus with $v = \left[\log_2 n\right] = \left[a + e\right] = a$,

we have

$$\frac{n - v + 1}{2^{v+1}} = \frac{2^{a+e} - (a) + 1}{2^{a+1}} = 2^{e-1} - \frac{(a-1)}{2^{a+1}} \leqq 1 \tag{31}$$

but with $v = a - 1$,

$$\frac{n - v + 1}{2^{v+1}} = \frac{2^{a+e} - (a-1) + 1}{2^a} = 2^e - \frac{(a-2)}{2^a} \geqq 1 \tag{32}$$

Hence we can write

$$M_2 = \left[\log_2 n\right] - 1 + \sum_{v=\left[\log_2 n\right]}^{n} \left(\frac{n - v + 1}{2^{v+1}}\right) \tag{33}$$

Again comparing the summation with the infinite geometric series we find that it is no greater than 2. Thus, we can write

$$M_2 \leqq \left[\log_2 n + 1\right] . \tag{34}$$

International Business Machines Corporation, Poughkeepsie, New York

Laboratory Publications, Product Development Laboratory, Data Systems Division IBM