We note that the average value of τ necessarily is less than

a) the delay incurred in entering the summand bits into the carry transmission gates, plus
b) the average propagation time of the longest carry sequence through the carry transmission gates, plus
c) the delay through an OR gate leading into the carry completion gate, plus
d) the longest possible delay through the carry completion gate.

Delay a) is 2. Delay b) is 2αn, caused by the delay of 2 in each carry transmission gate. Delay c) is 1. Delay d) is n - 1. Hence,

\[ \tau < 2 + \alpha n + 2\alpha n. \]  
(15)

Substituting (14) into (15) yields

\[ \tau < 4 + \alpha n + 2 \log_2 n, \]  
(16)

which completes our derivation.

ACKNOWLEDGMENT

The author is indebted to Dr. S. Amarel of RCA Labs. for many discussions on the general approaches to sub-systems evaluation.

BIBLIOGRAPHY

[10] F. H. Fowler, Jr., internal report, RCA.

Conditional-Sum Addition Logic*

J. SKLANSKY†, SENIOR MEMBER, IRE

Summary—Conditional-sum addition is a new mechanism for parallel, high-speed addition of digitally-represented numbers. Its design is based on the computation of “conditional” sums and carries that result from the assumption of all the possible distributions of carries for various groups of columns.

A rapid-sequence mode of operation provides an addition rate that is invariant with the lengths of the summands. Another advantage is the possibility of realizing the adder with “integrated devices” or “modules.”

The logic of conditional-sum addition is applicable to all positive radices, as well as to multimundand operation.

In a companion paper, a comparison of several adders shows that, within a set of stated assumptions, conditional-sum addition is superior in certain respects, including processing speed.

* Received by the PGEC, December 2, 1959; revised manuscript received, March 31, 1960.
† RCA Labs., Princeton, N. J.

I. INTRODUCTION

CONDITIONAL-sum addition is a new scheme of parallel, high-speed addition for digital computers. A comparative evaluation of several binary adders indicates that the conditional-sum adder (CSA) is quantitatively superior in certain important respects, including computation speed.

In the present paper the basic concepts of CSA logic are presented, and a specific AND-OR-NOT network realizing the CSA is described.

II. NORMAL MODE OF OPERATION

We explain the normal operation of CSA b the following example.

Fig. 1 shows the conditional-sum addition of two binary-coded numbers:

\[
\begin{align*}
x & = 101110110110101101101 \\
y & = 0001100110110110110.
\end{align*}
\]

During \( \tau_1 \) conditional sums and carries for pairs of columns (namely, column numbers 0 and 1, 2 and 3, \ldots, and 14 and 15) are computed simultaneously, under the assumptions \( (A) \) that the carry brought to each pair of columns is 0, and \( (B) \) that this carry is 1. The rows belonging to \( \tau_1 \) are arranged in a manner similar to those of \( \tau_0 \).

Continuing this process, the “conditional” sums and final carries are computed for tetrads of columns during \( \tau_2 \), for octads of columns during \( \tau_3 \), and for the entire sixteen-column group during \( \tau_4 \). During \( \tau_4 \) the sums and final carry produced are no longer “conditional,” but are equal to the sum bits and the final carry for the two summands, \( x_4 \) and \( y_4 \).

To understand the process in more detail, consider the pair of \( \tau_1 \) columns \((0, 1)\) (referred to hereafter as a “\( \tau_1 \) array”). The entries for the upper half of \( \tau_0 \)-column 0 are the same as those of the upper half of \( \tau_0 \)-column 0, since the entries in these halves both correspond to the case \( c_0 = 0 \). However, since the carry for column 0 will no longer be needed, only the sum bit need be entered in the \( \tau_1 \) array. The entries in the upper half of \( \tau_1 \)-column 1 are found from \( \tau_0 \)-column 0 in the following way: If the second entry of \( \tau_0 \)-column 0 is \( 0 \) — i.e., if \( c_0 = 0 \)— then the upper half of \( \tau_1 \)-column 1 is identical to the upper half of \( \tau_0 \)-column 1; if \( c_0 = 1 \), then the upper half of \( \tau_1 \)-column 1 is identical to the lower half of \( \tau_0 \)-column 0. The reason for this is that the upper half of any \( \tau_0 \) column corresponds to a zero-valued incoming carry for that column, while the lower half corresponds to a one-valued incoming carry. Thus, since \( \tau_0 \)-column 0, viz.,

\[
\begin{bmatrix}
1 \\
0
\end{bmatrix},
\]

has a second entry of 0, it follows that \( \tau_1 \)-column 1 is identical to the upper half of \( \tau_0 \)-column 1, viz.,

\[
\begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix}
\]

Summarizing,

\[
\begin{bmatrix}
1 & 1 \\
0 & 0 \\
0 & 0
\end{bmatrix}
\]

\( \Rightarrow \)

\[
\begin{bmatrix}
1 & 1 \\
0 & 0
\end{bmatrix}
\]

In a similar manner,

\[
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
1 & 1
\end{bmatrix}
\]

\( \Rightarrow \)

\[
\begin{bmatrix}
0 & 0 \\
0 & 1 \\
1 & 1
\end{bmatrix}
\]

(We define our symbols in Appendix 1.) For column 1 we have

\[
\begin{align*}
s_1^0 &= x_1 \oplus y_1 = 1 \oplus 0 = 1 \\
c_1^0 &= x_1 \land y_1 = 1 \cdot 0 = 0 \\
s_1^1 &= s_1^0 \oplus 1 = 0 \\
c_1^1 &= x_1 \lor y_1 = 1 \lor 0 = 1
\end{align*}
\]

In a similar manner we find the other entries in the \( \tau_0 \) columns.
Fig. 2—The logic circuit of a seven-bit conditional-sum adder. (a) The over-all circuit; (b) an AND-OR-NOT circuit for $H$, a portion of which is the circuit for $HA$; (c) and (d): $Q_1$ and $Q_2$ in terms of $M$; (e) an AND-OR-NOT realization of $M$. The circuits for $Q_3$ and other $Q_i$'s can be inferred from (c) and (d).

In the above situation, the lower half of the $r_1$ array is determined by the lower right-hand carry bit in the $r_0$ array, i.e., the bit in the lower right-hand corner.

For later intervals, $r_i$, the operation is similar except that greater numbers of columns are involved at each step. For instance, to obtain columns 4 to 7 of interval $r_2$ the following transformation takes place:

$$
\begin{align*}
\begin{bmatrix}
1 & 1 & 0 & 1 \\
0 & 1 \\
0 & 0 & 1 & 0 \\
1 & 1
\end{bmatrix} & \Rightarrow \\
\begin{bmatrix}
0 & 0 & 0 & 1 \\
1 \\
0 & 0 & 1 & 0 \\
1
\end{bmatrix}
\end{align*}
$$
For each interval \( \tau_j \) the columns in Fig. 1 are marked off in groups of 1, 2, \( \ldots \), \( 2^j \) \( \ldots \) according to the size of the group of columns participating in the transformation from \( \tau_{j-1} \) to \( \tau_j \).

III. RAPID-SEQUENCE MODE OF OPERATION

In the event several sums are to be computed successively, one may increase the effective speed of the adder by storing the results of cycle \( \tau_1 \) for use during cycle \( \tau_{j+1} \); simultaneously another set of conditional sums and carries could be computed for a different pair of summands. Consequently, the addition speed of this "rapid-sequence" mode of operation would be faster than that of the normal mode by the factor

\[
D = \frac{1}{\tau_R} (\tau_0 + \tau_1 + \cdots + \tau_1 + \cdots + \tau_p).
\]

where \( \tau_R \) pulse repetition period. When \( \tau_R = \tau_0 = \tau_1 = \cdots = \tau_p \), this expression reduces to

\[
D = p + 1.
\]

Since \( n = 2^p \) (see Fig. 1), \( D \) can be expressed in terms of \( n \):

\[
D = \log_2 2n.
\]

IV. A SUGGESTED LOGIC CIRCUITRY

The suggested logic circuitry, indicated in Fig. 2, consists entirely of AND gates, OR gates, NOT gates, and their interconnections. The timing and storage circuitry for the controlling sequence of operations is omitted. Actually this timing and storage circuitry is not necessary for the basic operation of adding two summands. However, when many summands are to be added in rapid sequence, in the manner discussed in the previous section, then it is advantageous to have timing control. Storage circuitry is needed here only for guaranteeing the proper synchronism of signals; if the AND-gates and OR gates inserted pure delays with no signal distortion, and if the delays of all the AND gates were exactly the same, then no storage circuitry would be necessary.

Fig. 2(a) shows the information-flow diagram for a seven-column adder; adders for greater numbers of columns can be inferred from the figure.

\( Q_1 \) and \( Q_2 \), shown as blocks in Fig. 2(a), may be realized in terms of a basic module, \( M \), whose AND-OR-NOT circuit is given in Fig. 2(e). The suggested \( M \) realizations for \( Q_1 \) and \( Q_2 \) are given in Fig. 2(c) and 2(d). The \( Q_i \)'s for \( i > 2 \) can be realized by networks easily inferred from the realizations of \( Q_1 \) and \( Q_2 \). A suggested AND-OR-NOT realization of \( H \) is given in Fig. 2(b).

For neatness, not all the connections are shown explicitly as continuous lines. For instance, the signals \( s_1^0 \) and \( s_1^1 \), produced by the \( H \) of column 1, are brought to the input terminals of \( Q_2 \) of column-pair (1, 2). These connections are indicated in the figure by labels on the input and output leads.

A warning to circuit designers: the maximum "fan-out" (the number of input leads emanating from an output terminal) is an increasing function of the summand length. This can be verified from an examination of Fig. 2, especially parts (c) and (d). The "fan-out" is an index of the load that a gate must be capable of handling.

V. EXTENSIONS OF THE BASIC CONCEPTS

The basic concepts of conditional-sum addition can be extended in at least the following two directions: 1) Positional number systems with radices greater than 2, in which a numeral \( x_N \cdots x_1 x_0 \) represents the number

\[
\sum_{i=0}^{N} x_i r^i,
\]

where \( r \), a positive integer, is the radix of the number system. 2) Multiple-summand addition, in which more than two summands are added simultaneously.

A. Higher Radices

It is possible to apply conditional-sum addition to higher-radix number systems with little change in the basic concepts. We illustrate this in Fig. 3 by an example in the decimal system. The alphabetical symbols here are the same as in Fig. 1, and the description of the operation is similar to that given in Section II for the binary case.

B. Multisummand Addition

The scheme of conditional-sum addition may be applied to the simultaneous addition of more than two summands. This may be done by storing and computing a conditional sum and a conditional carry for each possible value of an incoming carry at each appropriate column. For instance, in four-summand addition the possible incoming carries for any column are 0, 1, 2, and 3. Conditional sums and carries must be computed for each of these four carries during each of the cycles \( \tau_j \).

For the general case of \( p \)-summand addition the possible
radix may be.$^9$ Thus it should not be difficult to syn-
thesize a conditional-sum adder that will handle both
multiple summands as well as radices greater than 2.

VI. HIGH SPEED

In a companion paper$^1$ we show that the CSA is
basically faster in processing speed than several other
well known adders. The primary assumptions in that
analysis are

1) That the fundamental building blocks are a two-
input AND gate, a two-input OR gate, and a one-
input NOT gate;
2) That the AND gate and OR gate impose equal
delays, while the NOT gate imposes no delay.

VII. THE POSSIBILITY OF USING
"INTEGRATED DEVICES"

We note that the conditional-sum logic circuitry,
just described lends itself to realization by a large
number of identical "integrated devices" or "modules"
(labeled $M$ in Fig. 2). The use of "modular" construction
is likely to be economically attractive, especially
with the recent development of "integrated" solid-state
devices.$^8$

$^1$ This is proved in Appendix II.
$^8$ J. T. Wallmark and S. M. Marcus, "Integrated devices using
direct-coupled unipolar transistor logic," IRE Trans. on Elec-

VIII. CONCLUSIONS AND SUMMARY

Conditional-sum addition has the following attractive
properties:

1) In the "normal" mode of operation, the addition
time per column is a decreasing function of the number
of columns, $n$. (This time is roughly $(\log_2 2n)/n$ gate
delays.)

2) In the "rapid-sequence" mode of operation, the
time between the production of successive $n$-bit sums is
invariant with respect to $n$.

3) Both the normal and the rapid-sequence modes of
operation are completely synchronous, so that the con-
trol circuitry associated with a CSA adder should be
simpler than for comparable asynchronous mechanisms.

4) The logic circuit suggested for CSA contains many
identical AND-OR-NOT subnetworks, thereby lending
itself to realization by "integrated devices" or "modules."

5) CSA may be extended to other modes of arith-
metic, specifically: multisummand addition and non-
binary radices.

6) CSA proves superior to other schemes in addition
logic in certain quantitative respects, including process-
ing speed.$^1$

APPENDIX I

DEFINITIONS OF SYMBOLS

$\oplus =$ plus, modulo 2.
$\cdot =$ AND.
$\lor =$ INCLUSIVE OR.
$i =$ column number, beginning with column 0.
$j =$ ordinal number of an interval, $\tau_i$, during which an array of conditional sums and carries are produced.

$x_i, y_i =$ summand bits of column $i$.
$c_i =$ carry bit entering column $i$.
$s_i =$ sum bit of column $i$.
$c_i^0 =$ carry generated at column $i - 1$, assuming $c_{i-1} = 0$.
$c_i^1 =$ carry generated at column $i - 1$, assuming $c_{i-1} = 1$.
$c_i^{00} =$ carry generated at column $i - 1$, assuming $c_{i-2} = 0$.
$c_i^{01} =$ carry generated at column $i - 1$, assuming $c_{i-2} = 1$.
$c_i^{10} =$ carry generated at column $i - 1$, assuming $c_{i-3} = 0$.
$c_i^{11} =$ carry generated at column $i - 1$, assuming $c_{i-3} = 1$.

$c_{i-1}^0 =$ carry generated at column $i - 1$, assuming $c_{i-2} = 0$.
$c_{i-1}^1 =$ carry generated at column $i - 1$, assuming $c_{i-2} = 1$.
$s_i^0 =$ sum generated at column $i$, assuming $c_i = 0$.
$s_i^1 =$ sum generated at column $i$, assuming $c_i = 1$.

$s_i^{00} =$ sum generated at column $i$, assuming $c_{i-1} = 0$.
$s_i^{01} =$ sum generated at column $i$, assuming $c_{i-1} = 1$.

The graphical symbols used in the figures are self-explanatory.
APPENDIX II

A Theorem for Multisummand Addition

In an addition of any \( p \) positive summands, the carry produced by any column has a maximum possible value of \( p - 1 \).

This result is independent of the radix, \( r \), and includes the case of a carry consisting of more than one digit, i.e., a carry \( \geq r \). There is no restriction on the length of the summands.

Proof: Consider column 0. The maximum possible value of its sum is \( pr - p \). We ask the reader to verify that the corresponding output carry of this column is \( p - p_1 \), where \( p_1 \) is the positive integer satisfying the diophantine inequality

\[
(p_1 - 1)r \leq p < p_1r
\]  

(4)

(i.e., \( p_1 \) is 1 plus the number remaining after the lowest-order digit of the \( r \)-ary representation of \( p - 1 \) is deleted). As a consequence of (4) and the fact that \( r > 1 \), it follows that

\[
1 \leq p_1 \leq p.
\]  

(5)

The maximum possible sum produced by column 1 is therefore \( pr - p_1 \). Hence, by an analysis similar to that used for column 0, we conclude that the corresponding output carry of column 1 is \( p - p_2 \), where \( p_2 \) is the positive integer satisfying

\[
(p_2 - 1)r \leq p_1 \leq p_1r.
\]  

(6)

Hence,

\[
1 \leq p_2 \leq p_1
\]  

(7)

Continuing in this manner, we find that the maximum possible input carry of column \( k \) is \( p - p_k \), where

\[
(p_k - 1)r + 1 \leq p_{k-1} \leq p_1r
\]  

(8)

and that

\[
1 \leq p_k \leq p_{k-1} \leq \cdots \leq p_1 \leq p.
\]  

(9)

Hence the output carry of column \( k \) cannot exceed \( p - 1 \). Since \( k \) is arbitrary, the theorem is proved.

ACKNOWLEDGMENT

The author is indebted to L. S. Levy and W. H. Cherry of RCA for ideas leading to the present work.

Constant-Weight Counters and Decoding Trees*

WILLIAM H. KAUTZ†, MEMBER, IRE

Summary—A class of counters is described in which the number of 1's in the flip-flops or register stages composing the counter remains constant as the counter advances from state to state. Simple digital circuit arrangements are described for the design of such counters, which may be used with a particular type of decoding tree as economical ring-type counters, to provide a separate output lead for each state. Some interesting theoretical questions concerning the minimization of these decoding trees are raised and partially answered. Finally, the costs of these counters are compared with one another, and with those of other types of counters, over a continuous range of values of the flip-flop/gate-input cost ratio.

I. INTRODUCTION

A \( N \)-digit ring counter may be viewed as a simple circulating shift register of \( N \) stages; it contains a binary number having only a single 1, the other digits being 0. Such a counter counts with period \( N \) as shift pulses are applied to the register. The \( N \) outputs which indicate the successive states of the counter are derived directly from the stages of the register, as shown in Fig. 1.1

![Fig. 1 — A simple ring counter.](image)

This paper describes some other ring-type counters based on this shift-register viewpoint. These counters are similar to ring counters in that an individual output is provided for each of the counts, but they differ from ring counters in that more than just a single 1 is circulated. The separation of the two or more 1's is varied on successive circulations by means of auxiliary logical connections between the stages of the register. As a re-

---

* Received by the PGEC, December 7, 1959.
Stanford Research Institute, Menlo Park, Calif.

† The interstage AND gates shown in this and following figures are intended to be symbolic only, and represent any of several known techniques for shifting binary digits from one register stage to another in response to a counting pulse \( e \).