YO884-0064 RRS P200 52-000

E. R. Barnes and V. G. Oklobdzija

IBM Technical Disclosure Bulletin Vol. 28 No. 3 August 1985

85A061864

NEW SCHEME FOR VLSI IMPLEMENTATION OF FAST ALU

FIG.1

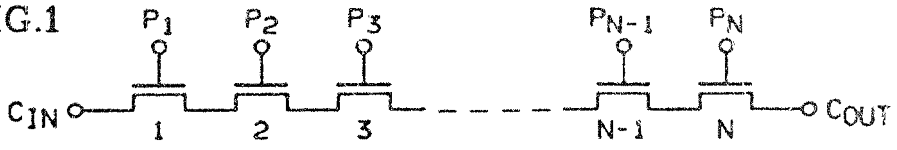
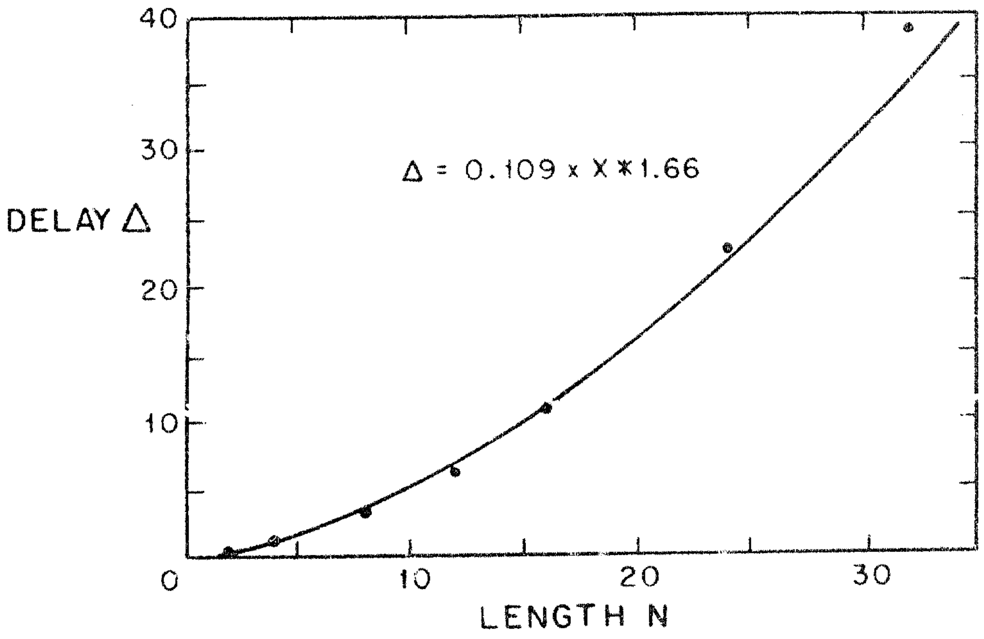


FIG.2



A fast ALU (arithmetic and logic unit) architecture is described which is implemented using an area-efficient method for propagation of the carry signal in a very large-scale integration (VLSI) pass-transistor arithmetic-logic unit (ALU) network using carry skip to obtain speed-up. Based on the signal propagation properties of n-channel metal-oxide-semiconductor (MOS) transistor chains, an algorithm which determines the best possible division into groups of variable size for the given propagation properties is suggested. Buffering is used in between each group of bits grouped by the algorithm. This approach yields higher speed than the traditional Manchester-Carry Chain, which utilizes buffering.

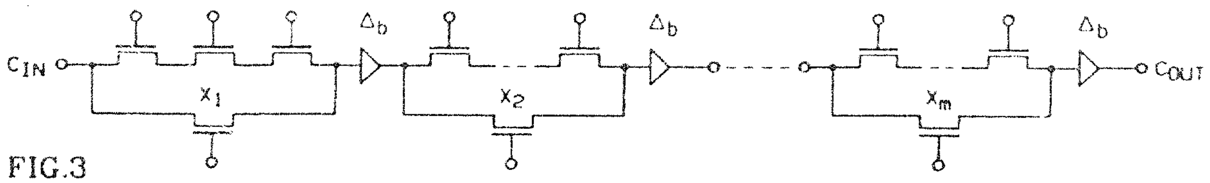


FIG.3

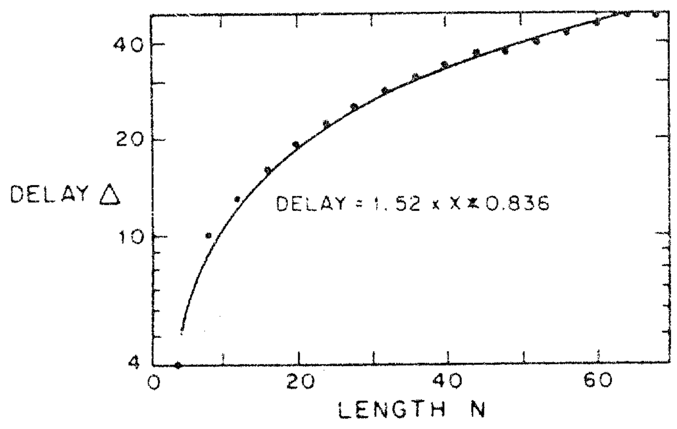


FIG.4

FOR:

N=16
x: 2-3-3-3-3-2
DELAY=16

N=32
x: 2-3-3-4-4-4-4-3-3-2
DELAY=28

N=48
x: 2-3-3-4-4-5-6-5-4-4-3-3-2
DELAY=37

N=64
x: 3-3-4-4-5-5-5-6-5-5-5-4-4-3-3
DELAY=49

It is shown that the delay of such an implementation does not depend linearly on the size of the adder in the number of bits, as do traditional carry-skip, but on the power of 0.836, which is in between carry-lookahead and carry-skip (traditionally implemented).

Conventionally, the pass-transistor chain is used for propagation of the carry signal in the VLSI implementation of an ALU unit. Buffering is used in between each group to amplify the carry signal and restore its level. Inside the groups, carry propagation is not linearly dependent on the size of the group because there is no buffering between the bit slices. Because of this, dividing the carry chain into groups of equal size does not yield an efficient implementation [1].

The development of the variable-length carry skip was determined as follows: In a chain of N pass transistors connected together (Fig. 1), the time taken for the signal to propagate from one end to the other is not linearly dependent on the length of the chain. Therefore, the assumption of linear dependence taken by the carry-skip scheme [2] does not hold for the pass-transistor network [3]. Theoretically, this time is $t = N^2$. By simulation of the carry chain consisting of MOS field-effect transistors it was found that this dependence to be:

$$t = N^p$$

for the particular case considered, where p is determined to be 1.6. This dependence is shown in Fig. 2.

With the new method (Fig. 3), a carry chain of length N is divided into sections: $x_1, x_2, \ldots, x_m$. Each section is terminated with a buffer, which serves to amplify the signal and restore its level but introduces a delay Δ_b. Skip sections are implemented over the group x_i. The optimum sizes of x_i are determined by the new algorithm, and proof that this division is the best possible is also demonstrated. The new delays for the various lengths of the carry chain (ALUs of various sizes) are presented in Fig. 4.

With N denoting the number of bits in the numbers to be added, let $x_1, x_2, \ldots, x_m$ denote the sizes of the groups into which the bits are divided. For simplicity it is assumed that

$$(1) \qquad x_i = x_{m-i+1}, \quad i \leq m - i + 1.$$

Note that this assumption does not exclude the case $x_1 = x_2 \ldots = x_m$ of groups of equal size. Thus, the optimization takes place over a larger class of groups than is usually considered.

The time required for a carry signal to ripple through x bits is proportional to x^p where $1 \leq p \leq 2$. It was found by simulation that for particular MOSFET technology that we examined, $p = 1.6$. The longest path for a carry signal generated in group i and terminating in group j is given by:

$$(2) \qquad (x_i - 1)^p + (j - i)\Delta_b + (j - i - 1) + (x_j - 1)^p$$

where Δ_b is the buffer delay for the buffers between the groups of bits. It is assumed that Δ_b is a positive integer.

Then, let Δ denote the length of the longest path over which a carry signal can propagate. The quantities $\Delta, x_1, \ldots, x_m$ then satisfy

$$(3) \qquad \sum_{i=1}^{m} x_i = N$$

$$(x_i - 1)^p + (j - i)\Delta_b + (j - i - 1) + (x_j - 1)^p \leq \Delta \text{ for}$$
$1 \leq i < j \leq m$.

The optimal group sizes for the carry chain can be found by choosing the x_i's to minimize Δ subject to these constraints. Since the quantities $\Delta, x_1, \ldots, x_m$ are restricted to positive integers, this is a nonlinear integer programming problem. It is complicated further by the fact that m is unknown. However, the entire optimization problem simplifies because of the assumption of symmetry $x_i = x_{m-i+1}$ for $i \leq m-i+1$. To see this, the following theorem is needed:

Theorem. If m = 2k is even, the conditions (3) hold if and only if

$$\sum_{i=1}^{m} x_i = N$$

and

(4a) $(x_i - 1)^p + (m - 2i + 1)\Delta_b + (m - 2i) + (x_{m-i+1} - 1)^p \leq \Delta$ for

$i = 1, \ldots, k$.

If m = 2k + 1 is odd, then conditions (3) hold if, and only if, in addition to (4a), we have

(4b) $(x_i - 1)^p + (k + 1 - i)\Delta_b + (k - i) + (x_{k+1} - 1)^p \leq \Delta$ for

$i = 1, \ldots, k$.

It is clear that, for a fixed m, Δ can be minimized subject to (3) by minimizing Δ subject to (4a) for m even or by minimizing Δ subject to (4a) and (4b) for m odd. The advantage of this is that each of the constraints (4a) involves only a single variable because of the assumption $x_i = x_{m-i+1}$, $i \leq m - i + 1$. This makes possible the following

simple algorithm for minimizing Δ subject to (4a), or (4a) and (4b), for a fixed value of m.

Case I m = 2k is even. Since $\sum_{i=1}^{m} x_i = 2\sum_{i=1}^{k} x_i = N$, this case

can only occur if N is even.

Step 0. Take $\Delta = (m - 1)\Delta_b + m - 2$. (4a) shows that the optimal Δ must be greater than or equal to this value.

Step 1. Define x_i to be the largest integer satisfying

$$2(x_i - 1)^p + (m - 2i + 1)\Delta_b + (m - 2i) \leq \Delta,$$
$$i = 1, \ldots, k.$$

Step 2. Set $x_{m-i+1} = x_i, i = 1,\ldots,k$.

If $\sum_{i=1}^{m} x_i < N$ increase Δ by 1 and repeat Step 1.

If $\sum_{i=1}^{m} x_i = N$, the current values of $\Delta, x_1,\ldots,x_m$ are optimal.

Stop

If $\sum_{i=1}^{m} x_i > N$, go to Step 3.

Step 3. The current value of Δ is optimal.

Examine the values

$$2(x_i - 1)^p + (m - 2i + 1)\Delta_b + (m - 2i),\ i = 1,\ldots,k.$$

Reduce the x, call it x_j, corresponding to the largest of these quantities by 1 and go to Step 2.

This algorithm produces the smallest integer value of Δ with which the conditions (4a) can be satisfied.

<u>Case II</u> $m = 2k + 1$ is odd.

In this case the algorithm is the same as before, except that Step 1 is replaced by the following:

Step 1. Define x_i to be the largest integer satisfying

$$2(x_i - 1)^p + (m - 2i + 1)\Delta_b + (m - 2i) \le \Delta,$$

$i = 1,\ldots,k$. If N is odd, define x_{k+1} to be the largest odd integer satisfying each of the inequalities

$$(x_i - 1)^p + (k + 1 - i)\Delta_b + (k - i) + (x_{k+1} - 1)^p \le \Delta,$$

$i = 1,\ldots,k$. If N is even, take x_{k+1} to be the largest even integer satisfying these conditions. Again it is easy to verify that this algorithm minimizes Δ subject to (4a) and (4b) for a fixed odd value of m.

A procedure for determining the optimal carry chain is now available. Beginning with $m = 2$, find the smallest values of Δ corresponding to a chain containing m groups for $m = 2,3,\ldots,m^*$. Choose a value of m which minimizes Δ over this range. The corresponding sizes $x_1,\ldots x_m$ are optimal.

m^* is defined as follows. Let $\Delta_1,\Delta_2,\ldots,$ denote the values of Δ corresponding to $m = 1,2,\ldots m^*$ is the smallest value of m satisfying

$$(m^* - 1)\ \Delta_b + m^* - 2 \leq \Delta_j$$

for some $i \leq j \leq m^*$. To see that the optimal Δ occurs in the sequence $\Delta_1,\Delta_2,\ldots,\Delta_{m^*}$, note that for $m > m^*$

$$\Delta \geq (m - 1)\Delta_b + m - 2 > \Delta_j.$$

References

[1] C. Mead and L. Conway, Introduction to VLSI Systems, Addison-Wesley Publishing Company, October 1980.

[2] T. Kilbuon, D. B. G. Edwards and D. Aspinall, "Parallel Addition in Digital Computers: A New Fast "Carry" Circuit," Proc. IEEE 106, 464 (September 1959).

[3] Kai Hwang, Computer Arithmetic: Principles of Architecture and Design, John Wiley and Sons, 1979.