7th Symposium on Computer Arithmetic

Proceedings
1985

Published by IEEE Computer Society Press
1109 Spring Street
Suite 300
Silver Spring, MD 20910

IEEE Catalog Number 85CH2146-9
Library of Congress Number 85-60193
IEEE Computer Society Order Number 632
ISBN 0-8186-0632-0 (Paper)
ISBN 0-8186-4632-2 (Microfiche)
ISBN 0-8186-8632-4 (Casebound)

Order from: IEEE Computer Society
Post Office Box 80452
Worldway Postal Center
Los Angeles, CA 90080

IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854

IEEE THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Foreword

The 7th Symposium on Computer Arithmetic (ARITH–7) was held at the University of Illinois in Urbana, Illinois, on June 4-6, 1985. A total of 47 papers were accepted for presentation. These papers covered theoretical foundations, arithmetic algorithms, processor architectures, hardware and software implementations, error controls and advanced applications of high-speed and high-accuracy arithmetic systems in modern digital computers. This year's symposium is divided into 11 technical sessions. The sessions include efficient adders and ALU designs, fast multipliers and dividers, floating-point arithmetic, systolic arithmetic schemas, directions in computer arithmetic, elementary function evaluation, rational and residue arithmetic, signal and image processing, large-scale scientific computations, fault-tolerant arithmetic, and new arithmetic systems.

Contributions to this Proceedings were made by 82 authors from almost all parts of the world. However, the majority of the papers received were from North America. All the previous symposia on computer arithmetic were held in the U.S.; however, ARITH–6 was held in Aarhus, Denmark in 1983. The interval between successive meetings in the series has been reduced from three years to two years in the last three meetings (1981, 1983, and 1985). This prompts us to suggest that the future symposia will be held alternately between North America and other continents every two-year period.

We would like to thank all the authors, referees, and program committee members for their contributions and timely efforts in promoting the scientific quality of this Symposium. Listed below are the referees and individuals whose efforts should be particularly acknowledged.

Agrawal, D.P.	Hennessy, John L.	Ramakrishnan, I.V.
Aiso, Hideo	Irwin, Mary Jane	Rao, T.R.N.
Atkins, Daniel E.	Jayakumar, R.	Robertson, James
Avižienis, Algirdas	Kong, Vivian	Swartzlander Jr., Earl
Bohlender, Gerd	Kornerup, Peter	Tanimoto, Steven L.
Brent, R.P.	Kulisch, Ulrich	Taylor, Fred J.
Capello, Peter R.	Kung, H.T.	Torng, H.C.
De Mori, Renato	Li, H.F.	Tseng, Ping-Shing
Despain, Alvin	Matula, David	Tu, Paul
Ercegovac, Milos D.	Milutinovic, Velkjo	Uhr, Leonard
Farhanz, Massoud	Ni, Lionel M.	Ullman, Jeffrey D.
Fisher, P.D.	Ong, Shauchi	Wakerly, John F.
Flynn, Michael J.	Osteeby, Ole	Woo, Bob Y.
Garner, Harvey L.	Piestrak, Stanislaw	Xu, Zhiwei

Thanks are also due to the IEEE Computer Society, the Technical Committee on Computer Architecture, and the University of Illinois for sponsoring and hosting this symposium. With the consent of the Program Committee, we dedicate the **Proceedings of the 7th Symposium on Computer Arithmetic** to Professor James E. Robertson for his pioneering and continuing contributions to this area. We believe that the various generations of his students and all participants of this symposium will join us in wishing Professor Robertson good health and many happy returns to join us in the future symposia on computer arithmetic.

Kai Hwang
University of Southern California

Daniel D. Gajski
Ahmed Sameh
University of Illinois

Conference Committee

General Chairman:
Daniel D. Gajski, *University of Illinois*

General Cochairman:
Ahmed Sameh, *University of Illinois*

Publicity Chairwoman:
Mary J. Irwin, *Pennsylvania State University*

Program Chairman:
Kai Hwang, *University of Southern California*

Program Committee:
Hideo Aiso, *Keio University*
Daniel Atkins, *University of Michigan*
Algirdas Avižienis, *University of California, Los Angeles*
Milos Ercegovac, *University of California, Los Angeles*
Michael Flynn, *Stanford University*
Harvey Garner, *University of Pennsylvania*
Mary Irwin, *Pennsylvania State University*
Peter Kornerup, *Aarhus University*
Ulrich Kulisch, *University of Karlsruhe*
H.T. Kung, *Carnegie-Mellon University*
David W. Matula, *Southern Methodist University*
Lionel Ni, *Michigan State University*
T.R. Rao, *University of Southwestern Louisiana*
Earl Swartzlander, Jr., *TRW*
Fred Taylor, *University of Florida*
D.P. Agrawal, *North Carolina State University*
Bob Y. Woo, *Weitek Corporation*

History of IEEE Meetings on Computer Arithmetic

1. June 16, 1969	One-day workshop preceding the IEEE Computer Group Conference Minneapolis, MN Organized by R.R. Shively
2. May 15-16, 1972	One-and-a-half day symposium University of Maryland, College Park, MD Organized by H.L. Garner and D.E. Atkins
3. Nov. 19-20, 1975	Two-day symposium Southern Methodist University, Dallas, TX Organized by T.R.N. Rao and D.W. Matula
4. Oct. 25-27, 1978	Two-and-a-half day symposium UCLA, Los Angeles, CA Organized by A. Avižienis and M.D. Ercegovac
5. May 18-19, 1981	Two-day symposium plus tutorial University of Michigan, Ann Arbor, MI Organized by K.S. Trivedi and D.E. Atkins
6. June 20-22, 1983	Three-day symposium Aarhus University Aarhus, Denmark Organized by T.R.N. Rao and P. Kornerup
7. June 4-6, 1985	Two-and-a-half day symposium University of Illinois Urbana, Illinois Organized by Kai Hwang, Daniel D. Gajski, and Ahmed Sameh

Table of Contents

Session 1
Efficient Adders and ALU Designs

Chair
Daniel Atkins

SOME OPTIMAL SCHEMES FOR ALU IMPLEMENTATION IN VLSI TECHNOLOGY

Vojin G. Oklobdzija and Earl R. Barnes

IBM T.J.Watson Research Center
P.O.Box 218
Yorktown Heights, NY 10598

ABSTRACT

An efficient scheme for carry propagation in an ALU implemented in n-MOS technology is presented. An algorithm that determines the optimum division of the carry chain of a parallel adder for various data path sizes is developed. This yields an implementation of a fast ALU which due to its regular structure occupies a modest amount of silicon. The speed of the implementation described is comparable to the carry look-ahead scheme. Our method is based on the optimization of the carry path implemented in n-MOS technology but the results can be applied to other technologies.

1. INTRODUCTION

An efficient implementation of an ALU in VLSI technology depends on many parameters. We consider an *efficient* implementation to be one which is fast, of a small and regular area, and low power. In many VLSI designs achieving the ultimate speed is not always important goal especially if this is achieved by consuming excessive area and power. Therefore Carry-Lookahead (CLA) scheme is not very attractive for VLSI implementations where area, regularity of structure and power are important. Also the determining factor, in case of an ALU, is wheather it is part of a critical path or not.

In this paper we considered Carry-Skip (CSA) scheme [1],[2],[3],[4] because it met our objective of reasonable performance achieved with a relatively small and regular area. This adder is in essence a Carry Lookahead for which the carry-generate portion which consumes a large amount of logic, has been eliminated. As in a Carry Lookahead adder the bits to be added are divided into groups. A circuit is provided for detecting when a carry signal entering a group will ripple through the group. When this condition is detected, the carry is allowed to skip over the group. Carry-Skip Adder (CSA) does not require excessive amount of logic (area) and the "skip" portion of the logic can be added to the existing carry chain of Ripple-Carry Adder (RCA), therefore not disturbing the inherently regular bit-slice implementation of an RCA (Fig.1).

The power requirements of a CSA are considerably lower than that of a CLA-ALU. In this paper we show how the carry chain in a CSA can be optimized to yield better speed which in the case of a multi-level optimized CSA-ALU can even outperform the speed of a CLA divided into the groups of constant size.

Lehman and Burla [3] studied a design of a CSA and suggested varying the size of the groups. By varying the sizes of the groups one can influence the maximum delay a carry signal can experience in propagating through the adder. Lehman and Burla [3] posed the problem of determining the optimal group sizes for minimizing the maximum delay [9]. They gave a heuristic method for obtaining economical group sizes. However, they did not solve the problem of determining optimal group sizes. For example, for a 48 bit adder they gave the group sizes 4 5 6 7 8 8 7 6 5 4 yielding the maximum delay of 14 [3]. We show that, under the same assumptions an optimal subdivision results in a delay of 12.

They also discussed the problem of achieving even faster addition by allowing carry signals to skip over blocks of groups. The problem of optimizing the carry chain is now complicated by having to choose both the optimal number of blocks and the optimal sizes of groups within blocks. Some rules for choosing economical block and group sizes are given [3]. However, the problem of determining the optimal sizes remains unsolved.

In this paper we consider the problem of designing a carry-skip adder in FET technology and give some optimal solutions. Actually, our solutions are more general in that we generally assume

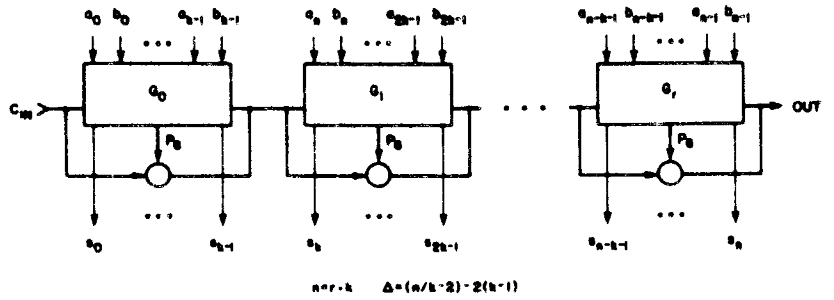

a_0 b_0 a_{k-1} b_{k-1}
a_k b_k a_{2k-1} b_{2k-1}
a_{n-k-1} b_{n-k-1} a_{n-1} b_{n-1}
G_0
G_1
G_r
C_{in}
OUT
P_G
s_0 s_{k-1} s_k s_{2k-1} s_{n-k-1} s_n
$n = r \cdot k$ $\Delta = (n/k - 2) - 2(k-1)$

Fig.1 Carry-Skip Adder

that the time required for a carry signal to skip over a group of bits is longer than the time required for the carry to ripple through a single bit. This assumption is relevant for adders designed in n-MOS technology. Lehman and Burla assumed their adder to be designed using discrete components where these two times are equal. Our analysis will include their problem as a special case.

2. DIVIDING THE ADDER INTO GROUPS

Let n denote the number of bits in a carry skip adder and let m denote the number of groups into which the bits are divided. Let $x_1, \dots, x_m$ denote the sizes of the groups beginning with the most significant bit. Let T denote the time required for a carry signal to skip over a group of bits. To be precise we should write $T = T(x)$ to indicate that T depends on the size x of the group over which the carry is skipped. However, T changes very slowly with x over the range of group sizes that concern us. So we assume that T is constant.
For a given n, the following three-step procedure gives an optimal way of dividing an n bit adder into groups of bits.

Procedure 2:

2(i) Let m be the smallest positive integer such that

$$n \le m + \frac{1}{2}mT + \frac{1}{4}m^2T + (1 - (-1)^m)\frac{T}{8}. \qquad (1)$$

2(ii) Let

$$y_i = \min\{1 + iT,\ 1 + (m + 1 - i)T\}, \quad i = 1, \dots, m$$

and construct a histogram whose i-th column has height y_i. For example, for $T = 3$ and $n = 48$, we have $m = 7$ and the following histogram

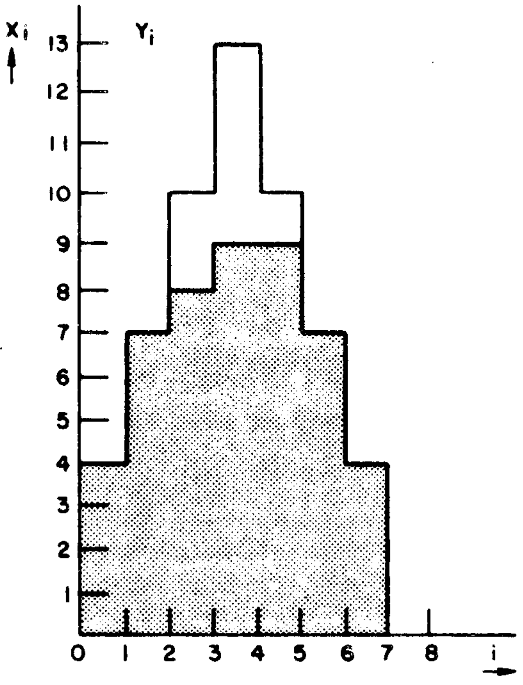


Fig.2. Histogram of segments y_i, x_i

2(iii) It is easily verified that the area of the histogram in (ii) is

$$m + \frac{1}{2}mT + \frac{1}{4}m^2T + (1 - (-1)^m)\frac{T}{8} \ge n$$

so these are at least n unit squares in the histogram. Starting with the first row, shade in n of the squares, row by row. Let x_i denote the number of shaded squares in column i of the histogram, $i = 1, \dots, m$.

Then $x_1, \dots, x_m$ is an optimal division of the adder. The maximum delay of a carry signal is mT.

Example 1. For a 48 bit adder we have, from Figure 2, $x_1 = x_7 = 4$, $x_2 = x_6 = 7$, $x_3 = 8$ and $x_4 = x_5 = 9$.

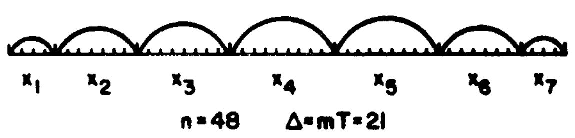


Fig.3. Carry chain of a 48-bit adder: n=48, mT=21

The maximum delay is experienced by a signal generated in the second bit position and terminating in the 47th bit position. The delay is $mT = 21$.

Example 2. Consider a 54 bit adder. From 2(i) we see that again $m = 7$. If we shade 54 squares in Figure 2, we see that

$$x_1 = x_7 = 4,\ x_2 = x_6 = 7,\ x_3 = x_5 = 10 \text{ and } x_4 = 12$$

yields an optimal division of the adder. Again the maximum delay is $mT = 21$.

Example 3. Consider a 64 bit adder. From 2(i) we compute $m = 8$. The corresponding histogram is shown in Figure 4. The optimal group sizes are:

$$x_1 = x_8 = 4,\ x_2 = x_7 = 7,\ x_3 = x_6 = 10,\ x_4 = x_5 = 11.$$

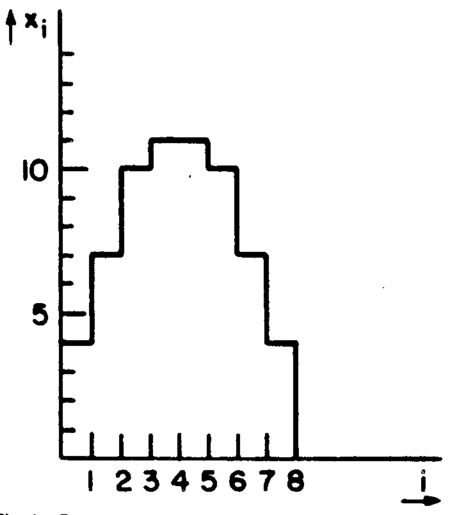


Fig.4. Segment histogram for a 64-bit adder: m=8

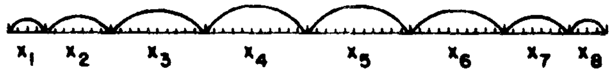


Fig.5. Carry chain of a 64-bit adder: m=8, mT=24

The delay of the longest signal is $mT = 24$.

Proof of Optimality We are going to prove optimality of the division of the carry chain described in 2(i)-(iii). First we need a lemma.

Lemma 1. *When the bits of a carry skip adder are grouped according to the scheme (i)-(iii), the maximum propagation time of a carry signal is mT*

Proof. The carry generated at the 2nd bit position and terminating at the (n-1) clearly has propagation time mT. We must show that any other signal has propagation time smaller than or equal to mT. Consider a signal originating in the ith group and terminating in the jth, $i < j$. Denote its propagation time by P. Clearly

$$P \le (x_i - 1) + (j - i - 1)T + (x_j - 1).$$

By construction, $x_i \le \min\{1 + iT,\ 1 + (m + 1 - i)T\}$ for each i so

$$P \le \min\{1 + iT,\ 1 + (m + 1 - i)T\} + \min\{1 + jT,\ 1 + (m + 1 - j)T\} + (j - i - 1)T - 2.$$

There are three cases to consider.

1. First assume

$$\min\{1 + iT,\ 1 + (m + 1 - i)T\} = 1 + iT$$

and

$$\min\{1 + jT,\ 1 + (m + 1 - j)T\} = 1 + jT.$$

In this case $1 + jT \le 1 + (m + 1 - j)T \Rightarrow 2jT - T \le mT$. It follows that

$$P \le 1 + iT + 1 + jT + (j - i - 1)T - 2 = 2jT - T \le mT.$$

2. Now assume that

$$\min\{1 + iT,\ 1 + (m + 1 - i)T\} = 1 + iT$$

and

$$\min\{1 + jT,\ 1 + (m + 1 - j)T\} = 1 + (m + 1 - j)T.$$

In this case we have

$$P \le 1 + iT + 1 + (m + 1 - j)T + (j - i - 1)T - 2 = mT.$$

3. Finally, assume that

$$\min\{1 + iT,\ 1 + (m + 1 - i)T\} = 1 + (m + 1 - i)T$$

and

$$\min\{1 + jT,\ 1 + (m + 1 - j)T\} = 1 + (m + 1 - j)T.$$

It follows that

$$P \le 1 + (m + 1 - i)T + 1 + (m + 1 - j)T + (j - i - 1)T - 2 = 2mT - (2iT - T) \le 2mT - mT = mT.$$

This completes the proof of the lemma.

Lemma 2. *Let Δ denote the maximum delay of a carry signal in a n bit carry skip adder with group sizes chosen optimally.* Then

$$(m - 1)T \le \Delta \le mT.$$

Proof. Since we have exhibited a division of the carry chain into groups in such a way that the maximum delay of a carry signal is mT we clearly have $\Delta \le mT$.

Let $x_1, x_2, \ldots, x_r$ denote the optimal group sizes corresponding to Δ. For the moment assume that $r = 2k$ is even. By considering carries originating in group i and terminating in group $r - i + 1$, $i = 1, \ldots, k$, we deduce the following system of inequalities.

$$\begin{array}{rcccccl}
(x_1 - 1) & + & (r - 2)T & + & (x_r - 1) & \le & \Delta \\
(x_2 - 1) & + & (r - 4)T & + & (x_{r-1} - 1) & \le & \Delta \\
\vdots & & & & & & \vdots \\
(x_k - 1) & + & (r - 2k)T & + & (x_{k+1} - 1) & \le & \Delta \\
 & & rT & & & \le & \Delta
\end{array}$$

If we add these inequalities and use the fact that $\sum_{i=1}^{r} x_i = n$, we obtain the inequality

$$n - 2k + (k + 1)rT - k(k + 1)T \le (k + 1)\Delta$$

which simplifies to

$$\frac{n - 2k}{k + 1} + kT \le \Delta.$$

The left hand side of this inequality assumes its minimum value at

$$k + 1 = \frac{\sqrt{n + 2}}{T}.$$

It follows that

$$\Delta \ge -(T + 2) + \sqrt{4nT + 8T}. \qquad (2)$$

Assume now that $r = 2k + 1$ is odd. We then have the system of inequalities

$$\begin{array}{rcccccl}
(x_1 - 1) & + & (r - 2)T & + & (x_r - 1) & \le & \Delta \\
(x_2 - 1) & + & (r - 4)T & + & (x_{r-1} - 1) & \le & \Delta \\
\vdots & & & & & & \vdots \\
(x_k - 1) & + & (r - 2k)T & + & (x_{k+1} - 1) & \le & \Delta \\
 & & (x_{k+1} - 1) & + & kT & \le & \Delta,
\end{array}$$

which implies that

$$\frac{n - 2k - 1}{k + 1} + kT \le \Delta.$$

By minimizing the left hand side of this inequality with respect to k we find that

$$\Delta \ge -(T + 2) + \sqrt{4nT + 4T}. \qquad (3)$$

By comparing this with (2) we see that (3) holds in all cases.

We will now produce an upper bound on mT. Since m is the smallest positive integer satisfying (1) we have

$$n > (m - 1) + \frac{1}{2}(m - 1)T + \frac{1}{4}(m - 1)^2T + (1 - (-1)^{m-1})\frac{T}{8}.$$

Since each size of this inequality is an integer, we can increase the right hand side by 1 to obtain

$$n \geq m - \frac{1}{4}T + \frac{1}{4}m^2T + (1 - (-1)^{m-1})\frac{T}{8}.$$

Solving this inequality for mT gives

$$mT \leq -2 + \sqrt{4nT + 4 + T^2 - (1 - (-1)^{m-1})\frac{T^2}{2}} \quad (4)$$

Combining this with (2) and (3) gives

$$mT - \Delta \leq \begin{cases} T + \dfrac{T^2 - 8T + 4 - (1 - (-1)^{m-1})\frac{T^2}{2}}{\sqrt{4nT + 8T} + \sqrt{4nT + 4 + T^2 - (1 - (-1)^{m-1})\frac{T^2}{2}}}, & r\text{-even} \\ T + \dfrac{T^2 - 4T + 4 - (1 - (-1)^{m-1})\frac{T^2}{2}}{\sqrt{4nT + 4T} + \sqrt{4nT + 4 + T^2 - (1 - (-1)^{m-1})\frac{T^2}{2}}}, & r\text{-odd} \end{cases} \quad (5)$$

For n sufficiently large we have $mT - \Delta < T + 1$ and since $mT - \Delta$ is an integer, $mT - \Delta \leq T$. This completes the proof of the lemma.

Theorem 1 *The scheme 2(i)-2(iii) given above for dividing the bits of a carry skip adder into groups is optimal for* $2 \leq T \leq 7$.

Proof. Assume the scheme is not optimal and let Δ be the maximum delay corresponding to an optimal division of the bits into groups. Assume there are r groups in the optimal division. Since a carry in signal to the least significant bit group can skip over each group we have $rT \leq \Delta \leq mT$ so $r \leq m$. If $r = m$ then $\Delta = mT$ and the theorem holds by Lemma 1. If $r = m - 1$ m and r have different parities and it follows from (5) that $mT - \Delta < T$ for $2 \leq T \leq 7$ so that $\Delta > (m - 1)T = rT$. This means that a signal which skips over each of the r groups has delay less than the maximum Δ. Similarly, if $r < m - 1$, $\Delta \geq (m - 1)T > rT$ so that a signal which skips over each group has delay $< \Delta$. It follows that a signal with delay Δ must start in a group i, ripple to the end of this group, then skip over $s < r$ groups and either terminate, or ripple through the first few bits of a group $j > i$. Let x_i and x_j denote the lengths of the ith and jth groups respectively. Assume that i is chosen as small as possible and j as large as possible. A signal originating in group i, rippling to the end of this group and then skipping over the next s group has delay

$$\begin{aligned} \Delta &\leq (x_i - 1) + sT \leq (x_i - 1) + (r - 1)T \\ &\leq (x_i - 1) + (m - 2)T. \end{aligned}$$

Since $\Delta \geq (m - 1)T$ this implies that $x_i \geq T + 1$. Divide group i into two groups such that the group containing the most significant bits has size T. Since the i-th group is the first group in which a signal having maximum delay can originate, this subdivision does not increase the delay of any carry signal of maximum delay. However, it increases the number of groups by 1.

Suppose now that a carry signal originates in a group i, ripples to its end, skips over $s \leq r - 2$ groups and finally ripples through the first few bits of a group j and terminates. We then have

$$\begin{aligned} \Delta &\leq (x_i - 1) + sT + (x_j - 1) \\ &\leq x_i + x_j - 2 + (m - 3)T \end{aligned}$$

So that either $x_i \geq T + 1$ or $x_j \geq T + 1$. This means that we can subdivide one of the groups i,j without increasing Δ. Continuing in this way, we can always increase the number r of group in an optimal division of a carry chain by 1 without increasing Δ if $r < m$. This means that we can arrive at an optimal division of the carry chain into m groups. We must then have $\Delta \geq mT$ which, together with Lemma 2, implies $\Delta = mT$. This completes the proof of the theorem.

3. DIVIDING GROUPS INTO BLOCKS

It is clear that the maximum delay of a carry signal in a carry skip adder can be further reduced if signals are allowed to skip over blocks of groups. We define a block to be an additional path allowing carry signal to skip directly over groups. In this section we will describe an efficient scheme for dividing the carry chain into blocks of groups. We assume that the time required for a carry signal to skip over a block of groups is T_b. Actually, the time required for a carry to skip over a block T_b is slightly longer than the time T_g required to skip over a group. But for the sake of simplifying the analysis we will assume these two times to be equal i.e. $T_b = T_g$. However, our technique extends to the case where $T_g \neq T_b$.

Let M denote the number of blocks into which the groups of bits are divided. Let Δ denote the maximum delay a carry signal can have in an adder divided into M blocks. Clearly, $\Delta \geq MT$. We will show how to choose the blocks such that $\Delta = MT$. We will also show how to choose M for an adder of length n.

Our blocks are chosen in such a way that the maximum delay of a signal originating and terminating in block i and $M + 1 - i$ is iT.

Consider a signal originating in the first of these blocks and terminating in the second. Such a signal will skip over $M - 2i$ blocks and will accordingly have delay $\leq (iT) + (M - 2i)T + iT = MT$ as desired. It follows from our work in Section 2 that in order for a signal originating and terminating in block i to have delay less or equal iT we must choose the length of the ith and $(M + 1 - i)th$ blocks to be less or equal the number of unit squares in a histogram with base of width i. Thus the maximum length of the ith and $(M + 1 - i)th$ blocks must be

$$i + \frac{1}{2}iT + \frac{1}{4}i^2T + (1 - (-1)^i)\frac{T}{8}, \quad i \leq \lceil \frac{M}{2} \rceil.$$

(Here we use the symbol $\lceil I \rceil$ to denote the smallest integer $\geq I$.) It follows that the maximum length of an adder divided into M blocks must be

$$2 \sum_{i=1}^{\lceil \frac{M-1}{2} \rceil} \left\{ i + \frac{1}{2} iT + \frac{1}{4} i^2 T + (1 - (-1)^i) \frac{T}{8} \right\} \tag{3.1}$$
$$+ \frac{(1-(-1)^M)}{2} \left\{ \lceil \frac{M}{2} \rceil + \frac{1}{2} \lceil \frac{M}{2} \rceil T + \frac{1}{4} \lceil \frac{M}{2} \rceil^2 T + (1 - (-1)^{\lceil M/2 \rceil}) \frac{T}{8} \right\}.$$

Thus for a given adder length n, we choose M to be the smallest positive integer such that the expression (3.1) exceeds or equals n. M is then the number of blocks into which our adder must be divided. The formal statement of our algorithm is as follows.

3(i) Choose M to be the smallest positive integer such that

$$n \le 2 \sum_{i=1}^{\lceil \frac{M-1}{2} \rceil} \left\{ i + \frac{1}{2} iT + \frac{1}{4} i^2 T + (1 - (-1)^i) \frac{T}{8} \right\}$$
$$+ \frac{(1-(-1)^M)}{2} \left\{ \lceil \frac{M}{2} \rceil + \frac{1}{2} \lceil \frac{M}{2} \rceil T + \frac{1}{4} \lceil \frac{M}{2} \rceil^2 T + (1 - (-1)^{\lceil M/2 \rceil}) \frac{T}{8} \right\}.$$

3(ii) Form M blocks labeled 1, 2, ..., M, with blocks i and $M + 1 - i$ each containing

$$i + \frac{1}{2} iT + \frac{1}{4} i^2 T + (1 - (-1)^i) \frac{T}{8}$$

bits, $i \le \lceil \frac{M}{2} \rceil$.

This construction is analogous to the construction of the histogram in 2(ii). If necessary, delete bits from the largest blocks in this chain until a total of exactly n bits remain in the M blocks.

3(iii) Treat each of the final blocks in 3(ii) as a complete carry chain and divide it into groups optimally using the algorithms 2(i)-2(iii).

Example 3.1. Consider a 32-bit adder. For $i = 1, 2, 3, \ldots$, and $T = 3$ the numbers

$$i + \frac{1}{2} iT + \frac{1}{4} i^2 T + (1 - (-1)^i) \frac{T}{8}$$

take on the values 4, 8, 15, 22, 32.... respectively. Since

$$32 \le 2\{4 + 8\} + 15$$

we must have $M = 5$ blocks in step 3(ii). These blocks have sizes 4, 8, 15, 8, 4 respectively. If we delete 7 units from the middle block we obtain block sizes 4, 8, 8, 8, 4 which add up to 32. Dividing each block into groups by the procedure 2(i)-2(iii) we obtain the following chain where each group has size 4.

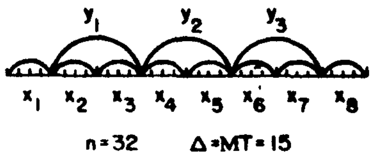


Fig.6. Carry chain of an 32-bit adder: MT=15

The maximum delay of a carry signal is $MT = 15$.

Example 3.2. Consider a 48 bit adder. Again we assume $T = 3$. Since

$$48 \le 2\{4 + 8 + 15\}$$

we must take $M = 6$ corresponding to block sizes 4, 8, 15, 15, 8, 4. The total number of units is 54. So we reduce the size of the two middle blocks by 3 each. This gives block sizes 4, 8, 12, 12, 8, 4 adding up to 48. If we divide each block into groups by the procedure 2(i)-2(iii), each group has size 4. The maximum delay of a carry signal is given by $MT = 18$.

Example 3.3. Consider a 64 bit adder and assume $T = 3$ Since

$$64 \le 2\{4 + 8 + 15\} + 22$$

we take $M = 7$ and start with blocks of sizes 4, 8, 15, 22, 15, 8, 4 respectively. The lengths of these blocks total 76. So we reduce the middle block by 12. The new block sizes are 4, 8, 15, 10, 15, 8, 4. The optimal division of these blocks into groups is given in Figure 7. We could have just

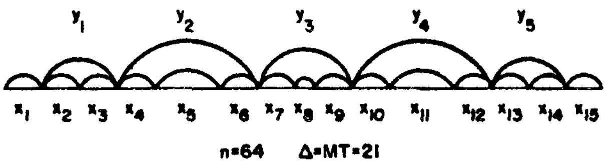

y_1
y_2
y_3
y_4
y_5
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15}
n=64 Δ=MT=21

Fig.7. Carry chain of a 64-bit adder: MT = 21

as well reduced the sizes of the three middle blocks obtaining final blocks of sizes 4, 8, 13, 14, 13, 8, 4. The maximum delay of a carry signal would still be $MT = 21$.

4. COMPARISON

In the literature comparison of schemes for ALU implementation is done mainly on the basis of the number of gates, propagation delay per gate and power consumed per gate [5]. However, if a VLSI implementation of a high-speed ALU is considered, these measures are not easily applied. For example, the propagation delay in terms of number of gates is not an adequate measure unless care is taken to implement the function exactly as specified by its logic (gate) representation. This is often not the case since the function is frequently merged into a group of transistors or implemented by using pass-transistors, precharge or other techniques applyed by the circuit designer in order to minimize delay and power.

In general, if the function is implemented in two levels of logic, the delay is not necessarily smaller than the implementation of the same function in three or more logic levels. This is due to the fact that in n-MOS technology the delay is heavily dependent on several factors:

1. *Gate type* : NOR gates are faster than NAND gates.
2. *Fan-in* : for a NAND gate, the delay is directly proportional to the number of inputs, since inside the n-input NAND gate the signal has to propagate through n-transistors. In case of

a NOR gate, delay is not strongly affected by the number of inputs, and therefore the use of NOR gate is preferable.

3. *Fan-out* : the speed of a gate will be different if the fan-out is larger than in the case of small fan-outs.

4. *Wiring* : speed is also dependent on the length of the wires i.e. "wiring capacitance" and the resistance of the long wires.

4.1 Comparison with Carry-Lookahead Scheme

For the purpose of our analysis we consider as a rough measure of delay the number of FET transistors through which the signal needs to propagate in order to reach the destination point. This seems to be satisfactory for measuring delay through a VLSI FET network. In addition we modify the delay equations for the points known to have either a substantial fan-out or considerable capacitive loading due to the long wires carrying signal to the distant points within the VLSI macro block. For example, we assume that the time required for a carry signal to skip over a block of groups is three times the time to ripple through one bit position of the carry-chain. These assumptions were confirmed by simulation performed on a 32-bit ALU which was actually implemented in n-MOS technology.

The Case T = 1.

In order to compare the speed of our carry skip adder with that of the Carry Lookahead Adder (CLA) we take $T = 1$. This modification is made in order to compare our results with the estimates for the CLA adder which are based on the gate delay calculations in which case it is assumed that each gate level introduces a delay $t_G = 1$ regardless of the gate fan-in or fan-out [5]. This assumption works favorably for CLA when calculating its speed. In practice we expect CLA to show worse performance than estimated while the calculation for our CSA schemes should be more accurate.

For our comparison we consider full-CLA with the groups of size $G_s = 4$. CLA adder of size n=32 bits is shown in the Fig.8 with the delays indicated at each signal input and output to or from the block. Delay calculation for CLA of the sizes n = 16, 32, 48, 64 shows delays for the critical path of the carry signal to be Δ = 6, 8, 10, 10 respectively [5].

First we consider our case where the adder is simply divided into groups of bits. The procedure in Section 2 shows that the maximum delay of a carry signal in an n bit adder is m, where m is the smallest integer satisfying

$$n \le m + \frac{1}{2}m + \frac{1}{4}m^2 + (1 - (-1)^m)\frac{1}{8}$$

The values of a delay m for our method compared with the CLA delays for several values of n are shown in the Table 1.

n	$\Delta(CSA)$	$\Delta(CLA)$
16	6	6
32	9	8
48	12	10
64	14	10

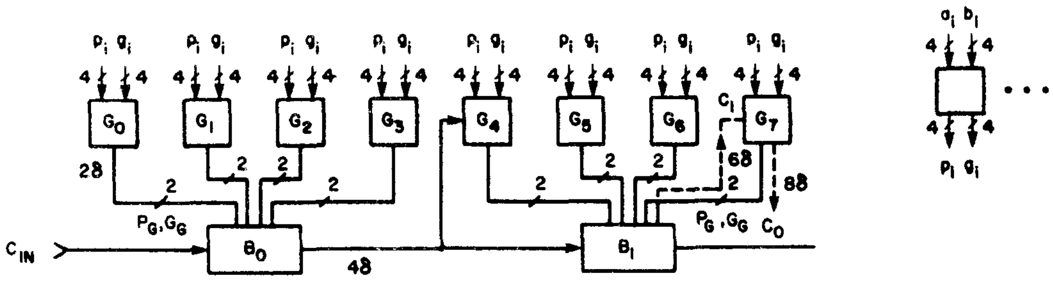
Table 1. Delay comparison with the CLA for the method of Section 2.

Consider now the case where the adder is divided into blocks of groups according to the procedure of Section 3. This algorithm shows that the maximum delay of a carry signal in an n bit carry skip adder is M, the smallest positive integer satisfying

$$n \le 2 \sum_{i=1}^{\lceil \frac{M-1}{2} \rceil} \{i + \frac{1}{2}i + \frac{1}{4}i^2 + (1 - (-1)^i)\frac{1}{8}\}$$
$$+ \frac{(1 - (-1)^M)}{2}\left\{\lceil \frac{M}{2} \rceil + \frac{1}{2}\lceil \frac{M}{2} \rceil + \frac{1}{4}\lceil \frac{M}{2} \rceil^2 + (1 - (-1)^{\lceil M/2 \rceil})\frac{1}{8}\right\}.$$

Recall that $\lceil r \rceil$ is the smallest integer $\ge$ r for any real number r.

Table 2. shows the values of delay M for CSA of Section 3 compared to that of CLA adder corresponding to several values of n.



n=32 , k=4 , Δ=8

Fig.8. Carry Signal Delay of a CLA Adder of the size n=32 bits

n	$\Delta(CSA)$	$\Delta(CLA)$
16	5	6
32	7	8
48	9	10
64	10	10

Table 2. Delay comparison between CSA of Section 3. and CLA

In the first case (Table 1.) we notice that CLA is faster than our method of Section 1. resulting in equal delay for n=16 and ending up with 40% advantage for n=64. However, in the second case (Table 2.) our method of Section 3. shows advantage in speed over CLA of approximately 20% for n=16 and no advantage for n=64.

Our scheme exhibits regularity in fan-in and fan-out throughout the entire carry chain and therefore yields simpler and regular implementation. The amount of logic required to implement our scheme should therefore be smaller then that of CLA. Therefore its implementation in VLSI technology should yield even better results.

5. CONCLUSION

Our goal in implementing this scheme was to achieve a regular structure without using an excessive chip area. Comparison of various ALU implementation schemes for area and power as a parameter show that after some point small improvement in speed is achieved by a large investment in area or power [6],[7]. In our approach we argue that this incremental improvement in speed is diminished due to the overhead in the wiring capacitance and device size. Therefore our approach is to maximize the speed not by increasing the power or adding a substantial amount of logic but rather by optimizing on the path of the signal with the critical timing (carry signal) by designing the ALU around the carry path. The method described in Sections 2. and 3. is especially applicable for the floating-point fraction ALU which is usually of large size n.

REFERENCES

[1] T.Kilburn, D.B.G.Edwards and D.Aspinall, *Parallel Addition in Digital Computers: A New Fast "Carry" Circuit,* Proc. IEE, Vol.106, Pt.B. P.464, September 1959.

[2] M.Lehman, *A Comparative Study of Propagation Speed-up Circuits in Binary Arithmetic Units*, Information Processing, 1962, Elsevier-North Holand, Amsterdam, 1963, p.671.

[3] M.Lehman and N.Burla, *Skip Techniques for High-Speed Carry-Propagation in Binary Arithmetic Units*, IRE Transactions on Electronic Computers, December 1961, p.691.

[4] L.P.Morgan and D.B.Jarvis, *Transistor Logic Using Current Switching and Routing Techniques and its Application to a Fast "Carry" Propagation Adder*, Proc. IEE, pt.B, Vol.106, p.467, September 1969.

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

[6] S.Ong and D.E.Atkins, *A Comparison of ALU Structures for VLSI Technology*, Proc. of the 6th Symposium on Computer Arithmetic, June 20-22, 1983, Aarhus University, Aarhus, Denmark.

[7] Robert K. Montoye, P.W.Cook, *Automatically Generated Area, Power, and Delay Optimized ALUs*, IEEE ISSCC Digest of Technical Papers, February 23-24, 1983, New York.

[8] A.Bilgory and D.D.Gajski, *Automatic Generation of Cells for Recurrence Structures*, Proc. of 18th Design Automation Conference, Nashville, Tennessee 1981.

[9] V.F.Demyanov, V.N.Malozemov, *Vvedenie v Minimaks*, Izdatelstvo Nauka , Fiziko-Matematicheskoi Literaturi, Moskva 1972.

REGULAR, AREA-TIME EFFICIENT CARRY-LOOKAHEAD ADDERS

Tin-Fook Ngai * - *Mary Jane Irwin* **

* Electrical Engineering, Stanford University, CA
** Computer Science, Penn State University, PA

Abstract

For fast binary addition, a carry-lookahead (CLA) design is the obvious choice [OnAt83, BaJM83]. However, the direct implementation of a CLA adder in VLSI faces some undesirable limitations. Either the design lacks regularity, thus increasing the design and implementation costs, or the interconnection wires are too long, thus causing area-time inefficiency and limits on the size of addition. Brent and Kung solved the regularity problem by reformulating the carry chain computation [BrKu82]. They showed that an n-bit addition can be performed in time $O(\log n)$, using area $O(n \log n)$ with maximum interconnection wire length $O(n)$. In this paper, we give an alternative log n stage design which is nearly optimum with respect to regularity, area-time efficiency, and maximum interconnection wire length.

Introduction

Our design follows Mead and Conway's λ design rules [MeCo80]. We assume two layers of interconnect (either metal, silicide or polysilicon). The only circuits used in the design are NMOS complex gates [MaJD83] and transmission gates (pass transistors). All interconnection wires are specified to be of length less than L_{max} which is assumed to be a constant for a particular fabrication technology. The time delay for driving a signal along a wire is assumed to be proportional to the length of the wire. Thus, constant propagation delay can be achieved by having the channel width of the driving transistor proportional to the length of the wire [BiPP81, Ngai84]. Finally, the computation is performed in a planar region.

First we introduce our carry-lookahead adder which is based upon traditional block carry-lookahead techniques [WaFl82]. Then we derive the NMOS design using a negative logic scheme. For the initial design, the external inputs and outputs are assumed to be available where they are required or generated. Finally, we will lift this assumption and consider the practical input/output problem in depth.

The Carry-lookahead Scheme

Let $a_{n-1}a_{n-2} \cdots a_0$ and $b_{n-1}b_{n-2} \cdots b_0$ be two n-bit binary numbers with a sum of $s_{n-1} \cdots s_0$. The carry-lookahead scheme computes the s_i's by

$$c_{i+1} = g_i + p_i c_i$$
$$s_i = a_i \oplus b_i \oplus c_i \qquad \text{for } i = 0,1,\ldots,n-1$$

where

$g_i = a_i b_i$ (*carry generate*)

$p_i = a_i + b_i$ (*carry propagate*)

$+$ denotes ***logical or***

xy denotes ***x and y***

$\oplus$ denotes ***exclusive or***

It is easy to show that

$$c_{i+m+1} = g_{i+m} + \sum_{l=0}^{m-1}\left[\prod_{j=l+1}^{m} p_{i+j}\right] g_{i+l} + \left[\prod_{j=0}^{m} p_{i+j}\right] c_i .$$

For large m, the above carry computation is difficult to implement due to the practical limitations on fan-in and fan-out. In order to reduce the complexity, it is common practice to group carries into blocks [WaFl82]. The block carry scheme is

$$c'_{i+1} = g'_i + p'_i c'_i$$
$$c_{ri} = c'_i$$
$$c_{ri+m+1} = g_{ri+m} + \sum_{l=0}^{m-1}\left[\prod_{j=l+1}^{m} p_{ri+j}\right] g_{ri+l} + \left[\prod_{j=0}^{m} p_{ri+j}\right] c'_i \qquad \text{for } 0 \le m \le r-2$$

where

$$g'_i = g_{ri+(r-1)} + \sum_{l=0}^{r-2}\left[\prod_{j=l+1}^{r-1} p_{ri+j}\right] g_{ri+l}$$

(***block carry generate***),

$$p'_i = \prod_{j=0}^{r-1} p_{ri+j}$$ (*block carry propagate*)

r is the blocking factor

The same technique can be applied iteratively to compute the block carries. This scheme is illustrated in Figure 1 for a 27-bit CLA adder using a blocking factor of three in three levels.