A COMPARISON OF ALU STRUCTURES FOR VLSI TECHNOLOGY

S. Ong

IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598

and

D. E. Atkins

Computer Research Laboratory
Department of Electrical and Computer Engineering
The University of Michigan
Ann Arbor, Michigan 48109

INTRODUCTION

Although many of the basic techniques of computer arithmetic have been known since the earliest days of electronic computing, there is a continuing need to re-evaluate them in the context of developments in VLSI circuit technology. Furthermore, recent work in complexity of algorithms, particularly the solution of recurrence relations, suggests new candidate structures for generating the carry vector and raises the questions as to their practicality in modern logic design practice.

This paper examines three classes of approach to the design of an arithmetic and logic unit (ALU) typical of a modern single-chip NMOS microprocessor. The alternate structures are based upon 1) the precharged Manchester carry ripple chain [1], 2) standard multi-level carry look-ahead [2],[3], and 3) structures which arise as special case of recurrence solvers [4],[5]. The structures are compared with respect to time, space, and power requirements based upon simulation of a 1 μm NMOS FET technology [6] enhanced by the polycide technology [7],[8].

OUTLINE OF THE GENERAL STRUCTURE

Consider the block diagram in Fig. 1 which combines four N-bit operands, A, $\overline{A}$, B and $\overline{B}$, to form S as the result of an ALU operation on A and B. The *preprocessing block* consists of two general logic function blocks: one performs the logic operations (including as a special case, carry propagation), the other computes the carry generation or carry kill. For the Manchester carry ripple approach, the precharge technique is applied to the general logic function block, as shown in Fig. 2.(a), for generating propagate and kill signals. The static general logic function block as shown in Fig. 2.(b) is used for the two other approaches discussed in this paper.

The *carry assimilation block* generates the carry vector. The various structures for the carry assimilation block for these three approaches will be discussed later in more detail. The *result block* combines the carry vector and the output of logic operations to produce an N-bit result S. The result blocks for three approaches are implemented using a static general logic function block shown in Fig. 2.(b).

PRECHARGED MANCHESTER CARRY RIPPLE

Use of the precharged Manchester chain for the carry assimilation is best illustrated in [1]. It takes advantage of the topological simplicity of pass transistor circuits and their relatively rapid propagation of low signals. This structure is

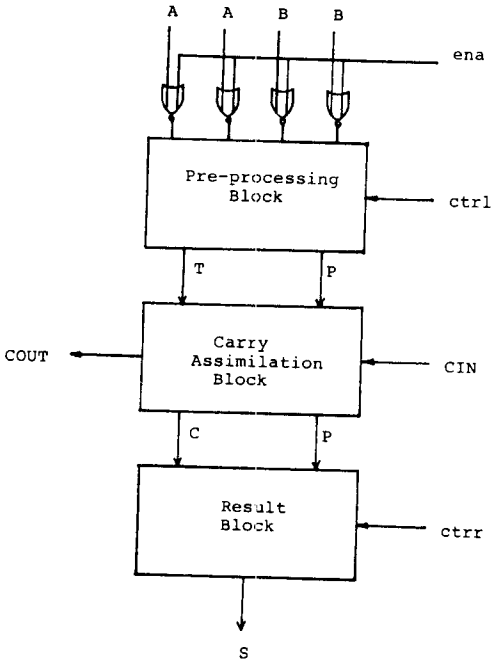

Fig. 1 Block diagram of an ALU.

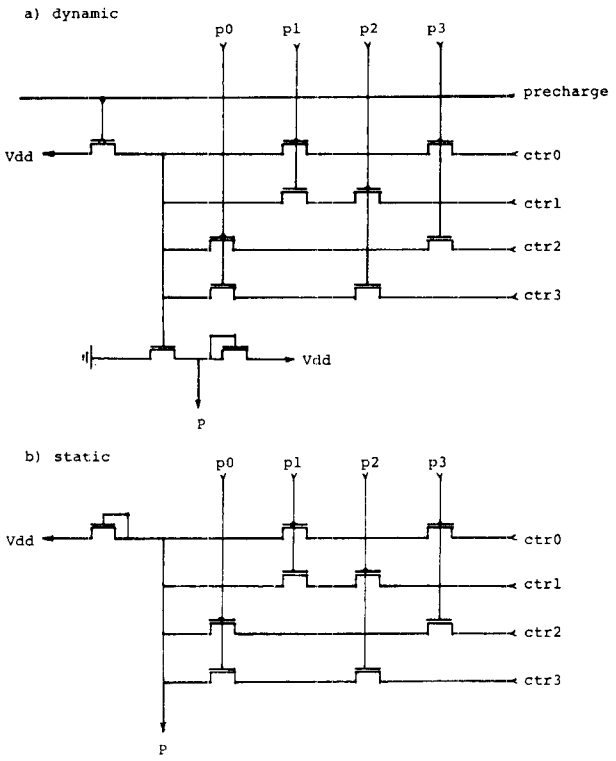

Fig. 2 General logic function block.

inherently repetitive and minimizes parasitics. It is widely believed that the precharged Manchester carry ripple is the best choice for ALU implementation using NMOS technology. Our comparison uses this structure as a baseline reference. To provide a fair and meaningful comparison, we first optimize this baseline design.

Fig. 3 is the carry-chain circuit for the carry assimilation block. The optimal number of pass transistors between adjacent buffers is first examined. Table 1 lists simulated average delay per pass transistor stage as a function of n, the number of pass transistors between adjacent buffers, based on the assumption that the total power consumption for the carry-chain is constant. For example, the buffers are twice as large (width-to-length ratio WLR=15 for the pull-up transistor and WLR=30 for the pull-down device) for the n=8 case as they are for the n=4 case (WLR=7.5 for pull-up and WLR=15 for pull-down). It is apparent that different total power consumption would result in different average delay per pass transistor stage. However, the general trends would stay the same. This also applies to the data in subsequent Tables. Table 1 confirms the conclusion derived in [1] that n=4 is the fastest organization. Thus n=4 will be used in the subsequent optimizations.

We next explore the effect of pass transistor size on circuit performance. An average delay per pass transistor stage as a function of WLR is simulated in Table 2. The performance is improved, at the expense of silicon area, as the WLR increases due to the fact that the pass transistors operate primarily in the linear region. They can be modeled as a transmission line. When these pass transistors are small, the resistance, as well as delay time, decreases as the WLR increases. However, the gate-to-channel capacitance of each transistor also increases as the WLR increases, and offsets the further performance improvement as the devices become large.

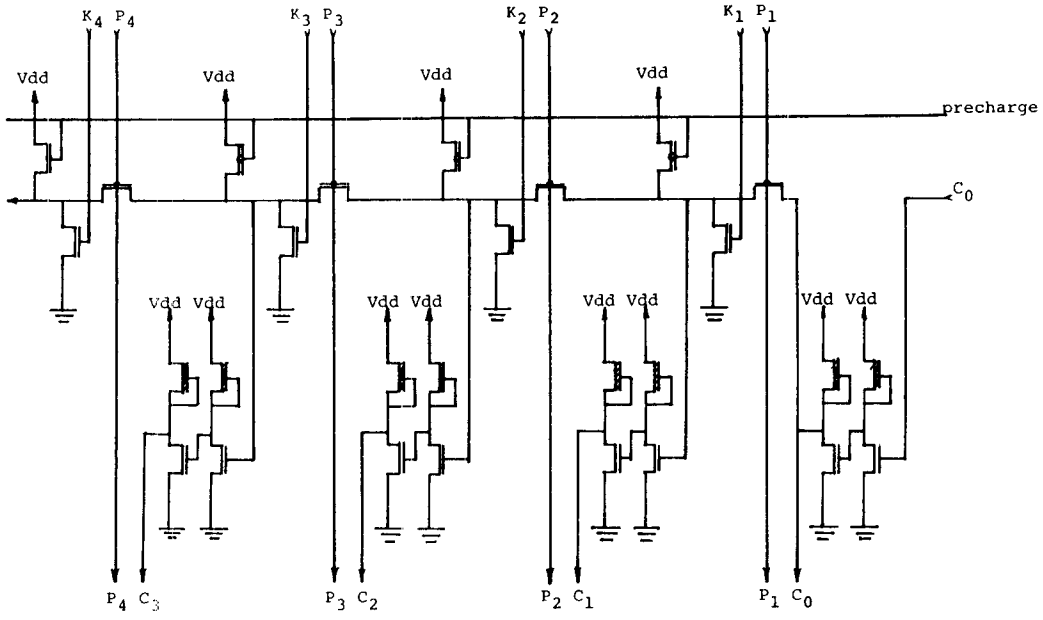

Fig. 3 Carry-chain circuit.

n number of pass transistor between buffers	t (ns) average delay per pass transistor stage
2	0.718
4	0.569
6	0.625
8	0.725

Table 1 Performance comparison of n.

WLR pass transistor width-to-length ratio	t (ns) average delay per pass transistor stage
5	0.745
7.5	0.625
10	0.569
15	0.525
20	0.51
30	0.52

Table 2 Performance comparison of pass transistor size.

WLR pull-down transistor width-to-length ratio	t (ns) average delay per pass transistor stage
10	0.633
15	0.569
20	0.539
30	0.528
40	0.551

Table 3 Performance comparison of pull-down device size.

C noralized wiring capacitance	t (ns) average delay per pass transistor stage
0.5	0.489
1	0.569
1.5	0.638
2	0.708
2.5	0.785

Table 4 Performance comparison of the wiring capacitance on the carry-chain.

Another factor which affects the carry-chain performance is the size of the pull-down transistor of the second invertor, because the precharged carry-chain discharges through this device. Table 3 lists simulated average delay per pass transistor stage as this device size varies. The average delay per pass transistor can be reduced as the device size increases until the gate-to-channel capacitance dominates.

It is well-known that the wiring capacitance affects MOS circuit performance. Several wiring capacitances have been investigated. Simulation shows that the wiring capacitance of the carry-chain is dominant. Simulations are shown in Table 4. If the wiring capacitance is doubled, the performance will degrade more than 24%. It illustrates the importance of minimizing carry-chain wiring capacitance in the physical circuit layout.

Based on the above studies, we exercise numerous variations of choices from Tables 2-4. We also vary the size of devices of the preprocessing block and the result block. Fig. 4 summarizes results from trial simulations for a 32-bit ALU using the precharged Manchester carry ripple approach. It is interesting to note that performance is limited to about 17 ns.

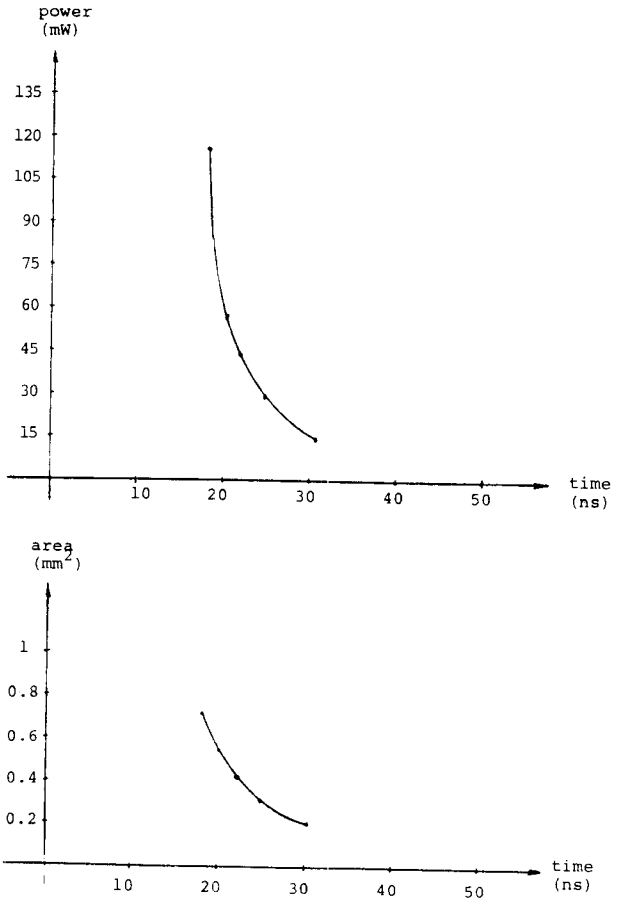

Fig. 4 Cost-performance of the precharge Manchester carry ripple approach.

STANDARD MULTI-LEVEL CARRY LOOK-AHEAD

A review of the principles of binary carry look-ahead addition may be found in [2],[3]. Although the carry look-ahead scheme has been used for higher performance in bipolar technologies (e.g. TI 74S182), the widely held belief for MOS technologies is that the look-ahead circuit is complicated and does not make a proportionate contribution to

performance [1]. Several carry look-ahead implementations have been reported lately in the context of VLSI design [9]-[11]. The results do not clearly draw useful conclusions, particularly regarding the trade-offs. We next examine two look-ahead structures, namely 2-bit and 4-bit look-ahead.

Figs. 5 and 6 are the floor plans of a 2-bit and a 4-bit look-ahead carry assimilations for the case N=16, respectively. NOR circuits are used to implement processing cells. Note that processing cells are lined horizontally, and wiring channels are used between two rows of processing cells. We use metal lines for horizontal wiring and polycide lines for vertical wires. This makes the structures very regular and easy to construct.

Fig. 7 shows a comparison of the two structures for a 32-bit ALU. Here the 4-bit look-ahead structure is always faster and requires less area. This is primarily due to the 4-bit look-ahead structure only requiring half the levels of the 2-bit look-ahead structure (Figs. 5 and 6) with less than twice the delay time per level.

The wiring capacitance is not a critical parameter for the 4-bit look-ahead structure. A simulation result is shown in Fig. 8. If the wiring capacitance is reduced to half, the performance improves only by 5.3%. This indicates that the device capacitance is dominant, and it will be more so for larger look-ahead groups (e.g. 8-bit look-ahead). Use of 8-bit look-ahead will not, therefore, improve performance greatly.

SPECIAL CASE OF RECURRENCE SOLVERS

It is well-known that the carrys can be generated as follows:

$$c_0 = CIN$$
$$c_i = g_i + (p_i \cdot c_{i-1})$$

where

$$g_i = a_i \cdot b_i$$
$$p_i = a_i \oplus b_i$$

for i=1,2, ... , n. Therefore, it is obvious that the carry generation is a special case of solving a linear recurrence [4]. Solutions for the recurrence relation discussed in [4],[5] can be applied to produce the carry vector. A similar approach has been used to implement an adder in [12].

Fig. 9 illustrates the structure for the case N=16. A comparison with the two carry look-ahead schemes in terms of the number of levels and processing cells is given in Table 5. Note that the number of levels required by this structure is about the same as the 4-bit look-ahead approach. A potential problem of this structure is the worst case fanout (N/2) as well as the long wires (the longest wire has to travel N/2 bits) to be driven by a single device. This problem can be eased by using proportionally larger drivers alone the critical path.

The physical layout scheme discussed for the carry look-ahead approaches is also used here. Larger processing cells, again constructed from NOR circuits, are accommo-

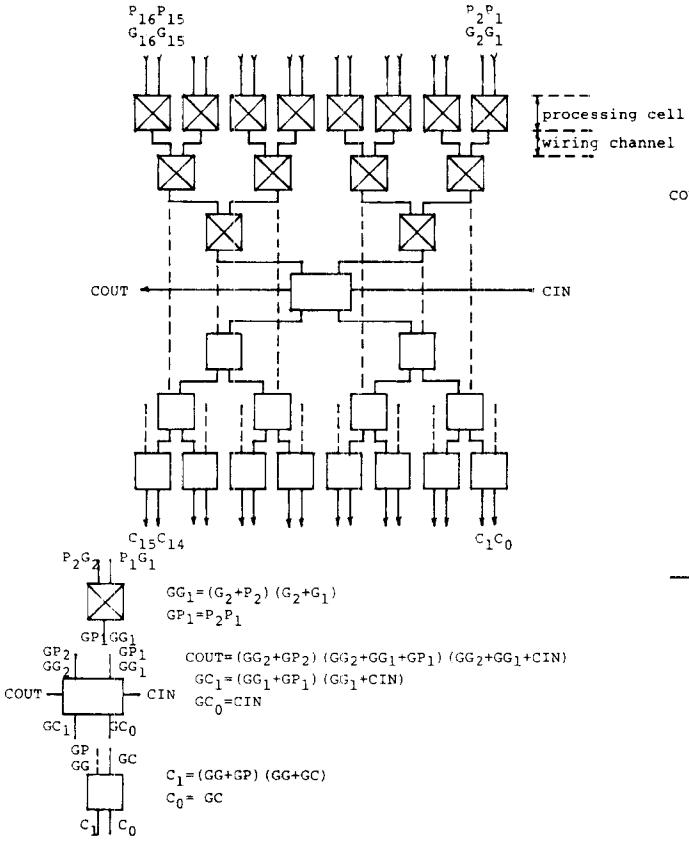

Fig. 5 Floor plan of a 2-bit look-ahead structure.

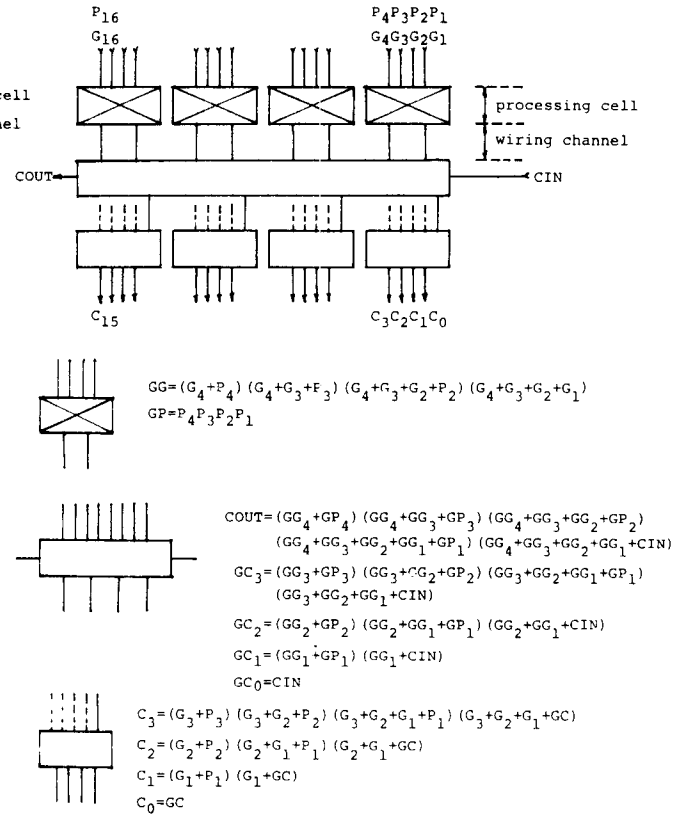

Fig. 6 Floor plan of a 4-bit llok-ahead structure.

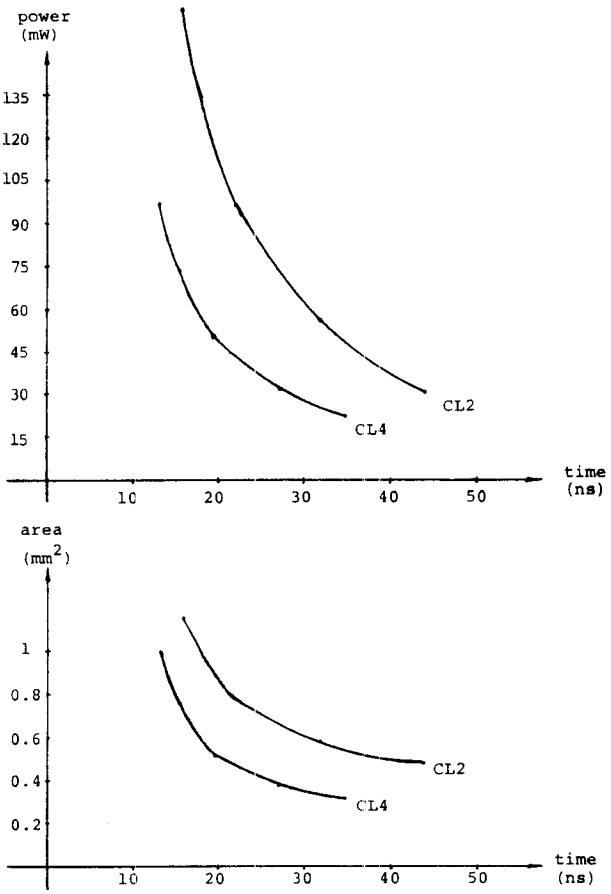

Fig. 7 Cost-performance of two look-ahead approaches.

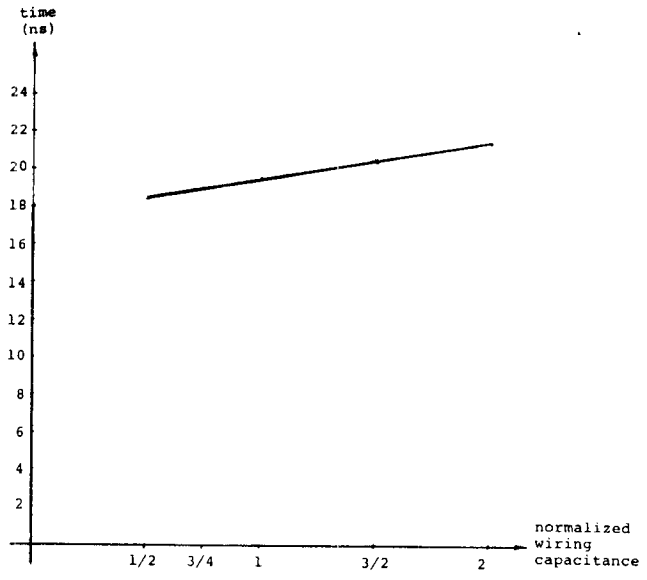

Fig. 8 Effect on performance of 4-bit look-ahead wiring capacitance.

scheme	number of levels	number of processing cells
2-bit look-ahead	$(2*\log_2 N)-1$	$(2*N)-3$
4-bit look-ahead	$(2*\log_4 N)-1$	$\frac{2*N}{3}-1$
recurrence solver	$\log_2(N-1)$	$\frac{N*\log_2 N}{2}$

Table 5 Comparison of schemes.

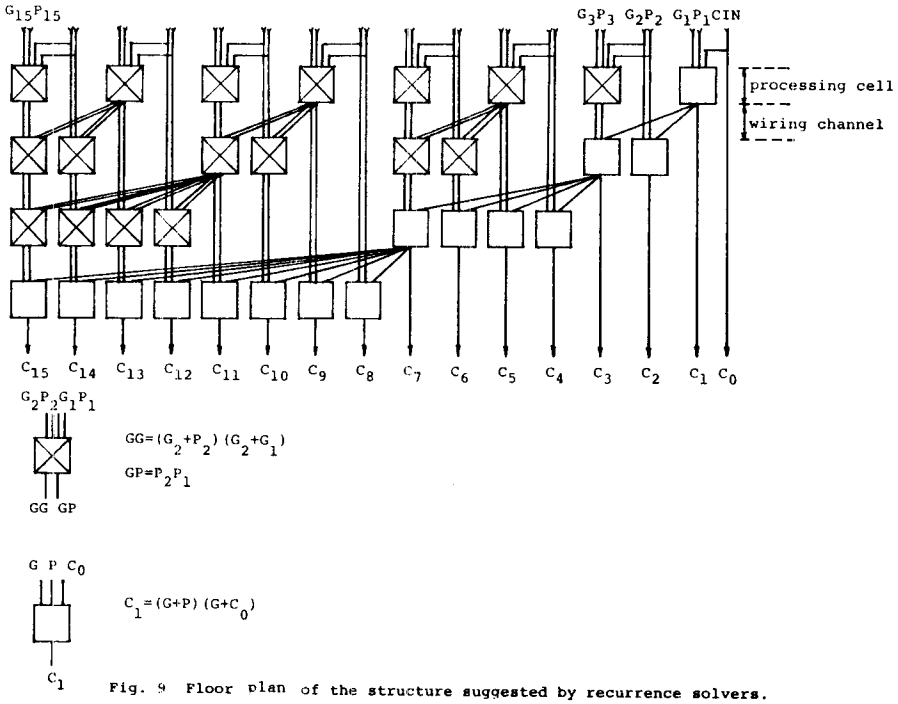

Fig. 9 Floor plan of the structure suggested by recurrence solvers.

dated through longer vertical dimensions. Fig. 10 is a graph which summarizes simulation results for a 32-bit ALU using this structure suggested by recurrence solvers. It can perform an ALU operation as fast as 10 ns.

COMPARISON AND CONCLUSION

A comparison among three alternate structures for implementing a 32-bit ALU using a 1 μm NMOS FET technology is shown in Fig. 11. The 4-bit look-ahead structure and the structure suggested by recurrence solvers provide alternatives to the Manchester carry-chain for a high performance ALU and can also be further optimized at the circuit level.

Under the assumptions made, the general conclusions are as follows:

1) The precharged Manchester carry ripple approach is best suited for the low power (<45 mW), and/or small area (<0.5 mm^2) applications.

2) For high performance (<17 ns) applications, both the 4-bit look-ahead approach and the structure suggested by recurrence solvers are proper candidates. The 4-bit look-ahead approach is slightly faster and requires less area than the structure suggested by recurrence solvers.

3) The 4-bit look-ahead is always faster and requires less area than the 2-bit look-ahead approach.

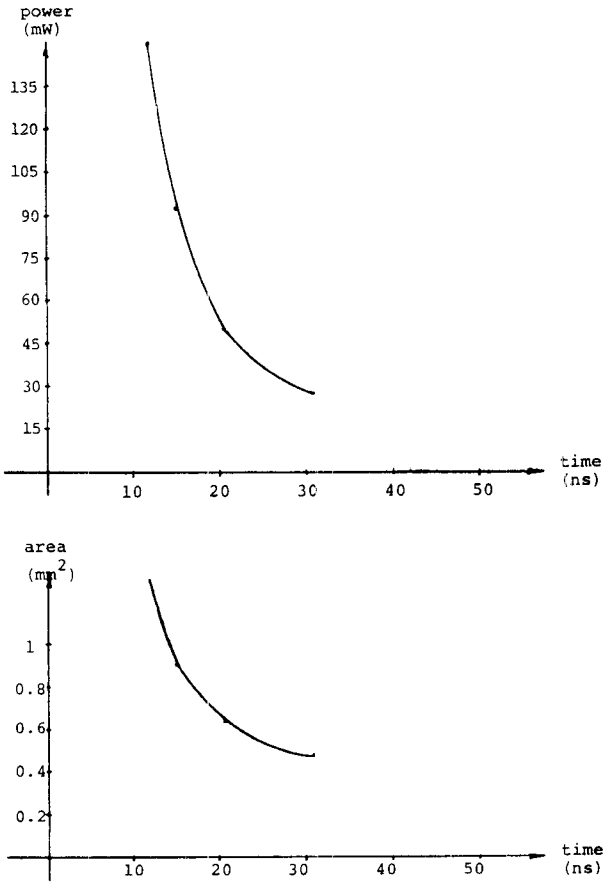

Fig. 10 Cost-performance of the structure suggested by recurrence solvers.

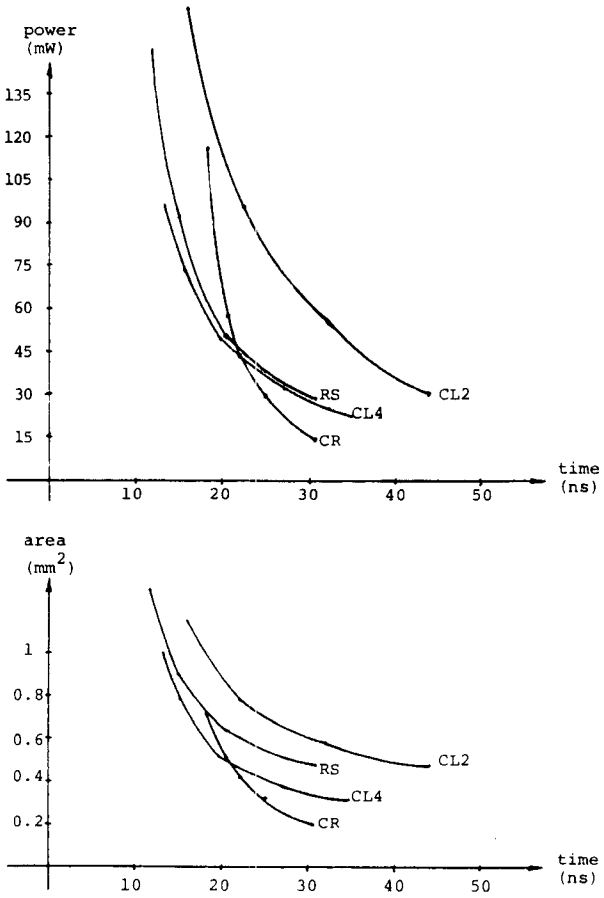

Fig. 11 A comparison of alternate approach.

REFERENCES

[1] C. M. Mead and L. A. Conway, *Introduction to VLSI systems.* Reading, MA: Addison-Wesley, 1980.

[2] F. J. Hill and G. R. Peterson, *Digital Systems: Hardware Organization and Design.* New York: Wiley, 1973, chp 11.

[3] G. A. Blaauw, *Digital System Implementation.* Englewood Cliffs, NJ: Prentice-Hall, 1976, chp 2.

[4] D. J. Kuck, *The Structure of Computers and Computation, Vol. I.* New York: Wiley, 1978, chp 2.

[5] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *J. of ACM,* vol. 27, pp. 831-838, Oct. 1980.

[6] H-N Yu et al., "1 μm MOSFET VLSI technology: part I - an overview," *IEEE Tran. on Electron Devices,* vol. ED-26, pp. 318-324, April 1979.

[7] M. Y. Tsai et al., "One-micron polycide (WSi_2 on poly-Si) MOSFET technology," *J. Electrochemical Society*, vol. 128, pp. 2207-2214, Oct. 1981.

[8] H. H. Chao et al., "A 34 μm^2 DRAM cell fabricated with a 1 μm single-level polycide FET technology," *IEEE J. of Solid-State Circuits*, vol. SC-16, pp. 499-505, Oct. 1981.

[9] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Tran. on Comp.*, vol. C-31, pp. 260-264, March 1982.

[10] J. Vuiliemin and L. Guibas, "On fast binary addition in MOS technologies," *Proc. of the 1982 Int. Conf. on Circuit and Computers*, pp. 147-150, 1982.

[11] R. Bechade and W. K. Hoffman, "Generalized 2-bit slice ALU," *Proc. of the 1980 Int. Conf. on Circuit and Computers*, pp. 1094-1098, 1980.

[12] R. K. Montoye, "Area-time efficient addition in charge based technology," *18th Design Automation Conf. Porc.*, pp. 862-872, 1981.