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

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

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*),

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

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.

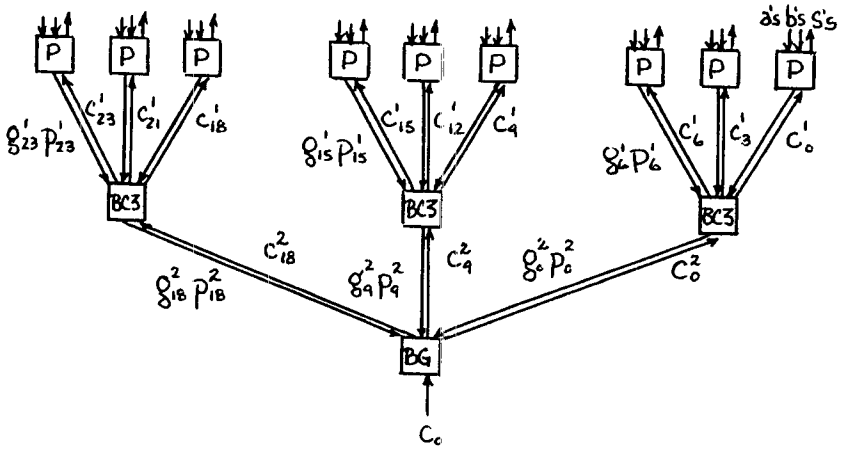


Figure 1. A 27-bit CLA Adder

Because we are using NMOS technology as our model, all complex gates use complementary logic. Thus, it is desirable to modify the above scheme to reflect the use of a negative logic system. Assume that at the k-th level of the iteration, the block carry computation is

$$c_{i+1}^k = g_i^k + p_i^k c_i^k$$

We can reformulate this computation into negative logic as

$$c_{i+1}^k = \neg\left[G_i^k + P_i^k(\neg c_i^k)\right],$$

where

$$G_i^k = \neg(g_i^k + p_i^k)$$

$$P_i^k = \neg(g_i^k)$$

$\neg x$ denotes the complement of x *(not x)*

Then for next level of iteration, we have

$$c_{i+1}^{k+1} = g_i^{k+1} + p_i^{k+1} c_i^{k+1}$$

$$c_{ri}^k = \neg(c_i^{k+1})$$

$$c_{ri+m+1}^k = \neg\left[G_{ri+m}^k + \sum_{l=0}^{m-1}\left(\prod_{j=l+1}^{m} P_{ri+j}^k\right) G_{ri+l}^k + \left(\prod_{j=0}^{m} P_{ri+j}^k\right) c_i^{k+1}\right] \quad \text{for } 0 \le m \le r-2$$

where

$$g_i^{k+1} = G_{ri+(r-1)}^k + \sum_{l=0}^{r-2}\left(\prod_{j=l+1}^{r-1} P_{ri+j}^k\right) G_{ri+l}^k$$

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

$$p_i^{k+1} = \prod_{j=0}^{r-1} P_{ri+j}^k = \neg\left[\sum_{j=0}^{r-1} g_{ri+j}^k\right]$$

With this reformulation, all block carries, block carry generates and block carry propagates can be directly implemented with complex gates.

Adder Design

We found that a blocking factor of three for the first two levels of the iteration and four for all higher levels resulted in a more area efficient VLSI NMOS design as will be demonstrated in the next section. With this scheme there are four types of basic units to be designed (only three of which are depicted in Figure 1). These are the primitive unit (P), the 3-input block carry unit (BC3), the 4-input block carry unit (BC4), and the block carry generation unit (BG). NMOS layouts of three of these units are shown in Figure 2.

Primitive unit (P) This basic unit includes four subunits: input/output, block propagate and block generate, carry generation and summation. We postpone the discussion of the input/output subunit. Here we assume that the input/output subunit is of low complexity and uses constant area. In the block propagate and block generate subunit, block propagate p^1 and block generate g^1 are formed directly from the operand bits $a_2a_1a_0$ and $b_2b_1b_0$. (We restrict our attention to the least significant unit for notational simplicity without loss of generality.)

$$g^1 = \neg\left[(g_2+p_2)(g_2+g_1+p_1)(g_2+g_1+g_0+p_0)\right]$$

$$= \neg\left[a_2b_2+(a_2+b_2)\left[a_1b_1+(a_1+b_1)(a_0+b_0)\right]\right]$$

$$p^1 = \neg(g_2+g_1+g_0) = \neg(a_2b_2+a_1b_1+a_0b_0)$$

The carry generation subunit computes all the carry bits internal to the primitive unit from the block carry c_0^k which is input to the unit.

$$c_0 = \neg(c_0^k)$$

$$c_1 = \neg(G_0+P_0c_0^k)$$

$$c_2 = \neg(G_1+P_1G_0+P_1P_0c_0^k)$$

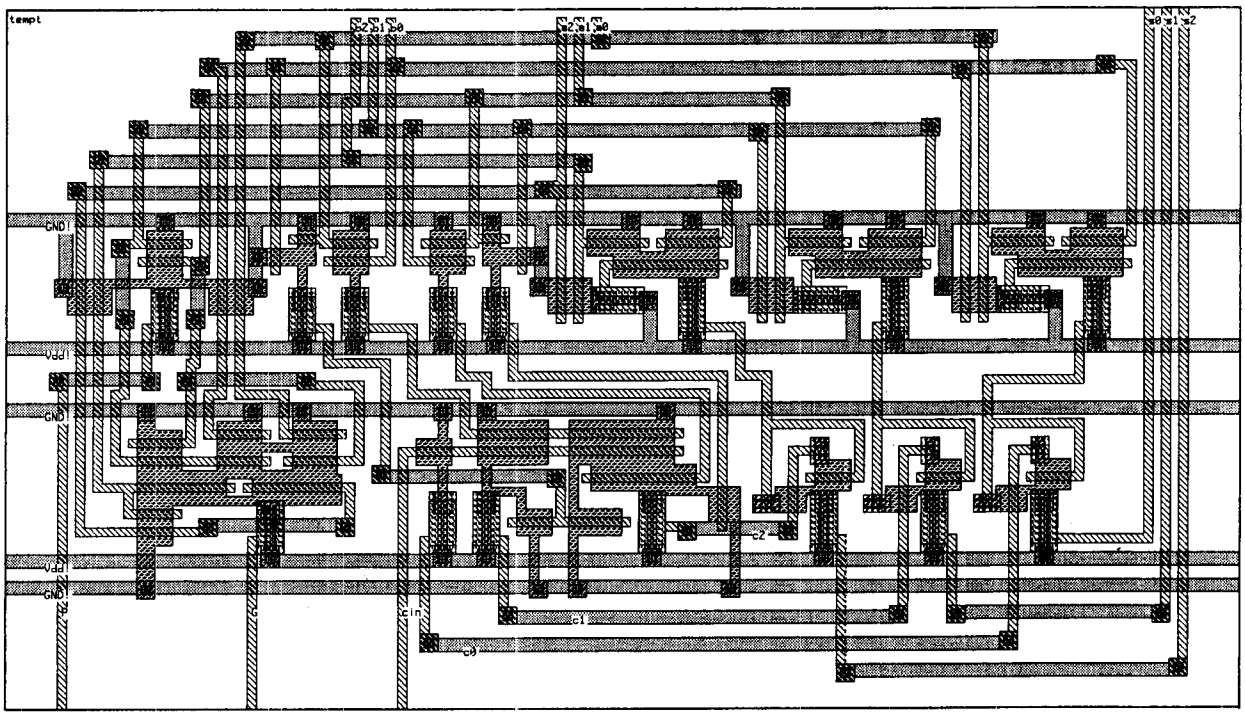


Figure 2a. The Primitive Unit Layout

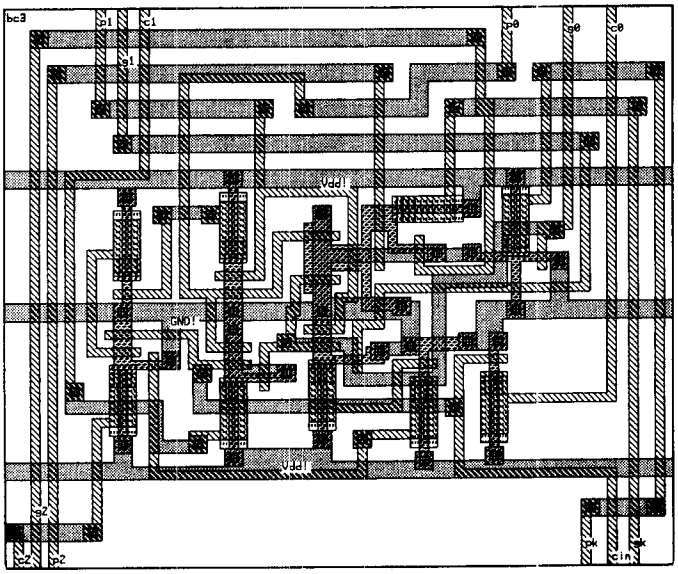


Figure 2b. The BC3 Layout

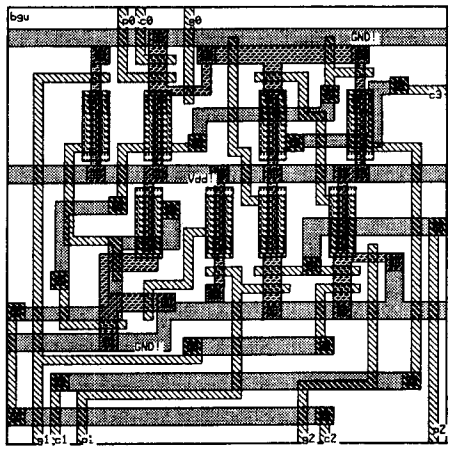


Figure 2c. The BG Layout

where

$$G_0 = \neg(g_0+p_0) = \neg(a_0+b_0)$$
$$G_1 = \neg(g_1+p_1) = \neg(a_1+b_1)$$
$$P_0 = \neg g_0 = \neg(a_0 b_0)$$
$$P_1 = \neg g_1 = \neg(a_1 b_1)$$

The summation subunit computes the sum bits from the internal carries and the operand bits.

$$s_i = a_i \oplus b_i \oplus c_i \qquad \text{for } i=0,1,2$$

The corresponding NMOS complex gate layout for the primitive unit is shown in Figure 2a.

Block carry units (BC3 and BC4) These two basic units form the block carry generates and block carry propagates from either three (BC3) or four (BC4) inputs which will serve as inputs to the next higher level and also compute the block carries which will serve as inputs to the subsequent lower level. Implementation of the block carry generates and propagates using complex gates is straightforward. For example in BC4

$$g^{k+1} = \neg\left\{ g_3^k + p_3^k\left[g_2^k + p_2^k\left[g_1^k + p_1^k(g_0^k + p_0^k) \right] \right] \right\}$$

$$p^{k+1} = \neg(g_3^k + g_2^k + g_1^k + g_0^k)$$

The block carries are computed only when the block carry generated by the next higher level becomes available.

$$c_0^k = \neg(c_0^{k+1})$$
$$c_1^k = \neg(g_0^k + p_0^k c_0^{k+1})$$
$$c_2^k = \neg(G_1 + P_1 c_0^{k+1})$$
$$c_3^k = \neg(G_2 + P_2 c_0^{k+1}) \quad \dagger$$

where

$$G_1 = g_1^k + p_1^k g_0^k$$
$$P_1 = p_1^k p_0^k$$
$$G_2 = g_2^k + p_2^k(g_1^k + p_1^k g_0^k) \quad \dagger$$
$$P_2 = p_2^k p_1^k p_0^k \quad \dagger$$

† in BC4 unit only

Figures 2b shows the corresponding NMOS complex gate layout for the BC3 unit.

Block carry generation unit (BG) This basic unit forms block carries at the highest level of iteration. If K is the total number of iteration levels, then

$$c_0^K = \begin{cases} \neg c_0 & \text{if } K \text{ is odd} \\ c_0 & \text{if } K \text{ is even} \end{cases}$$

and

$$c_1^K = g_0^K + p_0^K c_0^K$$
$$c_2^K = g_1^K + p_1^K(g_0^K + p_0^K c_0^K) \quad \dagger$$
$$c_3^K = (g_2^K + p_2^K g_1^K) + (p_2^K p_1^K)(g_0^K + p_0^K c_0^K) \quad \dagger$$

† as needed

Figure 2c shows the corresponding NMOS complex gate layout for the BG unit.

Area and Time Measures

Figure 3 is an NMOS layout for a 27-bit adder using the scheme derived above. This layout has been fit in area $O(n)$ by using a recursive technique for embedding tree-like structures in the plane [Ull84]. As can be seen in the floorplan schematic in Figure 4a, a 27-bit adder can be packed most densely by using a blocking factor of three at each of the levels. For larger adders, a blocking factor of three for the first two levels and four thereafter leads to the most efficient layout as seen in Figures 4b and 4c. The flexibility that comes from being able to vary the blocking factor at various levels can be used to make optimum use of space. Note that all interconnections between the basic units involve only nearest neighbors with a maximum wire length of $O(\sqrt{n})$.

Brent and Kung's adder (BrKu82] can also be embedded in $O(n)$ area with a maximum wire length of $O(\sqrt{n})$, but the lack of flexibility in the blocking factor (only two) results in a much less densely packed layout with considerably more area dedicated to interconnect.

Obviously, any $\log n$ stage adder can add two n-bit numbers in time $O(\log n)$. Another motivation in using negative logic was to speed up the addition even more. At each level, we need to form the carry generate and carry propagate terms and to form the block carries out after receiving the block carry in. While a positive logic approach would require four NMOS gate delays to accomplish these operations, the theoretical lower bound for the negative logic approach should be two gate delays. SPICE simulations have indicated that three gate delays may be a more realistic measure for our complex gates. Table 1 gives a comparison of the computation time for a serial scheme, Brent and Kung's scheme, and our scheme for various n. T is the average delay of a simple NAND/NOR gate measured at the inversion voltage level. The delay of an exclusive or gate is assumed to be 2T.

Table 1. Comparison in T

n	Serial	B&K	N&I
8	18	15	
9	20		8
16	34	19	
27	56		12
64	130	27	
108	218		16
256	514	35	
432	866		20

For large n, our design uses time $2T \times \log n$ as compared to $2T \times 2\log n$ for Brent and Kung's design.

Input-Output Consideration

In the previous section, our design was based on the assumption that external inputs and outputs are locally available at the primitive units. When more realistic input and out-

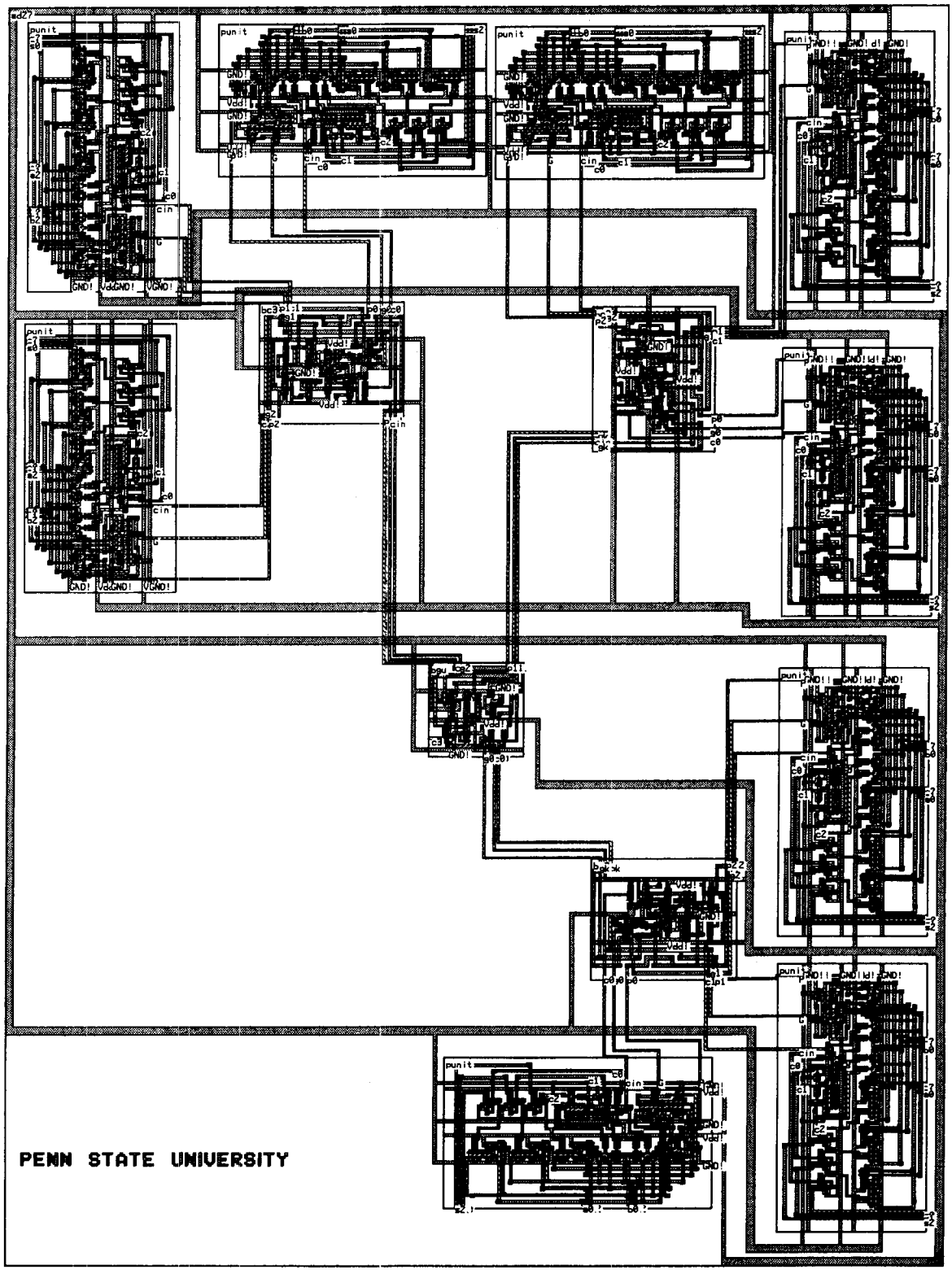


Figure 3. Layout of a 27-bit CLA Adder

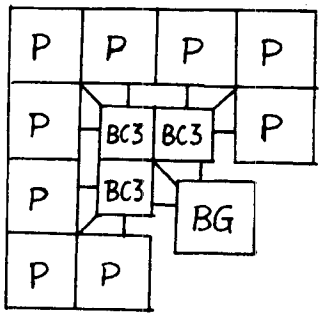


Figure 4a. A 27-bit CLA Floorplan

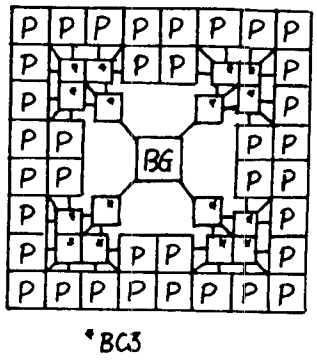


Figure 4b. A 108-bit CLA Floorplan

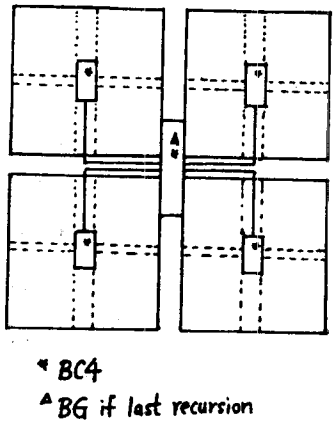


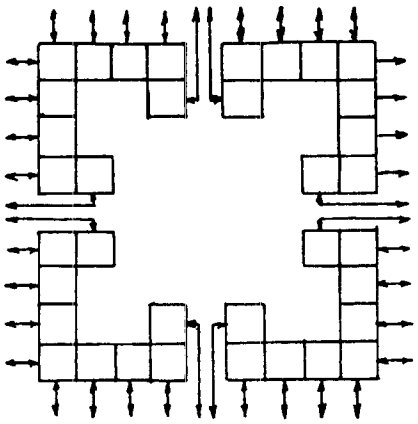
Figure 4c. A Recursive CLA Floorplan

put constraints are considered, we face the following well known dilemma - one can pack n processing elements densely in an area of $O(n)$ with the smallest linear dimension (such as in a circle or in a regular polygon) but if all inputs and outputs are only available on the boundary of the area, how can the $O(n)$ I/O wires be routed across the boundary of length $O(\sqrt{n})$! In our CLA design, there are two obvious possible solutions.

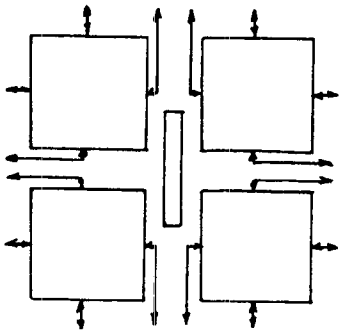
The first solution is to relax the area efficiency requirement; let all I/O wires be routed across the boundary as shown in Figure 5a. The area is then of $\Omega(n^2)$. As n increases, the portion of the area used for interconnection increases significantly. Obviously for very large n, this solution uses the most area for input/output wires and the most time for input and output along the I/O wires. Area and time complexities become $O(n^2)$ and $O(n)$, respectively. Although such asymptotic behavior is undesirable, this solution may be appealing for not too large n's. It is estimated that for n in the thousands, the total area is only several times of that of the previous "generic" adder and the total time (including the input/output time) is less than twice of that of the generic adder. Furthermore, although the length of the I/O wires is of $O(n)$, the maximum size of addition allowed is still large due to the fact that the proportional constant with n is small.

The other solution relaxes the time efficiency requirement by assuming that inputs and outputs are time multiplexed. In this case the number of I/O wires can be reduced to match the length of the boundary. However, now input and output will take time $\Omega(\sqrt{n})$. Besides the data lines, common address lines, clock and control lines must be provided as shown in Figure 5b. In each primitive unit, the input-output subunit has registers and the decoding circuit. With the appropriate address on the bus, data may be loaded into or read out of the registers. Asymptotically, this solution uses area proportional to $n\log^2 n$. But practically for an n as large as 10^5, the addition I/O wire area is still no more than that of the generic adder. This solution will require $c\sqrt{n}$ clocks for each input (output) with the minimum clock period allowed proportional to $\sqrt{n}$. The total I/O time is then proportional to n which is much longer than the addition time. Therefore, this solution would not be desirable for just additions. By expanding the functionality of the primitive unit, such as performing an accumulation function, the design may be appealing. For example, multiplication can then be done in time proportional to $n \log n$. This outweights the I/O time and results a multiplier of time $O(n \log n)$. This is only desirable in the cases where processing time is much longer than the input/output time. Furthermore, the input-output wire reduction also conforms to the limited I/O pin restriction for VLSI chips.

The input-output difficulties discussed above are due to the planar restriction on VLSI processing. Recently, a considerable amount



(a) 108 bit structure



(b) recursive scheme

Figure 5a. Boundary I/O

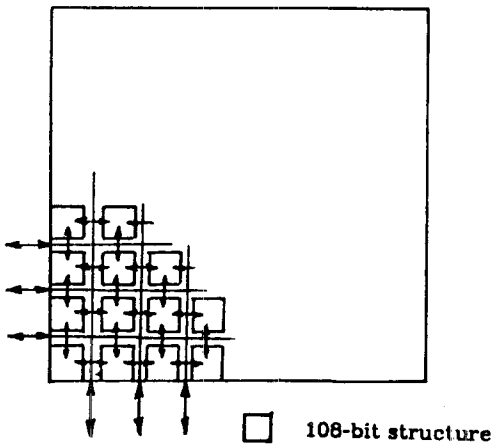


Figure 5b. Shared Bus I/O

of research effort has been put into developing 3-D VLSI processing and 3-D microassembly techniques [GrNE84]. Our generic adder would be ideal for these new emerging technologies.

Conclusion

We have shown that using the conventional iterative CLA scheme, nearly optimal CLA adders can be obtained. The layout of the adder is highly regular. Neglecting input-output restriction, the addition of two n-bit numbers can be performed in time $O(\log n)$, using area $O(n)$ with maximum interconnection wire length $O(\sqrt{n})$. When input-output is considered, two different alternative designs are evaluated. Even for practically large n's, it is shown that reasonably good area and time efficiencies can be achieved.

Acknowledgements

This work is supported in part by the Army Research Office under Contract DAAG29-83-K-0126. Thanks go to Shishpal Rawat for work on the CIF layout plots.

References

[OnAt83] S.Ong and D.E.Atkins, "A Comparison of ALU Structures for VLSI Technology," *Proceedings of the Sixth Symposium on Computer Arithmetic*, Aarhus, Denmark, June 1983.

[BrKu82] R.P.Brent and H.T.Kung, "A Regular Layout for Parallel Adders" *IEEE Transactions on Computers*, Vol. C-31, March 1982.

[BaJM83] M.A.Bayoumi, G.A.Jullien and W.C.Miller, "An Area-Time Efficient NMOS Adder," *INTEGRATION, The VLSI Journal*, Vol. 1, 1983.

[MeCo80] C.A.Mead and L.A.Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

[MaJD83] J.Mavor, M.A.Jack and P.B.Denyer, *Introduction to MOS LSI Design*, Addison-Wesley, 1983.

[WaFl82] S.Waser and M.J.Flynn, *Introduction to Arithmetic for Digital Systems Designers*, Holt, Rinehart and Winston, 1982.

[Ngai84] T.F.Ngai, "On the Capacitive Model of Computation in VLSI: Its Limitation and Applicability" in preparation.

[BiPP81] G.Bilardi, M.Pracchi and F.P.Preparata, "A Critique and an Appraisal of VLSI Models of Computation" *VLSI Systems and Computations*, (edited by H.T.Kung, et.al.), Computer Science Press, 1981.

[Ull84] J.Ullman, *Computational Aspects of VLSI*, Computer Science Press, 1984.

[GrNE84] J.Grinberg, G.R.Nudd and R.D.Etahcells, "A Cellular VLSI Architecture", *IEEE Computer*, January 1984.