Design Community Projects

DESIGN OF AN nMOS PARALLEL ADDER

by Richard P. Brent and Robert R. Ewin
Department of Computer Science
The Australian National University, Canberra.

Introduction

For serial n-bit adders such as the "Manchester carry chain" adder described in Mead and Conway [5, pp.150,167], the worst-case addition time is proportional to n, because a carry may have to propagate through n bit positions. The constant of proportionality can be reduced by techniques such as pre-charging busses and optimising the placement of inverters in the carry chain [5, p.23], but there is no escaping from the fact that the worst-case carry propagation time is of order n. Assuming random inputs, the average carry propagation time [5, p.252] is of order log(n), but it is the worst-case rather than the average which determines the cycle time of most synchronous (as opposed to self-timed) arithmetic units.

We assume that signal propagation times along metal wires without fanout are at worst comparable to a small number of inverter delays. This is true for current (and anticipated) nMOS technology and adders of practical size: see [1, 5]. Under this assumption, parallel or "carry lookahead" adders with a worst-case addition time proportional to log(n) are possible, and should be faster than serial adders unless n is small.

Mead and Conway [5, p.150] considered several known carry lookahead schemes for implementation with nMOS VLSI technology, but concluded that "they added a great deal of complexity to the system without much gain in performance". Brent and Kung [3] claimed that this conclusion did not apply to their scheme, which uses only a small number of basic (leaf) cells arranged in a regular pattern.

In the next section we describe a variant of Brent and Kung's scheme. The variant has some advantages over the original scheme for implementation with nMOS technology. We then compare the proposed parallel adder with a serial adder like that described in [5], and document a 5-bit prototype parallel adder which has been submitted to the CSIRO VLSI Program for implementtion on the first Australian Multiproject chip (AUSMPC 5/82, see [6]) with 5-micrometre nMOS technology (i.e. λ = 2.5 micrometres).

A Variant of the Brent-Kung Parallel Adder

We use the same notation as in [3]. Carries are propagated using two binary trees which are embedded in a rectangular array of "white" and "black" cells: see [3, Fig.5]*. The function of the white and black cells is

* Note that the rightmost black cell at level T = 5 of [3, Fig.5] should be a white cell; similarly at level T = 6.

illustrated in Figs. 1 and 2.

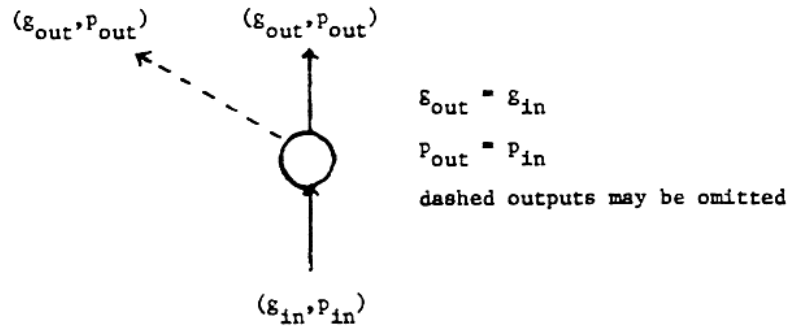


FIG. 1. WHITE CELL OF [3]

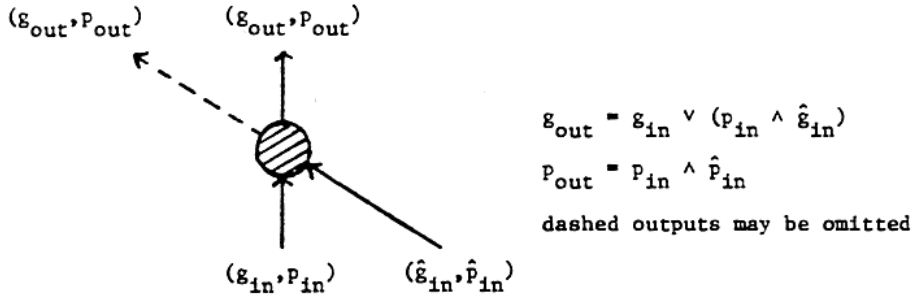


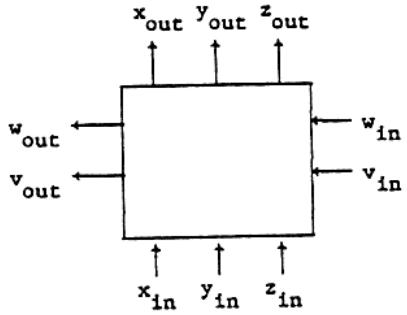
FIG. 2. BLACK CELL OF [3]

We wished to avoid diagonal connections between cells, so we introduced two types of white cells which use only "Manhattan" (i.e. right angle) geometry. One type of white cell (WA) transmits inputs from east to west as well as from south to north, while the other type (WB) transmits inputs from south to both west and north (see Figs. 3 and 4). Thus, a diagonal connection is replaced by a row of zero or more WA cells to the west of one WB cell.

With nMOS technology it is easier to implement NOR and NAND than OR and AND gates. Hence, we decided to complement all variables transmitted vertically by a row of white and black cells. This necessitated the introduction of two types of black cells, one the dual of the other (see Figs. 5 and 6).

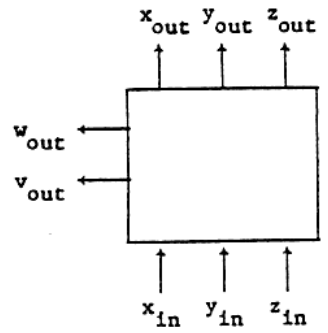
With the scheme of [3] it is necessary to transmit the "carry propagate" values $p_i = a_i \oplus b_i$ from south to north through the adder. We expanded the black and white cells to include an extra input and output for the p_i lines. For consistency, the p_i values are complemented by each row of black and white cells.

We use the generic name "BW cell" for any of the WA, WB, BA or BB cells. All BW cells must have the same width (W) and height (H) so that they can be stacked in a regular rectangular array. Note that the output y_{out} of BW cells on the top row is not used, but for the sake of regularity we avoided modifying cells in the top row to take advantage of this.



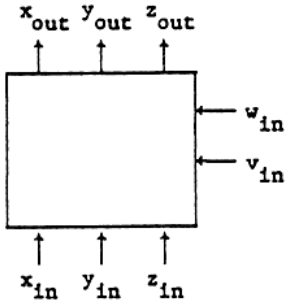
$v_{out} = v_{in}$
$w_{out} = w_{in}$
$x_{out} = x_{in}'$
$y_{out} = y_{in}'$
$z_{out} = z_{in}'$

FIG. 3. WA CELL INPUTS AND OUTPUTS



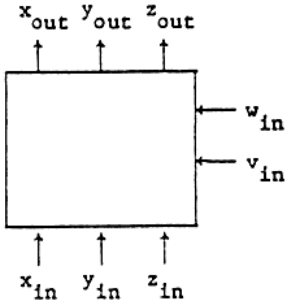
$v_{out} = x_{in}$
$w_{out} = y_{in}$
$x_{out} = x_{in}'$
$y_{out} = y_{in}'$
$z_{out} = z_{in}'$

FIG. 4. WB CELL INPUTS AND OUTPUTS



$x_{out} = (x_{in} \vee (y_{in} \wedge v_{in}))'$
$y_{out} = (y_{in} \wedge w_{in})'$
$z_{out} = z_{in}'$

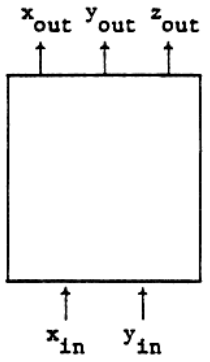
FIG. 5. BA CELL INPUTS AND OUTPUTS



$x_{out} = (x_{in} \wedge (x_{in} \vee v_{in}))'$
$y_{out} = (y_{in} \vee w_{in})'$
$z_{out} = z_{in}'$

FIG. 6. BB CELL INPUTS AND OUTPUTS

To complete the adder we need three additional types of cells illustrated in Figs. 7-9: G cells to compute carry-generate and carry-propagate signals, S cells to combine the final carry and sum signals, and I cells which are simply inverters. The S cells are placed north of two BW cells and connected to one output of each (the "y_{out}" output from these BW cells is discarded). G cells and S cells must have the same width (W) as BW cells, but may differ in height.

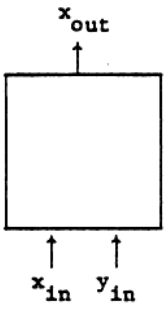


$x_{out} = x_{in} \wedge y_{in}$

$y_{out} = x_{in} \oplus y_{in}$

$z_{out} = x_{in} \oplus y_{in}$

FIG. 7. G CELL INPUTS AND OUTPUTS



$x_{out} = x_{in} \oplus y_{in}$

FIG. 8. S CELL INPUTS AND OUTPUT

$x_{out} = x_{in}'$

FIG. 9. I CELL INPUT AND OUTPUT

An n-bit adder can be formed from a row of n G cells, several rows of n BW cells, and a row of n-1 S cells (possibly with an I cell at each end of the row). This is illustrated for the case n = 16 in Fig. 10 (compare Fig. 5 of [3]). There are approximately $(2\log_2 n-1)$ rows of BW cells. Note that the I cells would be omitted if there were an even number of rows of BW cells (e.g. if $12 < n < 15$, when there would be 6 such rows).

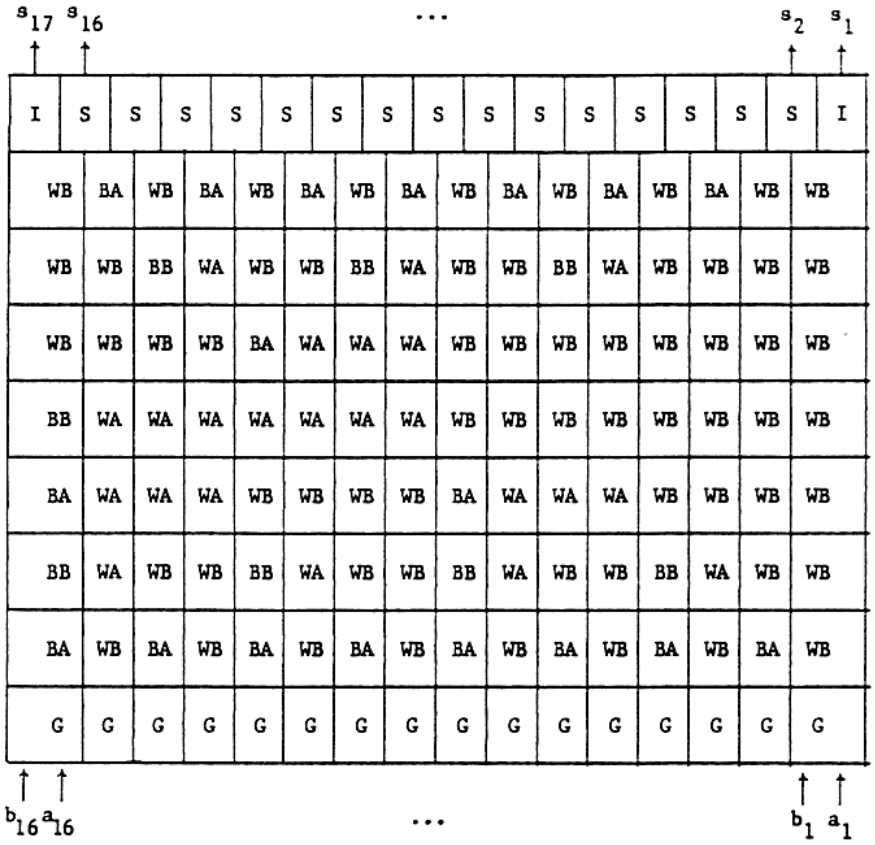


FIG. 10. CELL LAYOUT FOR CORE OF 16-BIT PARALLEL ADDER

The width of the n-bit adder is nW, where W is with width of BW, G and S cells (W = 31 λ for our design). The height is kH + H2, where:

k = number of rows of BW cells ≤ $[2.\log_2 n] - 1$,
H = height of BW cells (39λ for our design), and
H2 = sum of heights of G and S cells (88 λ = 55 λ + 33 λ for our design).

Table 1 gives some illustrative values, and also includes the "regularity factor" (defined here to be the total number of cells divided by the number of different cells.)

n	width (λ)	height (λ)	regularity factor
8	248	283	8
12	372	322	16
16	496	361	21
24	744	400	40
32	922	439	50
48	1488	478	96
64	1984	517	119

TABLE 1. DIMENSIONS AND REGULARITY OF n-BIT PARALLEL ADDER

The adder may be regarded as an n-bit unsigned or 2s complement adder with carry-out but no carry-in. If a carry-in (useful for multiple-precision arithmetic and 2s complement subtraction) is desired, the simplest solution is to use an (n+1)-bit adder with both the low-order inputs set to the carry-in and the low order output discarded.

The adder could be pipelined by running clock lines in an east-west direction through each row of cells.

Comparison With a Serial Adder

Mead and Conway [5] suggest implementing the carry propagation section of a serial nMOS adder with a "Manchester" chain of pass transistors. A pair of inverters is inserted after each block of 4 (or some other well-chosen number) of pass transistors to restore signal levels and minimise the carry propagation time. We shall assume that the delay per block of 4 bits is $100\,\tau$ (see [5, p.167]). Thus, the worst-case carry propagation time for an n-bit adder is about $25\,n\tau$.

By comparison, our parallel adder requires about $(2.\log_2 n-1)$ rows of BW cells for carry propagation. (We ignore the rows of G and S cells here as they or something similar would also be required for a serial adder.) Each pair of rows of BW cells causes a delay of approximately $60\,\tau$ (for two 4:1 inverters with effective fanout of 4 and a factor of 3 for parasitic capacitances). Hence, the total carry propagation time is about $(60\log_2 n-12)\tau$. Table 2 shows how this compares with the serial carry propagation time $25\,n\tau$.

n	Parallel(τ)	Serial(τ)	Ratio
4	108	100	1.08
8	168	200	0.84
16	228	400	0.57
32	288	800	0.36
64	348	1600	0.22
128	408	3200	0.13
256	468	6400	0.07

TABLE 2. PARALLEL VERSUS SERIAL CARRY PROPAGATION TIMES

The comparison made in Table 2 is rather crude, but sufficient to suggest that a parallel adder is worth considering seriously if $n > 16$. A detailed analysis would have to take into account the effects of pre-charging, optimal inverter ratios, optimal block size, stray capacitances, etc.. The parallel adder consumes more power (see Section 4.4) and slightly more area (see Table 1) than a serial adder, but this is probably insignificant if the adder is a critical component of a processor ALU. Note that the parallel adder has area $\Omega(n.\log n)$, asymptotically less than the $\Omega(n^2/\log^2 n)$ area required by a fast shifter [2] which would probably be a component of the ALU.

For an independent comparison of a serial adder with a different variant of the Brent-Kung parallel adder, see [8].

A 5-bit Prototype

To test the design proposed earlier, we have submitted a small prototype for fabrication on the first Australian Multiproject Chip, AUSMPC 5/82 [6]. We chose to implement only a 5-bit adder because of space limitations on the Multiproject Chip. n = 5 is large enough to test the concept of a parallel adder, and all cells are used (though only once in the case of WA and BB cells.)

(a) Functional Description

In order to test the dynamic behaviour of the adder, one modification was made to the scheme described earlier: the least significant bit of the output was fed back to the least significant bit of the input (under control of a mask c). More precisely, if the adder has inputs $a_4 \cdots a_0$ and $b_4 \cdots b_0$ and outputs $s_5 \cdots s_0$*, we set $b_0 := c$ and $a_0 := s_0 \hat{} c$.

Thus, the prototype functions as a 4-bit adder (with $s_5 \cdots s_1$ the sum of $a_4 \cdots a_1$ and $b_4 \cdots b_1$) if c = 0. If c = 1 then the low order bit slice functions as a chain of 7 inverters, so an oscillating carry is fed into bit position 1.

For example, with inputs $a_4 \cdots a_1 = 1001$, $b_4 \cdots b_1 = 1010$, and c = 1, we would expect output $s_5 = 1$, $s_4 = 0$, and $s_3 s_2 s_1$ oscillating, because the 5-bit sum oscillates between 10011 and 10100.

The leaf cells used in the prototype are as described earlier with the exception of T cells (just non-inverting superbuffers to drive the connections to the output pads) and a G2 cell to implement the feedback described above. The G2 cell differs from the G cell in that $y_{out} = x_{in} \hat{} y_{in}'$, $z_{out} = x_{in} \hat{} y_{in}'$ (this change requires only the removal of one diffusion wire). The cells are arranged as shown in Fig. 11.

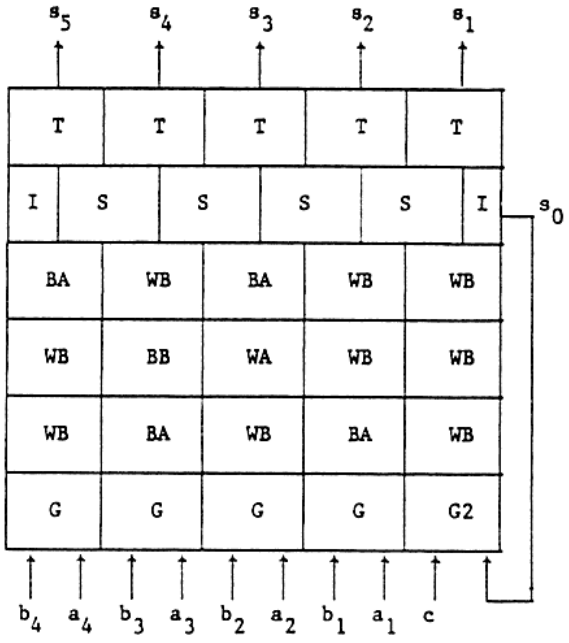


FIG. 11. CELL LAYOUT FOR CORE OF 5-BIT PROTOTYPE

* It is convenient here to use subscripts starting from 0 instead of 1.

The prototype is purely combinational, i.e. it does not include any clock lines. However, it would not be difficult to run polysilicon clock lines from east to west across some or all of the cells in order to give a synchronous (and possibly pipelined) adder.

(b) Leaf Cells

In laying out the leaf cells we tried to minimise width rather than height so that an n-bit adder would be as narrow as possible (see Table 1).

In the worst-case, the v and w lines which run east to west through WA cells have to cross about $\frac{n}{2}$ cells (though the prototype is too small to show this). Hence, we decided to run these lines in metal to avoid long polysilicon or diffusion lines. This made it desirable to run VDD and GND lines horizontally in metal through each cell. We did not attempt to share VDD and GND lines between adjacent cells because this would apparently save height at the expense of width.

Equivalent circuit diagrams for the BA, BB, G, G2 and S cells are shown in Figs. 12-15. (The circuit diagrams for WA, WB, I and T cells are omitted as they are trivial.) Note that our S cell has two less active devices (2 pullups and 5 pulldowns) than the "exclusive-or" circuit given in [7].

$$X_{OUT} = (X_{IN} \vee (Y_{IN} \wedge Y_{IN}))'$$
$$Y_{OUT} = (Y_{IN} \wedge W_{IN})'$$
$$Z_{OUT} = Z_{IN}'$$

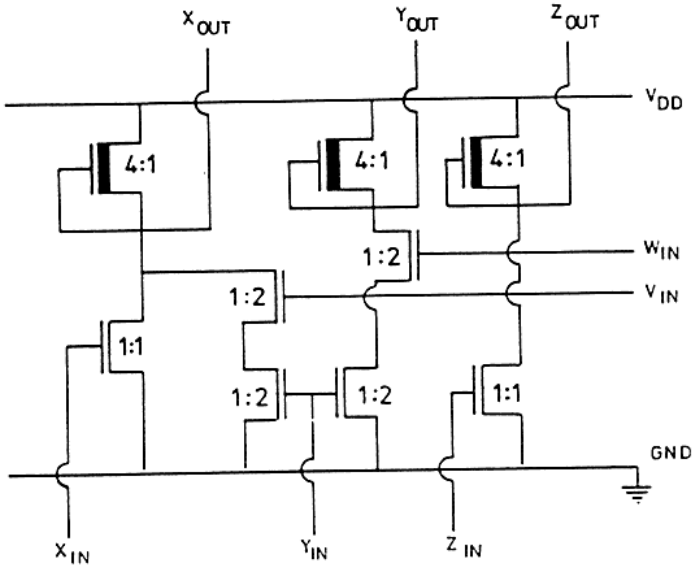
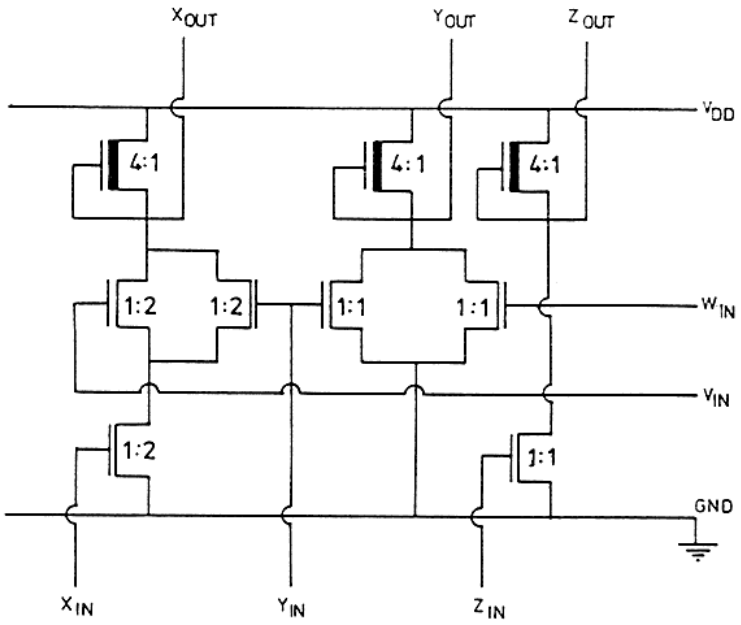


FIG. 12. CIRCUIT FOR BA CELL

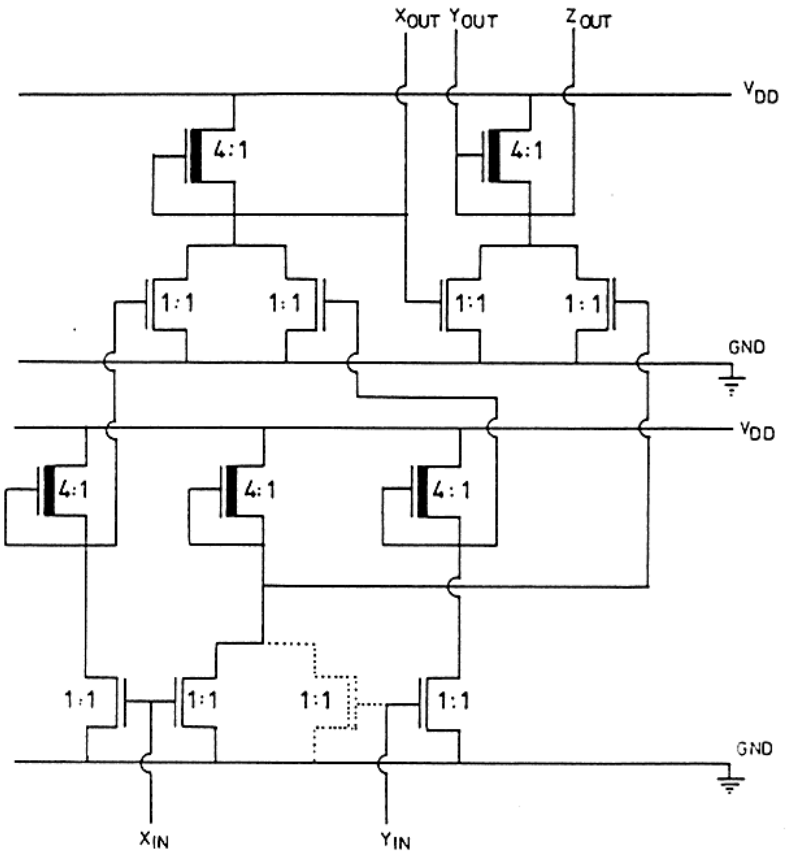


$X_{OUT} = (X_{IN} \wedge (Y_{IN} \vee V_{IN}))'$

$Y_{OUT} = (Y_{IN} \vee W_{IN})'$

$Z_{OUT} = Z_{IN}'$

FIG. 13. CIRCUIT FOR BB CELL



$X_{OUT} = X_{IN} \wedge Y_{IN}$

$Y_{OUT} = X_{IN} \odot Y_{IN}$

$Z_{OUT} = X_{IN} \odot Y_{IN}$

FIG. 14. CIRCUIT FOR G CELL (omit dotted lines for G2 cell)

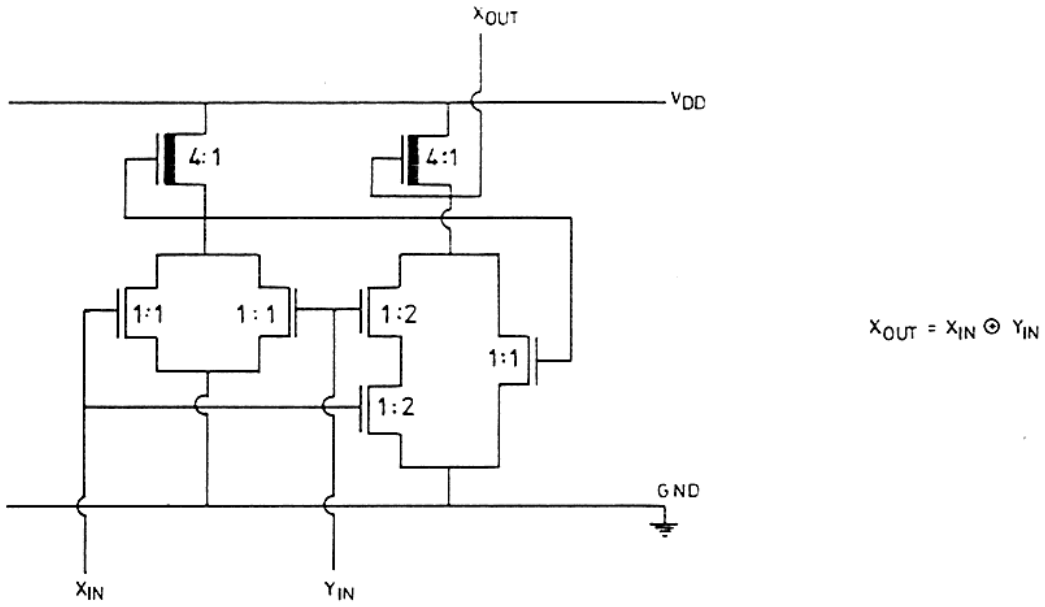


FIG. 15. CIRCUIT FOR S CELL

(c) Floor Plan

The floor plan and metal layer for the 5-bit prototype is shown in Fig. 16. Output pads are as in [4]; input pads incorporate a 4 λ-wide diffusion guard ring and a 12:1 inverter followed by an inverting superbuffer to provide TTL-compatibility and to drive the connections to the adder.

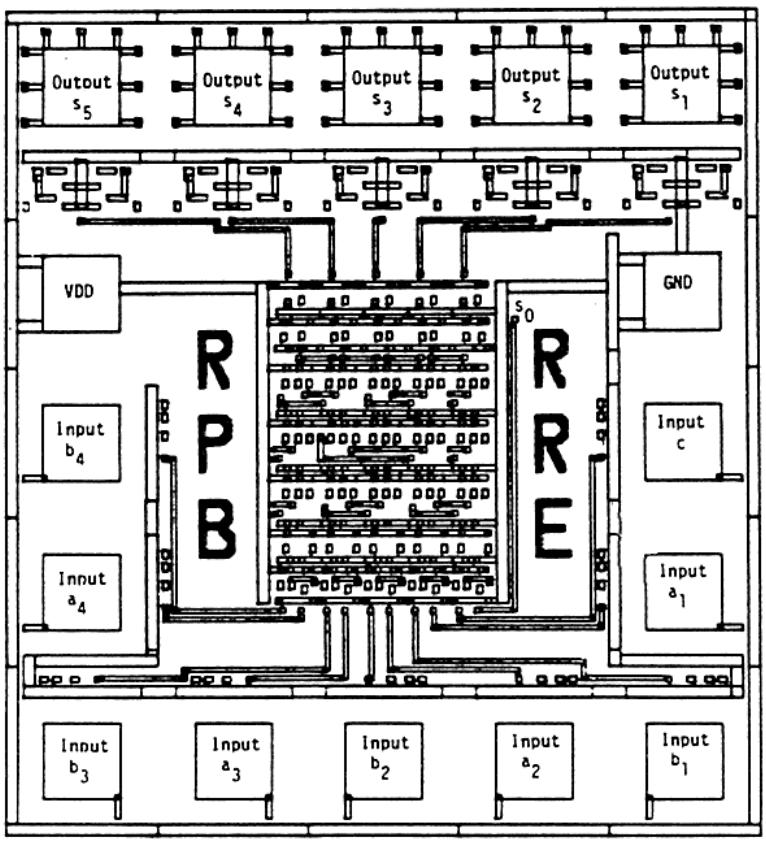


FIG. 16. PROTOTYPE FLOOR PLAN AND METAL LAYER

An 8 λ-wide metal VDD ring runs around the outer edges of the pads, while GND runs along the inner edges of the pads. All connections between the pads and the core are in 3 λ-wide metal and are driven by superbuffers. Details of the core are shown in Fig. 17.

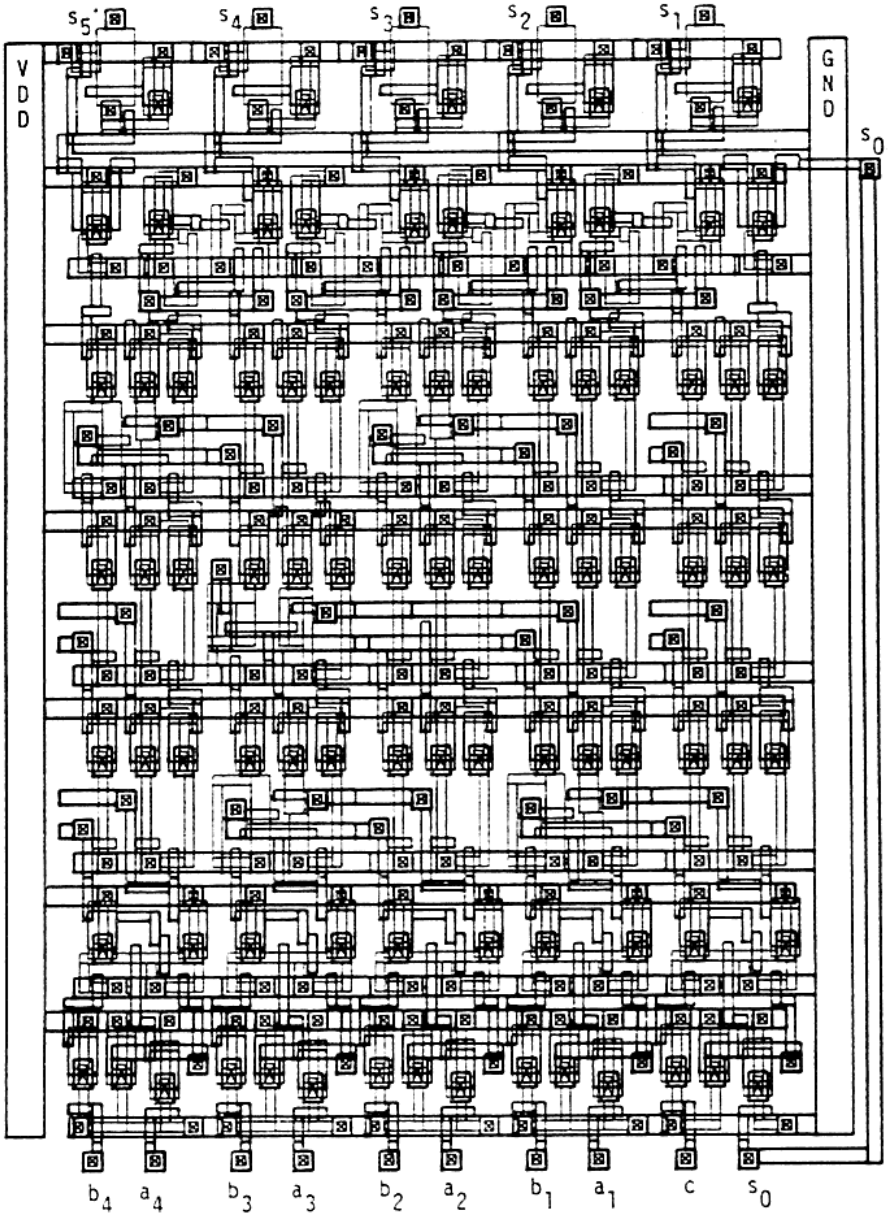


FIG. 17. DETAILED CORE LAYOUT

(d) Electrical Considerations

The DC power consumption of the prototype has been estimated, assuming electrical parameters as in [5, p.51], i.e. VDD = +5 v, V_{th} > + 1 v, gate resistance = 10 KΩ/□. The estimates for each cell and the total for the prototype are given in Table 3. The total is a conservative estimate because not all cells can draw their maximum power simultaneously. The estimates for the output pads assume a 2 KΩ load.

Cell type	Power per cell (mW)	Cells	Total (mW)
WA	1.50	1	1.50
WB	1.50	9	13.50
BA	1.56	4	6.24
BB	1.58	1	1.58
I	0.50	2	1.00
G	2.06	4	8.24
G2	2.00	1	2.00
S	1.06	4	4.24
T	2.00	5	10.00
input pad	3.00	9	27.00
output pad	20.00	5	100.00
			175.30

TABLE 3. MAXIMUM DC POWER CONSUMPTION ESTIMATES

From Table 3, the total DC power consumption is less than 180 mW, which is quite reasonable [5, p.133]. Note, however, that the power consumption of the adder core is about $20 W/cm^2$. Thus cMOS, with its low DC power consumption, might be a better technology than nMOS for implementing a large parallel adder. (Alternatively, the core could be redesigned to use "red-green function blocks" [5, Plate 10] where possible.)

Total DC current is estimated to be at most 36 mA (10 mA for the core, 26 mA for the pads), so the maximum current density in metal wires should not exceed 1 $mA/\mu m^2$ if wires are 1 µm thick. For an n-bit adder with $n > 24$ it might be necessary to increase the width of the 4 λ-wide metal VDD and GND lines crossing each cell, in order to keep the current density down to 1 $mA/\mu m^2$ [5, p.133]. (Similarly if scaling down λ reduced wire thickness below 1 µm.)

(e) Delay Estimation

Excluding the delay introduced by the input and output pads, connects, and T-cells (superbuffers), there are 4 inverter-pair delays associated with each path through the adder (1 inverter in each of 3 BW cells, 3 in a G cell, and 2 in an S cell). The effective pullup/pulldown ratios are approximately 4:1 and effective fanout $C/C_G \leq 5$. Allowing a margin for parasitic capacitances, each inverter-pair delay could be about 75 τ [5, p.12], giving a total worst-case delay of 300 τ. The value of τ is uncertain, but it probably lies in the range 0.25 to 1 ns [4, p.70; 5, p.35].

(f) Design Tools and Checking

The prototype was described in BELLE, a Pascal-based procedural layout language distributed by the CSIRO VLSI Program and modified to run on a Univac 1100/82 at the Australian National University. Leaf cells were drawn by hand on graph paper, then described and composed using

BELLE. Although BELLE was adequate for the task, it did not provide much assistance for mundane but time-consuming tasks like routing wires between the core and the pads.

BELLE generates CIF [4,5], which we could display on a Tektronix 4014 (high resolution monochrome storage tube) display or plot on a Calcomp 960 or Tektronix 4662 plotter (with colour provided by manual pen changes). A Pascal program, Viewcif (distributed by CSIRO and modified to run at ANU) was used to display or plot CIF. The "window" facility provided by Viewcif was useful for checking (on the display) the effect of small changes to the BELLE program.

No checking programs were available at ANU, so all design rule and electrical rule checking was performed by hand and eye. However, the CSIRO VLSI Program ran our CIF file through their suite of checking programs. No design rule or static electrical rule violations were detected, and logic-level simulation showed that the circuit should function as intended.

Conclusion

At the time of writing (May 1982) the prototype has not been fabricated. We hope that fabrication and testing will be complete by December 1982.

Independent of actual fabrication, detailed design of a prototype has shown that the Brent-Kung parallel adder (improved as described earlier) is practical for nMOS VLSI technology, and is a serious contender with the usual serial n-bit adder designs if $n > 16$.

Acknowledgements

Thanks are due to the CSIRO VLSI Program for making AUSMPC 5/82 possible, to Steve Rothwell for implementing BELLE and Viewcif on the ANU Univac 1100/82, and to Alan Bell for suggesting the idea of the feedback loop, described during the functional description of the prototype section of this article. □

References

[1] Bilardi, G., Pracchi, M. & Preparata, F.P. "A Critique and an Appraisal of VLSI Models of Computation", VLSI Systems and Computations (edited by Kung, H.T., Sproull, B. & Steele, G.) Computer Science Press, Rockville, Maryland, 1982, pp.81-88

[2] Brent, R.P. & Kung, H.T. "The Area-time Complexity of Binary Multiplication", J. ACM 28, 1981, pp.521-534

[3] Brent, R.P. & Kung, H.T. "A Regular Layout for Parallel Adders", IEEE Trans. Comps. C-31, 1982, pp.260-264

[4] Hon, R.W. & Sequin, C.H. "A Guide to LSI Implementation", (second edition), Xerox Palo Alto Research Center Report SSL-79-7, January 1980

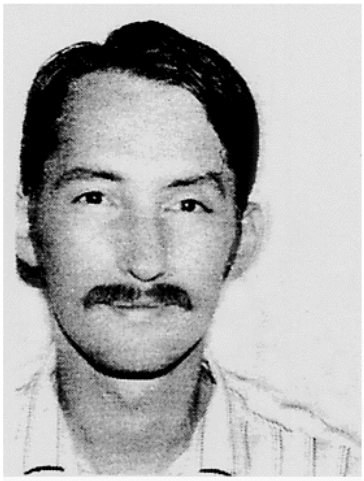
[5] Mead, C.A. & Conway, L.A. "Introduction to VLSI Systems", Addison-Wesley, 1980

[6] Mudge, J.C. & Clarke, R.J. "Australia's First Multiproject Chip Implementation System", Microelectronics '82, Inst. of Engineers, Australia, May 1982, pp.72-77

[7] Sandell, J. "An Unusual Exclusive NOR/OR Gate", AUSMPC Newsletter 1, March 1982, pp.28 and 29

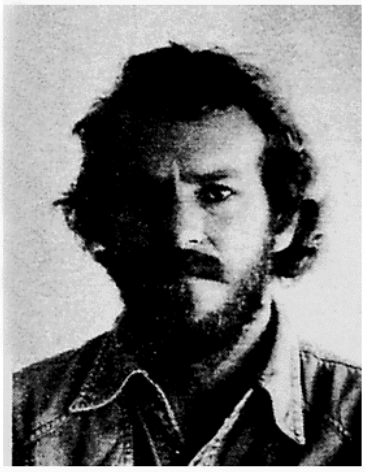
[8] Guibas, L. & Vuillemin, J. "On Fast Binary Addition in nMOS Technologies", preliminary draft, 1981.

EDITOR'S NOTE: This article is an edited version of the full report which is available from the authors.



Richard Brent obtained B.Sc. and D.Sc. degrees from Monash University, and a Ph.D. in Computer Science from Stanford University in 1971. After working at IBM Research, Yorktown Heights, in 1971-72, he moved to the Australian National University, where he is currently Professor of Computer Science.

Richard's interests include computer arithmetic, systolic arrays suitable for VLSI implementation, and the complexity theory of VLSI. He is a member of the IEEE, ACM, ACS etc. and a Fellow of the Australian Academy of Science.



Robert Ewin obtained a B.Sc. degree from Monash University in 1971 and an Arts degree from ANU in 1980. He worked with the Government for three years before becoming a tutor in the Department of Computer Science at the Australian National University from 1975 until 1980. His current position is as a programmer in the same Department.

Robert's interests include programming languages, formal and natural language theory, and the use of microcomputers in teaching and research environments.