# The Redundant Cell Adder

Tom Lynch and Earl Swartzlander
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas 78712

## Abstract

*This paper describes the design of the 56-bit significand adder for the Advanced Micro Devices Am29050[1] microprocessor. This is a 1 μm design rule CMOS realization of a high performance RISC microprocessor that implements IEEE Standard 754 floating point arithmetic. To achieve an add time of under 4 ns for the 56-bit significand and to avoid multistage pipelines which significantly impair compiler efficiency, a redundant cell adder has been developed. This new design is a key to realizing the high performance for floating point arithmetic that is achieved by the Am29050 microprocessor.*

## 1 Introduction

Classic high speed adders include carry lookahead, carry skip, carry select, and conditional sum adders. A variant of the binary carry lookahead adder was analyzed by Brent and Kung [1]. Another variant, based on reformulating the carry equations was developed by Ling [2]. Fast implementations of carry lookahead adders have been developed by Hwang and Fisher [3] in CMOS and Bewick, Song, and Flynn [4] in bipolar ECL. For large word sizes, the carry lookahead adder is generally considered to be the fastest. For gate level designs where each gate has unit delay, the carry lookahead adder greatly reduces delay with a modest increase in complexity relative to the ripple carry adder. In dynamic CMOS implementations where the delay depends on the gate size and loading, the use of Manchester carry chains [5] to realize the carry lookahead logic significantly reduces the gate loading, which produces a substantial speed increase.

This paper presents the design of a redundant cell adder implemented in dynamic CMOS which combines carry lookahead adders realized with Manchester carry chains and the carry select adder concept to achieve approximately twice the speed of the traditional carry lookahead adder. Because this is a cellular adder that performs redundant calculations to generate the control signals for a carry select adder, it is referred to as the "redundant cell adder".

## 2 Logic Design of the Redundant Cell Adder

Notation: $a_i$ and $b_i$ denote the i-th bits of the words to be added, $p_i$ denotes that a carry will propagate across bit position i (i.e., $p_i = a_i + b_i$), and $g_j$ denotes that a carry is generated at bit position j (i.e., $g_j = a_j b_j$). $p_{i:j}$ signifies that a carry will propagate from bit j (LSB) to bit i (MSB). Similarly $g_{i:j}$ denotes that a carry is generated in each of the bit positions from j to i inclusive. The fundamental carry operation, "fco," introduced by Brent and Kung [1] is used:

$$(p_{i:j},g_{i:j}) = (p_{i:k+1},g_{i:k+1})\text{fco}(p_{k:j},g_{k:j}) \quad (1)$$

which is defined as:

$$p_{i:j} = p_{i:k+1}p_{k:j} \text{ and } g_{i:j} = g_{i:k+1} + p_{i:k+1}g_{k:j}$$

Brent and Kung have shown the associativity of the fco. For: $i > m > k > j$.

$$[(p_{i:m+1},g_{i:m+1}) \text{ fco } (p_{m:k+1},g_{m:k+1})] \text{ fco } (p_{k:j},g_{k:j}) = (p_{i:m+1},g_{i:m+1}) \text{ fco } [(p_{m:k+1},g_{m:k+1}) \text{ fco } (p_{k:j},g_{k:j})] \quad (2)$$

A block diagram of a 64 bit redundant cell adder is shown in Figure 1. The adder uses a tree of Manchester carry chain carry lookahead modules (on the left) to calculate the carries for each of the eight bit carry select adders (on the right). On the left, the Manchester carry chain carry lookahead modules (Mcc) produces group generate and propagate signals for four bit groups, except for the least significant group, where carry $c_4$ is produced. On the second level of the tree, the first level signals are

[1] Am29050 is a trademark of Advanced Micro Devices

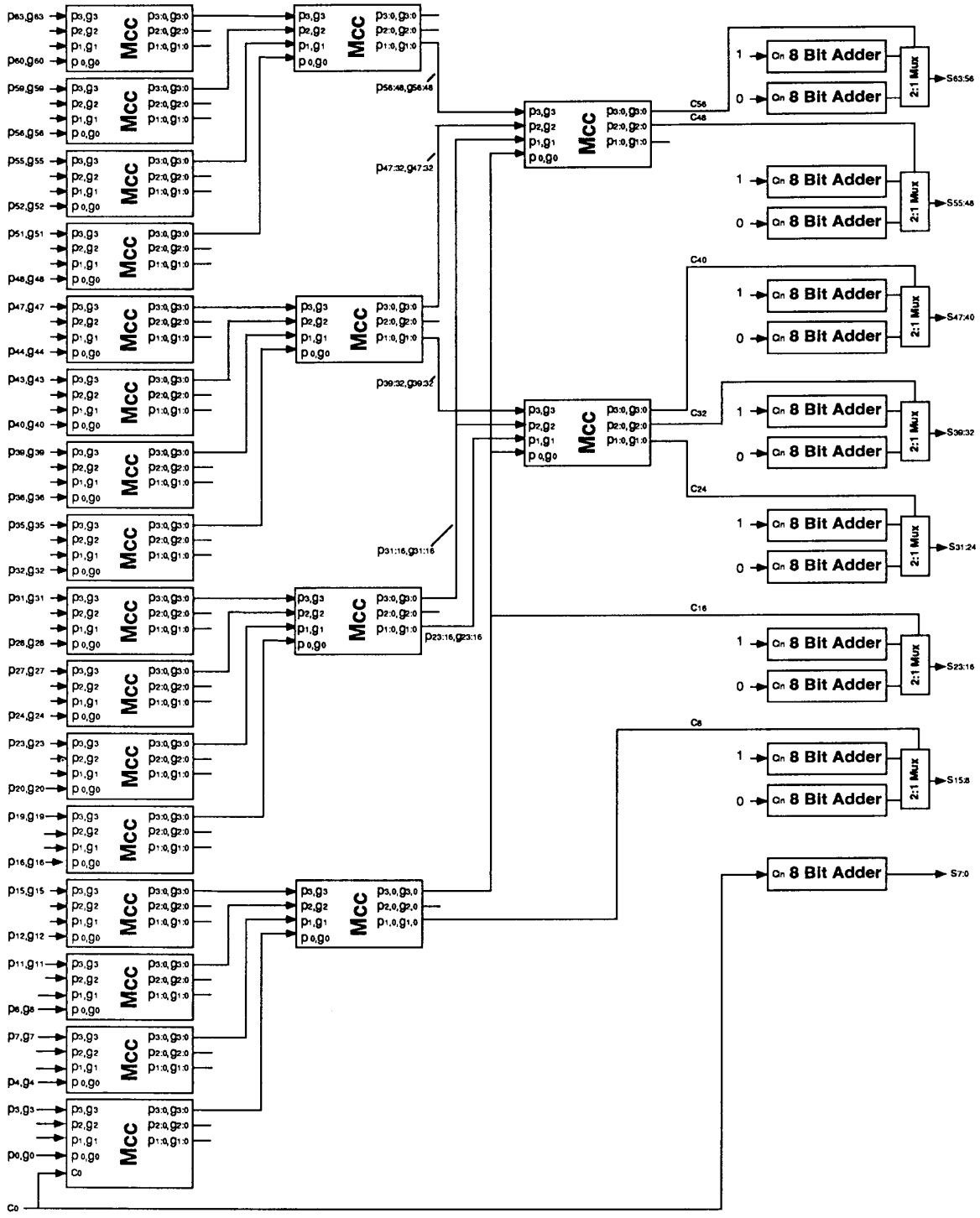


**Figure 1. Redundant Cell Adder Block Diagram.**

combined to produce group propagate and generate signals for 16 bit boundaries, and overlapping eight bit boundaries. For example, $p_{47:32}$,$g_{47:32}$ and the overlapping eight bit boundary signals $p_{39:32}$,$g_{39:32}$ are produced. Carries $c_8$ and $c_{16}$ are available from the least significant group on the second level.

On the third level two Manchester carry chain modules are used. The most significant third level module combines the three less significant 16 bit boundary group propagate and generate signals, $p_{15:0}$,$g_{15:0}$; $p_{31:16}$,$g_{31:16}$; and $p_{47:32}$,$g_{47:32}$, and the most significant eight bit boundary group propagate and generate signals $p_{55:48}$,$g_{55:48}$ to produce carries $c_{48}$ and $c_{56}$ (note that carry $c_{32}$ is generated both here and at the lower module in this column).

The idempotency of the fundamental carry operation must be shown to explain the operation of the least significant third level carry lookahead module. As shown in Equation (2), combining group p and g signals with themselves produces correct results. The result for propagate follows trivially since ANDing any signal, say x, with itself returns x, while the result for generate follows by factoring $g_{i:j} + p_{i:j}g_{i:j} = g_{i:j}\,(1 + p_{i:j}) = g_{i:j}$. Thus:

$$(p_{i:j},g_{i:j}) = (p_{i:j},g_{i:j}) \text{ fco } (p_{i:j},g_{i:j}) \qquad (3)$$

Because of this idempotency property, the fundamental carry operation can be applied to overlapping regions:

*Theorem:*
Given $i > m \geq k > j$, then $(p_{i:j},g_{i:j})$ can be derived from $(p_{i:k},g_{i:k})$ fco $(p_{m:j},g_{m:j})$

*Proof:*
The proof begins by applying Equation (1) to expand both terms in $(p_{i:k},g_{i:k})$ fco $(p_{m:j},g_{m:j})$:

$$(p_{i:k},g_{i:k}) = (p_{i:m+1},g_{i:m+1}) \text{ fco } (p_{m:k},g_{m:k}) \qquad (4)$$
and
$$(p_{m:j},g_{m:j}) = (p_{m:k},g_{m:k}) \text{ fco } (p_{k-1:j},g_{k-1:j}) \qquad (5)$$

Substituting these for $(p_{i:k},g_{i:k})$ and $(p_{m:j},g_{m:j})$ in $(p_{i:k},g_{i:k})$ fco $(p_{m:j},g_{m:j})$ yields:

$$(p_{i:k},g_{i:k}) \text{ fco } (p_{m:j},g_{m:j}) = [(p_{i:m+1},g_{i:m+1}) \text{ fco } (p_{m:k},g_{m:k})] \text{ fco } [(p_{m:k},g_{m:k}) \text{ fco } (p_{k-1:j},g_{k-1:j})]$$

Associativity from Equation (2) is applied to move the brackets:

$$(p_{i:k},g_{i:k}) \text{ fco } (p_{m:j},g_{m:j}) = [(p_{i:m+1},g_{i:m+1}) \text{ fco } [(p_{m:k},g_{m:k}) \text{ fco } (p_{m:k},g_{m:k})]] \text{ fco } (p_{k-1:j},g_{k-1:j})$$

By the idempotency of the fco operation from Equation (3) $[(p_{m:k},g_{m:k}) \text{ fco } (p_{m:k},g_{m:k})]$ reduces to $(p_{m:k},g_{m:k})$:

$$(p_{i:k},g_{i:k}) \text{ fco } (p_{m:j},g_{m:j}) = [(p_{i:m+1},g_{i:m+1}) \text{ fco } (p_{m:k},g_{m:k})] \text{ fco } (p_{k-1:j},g_{k-1:j})$$

Applying Equation (1) to the bracketed quantity:

$$(p_{i:k},g_{i:k}) \text{ fco } (p_{m:j},g_{m:j}) = (p_{i:k},g_{i:k}) \text{ fco } (p_{k-1:j},g_{k-1:j})$$

Applying Equation (1) again:

$$(p_{i:k},g_{i:k}) \text{ fco } (p_{m:j},g_{m:j}) = (p_{i:j},g_{i:j}) \qquad \text{QED}$$

Since overlapping regions can be combined with the fundamental carry operation, the overlapping group propagate and generate signals $p_{31:16}$,$g_{31:16}$; $p_{23:16}$, $g_{23:16}$; and $c_{16}$ can be applied to the least significant carry lookahead module on the third level to produce carries for $c_{24}$, $c_{32}$, and $c_{40}$. Since carries on all eight bit boundaries are now known, these are used in standard carry select fashion to select the correct result from the eight bit adders.

## 3 Implementation of the Redundant Cell Adder

The redundant cell adder was implemented for the Am29050 microprocessor using AMD's 1 μm minimum drawn feature size, two layer metal CMOS process. The basic Manchester carry chain cell is shown in Figure 2. Outputs are tapped from the appropriate points in the chain. In order to make the Manchester carry chain as fast as possible, each series transistor is sized to approximately fill the bit cell where it is placed. The tree is laid out by placing the Mcc modules in roughly the same position as they are shown in the block diagram of Figure 1. The third level (and higher level if necessary) Mcc modules are placed in holes left in the column of second level Mcc modules to reduce the width of the layout.

The adder floor plan is shown in Figure 3. Metal two is used for long horizontal runs carrying the input operands, the bit propagate and generate signals, the calculated carries, and the results. Metal one runs vertically, and is used for local interconnect. The LSB of the adder is at the bottom. The inputs come from the left, while the outputs leave at the right. The first block in the floor plan (going from left to right) is the propagate and generate logic. The second block is a stack of four bit Mcc modules connected in pairs to form eight bit ripple carry sections. A ZERO carry is input to each section in this column.

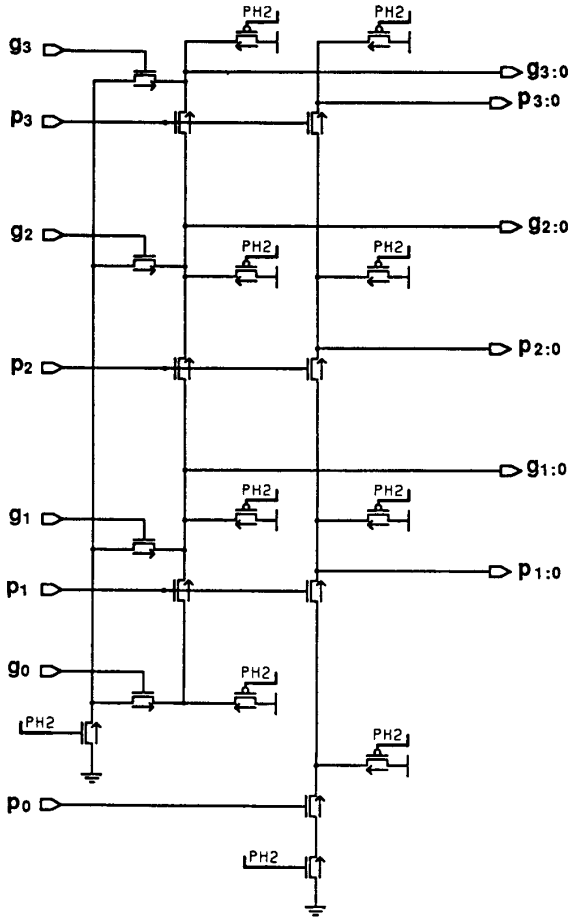


**Figure 2. Manchester Carry Chain Cell.**

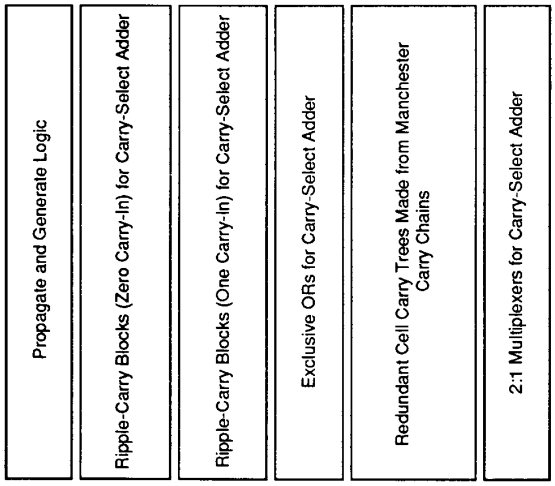


**Figure 3. Redundant Cell Adder Floor Plan.**

The third column consists of ripple carry sections with a carry in of ONE. Next are the exclusive-OR gates for calculating the sums from the ripple carries and the bit propagates. The fifth column is the carry tree. The last column consists of multiplexers which select between the eight bit add results.

For the IEEE 754 floating point standard, double precision significand calculation, only 56 bits are required, so carry $c_{56}$ is the carry out, and the top carry select adder (bits 56 through 63) and its multiplexer are omitted. Also, the associated carry tree logic may be omitted, which includes the top four Manchester carry chain blocks on the first level and the top Manchester carry chain block on the second level.

## 4 Performance

The width of the adder is roughly proportional to log n. The second level of the adder has holes large enough to contain three Manchester carry chains, so the third (and fourth if required) levels of the tree can be packed into the second level for adders of up to 256 bits. The total width is the sum of the widths of the blocks in the floor plan:

$$W = W_{pg} + (1 + \log_{16}N)\, W_{Mcc} + W_{csa8} \qquad (6)$$

Where: W is the total width of the N bit adder, $W_{pg}$ is the width of the propagate/generate logic, $W_{Mcc}$ is the width of the Manchester carry chain logic, and $W_{csa8}$ is the width of the eight bit carry select adder. W is 450 μm wide (for $24 \le N < 256$), as implemented in the AMD 1μm CMOS technology. The total area is the product of the width times the height:

$$A = W\, N\, H_c \qquad (7)$$

Where: $H_c$ is the height of a bit slice adder cell. Since each bit cell is 71.6 μm high, the 56 bit adder shown on Figure 4 is 4010 μm high for a total area of $1.8 \times 10^6$ square μm. The height and area of adders for word sizes from 24 to 256 bits scales in direct proportion to the wordsize.

As with any carry select adder, the delay is determined by the slower of the sum computation and the carry computation. In this implementation, the sum computation is faster. For the carry computation, propagate and generate signals are already set up when the clock asserts. First, the least significant Manchester carry chain in the first level fires. The critical delay path signal travels through three levels of Manchester carry chains. The worst case load is seen by $c_{16}$ since it drives two

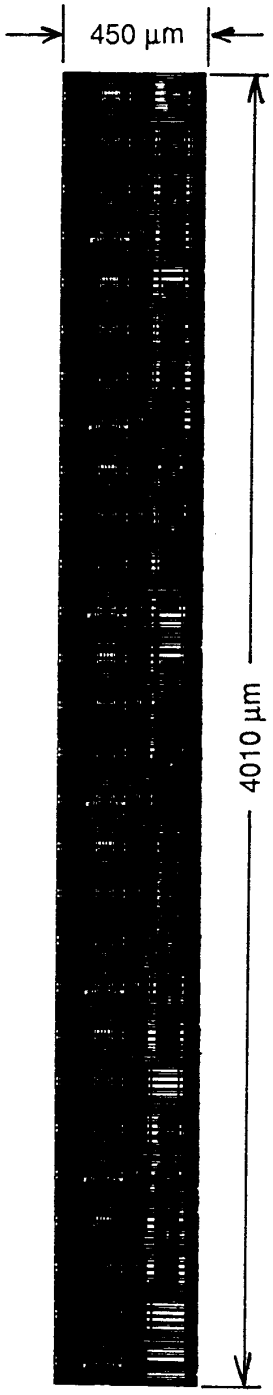


Figure 4. Redundant Cell Adder Layout.

Manchester carry chain inputs; the multiplexers are well buffered, so they present a small load. Finally the critical carry signal arrives at the multiplexer for the most significant eight bit adder.

$$D = (\log_4 N)\, D_{Mcc} + D_{mux} \quad (8)$$

Where: D is the delay of the N bit adder, $D_{Mcc}$ is the delay of the Manchester carry chain, and $D_{mux}$ is the delay of a 2:1 multiplexer. In the Am29050 microprocessor, the time from the rising edge of the clock to the sum is approximately 3.2 ns as shown in the electron beam timing waveforms on Figure 5. With comparable CMOS technologies the speed should be nearly constant for $24 \leq N < 64$.

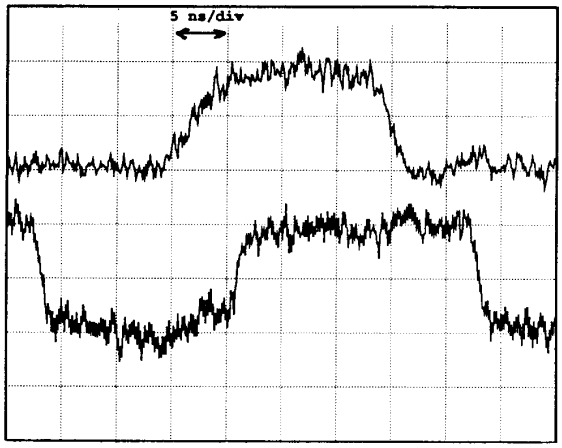


**Figure 5. Measured Performance of the Redundant Cell Adder. The Top Waveform is the Clock and the Lower Waveform is the Most Significant Sum Bit.**

## 5 Conclusion

The idempotence of the fundamental carry operation allows the construction of a fast adder that has small performance degradation as the add width grows. This adder is very fast, achieving a 3.2 ns measured add time for 56 bit operands and is of reasonable size, as shown by the Am29050 microprocessor implementation.

## Acknowledgement

The design and initial implementation of the redundant cell adder was performed by the first author while he was employed by Advanced Micro Devices. Special thanks

are due Stephen McIntyre, Danny English, and Tom Burghart for their help in implementing this adder.

## References

[1] R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders," *IEEE Transactions on Computers*, Vol. C-31, 1982, pp. 260-264.

[2] H. Ling, "High-Speed Binary Adder," *IBM Journal of Research and Development*, Vol. 25, pp. 156-166, May 1981.

[3] I. S. Hwang and A. L. Fisher, "A 3.2 ns 32-bit CMOS Adder in Multiple Output Domino Logic," *1988 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 140, 141, 332, and 333.

[4] G. Bewick, P. Song, G. DeMichel, and M. J. Flynn, "Approaching a Nanosecond; a 32-bit Adder," *Proceedings of the 1988 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 221-226.

[5] T. Kilburn, D. B. G. Edwards, and D. Aspinall, "Parallel Addition in Digital Computers: A New Fast "Carry" Circuit," *IEE Proceedings*, Vol. 106, pt. B, 1959, pp. 464-466.