A Simple Ratioed Self-Clocked Regenerative Differential Multistate Adder

Piotr Mitros, Anant Saraswat

Abstract— **We present a design for a 64 bit adder that uses an analog neural network-type circuit for computing carries. The circuit is composed of cascaded two bit stages. Each stage sums the input and carry bits with appropriate weights together with the inverse bits with negative weights, and digitizes the result to generate a carry bit. In this way, we can compute the carry for several inputs bits in one stage.**

Keywords— **Adder, Neural Network, Differential, Low Speed, Self Clocked**

I. Introduction

TRADITIONALLY, adders are built using pure digital techniques, and so can calculate at most one bit per stage. We present a ripple adder based on a crude current-mode DAC, wire add, and a sense amp compare stage, to generate carry bits for future stages. This is shown in figure 2. With this circuit, we can compute the carry bit for several input bits in one stage.

II. Circuit Overview

The schematic for the carry computation of a single stage is shown in figure 3. Our key insight is that a carry bit simply tells us if more than half of the input bits are high. Hence, we can use a sense amplifier to compare the number of high bits in (a, b, c) to $\left(\bar{a}, \bar{b}, \bar{c}\right)$. We implement this as a sense amplifier with multiple input transistors.

In a basic circuit, we would have three transistors per side: a, b and c on one, and $\bar{a}$, $\bar{b}$ and $\bar{c}$ on the other to generate c_{out} and $\bar{c}_{out}$. We can extend this by replacing c_n with a_{n-1}, b_{n-1}, c_{n-1}, scaling each to be a third of the size.

We feed in a clock to the first stage when all of the inputs are ready. The carry outputs of each stage are fed into a NAND gate, and the output of the NAND gate drives the clock of the successive stage. Hence, the circuit is fully self-clocked.

We found it necessary to buffer the carry bits after several stages (in the final circuit, every four stages) to prevent degradation. Due to the fully differential nature of the circuit, we could use a single inverter as a buffer.

We used a traditional static CMOS full adder to generate the final sum from the carry bits from the previous stage's sense amplifier and the original inputs. We also place an identical complement adder for full differential outputs, as well as for matching.

III. Layout

The full floorplan is shown in figure 6, and would take about $12000\mu^2$. The wire parasitics are shown in the appendix in the Spice file, as is a layout of the critical main

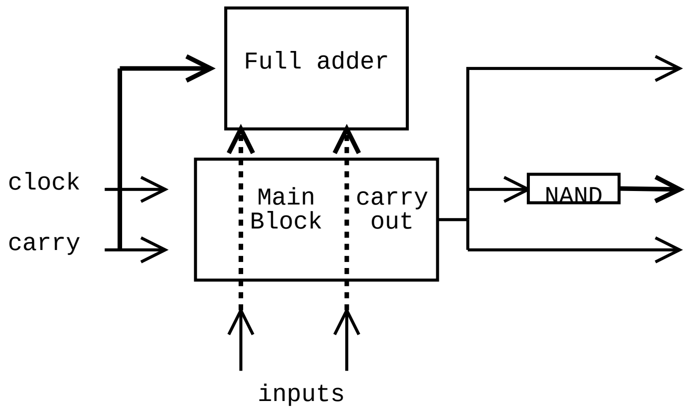

Fig. 1. Four Bit Slice

block. To improve matching, we used multiple same size transistors to emulate larger ones. We designed a skew-symmetric layout for the sense amplifier to give good cancelling of mismatch inspite of the inherent unsymmetrical layout. Beyond that, layout was completely symmetrical for the differential portions of the circuit. Cascaded, the layout was somewhat unorthodox, as multiple alternating positive and negative power supply rails ran the width of the circuit, rather than longer rails on the sides.

IV. Extending to More Bits

In the initial circuit, we only computed two input bits per stage.

In theory, this architecture can scale arbitrarily, with no decrease in RC time constant, since as we add transistors, R scales linearly with C. In practice, we run into several constraints. We are much more sensitive to noise levels and device mismatches. Next, as we increase the number of bits per stage, the closest starting points for the sense amp move in, and the sense amp slows down. Finally, as we scale gate size, the size of the driving transistor must increase.

In practice, we found that computation with even three bits was difficult (though possible), and we ended up using two bits per stage. Beyond three, we were crippled by noise problems, presumably stemming from charge injection, imperfect matching of transistors, unmatched input voltage levels and times, and swarms of other analog effects. We believe these problems could be overcome with better self-biasing schemes, but not without significantly more work.

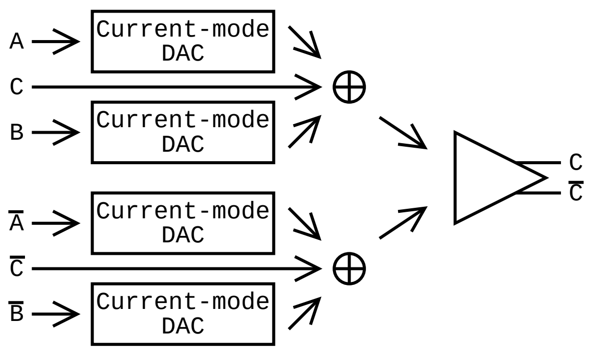

Fig. 2. Carry Stage Block Diagram

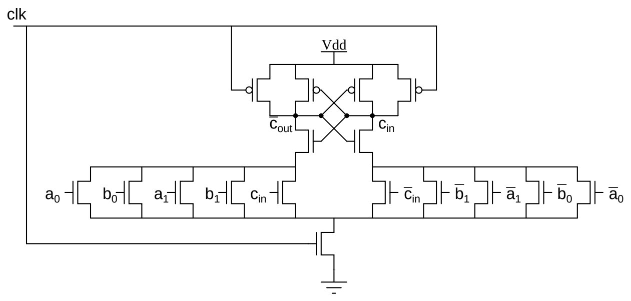

Fig. 3. Calculate Carry Stage

V. Results and Conclusion

The worst-case scenario occurs when the input bit vectors to each stage differ by only one LSB. The table lists the worst-case delay, as measured from the rising edge of the input clock to a change in the last carry bit, under different process corners. Figures 5 and 4 graphically show the worst-case transitions for the typical and slow corners, respectively.

Fast Corner/$3.3V/0^o$	$41ns$
Typical Corner/$3V/27^o$	$57ns$
Slow Corner/$2.7V/85^o$	$85ns$

These results were achieved without implementing traditional logarithmic carry-lookahead or carry-select techniques. With these improvements, our speeds would improve further. We also believe that more bits could be combined per stage for an additional improvement in speed.

Copies of the Spice and Magic files are available in /mit/pmitros/classes/6.384/project/ and /mit/saraswat/6.374/proj-handin/.

VI. Acknowledgments

We would like to thank Tom Knight, who suggested combining a differential ratioed domino stage with the sense amplifier in a previous iteration of the circuit. We would also like to express gratitude to all the nonelectonics people who were patient enough to listen to us ranting on about how our circuit worked.

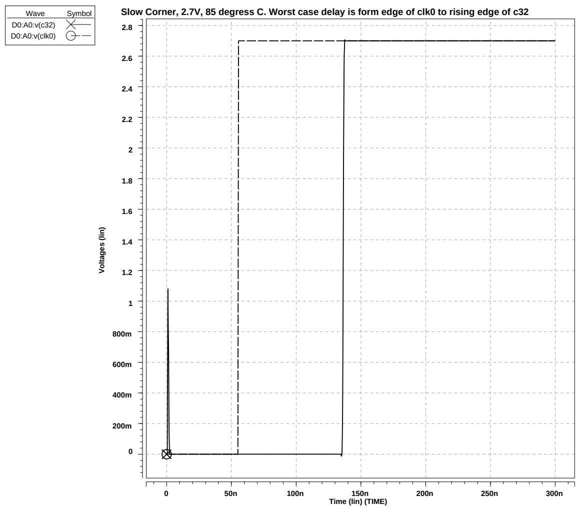

Fig. 4. Worst-case performance, slow corner

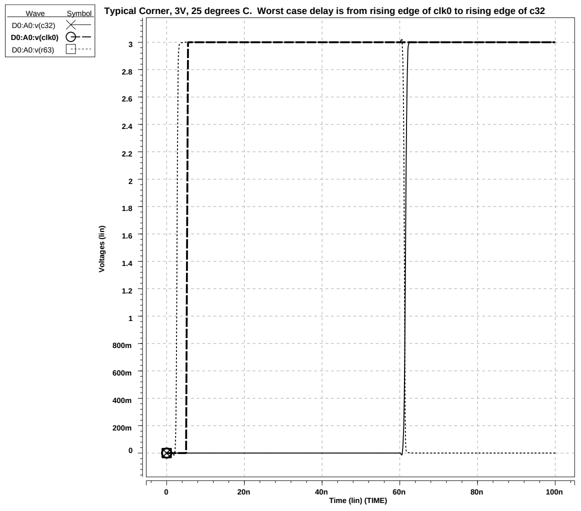

Fig. 5. Worst-case performance, typical corner

Piotr Mitros and Anant Saraswat were trained in the art of circuit hacking through the world renowned Random Nerds program. They can be reached at pmitros@circuit-hacker.org and saraswat@circuit-hacker.org.

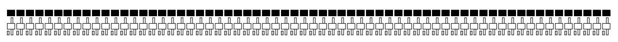

Fig. 6. Floor plan