A Segmented Parallel-Prefix VLSI Circuit with Small Delays for Small Segments

Bradley C. Kuszmaul[*]
MIT CSAIL, The Stata Center, 32 Vassar St. Cambridge, Massachusetts 02420
bradley@mit.edu

ABSTRACT

I present a VLSI circuit for segmented parallel prefix with gate delay $O(\log S)$ and wire delay $O(\sqrt{S})$ for segment size S, and total area $O(N)$. Thus, for example, for the problem of adding random numbers, in most cases, the addition would complete with only $O(\log\log N)$ gate delay and $O(\sqrt{\log N})$ wire delay.

Categories and Subject Descriptors: B.7.2 **[Integrated Circuits]**: Design Aids—*Layout.*

General Terms: Algorithms, Design.

Keywords: VLSI layout parallel-prefix circuit.

1. SUMMARY

One of the limitations to increasing clock speed for microprocessors is the time required for a VLSI circuit to perform an addition of two integers [1]. Fast-carry-lookahead circuits implement addition of N-bit numbers in order $O(\log N)$ gate delays, and, with the proper VLSI layout, in area $O(N)$ and with $O(\sqrt{N})$ wire delay, which is tight for the general problem of computing a value across an entire chip. Figure 1 shows an example layout of a circuit that achieves only $O(\log S)$ gate delays and $O(\sqrt{S})$ wire delay, where S is the length of the longest carry chain, however.

The basic idea is that for fast-carry lookahead, the values from some distant parts of the chips can be ignored whenever the carry is "killed". The same idea works for any segmented parallel prefix circuit.

To achieve these bounds both the gate delays and the wire delays must be kept small. To reduce the gate delays, insert "shortcuts" (shown as dashed lines in the figure) between distant-cousin subtrees that are adjacent in the carry chain. To reduce the wire delays, lay out subtrees in a modified H-tree layout (reflected by the node numbering in the figure) so that the Euclidean distance between leaves i and j is $O(\sqrt{|j-i|})$. See [4] for more details.

Related work

Cheng et al [2] describes carry-lookahead adder circuitry with $O(\log N)$ worst case gate delay using self-timed circuitry, and $O(\log\log N)$ average case gate delay for addition of random numbers. In contrast, this result addresses gate delay, wire delay, and chip area, and also shows how to achieve corresponding results for any segmented parallel prefix circuit.

Greenberg [3] shows how to implement fat-tree routing network with shortcuts that are similar to the shortcuts used here. An analogous performance property for search trees is the "finger" property described by Sleator and Tarjan [7].

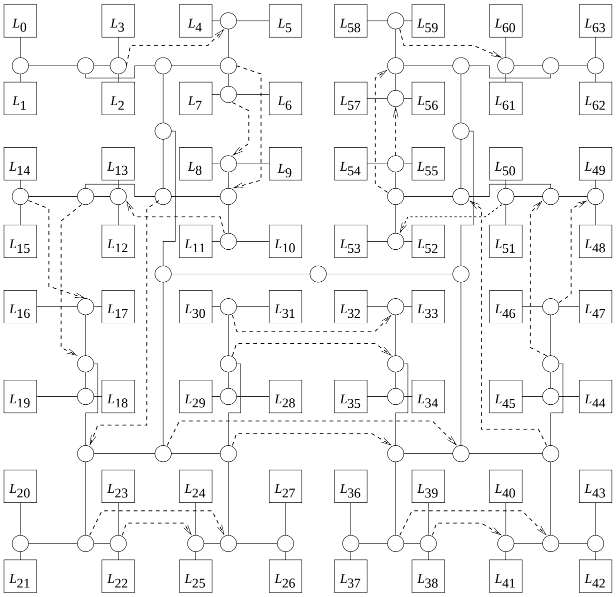

Figure 1: A 64-node parallel prefix tree with small delays for small segments. The leaves are labeled $L_0,\ldots,L_{63}$. Shortcut wires are shown as dashed arrows.

In the past, parallel prefix circuits have been important for parallel computers (see, e.g., [6]), but I came across this problem in a discussion with Andy Glew about the scaling of superscalar processors, where it turns out almost everything done by a superscalar processor can be done asymptotically optimally using segmented parallel prefix circuits [5].

2. REFERENCES

[1] D. Brooks and M. Martonosi. Dynamically exploiting narrow width operands to improve processor power and performance. In *HPCA'99*, pages 13–22, Jan. 1999.

[2] F.-C. Cheng, S. H. Unger, and M. Theobald. Self-timed carry-lookahead adders. *IEEE Trans. Comput.*, 49(7):659–672, July 2000.

[3] R. I. Greenberg. The fat-pyramid and universal parallel computation independent of wire delay. *IEEE Trans. Comput.*, 43(12):1358–1364, 1994.

[4] B. C. Kuszmaul. A segmented parallel-prefix vlsi circuit with small delays for small segments (full version). `http://kuszmaul.net/p/Kuszmaul05b`, MAY 2005.

[5] B. C. Kuszmaul, D. S. Henry, and G. H. Loh. A comparison of asymptotically scalable superscalar processors. *Theory of Comput. Syst.*, 35:129–150, 2002.

[6] C. E. Leiserson et al. The network architecture of the Connection Machine CM-5. *J. Parallel & Dist. Comp.*, 33(2):145–158, 1996.

[7] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, July 1985.

[*]This research was supported in part by the Singapore-MIT Alliance, NSF CAREER Grant 9702980, and NSF Grants ACI-0324974, and CNS-0305606.

SPAA'05, July 18–20, 2005, Las Vegas, Nevada, USA.
ACM 1-58113-986-1/05/0007.