A Taxonomy of Parallel Prefix Networks

David Harris
Harvey Mudd College / Sun Microsystems Laboratories
301 E. Twelfth St. Claremont, CA 91711
David_Harris@hmc.edu

Abstract - Parallel prefix networks are widely used in high-performance adders. Networks in the literature represent tradeoffs between number of logic levels, fanout, and wiring tracks. This paper presents a three-dimensional taxonomy that not only describes the tradeoffs in existing parallel prefix networks but also points to a family of new networks. Adders using these networks are compared using the method of logical effort. The new architecture is competitive in latency and area for some technologies.

I. Introduction

A parallel prefix circuit computes N outputs $\{Y_N, \ldots, Y_1\}$ from N inputs $\{X_N, \ldots, X_1\}$ using an arbitrary associative two-input operator $\circ$ as follows [13]

$$\begin{aligned} Y_1 &= X_1 \\ Y_2 &= X_2 \circ X_1 \\ Y_3 &= X_3 \circ X_2 \circ X_1 \\ &\vdots \\ Y_N &= X_N \circ X_{N-1} \circ \cdots \circ X_2 \circ X_1 \end{aligned} \quad (1)$$

Common prefix computations include addition, incrementation, priority encoding, etc. Most prefix computations precompute intermediate variables $\{Z_{N:N}, \ldots, Z_{1:1}\}$ from the inputs. The prefix network combines these intermediate variables to form the prefixes $\{Z_{N:1}, \ldots, Z_{1:1}\}$. The outputs are postcomputed from the inputs and prefixes. For example, adders take inputs $\{A_N, \ldots, A_1\}$, $\{B_N, \ldots, B_1\}$ and C_{in} and produce a sum output $\{S_N, \ldots, S_1\}$ using intermediate generate (G) and propagate (P) prefix signals. The addition logic consists of the following calculations and is shown in Fig. 1.

- **Precomputation**: $$G_{i:i} = A_i \bullet B_i \quad ; \quad G_{0:0} = C_{in} \qquad P_{i:i} = A_i \oplus B_i \quad ; \quad P_{0:0} = 0 \quad (2)$$
- **Prefix:** $$G_{i:j} = G_{i:k} + P_{i:k} \bullet G_{k-1:j} \qquad P_{i:j} = P_{i:k} \bullet P_{k-1:j} \quad (3)$$
- **Postcomputation:** $$S_i = P_i \oplus G_{i-1:0} \quad (4)$$

There are many ways to perform the prefix computation. For example, serial-prefix structures like ripple carry adders are compact but have a latency O(N). Single-level carry lookahead structures reduce the latency by a constant factor. Parallel prefix circuits use a tree network to reduce latency to

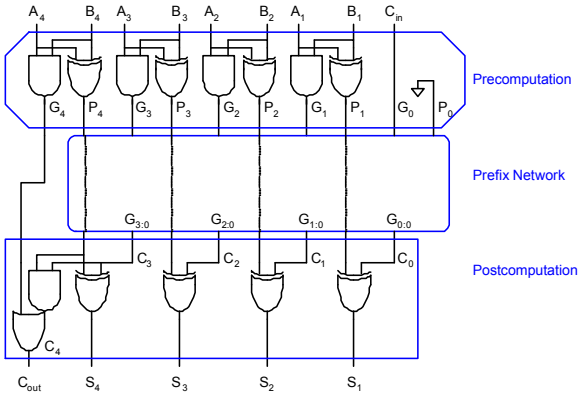

Fig. 1. Prefix computation: 4-bit adder

O(log N) and are widely used in fast adders, priority encoders [3], and other prefix computations. This paper focuses on valency-2 prefix operations (i.e. those that use 2-input associative operators), but the results readily generalize to higher valency [1].

Many parallel prefix networks have been described in the literature, especially in the context of addition. The classic networks include Brent-Kung [2], Sklansky [11], and Kogge-Stone [8]. An ideal prefix network would have $\log_2 N$ stages of logic, a fanout never exceeding 2 at each stage, and no more than one horizontal track of wire at each stage. The classic architectures deviate from ideal with $2\log_2 N$ stages, fanout of $N/2+1$, and $N/2$ horizontal tracks, respectively. The Han-Carlson family of networks [5] offer tradeoffs in stages and wiring between Brent-Kung and Kogge-Stone. The Knowles family [7] similarly offers tradeoffs in fanout and wiring between Sklansky and Kogge-Stone and the Ladner-Fischer family [10] offers tradeoffs between fanout and stages between Sklansky and Brent-Kung. The Kowalczuk, Tudor, and Mlynek prefix network [9] has also been proposed, but this network is serialized in the middle and hence not as fast for wide adders.

This paper develops a taxonomy of parallel prefix networks based on stages, fanout, and wiring tracks. The area of a datapath layout is the product of the number of rows and columns in the network. The latency strongly depends on fanout and wiring capacitance, not just number of logic levels. Therefore, the latency is evaluated using the method of logical effort [12, 6]. The taxonomy suggests new families of networks with different tradeoffs. One of these networks has area comparable with the smallest

known network and latency comparable with the fastest known network.

Section II reviews the parallel prefix networks in the literature. Section III develops the taxonomy, which reveals a new family of prefix networks. Performance comparison appears in Section IV and Section V concludes the paper.

II. Parallel Prefix Networks

Parallel prefix networks are distinguished by the arrangement of prefix cells. Fig. 2 shows six such networks for N=16. The upper box performs the precomputation and the lower box performs the postcomputation. In the middle, black cells, gray cells, and white buffers comprise the prefix network. Black cells perform the full prefix operation, as given in EQ (3). In certain cases, only part of the intermediate variable is required. For example, in many adder cells, only the $G_{i:0}$ signal is required, and the $P_{i:0}$ signal may be discarded. Such gray cells have lower input capacitance. White buffers are used to reduce the loading of later non-critical stages on the critical path. The span of bits covered by each cell output appears near the output. The critical path is indicated with a heavy line.

The prefix graphs illustrate the tradeoffs in each network between number of logic levels, fanout, and horizontal wiring tracks. All three of these tradeoffs impact latency; Huang and Ercegovac [4] showed that networks with large number of wiring tracks increase the wiring capacitance because the tracks are packed on a tight pitch to achieve reasonable area.

Observe that the Brent-Kung and Han-Carlson never have more than one black or gray cell in each pair of bits on any given row. This suggests that the datapath layout may use half as many columns, saving area and wire length.

III. Taxonomy

Parallel prefix structures may be classified with a three-dimensional taxonomy (l,f,t) corresponding to the number of logic levels, fanout, and wiring tracks. For an N-bit parallel prefix structure with $L = \log_2 N$, l, f, and t are integers in the range [0, L-1] indicating:

- Logic Levels: $L + l$
- Fanout: 2^f+1
- Wiring Tracks: 2^t

This taxonomy is illustrated in Fig. 3 for N=16. The actual logic levels, fanout, and wiring tracks are annotated along each axis in parentheses. The parallel prefix networks from the previous section all fall on the plane $l + f + t = L$-1, suggesting an inherent tradeoff between logic levels, fanout, and wiring tracks. The Brent-Kung (L-1,0,0), Sklansky (0,L-1,0), and Kogge-Stone (0,0,L-1) networks occupy vertices. The Ladner-Fischer (L-2,1,0) network saves one

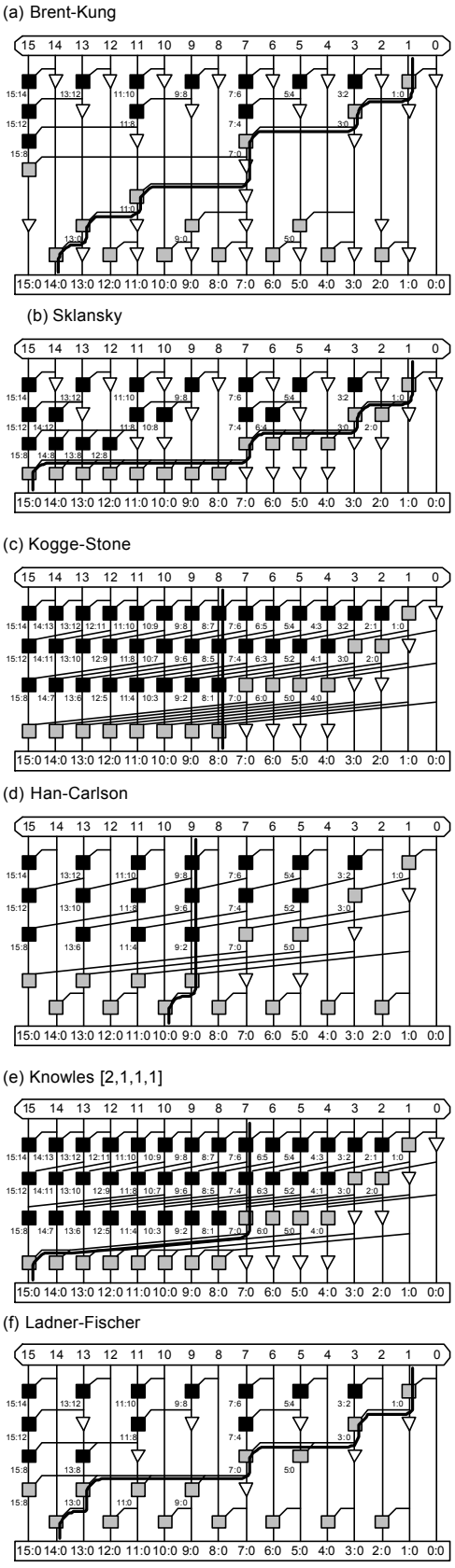

Fig. 2. Parallel prefix networks

level of logic at the expense of greater fanout. The Han-Carlson (1,0,*L*-2) network reduces the wiring tracks of the Kogge-Stone network by nearly a factor of two at the expense of an extra level of logic. In general, Han and Carlson describe a family of networks along the diagonal (*l*, 0, *t*) with $l + t$ = L-1. Similarly, the Knowles family of networks occupy the diagonal (0, *f*, *t*) with $f + t = L$-1 and Ladner-Fischer occupy the diagonal (*l*, *f*, 0) with $l + f = L$-1.

Knowles networks are described by *L* integers specifying the fanout at each stage. For example, the [8,4,2,1] and [1,1,1,1] network represents the Sklansky and Kogge-Stone extremes and [2,1,1,1] was shown in Fig. 2e. In general, a (0, *f*, *t*) network corresponds to the Knowles network [2^f, 2^{f-1}, …, 1, 1], which is the Knowles network closest to the diagonal.

The taxonomy suggests yet another family of parallel prefix networks found inside the cube with $l, f, t > 0$. Fig. 4 shows such a (1,1,1) network. For N=32, the new networks would include (1,1,2), (1,2,1), and (2,1,1).

IV. Results

Table 1 compares the parallel prefix networks under consideration. The delay depends on the number of logic levels, the fanout, and the wire capacitance. All cells are designed to have the same drive capability; this drive is arbitrary and generally greater than minimum. Networks with $l > 0$ are sparse and require half as many columns of cells. The wire capacitance depends on layout and process and can be expressed by *w*, the ratio of wire capacitance per column traversed to input capacitance of a unit inverter. Reasonable estimates from a trial layout in a 180 nm process are $w = 0.5$ for widely spaced tracks and $w = 1$ for networks with a large number of tightly spaced wiring tracks.

The method of logical effort is used to estimate the latency adders built with each prefix network, following the assumptions made in [6]. Tables 2-4 shows how the latency depends on adder size, circuit family, and wire capacitance.

V. Conclusion

This paper has presented a three-dimensional taxonomy of parallel prefix networks showing the tradeoffs between number of stages, fanout, and wiring tracks. The taxonomy captures the networks used in the parallel prefix adders described in the literature. It also suggests a new family of parallel prefix networks inside the cube. The new architecture appears to have competitive latency in many cases.

REFERENCES

1 A. Beaumont-Smith and C. Lim, "Parallel prefix adder design," *Proc. 15th IEEE Symp. Comp. Arith.*, pp. 218-225, June 2001.

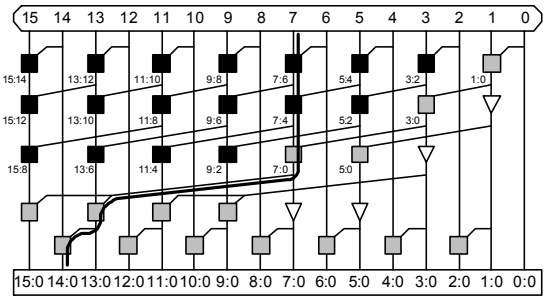

Fig. 4. New (1,1,1) parallel prefix network

2 R. Brent and H. Kung, "A regular layout for parallel adders," *IEEE Trans. Computers*, vol. C-31, no. 3, pp. 260-264, March 1982.

3 C. Huang, J. Wang, and Y. Huang, "Design of high-performance CMOS priority encoders and incrementer/decrementers using multilevel lookahead and multilevel folding techniques," *IEEE J. Solid-State Circuits*, vol. 37, no. 1, pp. 63-76, Jan. 2002.

4 Z. Huang and M. Ercegovac, "Effect of wire delay on the design of prefix adders in deep submicron technology," *Proc. 34th Asilomar Conf. Signals, Systems, and Computers*, vol. 2, pp. 1713-1717, 2000.

5 T. Han and D. Carlson, "Fast area-efficient VLSI adders," *Proc. 8th Symp. Comp. Arith.*, pp. 49-56, Sept. 1987.

6 D. Harris and I. Sutherland, "Logical effort analysis of carry propagate adders," *Proc. 37th Asilomar Conf. Signals, Systems, and Computers*, 2003.

7 S. Knowles, "A family of adders," *Proc. 15th IEEE Symp. Comp. Arith.*, pp. 277-281, June 2001.

8 P. Kogge and H. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence relations," *IEEE Trans. Computers*, vol. C- 22, no. 8, pp. 786-793, Aug. 1973.

9 J. Kowalczuk, S. Tudor, and D. Mlynek, "A new architecture for an automatic generation of fast pipeline adders," *Proc. European Solid-State Circuits Conf.*, pp. 101-104, 1991.

10 R. Ladner and M. Fischer, "Parallel prefix computation," *J. ACM*, vol. 27, no. 4, pp. 831-838, Oct. 1980.

11 J. Sklansky, "Conditional-sum addition logic," *IRE Trans. Electronic Computers*, vol. EC-9, pp. 226-231, June 1960.

12 I. Sutherland, R. Sproull, and D. Harris, *Logical Effort*, San Francisco: Morgan Kaufmann, 1999.

13 R. Zimmermann, *Binary Adder Architectures for Cell-Based VLSI and their Synthesis*, ETH Dissertation 12480, Swiss Federal Institute of Technology, 1997.

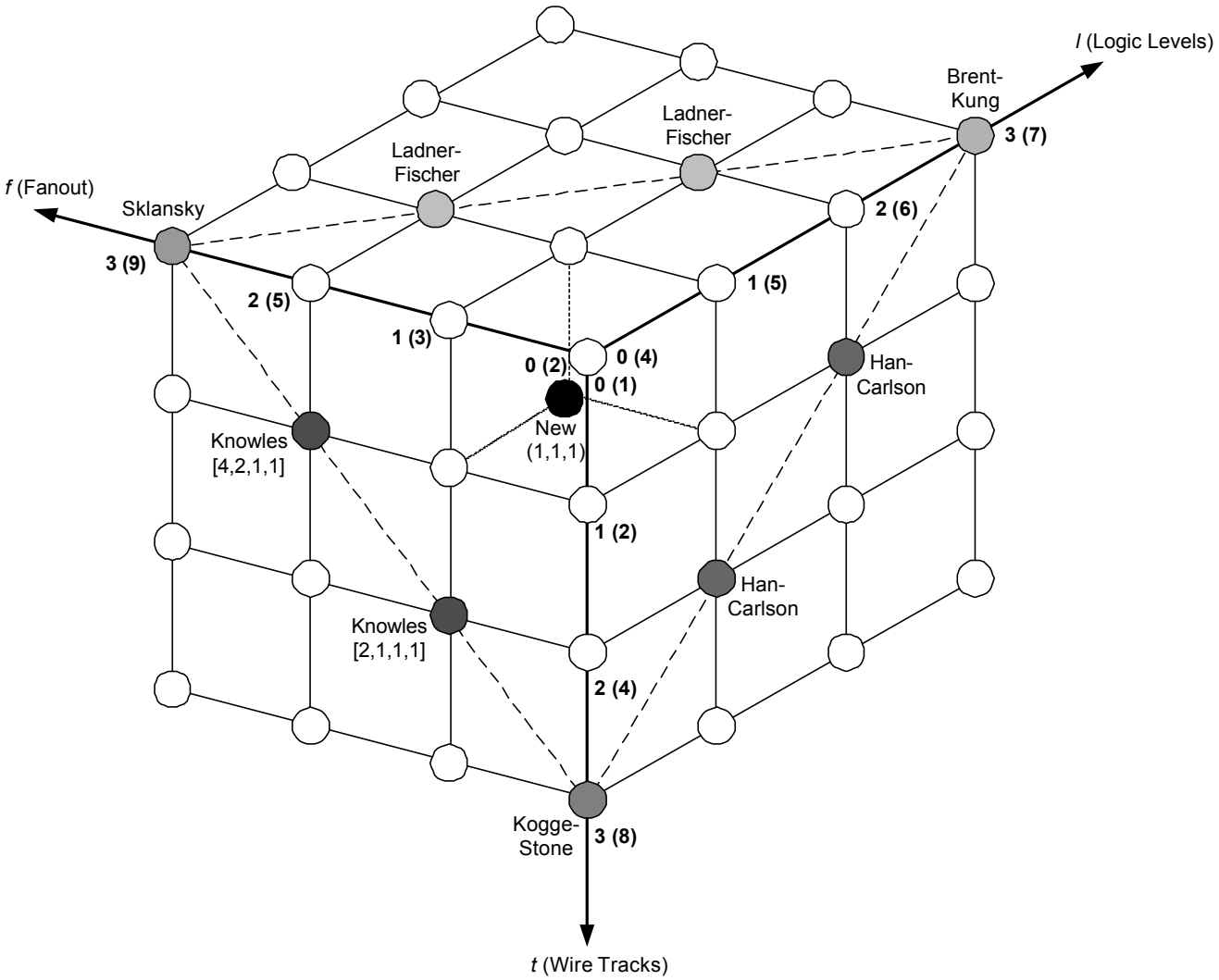

Fig. 3. Taxonomy of prefix graphs

Architecture	Classification	Logic Levels	Max Fanout	Tracks	Cols
Brent-Kung	(L-1, 0, 0)	$L + (L - 1)$	2	1	N/2
Sklansky	(0, L-1, 0)	L	N/2 + 1	1	N
Kogge-Stone	(0, 0, L-1)	L	2	N/2	N
Han-Carlson	(1, 0, L-2)	L + 1	2	N/4	N/2
Knowles [2,1,…,1]	(0, 1, L-2)	L	3	N/4	N
Ladner-Fischer	(1, L-2, 0)	L + 1	N/4 + 1	1	N/2
(1, 1, 1)	(1, 1, L-3)	L + 1	3	N/8	N/2

Table 1. Comparison of parallel prefix network architectures

	N = 16	N = 32	N = 64	N = 128
Brent-Kung	10.4 / 9.9	13.7 / 13.0	18.1 / 17.4	24.9 / 24.2
Sklansky	13.0 / 8.8	21.6 / 12.4	38.2 / 18.3	70.8 / 28.2
Kogge-Stone	9.4 / 7.4	12.4 / 10.0	17.0 / 14.1	24.8 / 21.5
Han-Carlson	9.9 / 7.7	12.1 / 9.4	15.1 / 12.0	19.7 / 16.1
Knowles [2,1,…,1]	9.7 / 7.9	12.7 / 10.3	17.3 / 14.5	25.1 / 21.8
Ladner-Fischer	10.6 / 8.4	15.2 / 10.8	23.8 / 14.5	40.4 / 20.3
(1, 1, 1)	10.7 / 8.1	12.9 / 9.8	15.9 / 12.4	20.5 / 16.5

Table 2. Adder delays: w=0.5; inverting static CMOS / footed domino

	Inverting Static CMOS	Noninverting Static CMOS	Footed Domino	Footless Domino
Brent-Kung	13.7 / 18.1	16.8 / 21.8	13.0 / 17.4	10.7 / 14.6
Sklansky	21.6 / 38.2	16.3 / 23.4	12.4 / 18.3	10.5 / 15.9
Kogge-Stone	12.4 / 17.0	13.4 / 18.0	10.0 / 14.1	8.7 / 12.7
Han-Carlson	12.1 / 15.1	13.3 / 16.4	9.4 / 12.0	7.9 / 10.3
Knowles [2,1,…,1]	12.7 / 17.3	13.6 / 18.3	10.3 / 14.5	8.9 / 12.9
Ladner-Fischer	15.2 / 23.8	14.5 / 19.1	10.8 / 14.5	8.9 / 12.1
(1, 1, 1)	12.9 / 15.9	13.8 / 16.9	9.8 / 12.4	8.3 / 10.6

Table 3. Adder delays: w=0.5; N = 32/64

	w = 0	w = 0.25	w = 0.5	w = 0.75	w = 1
Brent-Kung	11.4 / 13.4	12.5 / 15.7	13.7 / 18.1	14.8 / 20.4	15.9 / 22.7
Sklansky	18.5 / 31.9	20.1 / 35.0	21.6 / 38.2	23.1 / 41.4	24.7 / 44.5
Kogge-Stone	9.3 / 10.7	10.9 / 13.9	12.4 / 17.0	13.9 / 20.1	15.5 / 23.3
Han-Carlson	10.5 / 11.9	11.3 / 13.5	12.1 / 15.1	12.9 / 16.7	13.7 / 18.3
Knowles [2,1,…,1]	9.6 / 11.0	11.2 / 14.2	12.7 / 17.3	14.3 / 20.4	15.8 / 23.6
Ladner-Fischer	13.6 / 20.6	14.4 / 22.2	15.2 / 23.8	16.0 / 25.4	16.8 / 27.0
(1, 1, 1)	11.2 / 12.6	12.1 / 14.3	12.9 / 15.9	13.8 / 17.6	14.6 / 19.2

Table 4. Adder delays: inverting static CMOS; N = 32/64