Logical Effort of Higher Valency Adders

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

Abstract – Higher valency parallel prefix adders reduce the number of logic levels at the expense of greater fan-in at each level. This paper uses the method of logical effort to evaluate the tradeoffs of higher valency for static and dynamic implementations of various adder architectures.

I. Introduction

Higher valency parallel prefx adders are popular for high performance applications such as microprocessor ALUs [5, 1, 3, 4]. A valency-v N-bit adder requires $O(\log_v N)$ logic levels, so a 64 bit addition requires as few as three levels of valency 4 propagate-generate gates as opposed to six levels of valency 2. However, the higher valency gates have greater logical effort and parasitic delay, are more complex to design, and are not always available in standard cell libraries. Is higher valency addition really faster? Domino gates have lower logical efforts than their static counterparts and hence can use greater fan-ins. Does this mean higher valencies are better suited to domino than static logic? This paper uses the method of logical effort to try to answer these questions. According to the logical effort model, the delays of valency-2, 3, and 4 designs are all approximately the same for a given architecture, circuit family, and wire load model.

This paper closely follows the methodology of [2]. It first describes the static and domino gates used to compute generate and propagate signals for the various valencies and tabulates the estimated logical effort and parasitic delay of each gate. It then shows the prefix networks and the critical paths that were examined. Finally, it calculates the delays for valency 2, 3, and 4 for each architecture and circuit family using the method of logical effort.

II. Logical Effort of Circuit Building Blocks

The three basic building blocks for an adder are the bitwise Propagate/Generate (PG) cells, the group PG cells in the prefix network, and the sum XORs, as shown in Fig 1. High performance datapath adders often build these cells from domino gates while static CMOS is preferable when design simplicity and power consumption take precedence over utmost performance.

Fig 2 shows implementations of the bitwise PG cells and the sum XOR gates using static CMOS and domino gates. The static designs use propagate and generate (PG) while the domino add kill (K) for monotonic sum computation. The transistor widths are specified in arbitrary units to deliver unit drive. Noninverting static CMOS gates add an inverter after each inverting stage. Footed domino gates require an extra clocked evaluation transistor. The logical efforts (LE) and parasitic delays (PD) are given in Table 1.

Fig 1 Adder block diagram

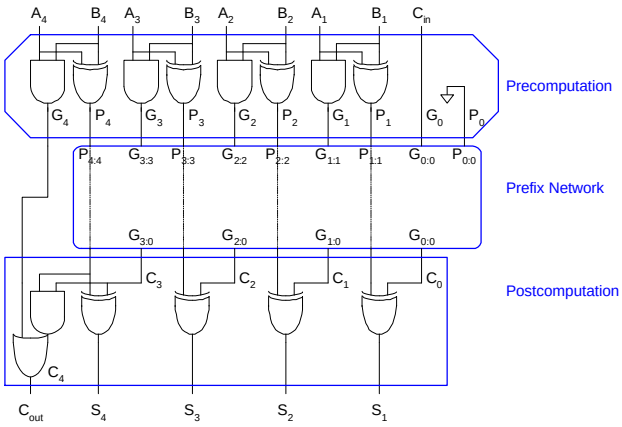


Fig 2 Bitwise PG and sm XOR gates

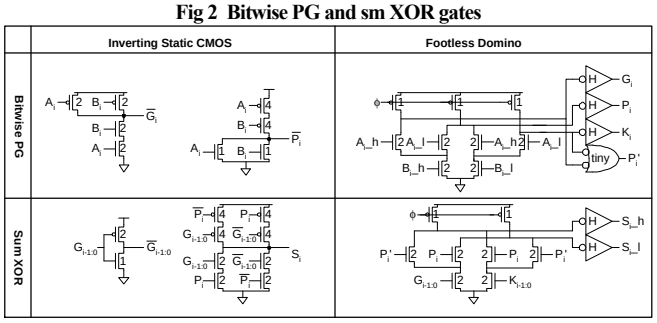


Table 1 Bitwise PG and sum XOR delay estimates

Cell	Term	Noninverting CMOS	Inverting CMOS	Footed Domino	Footless Domino
Bitwise	LE_{bit}	9/3	9/3	6/3 * 5/6	4/3 * 5/6
	PD_{bit}	6/3 + 1	6/3	7/3 + 5/6	5/3 + 5/6
Sum XOR	LE_{xor}	9/3	9/3	3/3 * 5/6	2/3 * 5/6
	PD_{xor}	9/3 + 12/3	9/3 + 12/3	7/3 + 5/6	5/3 + 5/6

Prefix networks consist of black cells, gray cells, and buffers. Black cells compute both propagate and generate signals. Gray cells compute only generate, and buffers reduce the loading presented by noncritical paths.

Fig 3 shows circuit implementations of propagate and generate gates for valency 2. Inverting static CMOS designs require alternating stages of the gates shown and their DeMorgan complements that accept inverted inputs and produce true outputs. [2] found that the difference in delay of the complementary stages is insignificant, so it will be ignored.

Fig 3 Valency 2 static and dynamic generate/propagate gates

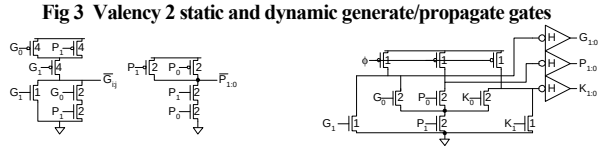


Fig 4 Valency 4static and dynamic generate/propagate gates

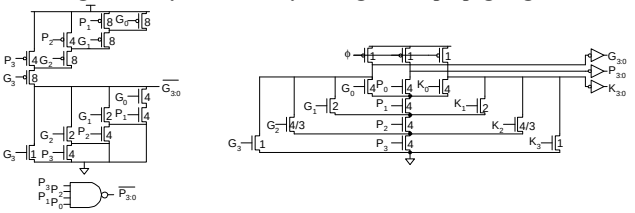


Table 2 Gray and black cell delay estimates

Valency	Term	Cell	Inverting CMOS	Noninverting CMOS	Footed Domino	Footless Domino
2	PDg		7/3	7/3 + 1	6/3 + 5/6	4/3 + 5/6
	PDp		2	2 + 1	3/3 + 5/6	2/3 + 5/6
	LEg1		5/3		1/2 * 5/6	1/3 * 5/3
	LEg0		6/3		3/3 * 5/6	2/3 * 5/6
	LEp1	Gray	6/3		3/3 * 5/6	2/3 * 5/6
	LEp1	Black	10/3		6/3 * 5/6	4/3 * 5/6
	LEp0	Black	4/3		3/3 * 5/6	2/3 * 5/6
3	PDg		13/3	13/3 + 1	10/3 + 5/6	7/3 + 5/6
	PDp		3	3 + 1	4/3 + 5/6	3/3 + 5/6
	LEg2		7/3		4/9 * 5/6	1/3 * 5/6
	LEg1		8/3		2/3 * 5/6	½ * 5/6
	LEg0		9/3		4/3 * 5/6	3/3 * 5/6
	LEp2	Gray	5/3		4/3 * 5/6	3/3 * 5/6
	LEp1	Gray	9/3		4/3 * 5/6	3/3 * 5/6
	LEp2	Black	10/3		8/3 * 5/6	6/3 * 5/6
	LEp1	Black	14/3		8/3 * 5/6	6/3 * 5/6
	LEp0	Black	5/3		4/3 * 5/6	3/3 * 5/6
4	PDg		17/3	17/3 + 1	13/3 + 5/6	10/3 + 5/6
	PDp		4	4 + 1	5/3 + 5/6	4/3 + 5/6
	LEg3		9/3		5/12 * 5/6	1/3 * 5/6
	LEg2		10/3		5/9 * 5/6	4/9 * 5/6
	LEg1		10/3		5/6 * 5/6	2/3 * 5/6
	LEg0		12/3		5/3 * 5/6	4/3 * 5/6
	LEp3	Gray	8/3		5/3 * 5/6	4/3 * 5/6
	LEp2	Gray	6/3		5/3 * 5/6	4/3 * 5/6
	LEp1	Gray	12/3		5/3 * 5/6	4/3 * 5/6
	LEp3	Black	14/3		10/3 * 5/6	8/3 * 5/6
	LEp2	Black	12/3		10/3 * 5/6	8/3 * 5/6
	LEp1	Black	18/3		10/3 * 5/6	8/3 * 5/6
	LEp0	Black	6/3		5/3 * 5/6	4/3 * 5/6

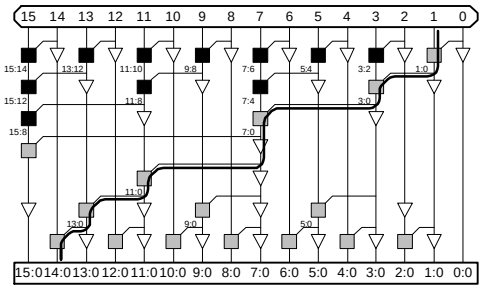
Similarly, Fig 4 shows the circuit designs for valency 4. Table 2 gives the logical efforts and parasitic delays for the various inputs to black and gray cells in each circuit family.

III. Adder Architectures

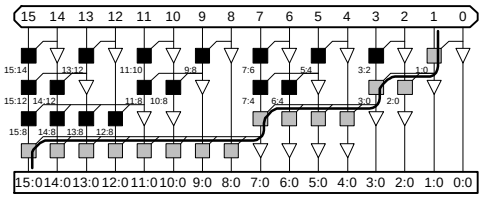
Adders are distinguished by the arrangement of cells in the group PG logic. Fig 5 shows typical parallel prefix architectures for valency 2 gates [6]. One of several paths may be most critical depending on the cell delays; the black highlighted lines indicate the path that was assumed to be critical in this study. Similarly, Fig 6 shows the analogous architectures for higher valency 3. Higher valency adders offer a number of hybrid tree / select architectures such as the spanning tree and sparse tree that reduce the number of cells in the parallel prefix network in exchange for adding short ripple networks; these variants are not considered in this study.

Fig 5 Valency 2 adder architectures

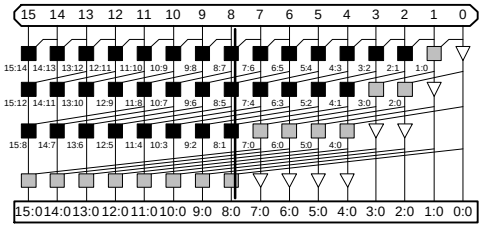
(c) Brent-Kung



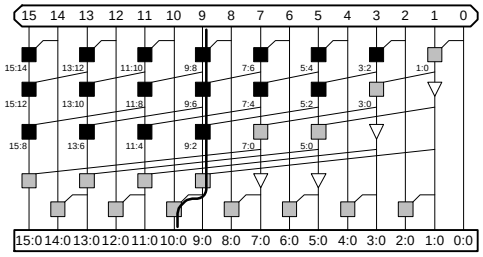
(d) Sklansky



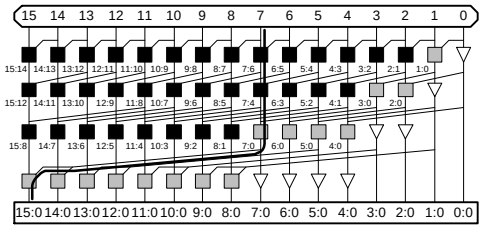
(e) Kogge-Stone



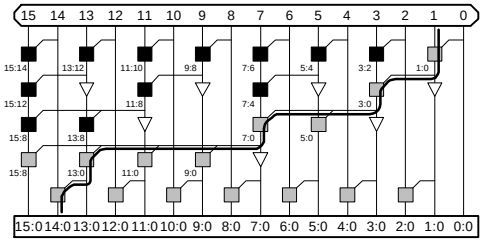
(f) Han-Carlson



(g) Knowles [2,1,1,1]



(h) Ladner-Fischer



IV. Logical Effort Delay Model

The method of Logical Effort provides a simple method for determining a lower bound on critical path delay in circuits with negligible wire capacitance. If the path has M stages, a path effort of F, and a parasitic delay of PD, the delay (in τ) achieved with best transistor sizes is

$$D = D_F + PD = MF^{1/M} + PD \quad \textbf{(1)}$$

where D is measured in units of τ, the delay of an ideal inverter with no parasitic capacitance driving an identical inverter. Delay is normalized to that of a fanout-of-4 inverter with the conversion 1 FO4 ˜ 5 τ.

In general, achieving least delay requires using different transistor sizes in each gate (although this delay model has assumed that all transistors in a branch scale uniformly). A regular layout with consistent transistor sizes in each type of cell is easier to build but may sacrifice performance. Consider designing all cells to have an arbitrary unit drive (i.e. output conductance). Define an inverter with unit drive to have unit input capacitance. For circuits with a single stage per cell (e.g. inverting static CMOS), the path effort delay is simply the sum of the effort delays of each stage:

$$D_F = \sum_{i=1}^{M} f_i \quad \textbf{(2)}$$

The total delay is still the sum of the path effort and parasitic delays.

In a circuit with two stages per cell (e.g. noninverting static CMOS or domino), let us design the first stage to have unit drive. Choose the size of the second stage for least delay. If the path has $C = M/2$ cells and the effort of the ith cell is F_i, the path effort delay is

$$D_F = \sum_{i=1}^{C} 2\sqrt{F_i} \quad \textbf{(3)}$$

[2] showed that the delay with uniform sizes is only slightly longer than the delay with arbitray sizes except on architectures like Sklansky that have unusually large fanouts on certain nodes. The uniform size designs are also easier to layout and permit closed-form results when wire capacitance is considered, so we focus on them in this paper.

Horizontal wires add capacitance to the load of each stage. Let the wire capacitance be w units per column spanned. w depends on the width of each column, the width and spacing between wires, and the size of a unit transistor; in a trial layout in a 180 nm process, w ˜ 0.5. While there is no closed-form solution for the minimum-delay problem with wire capacitance, the delay assuming fixed cell sizes is readily calculated by adding the wire capacitance to the stage effort f_i or F_i in EQ (2) or (3).

V. Results

The adder delays were evaluated using a MATLAB script. Fig 7 plots delay (in FO4 inverter delays) vs. number of bits for various adder architectures, and circuit families assuming w = 0.5. The three curves on each set of axes indicate valency 2, 3, and 4 delays.

The delay is nearly independent of the valency for both static and domino designs of most architectures. Brent-Kung architectures are an exception that benefit from higher valency for noninverting circuits because the stage effort is too low with valency 2, but Brent-Kung is not the fastest architecture in any case. Domino gates are consistently faster than static and footless domino is faster than footed. The designs with two gates per stage (all but inverting CMOS) are better at driving the heavy wire loads and hence perform better for wide adders.

VI. Conclusions

The logical effort model facilitates rapid comparison of a wide variety of adder architectures using multiple circuit families while accounting for the costs of fanout and interconnect.

Under the assumptions made in this paper, the delay is nearly independent of the valency for both static and domino designs of most architectures. Brent-Kung architectures are an exception that benefit from higher valency for noninverting circuits because the stage effort is too low with valency 2, but Brent-Kung is not the fastest architecture in any case. Valency 2 designs are the simplest to implement.

This paper has not considered the area, power, or wiring tradeoffs of higher valency adders. In practice, the logical efforts of gates are likely to be lower on account of velocity saturation, but the parasitic delays are likely to be higher when internal nodes are considered. Simulations of extracted layouts could answer these questions.

References

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

2 D. Harris and I. Sutherland, “Logical effort of carry propagate adders,” *Proc. 37th Asilomar Conf. Signals, Systems, and Computers*, pp. 673-678, 2003.

3 T. Lynch and E. Swartzlander, “A spanning tree carry lookahead adder,” *IEEE Trans. Computers*, vol. 41, no. 8, Aug. 1992, pp. 931-939.

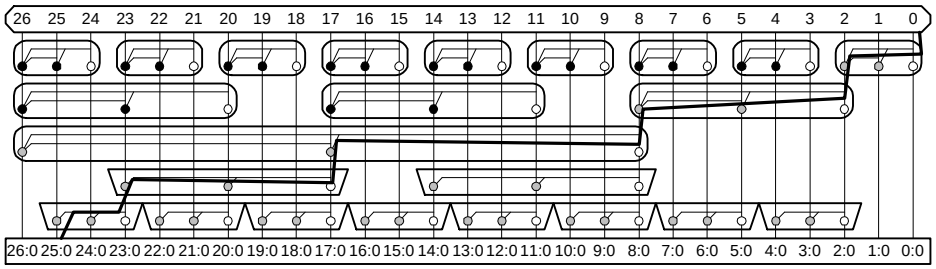
4 S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar, "A 4-GHz 130-nm address generation unit with 32-bit sparse-tree adder core," *J. Solid-State Circuit*, vol. 38, no. 5, May 2003, pp. 689-695.

5 S. Naffziger, “A subnanosecond 0.5 µm 64b adder design,” *Intl. Solid-state Circuits Conf.*, 1996, pp. 362-363.

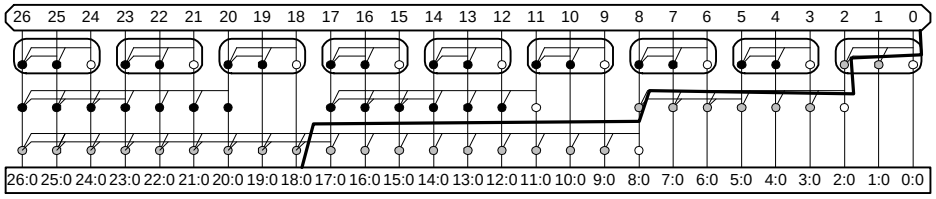
6 N. Weste and D. Harris, *CMOS VLSI Design*, Addison-Wesley, 2005.

Fig 6 Valency 3 adder architectures

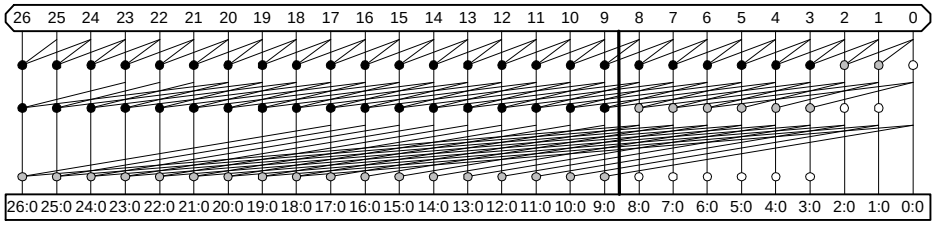
(a) Brent-Kung



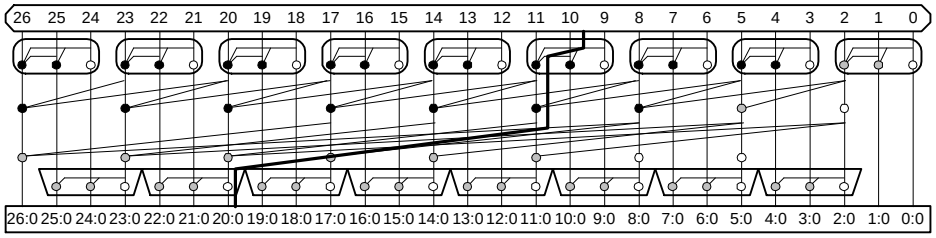
(b) Sklansky



(c) Kogge-Stone



(d) Han-Carlson



(e) Ladner-Fischer

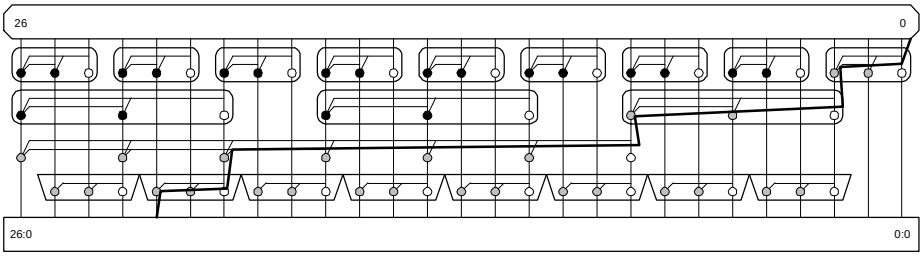


Fig 7 Adder delay vs. # of bits (logical effort model results)