Robust Energy-Efficient Adder Topologies

Dinesh Patil, Omid Azizi, Mark Horowitz
Stanford University
(ddpatil,oazizi,horowitz)@stanford.edu

Ron Ho
Sun Microsystems
ron.ho@sun.com

Rajesh Ananthraman
nVidia Inc.
rajesha@nvidia.com

Abstract

In this paper we explore the relationship between adder topology and energy efficiency. We compare the energy-delay tradeoff curves of selected 32-bit adder topologies, to determine how architectural features and design techniques affect energy efficiency. Optimizing different adders for the supply and threshold voltages, and transistor sizing, we show that topologies with the least number of logic stages having an average fanin of two per stage, and fewest wires are most energy efficient. While a design with fully custom sizes can be extremely tedious to layout, we show that custom sizing can be used as a guide to group different gates in the design, resulting in a manageable layout overhead without significant loss of energy efficiency.

1. Introduction

Adder structures are ubiquitous in modern digital systems and are often present in critical timing blocks. Consequently, designers have studied adder topologies extensively and have developed many techniques to improve addition algorithms. While initial papers [15, 9, 13, 8] focused mainly on performance, recent research [25, 16, 12] has focused on energy, now a critical issue in digital design. We extend this work by systematically exploring 32-bit custom designed adder topologies for their energy efficiency. We develop a relation between topological choices and energy efficiency by comparing the Pareto-optimal energy-delay (E-D) tradeoff curves of selected adder topologies in different logic styles, based on sizing, supply voltage and threshold voltage optimization. We use the digital circuit optimization tool developed at Stanford University [17] to obtain these curves under different design constraints. Using this tool, we also explore how topology affects the statistical behavior of adders under random process variations.

In order to distinguish different adder topologies for energy-efficiency, we start with describing the topological parameters that affect energy and delay. Next we describe the optimization framework with gate delay and energy models, and discuss the design constraints that we include in the optimization. The results and insights obtained using this tool are described in Section 4, where we compare the Pareto-optimal E-D tradeoff curves of various adders for a common set of design constraints. In addition, we discuss the relation between topology and statistical robustness.

2. Adder topologies

Adder topologies can be separated into their Sum Generation Logic (SGL) and Carry Propagation Logic (CPL) [23]. As CPL dominates the adder delay and energy, we use the following four parameters of the CPL based on Harris's work [6] to describe adder topologies.

1. Radix (R): In tree adders, we define R as the average number of bits combined at each logic stage[1] of the CPL. In linear carry-skip or carry-select adders, R refers to the average number of bits combined per stage to generate a block Propagate-Generate (PG) term.

2. Logic depth (L): L indicates the total number of stages in the CPL, and is at least $\mathbf{log_R N}$ for an N-bit adder. Kogge-Stone and Sklansky adders are examples of minimum logic depth adders [9]. Note that the number of logic stages in the adder can be more than L.

3. Fanout (F): F represents the maximum logical branching seen by any stage in the CPL. For example, Kogge-Stone and linear ripple carry adders have $F = 2$, while a 32-bit Sklansky adder has $F = 17$.

4. Wiring tracks (T): T measures the maximum number of wires running across the bit pitch between any successive levels of the CPL. Brent-Kung, Sklansky and linear Manchester carry-chain adders have the smallest T of 1.

[1] In this paper, every gate, including an inverter, is a logic stage

For carry generation, R, L, F, and T are interdependent [6]. Figure 1 shows the CPL dot diagrams of selected 16-bit adder topologies with different R, T, L and F numbers. For a given radix, Kogge-Stone (KS), Brent-Kung (BK) and Sklansky[2] [21] maximize one of the three parameters – T, L, and F respectively while minimizing other two. The Knowles, Ladner-Fischer (LF) and Han-Carlson (HC) topologies [9] tradeoff two of these three parameters keeping the third fixed [6]. Simple linear adders can also be described using R, T, L and F. For example, the 32-stage ripple carry adder can be said to have $(R, T, L, F) = (1, 1, 32, 2)$ and a 32-bit multiple output Manchester carry chain adder can be described by $(32, 1, 1, 2)$. A carry skip and/or sum-select scheme in linear adders can be similarly described based on how they modify the topology.

We consider the following frequently occurring design scenarios.

1. **External buffering:** Inverters can be added at all the inputs or all the outputs of the adder to best match the load it is driving, without changing its R, T, and L numbers, though with a possible inversion of the resulting sum. Hence for a given adder topology and design constraints, we optimize with all possible external buffering and choose the best of the tradeoff curves. This allows for a fair comparison between adders with different L.

2. **Internal buffering and restructuring:** Logic functions can be evaluated using a single complex high fanin (higher valency) gate or a series of smaller low fanin (lower valency) gates. The radix R is smaller in the latter case[3]. In our definition, R depends on the number of stages needed to do a particular logic operation on a given number of input bits. Thus, inverters appended to a complex gate in the middle of the critical path (as opposed to external buffering) reduces the overall radix. For example, a domino gate that consumes four bits and generates a 4-bit PG term has a radix of 2, because the dynamic gate and the inverter are two separate stages of computation. A radix 4 domino design would consist of alternating 4 stack dynamic and 4 input static stages. Even for the same radix, however, one can have two different designs. For example, a 4 input gate followed by

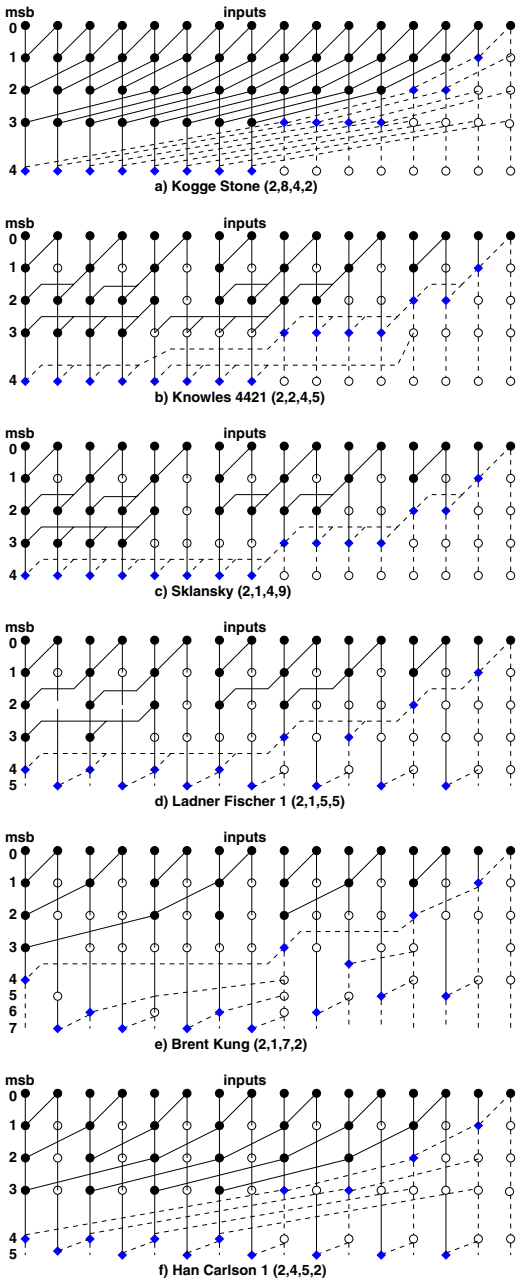


Figure 1. CPL dot diagram of selected adders with their (R,T,L,F). Solid lines and circles are PG signals and PG combine cells, dashed lines and diamonds are carry signals and carry generate cells, and empty circles represent wires or buffers.

[2]The Sklansky CPL design belongs to the Ladner-Fischer [10] family of adders. However Sklanky's adder predates Ladner-Fischer adders and can be easily reduced to Ladner-Fischer design with removal of conditional-sum logic. Hence in this paper, we will use refer the Ladner-Fischer adder with highest F as Sklansky design.

[3]Although radix refers to the logic function and "valency" refers to the implementation, from a circuit point of view a higher radix CPL implemented with lower valency gates behaves similar to a lower radix CPL. Hence, for extracting the E-D tradeoff, we consider valency and radix as equivalent.

an inverter and a two level tree of 2 input gates are both radix 2, but use different implementations. In these cases, we pick the best of the two designs for comparison.

3. **Sum selection:** The SGL usually consists simply of XOR logic. However, in sum select adders, the SGL generates a pair of speculative sums, to be correctly chosen when the respective carry arrives. Because only every k^{th} carry needs to be generated and fanned out to k muxes, a k-bit sum selection scheme re-distributes the logical fanout of the CPL by increasing the fanout of the final carry to k. This can potentially change the F of the CPL, independent of its R,T and L, creating multiple adders with the same R,T and L numbers. Sum selection is a very common technique used for high performance adders today [15, 25]. Because the CPL is a more critical part of the adder, we first find the most energy-efficient CPL structure and then explore the related sum selection techniques. The CPL and SGL, being in parallel, are almost independent for optimization purposes. This sequential procedure should therefore give optimal results.

4. **Ling adders [11]:** These use a reformulation of the Propagate-Generate (PG) equations of tree adders. Because Ling's equations are also associative and fall into a tree structure, they can be described using R, T and L. In this work, we look at the best PG adder structures and then compare them with similarly constructed Ling adders. Sum selection schemes are separately explored in both cases.

3. Optimization framework

In our analysis we have focused on three principal design metrics: energy, delay, and output load (Cout). We optimize adders for sizing, supply ($V_{\rm dd}$) and threshold voltages ($V_{\rm thN}$ and $V_{\rm thP}$). The optimization tool uses generalized posynomial delay and energy models [2] for gates and formulates the circuit design problem as a large geometric program [1, 5], which is solved efficiently using convex optimization solvers like MOSEK [14]. For a given Cout, we generate E-D tradeoff by optimizing the delay at different total energy constraints.

3.1 Energy and Delay Models

We model the delay of a gate as

$$\tau_d = \frac{C_{\rm load} V_{\rm dd}}{2 I_{\rm d}} + \kappa \tau_{\rm in}$$

where $\tau_{\rm in}$ is the input slope, κ is a fitting constant determined by $V_{\rm dd}$ and $V_{\rm th}$, and $C_{\rm load}$ is the total capacitance driven by the gate, including self loading and wire capacitance. The saturation current $I_{\rm d}$ is given by the velocity saturated current model [3]. The delay of a stack of transistors is formulated by modeling it as an equivalent transistor with accounting for intermediate capacitances. Details of the model can be found in [19, 18]. Using this delay model, we can solve the delay optimization geometric program,

$$\begin{array}{ll} \text{minimize} & T_{\rm cycle} \\ \text{subject to} & \max(T_1, \ldots, T_N) \le T_{\rm cycle}, \\ & E_{\rm dyn} + E_{\rm stat} \le E^{\max}, \\ & f_j(W, V_{\rm thN}, V_{\rm thP}, V_{\rm dd}) \le 1, \\ & \qquad\qquad j = 1, \ldots, m, \\ & d_g(W, V_{\rm thN}, V_{\rm thP}, V_{\rm dd}, C_{\rm load}) \le 1, \\ & \qquad\qquad g = 1, \ldots, n. \end{array}$$

Here, W is the vector of all the gate sizing variables, $E_{\rm dyn}$ and $E_{\rm stat}$ are the dynamic and leakage energies, T_i is the signal arrival time of the i^{th} output out of N outputs and $T_{\rm cycle}$ is the overall circuit delay, while $f_j(W, V_{\rm thN}, V_{\rm thP}, V_{\rm dd})$ represent constraints like device width bounds, signal slopes, input capacitances constraints, sizing ratios and so on. The adder netlist is treated as a directed acyclic graph, wherein the constraints for the signal propagation delay, $d_g(W, V_{\rm thN}, V_{\rm thP}, V_{\rm dd}, C_{\rm load})$ are formulated by writing the delay of each gate as the maximum of the delay of all the paths converging at its output from its inputs [4, 18]. The dynamic energy dissipation is modeled as

$$E_{\rm dyn} = \sum_{i \in edges} \alpha_i C_i V_{\rm dd}^2,$$

where α_i is the activity factor of edge i obtained by switch level simulations using the input activity factor of 0.25, and C_i is the capacitance switched on the i^{th} edge, including the driver's parasitics and wire capacitance. The input activity factor was derived from simulating selected SPECINT benchmark traces in a mirco-architecture simulator and observing the activity at ALU inputs. Crow-bar switching energy can be incorporated in this formula by slightly increasing the activity factor. The average leakage energy dissipation is given by

$$E_{\rm stat} = T_{\rm cycle} V_{\rm dd} \sum_{i \in gates} (\delta_i I_{\rm leakN} + (1 - \delta_i) I_{\rm leakP})$$

where δ_i is the duty factor of the output of gate i and $I_{\rm leak}$ is the leakage current, which is a function of the $V_{\rm th}$ and W. The duty factor δ_i is calculated during the activity factor extraction.

The modeling of parasitics is necessary for finding the delay of large gates. We account for the intermediate capacitances in the delay of a stack of transistors, but we do not account for the fact that if there are many stacks in parallel (like for example in a 4-bit Generate gate), there could be parasitic capacitance from neighboring partially turned on stacks. Thus we underestimate the parasitic delay of large fanin gates.

3.2. Design constraints

Most of the critical wires in an adder run across bits and thus their lengths are set by the bit pitch. Wires along the bit pitch generally have lengths set by the number of transistors in that bit pitch, hence we capture these using fixed capacitances. This assumption is invalid in cases of high energy, when gate sizes become very large. However, in this region, transistor capacitances dominate the wire capacitances anyways, so the resulting sizing errors should be small.

We have constrained the input capacitance (Cin) at any input to be less than 25fF, or roughly 15μm of transistor width. This is a reasonable load within the driving capability of library flip-flops in a 90nm CMOS technology. Except at the high energy points, this constraint does not come into play. For small loads, if the Cin constraint is active, the adder has already entered the region of diminishing performance returns for added energy. For large loads, external buffering at the output of the adder is always more efficient than increasing the sizes of the gates in the adder. Adding inverters at the output generally reduces the required input capacitance to fall within the specified Cin constraint. To check the effect of a Cin constraint we optimize a few adders without the input constraint and show that it makes little difference.

Reasonable signal slopes are maintained at every net, by limiting the delay of every logic stage. The minimum transistor width is constrained to $0.25\mu m$. $V_{\rm dd}$ ranges from 0.5V to 1.3V, while $V_{\rm th}$s range continuously from about 0.2V to 0.4V. Both are common for all gates in the netlist. All dynamic gates have footers, keepers and intermediate precharge transistors, which are used for stacks of three or more transistors. The sizes of all these three devices are ratioed to their respective NMOS pull-down transistors in order to track their sizes under optimization. The final XOR gate for sum generation (or sum select mux) is static in all cases except for dual-rail domino circuits.

Because different topologies have different logic depth, fanout, and internal loading and hence can be optimal under different load conditions, we generate the pareto-optimal E–D tradeoff curves for various values of Cout.

4. Results and analysis

The delay of an N-bit adder primarily depends on how fast the carry reaches each bit position. Parallel prefix logic networks [9], which use tree structures to compute the carry, are very efficient for large N [22]. Hence, for a systematic traversal of adders, we start with the triangular region based on L, T and F (Section 2) of radix 2 parallel prefix adders built in static CMOS logic. We will show in Section 4.3 that this is the optimal radix.

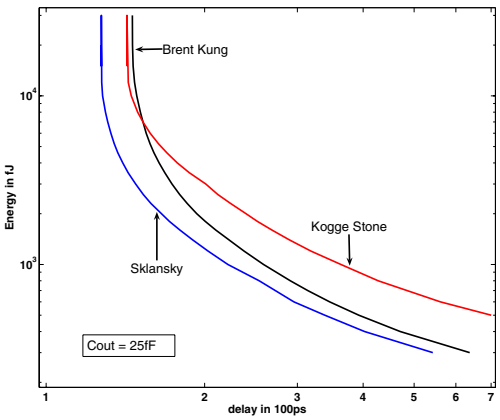
We first consider the corner adders: Kogge-Stone, Brent-Kung and Sklansky, designed in 90nm static CMOS logic. Figure 2 shows the pareto-optimal E–D tradeoffs (with external buffering, if necessary) of these adders for Cout = 25fF and Cout = 100fF. Figure 2(b) also shows the E–D curve for a Sklansky adder with no input capacitance constraint. Clearly this constraint is active only in high energy regions.

Figure 3 shows how the supply and threshold voltages change across the E–D curve for a Sklanksy adder. When $V_{\rm dd}$ reaches its upper bound, the threshold voltages continue to decrease. At the lower bound of $V_{\rm dd}$, the design becomes infeasible due to signal slope constraints, even though $V_{\rm th}$s have not hit their upper bounds. If the input activity factor is increased (decreased), the supply and threshold voltages both decrease (increase), increasing (decreasing) the leakage power in relation to the increase (decrease) in active power.

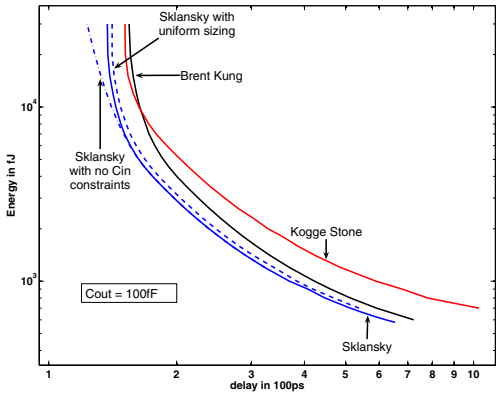
The E–D curves show diminishing returns as we go towards either higher performance or lower energy. At higher energies, as device sizes continually increase, the effect of wires and Cout decreases. After a certain point, the gates would largely be driving their own parasitic capacitance and further improvement in delay would not be possible. With Cin constrained, after a certain energy, the design becomes identical to one based solely on logical effort [22], for which the marginal cost of energy for improvement in delay is infinite. Similarly, as energy is lowered, the supply and threshold voltage both change (see Figure 3) until the design enters a region where, analogous to the minimum delay solution, the marginal cost in delay for lowering the energy is very high.

The three radix 2 adders in Figure 2 each minimize two parameters out of L, T and F, at the cost of the third. Our experiments indicate that the Sklansky adder, which has the highest F, but smallest L and T, is the most energy efficient. Clearly, these three parameters do not trade equally with each other.

A large logical fanout at a particular stage does not necessarily imply a that that stage will be slow. What



(a) Cout = 25fF



(b) Cout = 100fF, with a Sklansky adder E–D curve for uniform sizing and one for no Cin constraint.

Figure 2. E–D curves for the three radix 2 corner adders.

matters to the delay is the *electrical* fanout (due to capacitive loading). In a Sklansky adder, at every n^{th} stage of the CPL tree, the carry drives $2^n + 1$ gates. After optimal sizing, we find that of these $2^n + 1$ PG combine gates at the $(n+1)^{th}$ stage, the one gate that drives $2^{n+1} + 1$ PG combine cells at the next stage (or the largest load in general) is sized much larger than the others, resulting in an overall electrical fanout closer to 2. This optimization of electrical-vs-logical fanout arises due to the possibility of differential sizing of the gates at the same stage. With its highest F of 16, Sklansky adder can take maximum advantage of differential sizing. Unlike F, L has a real cost. A Brent-Kung adder has the same T, but almost twice the L as the Sklansky adder. This may seem useful at for a large Cout, but inverters are far more efficient than P/G gates and can

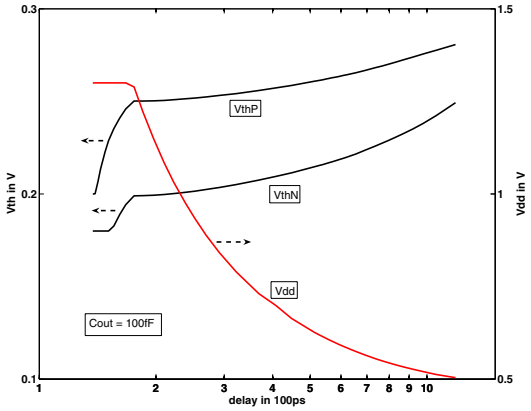


Figure 3. Change in $V_{\rm dd}$ and $V_{\rm th}$s across the E–D space for a Sklansky adder.

always be padded to a lower logic depth design to make up for the required gain at lower energy cost.

Like L, T also has real costs. A larger T means a higher portion of the total energy consumed in wires. With its smallest T of 1, the Sklansky design spends most of the energy budget in driving useful logic, with the least amount wasted in wires. A Kogge-Stone adder, having the largest T, suffers from high energy loss to result in poor energy efficiency. Figure 4 shows the percentage of total adder energy consumed in wires for the three adders. Note that wire energy changes with $V_{\rm dd}$, which changes with the optimal E–D point.

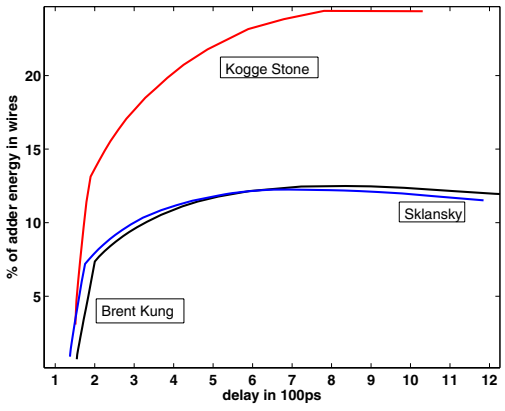


Figure 4. Energy consumed in wires.

At lower energies, resources are constrained and adders with fewer gates have an advantage. The Brent Kung adder is most economical in gate count. Hence

it comes closer to Sklansky at lower energies, unlike Kogge-Stone, which has a comparable number of gates to Sklansky, but more wires.

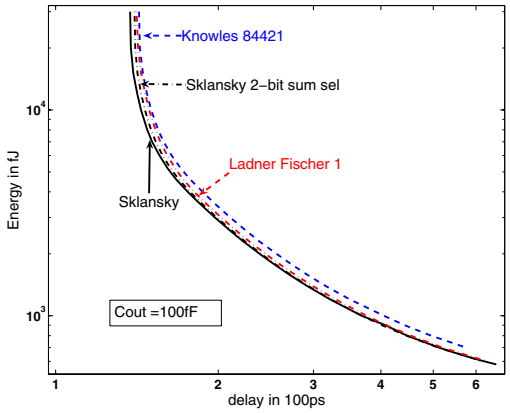


Figure 5. Comparison of Sklansky adder E–D curves to its closest neighbors and to a 2-bit sum select scheme.

To confirm our inferences about the Sklansky design we optimized its two closest radix 2 adders – a LF adder with an extra logic level ($(R, T, L) = (2, 1, 6)$) and a Knowles adder with an extra wire track ($(R, T, L) = (2, 2, 5)$). We also optimized a 2-bit sum select Sklansky adder to check if reduced fanout (F=8) in Sklansky CPL at the cost of increased fanout for the final carry gives any benefit. We found that the cost of generating the conditional sum in SGL was more than the advantage of having CPL and SGL in parallel. The results shown in Figure 5 confirm that Sklansky was better than its three closest relatives. Due to lower gate count, the selected Ladner-Fischer adder tends to compete with Sklansky adder at lower energies.

4.1. Layout efficiency

While adders in ASIC flow can have each cell sized individually, a custom designed adder, with all gates sized differently, can be extremely tedious to layout. For ease of layout, gates at the same stage are usually sized the same. This constraint severely penalizes the high fanout stages, as it incorrectly allocates as many resources as the critical path to the side paths. However, a close observation of the gate sizes in the fully custom designed Sklansky adder shows that of all the fanout gates at each stage, only one gate needs to be large and the rest can be sized uniformly smaller. Other gates, like the initial bit PG generate gates or the final XOR gates or external buffers, can be divided in uniformly sized blocks of 8 or 16. This tremendously reduces the layout effort, bringing it to within twice the layout effort of an adder with completely uniform stage sizing. Figure 6 shows this *semi-uniform* sizing for a

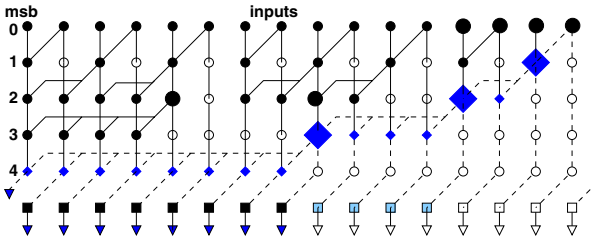


Figure 6. Semi-uniform sizing for a 16-bit Sklansky adder. Sizes of different shapes indicate the relative cell sizes within that level. ■ are XORs and ▼ are inverters.

16-bit Sklansky adder and Figure 2(b) shows that the 32-bit *semi-uniformly* sized Sklansky adder is still much better than the fully custom sized Kogge Stone or Brent Kung adder.

4.2. Other logic styles and topologies

Given the effect of loading and parasitics is similar in all logic styles, the topology that is the most energy-efficient in one, will be the most energy-efficient in the other as well. Following are the results of our experiments on other logic styles designed on the basis of the results from static CMOS logic.

1. **Domino and dual rail designs.** We also made Sklansky designs in radix 2^4 domino and dual rail domino logic. Similar to the static case, a fully dynamic 2-bit sum select scheme does not give any benefit. However, a 2-bit sum select using static SGL improves the energy-efficiency due to lower activity factor in the SGL. On the other hand, dual-rail designs consume almost twice the energy with the only benefit that the XORs in the SGL are faster. Hence they are better than the domino designs only at high energy by a small amount. Figure 7 shows the E–D curves of selected Sklansky domino/dual-rail topologies.

   Ling adder performs about 5% better in delay than the PG adder if only 4 transistor stacks are allowed, but it is worse than an equivalent PG adder (which

[4] Because the fanin of the domino gate (dynamic gate and inverter) is 4, some researchers [25] call this radix 4.

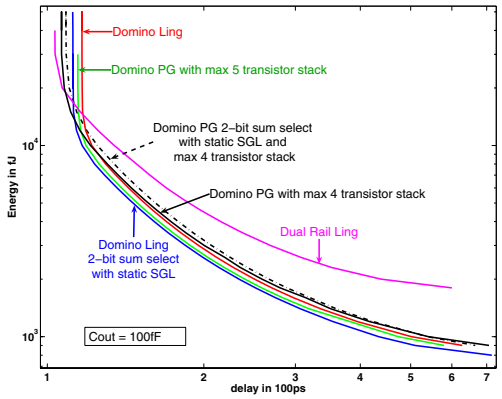


Figure 7. Comparison of Sklansky domino and dual-rail domino tree adders.

will have a 5 transistor stack in the first dynamic gate). However, as mentioned before, our modeling of the parasitic delay is most optimistic in the 5 stack gate, so we expect the two designs to be comparable in practice. Due to large fanin right at the inputs, the Cin constraint becomes active in these adders pretty early on, which is why the 4 stack PG adder looks better at higher energy.

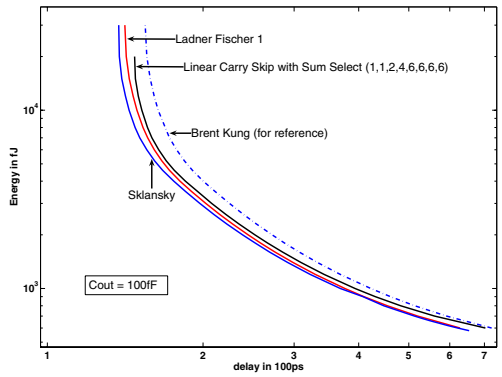


Figure 8. Comparison of the linear adder to closest tree adders.

2. **Linear adders.** While a linear ripple carry adder is extremely inefficient due to a large logic depth, a full 32-bit Manchester carry chain adder suffers from excessive parasitic delay. Carry skip and sum selection techniques exploit parallelism by overlapping the PG generation of a block of bits with the ripple carry inside the block. If, using similar gates, linear adders can be designed to have similar number of stages and wire tracks as the best tree adders, they should be equally energy-efficient. We designed such a linear carry-skip sum-select adder, with block sizes of 1,1,2,4,6,6,6 and 6, resulting in 9 logic stages, similar to LF1 adder. Figure 8 clearly shows that not surprisingly, with a T equal to that of a Knowles 84421 adder, this linear adder compares well with the tree designs.

   While their L is comparable for $N = 32$ bits, the logic depth of linear adders increases faster than that of a tree structure, where it is logarithmic with N. For example, for $N = 64$, L for Sklansky adder increases by unity, whereas for the best linear adder, it increases by 3. Hence linear adders are less efficient for higher N.

4.3. Optimal radix

From our definition of the radix, a logical operation performed on N inputs in p stages has a radix of $log_p N$ regardless of the kind of gates used. For delay, p is optimal ($p_{\rm opt}$) when each stage has a delay of about a FO4[5] [22]. The value of $p_{\rm opt}$ depends on the load and the kind of gates used in the design. For all reasonable adder loads (where Cout is at least about Cin), the number of stages in a minimum logic depth radix 2 adder is about equal to or less than their respective $p_{\rm opt}$. External buffering makes up for the lack of sufficient number of stages. Also, $p_{\rm opt}$s for these radices for $N = 32$ and above are about the same [7]. Therefore the choice of optimal radix really comes down to the parasitic delay of gates involved. Because parasitic delay grows at least as the square of the number of inputs [24], the parasitic delay of higher fanin gates dominates any gain from the reduced L, given that inverters are padded to reach p_{opt} stages. Complex high fanin gates like the ones that produce the group generate signals grow in the number of parallel transistor stacks as well, significantly increasing their parasitic delay.

From the energy point of view, using higher fanin gates does not save on switching activity. Adders being multiple output structures, the intermediate signals generated by using trees of lower fanin gates are typically used for other computation. Also, while the propagate function benefits from having a large fanin gate, as it is an AND function and activity factor reduces with more inputs, switching simulations show that the generate operation maintains the switching factor. Hence

[5] A fanout-of-four (FO4) delay is the delay of a single inverter driving 4 copies of itself.

overall, higher fanin PG cells have a marginal reduction of switching activity over trees of lower fanin cells. For domino designs however, because NMOS is faster than PMOS, it is better to have series NMOS stacks in the dynamic gate than having PMOS stacks in the following static stage. Hence the gates with four input dynamic gates followed by an inverter are more efficient. Note that the radix is 2 in both cases.

To validate this intuition we designed static CMOS 32-bit Sklansky adder, one with radix 3 and another with radix $\sqrt{6}$ (using alternate 3 bit and 2 bit combine stages). To avoid the irregularity of a 32-bit radix 3 adder, we also designed 27-bit adders with radix 2 and 3. Figure 9 compares the E–D tradeoff and confirms our understanding. The results remain unchanged for Cout = 25f.

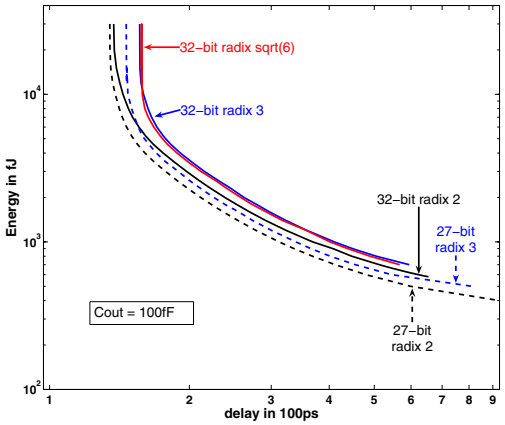


Figure 9. E–D tradeoff curves of 32-bit and 27-bit Sklansky adders of different radices.

4.4. Effect of buffers on energy-efficiency

External buffers increases the number of stages in the adder. Hence one might expect that while they are useful for adders with small logic depth driving large loads, they would be inefficient for an adder with L that is already at or bigger than the optimal L. However, inverters are the most efficient drivers. Hence, in addition to buffering, padding inverters at the output leads to a reduction in the size of the complex gates that precede them, thus saving energy. The reduction of gate sizes on the side paths helps to reduce the load on the critical path and more than compensates for the increased delay due to the extra logic stage. Figure 10 shows that while buffering is inefficient in the high energy region due to the delay added by the extra inverter stage(s), as the energy budget is reduced, the same design padded with a single inverter stage does much better than the original. The potential increase in delay by adding an extra logic stage is more than compensated by the energy benefit from smaller SGL gates, even in the Brent Kung adder, which already has a large L. In fact, all

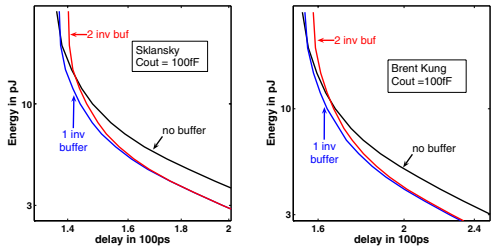


Figure 10. Effect of external buffering on radix 2 Sklansky and Brent Kung adders.

the E–D curves shown in this paper are with a single inverter padding, except for the dual rail ling adder design, which was more efficient without external buffering. This is expected, because there the outputs are already driven by the inverter in the dual rail domino gate. Selective padding of different paths can possibly bring more gains, but would affect the logical functionality of the design.

4.5. Adder design space

The E–D tradeoff curves of the different adders across different logic styles intersect at points at where one adder achieves better energy efficiency than the other. Static adders are good at low performance regions because, with lower switching activity factor and no clock load, they consume lower energy; but they saturate quickly as higher performance is desired, due to inherently large logical effort. With lower logical effort gates, dynamic adders have potentially higher performance and at higher energy they perform much better than their static counter parts. The overall Pareto-optimal curve is the lower bounding curve of all these curves. It gives an indication of the E–D space of optimally designed adders. Figure 11 shows the complete 32-bit adder E–D space. Incidently, at or before the point where Cin constraint becomes active, the next logic family takes over the pareto-optimal curve. We can also observe that sensitivity to R,T, and L to energy-efficiency depends on the location on E–D curve. At

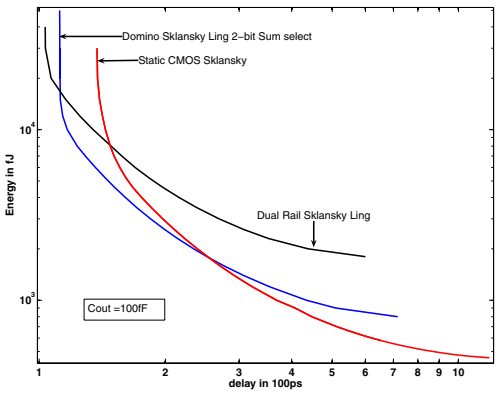


Figure 11. 32-bit adder E–D space.

low energy side, wire tracks T have a large weight because they consume a greater portion of the energy budget. At low delay points logic depth L matters more because every added stage adds a minimum delay, which causes external buffering to be inefficient at high energies and enables the Kogge-Stone E–D curve to cross the Brent-Kung one. Change in gate to drain/overlap capacitance ratio, the transistor drive capability (β) or the extent of velocity saturation in a given technology affects which radix is more optimal.

4.6. Effect of random process variations

Random process variations reduce the energy efficiency in two ways. Firstly, they push the circuit delay outward, because $T_{\rm cycle}$ is the *maximum* of the delays of all paths, and secondly, due to exponential dependence of leakage current on threshold voltage, they cause an increase in the average leakage energy. The tail of the delay Probability Distribution Function (PDF) gets worse as the number of critical paths increases, because the $\mathbf{max}()$ is taken over larger set of random variables. The standard deviation to mean (σ/μ) ratio improves as the length of the critical path increases, due to more averaging.

The tool is capable of doing statistical optimization by adding delay margins to the gate [19] and using the statistical estimate of energy [18]. The results are verified by doing Statistical Static Timing Analysis(SSTA) using monte carlo simulations. We assume independent gaussian delay distributions for every gate. As a very conservative estimate, we choose $\sigma(V_{\rm th})$ to be constant at 10% of the medium $V_{\rm th}$ device and the $\sigma(\beta)/\beta$ to be 5%, for a 1μ device. The overall $\sigma(I_{\rm d})$ changes with device size as per Pelgrom's model [20].

If only deterministic delay models are used, the optimizer tends to make all paths in the adder critical. While Kogge Stone adder has many paths of similar length, Brent Kung has a single long path with many small side paths, that the optimizer downsizes to make them critical. These small paths suffer from large variability. Consequently, SSTA performed on nominally designed Brent Kung, Kogge Stone and Sklansky adder shows that Brent Kung adder has the largest and Kogge Stone the smallest spread in their delay PDF. However, when statistical optimization is performed (with the same constraints as before), SSTA results show a complete reversal. Brent Kung adder is the most robust while Kogge Stone the least. The statistical optimizer, having knowledge of variations, prevents the side paths from being downsized to criticality while maintaining a reasonable overall delay. There is ample opportunity to do this in the Brent Kung adder, but Kogge Stone, owing to its uniform structure, is a poor candidate. So while the σ/μ ratio for Kogge Stone improves, its 95^{th} percentile delay (t_{95}), which depends on the number of critical paths, remains the same. For robustness (as measured by σ/μ ratio), Sklansky adder topology falls in the middle of the two extremes. However, even with the overestimated amount of process variations we have considered, Sklansky topology is still the most energy efficient among the three as the t_{95} points in Table 1 show.

Table 1. SSTA results on statistically optimized adders at 2pJ energy

adder	mean (μ) in ps	σ in ps	σ/μ in % before	σ/μ in %	t_{95} in ps
KS	456	8	2.9	1.78	470
Sklansky	318	5	3.5	1.51	326
BK	352	4.5	4.3	1.28	360

5. Conclusions

With careful optimization for energy-efficiency, most efficient topologies lead to a design that is not far from the best. These adder designs in a given logic family have the same shape for their E–D curves and hence rarely cross each other. Because it has the fewest wires and minimum logic depth, a radix 2 Sklansky adder topology is the most energy-efficient.

Changing the logic family simply shifts the E–D curve, while retaining its shape. Sum-selection improves the energy-efficiency of dynamic adders if the conditional sum generate logic is in static CMOS. Dual rail domino adders are useful only at the highest performance, or if dual rail outputs are needed.

Because logic gates are relatively poor drivers, for any reasonable load, it is always beneficial at lower energies to have inverters drive the output load, as they are the most energy-efficient drivers. At higher energies, however, the inefficiency due to increased number of stages dominates the energy advantage of external buffering.

Overall, circuit optimization can change the delay of 32-bit adders designed in 90nm by an order of magnitude while the energy spans about 60X.

References

[1] S. Boyd, S.-J. Kim, D. Patil, and M. Horowitz. "Digital circuit optimization via geometric programming". *Operations Research*, 53(6):899–932, 2005.

[2] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. "A tutorial on geometric programming", 2004. Manuscript available from `http://www.stanford.edu/boyd/~gp_tutorial.html`.

[3] K. Chen, H. Hu, P. Fang, M. Lin, and D. Wollesen. "Predicting CMOS speed with gate oxide and voltage scaling and interconnect loading effects". *IEEE Transactions on Electron Devices*, 44(11):1951–1957, 1997.

[4] A. Conn, I. Elfadel, W. M. Jr., P. O'Brien, P. Strenski, C. Visweswariah, and C. Whan. "Gradient-based optimization of custom circuits using a static-timing formulation". In *Proceedings of the 36th IEEE/ACM Design Automation Conference*, pages 452–459, June 1999.

[5] J. Fishburn and A. Dunlop. "TILOS: A posynomial programming approach to transistor sizing". In *Digest of Technical Papers of IEEE International Conference on Computer-Aided Design(ICCAD)*, pages 326–328. IEEE Computer Society Press, 1985.

[6] D. Harris. "A taxonomy of parallel prefix networks". In *Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, pages 2213–2217, Nov. 2003.

[7] D. Harris. "Logical effort of higher valency adders". In *Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, pages 1358–1362, Nov. 2004.

[8] D. Harris and I. Sutherland. "Logical effort of carry propagate adders". In *Proceeding of the the Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, pages 873–878, 2003.

[9] S. Knowles. "A family of adders". In *Proceedings of 14th IEEE Symposium on Computer Arithmetic*, pages 30–34. IEEE Computer Society Press, 1999.

[10] R. E. Ladner and M. J. Fischer. "Parallel prefix computation". *Journal of ACM*, 27(4):831–838, Oct 1980.

[11] H. Ling. "High speed binary adder". *IBM Journal of Research and Development*, 25(3):126–130, May 1981.

[12] D. Marković, V. Stojanović, B. Nikolić, M. Horowitz, and R. Brodersen. "Methods for True Energy-Performance Optimization". *IEEE Journal of Solid-State Circuits*, 39(8):1282–1293, 2004.

[13] S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar. "A 4GHz 130-nm address generation unit with 32-bit sparse-tree adder core". *IEEE Journal of Solid State Circuits*, 38(5):689–695, 2003.

[14] MOSEK ApS. *The MOSEK Optimization Tools Version 3 User's Manual and Reference*, 2002. Available from `http://www.mosek.com`.

[15] S. Naffziger. "A sub-nanosecond 0.5 m 64 b adder design". In *Digest of Technical Papers of IEEE International Solid-State Circuits Conference*, pages 362–363, 1996.

[16] V. G. Oklobdzija, B. R. Zeydel, H. Q. Dao, S. Mathew, and R. Krishnamurthy. "Comparison of high-performance VLSI adders in the energy-delay space". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(6):754–758, 2005.

[17] D. Patil and S.-J. Kim. *"The Stanford circuit optimization tool (SCOT) user guide"*, 2006. Available from `http://www.stanford.edu/class/ee371/tools/SCOT_UserGuide.pdf`.

[18] D. Patil, S. J. Kim, and M. Horowitz. "Joint supply, thresold voltage and sizing optimization for design of robust digital circuits". Technical report, Department of Electrical Engineering, Stanford University, April 2006. Avaliable from `http://mos.stanford.edu/papers/JointVddVthSizing.pdf`.

[19] D. Patil, S. Yun, S.-J. Kim, S. Boyd, and M. Horowitz. "A new method for design of robust digital circuits". In *Proceedings of 6th International Symposium on Quality Electronic Design (ISQED)*, 2005.

[20] M. Pelgrom. "Matching Properties of MOS Transistors". *IEEE Journal of Solid State Circuits*, 24(5):1433–1439, Oct. 1989.

[21] J. Sklansky. "Conditional-sum addition logic". *IRE Transactions on Electronic Computers*, EC-9:226–231, 1960.

[22] I. Sutherland, B. Sproull, and D. Harris. *"Logical Effort: Designing Fast CMOS Circuits"*. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[23] Y. Wang and K. K. Parhi. "A unified adder design". In *Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 177–282, 2001.

[24] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison Wesley, third edition, 2004.

[25] R. Zlatanovici and B. Nikolić. "Power-performance optimal 64-bit carry-lookahead adders". In *Proceedings of the 29th European Solid-State Circuits Conference (ESSCIRC)*, pages 321–324, 2003.