Chapter 11

POWER-DELAY CHARACTERISTICS OF CMOS ADDERS

Vojin G. Oklobdzija and Bart R. Zeydel
ACSEL Laboratory, University of California Davis

Abstract: Choosing the right algorithm and a corresponding adder topology depends on many factors closely related to the technology of implementation. With transition to CMOS where circuit delay has a complex relation to implementation parameters and with transition to deep-submicron technology with its own complexity, to make the right choice becomes even more difficult. This relationship is even more complicated with inclusion of power consumption. In this chapter we present this complex relationship and highlight the important factors that influence the right choices in the algorithm, circuit topology, operating conditions and power consumption

Key words: adders, digital arithmetic, digital circuits, energy-delay optimization, VLSI arithmetic, fast digital circuits.

1. INTRODUCTION

For almost half a century realizations of addition algorithms have been continually refined to improve performance due to changing technology and operating constraints. With each technology generation, the gap between the underlying algorithms for addition and efficient realization of those algorithms has grown. Many of the adders in use today were developed for older technologies and under a different set of constraints than those imposed by current technology, such as energy-efficiency. To solve this problem a method for analyzing designs in the energy-delay space was developed [15,16] which allowed for the energy-delay tradeoffs to be taken into account. In addition this method provides guidance for algorithm selection and realization. Using the method we explore the leading addition

recurrence algorithms and their realizations that have been developed, to identify favorable characteristics of each for efficient realization in modern CMOS technology. A comparison of various schemes in the energy-delay space is presented to demonstrate the relative performance and energy-efficiency of the proposed structures.

The most important step in the process of VLSI adder design is selection of the initial adder topology which is expected to yield desired performance in the allotted power budget. However, the performance and power will be known only after a time consuming design and simulation process is completed. Therefore, the validity of the initial selection will not be known until the late stage of the design process, or even after several schemes under consideration have been designed and completed. Going back and forth between several designs is often prohibited by the design schedule, making it impossible to correct initial mistakes. Thus an uncertainty always remains as to whether a higher performance or lower power was possible with a more appropriate choice, different topology or simply more effort. This problem is aggravated by a lack of proper delay and power estimation techniques that are guiding development of computer arithmetic algorithms. The majority of algorithms in use today are based on out-dated methods of counting the number of logic gates on the critical path, thus, producing inaccurate and misleading results. The importance of transistor sizing, load effects and power are not taken into account by most.

Different adder topologies may influence fan-out and wiring density, thus influencing design decisions and yielding better area/power trade-offs than known cases [1]. This emphasizes the disconnect existing between algorithms and implementation. The importance of for fan-in and fan-out effects on the critical path was demonstrated at the time CMOS technology started replacing nMOS [2]. Similar conclusions were expressed later in the Logical Effort (LE) method of Sutherland and Sproull [3] regarding critical-path delay estimation. Further, Logical Effort method was introduced into common practice by Harris [4]. Comparison of delay estimates of various VLSI adders obtained via Logical Effort, to simulation results obtained using H-SPICE [5] demonstrates good matching confirming validity of the Logical Effort. This matching is well under 10% in most cases (Table I). However, this is still an incomplete picture, because delay and energy can be traded against each other, thus, the energy aspect of this analysis is missing.

A method for estimation adder performance which allows for the energy-delay tradeoffs of a design was developed following the Logical Effort guidelines [16]. Using this method it is possible to compare different adders in the energy-delay space (see Fig.1). This method satisfies two requirements: it is simple and quick, yet sufficiently accurate to guarantee correct selection of the appropriate algorithm (topology).

In this chapter we elaborate on the problem of power and performance design trade-offs and estimation of it, showing how different technology parameters affect performance of different algorithms. Further, we want to show how the best algorithm and topology should be selected and point to the most important factors in their selection. Finally, we show the best topology for a power-efficient adder that was obtained in such a way.

2. COMPARISON OF VLSI ADDERS

The most common approach in comparing VLSI adders was to use of a single delay point [1,2]. An example of such comparison (based only on delay) of high-performance 64-bit adders is shown in Table 1. The comparison is showing LE delay estimates and H-SPICE pre-layout simulation in 130nm technology.

Table 11-1. Delay Comparison of 64-bit Adders Using Logical Effort

Circuit Family	Adder Topology	HSPICE (F04)	LE Estimate (F04)
Static	Kogge-Stone [7]	11.8	10.9
	Mux Based Adder [8]	11.4	12.8
	Han-Carlson [9]	12.8	13.3
Dynamic	Kogge-Stone [7]	8.7	9.2
	Ling [11]	9.0	9.5
	Han-Carlson [9]	9.8	9.9

The comparison, show a significant speed difference between Static CMOS and Dynamic CMOS implementations. This fact has been well known to practitioners: all high-speed processors use Dynamic CMOS logic [10,11,14]. However, while the delay difference between different circuit families is more apparent, the delay difference between topologies using the same circuit family is relatively small, making it difficult to know which design can be improved further in terms of speed. Energy is also important because if too much power is used in order to achieve a target delay, hot spots can be created [6].

To illustrate the problem, suppose that two adders A and B were compared against each other based on delay only. Such a hypothetical comparison is illustrated in Fig. 1, where the delay of adder A and adder B are shown as points A and B respectively. From the single point comparison, adder A appears faster than adder B leading to a conclusion that the topology of the adder A is better. However, such a comparison provides an incomplete and potentially misleading picture. If we consider that energy can be traded

for delay it is clear that further analysis is needed. Hypothetical energy-delay dependencies of two designs, A and B, optimized under the same constraints are illustrated in Fig. 1.

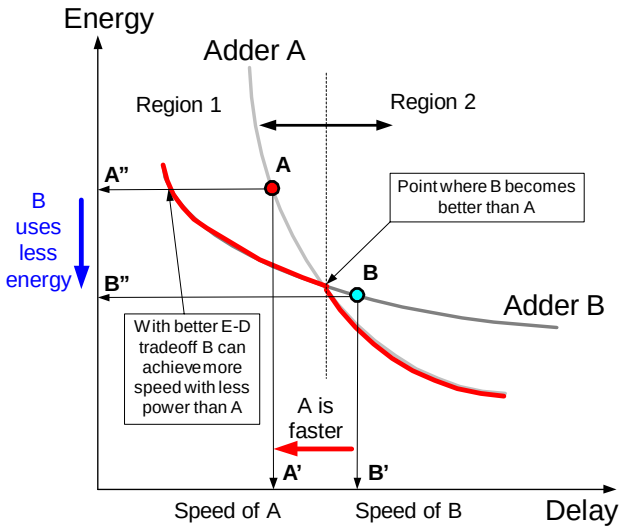


Figure 11-1. Energy-Delay Dependency

As the curves show, adder B has more room for delay improvement, it uses less energy in the high-performance region (Region 1) as compared to the adder A.

On the other hand, if lower computational energy is the design objective, adder A is the better choice as it uses less energy in the low performance region (Region 2), compared to adder B.

The challenge is be able to make such comparison early in the design process and without significant time overhead. A method for estimating energy and delay with relatively low effort and in a short amount of time has been developed [16]. Yet the method provides sufficient accuracy to make appropriate choices of algorithm and circuit topology.

3. DELAY AND ENERGY ESTIMATION

The speed of a VLSI adder depends on several factors: technology, circuit family, adder topology, transistor sizes, wires, leakage currents and second order effects. As a result there are no simple rules to be applied when estimating delay. Skilled engineers are capable of fine-tuning the design to

obtain the best performance and lowest energy through transistor sizing. However, this is often an ad-hoc process not leading to the best solution. Thus it is difficult, if not impossible, to predict the best topology.

3.1 Delay Estimation

Introduction of LE was a significant step forward because it provided a better way to estimate delay. Further, LE provides an optimal sizing for delay. There is a tradeoff in delay estimation where improved accuracy is paid for by complexity or resorting to CAD tools. LE simplifies the delay model to a single parameter referred to as *stage effort f*, which is used during optimization and modeling. The LE model for gate delay is $t_d = (f + p)\tau$, where $f = gh$ [3]. Each gate has a *logical effort*, *g*, which represents its drive capability relative to an inverter. The term *h* represents the effective fan-out of the gate (C_{out}/C_{in}). The parasitic delay, *p*, corresponds to the delay associated with parasitic capacitances. The term τ is the per fan-out delay increment of an inverter, and is used to introduce technology independent estimation of delay.

The accuracy of LE can be improved by obtaining the coefficients *g* and *p* through H-SPICE characterization for the particular technology to be used. This step incorporates characteristics of a particular technology, slopes, and layout estimates into the LE parameters. Gate characterization is performed under the constraint of fixed input-to-output slope relationship to obtain the best matching. This would improve the accuracy of LE estimation considerably and bring it well within 10% of H-SPICE simulation, as shown in Table 1.

The effect of gate-to-gate wiring is not accounted for using basic LE modeling, and is often ignored in comparisons. However, we have observed that in 130nm technology, for example, wire resistance and capacitance can contribute up to 1F04 delay degradation in 64-bit adders. The wire capacitance introduces a constant load at the output of each gate, which can be estimated from the wire length. The impact of wire resistance can be estimated using, $T_{wire}=0.38R_{wire}C_{load}$, which provides reasonable matching versus H-SPICE. The comparison results are shown in Table II.

Table 11-2. Worst Case Delay Impact of Wire Resistance in 130nm 64-bit Adders

Wire Length (bits crossed)	HSPICE (no resistance)	HSPICE (with resistance)	Estimate
80μm (8-bits)	54.7 ps	58.5 ps	58.9 ps
160μm (16-bits)	57.7 ps	66.0 ps	66.8 ps
320μm (32-bits)	64.0 ps	84.7 ps	84.2 ps

Application of LE to simple path delay estimation and size optimization is straightforward; however, it is often difficult to apply the analysis to complex paths due to branching. LE defines branching as $b=(C_{on}+C_{off})/C_{on}$, where the terms C_{on} and C_{off} must be determined relatively. This analysis becomes prohibitively complex when branches have differing gate types and number of stages. In addition constant loads such as wiring, require iterative computation. As a result the optimization of a complex path using the LE gate delay model must be performed by changing individual *f's* of each gate to achieve minimal delay. Instead a simple paper and pencil method (as suggested by LE) use of MS-Excel or other simulation tools such as MathLab is more appropriate because of its built-in gradient based optimization feature, and the fact that it does not require considerable overhead than paper and pencil analysis.

3.2 Energy Estimation

The use of LE for delay optimization provides not only a delay estimate, but the corresponding gate sizing. However, it does not provide for the energy estimation, ignoring it completely. By including the energy model of each gate, obtained from its LE sizing, the total energy of a design can be estimated as developed in [15,16].

The energy of a gate is primarily a function of the output load, C_L, and parasitic capacitance (proportional to gate size). The relative energy associated with C_L and parasitic capacitance varies depending on the effective fan-out, *h*. For small values of *h*, the parasitic energy is comparable to the energy associated with the output load, while for larger values of *h* the energy associated with the output load increases relative to the parasitic energy.

Gate energy can be extracted from H-SPICE simulation by varying C_L and gate size. A linear dependence of energy on C_L and size is observed in [15], which results in the following energy model:

$$E = E_p \cdot gate\ size + E_g \cdot C_L + E_{internal\text{-}wire}$$

where E_p is the energy per unit size, E_g is energy per unit load, and $E_{internal\text{-}wire}$ is an offset for internal wiring introduced by layout estimation. The energy model directly accounts for parasitics, local wiring, and output load, while performing a best fit for crowbar current and leakage. E_g, E_p, and $E_{internal\text{-}wire}$ are obtained using the same gate characterization setup as LE with the slight overhead of performing the characterization for multiple gate sizes.

4. ENERGY-DELAY ESTIMATION METHOD

The objective of the Energy-Delay Estimation method (EDE) is to provide a way to compare designs in the energy-delay space [16], that is relatively simple and quick, so that it can be performed before design decisions were made and committed.

LE provides reasonable delay results and sizing, however it does not account for wiring. To improve LE the inclusion of wires and correct handling of branches must be done. As we are interested in comparing designs over a range of performance targets, each design should be compared over the same range of path gain, $H=C_{out}/C_{in}$. After characterizing a technology to find g, p, E_g, E_p, and E_{wire} for each gate, the following steps are performed to obtain a delay and energy estimate of a design for each H:

1. Determine the critical path of the design.
2. Optimize the delay of the critical path to determine $f_{opt.}$
3. Use f_{opt} to size the gates on the critical path.
4. Estimate the energy of the entire design.

Using the sizing from Step 3 we can estimate the energy of the critical path. However, Step 4 requires an estimate for the energy of the entire design and not just the critical path. The energy of gates within a design can be estimated according to two cases: gates on paths with the same number of stages as the critical path; and gates on paths with fewer stages than the critical path.

For paths with the same number of stages as the critical path, the size of each gate is proportional to the gates on the critical path, allowing for the energy of each gate to be computed directly. To facilitate this analysis, the energy of each gate is assumed to have the same energy of the gate on the critical path.

For paths with different number of stages than the critical path, the size of each gate is not directly proportional to the gates on the critical path. Instead, to obtain an energy estimate, the path must first be sized to have the same delay as the critical path. Once the sizing is obtained, the energy of each gate can be estimated. Similar to first case, any subsequent paths with the same number of stages can be computed proportionally to this path.

The energy of each gate depends on its switching activity. In application of EDE to VLSI adders it is common to use a 15% switching activity factor for each gate in the static adders and a 50% switching activity for each gate in the dynamic adders. These switching factors were obtained as an average

of designs that were analyzed based on experience consistent with the “rule of thumb” used in the industry.

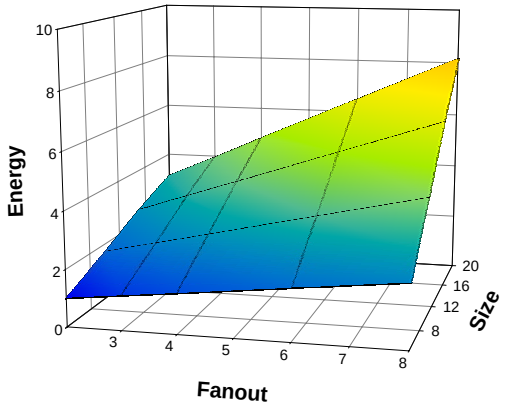


Figure 11-2. Energy dependence of a gate on fanout and size

5. ENERGY-DELAY ESTIMATION OF ADDERS

Energy-Delay Estimation is a useful tool in comparing various tradeoffs in adder design, such as the algorithm or circuit topology. The choice of the adder topology and design style is also dependent on the required performance and pressures to meet the critical path. In some instances, the adder may not be in the critical path and the speed requirements may be relaxed, or it may be possible to improve the speed of the clocked storage elements (flip-flops and latches) and meet the required timing in this way. The analysis of these tradeoffs was performed by Zyuban who termed these design characteristics “hardware intensity” [17,18]. Hardware intensity defines the design point in terms of the trade-off between energy and delay. A tangent on the energy-delay curve represents the percentage of delay reduction being paid for by the percentage increase in energy. Thus, various algorithms can be examined in various design regions and the best one can be chosen. Also, various circuit design styles can be examined. For example, the analysis of three different circuits design styles: Static CMOS, Dynamic CMOS and Compound (Dynamic-Static) CMOS design, reviled that Compound CMOS achieves speed of dynamic design while maintaining the low energy of static-CMOS design. The figure comparing the three different design styles is shown in Fig.3.

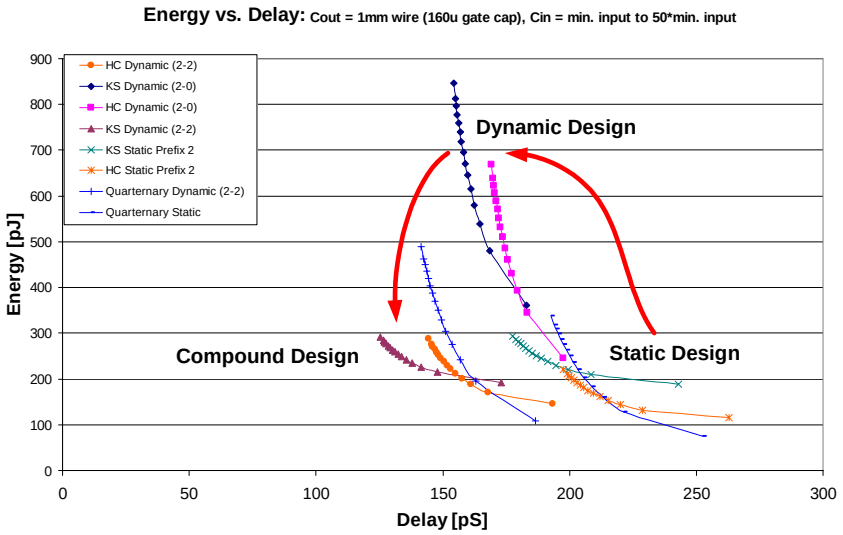


Figure 11-3. Different design styles: Compound CMOS shows benefits of Static and Dynamic CMOS circuits.

5.1 Domino and Compound Domino CMOS Analysis

In order to improve adder performance, Domino CMOS logic is often used for implementation of adder Carry-Merge (CM) blocks resulting in the circuit shown in Fig. 4a. The static CMOS inverter is necessary after each dynamic block in order to make the logic behave in a “domino” fashion. The inversion of the signal, which is necessary in the Domino CMOS logic block, can be achieved with a more complex inverting static gate, which performs additional function. This is often referred to as Compound-Domino, or Dynamic-Static CMOS. Thus, two Domino Carry-Merge stages can be merged into one by replacing the inverter with an AOI (Fig. 4b).

The difference between Domino and Compound-Domino CMOS circuit realization of the Carry-Merge stage is shown in Fig. 4.

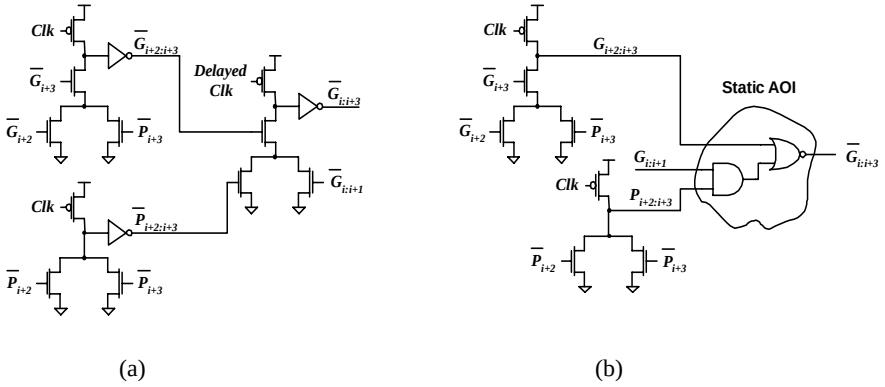


Figure 11-4. (a) Carry-Merge: Domino Implementation (b) Carry Merge: Compound Domino Implementation

Comparison of the 64-bit Kogge-Stone (KS) [7] and Han-Carlson (HC) [9] adders implemented in: Dynamic CMOS Domino and Compound-Domino CMOS is shown in Fig. 5.

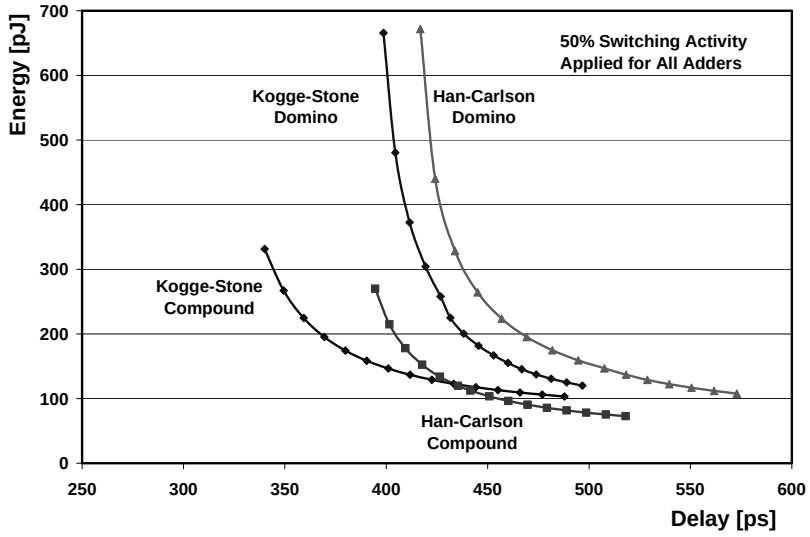


Figure 11-5. Comparison of 64-bit HC and KS Domino and Compound-Domino Adders in 130nm technology

Comparing Domino to Compound-Domino, EDE helps us to appreciate the benefits obtained by utilizing Compound-Domino logic. For the same energy budget (e.g. 200pJ) Compound-Domino KS yields 20% delay

improvement over Domino KS. EDE provides a clear picture of the impact Compound-Domino can have on adder design.

6. ADDER COMPARISON

The ability, provided by EDE method, to observe differences between implementations of the same adder using different circuit families is beneficial in selecting appropriate circuit design style. However it is also important to see tradeoffs between adder topologies implemented using the same circuit family. The accuracy of EDE for demonstrating tradeoffs in the energy-delay space is shown by comparing optimized H-SPICE results for 32-bit Compound-Domino KS [7] and QT [10] adders versus EDE results in 100nm technology. The simulation results for KS and QT in 130nm were shown. Comparison of the simulation results with EDE estimation adopted for 100nm is shown in Fig. 6 [13]. EDE estimates demonstrate the same tradeoffs as observed in simulation. These results confirmed the validity of choosing the QT adder over KS or HC as a viable option for reducing energy without sacrificing performance.

Compound-Domino and static 64-bit adders were analyzed using EDE to see what tradeoffs exist (Fig. 7). Different points on the energy-delay curve were obtained by varying the size of the input gates for each adder. The output of each adder was loaded with a 1mm wire. A range of H was chosen from H=2 to the maximum H (i.e. where minimum input size occurs).

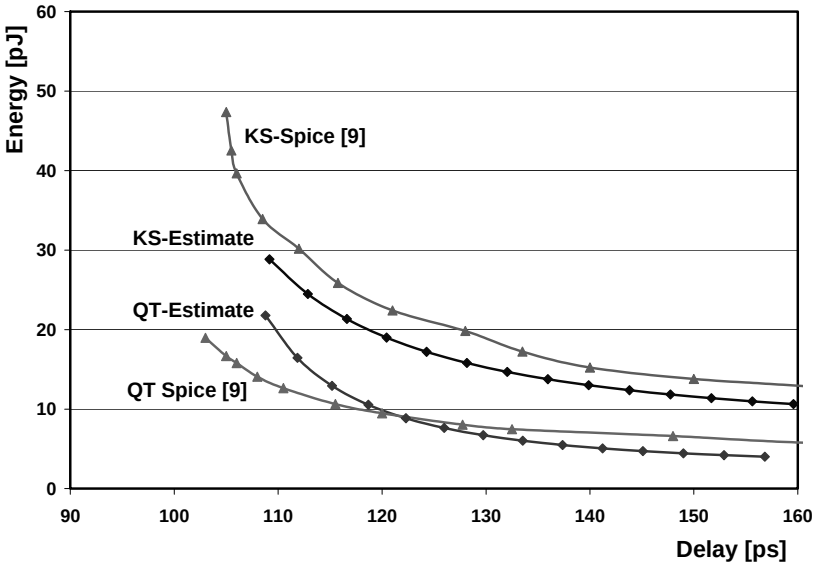


Figure 11-6. Comparison of 32-bit QT and KS adders: EDE vs. simulation in 100nm technology.

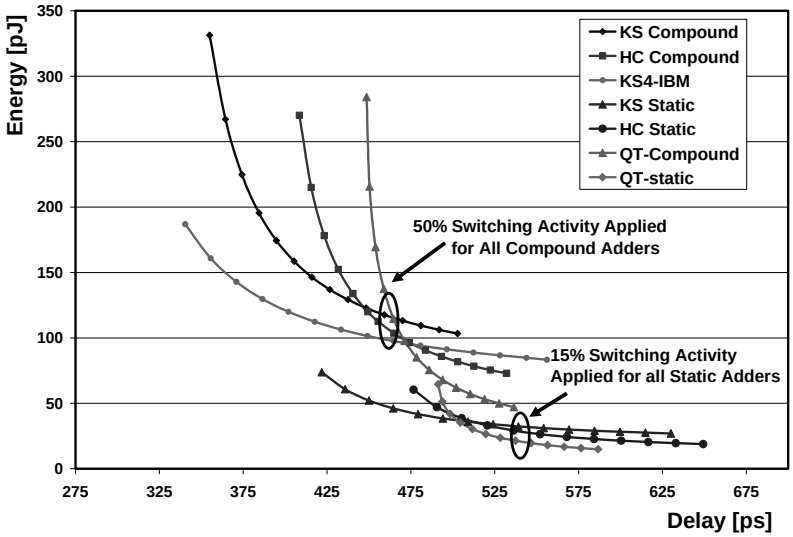


Figure 11-7. EDE Analysis of 64-bit Compound-Domino and Static Adders in 130nm technology

The results show the benefits associated with sparse designs: HC and QT, however the benefits of Compound-Domino QT are lesser at 64-bits than for the 32-bit design. This is a result of the one extra stage that the QT implementation used versus the KS implementation. In the 32-bit design the QT implementation used the same number of stages as KS. We also observe that a Compound-Domino KS prefix-4 adder KS4-IBM [12] which utilizes fewer stages at the cost of increased gate complexity and branching shows further benefits. The increased gate complexity in the KS4-IBM adder is offset by a significant reduction in parasitic delay associated with the number of stages, which allows for the KS4-IBM to achieve lower delay with less energy penalty than the other designs. At lower performance targets this overhead is dominating and designs such as QT and HC achieve lower energy.

6.1 Representative High-Performance Adders

We show comparison of high-performance adders used in leading processors in the industry. All of the adders compared were implemented using compound-domino design style which combines the best of the "both worlds": low-power and high-performance as it has been realized by the design community. The comparison, shown in Fig. 8, includes: IBM implementation of KS adder [12], Kogge-Stone 4-2 consisting of fan-in of 4 dynamic and fan-in of two static compound domino stage [7], Quarternary adder developed by Intel [10], Ling adder used by IBM and Intel Itanium processor and Han-Carlson adder [9]. We choose to include Energy-Delay^2 factor as a figure of merit of their respective designs. As we know, this is only one point on the E-D curve and the one chosen by high-performance designs [24]. In Zyuban's analysis, this represents "hardware intensity" equal to 2, which means that they are willing to trade 2% of energy increase for 1% increase in speed [17,18]. The tangent to the E-D curve at this point has a slope of 2. We see that even though it is not the fastest, IBM adder (developed by Park) shows the best ED^2 figure of merit.

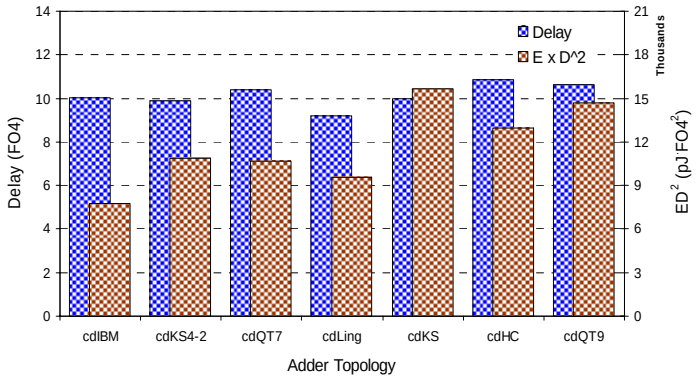


Figure 11-8. Comparison of representative high-performance adders used in the industry. "cd" designates Compound-Domino circuit design style.

6.2 Contribution of Wires

In order to properly evaluate all of the adder topologies an important factor such as the wire effect and contribution to adder performance and energy consumption needs to be properly qualified. The wires contribute in two ways: add delay to the adder (effect of long wires), and use energy. In the past, these effects were negligible, but as the technology continues to scale wire effects could make a substantial difference. If we are to model performance in the energy-delay space, the energy contribution of wires, as well as delay must be appropriately included in the model.

Fig. 9 shows the wire effects expressed as a total wire capacitance in the adder. This capacitance contributes to performance and energy deterioration. The impact of wires on the delay is shown in Fig. 10. Depending on the wire length of wires on the critical path, wire delay can impact the adder delay by up to 1FO4 (as simulated for 100nm technology node). Given the speed of the adders, this this amounts to more than 10% of the total delay.

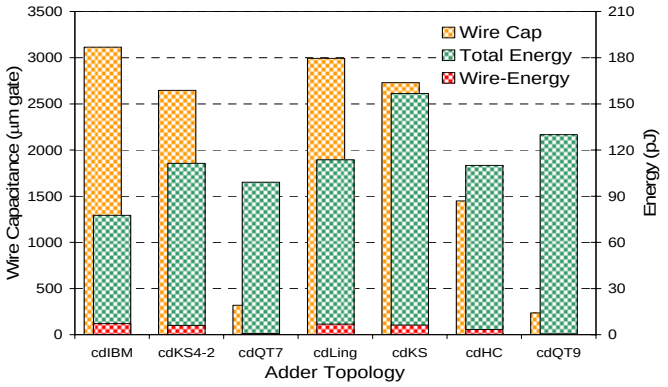


Figure 11-9. Impact of Wires on High-Performance Adders

This impact is expressed in total wire capacitance, which influences delay as well as energy. Wire energy is shown as a fraction of the total energy of the adder. The trade-off between. wire and delay is best seen on the IBM adder.

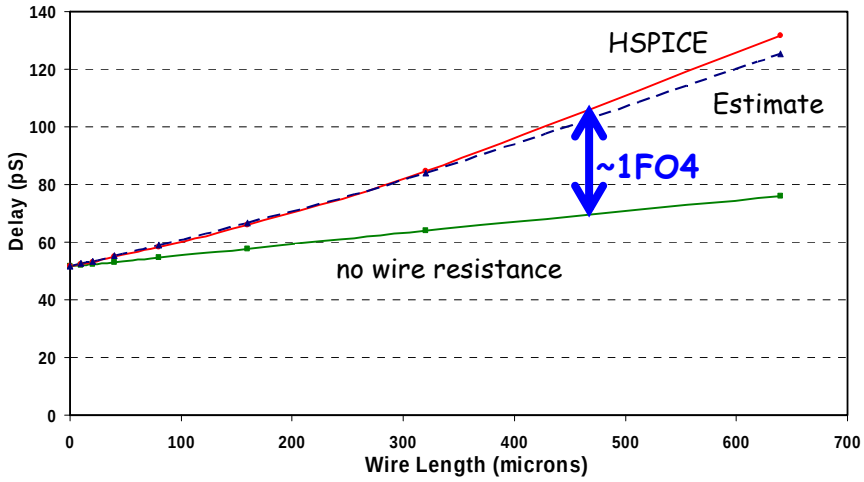
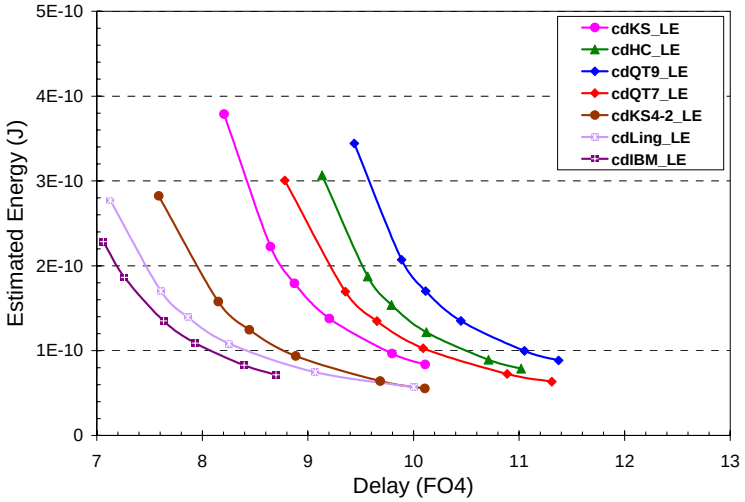


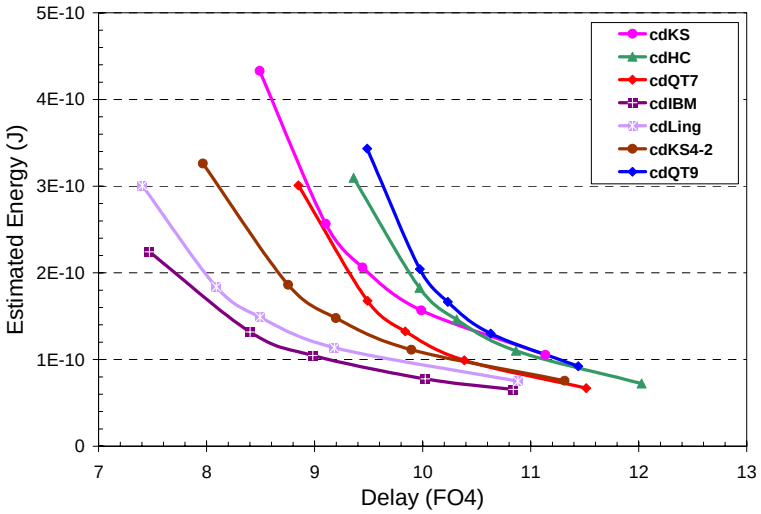
Figure 11-10. Worst Case Delay Impact of Wires

However, given the difference between the fastest and the slowest adders chosen from the group of the best adders, this difference can change their

ranking. Comparison of representative adders, with and without wire effect is shown in Fig. 11(a),(b).



(a)



(b)

Figure 11-11. Energy-Delay behavior of representative high-performance adders used in the industry: (a) Energy-Delay with wire excluded from the model, (b) Energy-Delay with wire

impact included. It should be noted that in general, wire effect do diminish performance differences between them, while energy increases only slightly.

The impact of wire is to diminish the differences between the best and the worse scheme. This shows that a hidden trade-off exists in some of these schemes between the wiring and logic complexity. We can trade a stage of the adder for a more intense wiring. If wire contribution is not properly accounted for, unfair advantage may be apparent, but this appearance is misleading. Looking at, for example, 6-stage KS with 7-stage QT7, in Fig. 11(a) and (b), one notice that they completely switch order when wire effects are properly accounted for.

By applying optimization techniques to transistor sizing [optim ref] the energy can be reduced even further. The ultimate energy-delay figure of the representatinve adders is shown in Fig. 12. This is about the best energy and performance one can achieve [25].

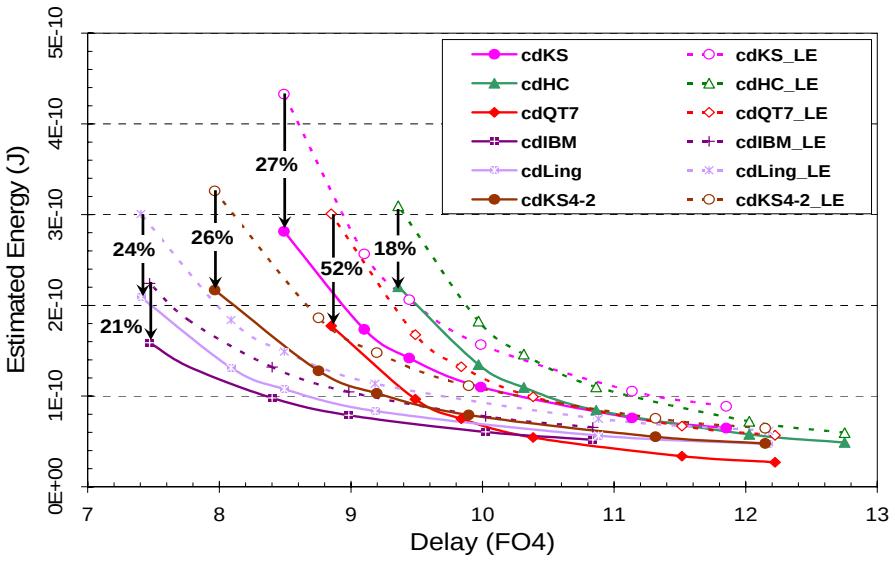


Figure 11-12. Energy-Delay behavior of representative high-performance adders after energy optimization. The best behavior is that of IBM adder designed by Park [12].

7. THE ULTIMATE ADDER TOPOLOGY

Efficient adder design requires proper selection of a recurrence algorithm and its realization. Using the insight obtained through the application of

EDE tool, we analyzed several algorithms for their flexibility and suitability for realization in CMOS. We found that the use of Ling's algorithm provides up to 12% improvement in performance of 32-bit static adders with similar recurrence trees. Using Ling's algorithm we developed general techniques for efficient realizations based on technology constraints. From these techniques we propose several high-performance realizations of Ling's algorithm that achieve better performance and energy efficiency than existing Ling and Weinberger designs [23].

Technology characteristics limit potential realizations of Weinberger's and Ling's recurrences for addition. The primary constraint in current CMOS is fan-in of a gate, which is commonly limited to between 3 and 5. Several realization techniques have been developed to map recurrence algorithms to CMOS under these constraints [23].

7.1 Combined Bit Operator and 1st Carry Stage

In dynamic adder implementations, one stage can be removed by combining the 1-bit operation for *g* and *t* into the first prefix computation stage [11,12]. This technique is more favorable to Ling's recurrence than to Weinberger's. Under the same stack height constraint a Ling realization can use a prefix of 1 more than a realization using Weinberger's.

Ling's first stage prefix-2 operation using 1-bit operators:

$$H_i = g_i + g_{i-1}$$

$$T_i = t_i \cdot t_{i-1}$$

Weinberger's first stage prefix-2 operations using 1-bit operators:

$$C_{i+1} = g_i + t_i \cdot C_i$$

$$T_i = t_i \cdot t_{i-1}$$

It is seen that the recurrence of H_i involves one less term than C_i, however the recurrence for T does not differ between the structures. The result is that the critical path through first stage of the recurrence tree has the same fan-in on the gates, whether using Ling or Weinberger. However when the 1-bit operators are combined with the first prefix stage of the carry tree (for example prefix 2) the resulting logic for Ling's transformation is:

$$H_i = a_i b_i + a_{i-1} b_{i-1}$$

$$T_{i-1} = (a_{i-1} + b_{i-1})(a_{i-2} + b_{i-2})$$

and for Weinberger's recurrence:

$$C_{i+1} = a_i \cdot b_i + (a_i + b_i)(a_{i-1} b_{i-1})$$

$$T_i = (a_i + b_i)(a_{i-1} b_{i-1})$$

The fan-in of the first logic stage for Ling's transformation is reduced by 1 in CMOS compared to Weinberger's. Subsequent stages for both have the same fan-in since the recursion is performed using the first stage operations with the prefix operator. Ling's transformation is especially useful for static realizations, as the first stage and bit operator stage can be combined, while in Weinberger such a combination would result in 3-stacked pMOS transistors.

7.2 Conditional Computation of Sum

Several adder implementations make use of conditional logic for the computation of sum. Conditional logic allows for the number of recurrence terms to be computed at the cost of increased fan-out in the recurrence tree. Both Weinberger's and Ling's recurrence fit well into conditional computation. The issue with conditional computation is determining how many bits to compute conditionally and using what structure. The number of gates in a critical path consists of the carry structure (and 1-bit operator gate if not combined), and the sum. For example a 4-gate carry tree the critical path would require 5-gates. The conditional sum calculation requires that the conditional sum be calculated prior to final sum, so in a realization with a 4-gate carry tree the conditional sum must be completed by in 4-gate delays.

The following is an estimate for the maximum number of bits that can potentially be rippled for an n-stage carry tree using conditional sum, ripple carry structure.

$$\#\,gates = \left(carry\_tree\_stages + sum\right) - sum - bit\_level\_operator$$
$$\#\,gates = carry\_tree\_stages - 1$$

As the number of stages in the carry tree increases, conditional computation is a viable solution for reducing energy. If however, the number of stages in the carry tree decreases, the possibility that the conditional sum becomes the critical portion of the design increases. The optimal number of bits to compute conditionally as well as the implementation of the conditional computation, either by rippling the recurrence or through the use of separate recurrence trees, is dependent on technology.

7.3 Ling Realizations and their Alternatives

7.3.1 Static Adders

For static adders technology limits designs to a 2stack nMOS transistors and 2 stack pMOS transistors. Knowles described in [1] how to create minimum depth carry trees for static adders using Weinberger's recurrence. The same trees can be constructed with Ling's transformation by combining the first stage of the carry tree as described in 7.1. The construction allows for one stage to be removed from the critical path of the adder.

7.3.2 Dynamic Adders

Ling's transformation shows advantages of reduced logic complexity of the critical path in CMOS technology and should therefore yield the good structures for addition. Several types of 64-bit Dynamic realizations of Ling's transformation are proposed in the following sections.

7.3.3 Fast Parallel Prefix Ling Adders

Ling's transformation only displays advantages over Weinberger's when the factored p_i is removed from the carry computation. As shown in [21,22] this can be accomplished through the use of conditional logic for the sum. The fastest Ling implementations are dependent on CMOS technology limitations. In current technology nMOS transistor stack height is commonly limited from 3 to 5, while pMOS transistors are further limited to a stack height of 2. A feature common to Weinberger's and Ling's transformations is that any prefix (up to 3 and 4 respectively) can be used in the first stage with a stack height of 5 when using operands as inputs to the first carry stage.

7.3.4 Ling with Full Prefix Carry Tree

A three stage adder can be constructed using a fully parallel prefix tree with Ling transformation. A technology limitation of 5 stack nMOS for dynamic stage allows for prefix-4 gates to be used in dynamic stages, while a limitation of 2 stack pMOS for static limits static gates to prefix 2. Under these constraints a full prefix tree with prefix 4, 2, 4, and 2 for the first, second, third and forth gates respectively can be constructed (Ling 4-2-4-2) (Fig. 13).

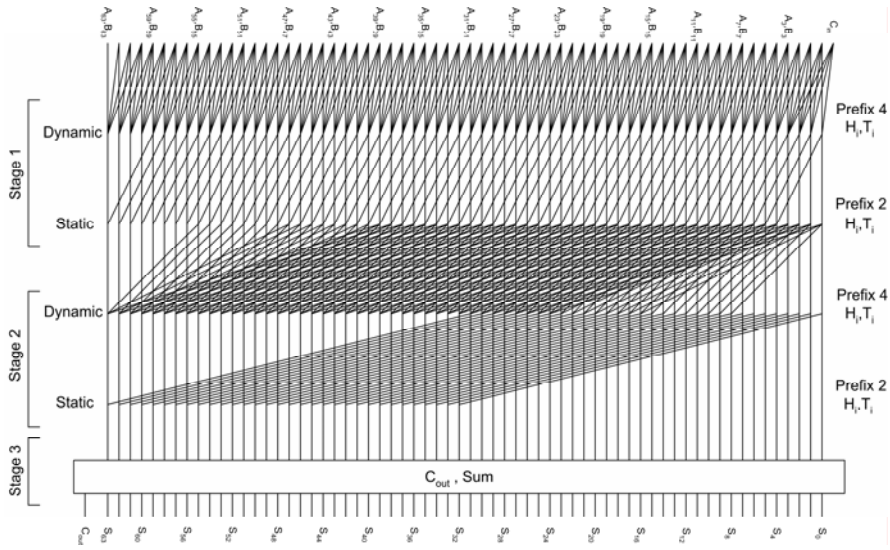


Figure 11-13. Prefix 4-2-4-2 Adder using Ling's Transformation

The equations for the first level H_i and T_i are:

$$H_i = A_i B_i + A_{i-1} B_{i-1} + (A_{i-1} + B_{i-1}) A_{i-2} B_{i-2} + (A_{i-1} + B_{i-1})(A_{i-2} + B_{i-2}) A_{i-3} B_{i-3}$$

$$T_i = (A_i + B_i)(A_{i-1} + B_{i-1})(A_{i-2} + B_{i-2})(A_{i-3} + B_{i-3})$$

Resulting in a worst case stack height of 4 nMOS transistors for both equations, since it is the first stage of the path that must be footed which bring the worst case height to our technology limit of 5. The second, third and forth level H_i and T_i computations follow traditional dot product operations for prefix 2 and prefix 4. Weinberger's can be used with the same full parallel prefix tree for the carry recurrence at an increase of one in the stack of the first stage for the carry recurrence relative to Ling. Variations of such structures within the technology limitations that are close to the minimal carry tree depth should be analyzed when determining an adder topology to for a technology.

7.3.5 Ling with Sparse-2 Prefix Carry Tree

The amount of wiring in an adder realization can be reduced without increasing the number of stages by generating every other H_i and performing a conditional two-bit sum combined with Prefix 4-2-4-2 carry tree. The conditional sum length was chosen based on the limitations on the number of conditional sum bits described in 7.2 (Fig. 14).

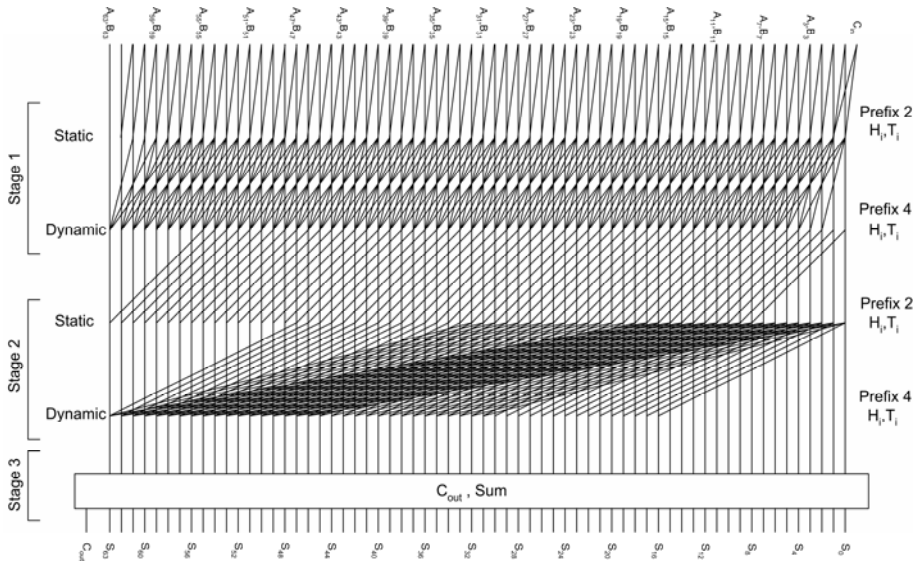


Figure 11-14. Sparse-2 Prefix 4-2-4-2 Adder using Ling's Transformation

The increased complexity of the sum computation requires no additional stages for the conditional computation of the sums. Similar to the full-prefix structure in 7.3.4 such a structure should be modified according to the technology implications on the conditional sum computation speed energy relative to the carry tree speed and energy.

7.4 Results

All results are obtained using estimates for 130nm technology by applying the energy-delay estimation method we developed in [15] to the entire adder. A comparison of 32-bit static adder implementations between Weinberger's recurrence and Ling's transformation is shown in Fig. 15.

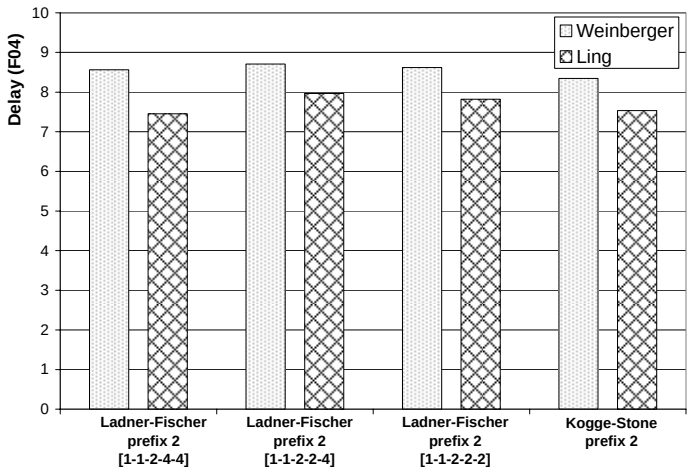


Figure 11-15. Comparison of 32-bit static Weinberger and Ling Adders.

Ling's transformation yields an improvement in delay of up to 12% confirming the reduced number of stages benefit that Ling can achieve in static implementations limited to a 2-stack of pMOS transistors. For dynamic implementations, technology constraints and adder size determine whether the advantage of using Ling's transformation is a logic stage or a reduction in transistor stack height.

A comparison of 64-bit Ling dynamic adders with and without conditional sum is shown in Fig. 16.

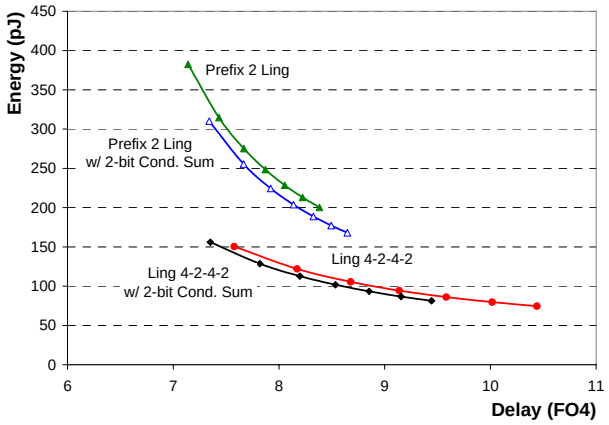


Figure 11-16. Comparison of Conditional Sum in High-Performance 64-bit Dynamic Adders.

The results show an energy savings for the 2-bit conditional sum variants. This is primarily due to the reduced switching activity of the static gates on the conditional path. In the fully parallel prefix-2 Ling carry tree, applying a 2-bit conditional sum improves energy at only a slight increase in delay. The delay penalty is due to increased loading of the adder input caused from the static gates of the conditional sum path. The Ling 4-2-4-2 with 2-bit conditional sum results in a slight energy savings and improved performance compared to the Ling 4-2-4-2 design. In contrast to the prefix-2 design, the static gates of the conditional sum path reduce the loading of the inputs to the adder due to their reduced complexity compared to the prefix-4 gates of the carry path. A comparison of the best 64-bit adder implementations for Ling [19] and Weinberger's recurrence [20] and the proposed realizations are shown in Fig. 17.

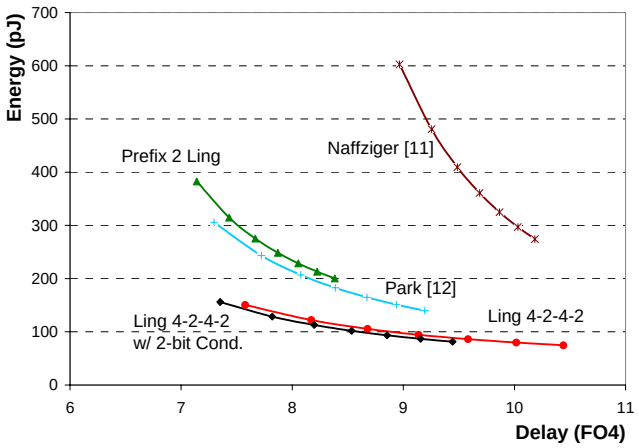


Figure 11-17. Energy-Delay Space Comparison of High-Performance 64-bit Dynamic Adders

The results show the significant advantage obtained by the proposed realizations. These realizations demonstrate better performance than the Weinberger adder and the previous best implementation of a Ling adder. While the best delay is obtained by the fully parallel prefix 2 Ling design, the most efficient design is the proposed Ling 4-2-4-2 with 2-bit conditional sum.

8. CONCLUSION

We presented Energy-Delay Estimation (EDE) method based that extends logical effort (LE) and its application to the analysis and selection of

high-performance VLSI adders. The EDE method has proven to be a much needed and effective tool in design space exploration, in particular when comparing high-performance adders in the early stages of design. Further, the sufficient accuracy of the method for adder selection in the energy-delay space was demonstrated when comparing designs implemented in 130nm and 100nm CMOS technologies using static and dynamic circuit styles. The method described in brings a new perspective to comparing arithmetic circuits, and advances the analysis and design of VLSI oriented computer arithmetic algorithms.

Ling and Weinberger addition recurrence algorithms demonstrate favorable characteristics for efficient CMOS mapping. Ling shows a fundamental advantage in CMOS by reducing the complexity of the first stage of the carry tree. Guidelines which aid in the selection of efficient realizations of Ling's transformation are given for both prefix selection for successive levels of recurrence and selection of conditional computation size. The proposed prefix 4-2-4-2 structures demonstrate up to 25% savings in energy at the same delay and from 0.5-1FO4 delay improvement at ISO-delay when compared to the fastest previous designs [12].

9. REFERENCES

[1] S. Knowles, "A Family of Adders", Proceedings of the 14th Symposium on Computer Arithmetic, Adelaide, Australia. April 1999.

[2] V. G. Oklobdzija and E. R. Barnes, "On Implementing Addition in VLSI Technology," IEEE Journal of Parallel and Distributed Computing, No. 5, 1988 pp. 716-728.

[3] E. Sutherland, and R. F. Sproull, "Logical Effort: Designing for Speed on the Back of an Envelop," IEEE Advanced Research in VLSI, C. Sequin (editor), MIT Press, 1991.

[4] I.E. Sutherland, R.F. Sproull, and D. Harris, "Logical Effort Designing Fast CMOS Circuits," Morgan Kaufmann Publishers, 1999.

[5] H.Q. Dao, V. G. Oklobdzija, "Application of Logical Effort Techniques for Speed Optimization and Analysis of Representative Adders," 35th Annual Asilomar Conference on Signals, Systems and Computers, 2001.

[6] D. Harris and S. Naffziger, "Statistical clock skew modeling with data delay variations," IEEE Trans. VLSI Systems, vol. 9, pp. 888–898, Dec 2001.

[7] P.M. Kogge and H.S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations", IEEE Trans. Computers Vol. C-22, No. 8, Aug. 1973, pp.786-793.

[8] Farooqui, V. G. Oklobdzija, F. Chehrazi, "Multiplexer Based Adder for Media Signal Processing", International Symposium on VLSI Technology, Systems, and Applications, Taipei, Taiwan, June 1999.

[9] T. Han, D. A. Carlson, and S. P. Levitan, "VLSI Design of High-Speed Low-Area Addition Circuitry," Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors,1987, pp.418-422.

[10] S.K. Mathew et al, "Sub-500-ps 64-b ALUs in 0.18 m SOI/bulk CMOS: design and scaling trends," IEEE Journal of Solid-State Circuits, vol.36, Nov. 2001, pp.1636-46.

[11] S. Naffziger, “A Sub-Nanosecond 0.5 m 64-b Adder Design”, 1996 IEEE International Solid-State Circuits Conference, Digest of Technical Papers, Feb. 1996, pp.362-363.

[12] J. Park, et. al., “470ps 64-Bit Parallel Binary Adder”, 2000 Symposium on VLSI Circuits Digest of Technical Papers.

[13] B. Davari, R.H. Dennard, G.G. Shahidi, “CMOS Scaling for High Performance and Low Power – The Next Ten Years,” Proceedings of the IEEE, vol. 83, April, 1995.

[14] S.K. Mathew et al, “A 4GHz 130nm Address Generation Unit with 32-bit Sparse-tree Adder Core,” IEEE Journal of Solid-State Circuits, vol. 38, May 2003, pp.689-695.

[15] V.G. Oklobdzija, B.R. Zeydel, H.Q. Dao, S. Mathew, R. Krishnamurthy, “Energy-Delay Estimation Technique for High-Performance Microprocessor VLSI Adders,” Proceedings of the 16th International Symposium on Computer Arithmetic, Santiago de Compostela, Spain, June 2003.

[16] V. G. Oklobdzija, B.R. Zeydel, H.Q. Dao, S. Mathew, R. Krishnamurthy, “Comparison of High-Performance VLSI Adders in Energy-Delay Space”, Transaction on VLSI Systems, Volume 13, Issue 6, pp. 754-758, June 2005.

[17] V. Zyuban and P. Strenski, “Balancing Hardware Intensity in Microprocessor Pipelines,” IBM Journal of Research and Development, Vol. 47, No. 5/6, September/November 2003.

[18] V. Zyuban, P. Strenski, “Unified Methodology for Resolving Power-Performance Tradeoffs at the Microarchitectural and Circuit Levels,” Proc. Int. Symp. on Low Power Electronics and Design, Aug. 2002, pp. 166-17.

[19] H. Ling, “High-Speed Binary Adder,” IBM Journal of Research and Development, vol. 25, no.3, pp. 156-166, May 1981.

[20] A. Weinberger, J.L. Smith, “ A Logic for High-Speed Addition,” Nat. Bur. Stand.. Circ., 591:3-12, 1958.

[21] J. Sklanski, “Conditional-Sum Addition Logic,” IRE Trans. on Electronic Computers, Vol. EC-9, No2, pp.226-231, 1960.

[22] O. J. Bedrij, “Carry-Select Adder,” IRE Trans. on Electronic Computers, Vol. EC-11, pp. 340-346, 1962.

[23] B.R. Zeydel, T.T.J.H. Kluter, V. G. Oklobdžija, "Efficient Mapping of Addition Recurrence Algorithms in CMOS", International Symposium on Computer Arithmetic, ARITH-17, Cape Cod, Massachusetts, USA, June 27-29, 2005.

[24] K. Nowka, IBM Austin Research Lab, Austin, TX, private communication, August 2001.

[25] V. G. Oklobdzija, “Energy-Delay Tradeoffs in CMOS Digital Circuits Design”, Presentation at Dallas IEEE CAS Workshop, Richardson, Texas, October 10, 2005. (IEEE Xplore: ……)