Energy-Delay Estimation Technique for High-Performance Microprocessor VLSI Adders

Vojin G. Oklobdzija[1], Bart R. Zeydel[1], Hoang Dao[1], Sanu Mathew[2], Ram Krishnamurthy[2]

[1]**ACSEL**
University of California
Davis, CA 95616
www.ece.ucdavis.edu/acsel

[2]**Intel Corporation**
Circuit Research Labs
Hillsboro, OR 97124
Sanu.k.Mathew@intel.com

Abstract

In this paper, we motivate the concept of comparing VLSI adders based on their energy-delay trade-offs and present a technique for estimating the energy-delay space of various high-performance VLSI adder topologies. Further, we show that our estimates accurately represent tradeoffs in the energy-delay space for high-performance 32-bit and 64-bit processor adders in 0.13μm and 0.10μm CMOS technologies, with an accuracy of 8% in delay estimates and 20% in energy estimates, compared with simulated data.

1. Introduction

In the course of VLSI processor design it is very important to choose the adder topology that would yield the desired performance. However, the performance of a chosen topology will be known only after the design is finished. Therefore a lingering question remains: could we have achieved a higher performance, or could we have had a better VLSI adder topology? The answers to those questions are generally not known. There is no consistent and realistic speed estimation method employed today by the computer arithmetic community. Most of the algorithms are based on out-dated methods of counting the number of logic gates in the critical path producing inaccurate and misleading results. The importance of loading and wire delay is not taken into account by most. Knowles has shown how different topologies may influence fan-out and wiring density thus influencing design decisions and yielding better area/power than known cases [1]. This work has further emphasized a disconnect existing between algorithms that are used to derive VLSI adder topologies and the final result. In previous work we have shown the importance of accounting for fan-in and fan-out on the critical path, not merely the number of logic levels [2]. This has led to the method of Logical Effort (LE) [3], which has been popularized by Harris [4]. Recently, we used Logical Effort to estimate the speed of various VLSI adders and we compared those results with those obtained using a more complex circuit simulation tool H-SPICE [5]. This comparison showed a good match and pointed to the right direction. However, the process of analysis was now time consuming and did not provide a comparison for various circuit sizing that could have been applied. This paper is organized as follows: the second section discusses speed estimation using more realistic measures such as logical effort, the third section introduces the energy effects and discusses the performance in the energy-delay space, the fourth section describes the estimation tool that was developed, the fifth section shows results applied to several well known adder topologies and compares them with simulated results in 0.13μm technology.

This work has been supported by SRC Research Grant No. 931.001 and California MICRO 01-063

2. Speed Estimation

The speed of a VLSI adder depends on many factors: the technology of implementation (and its own internal rules), circuit family used for the implementation, sizing of transistors, chosen topology of the VLSI adder, and many other second order effect parameters. There were no simple rules that could be applied when estimating VLSI adder speed. Skilled engineers are capable of fine-tuning the design by carefully selecting transistor sizes, obtaining the best performance and energy trade-off. Therefore it is very difficult, if not impossible, to predict which of the topologies developed by the computer arithmetic community is best, even if it is really useful.

2.1 Logical Effort

Logical Effort methodology takes into account the fact that the speed of a digital circuit block is dependent on its output load (fan-out) and its topology (fan-in). Further, LE introduces technology independence by normalizing the speed to that of a minimal size inverter

which makes the comparisons of different topologies, implemented in different technologies, possible. For proper understanding and further reading of this paper the reader should be familiar with the LE methodology [3,4]. We will briefly describe some of the main features of LE in this sub-section. The delay expression of a logic block in LE is given as:

$$d = f + p \tag{1}$$

where p = parasitic delay, f = effort or stage delay. Further f = gh where g is defined as logical effort and h as electrical effort. Thus:

$$d = gh + p \tag{2}$$

This dependency is illustrated in Fig. 1.

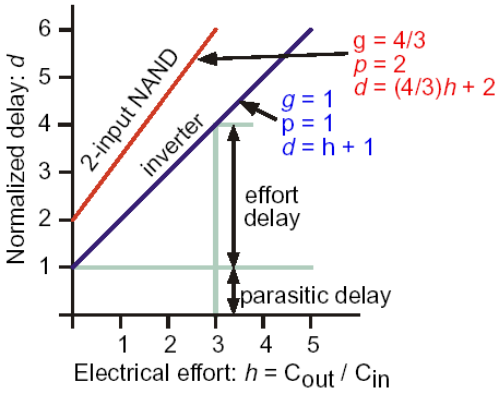

Fig. 1. Delay expressed in terms of a minimal size inverter [3,4]

An important result of LE is that it provides a way of determining appropriate transistor sizing of the critical path to minimize delay. LE also provides an estimate of the critical path delay. Logical Effort results are summarized in Table 1.

Table 1: Logical Effort Equations

Path Logical Effort	$G = \prod g_i$
Path Electrical Effort:	$H = \prod p_i = \frac{C_{out}}{C_{in}}$
Branching Effort	$b = \frac{C_{off-path} + C_{on-path}}{C_{on-path}}$
Path Branching Effort:	$B = \prod b_i$
Path parasitic delay	$P = \sum p_i$
Path Effort:	$F = GBH$

Logical Effort tells us that the delay will be minimal when each stage bears equal effort given as:

$$\hat{f} = g_i h_i = F^{\frac{1}{N}} \tag{3}$$

In such a case, delay of the path will be equal to:

$$D = N\hat{f} + P \tag{4}$$

In order to calculate optimal transistor sizes to achieve minimal delay, we start from the output and calculate C_{in} for each stage, which determines the sizing of each stage.

2.2. 64-bit Adder Speed Comparison using Logical Effort

We used several representative topologies and performed critical path analysis using Logical Effort technique to compare performance. The adders that were examined were: (a) Static: Kogge-Stone (KS) radix-2 [6], Mux-based carry-select [7], and Han-Carlson (HC) radix-2 [8,9] (b) Dynamic: KS radix-2, Ling Adder [10], HC radix-2 and CLA adder with 4-bit grouping.

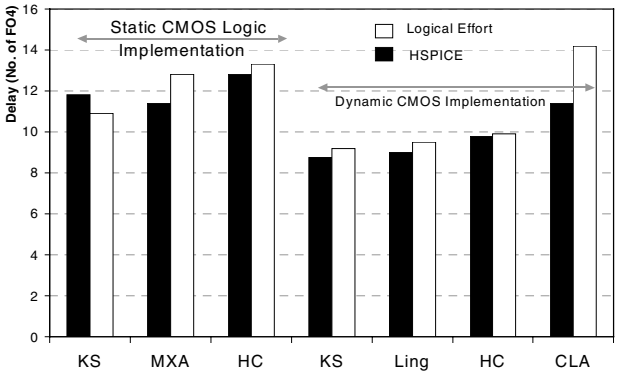

Fig. 2. Speed estimation of various VLSI adders using Logical Effort vs. H-SPICE results

The results obtained using Logical Effort were compared with the results obtained using H-SPICE simulation. The comparison results are shown in Fig. 2. Wire delays were accounted for by estimating the length of the wire and assigning appropriate delay to it, however, the portion of the wire delay was not significant (less than 10% of the total delay) due to the proximity of the cells. The first obvious observation is that there is a huge difference between Static CMOS and Dynamic CMOS implementations. This demonstrates the dependency on logic design style, which obscures any differences between different VLSI adders. This fact has been known by practitioners and rarely would we see a Static CMOS adder in places where high-speed is required. The prediction error is under 10% in most cases.

We are still in the process of refining the LE calculation in order to gain better accuracy. However, our objective is to have a simple "back of the envelope" method for quick estimation and evaluation of different VLSI adder topologies without venturing into CAD tool complexity. Therefore, we compromised by using MS-Excel as a tool for comparison, because of its simplicity and ability to perform complex calculations. An example of the Excel tool used is shown in Table 3 (CM – represents Carry-Merge cell, Dk1ND2 – represents a "footed" dynamic NAND). Before the analysis, it is necessary to characterize the technology used. This step needs to be done only once, but it improves the accuracy of the LE since the characteristics of the technology are taken into account.

Table 2: Normalized LE parameters
0.10µm technology, FO4=19pS

Gate type	LE (g)	Parasitics (p inv)
Inverter	1	1
Dyn. Nand	0.6	1.34
Dyn. CM	0.6	1.62
Dyn. CM-4N	1	3.71
Static CM	1.48	2.53
Mux	1.68	2.93
XOR	1.69	2.97

Table 3: Delay Comparison of Static and Dynamic implementation of Kogge-Stone Prefix-2 Adder

Prefix-2 Kogge-Stone (Static)

Stages	Bit Span	Branch Effort (b_i)	LE (g_i)	Parasitic (p_i)	Total Branch (B)	Total LE (G)	Path Effort (F)	Fopt (f)	Effort Delay (ps)	Parasitic Delay (ps)	Wire Delay (ps)	Total Delay (ps)	Total Delay (FO4)
g0 (NAND2)	0	2.0	1.11	1.84									
C0 (OAI)	2	2.2	1.55	2.26									
C2 (AOI)	4	2.4	1.52	2.76									
C6 (OAI)	8	2.8	1.55	2.26									
C14 (AOI)	16	3.6	1.52	2.76	1.66E+03	2.26E+01	3.76E+04	3.22	106	88	14	**209**	**11.0**
C30 (OAI)	32	5.2	1.55	2.26									
C62 (AOI)	0	1.0	1.52	2.76									
S63 (TGXORs)	0	1.0	1.56	2.59									
INV (INV)	0	3.0	1.00	1.00									

Prefix-2 Kogge-Stone (Dynamic)

Stages	Bit Span	Branch Effort (b_i)	LE (g_i)	Parasitic (p_i)	Total Branch (B)	Total LE (G)	Path Effort (F)	Fopt (f)	Effort Delay (ps)	Parasitic Delay (ps)	Wire Delay (ps)	Total Delay (ps)	Total Delay (FO4)
g0 (Dk1ND2)	0	2.0	1.02	1.34									
C0 (OAI)	2	2.2	1.36	1.69									
C2 (DAOI)	4	2.4	0.68	1.33									
C6 (OAI)	8	2.8	1.36	1.69									
C14 (DAOI)	16	3.6	0.68	1.33	1.66E+03	1.26E+00	2.09E+03	2.34	77	60	14	**151**	**8.0**
C30 (OAI)	32	5.2	1.36	1.69									
C62 (DAOI)	0	1.0	0.68	1.33									
S63 (TGXORs)	0	1.0	1.56	2.59									
INV (INV)	0	3.0	1.00	1.00									

Characterization is performed using SPICE simulation of the gate delay for various output loads driving a copy of itself, according to the LE rules. This is repeated for each cell used in the logic library. Characterization of dynamic gates requires special attention due to the fact that only one transition is of interest. Obtained results are compared to that of an inverter and parameters such as parasitic delay (p) and effort (g) were normalized with respect to that of an inverter. Select results are shown in Table 2.This step preserves LE features, allowing delay results to be presented in terms of fan-out of 4 (FO4) delay, relatively independent of the technology of implementation. The LE-based delay estimation tool works on the logic stages in the critical path, assigning branch effort (b_i), logical effort (g_i) and parasitic effort (p_i) to each gate (Table 3). In computing the branch effort, we take into consideration the worst-case interconnect at each stage. In a 64-bit Kogge-Stone adder, the worst-case interconnect in stage 6 (CM C30) spans 32 bit-slices. We make an assumption that the adder bit-pitch is 10um, which would result in a 320µm wire. To account for the propagation delay through a wire we incorporate an Elmore delay model in Table 3, which corresponds to the critical-path interconnect delay in the adder. A comparison of representative VLSI adders implemented in static and dynamic CMOS design style is presented in Table 4. It is interesting to see that there are indeed very small speed differences between the three fastest dynamic adders: KS, HC and Quarternary (QT). The advantage of KS is achieved by reduced parasitic delays resulting from fewer stages. It is also very difficult to determine the fastest VLSI adder from the results presented in Table 4.

Table 4. Comparison of representative VLSI adders using Logical Effort (wire delay estimate included)

Adders	Stages	Total Branch (B)	Total LE (G)	Path Effort (F)	f_{opt}	Effort Delay (pS)	Parasitic Delay (pS)	Wire Delay (pS)	Total Delay (pS)	Total Delay (FO4)
Static MXA	15	11600	0.369	4280	1.75	96	93	14	203	10.7
Static KS	9	1660	22.6	37600	3.22	106	88	14	209	11
Static HC	10	1660	22.6	37600	2.87	105	92	14	212	11.1
Dynamic KS	9	1660	1.26	2090	2.34	77	60	14	151	8.0
Dynamic HC	10	1660	1.26	2090	2.15	79	64	14	157	8.26
Dynamic QT	10	1540	2.08	3220	2.24	82	68	8	158	8.3
Dynamic LNG	10	1430	0.973	1400	2.06	76	70	15	161	8.47
Dynamic CLA	14	20600	0.627	12900	1.97	101	81	12	195	10.26

The differences between presented topologies are small and fall within the margin of error introduced by inaccuracy of the estimation method. This further emphasizes difficulties in comparing VLSI adder topologies and determining the best one.

3. Energy-Delay Tradeoffs

Comparing VLSI adders becomes more difficult when the notion of power or energy used for computation is introduced. Suppose that an adder A is compared with an adder B and that adder A is faster than adder B. Based on speed only, our inclination would be to use adder A in our design. However, this is not the complete picture. If the energy consumed is considered, and if adder B turns out to be using less energy, we may chose adder B, depending on the power requirements imposed on our design. However, power (energy) can be traded for speed and vice versa. Fig. 3 illustrates the hypothetical energy-delay dependencies of adders A and B.

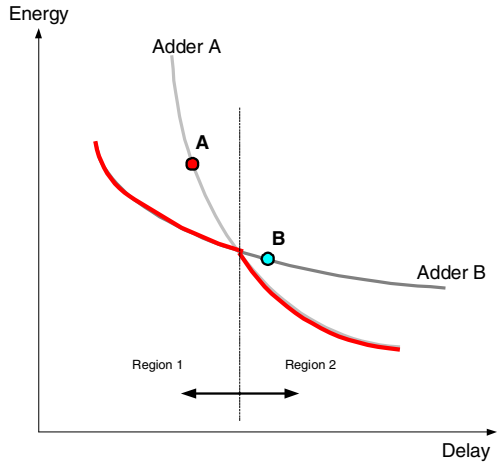

Fig. 3. Energy-Delay dependency

In this example, adder A would be chosen as a better adder topology if we were just to compare two design points A and B with respect to speed and disregard the energy aspect. As the curves show, adder B has more room for improvement and with further energy-delay optimization this adder would move to the point where better performance is obtained by using the adder topology B than topology A (Region 1). However, if low-energy of operation is our objective, we see that adder topology A is better because it can achieve lower energy for the same delay (Region 2). Thus, Fig. 3 illustrates the importance of taking the energy into account, not just merely the speed of the adder.

3.1. Energy Estimation

Logical Effort method does include an estimation of energy. It only provides one point on the Energy-Delay curve corresponding to a sizing optimized for speed. Where this point lies on the Energy-Delay curve remains an unknown. In order to generate Energy-Delay estimates it is first necessary to include some way of estimating energy into the Logical Effort method. We start by characterizing each cell in terms of energy. The energy depends on at least two parameters: output load and cell size. Energy of a two-input NAND gate as a function of its size and fan-out load is shown in Fig. 4

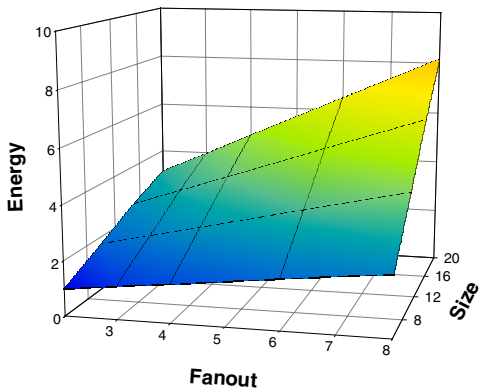

Fig. 4. Energy dependency of a 2-input NAND cell

Thus, the energy estimate will depend on the sizing determined by logical effort as well as the fan-out load on the output of the cell. Each cell used in the design is characterized and parameters determining the energy dependency on the cell size and fan-out load were stored in the table from which dependency parameters were determined. The method of logical effort simplifies the energy calculations as it roughly equalizes the input and output slopes of each gate. This implies that for a given output load and gate size, which defines the output slope, there is only one input slope that is possible. Although it is true that parasitics and unequal stage effort due to wiring will result in some variations in the slopes, it provides enough accuracy for the analysis being performed.

Optimal sizing for speed is determined for each adder by using a modified logical effort methodology. From the information about the size and topology of an adder, the energy consumption is determined. We report worst-case energy for the estimates, defined as the energy consumed when every internal node is switching.

3.2. Sizing

In general, producing various points on the Energy-Delay curve poses a sizing problem. Assuming that we start from some given size - e.g. minimal, Logical Effort should give us an answer to how the circuit blocks should be sized to achieve minimal delay. From the assigned sizes, we can calculate the energy that will be consumed by the adder for a given input activity. However, this is just one point on the Energy-Delay curve. From this point we can move in both directions; toward smaller and toward larger sizes. Such an Energy-Delay curve, produced for two 64-bit implementations of KS and HC adders is shown in Fig. 5.

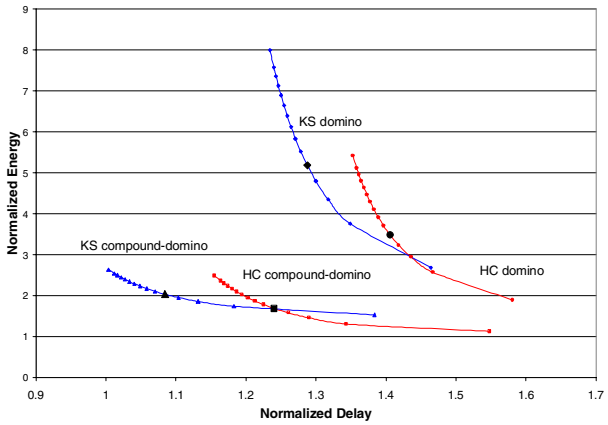

Fig. 5. Estimated Energy-Delay dependency of 64-bit Prefix-2 KS and HC Adders for Domino and Compound-Domino implementations

In order to stay on the same curve we vary the size of the inputs by assigning different values to the input capacitance C_{in}, or by assigning different values to H (electrical effort). This results in different sizes (all optimal in terms of speed) determined by LE methodology using different energies. The adders were implemented as regular domino and compound-domino. The single points obtained when using the same input size for each adder implementation are shown in Fig. 5. The difference between regular domino and compound-domino is in the Carry-Merge stage. Given that dynamic CMOS Domino logic is used for implementation of Carry-Merge blocks as in Fig. 6.

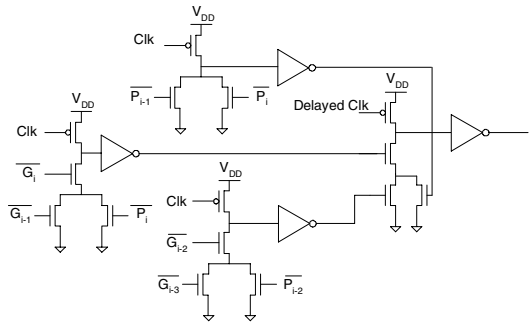

Fig. 6. Carry-Merge: Regular Domino Implementation

It has been realized that the inverter, which is necessary in the CMOS Domino logic block, can be replaced with a static AND-NOR gate, referred to as compound-domino. Thus, two domino blocks are merged into one with the advantage that an additional function is achieved by replacing the inverter. This is shown in Fig. 7.

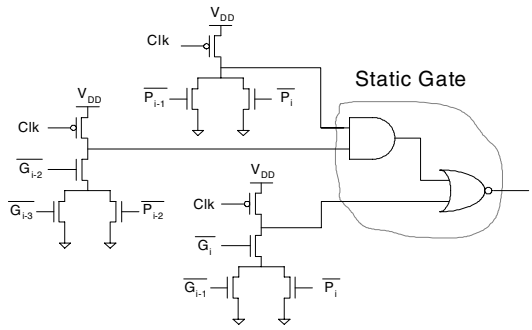

Fig. 7. Carry-Merge: Compound-Domino Implementation

When using compound-domino circuits for adders, it is important to note that the dynamic and the static outputs are potentially attached to long wires, while in domino only the inverter outputs are attached to long wires. Note that “footless domino”, i.e. a circuit where the bottom transistor is eliminated, is used. A 64-bit critical

path contains 6 carry-merge stages; none of which contain a stack of more than two n or p transistors. A critical path in the HC adder is shown in Fig. 8 [9]. It contains one more stage in the critical path, but it eliminates approximately one half of the blocks in the carry-merge tree, thus bringing some potential energy advantages over a KS implementation.

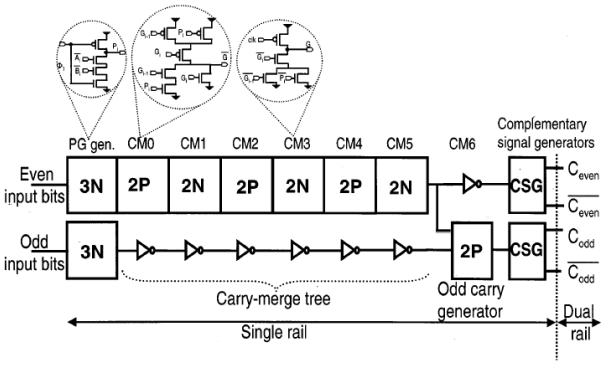

Fig. 8. Critical Path in the 64-bit Han-Carlson adder [9]

The energy-delay chart in Fig. 5 shows domino HC having an advantage over domino KS in terms of lower energy, but domino KS can stretch further in terms of lower delay at the expense of increased energy. When comparing compound-domino implementations, both HC and KS result in lower energy than the regular domino implementations. If one is concerned about energy, HC is better. However, as the speed becomes increasingly important the advantage moves in favor of KS. Both schemes suffer from high gate counts, large transistor sizes, and excessive wiring, resulting in large layout areas and high-energy consumption. In an attempt to arrive at a denser design, a Ladner-Fischer adder described in [1,11] trades off wiring complexity by exponentially growing the fan-outs of successive carry-merge gates to 1,2,4,8 and 16 respectively. However, this does not address the problem of high gate counts and large-transistor sizes. The idea of reducing energy by breaking the scheme into 4-bit blocks, which are conditionally added, was further developed by Intel [12,13]. It was realized that pruning down the main tree to obtain a sparse-tree that propagates P and G signals for 4-bit sections of the adder and combining this with a modest increase in hardware complexity due to the conditional sum technique, might be a good energy-delay trade-off. A 32-bit Quaternary-Tree (QT) adder core as described in [12], consists of a sparse-tree that generates 1 in 4 carries (Fig. 9) and a parallel side-path of 4-bit conditional sum blocks, resulting in an 8-stage design (same number of stages as KS).

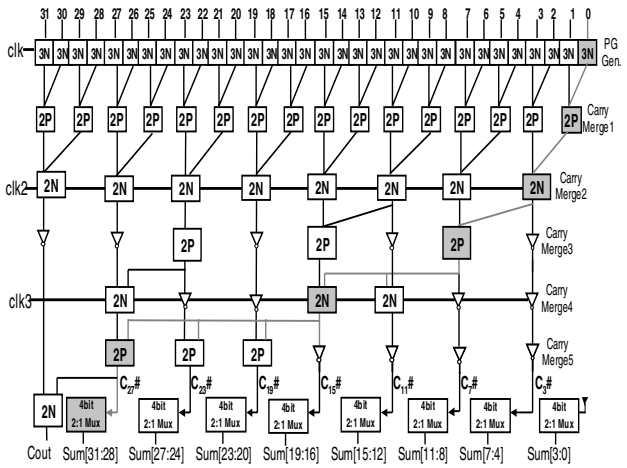

Fig. 9. 32-bit Quaternary-tree Adder Core [12]

Carries generated by the sparse-tree select the final sum using a 2:1 multiplexer. QT adder achieves energy reduction due to two factors: (a) reduced fan-outs and reduced wiring as compared to KS and HC structure (resulting in smaller transistor sizes) and (b) use of non-critical ripple-carry conditional sum blocks (with much smaller transistor sizes than main tree). In contrast to a Ladner-Fischer design, the QT adder tree has unit fan-outs on all generate gates, except the 3 highlighted gates in the Carry-Merge3, Carry-Merge4 and Carry-Merge5 stages. These 3 gates have fan-outs of 2,3 and 4 respectively (In both cases, we ignore the presence of the buffering inverters at the LSBs). Parallel-prefix logic removed from the main-tree is performed in the conditional sum blocks using an energy-efficient ripple-carry scheme with smaller transistor sizes. The structure of conditional sum blocks is shown in Fig. 10.

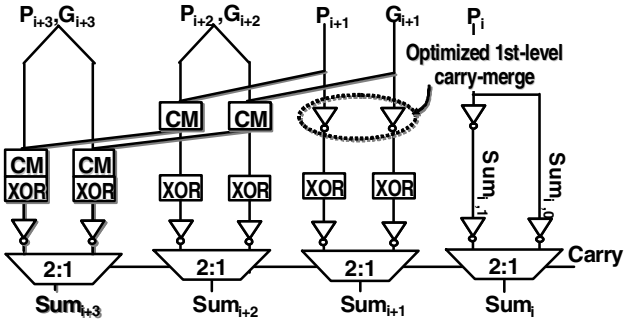

Fig. 10. Quaternary-tree Adder: Conditional Sum [12]

A 64-bit QT adder (Fig. 11) has 2 levels of conditional carry generation with the main tree generating 1 in 16 carries, which select betwwen the 1 in 4 conditional carries generated by the intermediate generator. The 1 in 4 carries then choose the appropriate 4-bit conditonal sum. Such a design has 10 stages (same as HC).

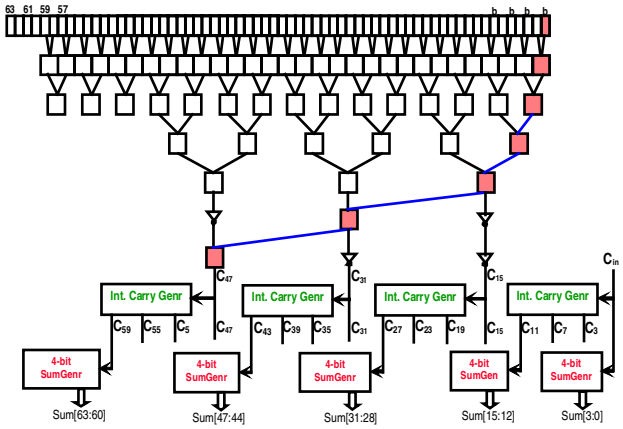

Fig. 11. 64-bit Quaternary-tree Adder Core

4. Estimation Method

The goal of the estimation method is to provide a simple, yet accurate, method for comparing designs in the energy-delay space. Logical effort has been shown to yield reasonably accurate results for the parallel prefix adders that were tested, thus finding it suitable for delay estimation and sizing. One of the issues with Logical Effort is branching, specifically with regards to long internal wires and different number of stages in the off path. These effects are difficult to account for and accurate accounting makes LE complex, thus simplifications are required to make the analysis linear. A more detailed account of these issues would result in greater delay accuracy when applying LE to adders.

4.1 Energy-Delay Curve Estimation

As previously described, the different points on the energy-delay curve are obtained by varying the size of the input gate for each adder. This provides a delay estimate and the sizing of the critical path, which in turn gives an energy estimate for the gates on the critical path. However, this estimation does not provide an energy estimate for the entire adder. The estimated energy for the critical paths is shown in Fig.12. From these estimates, it can be seen that the critical path of KS uses less energy for a given delay then HC. This is explained by the one extra stage that HC uses. However as mentioned previously, HC uses approximately half of the gates in the CM-section than KS. This must be accounted for in the total energy estimate for the adders. The critical path energy-delay estimate provides an idea of the minimal delay that can be achieved, which is shown by the vertical asymptote of the curves in Fig.12. Since we are interested in comparing adders not only by delay, but also in the energy-delay space, we need to determine a method for estimating the energy of an adder. By determining the number of gates per stage and assuming the same energy for those gates as the corresponding energy of the critical path gate in the same stage, we were able to obtain reasonably accurate results (within 20%). This provided a simple method to apply to most parallel prefix designs, however for a design like QT where the number of gates on each path is different, more care must be taken in determining what energy the off-path circuits consume.

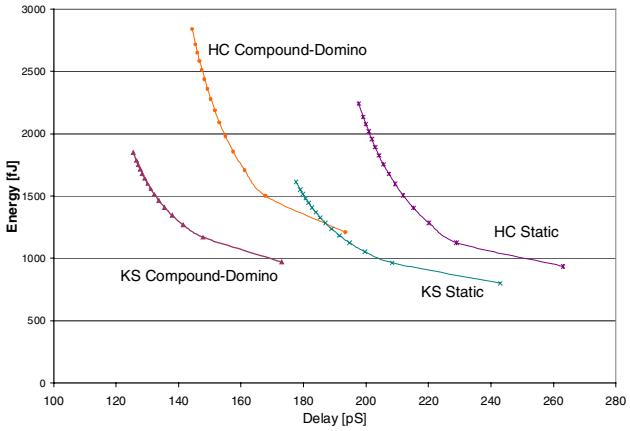

Fig. 12. Energy Delay comparison of 64-bit KS and HC Compound-Domino and Static adders

5. Results

The accuracy of our energy-delay estimation was tested on 32-bit KS and QT compound-domino adder implementations, with comparison to simulation results in 0.13µm technology [12]. Each of the energy-delay simulation points was obtained using a circuit optimizer. The energy reported was obtained by running the worst-case energy vector for each topology. Since the simulated data points were in 0.13µm technology, we needed to extrapolate the results and normalize them to the same technology as our estimates (0.10µm). We used a rule of thumb of 30% performance improvement per generation as well as a 50% energy improvement. The energy-delay estimates were obtained by varying the size of the input gate to the adder, as previously described. The estimated energy assumes an estimated worst-case switching activity associated with the topology of the adder and the logic design style. A more accurate method would require detailed modeling of the switching activity within each adder core, which is not possible prior to implementation. The comparison of the estimated energy-delay versus simulated is shown in Fig. 13. At iso-delay the simulations show 55% difference in energy at the knee of the curves, while estimation shows 35% difference. At iso-energy, the simulations show 21% delay improvement at the knee of the curves, while estimation shows 13%.

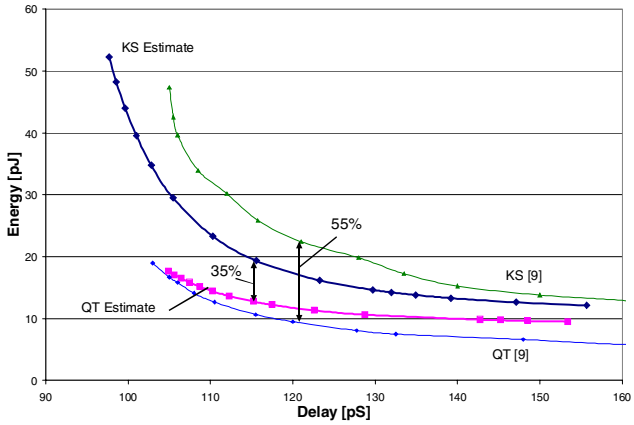

Fig. 13. Energy-Delay comparison of 32-bit QT and KS adders: estimated vs. simulation in 0.10µm technology

As the intent of this method is to compare the energy-delay characteristics of a given design, this comparison shows the estimation accurately represents tradeoffs in the energy-delay space for different architectures. To further explore the energy-delay design space we analyzed 64-bit KS, HC and QT compound-domino and static topologies, shown in Fig. 14.

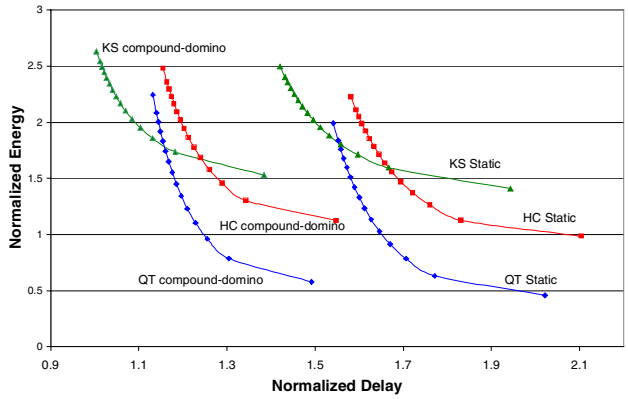

Fig. 14. Energy-Delay comparison of 64-bit KS, HC and QT adders

The 64-bit QT implementation used has one extra stage than the KS, which accounts for the higher performance that can be achieved by KS for both compound-domino and static topologies. The difference in energy for iso-delay is more significant in the 64-bit design space. This is due to the increased number of stages over which the delay is evenly distributed, resulting in increased gate sizes, and increased number of gates, e.g. a 32-bit KS has 417 gates, while a 64-bit KS has 1025 gates. Thus in the 64-bit design space, energy efficient designs display greater savings, which explains why QT uses substantially less energy than either KS or HC. The QT compound-domino design achieves an 86% reduction in energy vs. HC compound-domino at the knee of the curve.

6. Conclusion

We have shown that an LE based analysis of logical circuits is an effective tool for a quick exploration of the energy-delay space for comparing the performance of high-performance adders. Further, a tool was developed based on this technique to quickly estimate the energy-delay space of 32/64-bit Kogge-Stone, Han-Carlson, and Quaternary-tree adders implemented in 0.13µm and 0.10µm CMOS technologies using static and dynamic circuit styles, thereby, accurately representing tradeoffs in the energy-delay space with an accuracy of 8% in delay estimates and 20% in energy estimates, compared with simulated data.

References

[1] S. Knowles, "A Family of Adders", Proceedings of the 14th Symposium on Computer Arithmetic, 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] R. F. Sproull, and I. E. Sutherland, "Logical Effort: Designing for Speed on the Back of an Envelop," IEEE Adv. Research in VLSI, C. Sequin (editor), MIT Press, 1991.
[4] D. Harris, R.F. Sproull, and I.E. Sutherland, "Logical Effort Designing Fast CMOS Circuits," Morgan Kaufmann Pub., 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] P.M. Kogge and H.S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations", IEEE Trans. on Comp. Vol. C-22, No.8, Aug. 1973, pp.786-793.
[7] A. Farooqui, V. G. Oklobdzija, F. Chehrazi, "Multiplexer Based Adder for Media Signal Processing", International Symp on VLSI Technology, Systems, and Applications, 1999
[8] 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.
[9] S.K. Mathew et al, "Sub-500-ps 64-b ALUs in 0.18µm SOI/bulk CMOS: design and scaling trends," IEEE Journal of Solid-State Circuits, vol.36, Nov. 2001, pp.1636-46.
[10] H. Ling, "High Speed Binary Adder", IBM Journal of Research and Development, Vol. 25, No 3, 1981, p.156-166.
[11] R.E. Laddner and M.J. Fischer, "Parallel prefix computation", Journal of ACM, Vol.27, No.4, 1980, pp.831-38.
[12] S.K. Mathew et al, "A 4GHz 130nm Address Generation Unit with 32-bit Sparse-tree Adder Core," 2002 Symposium on VLSI Circuits Digest of Technical Papers, pp.126-127.
[13] J. Sklansky, "Conditional-sum addition logic." IRE Transactions on Electronic Computers, vol.9, 1960, pp. 226-231.