Post-Layout Comparison of High Performance 64b Static Adders in Energy-Delay Space

Sheng Sun
Advanced Micro Devices, Inc.
Sunnyvale, CA 94088
sheng.sun@amd.com

Carl Sechen
EE Dept, University of Texas at Dallas
Richardson, TX 75083
carl.sechen@utdallas.edu

Abstract

Our objective was to determine the most energy efficient 64b static CMOS adder architecture, for a range of high-performance delay targets. We examine extensively carry-lookahead (CLA) and carry-select adders with a wide range of tradeoffs in logic levels, fanouts and wiring complexity. We propose sparse CLA adder architectures based on buffering techniques to reduce logic redundancy and improve energy efficiency. All the designs were implemented using an energy-delay layout optimization flow with full RC extraction. Our new 64b adder designs have a relative delay as low as 9.9 FO4 (fanout-of-four inverter) delays and promise better scaling for smaller technology nodes. They yield the best energy efficiency for a wide range of delay targets and are 30%, 15% and 7% more energy efficient than full Kogge-Stone, sparse-2 Kogge-Stone and Han-Carlson, respectively, at the fastest points. They consume only about 1/3 the energy of dynamic adders.

1. Introduction

A 64b adder is an essential component of state-of-the-art high-performance microprocessors. Traditionally, dynamic circuits were used to achieve best performance [23][24]. However, the dominant concern nowadays is energy efficiency and robustness. Static CMOS circuits have recently gained renewed interest for high performance applications [19]. A static CMOS design reduces power considerably since it does not consume any clock power and has low switching activity.

Carry lookahead (CLA) [4] and carry select [3] are the most popular high performance addition techniques [1]. Brent-Kung [5], Kogge-Stone [7] and Ladner-Fischer [6] are three basic CLA tree structures. The most promising high performance architectures are those with nearly minimal logic depth. Brent-Kung has large logic depth due to back propagation. The Knowles family of adders [9], with Kogge-Stone and Ladner-Fisher as the two end cases, all have minimum logic depth. Logic depth is reduced either by replicating carry tree as in Kogge-Stone (with large logic redundancy), or by increasing tree node fanouts as in Ladner-Fischer. Han-Carlson (HC) [8] is a sparser variant of Kogge-Stone and has one extra level. Ling's equations [10] can eliminate a preliminary level. Carry select can be combined with the CLA tree to achieve a sparser CLA tree, probably reduced fanouts and reduced logic levels.

This work explores the tradeoffs of high performance tree adders and their hybrids. We also propose sparse CLA architectures based on buffering techniques to reduce logic redundancy and improve energy efficiency. This work aims to demonstrate the best architectural candidates in static CMOS for a range of energy and delay (E-D) targets.

Numerous publications have compared major adder architectures [9][13][14]. However, they tend to use simplified models or have done layouts on a limited number of architectures (and only for a single delay point). To evaluate many adder architectures for VLSI design in energy-delay space, we utilized the automatic layout flow in [16], which seeks to optimally minimize energy via cell sizing for a specific delay target.

2. 64b adder architectures

We focus on CLA tree adders, and hybrids with carry select, classified as the two basic types in Fig.1. For one type, the CLA tree generates a carry at each bit, and finally an XOR or XNOR gate is used to obtain the sum. For the other type, the CLA tree only generates a carry for each sub-block, and then the carry is used to select one of the two pre-computed sum blocks. For the hybrid structure, sparseness 2 or 4 (with carries generated every other bit or every 4 bits) is normally appropriate for static CMOS designs. Non-critical presums (4b for sparseness 4) are computed in ripple carry fashion using downsized gates for energy efficiency.

1-4244-1258-7/07/$25.00 ©2007 IEEE

Radix-2 (carry-merge of 2 bits) is most suitable for static designs [1]. For a 64b radix-2 adder, the minimum logic depth is $1+\log_2 64+1=8$, which is the case for Kogge-Stone, Ladner-Fisher, all other Knowles adders and their sparse hybrids. Han-Carlson has 9 levels. By applying Ling's equations [10][25], we can eliminate one level. We will first present a range of basic architectures that have near minimum logic depth. Then we propose adder architectures based on buffering techniques to reduce logic redundancy and improve energy efficiency. We also discuss Brent-Kung variants that have more logic levels. We have a good diversity of architectures, with various tradeoffs of logic levels, fanouts and wiring flux.

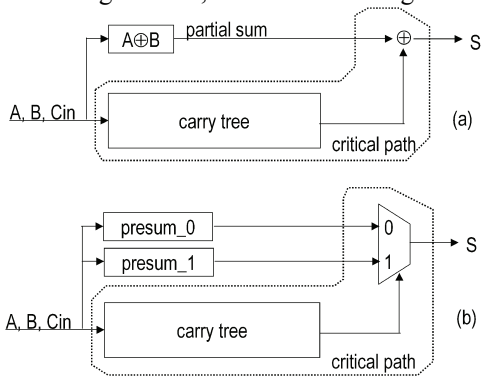


Fig.1. General structures of two types of CLA adders (a) pure CLA without carry select (b) Hybrid sparse CLA with carry select

Unless otherwise noted, all the architectures implement equation (1) for preliminary single-bit generate and propagate and (2) for group GP logic, represented by symbols in Fig. 2. The figure shows non-inverting logic. But actual carry-merge cells are implemented as CMOS inverting gates as (3)-(4), alternating along logic stages.

$$g_i = A_i \bullet B_i,\ p_i = A_i + B_i \quad (1)$$

$$G_{i:j} = G_{i:k} + P_{i:k} \bullet G_{k-1:j},\ P_{i:j} = P_{i:k} \bullet P_{k-1:j},\ i \geq k > j \quad (2)$$

$$\overline{G_{i:j}} = \overline{G_{i:k} + P_{i:k} \bullet G_{k-1:j}},\ \overline{P_{i:j}} = \overline{P_{i:k} \bullet P_{k-1:j}}; \quad (3)$$

$$G_{i:j} = \overline{\overline{G_{i:k}} \bullet (\overline{P_{i:k}} + \overline{G_{k-1:j}})},\ P_{i:j} = \overline{\overline{P_{i:k}} + \overline{P_{k-1:j}}} \quad (4)$$

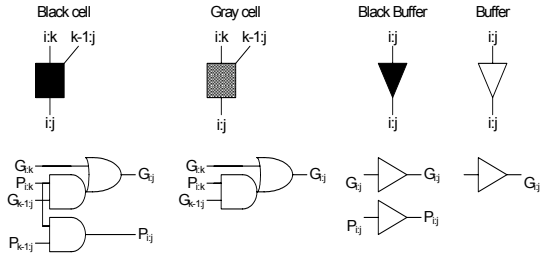


Fig. 2. Group GP cells

Our adder implementations use only maximum-two-stack gates, including inverter, NAND2, NOR2, AOI21, and OAI21 complementary static gates. Pass-transistor gates are also used extensively for their efficiency in implementing XOR/XNOR and MUX functions, essential for adder sum calculation and carry select. From now on, we will only show the critical CLA trees.

2.1. Basic Architectures with Near Minimum Logic Depths

For clarity, only shown are the wirings for the most significant and the least significant merge cells, unless otherwise noted. The intermediate wirings in between are the same but with shifted positions. Bit "-1" is adder carry-in. Lateral fanout is the fanout to higher bit positions, rather than the same bit position.

Full Kogge-Stone [7]. The full Kogge-Stone has fanout of 2, almost uniformly. It may be the fastest at schematic level. But with its large number of cells (with logic redundancy) and wires, it may not be very fast with extracted RC parasitics, and certainly not energy efficient.

Knowles [2,2,1,1,1,1] (Fig. 3). The lateral fanouts are 2 for L5 (logic level 5) and L6. For L5 and L6, only lateral fanouts are shown. L1 to L4 are the same as for the full Kogge-Stone. The idempotent property [9] is used. For instance, the two sub-trees of L6:61 have overlap at bit 46. Compared to full Kogge-Stone, it has more fanout on the critical paths at L5 and L6, but may only have a minor delay penalty due to the significance of long wire load (relative to the load of gate fanouts). There is only half the number of long wires, which improves routing congestion and promises better routing quality. The odd bits of L5 and L6 do not have any lateral fanout and become less critical, causing the total number of critical paths to be reduced, which promises lower energy. It has almost the same number of cells as the full Kogge-Stone approach.

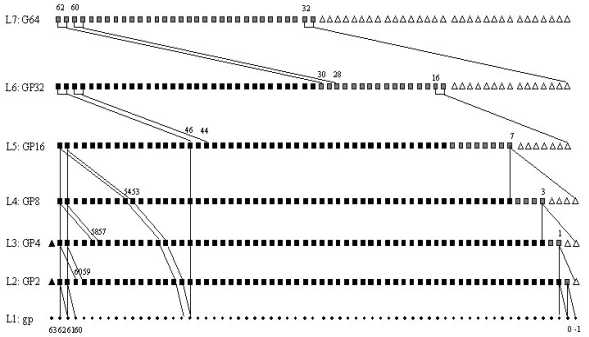


Fig. 3. Knowles [2,2,1,1,1,1]

Kogge-Stone Sparseness 2 and Han-Carlson. Based on full Kogge-Stone, the two architectures have only half of the carry-merge nodes. The difference between sparseness 2 (SP2) and Han-Carlson (HC) is that the former uses carry select but the latter uses one extra level to generate carries for odd bits. SP2 has more cells than HC due to pre-computed sums. Both are often regarded as energy efficient fast designs. We also applied Ling's approach to SP2 to save one level (referred to as KS_SP2_Ling). KS_SP2_Ling shares the same CLA tree as SP2 except the first level, and uses different sum equations [10][25].

Kogge-Stone Sparseness 4 (Fig. 4). The global carry merge tree contains only about ¼ of the full KS tree (except L2). Carry-select does not reduce node fanouts of the Kogge-Stone type of tree structures, but the sparseness may provide energy efficiency.

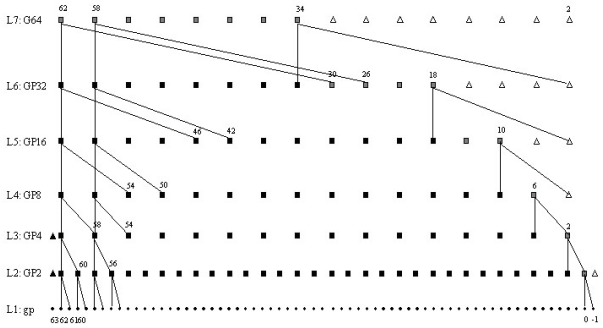


Fig. 4. Kogge-Stone sparseness 4

Sparse Knowles [8,4,2,1,1,1] (Fig. 5). We obtain this sparse tree from the other end of the Knowles prefix adders -- the Ladner-Fischer tree. Only lateral wirings are shown. The original 64b Ladner-Fischer tree has lateral fanouts of [32,16,8,4,2,1]. The sparse tree has fanouts reduced by 4X to [8,4,2,1,1,1]. We refer to this structure as Knowles [8,4,2,1,1,1] (shortened as KN8421). Compared to Kogge-Stone sparseness 4, it has more fanout, but reduced wires for L3 to L6. It has fewer full carry-merge cells (replaced by buffers) due to the lack of logic redundancy.

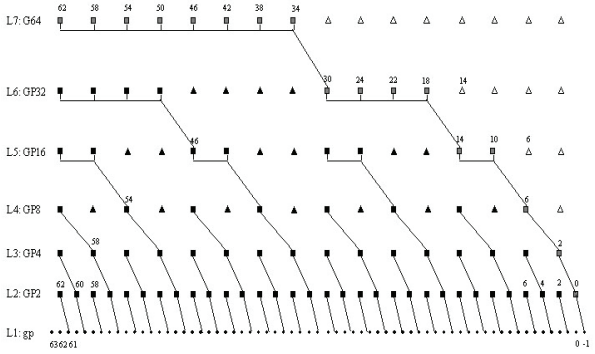


Fig. 5. Sparse Knowles [8,4,2,1,1,1]

The fanouts are still too large for fast implementation. The design in [15] employs a similar architecture but uses two extra levels to buffer large FO. Next we show our approaches to mitigate the effective fanout.

2.2. New Architectures with Buffering

Buffering is normally used to drive large loads, either due to a large number of fanout gates, or due to long wires. We may classify the buffering techniques into three types: 1) Buffers are inserted between drivers and their large loads, as in Fig. 6(a). This approach increases logic levels. 2) Make multiple copies of the driver, as in Fig. 6(b). The replica has been termed a "helper" in [12]. The helper essentially sizes up the driver. 3) Reduce the load to the critical path by using a "reducer", as in Fig. 6(c). The reducer isolates the non-critical loads from the driver. These three techniques can be combined in practice.

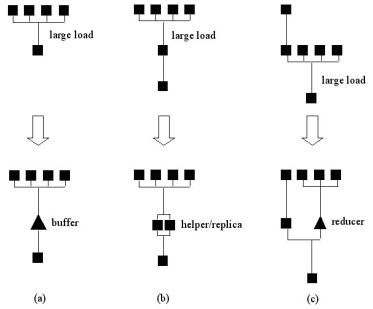


Fig. 6. Buffering: (a) normal (b) helper/replica (c) reducer

Based on the sparse Knowles [8,4,2,1,1,1] architecture, we propose two improved architectures that include specialized buffering. In Fig. 7, the maximum allowable equivalent fanout (FO) is set to about 3 (assuming the load of a buffer/inverter is relatively small). Bit 30 of L6 uses helpers with 3 replicas to drive 8 gate loads and long wires. Non-critical bits (24,22,18) use a reducer at L6:14. For L6 critical bits (58,54,50), a helper was used at bit 46 of L5. We refer to this structure as KN8421_FO3.

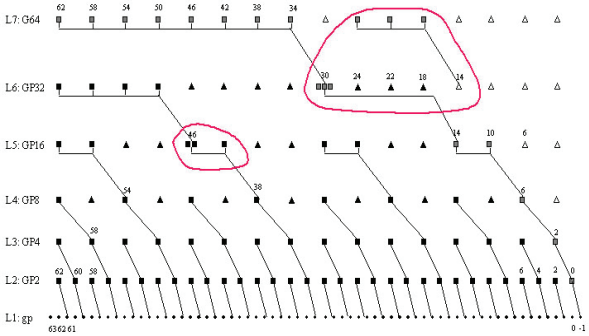


Fig. 7. New buffered sparse KN8421_FO3

Similarly in Fig. 8, the maximum allowable equivalent FO is set to about 2. The carries at L7 drive four fanouts for 4b carry select. However, its L2 stage has only one fanout compared to two in the KS2_SP2 or HanCarlson structures. We also experimented with a buffering level at L7 to drive fanouts of 4. Therefore, by using just a few extra buffers/replicas, these two modified architectures reduce critical path fanouts dramatically from the original sparse Knowles [8,4,2,1,1,1]. They promise better performance with energy efficiency.

The fanout of 2 is near optimal for 2-stack static AOI/OAI gates (carry merge cells), with logical effort around 2 and stage effort around 4 [11].

Fig. 8 also employs some techniques that reduce power by about 10%. Some extra buffers unnecessary for speed are removed to save power. In addition, sum bits from 0 to 14 are generated by short ripple adders, which utilize fast carries at bit 0, 2, 6 and 10, respectively. Carry signals on the critical path (-1, 0, 2, 6) are all buffered using reducers to avoid a speed penalty due to extra branch load. We refer to this improved design as KN8421_FO2_LP (which has 8 levels) and the version with an extra buffering level at L7 for carry select as KN8421_FO2_LP2 (which has 9 levels). The same power-saving techniques are also applicable to the KN8421_FO3 version (referred to as KN8421_FO3_LP).

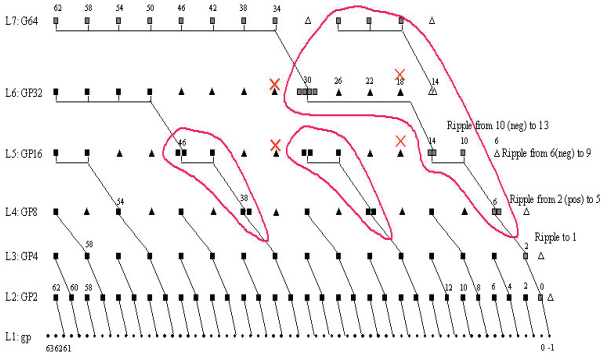


Fig. 8. New buffered sparse KN8421_FO2_LP (LP2)

2.3. Variants of Brent-Kung (More Levels)

So far we have been focusing on architectures with minimum logic depth. It is also worthwhile to explore the influence of larger logic depth. Brent-Kung structure limits fanout to be unity (not counting buffer fanout) and needs many more levels to compute the final carries. The 64b Brent-Kung version has 13 levels (including the final sum level), as shown in Fig. 9. It is not suitable for an ultimate speed requirement. On the other hand, it has much less logic redundancy than the Kogge-Stone type of CLA trees and hence many fewer cells and consumes less energy.

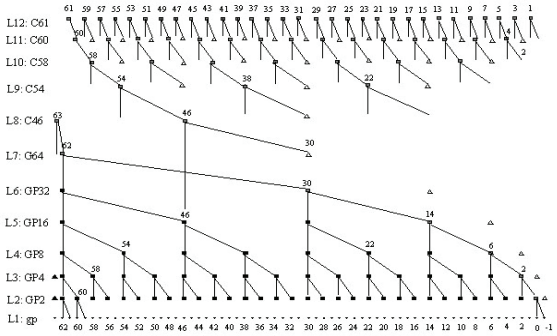


Fig. 9. Original Brent-Kung

We would like to explore logic levels between the Brent-Kung end and those of nearly minimum depth. Fig. 10 is a sparse design with 10 logic levels (final sum level not shown), referred to as BK_L10. Sparseness 4 reduces 2 levels. By eliminating the original bit 30 buffer at L7, we further reduced one level. The lower half of the carry tree is either maintained or uses carry ripple to achieve low energy consumption. Some extra buffers/inverters are needed to maintain signal polarities.

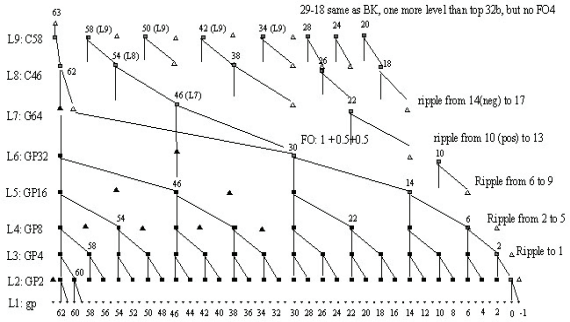


Fig. 10. Brent-Kung sparseness 4, with 10 levels (BK_L10)

By increasing the effective load in L6 and L7, we can reduce further one level (Fig. 11). By using duplication, the maximum effective FO in the tree is about 2. A few buffers are added to act as reducers (e.g., bits 26 and 18 in L4 and L5).

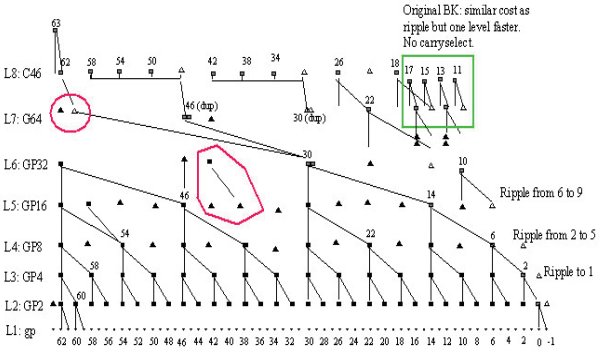


Fig. 11. Brent-Kung sparseness 4, with 9 levels (BK_L9)

We would expect the BK_L9 to be comparable to our previous KN8421_FO2 structures. BK_L9 has 9 levels, one more level than KN8421_FO2_LP, but with slightly smaller load in early carry-merge stages. It has same number of stages as KN8421_FO2_LP2, but the latter has one dedicated buffer level to drive larger carry select load, which could be more efficient.

3. Optimization and implementation flow

1. Gate sizing (AMPS) runs to determine Energy-Delay (E-D) plots (using a wire load model)
2. Select E, D points and execute iTools placement and routing
3. Extract actual 3D wire loads (Assura RCX and Star-RCXT) and determine E, D
4. Re-run sizer for one or more of the layout points
5. iTools ECO place and route
6. Extract 3D actual wire loads and determine E, D
7. If necessary, repeat 4-6 once

Fig. 12. Sizing and layout flow [16]

Fig. 12 outlines the optimization (with sizing) and implementation flow we used [16]. The Synopsys tools AMPS, Pathmill, and Nanosim are used for sizing optimization, transistor-level static timing analysis, and transistor-level energy simulation, respectively. Energy per operation is obtained by averaging over 5000 random vectors. What is unique in this flow is the iTools place and route tool from InternetCAD.com. iTools's great stability eliminates the need of multiple iterations as required by a regular P&R tool and guarantees rapid convergence. Assura parasitic extraction (RCX) from Cadence was used to extract the library cells. Star-RCXT from Synopsys was used for faster top-level full 3D RC extraction of the interconnect.

We used the IBM 0.13um process in this work. Our cell library was optimized for energy efficiency, containing only energy efficient logic gates, each with many drive strengths and beta ratios (with transistor sizes available in the range of 0.28um to 10.92um.)

For the adder design, a structural Verilog netlist is directly fed into the flow. The whole layout flow for a 64 adder design only needs one day to get an optimized E-D curve from full layout extraction, with 10-20 or more energy-delay points.

4. RESULTS AND DISCUSSIONS

Unless otherwise specified, all designs have output load of 30fF; maximum input capacitances are reasonably under control in the range of 30-40fF (for largest/fastest points); input signals are made realistic by being buffered by resistances equivalent to D-flip-flop (DFF) drive strengths; and the FO4 inverter delay is 51ps at 1.5V, using actual (but rather slow) MOSIS fabrication run models for the IBM 0.13um process. We first compare the architectures using energy-delay (E-D) curves. We also tabulate some detailed results later in this section. We focus on the fastest points.

The full Kogge-Stone adder (KS2_full or simply KS) gains several picoseconds at a cost of 17% more energy than the KN221111. At KN221111's fastest point, KS consumes 9.3% more energy at the same speed (Fig. 13). KN221111 is more energy efficient when a tiny delay penalty can be tolerated.

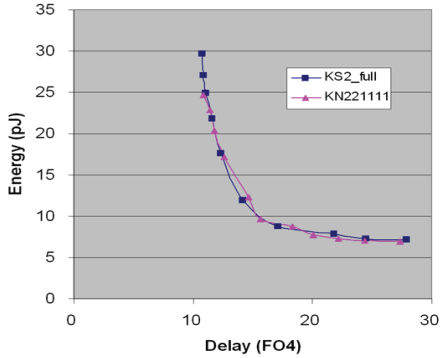


Fig. 13. KS vs. KN221111

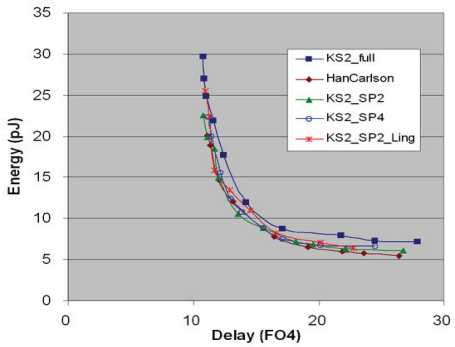


Fig. 14. KS and sparse variants

Fig. 14 compares the full Kogge-Stone with its sparse variants. KS clearly consumes much more energy. HanCarlson (HC) is around 20ps slower but is about 20% more energy efficient than KS for the same speed at its fastest point. KS2_SP2 is only a couple of picoseconds slower than KS with 17% less energy for same speed. KS2_SP2 is faster than HC with more energy. KS2_SP4 has similar energy as KS2_SP2 and similar delay as HC at its fastest point. KS2_SP2_Ling consumes more energy than KS2_SP2 and is slower. Even though it has one fewer level on the critical paths, the first level AOI22 and OAI22 are rather slow and the more complex pre-sum blocks increase energy consumption. We do not see the advantage of the Ling method normally reported in dynamic designs, where more complex gates are favored to reduce logic depth.

Overall, the KS2_SP2 and HC are the best candidates in this group for energy-efficient designs.

Fig. 15 shows that the two versions of KN8421_FO2 have very similar energy and delay characteristics. The FO3 version is clearly slower and less energy efficient.

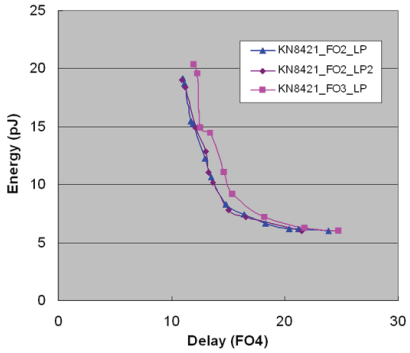


Fig. 15. KN8421 FO2 (both LP and LP2) & FO3

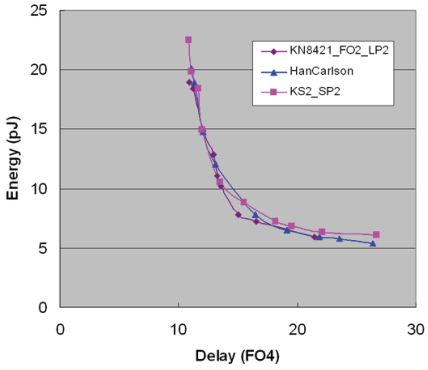


Fig. 16. FO2 vs. HC and SP2

Fig. 16 compares the KN8421_FO2_LP2 with HC and KS2_SP2. Our new design achieves the best speed and energy efficiency for the whole delay range. It has no logic redundancy, more simple cells (buffers) and more cells in non-critical pre-sums. It has fewer long wires and is also better at driving wiring capacitance due to helpers: KN8421_FO2_LP has 4 drivers for a long wire, but HC and KS2_SP2 have only one driver. Our new designs promises better performance for 65nm processes and below (as we observed in recent work), where wire capacitance becomes much more costly.

The Brent-Kung (BK) family provides different tradeoffs (Fig. 17). The original Brent-Kung is a much lower energy design. Carry select structures BK_L9 and BK_L10 achieve faster speed. The L9 version is faster than L10. However, BK_L9 is still worse than KN8421_FO2_LP2 in performance. The results indicate that 8-levels with fanout of 2 seems optimal as in Kogge-Stone and KN8421_FO2_LP (for KN8421_FO2_LP2, the number of levels is 9, but one is an efficient buffering level).

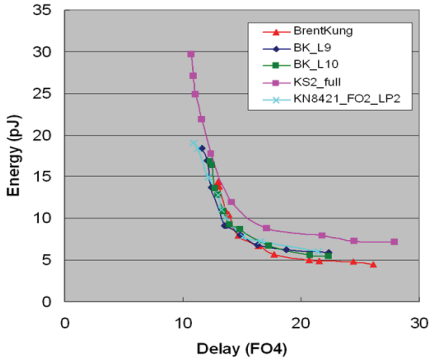


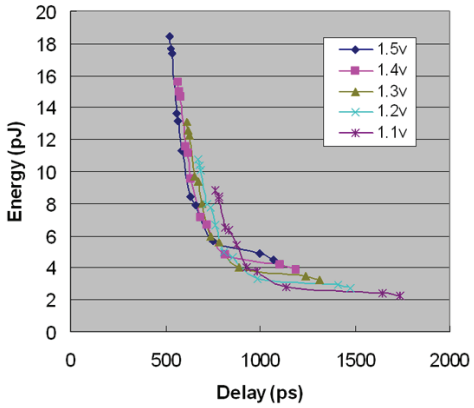
Fig. 17. Brent-Kung variants

Fig. 18 shows how the KN8421_FO2_LP2 design points scale with VDD. (It is not a surprise we found that all architectures have similar scaling characteristics). Fig. 18(a) shows the un-normalized delay (picoseconds) and Fig. 18(b) uses the normalized delay with respect to the inverter FO4 delay.

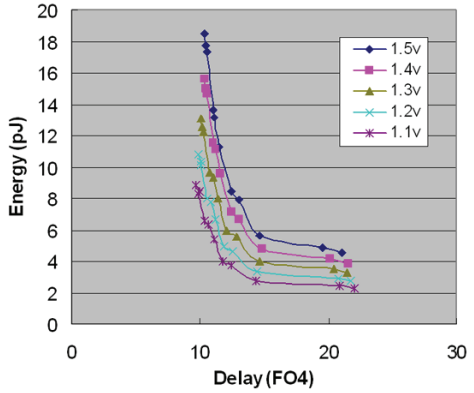
Adder delays follow FO4 inverter delays closely, and the relative delay is almost constant in terms of FO4 when VDD varies. There is a different deviation for the two ends. For the fast points (with large sizes), smaller VDD results in better relative FO4 delays. This is possibly because wire RC delays contribute significantly for large designs and increasing VDD does not provide the same speedup as for pure gate delays (since wire RC delays are not sped-up). On the other hand, for the low power points, the FO4 numbers increase with lower VDD, possibly due to the effect of threshold voltage (V_t) increases for narrow transistor widths (pages 48-50 of [2]). Those small transistors have a larger V_t loss for overdrive (VDD-V_t) and the influence is more significant for lower VDD.

We can observe another point from Fig. 18(a). The highest VDD (1.5V) gives better energy efficiency for fast designs in a quite wide delay range (this conclusion may need correction when leakage becomes more significant for smaller technology nodes). Even though VDD is a strong factor of energy (square rule), the speed-up by upsizing is even more costly. On the other hand for the low power end, low VDD is desired as generally expected.

Fig. 19 shows the wasted current percentage of the KN8421_FO2_LP2, with 11 points (number 1 is the largest/fastest, and 11 is the smallest). In our technology process, the wasted current is mainly short-circuit current (leakage is negligible). The wasted current consumes from 2% to 10% of the energy; the higher the VDD, the larger the percentage. Short-circuit power scales faster than VDD^2 as for capacitive switching power [17].



(a)



(b)

Fig. 18. VDD Scaling (KN8421_FO2_LP2) (FO4 inverter delay is 51ps, 55ps, 61ps, 68ps and 79ps for VDD 1.5v, 1.4v, 1.3v, 1.2v and 1.1v, respectively) (a) Delay in pico-seconds (b) Delay in fanout-of-four inverter delay

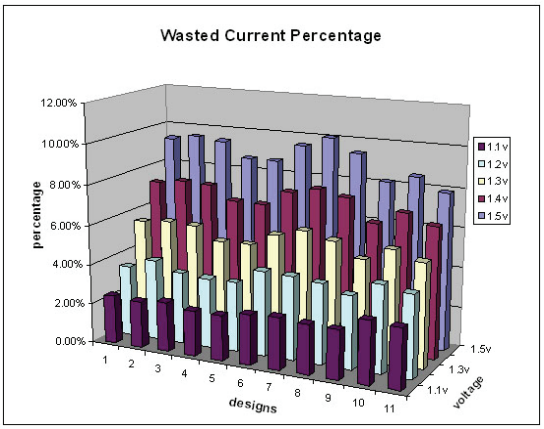


Fig. 19. Wasted current (short-circuit current) percentage (KN8421_FO2_LP2)

Table 1. 64b adder results @ fastest points (load=30fF, 1.5V, FO4 = 51ps. KS_SP2_L: KS_SP2_Ling, FO2_LP: KN8421_FO2_LP, FO2_LP2: KN8421_FO2_LP2; #C: #cells, #L: #levels, E: Energy,EDP:E-Delay Product)

Adder	# C	# L	Delay		E (pJ)	EDP (ns* pJ)	Area (um^2)
			ps	FO4			
KS	904	8	548	10.7	29.7	16.3	34209
			554	10.9	27.0	15.0	32358
KN221111	901	8	554	10.9	24.6	13.6	29074
HC	615	9	566	11.1	20.1	11.4	23950
KS_SP2	677	8	552	10.8	22.5	12.4	26006
KS_SP4	670	8	570	11.2	22.3	12.7	25212
KS_SP2_L	674	7	567	11.1	25.5	14.5	28573
FO2_LP	**681**	**8**	**559**	**11.0**	**19.6**	**11.0**	**23399**
FO2_LP2	**693**	**9**	**558**	**10.9**	**19.0**	**10.6**	**21769**
FO3_LP	623	8	612	12.0	20.4	12.5	22587
Brent-Kung	471	13	667	13.1	14.4	9.6	16969
BK_L9	677	9	590	11.6	18.3	10.8	20683
BK_L10	607	10	626	12.3	16.8	10.5	19580

Table 2. 64b static adder results @ fastest points (load=5fF, 1.5V)

Adder	# C	# L	Delay		E (pJ)	EDP (ns* pJ)	Area (um^2)
			ps	FO4			
Full KS	904	8	532	10.4	25.0	13.3	32384
FO2_LP	**681**	**8**	**534**	**10.5**	**16.8**	**9.0**	**21634**
FO2_LP2	**693**	**9**	**526**	**10.3**	**18.5**	**9.7**	**24126**
			532	**10.4**	**17.7**	**9.4**	**23602**
BK_L9	677	9	543	10.6	16.3	8.9	20344

Table 3. Comparison with literature (S: static, D: dynamic, P: peak, E: energy per op., I: IBM SOI)

64b Adder	Process (um)	Type	VDD (V)	Delay (ps)	FO4	E (pJ)	Area (um^2)
FO2_LP2 (load: 5fF)	0.13	S	1.5	526	10.3	18.5	165x146
			1.2	671	9.9	10.8	
FO2_LP2 (load: 30fF)	0.13	S	1.5	558	10.9	19.0	154x141
			1.2	715	10.5	11.1	
PATMOS [18]	0.13	S	1.2		12.5	~18	
TVLSI '04 [19] (load: 40fF)	0.13I	S	1.1	326	11.9*	11.5	461x151
	0.18I	S	1.5	541	14.2*	29	
ISLPED '02 [20]	0.18I	S	1.5	720		96	735x280
Knowles [9]	0.25	S	2.5		11.8		
TCOMP '05[21]	0.18	S	1.8	800			33904
ICECS '03[22]	0.18	S	1.8	980		54	14737
ISSCC '06 [23]	0.13	D	1.2	238	3.9	30	
ISSCC '06 [24]	0.09	D	1.0	240	7.7	P: 62	417x75
		D	1.3	180		P:109	
ESSCIRC [25]	0.13	D	1.2		6.8	33.5	

* As in [26], FO4 inverter delay is 38ps for the IBM 0.18um SOI process; we assume linear scaling for IBM SOI 0.13um FO4 delay.

Table 1 lists the fastest points (the leftmost point in each E-D curve) we obtained for the major architectures we have implemented, with load of 30fF at sum bits. Table 2 presents the results for a couple of representative designs with 5fF load (around a DFF load).

For 30fF load, the full Kogge-Stone achieves a slight speed advantage (10ps or 0.2 FO4). However, when we count the input driver delay, our new KN8421_FO2 designs achieve the same speed. More importantly, the new designs with less wiring cost will scale better for a new technology.

The new design FO2_LP2 is 30%, 15% and 7% more energy efficient (in terms of Energy-Delay Product) than full Kogge-Stone, sparse-2 Kogge-Stone and Han-Carlson, respectively, at the fastest points.

Table 3 compares our new static 64b adder designs with the most recent high performance and energy efficient designs, for both static and dynamic circuits. The best static designs consume around 1/3 the energy of the dynamic designs and our designs compare favorably to the other best static designs, in FO4 delay, energy efficiency and area.

5. Conclusion

We examined extensively CLA and carry select adder architectures and considered their variants. We proposed improved architectures based on sparse CLA trees with specialized buffering techniques to reduce logic redundancy. The studied architectures span a wide range of tradeoffs in logic levels, fanouts and wiring complexity.

Our newly developed 64b adder designs are the fastest static CMOS adders ever reported, with a relative delay as low as 9.9 FO4 delays (for 1.2V and 5fF output loads) and as low as 10.5 FO4 delays (for 30fF output loads). They yield the best energy efficiency for a wide range of delay targets (except at low power where Brent-Kung is preferred) and are 30%, 15% and 7% more energy efficient than full Kogge-Stone, sparse-2 Kogge-Stone and Han-Carlson, respectively, at the fastest points. They consume only about 1/3 the energy of the most energy-efficient dynamic adders.

Acknowledgment

We wish to thank Miodrag Vujkovic and Dave Wadkins for the flow support.

References

[1] N. Weste, D. Harris, "CMOS VLSI Design: A Circuits and Systems Perspective," 3rd ed., Addison-Wesley, 2004

[2] J. P. Uyemura, "CMOS Logic Circuit Design," Kluwer Academic Publishers, 1999.

[3] Bedrij, "Carry-select adder," IRE Trans. Electronic Computers, vol. EC-11, June 1962, pp. 340-346.

[4] A. Weinberger, J.L. Smith, "A Logic for High-Speed Addition," Nat'l Bur. Standard, Cir. 591, pp.3-12, 1958

[5] R.P. Brent, H.T. Kung, "A Regular Layout for Parallel Adders," IEEE Trans., C-31(3):260-264, March 1982.

[6] R.E. Ladner, M.J. Fischer, "Parallel Prefix Computation," J.ACM, 27(4):831-838, Oct. 1980.

[7] P. Kogge and H. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recursive Equations," IEEE Tran. Comp, August 1973.

[8] T. Han, D.A. Carlson, "Fast Area-Efficient VLSI Adders," IEEE Symp. Comp. Arithmetic, pp.49-56, 1987

[9] S. Knowles, "A Family of Adders," Proc. 14th IEEE Symp. Computer Arithmetic, pp30-34, April 1999.

[10] H. Ling, "High-Speed Binary Adder," IBM J. Research. Develop. Vol. 25, No. 3, May 1981.

[11] Sutherland, et al. "Logical Effort: Designing Fast CMOS Circuits," Morgan Kaufmann, 1999

[12] D. Harris, I. Sutherland, "Logical effort of carry propagate adders," The 37th Asilomar Conf. Signals, Systems & Computers, Vol 1.1, Nov. 2003, pp:873-878

[13] V.G. Oklobdzija, et al. "Energy-delay estimation technique for high-performance microprocessor VLSI adders," Proc, ARITH-16, 2003, pp:272-279.

[14] Y. Choi, E. E. Swartzlander, "Parallel prefix adder design with matrix representation," 17th IEEE Symp. Comp. Arithmetic, 2005 (ARITH-17), pp90-98.

[15] S. Mathew, et al., "A 4GHz 300mW 64b Integer Execution ALU with dual supply voltages in 90nm CMOS," ISSCC 2004, Dig. Tech. Papers, pp. 162-519.

[16] M. Vujkovic, D. Wadkins B. Swartz, C. Sechen, "Efficient timing closure without timing driven placement and routing," Proc. DAC, June 2004, pp 268-273.

[17] H. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," IEEE JSSC, Vol. 19, no. 4, Aug 1984, pp468-473.

[18] R. Zlatanovici, et al. "Power - performance optimization for custom digital circuits," Proc. Integrated Circuit and System Design: PATMOS 2005, pp.404-414.

[19] A. Neve, et al. "Power-delay product minimization in high-performance 64-bit carry-select adders," IEEE Tran. VLSI, Vol. 12, March 2004, pp. 235-244.

[20] Neve, et al."Design of a branch-based 64-bit carry-select adder in 0.18 um partially depleted SOI CMOS," Proc. ISLPED, 2002, pp.108-111.

[21] G. Dimitrakopoulos, D. Nikolos, "High-speed parallel-prefix VLSI Ling adders," IEEE Tran. Computers, Vol.54, Issue 2, Feb. 2005, pp.225 – 231.

[22] S. Perri, et al. "A low-power sub-nanosecond standard-cells based adder," ICECS, 2003, vol.1, pp. 296-299.

[23] K.H.Chong, L. McMurchie, C. Sechen, "A 64b Adder Using Self-Calibrating Differential Output Prediction Logic" ISSCC 2006, Dig. Tech. Papers, pp. 440-441.

[24] S. Kao, et al. "A 240ps 64b Carry-Lookahead Adder in 90nm CMOS," ISSCC 2006, Dig. Tech. Papers, pp. 438-439.

[25] R. Zlatanovici, B. Nikolic, "Power - performance optimal 64-bit carry-lookahead adders," European Solid-State Circuits, ESSCIRC 2003, pp.321-324.

[26] D. Stasiak, F. Mounes-Toussi, S.N. Storino, "A 440-ps 64-bit adder in 1.5-V 0.18 um partially depleted SOI technology", IEEE JSSC, Oct. 2001, pp. 1546-1552.