Energy-Efficient Design Methodologies: High-Performance VLSI Adders

Bart R. Zeydel, *Member, IEEE*, Dursun Baran, *Student Member, IEEE*, and Vojin G. Oklobdzija, *Fellow, IEEE*

***Abstract*—Energy-efficient design requires exploration of available algorithms, recurrence structures, energy and wire tradeoffs, circuit design techniques, circuit sizing and system constraints. In this paper, methodology for energy-efficient design applied to 64-bit adders implemented with static CMOS, dynamic CMOS and CMOS compound domino logic families, is presented. We also examined 65 nm, 45 nm, 32 nm, and 22 nm technology nodes to explore the applicability of the results in deep submicron technologies. By applying energy-delay tradeoffs on various levels, we developed adder topology yielding up to 20% performance improvement and 4.5× energy reduction over existing designs.**

***Index Terms*—Arithmetic and logic structures, computer arithmetic, energy-efficient design, high-speed arithmetic, low-power design, VLSI.**

I. Introduction

ADVANCES in CMOS technology have led to a renewed interest in the design of basic functional units for digital systems. As technology scaling no longer achieves constant power density, the energy-efficiency of functional units is of increasing importance to system designers. Of these functional units, the adder is a basic block to apply the energy-efficient design methodologies. Parallel prefix adders have been implemented with different algorithms and circuits techniques under different constraints [13]. This wide implementation space makes the adders a good example study to explore the design methodologies. In the rest of the paper, energy and delay tradeoffs for the adders will be presented to give architectural insights to circuit designers about the possible energy saving techniques that will be applicable to larger class of prefix computation algorithms.

For over half a century, changing technology and operating constraints have necessitated the refinement of adder implementations to obtain improvements in performance [1]–[13]. Recently, energy [14] has been introduced into this analysis, making profound changes of the design exploration process. In early integrated circuits, the primary constraint was area, which led to the development of several simple schemes, such as carry-skip, which improves the speed of addition while maintaining low gate-count [6], [8]. As technology scaling continued, allowing for more logic gates per chip, complex parallel prefix schemes, yielding fast adder designs became viable [9]–[13]. In modern CMOS technologies, transistor sizing has been used to find the optimal tradeoff between speed and energy consumption of an adder [15]–[18]. In this environment, implementations must be compared by optimizing the circuit sizing for speed under energy, output load, input size, and performance constraints [15], [16]. The energy-delay tradeoff that exists for each of these constraints has blurred the comparison picture requiring complex analysis before a distinction can be made [15].

With each new technology generation, the gap between addition algorithms and energy-efficient realizations has grown. Guidance for energy-efficient addition algorithm selection has been nonexistent and only recently designs were compared using more than a single implementation point [15]. Further adding to the problem are the reduced benefits obtained from technology scaling. Supply and threshold voltages can no longer be reduced at the same rates as in previous technologies and static power dissipation from leakage continues to grow. Thus, improvements in energy efficiency must come from either restructuring the adder, optimally sizing transistors, or through the use of new devices.

The intent of this paper to provide a list of energy-efficient circuit techniques that will be applicable to any prefix computation algorithms. First, the adder topologies are given to familiarize the reader to the adder circuits. Then, the leading addition recurrence algorithms are explored along with the best published realizations of these algorithms to identify favorable characteristics of each. The paper is organized as follows. Section II presents the adder topologies and Section III examines Weinberger's recurrence for addition and attempts to clarify the presentation of the recurrence examined by Ling. Section IV presents energy-efficient methodology on VLSI adder realizations in modern CMOS technologies. In Section V, three adder realizations created from the methodology in Section IV are presented. Section VI presents the results of technology characterization across 65 nm, 45 nm, 32 nm, and 22 nm technology nodes. Section VII presents an energy-delay space comparison of the proposed adders to the best published designs. Section VIII concludes the paper.

II. Adder Circuit Topologies

In this paper, the adder circuits are modeled using the topologies defined below. The main purpose is to provide structural details about the adder designs for possible energy optimizations.

1) *Logic Depth (LD):* The maximum number of logic stages from output to inputs. In this paper, each logic gate is

Manuscript received November 05, 2009; revised March 12, 2010; accepted March 22, 2010. Current version published June 09, 2010. This paper was approved by Associate Editor Stefan Rusu. This work was supported in part by SRC Research Grant 2009-HJ-1836, California MICRO, Intel Corporation, and IBM Corporation.

B. R. Zeydel is with Plato Networks Inc., Santa Clara, CA 95051 USA (e-mail: bart@acsel-lab.com).

D. Baran and V. G. Oklobdzija are with the Erik Jonsson School of Engineering and Computer Science, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: dursun@acsel-lab.com, vojin@acsel-lab.com).

Digital Object Identifier 10.1109/JSSC.2010.2048730

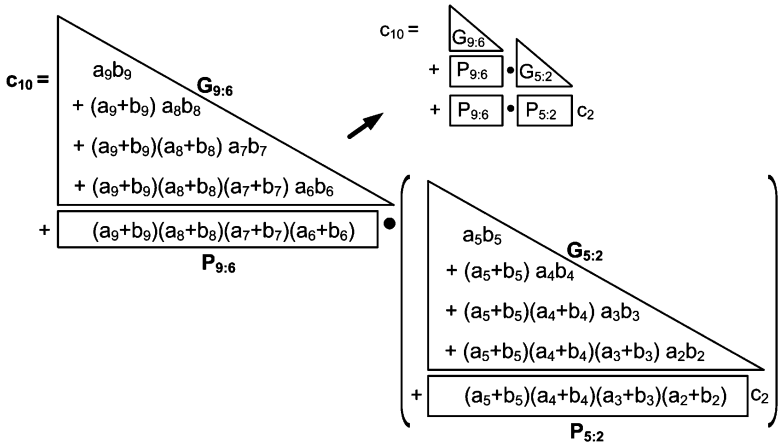

Fig. 1. Original presentation of Weinberger's Recurrence [1].

counted as a stage for fully static implementations. However, in compound designs, the dynamic gate and the following static gate are counted as one stage. For dynamic designs, some authors define a logic stage by counting the dynamic gate and the following static gate separately [34].

2) *Prefix (P):* The number of bits combined at each logic stage as defined above. For example, the two-input dynamic gate and the following inverter is defined as a prefix-2 stage for domino designs. The two-input dynamic gate and the following two-input static gate is defined as a prefix-4 stage for compound domino designs. This definition is more convenient with the definition of the stage used in this paper.
3) *Fan-out (F):*The maximum number of logical branching at any stage of the digital block.
4) *Wiring Complexity (WC):* For adders, the maximum number of wire tracks passing through the bit pitch at any logic stage is used as a rough estimate for the wiring complexity.

Prefix adders are consisting of two blocks, namely, sum and carry blocks. The topologies are reported for the critical block, that is, the carry block for adders. For example, the carry block of the static 64-bit Kogge-Stone prefix-2 adder is defined as $(\mathrm{LD},\mathrm{P},\mathrm{F},\mathrm{WC}) = (6,2,2,32)$. For the static 64-bit Han-Carlson prefix-2 adder, it is defined as $(\mathrm{LD},\mathrm{P},\mathrm{F},\mathrm{WC}) = (7,2,2,16)$. From this analysis, it is clear that the Han-Carlson adder trades logic depth for wiring complexity. At the same prefix, LD, F, and WC can be traded for each other to construct different adder implementations. A detailed analysis of adder topologies is provided in [13].

III. Addition Algorithms

Weinberger presented the most widely known carry recurrence for VLSI addition in 1958 [1]. Over the years, several addition algorithms have been developed. These algorithms manipulate the carry and sum equations in an attempt to improve the speed of addition. The equations for sum and carry are defined and indexed as follows in this paper:

$$s_i = a_i \oplus b_i \oplus c_i \tag{1}$$

$$c_{i+1} = a_i \cdot b_i + (a_i + b_i) \cdot c_i. \tag{2}$$

Ling modified this algorithm to reduce the complexity of the carry computation at the cost of increased complexity in the sum computation [3]. An analysis was later performed by Doran to determine the set of recurrences which have recurrence properties that are similar to Weinberger's and Ling's [4].

A. Weinberger's Recurrence

Weinberger [1], demonstrated that addition speed could be improved by parallelizing the computation of carry. Although widely credited with only the Carry Look-Ahead Adder, Weinberger's recurrence was not limited in group size or number of levels for carry computation [1]. The fundamental advancement of his work was the introduction of generate and propagate. Weinberger defined the terms: bitwise generate (g), bitwise propagate (p), group generate (G), and group propagate (P). These terms allow for carry computation to be performed in parallel, yielding a significant improvement in performance compared to ripple-carry addition. Weinberger's original presentation of the recurrence is shown in Fig. 1. For a group of 4 bits, Weinberger recurrence has ten terms for the generation of $G_{5:2}$ from the inputs and four terms for the generation of $P_{5:2}$. The maximum transistor stack height is 5 and it is not in the limit of ECL technology.

Weinberger demonstrated that G and P could be used to create blocks of arbitrary size and parallelized to form multiple levels of recurrence [1] (Fig. 1). Thus, the majority of parallel prefix adders proposed for high-performance addition are specific realizations of Weinberger's recurrence, e.g., Kogge-Stone [9], Brent-Kung [10], Han-Carlson [11], Ladner-Fischer [12], and those described by Knowles [13].

B. Ling's Recurrence

IBM ECL technology limitations on fan-in (limited to 4) and wired-OR (limited to 8) motivated Ling to develop a transformation that reduced the fan-in of Weinberger's recurrence [3]. As originally presented [3] Ling's transformation is very difficult to understand. For clarity, a simple derivation of Ling's transformation will be shown. This derivation provides the physical meaning of the signals used in Ling's transformation and identifies the favorable characteristics of Ling for implementation in

modern CMOS technology. In the derivation, the bitwise generate signal is defined as: $g_i = a_i\ b_i$ and the bitwise propagate signal is defined as: $t_i = a_i + b_i$. Note that the propagate signal t_i is the same as Weinberger's p_i (when implemented using an OR). To maintain consistency with Ling's original paper, t_i will be used for propagate.

Ling's transformation reduces the complexity of Weinberger's recurrence by factoring t_i from c_{i+1} to create a pseudo-carry (h_i) on which the recurrence is performed. The transformation is shown below on c_1 to form h_0. The carry-out signal, c_1, of the first bit position is

$$c_1 = g_0 + t_0 \cdot c_0. \tag{3}$$

Ling's transformation uses the property $t_i \cdot g_i = g_i$ to form

$$c_1 = t_0 \cdot g_0 + t_0 \cdot c_0 = t_0 \cdot (g_0 + c_0) \tag{4}$$

where $g_0 + c_0 = h_0$, which leads to

$$c_1 = t_0 \cdot (g_0 + c_0) = t_0 h_0. \tag{5}$$

The general transformation of c_i is defined as

$$c_i = \begin{cases} t_{i-1} h_{i-1}, & i > 0 \\ c_0, & i = 0 \end{cases} \tag{6}$$

where the pseudo-carry, h, is defined as

$$h_i = g_i + c_i. \tag{7}$$

The physical meaning of the *pseudo-carry* signal (h) can be described as follows. By factoring t_i out of the carry expression and propagating h_i instead of c_{i+1}, all cases where carry is generated and/or propagated from the stage preceding stage i are included in h_i. This includes the case where a carry-in to the ith stage can be assimilated (which should not result in a carry-out). The assimilate condition is handled when forming c_{i+1} by AND'ing h_i with t_i to produce c_{i+1}. If the carry-assimilate (carry-kill) condition exists then $t_i = 0$, which results in $c_{i+1} = 0$.

A recurrence for h_i can be defined as has been done previously for Weinberger's c_i. The group pseudo-carry $(H_{i:j})$ and transmit $(T_{i:j})$ which allow for parallel prefix computation can be defined over the group of bits $i:j$ (capital letters are used to refer to the group):

$$T_{i:j} = t_i \cdot t_{i-1} \cdot \cdots \cdot t_j \tag{8}$$

$$H_{i:j} = g_i + g_{i-1} + t_{i-1} \cdot g_{i-2} + t_{i-1} \cdot t_{i-2} \cdot g_{i-3} + \cdots + t_{i-1} \cdot t_{i-2} \cdot \cdots \cdot t_{j+1} \cdot g_j. \tag{9}$$

The recurrence can be expressed using the "•" operator as

$$\begin{pmatrix} H_{i:j} \\ T_{i-1:j-1} \end{pmatrix} \bullet \begin{pmatrix} H_{j-1:k} \\ T_{j-2:k-1} \end{pmatrix} = \begin{pmatrix} H_{i:j} + T_{i-1:j-1} \cdot H_{j-1:k} \\ T_{i-1:j-1} \cdot T_{j-2:k-1} \end{pmatrix} \tag{10}$$

The transformation from Weinberger's recurrence to Ling's recurrence for a group of 4 bits is shown in the example in Fig. 2. This figure should dispel any difficulties associated with understanding the original Ling's derivation [3]. As shown in Fig. 2, Ling recurrence has seven terms for the generation of $H_{5:2}$ from the inputs and four terms for the generation of $T_{4:1}$. The maximum transistor stack height is 4 and it is in the limit of ECL technology.

From (8), (9) and Fig. 2, it is clear that the recurrence can be realized using groups of any size and number of levels, making it suitable for use with the parallel prefix structures that have historically been used with Weinberger's recurrence [19], [20].

The advantage of using *pseudo-carry* instead of carry is offset by the increased complexity of sum computation, which requires the *real* carry to form individual sum signals. In CMOS technology s_i can be efficiently calculated conditionally, thus avoiding the AND operation on the critical carry path:

$$s_i = \begin{cases} a_i \oplus b_i & h_{i-1} = 0 \\ a_i \oplus b_i \oplus t_{i-1} & h_{i-1} = 1 \end{cases}. \tag{11}$$

Conditional sum computation in a Ling adder can also be extended to several bit position to create a sparse recurrence structure [19], [20].

In 1988, Doran analyzed all possible recurrences for binary addition that could be created from inputs to adjacent bits $(a_i, b_i, a_{i-1}, b_{i-1})$. In his analysis he found that there were four variants with Ling-like properties [4]. The conclusion from his work is that the only recurrences worth considering for CMOS realization are Weinberger and Ling.

IV. Design Methodologies

Concerns about energy consumption have forced digital designers to develop techniques for improving energy efficiency. Several approaches have been proposed to improve energy efficiency: proper selection of circuit family and prefix; reducing the number of logic stages without increasing gate count; reducing switching activity; reducing the number of logic gates; and reducing the wiring complexity. This section presents the approaches for the optimal construction of high-performance VLSI adders in a given technology.

A. Logic Family Selection

In VLSI design, the selection of logic family is dictated by the system performance target. In structures where the performance target is relaxed or where energy is the primary constraint, static circuits are preferred due to their lower switching activity. In addition to that, static circuits are robust and become more preferable as technology scales down. However, in high-performance microprocessors, dynamic circuits are often required in order to achieve desired target frequency. There are two types of dynamic circuit families used in modern digital systems: (a) dynamic CMOS domino and (b) CMOS compound domino. The main difference between these families is that CMOS domino utilizes a static inverter at the output, while CMOS compound domino uses a static inverting logic stage. This helps to reduce the power by eliminating power hungry dynamic stages and bundling their functionality in the static CMOS inverting stage. A prefix-2 computation performed in the static gate of the compound domino circuit allows for the number of stages in a 64-bit prefix-2 Kogge-Stone carry structure to be reduced from

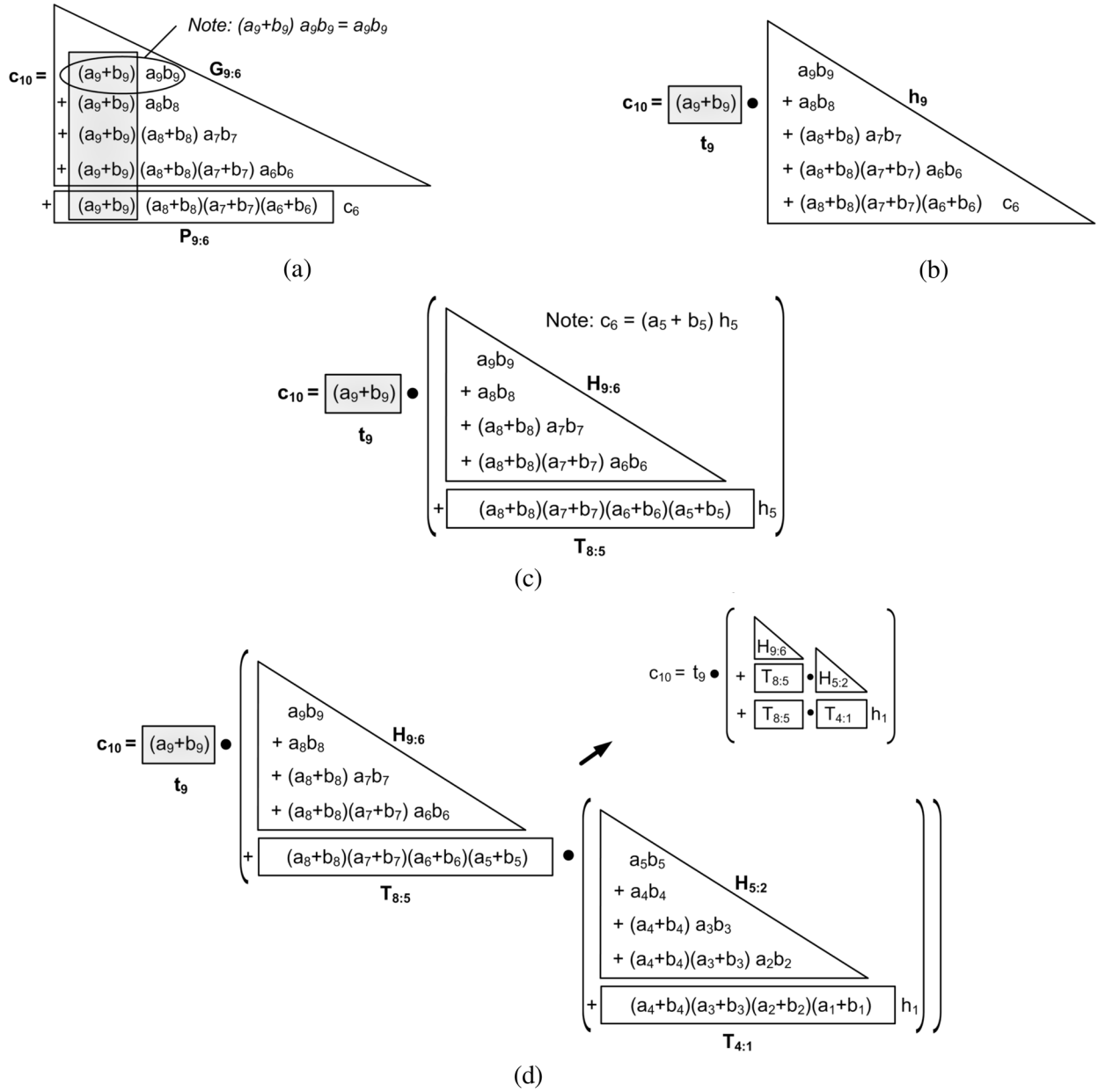

Fig. 2. (a) Weinberger's Recurrence. (b) Ling's transformation. (c) Ling's Recurrence. (d) Multi-level Ling.

six CMOS domino stages to three CMOS compound domino stages.The savings diminishes for higher prefix structures. For example, a 64-bit prefix-4 Kogge-Stone carry structure requires three CMOS domino stages compared to two CMOS compound domino stages. As a summary, static circuits are good for power and domino circuits are good for speed. Compound domino designs can combine the speed advantage of dynamic designs and the power advantage of static designs.

B. Prefix Selection

In static CMOS logic, the prefix is mostly limited to 2 because of transistor stack height limitation while dynamic designs enable to use of higher prefixes. As prefix of the design increases the logic depth decreases and it is expected to lead to delay improvement. However, higher prefix requires more complex gates with increased stack height resulting in higher gate delay. Therefore, there is an optimal prefix that depends on the design constraints and implementation. Given a technology constraint on transistor stack height, it is possible to determine which prefixes are feasible for Weinberger's and Ling's recurrence (note: Ling always enables a prefix of one more in the first recurrence stage as compared to Weinberger). These prefix values can be used to construct any of the previously published recurrence structures for either pseudo-carry or carry.

C. Conditional Sum

Dynamic circuits have inherently high switching activity. To reduce the switching activity and energy used in the wiring tracks, Mathew [14] suggested to conditionally compute the sum, using static logic gates. This approach decouples the computation of sum for a group of bits from the carry computation. The carry or pseudo-carry is computed with high-speed

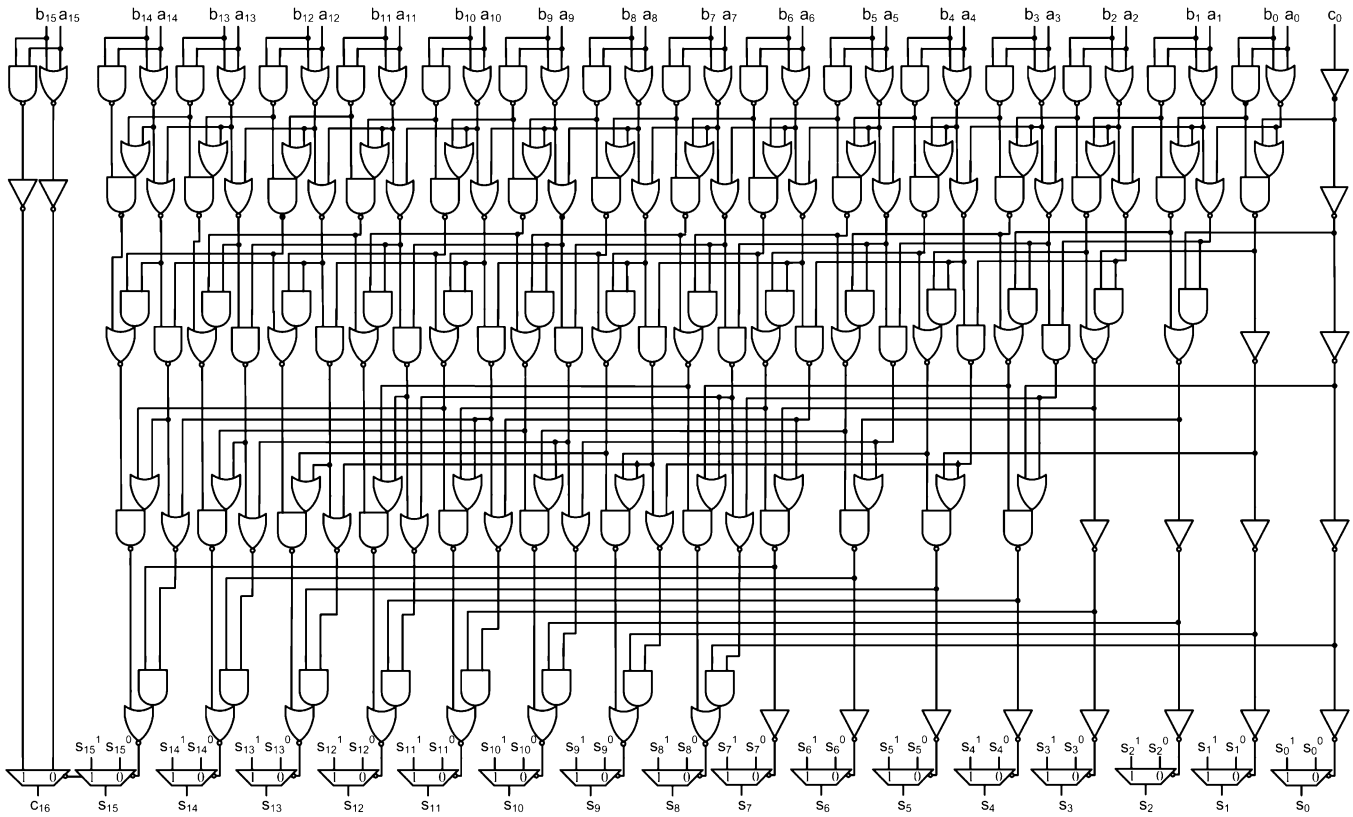

Fig. 3. 16-bit static minimum depth (fully parallel) Kogge-Stone Weinberger adder *(KS2-(W))*.

dynamic gates and is used to select the conditionally computed sum. Conditional logic demonstrates obvious benefits for dynamic adders where the switching activity of many gates is reduced from 50% (for the original gates on the dynamic paths) to 15%–20% (for the new static path gates).

The loading of the carry signal in a sparse design increases directly with the number of sum bits that are selected conditionally. For example, a carry signal in a design using a 2-bit conditional sum, often referred to as Sparse-2 (Fig. 4), has twice the load at the output of the carry signal as compared to the same carry signal in a Kogge-Stone (KS) design (Fig. 3). However, propagate and generate signals have fewer connections (fan-out). This mitigates the penalty of increased loading on carry when using conditional sum. Furthermore, by eliminating half of the wires in the carry structure, the load at the output of the bitwise propagate and generate gates is reduced. The energy saved by the reducing the wiring complexity can be used to make up for any speed lost.

Another factor is the implementation of the conditional sum block. The complexity of conditional sum computation differs in Ling and Weinberger adders. In Weinberger adders, the sum is selected by the carry-in to the group of bits [Fig. 5(a)], while in Ling adders the sum is selected by the pseudo-carry-in to the group [Fig. 5(b)]. In Ling, the pseudo-carry must be combined with t_i to form the *real* carry. The difference between the structures for 2-bit conditional sum computation is shown in Fig. 5. The critical path of the circuitry in the Ling adder is more complex, going through an XOR, MUX, and XOR compared to the critical path for Weinberger which is through only two XORs. However, the worst case number of input connections in Ling is fewer than in Weinberger (Fig. 5). Each input in Ling is connected to two logic gates (note: a_{i+1} and b_{i+1} are attached to the XOR and the NOR of the succeeding 2-bit conditional sum block). In Weinberger, a_i and b_i each connect to three gates; however, a_{i+1} and b_{i+1} each connect to only one gate.

It is interesting to note that in a KS design, the number of input connections for the conditional sum computation for a_i and b_i using Weinberger recurrence is one XOR gate, while using Ling recurrence results in each input being attached to one XOR gate and one NOR gate (used to compute transmit which is required to create carry from pseudo-carry). Thus, Ling has the same number of input connections in the 1-bit and 2-bit conditional sum block.

The conditional sum technique is a good approach for energy saving, because it trades the complexity of the critical carry block for the complexity of noncritical sum block. The energy reduction of the carry block will be higher than the energy cost of the sum block as far as the sum block complexity is less than the carry block. The benefit of conditional sum depends on bit-pitch and technology. Bit-pitch determines the wire length required in the design. Larger bit-pitches result in more wire capacitance, which degrades the energy and delay of a KS design faster than that of a sparse design. Technology determines the optimal sparseness a design should have. When the intrinsic delay of the conditional sum path becomes similar to the intrinsic delay of the carry path, the energy savings is reduced. Therefore, the optimal sparseness of a design should be chosen such that the intrinsic delay of the conditional sum block is less

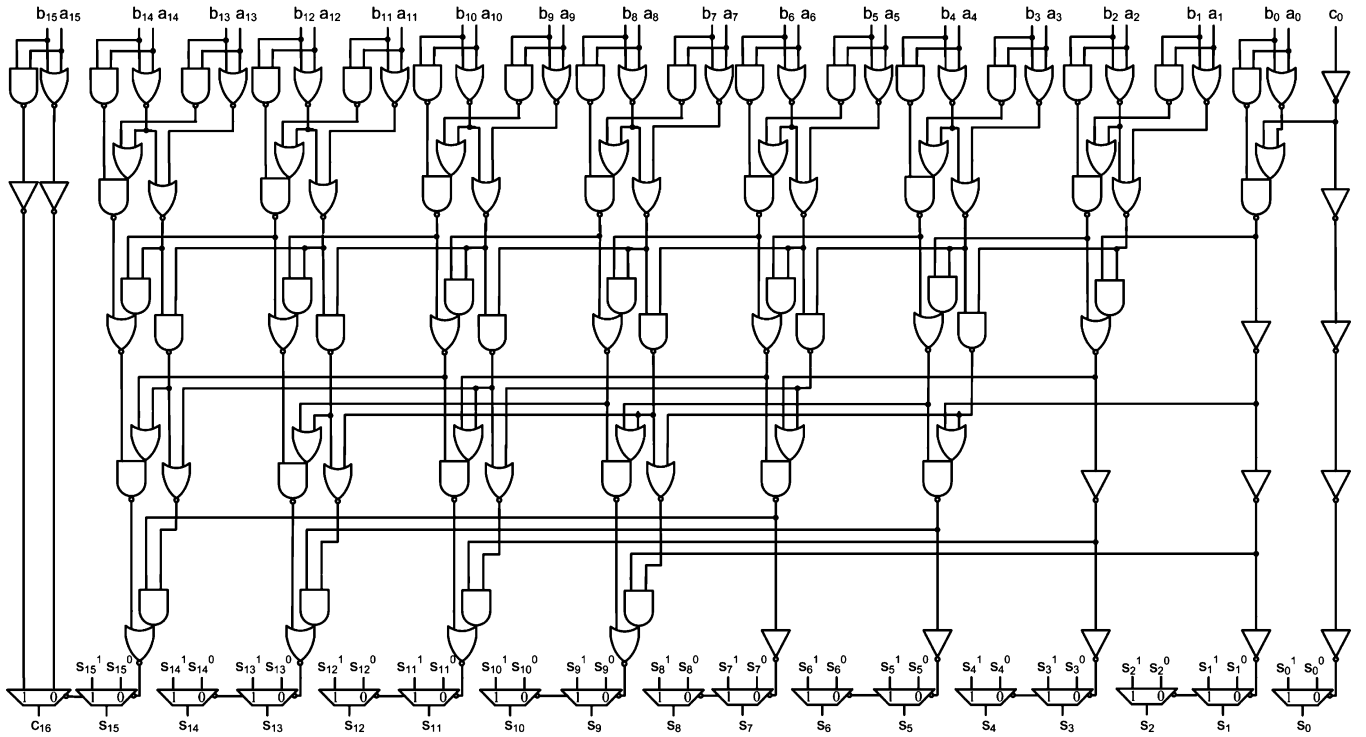

Fig. 4. 16-bit static minimum depth (fully parallel) Sparse-2 Weinberger adder.

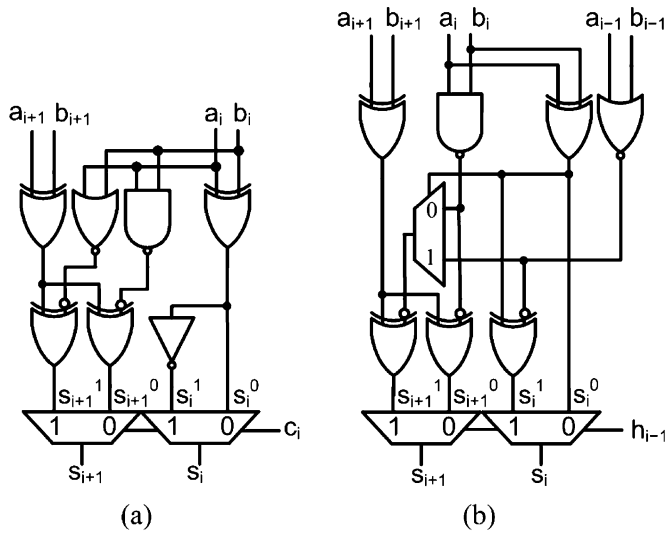

Fig. 5. Two-bit conditional sum computation. (a) Weinberger and (b) Ling.

than the intrinsic delay of the carry path. A simple guideline to follow is: the number of stages in the conditional sum block should be less than or equal to the number of stages in the carry path. For static designs which are typically limited to prefix-2, the conditional sum block should not exceed 4 bits for adders up to 64 bits. Dynamic adders, which can utilize prefix-4, should use either 2-bit conditional sum or 4-bit conditional sum blocks. It should be noted that it is possible to optimize the size of conditional blocks based on the relative delay of each carry path in the adder (similar to how the variable block adder optimizes the carry-skip structure [8]), however this approach significantly increases design time and is not examined in this work.

D. Logic Depth

Logic depth is defined as the number of logic stages from input to output. The number of stages depends on the prefix of design. A minimum depth adder implements addition in the minimum number of logic stages in a given technology (e.g., Kogge-Stone [9]). Minimum depth adders are used when high performance is required [13], [21], [23].

Minimum-depth sparse [14] structures can also be implemented using conditional sum. It should be noted that a sparse structure does not inherently have to be minimum depth. A sparse structure can also be implemented using additional logic stages to compute the carry for each bit, as in Han-Carlson [11]. The circuits for computing sum are simpler in Han-Carlson design than in a sparse design with 2-bit conditional sum. However, the carry signal computation is more complex: the carry signal in the sparse adder is loaded directly by two MUXs, while the carry signal of the Han-Carlson design has an additional stage consisting of an OAI and inverter which are each loaded by a MUX (Fig. 6). When only examining the carry structure, it seems clear that a minimum-depth structure has better energy and delay characteristics (fewer logic stages and gates and less loading). However, this analysis ignores the input loading of the adder. In Han-Carlson design, the worst case input loading from the sum computation is an XOR gate, while the worst case input loading of the Sparse-2 design is one XOR, one NAND, and one NOR gate. Additionally, fewer gates are required for computing conditional sum in the Han-Carlson design, where each sum is computed using one XOR and one inverter (which create the inputs to the MUX for sum selection). Thus, for a group of 2 bits, Han-Carlson requires two XORs and two inverters (INV). In the sparse design, the number of

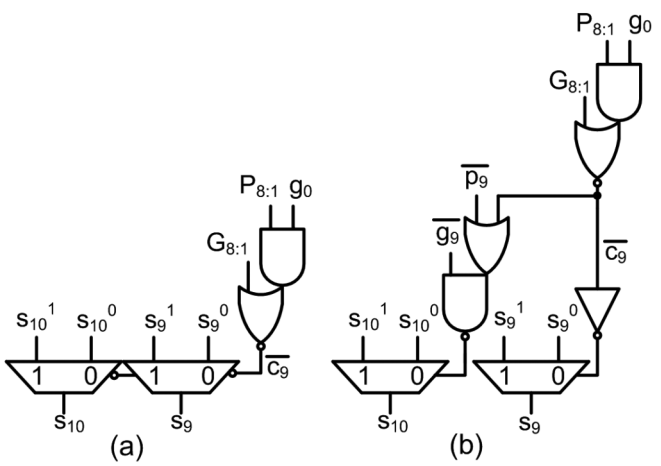

Fig. 6. Carry loading in (a) Sparse-2 design and (b) Han-Carlson design.

gates required for computing the 2-bit conditional sum is four XORs, one NAND, one NOR, and one INV [Fig. 5(a)].

Structures utilizing schemes with more stages than minimum depth such as Han-Carlson and Ladner-Fisher [12] demonstrate delay penalties or increased energy at the same delay due to the increased number of logic stages and higher loading on the critical path. Therefore, the most efficient realization for the highest performance designs are the minimum-depth designs. For designs in which the delay is relaxed (e.g., low power), it is often desirable to reduce gate count (and in turn energy) by increasing the delay. In these cases, designs that use more stages can be more efficient.

E. Merging Bitwise Operations Into First Recurrence Stage

The first logic stage of most high-performance adder realizations contains gates which compute the bitwise generate and propagate (or transmit). These signals are then used as inputs to a recurrence structure which computes each carry or pseudo-carry.

In static Weinberger adders, the bitwise operations for generate and propagate are often computed in a separate logic stage than the first recurrence stage due to an nMOS and pMOS transistor stack height limitation of 2. Transistor stack height refers to the number of transistors connected in series between a supply terminal and the output. As transistor stack height increases, the delay of the path becomes larger. Often, stack height limitations are chosen in order to simplify the design process, in order to improve the effectiveness of static timing tools [24].

Using a stack height of 3, the bitwise operations can be merged directly into the first recurrence stage [Fig. 7(a)]. This approach allows for a single stage to be removed from any Weinberger adder realization. As discussed previously, Ling's transformation reduces the transistor stack height in the first stage of the recurrence by one transistor. Therefore, in static Ling adders, with a transistor stack height limitation of 2, a prefix-2 computation of $H_{i:i-1}$ and $T_{i:i-1}$ can be performed in the first stage without violating the transistor stack height limitation [Fig. 7(b)]. This modification allows for one logic stage to be removed from the critical path of a static Ling adder

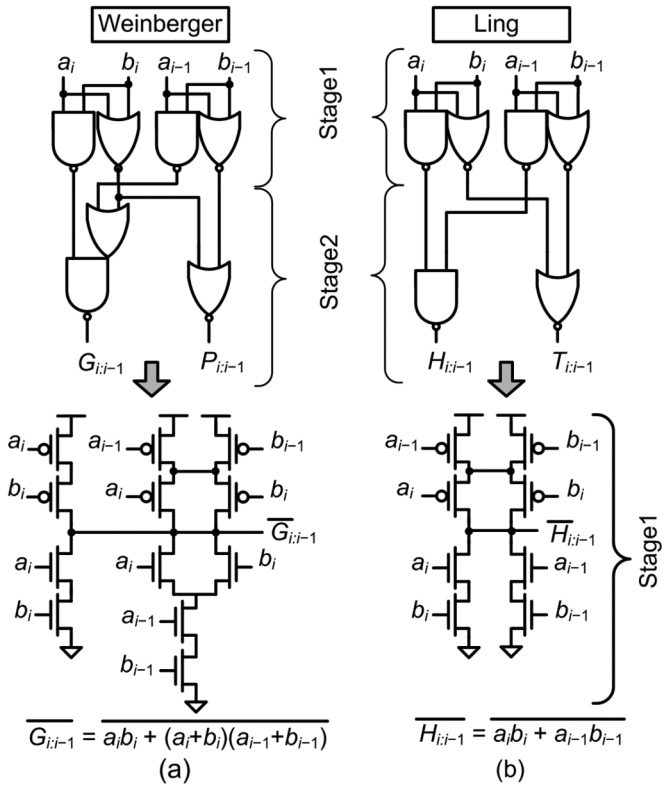

Fig. 7. Circuits for combined bitwise operations and first prefix-2 for (a) $\overline{G_{i:i-1}}$ and (b) $\overline{H_{i:i-1}}$.

compared to a static Weinberger adder when the transistor stack height is limited to 2.

In Fig. 7, the circuit for computing $\overline{H_{i:i-1}}$ has only two nMOS transistors in series compared to the three nMOS transistors in series required for computing $\overline{G_{i:i-1}}$. Notice that the stack height difference between $\overline{G_{i:i-1}}$ and $\overline{H_{i:i-1}}$ is due to the removal of $a_i + b_i$. The logic gates which compute $\overline{P_{i:i-1}}$ and $\overline{T_{i:i-1}}$ are not shown because they are identical and have a worst case transistor stack height of 2. Subsequent carry stages for both Weinberger and Ling have the same stack height since the recursion is performed using the same prefix operator " $\bullet$ " on either G and P for Weinberger or H and T for Ling [19], [20]. Thus, there is no difference between a Weinberger adder and a Ling adder in subsequent recurrence stages. Only the first recurrence stage and the sum computation differ. An example of a 16-bit Ling adder with bitwise operations merged into the first recurrence stage is shown in Fig. 8.

Dynamic adders can implement even higher prefix circuits in the first logic stage [21], [22]. In dynamic circuits, both Weinberger and Ling adders can combine bitwise operations into the first logic stage. The circuit for computing Weinberger's G requires one more nMOS transistor in series compared to Ling's H. Depending on transistor stack height limitations, this can result in a reduction of one logic stage in dynamic adders using Ling recurrence as compared to those using Weinberger recurrence.

Merging the bitwise operations into the first recurrence stage reduces the number of logic stages from six to five in Ling's KS design. This reduction is achieved without significantly increasing the complexity of the logic gates (only the first gate is

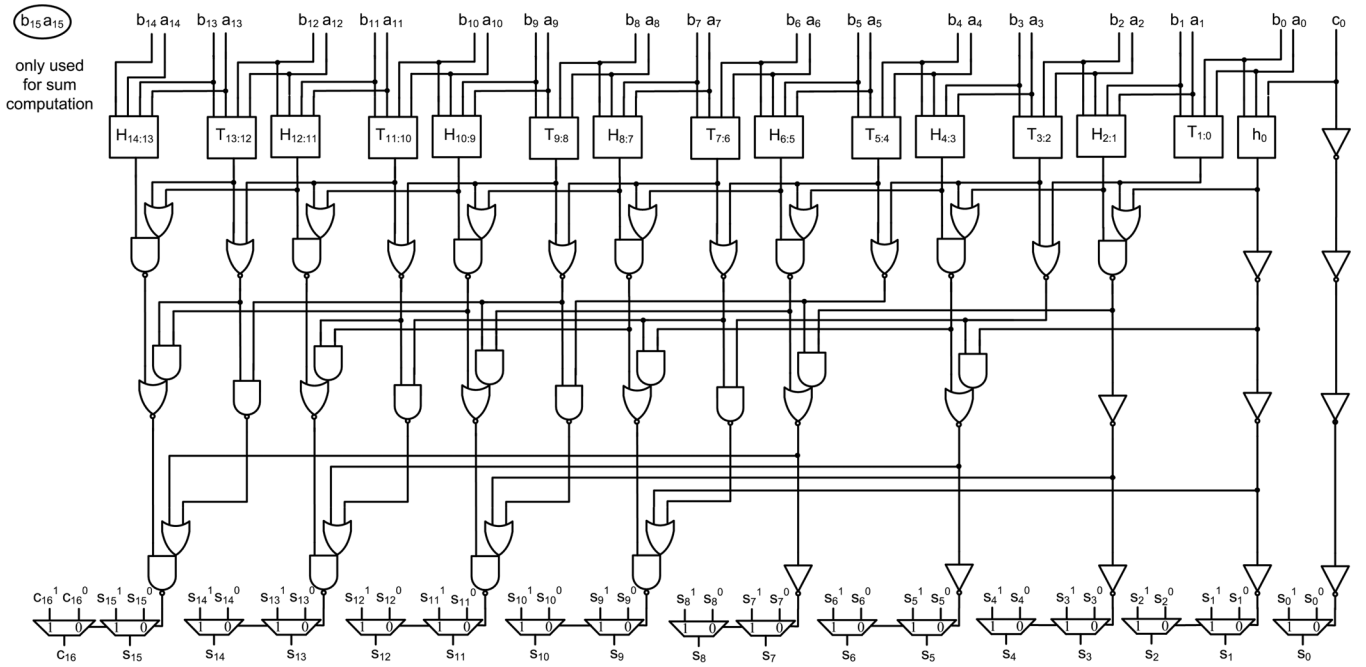

Fig. 8. 16-bit static minimum depth (fully parallel) Sparse-2 Ling adder with merged first recurrence stage and bitwise operations *(KS2-S2-(L))*.

slightly more complex). A portion of the energy saving comes from the reduction of logic gates. The additional energy savings is a result of the reductions in the size of subsequent stages. By having one less stage, the gates in each stage of the merged designs can run slower (i.e., can be smaller) than the gates in the design with more stages which must run faster (i.e., be larger) in order to operate the adder at the same speed. In summary, the merging technique is a good approach to reduce the energy consumption as far as the increase in the complexity of the merged stage is acceptable. For high-performance applications, it is an effective method for energy saving since it reduces the number of stages which provides extra room for delay relaxation.

F. Use of Ling Recurrence

Ling's recurrence demonstrates realization advantages compared to Weinberger by reducing the complexity of the carry block using pseudo-carry, resulting in improved performance. In addition, merging the bitwise operations into the first recurrence stage is achieved without exceeding a stack height limitation of 2. It means that Ling recurrence allows the removal of a logic stage from the critical path of the carry block, incurring energy saving.

It should be noted that Ling's recurrence is not compatible with adders that are built upon full adders, such as carry-skip, variable block, and carry-select. There are two limitations of Ling which make implementation of these types of adders impractical. First, Ling cannot use an XOR for propagate (thus requiring an additional gate for computing propagate instead of reusing partial sum). Second, the pseudo-carry must be combined with the bitwise propagate to form the true carry-in to a block (adding additional logic stages in the path compared to Weinberger). Thus, Ling should only be used in adders which are not built on top of full adders such as parallel prefix adders, and where the sum computation is separate from the computation of carry.

TABLE I
LE PARAMETERS IN 65 NM, 45 NM, 32 NM, AND 22 NM CMOS TECHNOLOGIES

	INV		NAND2		NOR2	
	g	p	g	p	g	p
65nm	1.00	1.47	1.17	2.70	1.63	3.54
45nm	1.00	1.52	1.20	2.85	1.69	3.72
32nm	1.00	1.50	1.25	2.92	1.77	3.91
22nm	1.00	1.55	1.39	3.10	1.78	4.12

G. Load Buffering

After the final XOR (MUX) stage, it is possible to add inverters to drive the output load because inverters are the most energy-efficient drivers. Extra delays come from the parasitic and effort delay of the added inverters. However, the delay of the original circuit will be reduced since it drives added inverters that are smaller than the output load. In addition, extra energy is consumed by added inverters but the original circuit's size is reduced. There is a tradeoff between the saved delay/energy and extra delay/energy coming from the added inverters. Load buffering provides energy savings for heavily loaded designs under the same delay constraints. As the load is reduced, the energy saving of the adder circuit cannot compensate for the extra energy consumed by the extra inverters. The energy saving of load buffering depends on the driving strength of the original circuit and the path gain.

As technology scales down, the logical effort and parasitic of the complex gates become larger as compared to the inverter (Table I). This trend means that the driving strength of the complex gates (e.g., NAND2 and NOR2) has weakened compared to the inverter in new technology nodes. Therefore, addition of buffer stages can provide a better energy-efficient solution for future technologies.

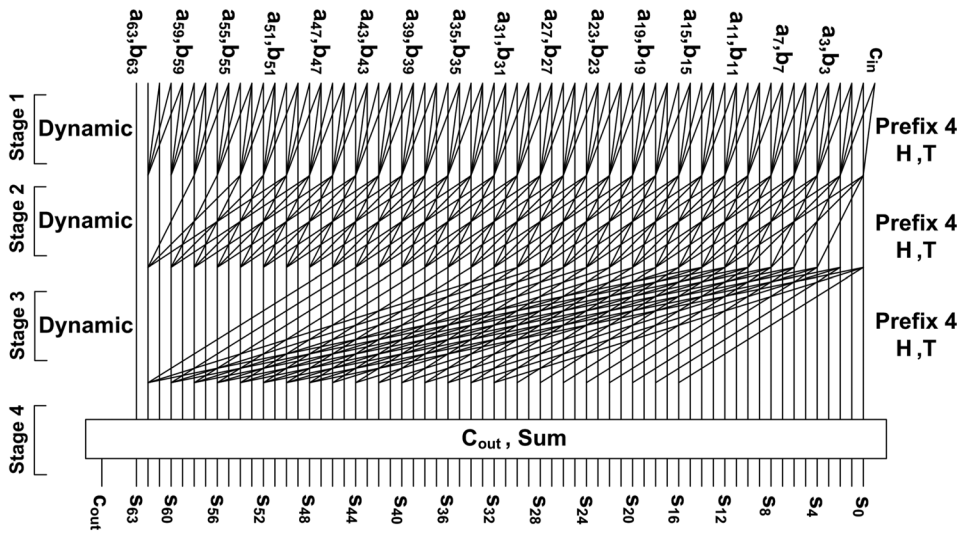

Fig. 9. 64-bit domino conditional four-stage *FZO-S2* adder.

V. Energy Efficient Adders

In this section, we present three of the best 64-bit adder designs in terms of performance and energy. They are obtained by carefully trading off the relevant features we observed while comparing various possible tradeoffs in terms of algorithm, recurrence, logic structure, technology, wiring, and energy-delay balance of each using our design estimation tools [16], [20]. These adders are products of the design methodologies given in Section IV. We obtained three designs, described below.

A. 64-Bit Static Kogge-Stone Prefix-2 Conditional Ling: KS2-S2-(L)

A prefix-2 KS Ling *(KS2-(L))* adder which merges bitwise operations into the first recurrence stage can be constructed in a similar fashion as a Weinberger KS adder. This adder can be implemented using static *(KS2-(L))*, domino *(KS2-(L))*, and compound domino *(KS4-(L))* logic styles for various bit widths. 64-bit carry can be computed in six static logic stages. When using compound CMOS domino logic, the dynamic gate in the first stage can operate directly on the input operands to perform a prefix-2 operation for H and T, followed by a static gate which performs a prefix-2 operation. Thus, a 64-bit carry can be computed in either three compound CMOS domino logic stages or six CMOS domino logic stages. The sum for each bit is conditionally computed and selected using the pseudo-carry. Conditional sum computation is done by use of (11).

The complexity of the *KS2-(L)* adder can be reduced by using a Sparse-2 design *(KS2-S2-(L))*. This approach allows for the number of gates in the carry structure to be reduced by half. To maintain four stages (for CMOS compound domino) or seven stages (for CMOS domino) in the adder, 2-bit conditional sum blocks are used. Sums are selected based on the pseudo-carry, h_{i-1}, to the block:

$$s_{i+1} = \begin{cases} a_{i+1} \oplus b_{i+1} \oplus g_i & \text{if } h_{i-1} = 0 \\ a_{i+1} \oplus b_{i+1} \oplus (g_i + t_i t_{i-1}) & \text{if } h_{i-1} = 1 \end{cases} \quad (12)$$

As is the case with the *KS2-(L)* adder, the *KS2-S2-(L)* adder can be implemented using static, domino, and compound domino logic gates and for various bit widths, such as 32 bits and 16 bits.

B. 64-Bit CMOS Domino Four-Stage Conditional Ling: FZO Adder

FZO-S1 (Sparse-1 implementation) is a 64-bit domino prefix-4 Kogge-Stone Ling (L) adder with merged bitwise operations. It is a four-stage adder optimized for speed. Using prefix-4 domino logic gates, the pseudo-carry for each bit can be computed in three domino logic stages. The dynamic gate in the first stage operates directly on the input operands to perform prefix-4 operations for H and T, followed by an inverter. The succeeding two stages perform prefix-4 operations for H and T, allowing for the 64-bit pseudo-carry to be computed in three domino logic stages. The sum for each bit is conditionally computed and selected using the pseudo-carry.

The complexity of a *FZO-S1* adder can be reduced by computing every other pseudo-carry, using a Sparse-2 design *(FZO-S2)*. This approach allows for the number of gates in the carry structure to be reduced by half in the *FZO-S1* adder. To maintain four domino logic stages in the adder, 2-bit conditional sum blocks are implemented (Fig. 9). Sums are selected based on the pseudo-carry, h_{i-1}, to the block as shown in (12).

C. 64-Bit CMOS Compound Domino Conditional Three-Stage Ling: EZO Adder

EZO-S1 (Sparse-1 implementation) is a three-stage Ling adder. It is similar to the *FZO-S1* adder, however, it utilizes CMOS compound domino logic gates instead of CMOS domino logic gates. A prefix-4 dynamic gate is used in the first stage of the recurrence for computing $H_{i:i-3}$ and $T_{i-1:i-4}$ directly from the input operands. The static gates in the first logic stage perform prefix-2 operations on H and T (the prefix-4 dynamic gate and the following prefix-2 static gate are defined as a prefix-8 compound domino stage). The equations used in the first stage dynamic gates for computing $H_{i:i-3}$ and $T_{i-1:i-4}$ are

$$H_{i:i-3} = a_i b_i + a_{i-1} b_{i-1} + (a_{i-1} + b_{i-1}) a_{i-2} b_{i-2} + (a_{i-1} + b_{i-1})(a_{i-2} + b_{i-2}) a_{i-3} b_{i-3} \quad (13)$$

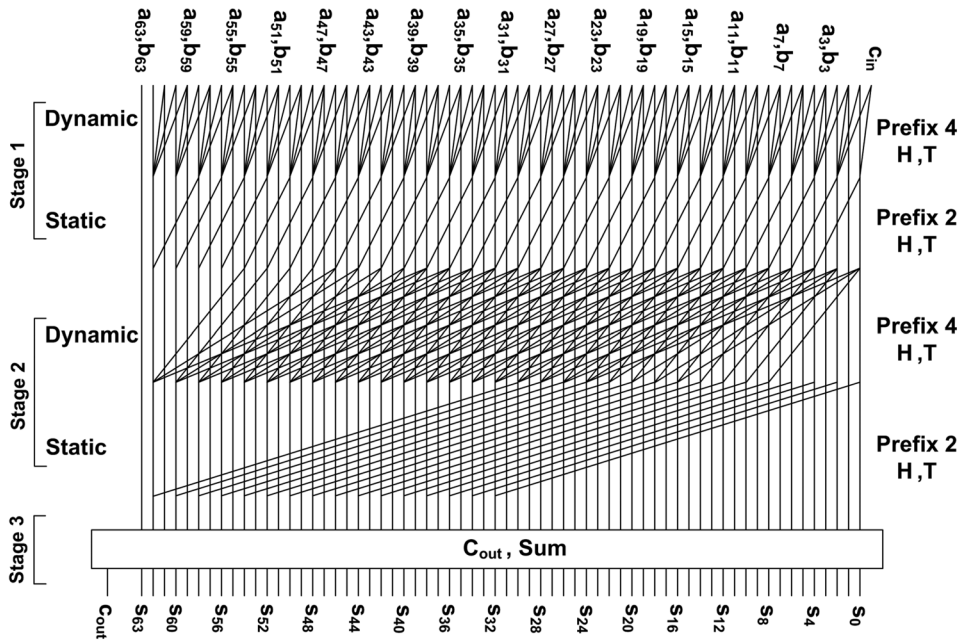

Fig. 10. 64-bit compound domino conditional three-stage *EZO-S2* adder.

$$T_{i-1:i-4} = (a_{i-1} + b_{i-1})(a_{i-2} + b_{i-2}) \times (a_{i-3} + b_{i-3})(a_{i-4} + b_{i-4}). \quad (14)$$

The second compound domino logic stage consists of a prefix-4 dynamic gate followed by a prefix-2 static gate. This structure allows for the pseudo-carry to propagate throughout the entire adder in two compound domino logic stages. The final logic stage performs the sum computation. Carry-out is computed conditionally and selected in the same stage as s_{63}, using h_{62} as the select signal.

The *EZO-S2* (Sparse-2 implementation) adder is optimized for lowest energy at the given speed. It uses conditional three-stage Ling arrangement in order to reduce the amount of wiring and reduce the number of gates as compared to *EZO-S1* by generating every other pseudo-carry without increasing the number of stages (Fig. 10). The logic stages are similar to those of *EZO-S1*. They consist of two compound domino stages and each stage consists of a dynamic prefix-4 followed by a static prefix-2 operation. The third stage is a 2-bit conditional sum selected with the pseudo-carry. The two-bit conditional sum is computed and selected as in (12).

VI. Performance Comparison and Technology Scaling

All results are obtained by characterizing the delay and energy of logic gates in 65 nm, 45 nm, 32 nm, and 22 nm CMOS technologies using the typical process corners. The delay of each logic gate is characterized under worst case single input switching conditions. Predictive Technology Models (PTM) [33] are used. Wire lengths are determined based on the bit-pitch of the register file. The summary of technology characterization is given in Table II.

In VLSI circuits, three types of interconnects are commonly used [28], [29]. Local and intermediate wires are used within logic gate and between gates, respectively. Global wires are used for the purpose of clock and power distribution. As technology scales, the length of local and intermediate wires is also scaled down. However, this is not the case for global wires since the size of the chip grows even in new technologies. In this analysis, local and intermediate wires are considered since global wires are not utilized inside the adder blocks. The bit-pitch is scaled down with respect to the minimal transistor width as shown in Table II. The wire capacitance per length given in Table II is taken from the interconnect report of the International Technology Roadmap for Semiconductors (ITRS) [32].

TABLE II
Technology Characterization Summary

	65nm	45nm	32nm	22nm
Power Supply(V)	1.1	1.0	0.9	0.8
[a]W_{min} (nm)	130	110	96	66
FO4(pS)	20.85	16.88	14.69	12.04
[b]INV Input Cap.(fF)	0.526	0.381	0.295	0.178
Wire Cap.(fF/μm)	0.19	0.18	0.17	0.16
Bit Pitch(μm)	2	1.69	1.47	1.01
Temperature(°C)	25	25	25	25

[a]Minimal transistor width.
[b]Input capacitance for minimum sized inverter.

VII. Results

The energy-delay estimation method presented in [16] is used to analyze the designs in the energy-delay space. The gate sizes of each adder are optimized for energy under a range of operating conditions (delay and input/output loading). The transistor sizing optimization process is iterative, and was performed using MATLAB. Logical Effort delay models were used along with the energy models presented in [16]. Each data point corresponds to a unique optimal sizing of the adder under the corresponding operating condition. In the gate sizing

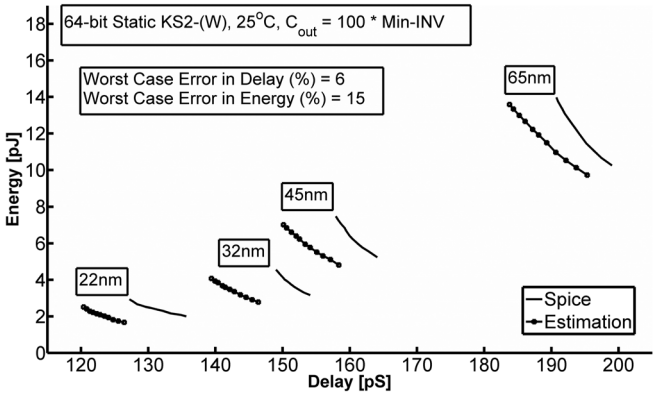

Fig. 11. Estimation versus H-Spice simulation.

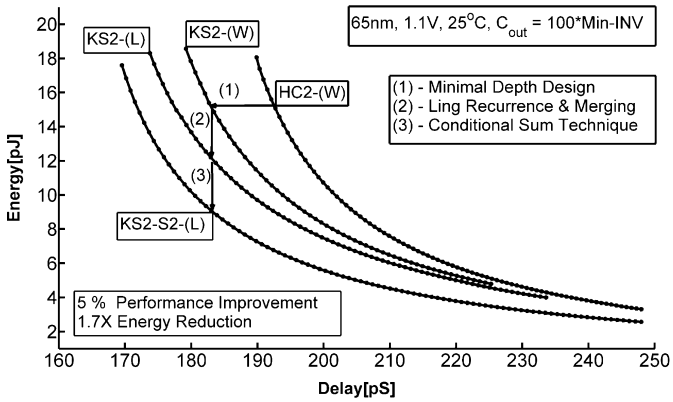

Fig. 12. Comparison of representative 64-bit static CMOS adders at 65 nm technology.

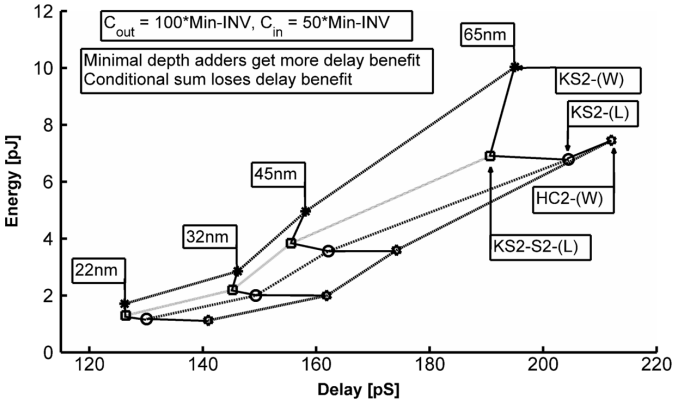

Fig. 13. Representative 64-bit static CMOS adders across CMOS technology nodes.

optimization, the output load is fixed to *100*Min-Inv* and the input loading is swept from *20*Min-Inv* to *100*Min-Inv*. For each input capacitance, the minimum delays are found. Then, the circuits are optimized for minimum energy under the delay constraints found in delay optimization *(delay-constrained energy optimization)*. In this paper, all plots show the energy-minimized points.

Energy consumption of a design directly depends on the switching activity rates of internal nodes. The switching activity rates at internal nodes are complex functions of logic gates and the nature of input vectors [30], [31]. For comparison purposes, temporally and spatially uncorrelated input vectors are assumed to measure switching activity factors of internal nodes. This simplifies the energy estimation. Under those assumptions, the input switching activity rate is 25% for totally random input vectors. The average switching activity rate at internal nodes is around 20% for static adders and around 50% for dynamic adders. The exact switching activity of each node is found by observing the toggles using the switch-level simulator with totally random input vectors. The worst case difference between the energy estimation using the fixed switching activity and the exact switching activity rate is less than 6% for the *64-bit KS2-(W)* adder in all logic families. We used exact switching activity factors in energy optimization for the final results. Therefore, all plots are generated using the exact switching activity rate at each internal node.

To justify the use of our energy-delay estimation method, we compared the results of energy-delay estimation and H-Spice simulation. For energy measurement, totally random input vectors under the aforementioned constraints (25% input switching activity rate) were generated and applied to the adder. Then, measured results were averaged to obtain mean energy consumption. Spice simulation results for the *64-bit static KS2-(W)* adder and the estimation results are sufficiently close, ranging from 65 nm to 22 nm technology nodes. The worst case error between Spice and estimation is less than 6% for delay and less than 15% for energy (Fig. 11).

A. Static CMOS Adders

To determine the potential benefit of methodologies described in Section IV, several static 64-bit adders were implemented. The Weinberger *(W)* recurrence was analyzed on Kogge Stone *(KS2-(W))* and Han Carlson *(HC2-(W))* structures. The Ling *(L)* recurrence was examined on Kogge-Stone *(KS2-(L))* and Sparse-2 *(KS2-S2-(L))* structures. It is possible to achieve 5% performance improvement and 1.7× energy reduction over existing designs (Fig. 12). The scenario when the technology will move to 22 nm is given in Fig. 13.

The merging of the bitwise operation into the first recurrence stage provides lower energy at the same performance in Ling's adder. For Weinberger's adder, it also leads to energy reduction. Weinberger's adder is mostly implemented without merging in the first stage and Ling is mostly implemented with merging because of the technology limitations. Load buffering is applied to all adders and the best ones (with or without external inverter stage) are selected. Merged designs show lower energy consumption at the same performance after adding of an inverter stage to drive the load. However, this is not the case for non-merged designs.

This result demonstrates that the traditional approach for constructing high-performance parallel prefix adders is not energy efficient. The minimum-depth KS and Sparse-2 design shows the best high-performance energy-delay characteristic while the merged adder has the lowest energy consumption at the same performance. The use of Ling recurrence along with the merging and conditional sum techniques provides the lowest energy at the same performance. The methodology presented in Section IV yields the most energy-efficient high-performance static CMOS adder *(KS2-S2-(L))*.

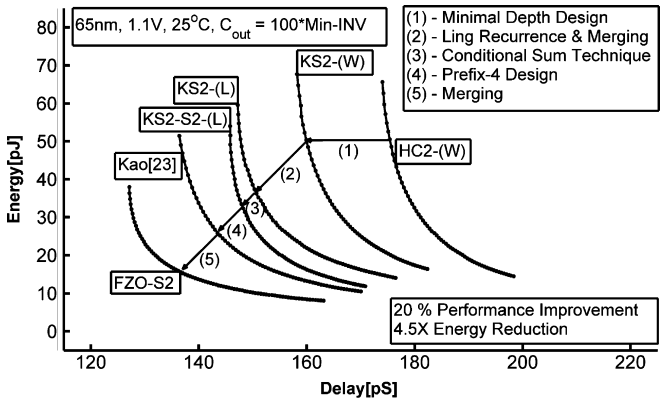

Fig. 14. Comparison of representative 64-bit CMOS domino adders at 65 nm technology.

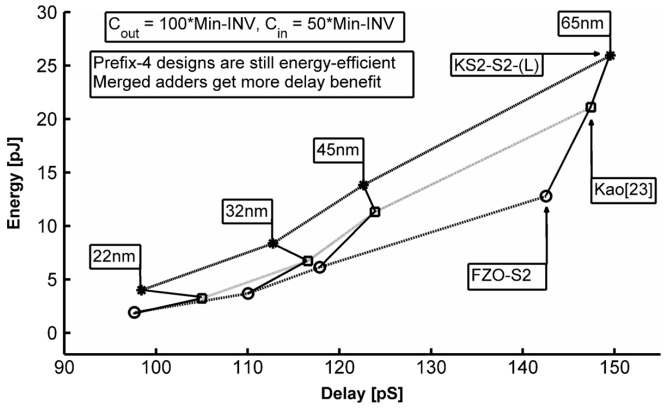

Fig. 15. Representative 64-bit CMOS domino adders across CMOS technology nodes.

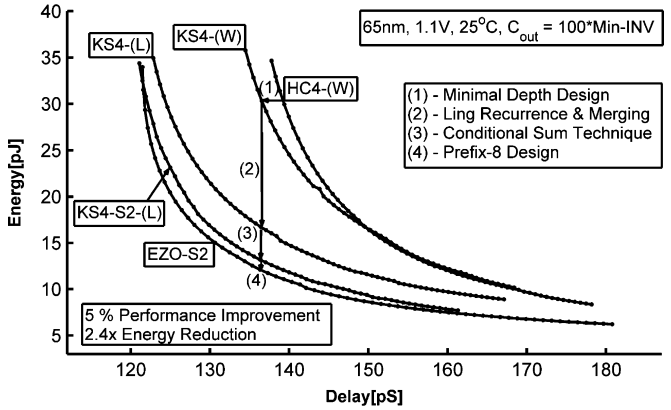

Fig. 16. Comparison of representative 64-bit compound domino CMOS adders at 65 nm technology.

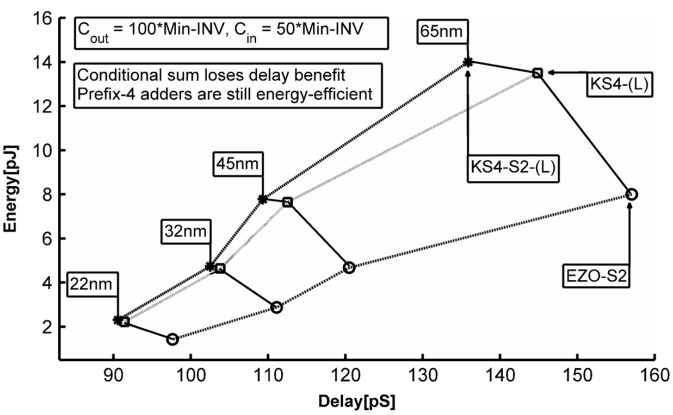

Fig. 17. Representative 64-bit CMOS compound domino adders across CMOS technology nodes.

B. Dynamic CMOS Domino Adders

Dynamic CMOS adders generally have additional constraints in order to maintain functionality compared to static CMOS adders. In dynamic adders, the number of stages must be even (e.g., dynamic followed by static); as well, the first stage of the adder must be footed (the clocked transistor must be part of the nMOS stack), in order to allow for changes on the inputs. Therefore, the impact of techniques suggested for optimal adder design can differ for dynamic adders as compared to static adders.

In dynamic CMOS domino adders, an inverter is required after every dynamic gate, guaranteeing that the number of stages is even. In these designs, the sum computation is performed using static gates, while the carry computation is performed with dynamic gates. The energy-delay results for the 64-bit domino implementations with KS and Sparse2 for both Ling and Weinberger, HC (W), the leading CMOS domino implementation by Kao [23], and the proposed *FZO-S2*, are shown in Fig. 14. The design by Kao [23] is similar to the *FZO-S2* design, however, it does not merge the bitwise operations into the first recurrence stage. The scenario when the technology will move to 22 nm is given in Fig. 15.

The results demonstrate similar tradeoffs as observed in the static 64-bit adders. The Ling adder demonstrates better performance than the Weinberger. The minimum depth designs also demonstrate the best performance, with HC running slower than KS and Sparse-2. The *FZO-S2* adder demonstrates the best energy-delay characteristic in all technology nodes. This adder is implemented using only four CMOS domino logic stages and appears to provide an optimal balance between number of stages, logic gates, and logic gate complexity for CMOS domino adders.

C. Dynamic CMOS Compound Domino Adders

The 64-bit CMOS compound domino realizations of both Ling and Weinberger adders using KS and Sparse-2 structures, along with HC (W) are shown in Fig. 16. The scenario when the technology will move to 22 nm is shown in Fig. 17. The results are similar to those for static CMOS and CMOS domino in that Ling provides better energy-efficient designs over Weinberger, and minimum-depth designs show a performance improvement over HC.

Using CMOS compound domino does not guarantee that merging will remove a logic stage from the adder. While merged 64-bit KS and Sparse-2 CMOS compound domino realizations eliminate a stage (reducing the number of stages from five to four), the Han-Carlson design, which merges the bitwise operations into the first recurrence stage, and the one that does not each have five compound domino stages. This is because the carry-tree in HC requires eight logic stages when implemented without merging and seven when implemented with merging.

However, in CMOS compound domino logic only an even number of stages can be used. Therefore, each design requires four compound domino logic stages for computing carry and one stage for computing sum. The result is that the merged Han-Carlson design suffers a significant performance penalty compared to the design without merging. This effect varies depending on the size of the adder. For example, in a 32-bit adder both types using Kogge-Stone structure (with and without merging) will have the same number of dynamic–static stages, while the Han-Carlson design, which merges the operations, will have one less dynamic–static stage than the design that does not. This appears to be the only situation when merging the bitwise operation into the first recurrence stage does not yield the most energy-efficient realization of the structure.

VIII. Conclusion

By taking advantage of possible energy-delay tradeoffs on various levels, algorithm, recurrence, wire delay and energy, circuit sizing and circuit topology, we have developed a 64-bit adder *(EZO-S2)* which operates at the highest achievable speed using up to four times less energy over leading CMOS implementations (e.g., Kao [23]). The energy benefit of using Ling's algorithm in the first recurrence stage was demonstrated in static adders and dynamic CMOS adders. CMOS compound domino adders demonstrated further savings dependent on the ability to reduce the number of stages in the adder. Additionally, we demonstrated that minimum-depth structures are best for high-performance adders. Introducing conditional sum in the last stage reduces the density of the wiring tracks. The portion of the energy saved in the wires is traded for speed improvement in the conditional stage, while the remaining portion is applied for overall speed improvement. The applied tradeoffs were used to design energy-efficient high-performance 64-bit static and 64-bit dynamic adder. We further examined how these tradeoffs change when the technology scales down. By doing so, we achieved 1.7× energy saving and a 5% performance improvement in static adders and 20% faster design and 4.5× energy reduction in domino adders at 65 nm technology. Implementation in compound domino family yields 2.4× energy reduction in the best performance range. We also generated the energy-delay characteristics of proposed and existing adders across all technology nodes. The results show that the given methodologies lead to the best design at all technology nodes.

Acknowledgment

The authors thank Dr. Mustafa Aktan for optimization tool support and Dr. Matthew Sanu of Intel Corporation for his ideas and contributions.

References

[1] A. Weinberger and J. L. Smith, "A logic for high-speed addition," *Nat. Bur. Stand. Circ.*, vol. 591, pp. 3–12, 1958.

[2] S. Winograd, "On the time required to perform addition," *J. ACM*, vol. 12, no. 2, pp. 277–285, Apr. 1965.

[3] H. Ling, "High-speed binary adder," *IBM J. Res. Devel.*, vol. 25, no. 3, pp. 156–166, May 1981.

[4] R. W. Doran, "Variants of an improved carry look-ahead adder," *IEEE Trans. Comput.*, vol. 37, no. 9, pp. 1110–1113, Sep. 1988.

[5] M. Lehman and N. Burla, "Skip techniques for high-speed carry propagation in binary arithmetic units," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 691–698, Dec. 1961.

[6] O. J. Bedrij, "Carry-select adder," *IRE Trans. Electron. Comput.*, vol. EC-11, pp. 340–346, 1962.

[7] J. Sklanski, "Conditional-sum addition logic," *IRE Trans. Electron. Comput.*, vol. EC-9, no. 2, pp. 26–231, 1960.

[8] V. G. Oklobdzija and E. R. Barnes, "Some optimal schemes for ALU implementation in VLSI technology," in *Proc. 7th Symp. Computer Arithmetic, ARITH-7*, pp. 2–8, reprinted in *Computer Arithmetic*, E. E. Swartzlander, Ed., vol. II, pp. 137-1423, 1985.

[9] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Comput.*, vol. C-22, no. 8, pp. 786–793, Aug. 1973.

[10] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Trans. Comput.*, vol. C-31, no. 3, pp. 260–264, Mar. 1982.

[11] T. Han and D. A. Carlson, "Fast area-efficient VLSI adders," in *Proc. 8th IEEE Symp. Computer Arithmetic*, Como, Italy, May 1987, pp. 49–56.

[12] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *J. ACM*, vol. 27, no. 4, pp. 831–838, Oct. 1980.

[13] S. Knowles, "A family of adders," in *14th IEEE Symp. Computer Arithmetic*, Adelaide, Australia, Apr. 14–16, 1999.

[14] S. K. Mathew *et al.*, "A 4 GHz 130 nm address generation unit with 32-bit sparse-tree adder core," in *Symp. VLSI Circuits Dig. Tech. Papers*, 2002, pp. 126–127.

[15] H. Q. Dao, B. R. Zeydel, and V. G. Oklobdzija, "Energy optimization of pipelined digital systems using circuit sizing and supply scaling," *IEEE Trans. VLSI Syst.*, vol. 14, no. 2, pp. 122–134, Feb. 2006.

[16] V. G. Oklobdzija, B. R. Zeydel, H. Q. Dao, S. Mathew, and R. Krishnamurthy, "Comparison of high-performance VLSI adders in energy-delay space," *IEEE Trans. VLSI Syst.*, vol. 13, no. 6, pp. 754–758, Jun. 2005.

[17] V. Zyuban and P. N. Strenski, "Balancing hardware intensity in microprocessor pipelines," *IBM J. Res. Develop.*, vol. 47, no. 5/6, pp. 585–598, Sep./Nov. 2003.

[18] D. Markovic, V. Stojanovic, B. Nikolic, M. Horowitz, and R. Brodersen, "Methods for true energy-performance optimization," *IEEE J. Solid-State Circuits*, vol. 39, no. 8, pp. 1282–1293, Aug. 2004.

[19] G. Dimitrakopoulos and D. Nikolos, "High-speed parallel-prefix ling adders," *IEEE Trans. Comput.*, vol. 54, no. 2, pp. 225–231, Feb. 2005.

[20] B. R. Zeydel, T. Kluter, and V. G. Oklobdzija, "Efficient mapping of addition recurrence algorithms in CMOS," in *17th IEEE Symp. Computer Arithmetic*, Cape Cod, MA, Jun. 2005.

[21] J. Park *et al.*, "470 ps 64-bit parallel binary adder," in *Symp. VLSI Circuits Dig. Tech. Papers*, 2000, pp. 192–193.

[22] S. Naffziger, "A sub-nanosecond 0.5 μm 64-b adder design," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 1996, pp. 362–363.

[23] S. Kao, R. Zlatanovici, and B. Nikolic, "A 240ps 64-bit carry-lookahead adder in 90 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC'06)*, San Francisco, CA, Feb. 2006.

[24] J. D. Warnock *et al.*, "The circuit and physical design of the POWER4 microprocessor," *IBM J. Res. Devel.*, vol. 46, no. 1, pp. 27–51, Jan. 2002.

[25] I. E. Sutherland, R. F. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. Boston, MA: Morgan Kaufmann, c1999.

[26] V. G. Oklobdzija and B. R. Zeydel, "Energy-delay characteristics of CMOS adders," in *High-Performance Energy-Efficient Microprocessor Design*, V. G. Oklobdzija and R. K. Krishnamurthy, Eds. New York: Springer, Jul. 2006.

[27] V. G. Oklobdzija, B. R. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, "Energy-delay estimation technique for high-performance microprocessor VLSI adders," in *16th IEEE Symp. Computer Arithmetic*, Santiago de Compostela, Spain, Jun. 15–18, 2003.

[28] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, Apr. 2001.

[29] R. Ho, K. Mai, and M. Horowitz, "Managing wire scaling: A circuit perspective," in *Proc. IEEE Interconnect Technology Conf. 2003*, Jun. 2003, pp. 177–179.

[30] D. Baran, M. Aktan, H. Karimiyan, and V. G. Oklobdzija, "Exploration of switching activity behavior of addition algorithms," in *MWSCAS 2009*, Cancun, Mexico, Aug. 2–5, 2009.

[31] D. Baran, M. Aktan, H. Karimiyan, and V. G. Oklobdzija, "Switching activity calculation of VLSI adders," in *ASICON 2009*, Changsha, China, Oct. 20–23, 2009.

[32] ITRS (International Technology Roadmap for Semiconductors) [Online]. Available: http://www.itrs.net/Links/2007ITRS/2007Chapters/2007Interconnect.pdf
[33] Predictive Technology Model (PTM) website. Nanoscale Integration and Modeling (NIMO) Group, Arizona State Univ., Tempe, AZ, 2007 [Online]. Available: http://www.eas.asu.edu/~ptm/
[34] D. Patil, O. Azizi, M. Horowitz, R. Ho, and R. Ananthraman, "Robust energy-efficient adder topologies," in *Proc. 18th IEEE Symp. Computer Arithmetic, ARITH'07*, Jun. 25–27, 2007, pp. 16–28.

Bart R. Zeydel (S'00–M'05) received the B.S. degree in 2001, and the Ph.D. degree in electrical and computer engineering from the University of California, Davis, in 2005.

In 2000, he worked at Mentor Graphics on the VRTX real-time operating system. In 2001, he worked at Fujitsu Microelectronics where he designed datapath elements for a VLIW processor, and at Telairity, where he developed portable hard-IP datapath blocks. In 2003, he was an intern at Intel Corporation's Circuits Research Laboratories, Hillsboro, Oregon, where he designed datapath elements for DSP and wireless products. Currently, he is with Plato Networks, where he designs datapath elements for a 10GBASE-T PHY. His research interests include high-performance and low-power datapath circuits, design methodologies for energy-efficient high-performance and low-power digital circuits, and the development of CAD tools for design in the energy-delay space.

Dursun Baran (S'09) was born in Tokat, Turkey, in 1984. He received the B.S. degree in electrical engineering from Bogazici University, Istanbul, Turkey, in 2007, and the M.S.E.E. degree from the Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas, Richardson, TX, in 2009. He is currently pursuing the Ph.D. degree in electrical engineering at the same school.

In 2007, he joined the Advanced Computer Systems Engineering Laboratory (ACSEL), The University of Texas at Dallas, working in the area of high performance and low power digital circuits. In the past he has designed wireless devices at Pozitif Electronics Inc., Istanbul, Turkey. His research interest is in the optimization and design of energy-efficient circuits.

Vojin G. Oklobdzija (S'78–M'82–SM'88–F'96) received the Dipl.Ing. degree from the School of Electrical Engineering, University of Belgrade, in 1971, and the Ph.D. degree from the University of California at Los Angeles in 1982. During his Ph.D. study, he worked at Xerox Corporation Microelectronic division and was involved in early workstation development of Xerox Alto.

From 1982 to 1991, he was with the IBM Thomas J. Watson Research Center, where he made contributions to the development of the first RISC processors, super-scalar and supercomputer design. In the course of this work, he obtained several patents, the most notable one on register renaming, which enabled a new generation of modern computers. From 1988 to 1990, he was an IBM visiting faculty member at the University of California at Berkeley, Professor and Emeritus Professor at the University of California at Davis, 1991–2006, Chair Professor at Sydney University, 2006–2007, and currently at The University of Texas at Dallas. He has actively served as a consultant for many companies including Sun Microsystems, Bell Laboratories, Texas Instruments, Hitachi, Fujitsu, Sony, Intel, Samsung, and Siemens Corporation (as a principal architect for the Infineon TriCore processor). He has published 170 papers, six books and a dozen book chapters in the areas of circuits and technology, computer arithmetic and computer architecture. His book *Computer Engineering* won Outstanding Academic Title award, out of 22,000 titles considered and is currently in its second edition. He has given over 200 invited talks and short courses in the USA, Europe, Latin America, Australia, China and Japan. He holds 15 U.S., six European, six Japanese, and six international patents.

Prof. Oklobdzija is serving on the Editorial Board of IEEE MICRO and the publishing board of Taylor-Francis, IEEE Fellow Committee, and numerous other IEEE committees. He served as an Associate Editor of IEEE TRANSACTIONS ON COMPUTERS from 2000 to 2006, IEEE TRANSACTIONS ON VLSI from 1995 to 2003, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II and *Journal of VLSI Signal Processing*, the ISSCC program committee from 1996 to 2003 and again in 2007, First Asian A-SSCC, International Symposium on Low-Power Design, Computer Arithmetic, ICCD, PATMOS, and numerous other conference committees. Currently, he is General Chair for the International Symposium on Low Power Design (ISLPED 2010) and 20th International Symposium on Computer Arithmetic (ARITH-20). He was a General Chair of the ARITH-13 (1997), DCAS-2008, Technical Program Chair for ISLPED 2008, and Track Chair for ICCD 2008. He is a Distinguished Lecturer of the IEEE Solid-State Circuits Society, Vice-President of IEEE CAS ,and a member of the IEEE CAS Board of Governors.

Dr. Oklobdzija has provided litigation consulting and expert witness services to major legal firms in USA and abroad including Townsend and Townsend, Arent Fox, Kellogg Huber, Dechert LLP, DLA Piper US LLP and Farella Braun Martel LLP. He directs the ACSEL Laboratory which is involved in digital circuits optimization for low-power, ultra low-power and high-performance systems. (http://www. acsel-lab.com)