TABLE IV
EXPERIMENTAL RESULTS OF NON-ZERO PRESCRIBED SKEW BUFFERED CLOCK ROUTINGS. TOTAL CAPACITANCE INCLUDES WIRE AND BUFFER CAPACITANCE IN pF. THE CPU TIME IS IN SECONDS

Case	Algorithm	Wirelen	#bufs	Total cap	CPU
r1	NS+	2917614	49	59.5	1
	MAT-MIC	1293616	13	26.2	1
r2	NS+	5881313	93	119.8	13
	MAT-MIC	2541324	25	51.4	1
r3	NS+	7287088	112	148.4	42
	MAT-MIC	3265058	28	66.0	1
r4	NS+	16276930	474	331.1	474
	MAT-MIC	6659865	62	134.6	3
r5	NS+	24933092	1968	507.7	1968
	MAT-MIC	9860004	99	199.5	10
Ave. improvement		59.6%	69.0%	60.0%	99.3%

order estimation metric in [1], slew rate for step input is roughly $\ln 9 \times Elmore\_delay$. Therefore, the slew rate at the sinks for step input is usually less than 385 ps. Please note that this metric is a conservative estimation. For ramp input with 100 ps slew rate, the slew rate at the sinks is usually no greater than 400 ps according to the metric developed in [12]. The CPU time are shown in the rightmost column of Table IV.

V. CONCLUSION

Even though traditional zero skew routing methods can be applied to achieve nonzero skews, they may bring huge wire and buffer area overhead as the difference among sink delay targets are ignored in their merging schemes. We propose a maximum delay-target-based merging scheme for general prescribed skew routings. Experimental results on benchmark circuits show that the proposed technique is very effective and efficient on minimizing clock network size for prescribed skews.

REFERENCES

[1] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990.

[2] W.-C. D. Lam, C.-K. Koh, and C.-W. A. Tsao, "Power supply noise suppression via clock skew scheduling," in *Proc. IEEE Int. Symp. Quality Electronic Design*, 2002, pp. 355–360.

[3] I. S. Kourtev and E. G. Friedman, "Clock skew scheduling for improved reliability via quadratic programming," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1999, pp. 239–243.

[4] R.-S. Tsay, "Exact zero skew," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1991, pp. 336–339.

[5] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng, "Zero skew clock routing with minimum wirelength," in *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, Nov. 1992, pp. 799–814.

[6] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao, "Bounded-skew clock and Steiner routing," *ACM Trans. Design Automation of Electronic Systems*, vol. 3, no. 3, pp. 341–388, Jul. 1998.

[7] M. Edahiro, "A clustering-based optimization algorithm in zero-skew routings," in *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 612–616.

[8] C.-W. A. Tsao and C.-K. Koh, "UST/DME: a clock tree router for general skew constraints," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 2000, pp. 400–405.

[9] A. Takahashi, K. Inoue, and Y. Kajitani, "Clock-tree routing realizing a clock-schedule for semi-synchronous circuits," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1997, pp. 260–265.

[10] S. Held, B. Korte, J. Maßberg, M. Ringe, and J. Vygen, "Clock scheduling and clock tree construction for high performance ASICs," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 2003, pp. 232–239.

[11] J. Chung and C.-K. Cheng, "Skew sensitivity minimization of buffered clock tree," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1994, pp. 280–283.

[12] C. V. Kashyap, C. J. Alpert, F. Liu, and A. Devgan, "Closed-form expressions for extending step delay and slew metrics to ramp inputs for *RC* trees," *IEEE Trans. Computer-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 4, pp. 509–516, Apr. 2004.

Comparison of High-Performance VLSI Adders in the Energy-Delay Space

Vojin G. Oklobdzija, Bart R. Zeydel, Hoang Q. Dao, Sanu Mathew, and Ram Krishnamurthy

***Abstract*—In this paper, we motivate the concept of comparing very large scale integration adders based on their energy-delay characteristics and present results of our estimation technique. This stems from a need to make appropriate selection at the beginning of the design process. The estimation is quick, not requiring extensive simulation or use of computer-aided design tools, yet sufficiently accurate to provide guidance through various choices in the design process. We demonstrate the accuracy of the method by applying it to examples of high-performance 32- and 64-b adders in 100- and 130-nm CMOS technologies.**

***Index Terms*—Adders, digital arithmetic, digital circuits, energy-delay optimization.**

I. INTRODUCTION

In the very large scale integration (VLSI) design process, selection of the initial topology expected to yield a desired performance in the allotted power budget is the most important step taken. However, the exact performance and power will be known only after a time consuming design and simulation process is completed. Therefore, the validity of the initial selection will not be known until late in the design process. Going back and forth between several choices is often prohibited by design schedule, making it impossible to correct mistakes committed at the beginning. Therefore an uncertainty always remains as to whether a higher performance or lower power could have been achieved using a different topology. This problem is aggravated by a lack of proper delay and power estimation techniques that are guiding development of computer arithmetic algorithms. The majority of algorithms used today are based on out-dated methods of counting the number of logic gates on the critical path, producing inaccurate and misleading results. The importance of transistor sizing, load effects and power are not taken into account by most.

Knowles has shown how different adder topologies may influence fan-out and wiring density, thus, influencing design decisions and yielding better area/power tradeoffs than known cases [1]. This work further emphasized the disconnect existing between algorithms and implementation. In our previous work, the importance of fan-in and fan-out effects on the critical path was demonstrated [2]. This led to similar conclusions expressed in the logical effort (LE) method of Sutherland and Sproull [3] regarding critical-path delay estimation. This method has been introduced into common practice by Harris [4]. To gain confidence in the method we compared LE delay estimate of various VLSI adders to simulation results [5] obtained using H-SPICE [17]. The matching was well under 10% in most cases (Table I). However, given that delay and energy can be traded against each other, inclusion of energy in the analysis might have resulted in different ranking.

Manuscript received July 20, 2004; revised December 6, 2004. This work was supported in part by the SRC under Research Grant 931.001 and in part by California MICRO 03-069.

V. G. Oklobdzija, B. R. Zeydel, and H. Q. Dao are with the Advanced Computer Systems Engineering Laboratory (ACSEL), Electrical and Computer Engineering Department, University of California, Davis, CA 95616 USA (e-mail: zeydel@acsel-lab.com).

S. Mathew and R. Krishnamurthy are with the Microprocessor Research Laboratories, Intel Corporation, Hillsboro, OR 97124 USA.

Digital Object Identifier 10.1109/TVLSI.2005.848819

TABLE I
DELAY COMPARISON OF 64-b ADDERS USING LOGICAL EFFORT

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

In this paper we present an estimation method which allows for the energy-delay tradeoffs of a design to be explored. Using this method we show comparison results in the energy-delay space, which in our view is the only proper way to compare designs (see Fig. 1). This estimation method has to meet two contradictory requirements: it has to be simple and quick, yet yield sufficient accuracy to guarantee appropriate selection.

The paper is organized as follows. Section II discusses approaches to comparing VLSI adders. Section III presents delay estimation based on LE and an energy model for LE sizing. Section IV describes the developed energy-delay estimation (EDE) method, which allows for comparison of adders in the energy-delay space. Section V shows EDE comparison of leading high-performance VLSI adders in 100- and 130-nm CMOS technology.

II. COMPARISON OF VLSI ADDERS

The most common approach for comparing VLSI adders is the use of a single delay point [1], [2]. Such a delay comparison of high-performance 64-b adders is shown in Table I. The results are shown for LE delay estimates and H-SPICE prelayout simulation in 130-nm technology.

These results show a significant difference between static CMOS and dynamic CMOS implementations. This fact has not been unnoticed by the practitioners, thus, high-speed processors all use dynamic CMOS implementations [10], [11], [14].

While the delay difference between different circuit families is apparent, the delay difference between topologies using the same circuit family is relatively small, making it difficult to know which design can stretch further in the energy-delay space. Energy is also important because if too much power is used to achieve a target delay, hot spots can be created [6], [14].

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

As the curves show, adder B has more room for delay improvement and can achieve lower energy in the high-performance region (Region 1).

On the other hand, if lower computational energy is the design objective, adder A is the better choice as it can achieve lower energy in the low-performance region (Region 2).

The challenge is to create such comparison curves early in the design process without significant time overhead, so that they can be used to guide our selection of the appropriate topology or algorithm. Our objective was to develop a method for estimating energy and delay that can be applied in a short amount of time, yet provides sufficient accuracy for tradeoff analysis.

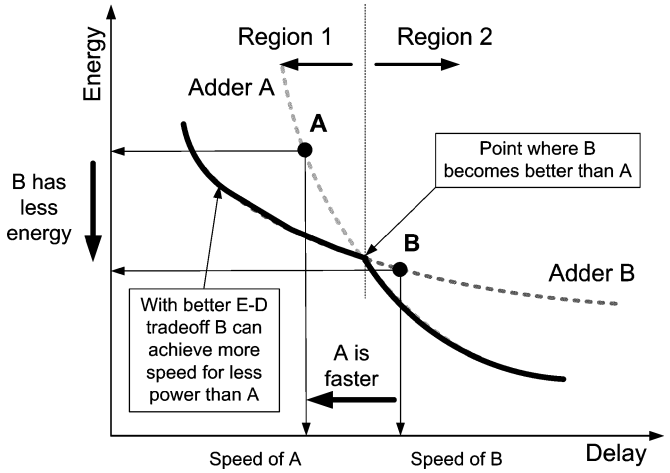

Fig. 1. Energy-delay dependency.

TABLE II
POTENTIAL DELAY IMPACT OF WIRE RESISTANCE IN 130 nm

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

III. DELAY AND ENERGY ESTIMATION

The speed of a VLSI adder depends on several factors: technology, circuit family, adder topology, transistor sizes, and second-order effects. As a result there are no simple rules to be applied when estimating delay. Skilled engineers are capable of fine-tuning the design to obtain the best performance and lowest energy through transistor sizing. However, this is often an ad hoc process. Thus, it is difficult, if not impossible, to predict the best topology. The introduction of LE was a significant step forward because it provided a better way to estimate delay. Further, LE provides an optimal sizing for delay.

A. Delay Estimation

There is a tradeoff in delay estimation where achieving improved accuracy is paid for by complexity of estimation, or the need to resort to computer-aided design (CAD) tools. LE simplifies the delay model to a single parameter referred to as *stage effort* f, which is used during optimization and modeling. The LE model for gate delay is $t_d = (f+p)\tau$, where $f = gh$ [3]. Each gate has an LE, g, which represents its drive capability relative to an inverter. The term h represents the effective fan-out of the gate $(C_{\rm out}/C_{\rm in})$. The parasitic delay, p, corresponds to the delay associated with parasitic capacitances. The term τ is the per fan-out delay increment of an inverter and is used to introduce technology independent estimation of delay.

The accuracy of LE can be improved by obtaining the coefficients g and p through H-SPICE characterization. This step incorporates characteristics of a particular technology, slopes, and layout estimates into the LE parameters. Gate characterization is performed under the constraint of fixed input–output slope relationship to obtain the best matching.

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

Application of LE to simple path delay estimation and size optimization is straightforward [4]; however, it is often difficult to apply the analysis to complex paths due to branching. LE defines branching as $b = (C_{\rm on} + C_{\rm off})/C_{\rm on}$, where the terms $C_{\rm on}$ and $C_{\rm off}$ must be determined relatively. This analysis becomes prohibitively complex when branches have differing gate types and number of stages. In addition, constant loads such as wiring require iterative computation. As a result the optimization of a complex path using the LE gate delay model must be performed by changing individual *f's* of each gate to achieve minimal delay. We chose to use MS-Excel instead of a simple paper and pencil method (as suggested by LE) because of its built-in gradient based optimization feature, and the fact that it requires only slightly more overhead than back of the envelope analysis.

B. Energy Estimation

The use of LE for delay optimization provides not only a delay estimate, but the corresponding gate sizing. By modeling the energy of each gate from its LE sizing, the total energy of a design can be estimated.

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

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

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

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

IV. EDE Method

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

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

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

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

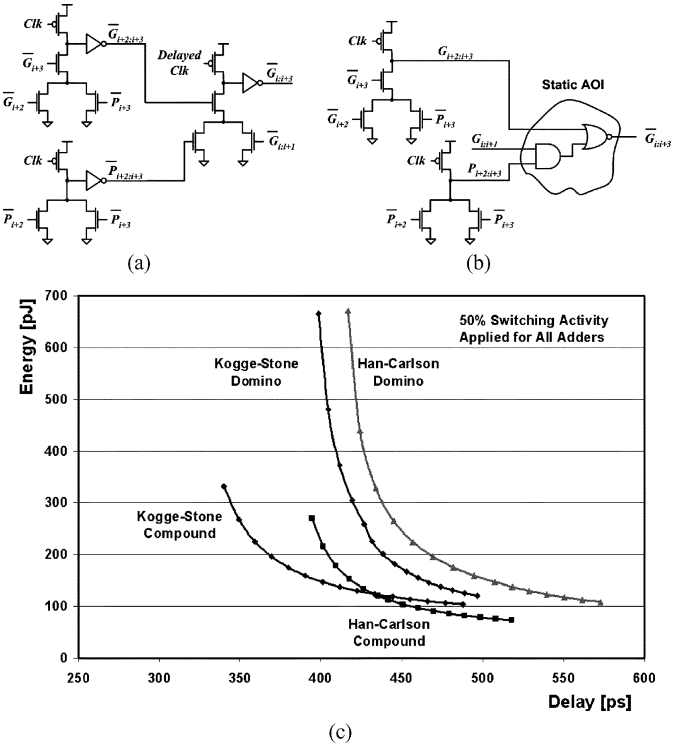

Fig. 2. (a) Carry merge: domino implementation. (b) Carry merge: compound domino implementation. (c) Comparison of 64-b HC and KS domino and compound-domino adders in 130-nm technology.

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

The energy of each gate also depends on its switching activity. In our application of EDE to VLSI adders, we chose to use a 15% switching activity factor for each gate in the static adders and a 50% switching activity for each gate in the dynamic adders. These switching factors were obtained as an average of designs that were analyzed and consistent with the "rule of thumb" used in industry and based on experience.

V. EDE Comparison of Adders

EDE was used to compare the design of Intel's Quaternary-Tree (QT) Adder [14] to other viable high-performance options such as Kogge–Stone (KS) [7] and Han–Carlson (HC) [9] implemented in: static, domino, and compound-domino CMOS. The analysis was to provide an answer to whether compound-domino QT was a valid choice given the power limitations and speed requirements of the given 32-b processor.

A. Domino and Compound Domino CMOS Analysis

The difference between domino and compound-domino CMOS logic is illustrated in the case of the carry-merge stage. Generally, in order to improve performance, domino CMOS logic has been used for implementation of carry-merge blocks resulting in the circuit shown in Fig. 2(a). The static CMOS inverter is necessary after each dynamic block in order to make the logic behave in a "domino" fashion. It

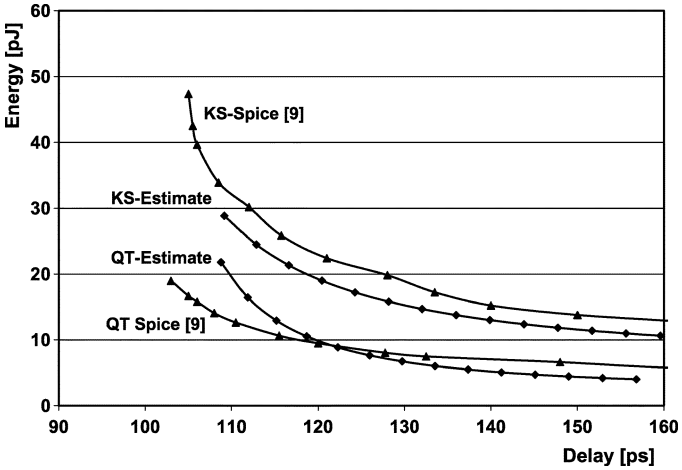

Fig. 3. Comparison of 32-b QT and KS adders. EDE versus simulation in 100-nm technology.

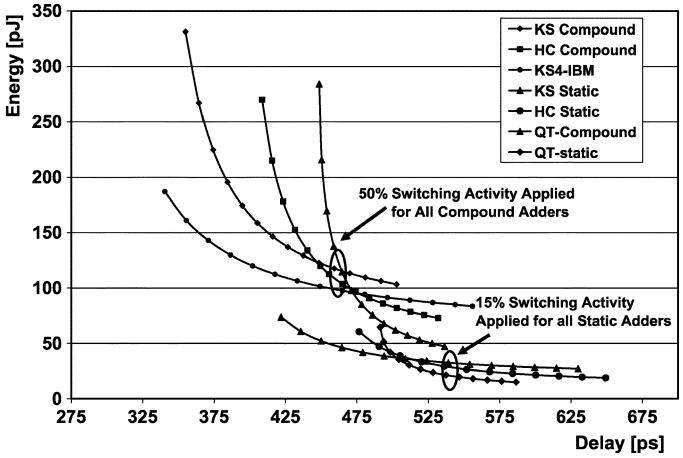

Fig. 4. EDE analysis of 64-b compound-domino and static adders in 130-nm technology.

was realized that the inversion of the signal, which is necessary in the domino CMOS logic block, can be replaced with a more complex inverting static gate, referred to as compound-domino. Thus, two domino carry-merge stages can be merged into one by replacing the inverter with an and-or-invert (AOI) [Fig. 2(b)].

The EDE estimations for 64-b HC and KS compound-domino and domino adders are shown in Fig. 2(c). Comparing domino to compound-domino, EDE helps us to appreciate the benefits obtained by utilizing compound-domino logic. For the same energy budget (e.g., 200 pJ) compound-domino KS yields 20% delay improvement over domino KS. EDE provides a clear picture of the impact compound-domino can have on adder design.

B. Adder Comparison

The ability, provided by EDE method, to observe differences between implementations of the same adder using different circuit families is beneficial in selecting appropriate circuit design style. However it is also important to see tradeoffs between adder topologies implemented using the same circuit family. The adequacy of EDE for demonstrating tradeoffs in the energy-delay space was analyzed by comparing optimized H-SPICE results for 32-b compound-domino KS and QT adders versus EDE results in 100-nm technology. The optimized H-SPICE simulation results for KS and QT were available only in 130 nm. In order to compare the simulation results with EDE estimation adopted for 100 nm, we used simple technology scaling rule of: 50% energy and 30% delay [13]. The results are shown in Fig. 3.

For the 32-b designs in Fig. 3 the EDE estimates demonstrate the same tradeoffs as observed in simulation. These results confirmed the validity of our estimation as well as the selection of the QT adder as a viable option for reducing energy without sacrificing performance.

Given the accuracy of the 32-b comparison, compound-domino and static 64-b adders were analyzed using EDE to see what tradeoffs exist (Fig. 4). Different points on the energy-delay curve were obtained by varying the size of the input gates of each adder. The output of each adder was loaded with a 1 mm wire. A range of H was chosen from $H = 2$ to the maximum H (i.e., where minimum input size occurs).

The results show the benefits associated with sparse designs (HC and QT) [10], [14], however the benefits of compound-domino QT are lesser at 64-b than for the 32-b design. This is a result of the one extra stage that the QT implementation used versus the KS implementation (in the 32-b design the QT implementation used the same number of stages as KS). We also observe that a compound-domino KS prefix-4 adder (KS4-IBM) [12] which utilizes fewer stages at the cost of increased gate complexity and branching shows further benefits. The increased gate complexity in the KS4-IBM adder is offset by a significant reduction in parasitic delay associated with the number of stages, which allows for the KS4-IBM to achieve lower delay for less energy penalty than the other designs. At lower performance targets this overhead is too great and designs such as QT and HC achieve lower energy.

VI. Conclusion

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

References

[1] S. Knowles, "A family of adders," in *Proc. 14th Symp. Computer Arithmetic*, Adelaide, Australia, Apr. 1999, pp. 30–34.

[2] V. G. Oklobdzija and E. R. Barnes, "On implementing addition in VLSI technology," *J. Parallel Distrib. Comput.*, vol. 5, pp. 716–728, 1988.

[3] I. E. Sutherland and R. F. Sproull, "Logical effort: designing for speed on the back of an envelop," in *Advanced Research in VLSI*, C. Sequin, Ed. Cambridge, MA: MIT Press, 1991.

[4] I. E. Sutherland, R. F. Sproull, and D. Harris, *Logical Effort Designing Fast CMOS Circuits*. San Mateo, CA: Morgan Kaufmann, 1999.

[5] H. Q. Dao and V. G. Oklobdzija, "Application of logical effort techniques for speed optimization and analysis of representative adders," in *Proc. 35th Annu. Asilomar Conf. Signals, Systems, and Computers*, 2001, pp. 1666–1669.

[6] D. Harris and S. Naffziger, "Statistical clock skew modeling with data delay variations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 4, pp. 888–898, Dec. 2001.

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

[8] A. Farooqui, V. G. Oklobdzija, and F. Chehrazi, "Multiplexer based adder for media signal processing," in *Proc. Int. Symp. VLSI Technology, Systems, and Applications*, Taipei, Taiwan, R.O.C., Jun. 1999, pp. 100–103.

[9] T. Han, D. A. Carlson, and S. P. Levitan, "VLSI design of high-speed low-area addition circuitry," in *Proc. IEEE Int. Conf. Computer Design: VLSI in Computers and Processors*, 1987, pp. 418–422.

[10] S. K. Mathew *et al.*, "Sub-500-ps 64-b ALU's in 0.18 μm SOI/bulk CMOS: design and scaling trends," *IEEE J. Solid-State Circuits*, vol. 36, no. 11, pp. 1636–1646, Nov. 2001.
[11] S. Naffziger, "A sub-nanosecond 0.5 μm 64-b adder design," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Technical Papers*, Feb. 1996, pp. 362–363.
[12] J. Park *et al.*, "470 ps 64-bit parallel binary adder," in *Proc. Symp. VLSI Circuits Dig. Tech. Papers*, pp. 192–193.
[13] B. Davari, R. H. Dennard, and G. G. Shahidi, "CMOS scaling for high performance and low power—The next ten years," *Proc. IEEE*, vol. 83, no. 4, pp. 595–606, Apr. 1995.
[14] S. K. Mathew *et al.*, "A 4 GHz 130 nm address generation unit with 32-bit sparse-tree adder core," *IEEE J. Solid-State Circuits*, vol. 38, no. 5, pp. 689–695, May 2003.
[15] V. G. Oklobdzija, B. R. Zeydel, H. Q. Dao, S. Mathew, and R. Krishnamurthy, "Energy-delay estimation technique for high-performance microprocessor VLSI adders," in *Proc. 16th Int. Symp. Computer Arithmetic*, Santiago de Compostela, Spain, Jun. 2003, pp. 272–279.
[16] V. G. Oklobdzija, *High-Performance System Design: Circuits and Logic*. Piscataway, NJ: IEEE Press, 1999.
[17] HSPICE User's Guide. [Online] Available: www.synopsys.com

A Novel FPGA Architecture Supporting Wide, Shallow Memories

Steven W. Oldridge and Steven J. E. Wilton

***Abstract*—This paper investigates an architecture designed to implement wide, shallow memories on a field programmable gate array (FPGA). In the proposed architecture, existing configuration memory normally used to control the connectivity pattern of the FPGA is made user accessible. Typically, not all the switch blocks in an FPGA are used to transport signals. By adding only a modest amount of circuitry, the configuration memory in these unused switch blocks (or unused paths within used switch blocks) can be used to implement wide, shallow buffers and other similar memory structures. The size of FPGA required to implement a benchmark circuit that makes use of the wide, shallow memories, is 20% smaller than a standard memory architecture. In addition, the benchmark circuit is on average 40% faster using the proposed architecture.**

***Index Terms*—Embedded memory, field programmable gate arrays (FPGAs).**

I. Introduction

In the past ten years, the capacity of field programmable gate arrays (FPGAs) has grown to such an extent that they are now being used to implement entire systems, and represent an alternative to system-on-chip (SoC) and application specific integrated circuit (ASIC) development. Logic and memory resources on the FPGA have reached levels such that they are capable of providing a complete, one-chip solution. Input/output (I/O) capabilities have also followed this trend, most notably with the introduction of high-speed I/O (such as RapidIO, POS-PHY Level 4, or UTOPIA IV) [1], [2] that allow gigahertz range signals to access an FPGA running at a lower frequency. High-speed I/O has also allowed users to implement high bandwidth circuits on an FPGA. The high-speed interface is accomplished by time demultiplexing an incoming signal to create a wide set of signals on the FPGA. In many applications, these signals must be buffered in wide, shallow memories as they are processed throughout the FPGA. These memories may be over 100 bits in width, but only a handful (8–32) words deep. As I/O bandwidth on FPGAs increases, and more circuits begin to take advantage of these high speed I/Os, the need for efficient wide data blocks, including wide, shallow memories, will grow.

Traditionally, vendors support the memory requirements of user circuits by embedding large random access memory (RAM) blocks [1], [3]–[6]. The RAM blocks provide a very dense method of implementing storage, but they are not well suited to wide, shallow memories. The aspect-ratio of most block memories allow a maximum width of 16–32 bits, meaning wide memories must be constructed by cascading several blocks. In addition, the depth of the memories at that ratio are deeper than those needed by simple buffering applications.

Another method for supporting on-chip memory is employed by Xilinx and Lattice Semiconductor (in the ORCA product line, formerly produced by Lucent) [4], [5]. In their devices, the 16-bit lookup-tables within each logic element can be configured as a RAM. Since each logic block can be configured as RAM individually, the memories can be placed close to the attached logic, and several logic blocks, each configured as a RAM, can be combined to implement wider structures. Unfortunately, the area overhead required to glue these smaller memories together into a large memory is significant. As an example, a 32 $\times$ 128 bit memory would require 256 look-up tables (LUTs) to implement the storage, and another 128 LUTs to implement the data output multiplexors.

In this paper, we present an alternate implementation of on-chip memory, designed specifically to support wide buffering applications. In our architecture, we reuse the configuration memory within each switch block, providing the user with access to this memory through additional circuitry. Normally, this memory is used to store the connection patterns required to implement the user circuit. Typically, however, not all the switches in all switch blocks in an FPGA are used to transport signals. By adding only a modest amount of circuitry, the configuration memory in these unused switch blocks can be used to implement wide, shallow buffers and other similar memory structures.

Utilizing unused switch blocks in this way has a number of advantages. The amount of storage within each switch block is significant; an FPGA with 128 tracks/channel contains 1088 configuration bits within each switch block. As we will show, the structure of the switch blocks leads to a natural implementation of wide, shallow memories. In addition, since the switch blocks are evenly distributed across the FPGA, user memories can be instantiated close to the logic that uses them, reducing routing delay. In order to realize these advantages, however, it is important that the additional circuitry does not slow the switch block unacceptably, or result in a significant area overhead.

The use of switch block memory as user storage has been described in [20]. That architecture provided 16 $\times$ 8 bit memories within each switch block. A significant difference between this and our architecture is that we provide 128-bit wide memories rather than 8-bit wide memories. This very wide data width is one of the key features of our architecture, and is important when implementing the wide shallow memories considered in this paper.

II. Baseline Programmable Connection

In this paper, we focus on island-style FPGAs [11]. Each horizontal and vertical channel consists of 128 parallel tracks, each track spanning four clusters (although the concept can be applied to FPGAs of any channel width and segment length, including FPGAs containing more than one segment length). We have concentrated on FPGAs with length-four segments, to be consistent with the architectures in [11]. At the intersection of each horizontal and vertical channel is a Wilton

Manuscript received May 22, 2002; revised December 30, 2004.
The authors are with the University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: steveo@ece.ubc.ca; stevew@ece.ubc.ca).
Digital Object Identifier 10.1109/TVLSI.2005.848817