Design of High-Speed Low-Power Parallel-Prefix VLSI Adders

G. Dimitrakopoulos[1⋆], P. Kolovos[1], P. Kalogerakis[1], and D. Nikolos[2]

[1] Technology and Computer Architecture Laboratory
Computer Engineering and Informatics Dept., University of Patras, Greece
{dimitrak,kolobos,kalogera}@ceid.upatras.gr
[2] Computer Technology Institute, 61 Riga Feraiou Str., 26221 Patras, Greece
nikolosd@cti.gr

Abstract. Parallel-prefix adders offer a highly-efficient solution to the binary addition problem. Several parallel-prefix adder topologies have been presented that exhibit various area and delay characteristics. However, no methodology has been reported so far that directly aims to the reduction of switching activity of the carry-computation unit. In this paper by reformulating the carry equations, we introduce a novel bit-level algorithm that allows the design of power-efficient parallel-prefix adders. Experimental results, based on static-CMOS implementations, reveal that the proposed adders achieve significant power reductions when compared to traditional parallel-prefix adders, while maintaining equal operation speed.

1 Introduction

Binary addition is one of the primitive operations in computer arithmetic. VLSI integer adders are critical elements in general-purpose and DSP processors since they are employed in the design of arithmetic-logic units, in floating-point arithmetic datapaths, and in address generation units. When high operation speed is required, tree-like structures such as parallel-prefix adders are used. Parallel-prefix adders are suitable for VLSI implementation since they rely on the use of simple cells and maintain regular connections between them. The prefix structures allow several tradeoffs among the number of cells used, the number of required logic levels and the cells' fanout [1].

The high-operation speed and the excessive activity of the adder circuits in modern microprocessors, not only lead to high power consumption, but also create thermal hotspots that can severely affect circuit reliability and increase cooling costs. The presence of multiple execution engines in current processors further aggravates the problem [2]. Therefore, there is a strong need for designing power-efficient adders that would also satisfy the tight constraints of high-speed and single-cycle operation.

Although adder design is a well-researched area, only a few research efforts have been presented concerning adders' performance in the energy-delay

⋆ This work has been supported by D. Maritsas Graduate Scholarship.

E. Macii et al. (Eds.): PATMOS 2004, LNCS 3254, pp. 248–257, 2004.

space [3]–[5]. On the other hand, several optimization approaches have been proposed that try to reduce the power consumption of the circuit, either by trading gate sizing with an increase in the maximum delay of the circuit [6], or by using lower supply voltages in non-critical paths [7]. The novelty of the proposed approach is that it directly reduces the switching activity of the carry-computation unit via a mathematical reformulation of the problem. Therefore its application is orthogonal to all other power-optimization methodologies. The proposed parallel-prefix adders are compared to the fastest prefix structures proposed for the traditional definition of carry equations using static CMOS implementations. In all cases the proposed adders are equally fast and can achieve power reductions of up to 11%, although they require around 2.5% larger implementation area. Additionally, for an 1.5% increase in the maximum delay of the circuit, power reductions of up to 17% are reported.

The remainder of the paper is organized as follows. Section 2 describes the intuition behind the proposed approach. In Section 3 some background information on parallel-prefix addition are given, while Section 4 introduces the low-power adder architecture. Experimental results are presented in Section 5 and finally conclusions are drawn in Section 6.

2 Main Idea

Assume that $A = a_{n-1}a_{n-2}\dots a_0$ and $B = b_{n-1}b_{n-2}\dots b_0$ represent the two numbers to be added, and $S = s_{n-1}s_{n-2}\dots s_0$ denotes their sum. An adder can be considered as a three-stage circuit. The preprocessing stage computes the carry-generate bits g_i, the carry-propagate bits p_i, and the half-sum bits h_i, for every i, $0 \le i \le n-1$, according to:

$$g_i = a_i \cdot b_i \quad p_i = a_i + b_i \quad h_i = a_i \oplus b_i, \tag{1}$$

where $\cdot$, $+$, and $\oplus$ denote the logical AND, OR and exclusive-OR operations respectively. The second stage of the adder, hereafter called the carry-computation unit, computes the carry signals c_i, using the carry generate and propagate bits g_i and p_i, whereas the last stage computes the sum bits according to:

$$s_i = h_i \oplus c_{i-1}. \tag{2}$$

The computation of the carries c_i can be performed in parallel for each bit position $0 \le i \le n-1$ using the formula

$$c_i = g_i + \sum_{j=-1}^{i-1} \left(\prod_{k=j+1}^{i} p_k \right) g_j \,, \tag{3}$$

where the bit g_{-1} represents the carry-in signal c_{in}. Based on (3), each carry c_i can be written as

$$c_i = g_i + \mathcal{K}_i, \tag{4}$$

where $\mathcal{K}_i = \sum_{j=-1}^{i-1} \left(\prod_{k=j+1}^{i} p_k \right) g_j$. It can be easily observed from (4) that the switching activity of the bits c_i equally depends on the values assumed by the carry-generate bits g_i and the term $\mathcal{K}_i$.

In order to quantify the difference between the switching activity of the bits $\mathcal{K}_i$ and the carries c_i, an experiment was performed. In particular, a set of 5000 random vectors were generated using Matlab and the switching activity of the bits c_i and $\mathcal{K}_i$ was measured for the case of an 8-bit adder. Figure 1 shows the measured switching activity for each bit position. It is evident that the bits $\mathcal{K}_i$ have reduced switching activity that range from 5% to 7.5%.

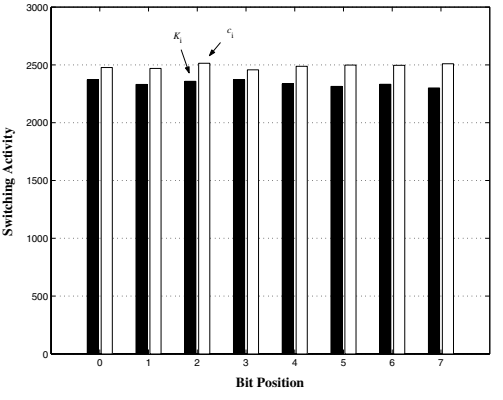

Fig. 1. Switching activity of the bits $\mathcal{K}_i$ and the traditional carries c_i.

The direct computation of the bits $\mathcal{K}_i$ using existing parallel-prefix methods is possible. However, it imposes the insertion of at least one extra logic level that would increase the delay of the circuit. Therefore our objective is to design a carry-computation unit that will compute the bits $\mathcal{K}_i$ instead of the bits c_i, without any delay penalty, and will benefit from the inherent reduced switching activity of the bits $\mathcal{K}_i$.

3 Background on Parallel-Prefix Addition

Several solutions have been presented for the carry-computation problem. Carry computation is transformed to a prefix problem by using the associative operator $\circ$, which associates pairs of generate and propagate bits and is defined, according to [8], as follows,

$$(g, p) \circ (g', p') = (g + p \cdot g', p \cdot p'). \tag{5}$$

Using consecutive associations of the generate and propagate pairs (g, p), each carry is computed according to

$$c_i = (g_i, p_i) \circ (g_{i-1}, p_{i-1}) \circ \ldots \circ (g_1, p_1) \circ (g_0, p_0). \tag{6}$$

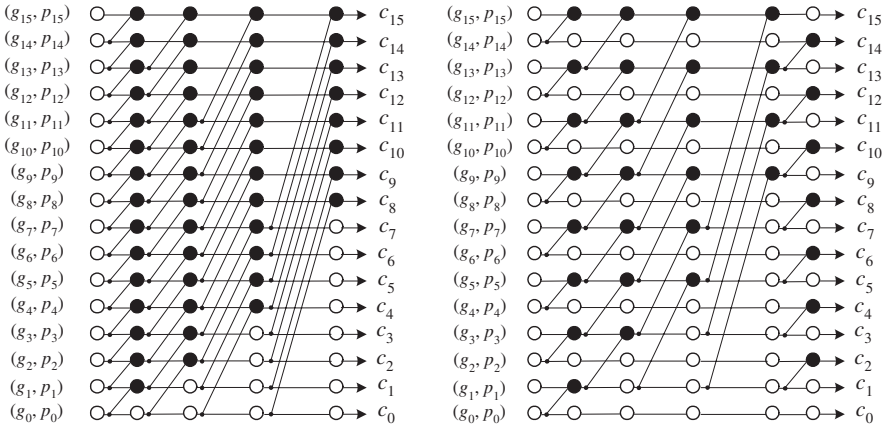

Fig. 2. The Kogge-Stone and Han-Carlson parallel-prefix structures.

Representing the operator $\circ$ as a node $\bullet$ and the signal pairs (g, p) as the edges of a graph, parallel-prefix carry-computation units can be represented as directed acyclic graphs. Figure 2 presents the 16-bit parallel-prefix adders, proposed by Kogge-Stone [9] and Han-Carlson [10], where white nodes $\circ$ are buffering nodes. A complete discussion on parallel-prefix adders can be found in [1] and [11].

4 The Proposed Design Methodology

In the following we will present via an example the parallel-prefix formulation of the computation of the bits $\mathcal{K}_i$. Assume for example the case of $\mathcal{K}_5$. Then according to (4) it holds that

$$\begin{aligned}\mathcal{K}_5 = \; & p_5 \cdot g_4 + p_5 \cdot p_4 \cdot g_3 + p_5 \cdot p_4 \cdot p_3 \cdot g_2 + p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot g_1 \\ & + p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot g_0 \\ & + p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_{\text{in}}. \end{aligned} \tag{7}$$

Based on the definition (1) it holds that $p_i \cdot g_i = g_i$. Then equation (7) can be written as

$$\begin{aligned}\mathcal{K}_5 = \; & p_5 \cdot p_4 \cdot (g_4 + g_3) + p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot (g_2 + g_1) \\ & + p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot (g_0 + c_{\text{in}}). \end{aligned} \tag{8}$$

Assuming that

$$G_i = g_i + g_{i-1} \tag{9}$$

and

$$P_i = p_i \cdot p_{i-1}, \tag{10}$$

with $G_0 = g_0 + c_{\text{in}}$ and $P_0 = p_0$, then equation (8) is equivalent to

$$\mathcal{K}_5 = P_5 \cdot G_4 + P_5 \cdot P_3 \cdot G_2 + P_5 \cdot P_3 \cdot P_1 \cdot G_0. \tag{11}$$

The computation of $\mathcal{K}_5$ can be transformed to a parallel-prefix problem by introducing the variable G_i^*, which is defined as follows:

$$G_i^* = P_i \cdot G_{i-1} \tag{12}$$

and $G_0^* = p_0 \cdot c_{\text{in}}$. Substituting (12) to (11) we get

$$\mathcal{K}_5 = G_5^* + P_5 \cdot G_3^* + P_5 \cdot P_3 \cdot G_1^*, \tag{13}$$

which can be equivalently expressed, using the $\circ$ operator as

$$\mathcal{K}_5 = (G_5^*, P_5) \circ (G_3^*, P_3) \circ (G_1^*, P_1). \tag{14}$$

In case of an 8-bit adder the bits $\mathcal{K}_i$ can be computed by means of the prefix operator $\circ$ according to the following equations:

$$\begin{aligned}
\mathcal{K}_7 &= (G_7^*, P_7) \circ (G_5^*, P_5) \circ (G_3^*, P_3) \circ (G_1^*, P_1) \\
\mathcal{K}_6 &= (G_6^*, P_6) \circ (G_4^*, P_4) \circ (G_2^*, P_2) \circ (G_0^*, P_0) \\
\mathcal{K}_5 &= (G_5^*, P_5) \circ (G_3^*, P_3) \circ (G_1^*, P_1) \\
\mathcal{K}_4 &= (G_4^*, P_4) \circ (G_2^*, P_2) \circ (G_0^*, P_0) \\
\mathcal{K}_3 &= (G_3^*, P_3) \circ (G_1^*, P_1) \\
\mathcal{K}_2 &= (G_2^*, P_2) \circ (G_0^*, P_0) \\
\mathcal{K}_1 &= (G_1^*, P_1) \\
\mathcal{K}_0 &= (G_0^*, P_0)
\end{aligned}$$

It can be easily derived by induction that the bits $\mathcal{K}_i$ of the odd and the even-indexed bit positions can be expressed as

$$\mathcal{K}_{2k} = (G_{2k}^*, P_{2k}) \circ (G_{2k-2}^*, P_{2k-2}) \circ \ldots \circ (G_0^*, P_0) \tag{15}$$

$$\mathcal{K}_{2k+1} = (G_{2k+1}^*, P_{2k+1}) \circ (G_{2k-1}^*, P_{2k-1}) \circ \ldots \circ (G_1^*, P_1) \tag{16}$$

Based on the form of the equations that compute the terms $\mathcal{K}_i$, in case of the 8-bit adder, the following observations can be made. At first the number of terms (G_i^*, P_i) that need to be associated is reduced to half compared to the traditional approach, where the pairs (g, p) are used (Eq. (6)). In this way one less prefix level is required for the computation of the bits $\mathcal{K}_i$, which directly leads to a reduction of 2 logic levels in the final implementation. Taking into account that the new preprocessing stage that computes the pairs (G_i^*, P_i) requires two additional logic levels compared to the logic-levels needed to derive the bits (g, p) in the traditional case, we conclude that no extra delay is imposed by the proposed adders. Additionally, the terms $\mathcal{K}_i$ of the odd and the even-indexed bit positions can be computed independently, thus directly reducing the fanout requirements of the parallel-prefix structures.

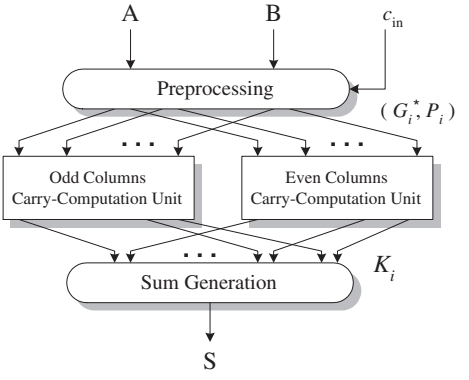

Fig. 3. The architecture of the proposed parallel-prefix adders.

The computation of the bits $\mathcal{K}_i$, instead of the normal carries c_i, complicates the derivation of the final sum bits s_i since in this case

$$s_i = h_i \oplus c_{i-1} = h_i \oplus (g_{i-1} + \mathcal{K}_{i-1})\,. \tag{17}$$

However, using the Shannon expansion theorem on $\mathcal{K}_{i-1}$, the computation of the bits s_i can be transformed as follows

$$s_i = \overline{\mathcal{K}}_{i-1} \cdot (h_i \oplus g_{i-1}) + \mathcal{K}_{i-1} \cdot h_i\,. \tag{18}$$

Equation (18) can be implemented using a multiplexer that selects either h_i or $(g_{i-1} \oplus h_i)$ according to the value of $\mathcal{K}_{i-1}$. The notation $\overline{x}$ denotes the complement of bit x. Taking into account that in general a XOR gate is of almost equal delay to a multiplexer, and that both h_i and $(g_{i-1} \oplus h_i)$ are computed in less logic levels than $\mathcal{K}_{i-1}$, then no extra delay is imposed by the use of the bits $\mathcal{K}_i$ for the computation of the sum bits s_i. The carry-out bit c_{n-1} is produced almost simultaneously with the sum bits using the relation $c_{n-1} = g_{n-1} + K_{n-1}$.

The architecture of the proposed adders is shown in Figure 3, and their design is summarized in the following steps.

- Calculate the carry generate/propagate pairs (g_i, p_i) according to (1).
- Combine the bits g_i, p_i, g_{i-1}, and p_{i-1} in order to produce the intermediate pairs (G_i^*, P_i), based on the definitions (9), (10), and (12).
- Produce two separate prefix-trees, one for the even and one for the odd-indexed bit positions that compute the terms $\mathcal{K}_{2k}$ and $\mathcal{K}_{2k+1}$ of equations (15) and (16), using the pairs (G_i^*, P_i). Any of the already known parallel-prefix structures can be employed for the generation of the bits $\mathcal{K}_i$, in $\log_2 n - 1$ prefix levels.

In Figure 4 several architectures for the case of a 16-bit adder are presented, each one having different area, delay and fanout requirements. It can be easily

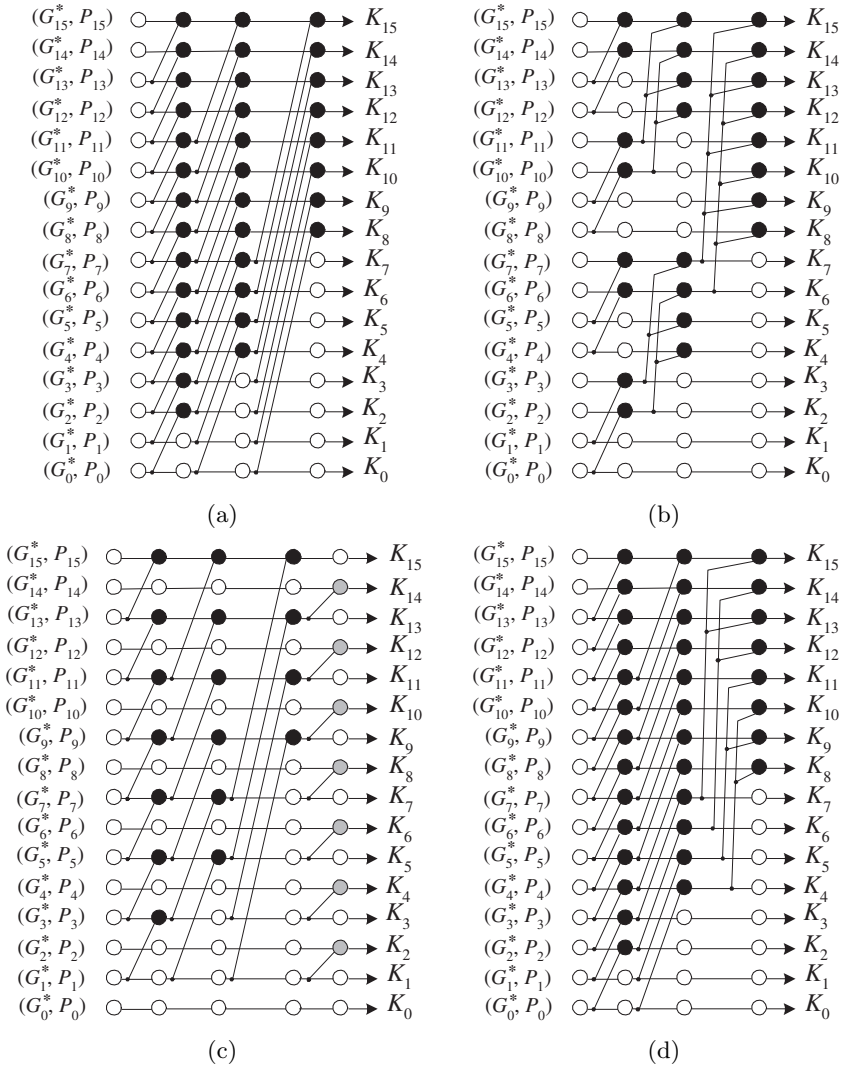

Fig. 4. Novel 16-bit parallel-prefix carry computation units.

verified that the proposed adders maintain all the benefits of the parallel-prefix structures, while at the same time offer reduced fanout requirements. Figure 4(a) presents a Kogge-Stone-like parallel-prefix structure, while in Figure 4(c) a Han-Carlson-like scheme is shown. It should be noted that in case of the proposed Han-Carlson-like prefix tree only the terms $\mathcal{K}_i$ of the odd-indexed bit positions are computed. The remaining terms of the even-indexed bit positions are computed using the bits $\mathcal{K}_i$ according to

$$\mathcal{K}_{i+1} = p_{i+1} \cdot (g_i + \mathcal{K}_i), \tag{19}$$

which are implemented by the grey cells of the last prefix level.

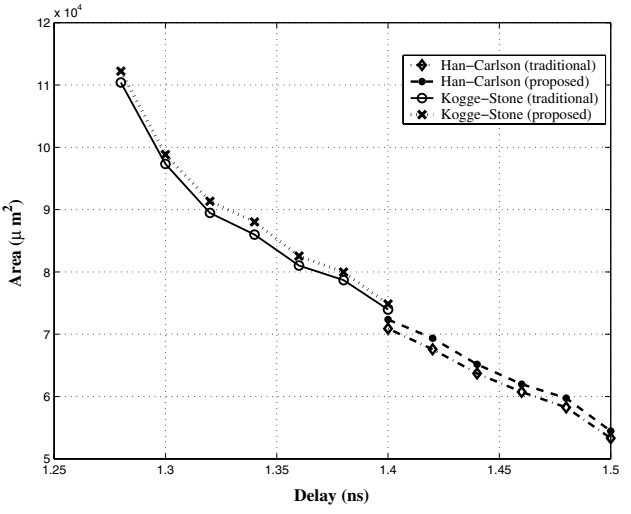

Fig. 5. The area and delay estimates for the traditional and the proposed 64-bit adders using a Kogge-Stone [9] and Han-Carlson [10] parallel-prefix structures.

5 Experimental Results

The proposed adders are compared against the parallel-prefix structures proposed by Kogge-Stone [9] and Han-Carlson [10] for the traditional definition of carry equations (Eq. (6)). Each 64-bit adder was at first described and simulated in Verilog HDL. In the following, all adders were mapped on the $0.25\mu m$ VST-25 technology library under typical conditions (2.5V, 25°C), using the Synopsys® Design Compiler. Each design was recursively optimized for speed targeting the minimum possible delay. Also, several other designs were obtained targeting less strict delay constraints.

Figure 5 presents the area and delay estimates of the proposed and the traditional parallel-prefix adder architectures. It is noted that the proposed adders in all cases achieve equal delay compared to the already known structures with only a 2.5% in average increase in the implementation area. Based on the data provided by our technology library, it is derived that the delay of 1 fanout-4 (FO4) inverter equals to $120 - 130$ps, under typical conditions. Thus the obtained results fully agree with the results given in [12] and [5], where the delay of different adder topologies is calculated using the method of Logical Effort.

In Figure 6 the power consumption of the proposed and the traditional Kogge-Stone 64-bit adders are presented versus the different delays of the circuit. It can be easily observed that the proposed architecture achieves power reductions that range between 9% and 11%. All measurements were taken using PrimePower of Synopsys toolset after the application of 5000 random vectors. It should be noted that although the proposed adders require slightly larger implementation area, due to the addition of the extra multiplexers in the sum

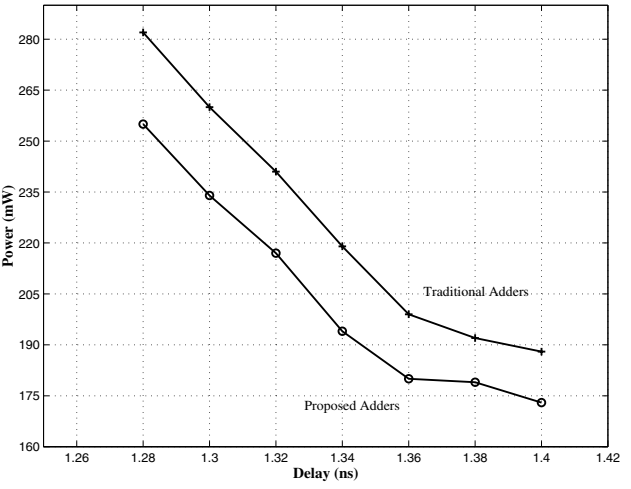

Fig. 6. Power and delay estimates for the traditional and the proposed 64-bit adders using a Kogge-Stone [9] parallel-prefix structure.

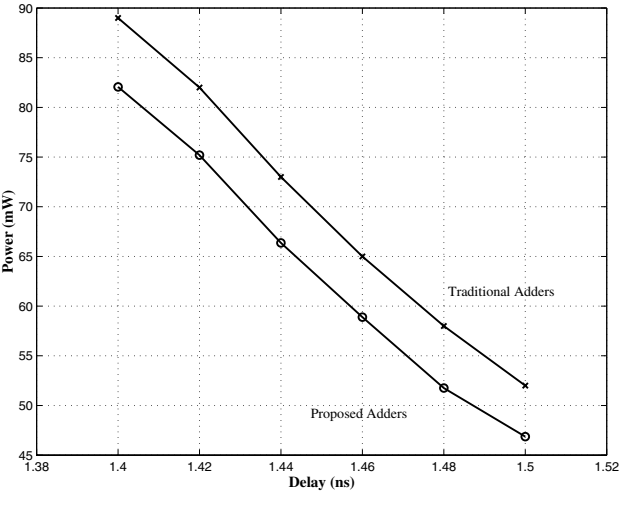

Fig. 7. Power and delay estimates for the traditional and the proposed 64-bit adders using a Han-Carlson [10] parallel-prefix structure.

generation unit, they still offer power efficient designs as a result of the reduced switching activity of the the novel carry-computation units. Similar results are obtained for the case of the 64-bit Han-Carlson adders, as shown in Figure 7.

Finally, since the application of the proposed technique is orthogonal to all other power optimization methods, such as gate sizing and the use of multiple supply voltages, their combined use can lead to further reduction in power dissi-

pation. Specifically, the adoption of the proposed technique offers on average an extra 10% gain over the reductions obtained by the use of any other circuit-level optimization.

6 Conclusions

A systematic methodology for the design of power-efficient parallel-prefix adders has been introduced in this paper. The proposed adders preserve all the benefits of the traditional parallel-prefix carry-computation units, while at the same time offer reduced switching activity and fanout requirements. Hence, high-speed datapaths of modern microprocessors can truly benefit from the adoption of the proposed adder architecture.

References

1. Simon Knowles, "A Family of Adders", in Proceedings of the 14th IEEE Symposium on Computer Arithmetic, April 1999, pp. 30–34.
2. S. Mathew, R. Krishnamurthy, M. Anders, R. Rios, K. Mistry and K. Soumyanath, "Sub-500-ps 64-b ALUs in 0.18-μm SOI/Bulk CMOS: Design and Scaling Trends" IEEE Journal of Solid-State Circuits, vol. 36, no. 11, Nov. 2001.
3. T. K. Callaway and E. E. Swartzlander, "Low-Power arithmetic components in Low-Power Design Methodologies", J. M. Rabaey and M. Pedram, Eds. Norwell, MA: Kluwer, 1996, pp. 161–200.
4. C.Nagendra, M. J. Irwin, and R. M. Owens, "Area-Time-Power tradeoffs in parallel adders", IEEE Trans. Circuits Syst. II, vol.43, pp. 689–702, Oct.1996.
5. V. G. Oklobdzija, B. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, "Energy-Delay Estimation Technique for High-Performance Microprocessor VLSI Adders", 16th IEEE Symposium on Computer Arithmetic, June 2003, pp. 15–22.
6. R. Brodersen, M. Horowitz, D. Markovic, B. Nikolic and V. Stojanovic, "Methods for True Power Minimization, International Conference on Computer-Aided Design, Nov. 2002, pp. 35–42.
7. Y. Shimazaki, R. Zlatanovici, and B. Nikolic, "A Shared-Well Dual-Supply-Voltage 64-bit ALU", IEEE Journal of Solid-State Circuits, vol. 39, no. 3, March 2004.
8. R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders", IEEE Trans. on Computers, vol. 31, no. 3, pp. 260–264, Mar. 1982.
9. P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations", IEEE Trans. on Computers, vol. C-22, pp. 786–792, Aug. 1973.
10. T. Han and D. Carlson, "Fast Area-Efficient VLSI Adders", in Proc. 8th IEEE Symposium on Computer Arithmetic, May 1987, pp. 49–56.
11. A. Beaumont-Smith and C. C. Lim, "Parallel-Prefix Adder Design", 14th IEEE Symposium on Computer Arithmetic, Apr. 2001, pp. 218–225.
12. D. Harris and I. Sutherland, "Logical Effort of Carry Propagate Adders", 37th Asilomar Conference, Nov. 2003, pp. 873–878.