ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland

Higher Radix Kogge-Stone Parallel Prefix Adder Architectures

Frank K. Gurkaynak‡, Yusuf Leblebici†, Laurent Chaouat‡ and Patrick J. McGuinness‡
† *Electrical and Computer Engineering Department Worcester Polytechnic Institute Worcester, MA 01609*
‡ *SoCDT Advanced Tools, Motorola, Inc. Austin, TX 78730*

Abstract

In this paper, we describe the design of radix-3 and radix-4 parallel prefix adders, that theoretically have logical depths of $log_3 n$ and $log_4 n$ respectively, where n is the bit-width of the input signals. The main building blocks of the higher radix parallel prefix adders are identified and higher radix structures of Kogge-Stone Adders are presented. We show that with the higher radix architectures the logic depth can be reduced by 50% and the cell count can be reduced as much as 47% for 64-bit adders. Simulation results indicate that radix-4 adders can be more than 30% faster than radix-2 realizations.

1 Introduction

The addition of two binary numbers is one of the most important arithmetic function in modern digital VLSI systems, taking a major part of the design effort of modern digital signal processors and general purpose microprocessors. The maximum operating speed of these processors depend largely on how fast the main computation block can process data. For a large number of applications, the speed critical computation block includes adders: either as stand-alone blocks or integrated into multiplier architectures. As a result, specialized speed optimized adder architectures are required for high performance systems.

The design of faster, smaller and more efficient adder architectures has been the focus of many research efforts and has resulted in a large number of adder architectures. Some architectures like Carry-Skip Adder, Conditional Sum Adder and Carry Select Adder[1] rely on a basic ripple carry adder structure that has been modified to shorten carry propagation path. The parallel prefix adders[2] are a more general form where a network is used to pre-calculate the carry signals. Some well known parallel prefix adder architectures using different carry-lookahead networks are: The Sklansky Binary Tree Adder [3], the Brent Kung Adder [4], and the Kogge-Stone Adder [5]. Although most of the above mentioned algorithms are formulated for any radii, practical implementations have generally been limited to radix-2 implementations.

In this paper, we discuss the theory, and feasibility of implementation of the radix-4 and radix-3 implementations of the Kogge-Stone Adder. An introduction to the parallel prefix problem is given in Section-2 of this paper. Section-3 defines standard building blocks, and introduces two new blocks, for building higher radix parallel prefix adders. The higher radix realizations of Kogge-Stone parallel prefix architecture is examined in detail in Section-4. Finally, Section-5 includes a summary of our results.

2 Parallel Prefix Problem

Most of the known adder architectures can be represented as a parallel prefix adder structure consisting of three main parts : Pre-processing, carry lookahead network and post-processing.

Assuming two binary input vectors A and B, the pre-processing part extracts two special signals propagate (p) and generate (g) using simple logic circuits. The calculation of the Sum is assigned to the post-processing step, which is like the pre-processing step a constant time operation. This leaves only the carry propagation (carry lookahead) problem, which is a recursive function, to be addressed. The carry propagation problem can be expressed in terms of a prefix problem where for a set of binary inputs $(x_i : i = 0, 1, 2, ..., n)$ the outputs $(y_i : i = 0, 1, 2, ..., n)$ are defined by the help of an associative binary operator $\bullet$ as:

$$\begin{aligned} y_0 &= x_0 \\ y_i &= x_i \bullet y_{i-1} \\ y_i &= x_i \bullet x_{i-1} \bullet \cdots \bullet x_1 \bullet x_0 \end{aligned} \quad (1)$$

Since the $\bullet$ operator is associative, it can be grouped in any order and computed in a number of levels. To express

0-7803-5482-6/99/$10.00 ©2000 IEEE

the sub-products let us introduce the notation $Y_{i:j}^{k}$, where k is the level of the sub-product and $i : j$ represent a continuous range that this sub-product covers. For the carry propagation problem let us define the sub-product couple (G, P) such that:

$$\begin{aligned}(G,P)_{i:i}^{0} &= (g_i, p_i) \\ (G,P)_{i:j}^{k} &= (G,P)_{i:q+1}^{k-1} \bullet (G,P)_{q:j}^{k-1}\end{aligned} \tag{2}$$

Where the desired

$$Carry_i = G_{i:0} \tag{3}$$

regardless of the number of levels necessary to cover the range $i : 0$. Depending on the algorithm the carry propagation network will have a different structure and shape. In general the following observations can be made:

- The maximum levels required to calculate the final *Carry* signal is referred as the depth of the prefix graph, and equals to the number of logic levels in the network. The depth of the carry propagate network is a function of the bit-width of the input. This number relates roughly to the delay of the network.
- The total number of binary associative operations within the network determine the active area required to compute the result.
- Secondary effects like the number of times a sub-range is used in subsequent operations (fan-out) and the distance between operators of an operation (connection length) also contribute to the overall performance of the system.

3 Higher Radix Parallel Prefix Adder Architectures

The delay of a parallel prefix adder is directly proportional to the number of levels in the carry propagation network stage. The majority of contemporary adder architectures rely on carry propagation networkscomposed of blocks that have only two inputs. The lower bound of the number of stages required for such networks lie at $log_2 n$ where n is the bit-width of the input vectors. This lower bound can be lowered by using more complex blocks that process 3 or even 4 inputs to obtain lower bounds of $log_3 n$ and $log_4 n$ respectively. Adders designed using these complex blocks can theoretically achieve higherprocessing speeds at the cost of additional area.

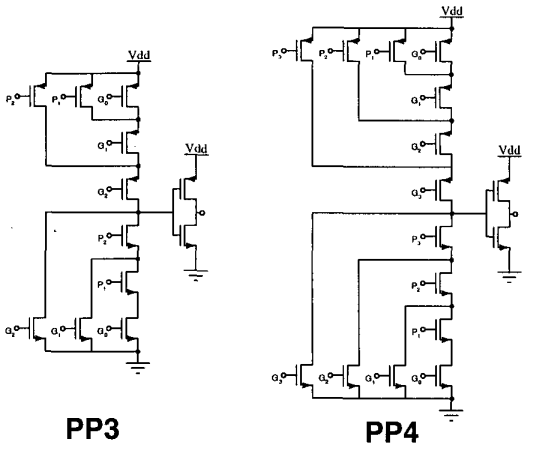


Figure 1: Transistor level schematics for the generate block of the parallel prefix cells with 3 and 4 inputs.

3.1 Building Blocks for Radix-4 Parallel Prefix Adders

The pre-processing and post-processing stages of a typical parallel prefix adders consist of .simple logic gates. The pre-processing stage can be realized by a simple half adder, or an AND gate and an OR gate. The post-processing stage is merely an XOR gate. The simple, 2-input, prefix function can be mapped to standard logic operations as follows:

$$\begin{aligned}(G,P)_{a\cup b}^{k} &= (G,P)_{a}^{k-1} \bullet (G,P)_{b}^{k-1} \\ G_{a\cup b}^{k} &= G_{a}^{k-1} + P_{a}^{k-1} \cdot G_{b}^{k-1} \\ P_{a\cup b}^{k} &= P_{a}^{k-1} \cdot P_{b}^{k-1}\end{aligned} \tag{4}$$

This function pair can be realized using an AND-OR gate and a separate AND gate sharing common inputs. We call this basic cell PP2 (parallel prefix-2) and define two additional cells PP3 and PP4 which realize the parallel prefix function for three and four inputs respectively. The PP3 and PP4 cells realize the following logic functions for the generate output. (The propagate structure is a 3 or 4 input AND gate).

$$G_{PP3} = G_2 + (P_2 \cdot (G_1 + P_1 \cdot G_0)) \tag{5}$$

$$G_{PP4} = G_3 + (P_3 \cdot (G_2 + M\,(P_2 \cdot (G_1 + P_1 \cdot G_0)))) \tag{6}$$

The transistor level schematics for these functions are shown in Figure-1. Simulation results for these blocks (Figure-2) indicate that the delay of a PP4 block is only

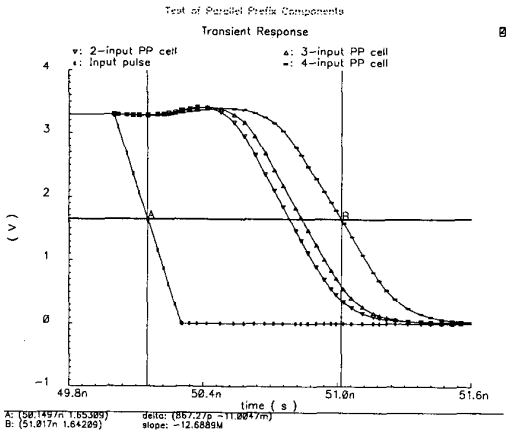


Figure 2: Simulation result of PP2, PP3 and PP4 cells using a standard 0.8 μm CMOS process at 3.3 Volts.

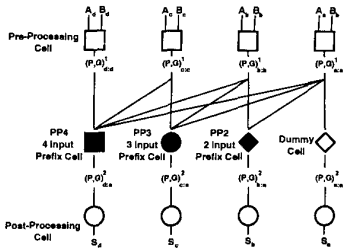


Figure 3: Symbols used in the graph representation of parallel prefix adders.

Table 1: Comparison of 64-bit Adder Architectures.

Name	Depth	Cells	PP2	PP3	PP4	Fan-out
RCA-64	64	64	64	-	-	1
SK2-64	6	192	192	-	-	32
KS2-64	6	321	321	-	-	2
KS3-64	4	216	40	176	-	3
KS4-64	3	171	21	21	129	4

1.2 times larger than that of a PP2 block under similar load conditions. Since a radix-2 design requires two PP2 blocks to compute the result of a single PP4 block, designs using radix-4 blocks can work faster, despite the fact that more complex basic cells are used.

4 Higher Radix Kogge-Stone Parallel Prefix Architecture

Using the newly defined prefix cells, higher radix adder structures can easily be designed. We will use a graph representation to provide a clearer view of the architecture. Figure-3 shows the main symbols used in the graphs. PP4, PP3 and PP2 cells are represented using filled symbols. The dummy cell shown as a blank diamond, does not contain any logic (or only a buffer) and can be considered a vacant position.

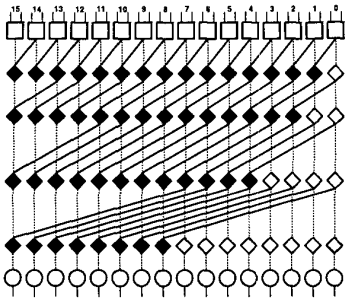
Among different parallel prefix adder realizations the Sklansky Binary Tree and the Kogge-Stone architectures have the least possible network depth of $log_r n$ where r is the radix and n is the bit-width of the inputs. The main advantage of the Kogge-Stone architecture is the maximum fan-out, which equals to the radix r, whereas the Sklansky Binary Tree adder has a maximum fan-out of $r^n - r^{n-1}$. Although the Kogge-Stone adder has a much lower fan-out, this reduction in fan-out comes at a cost of increased cell count. The Radix-2 Kogge-Stone Adder can be seen in Figure-4(a). The Radix-3 and Radix-4 implementations of the adder can be seen in Figure-4(b) and (c) respectively.

We have run transistor level simulations to compare the relative performances of radix-4 and radix-2 realizations of 64-bit Kogge-Stone Parallel Prefix Adders. Figure-5 shows a typical simulation result for both adder architectures. It can be seen that the Radix-4 architecture is 32.5% faster (3.98 ns for Radix-4, 5.89ns for Radix-2 in a 0.8 μm CMOS design using 3.3V supply voltage.)

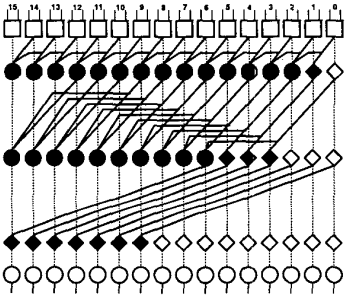
5 Summary and Conclusions

In this work we have presented the topologies for Radix-4 and Radix-3 parallel prefix adders, that have a theoretical carry propagation network depth of $log_4 n$ which essentially doubles the speed of these adder structures. Although more complex base cells result in slightly larger delays, we have found that on the average the delays can be reduced as much as 32.5%.

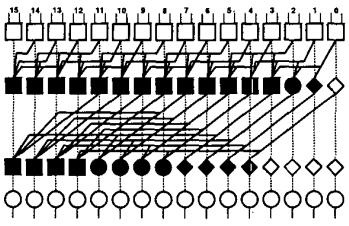
Table-1 compares five realizations of 64-bit adders. A standard Ripple Carry adder (RCA-64), a Sklansky Binary Tree adder (SK2-64) and three realizations of Kogge-Stone adders with different radii. The table lists both the total number of cells and the brake-down into individual cell categories. It is important to note that the proposed architectures not only reduce the depth of the carry prop-



(a)



(b)



(c)

Figure 4: K ogge-StoneA dderstructure for (a) Radix-2, (b) Radix-3 and (c) Radix-4.

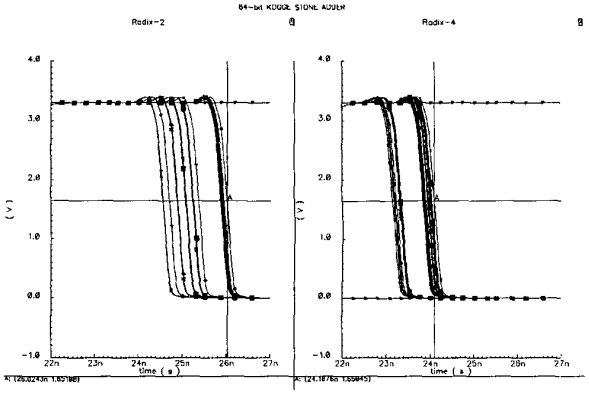


Figure 5: Simulation result comparing 64-bit radix-2 (5.89ns) and radix-4 (3.98ns) Kogge-Stone realizations.

agation netw ork as much as 50%, but the number of cells required also decrease by as muc h as 47% (for the Radix-4 K ogge Stone Alder) when compared to the Radix-2 realizations.

References

[1] J. J. F. Cavannagh, "Digital Computer Arithmetic: Design and Implementation", McGraw-Hill, 1984

[2] R. Zimmermann, "Non-heuristic optimization and synthesis of parallel-prefix adders", in Proc. Int. Workshop on Logic and Architecture Synthesis, Grenoble, France, Dec. 1996, pp 123-132.

[3] J. Sklansky, "Conditional Sum A ddition Logic", IRE T rans. Electron. Comput, EC-9(6), pp 226-231, June 1960.

[4] R.P. Brent and H.T. Kung, "A Regular Layout for Parallel Adders", in IEEE Trans. on Computers, Vol C-31, No 3, March 1982.

[5] 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, No 8, August 1973.