Design and Implementation of Carry Tree Adders using Low Power FPGAs

Sivannarayana G1, Raveendra babu Maddasani2 and Padmasri Ch3.
Department of Electronics & Communication Engineering1,2,3, Al-Ameer college of engineering1,3, JNT University college of Engineering3, Jawaharlal Nehru Technological University, Kakinada, India1,2,3.

Abstract— The binary adder is the critical element in most digital circuit designs including digital signal processors (DSP) and microprocessor data path units. As such, extensive research continues to be focused on improving the power delay performance of the adder. To resolve the delay of carry-look ahead adders, the scheme of multilevel-look ahead adders. These adders have tree structures within a carry-computing stage similar to the carry propagate adder. However, the other two stages for these adders are called pre-computation and post-computation stages. In pre-computation stage, each bit computes its carry generate/propagate and a temporary sum. In the post-computation stage, the sum and carry-out are finally produced. The carry-out can be omitted if only a sum needs to be produced.

Index Terms—Adders, carry-propagate, carry-generate
Power delay.

I. INTRODUCTION

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. Therefore, there is a strong need for designing power-efficient adders that would also satisfy the tight constraints of high-speed. The design of high-speed, low-power and area efficient binary adders always receives a great deal of attention. Among the hundreds adder architectures known in the literature, when high performances are mandatory, parallel-prefix trees are generally preferable.

Optimizing a parallel-prefix tree architecture and its transistor-level implementation for a specific design is not trivial since the designer has to choose: i) the radix of the carry tree (i.e. the number of carriers grouped in each step of the computation); ii) the tree architecture; iii) the logic style. As is well known, all these choices are crucial for both speed and power. parallel-prefix trees realized using dynamic domino logic achieve higher speed performances at the expense of consumed energy; where as, using static logics lowers power consumption, but sacrifices computational speed. This paper proposes a novel approach to optimize the implementation of the basic logic modules, namely the preprocessing stage and the associative dot operator, typically used within parallel-prefix adders.

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, or by using lower supply voltages in non-critical paths. 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 basic idea exploited in the proposed designs consists of: 1) increasing speed by reducing the complexity of the pull-down networks (PDNs) of each dynamic gate; and 2) saving power by reducing the number of dynamic stages within the overall structure of the generic parallel-prefix tree.

The paper is organized as follows: in Section II, a brief background on the parallel-prefix adder trees is provided, the novel circuits are then described in Section III where comparison results are also presented and discussed; finally, conclusions are drawn.

II. MODEL:

Assume that $A = a_n\ldots a_2 \ldots a_0$ and $B = b_n\ldots b_2 \ldots b_0$ represent the two numbers to be added, and $S = s_n\ldots s_2 \ldots s_0$ 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 $i$, $0 \leq i \leq n - 1$, according to:

$$
g_i = a_i \cdot b_i
$$

$$
p_i = a_i + b_i
$$

$$
h_i = a_i \oplus b_i
$$

(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:

$$
s_i = h_i \oplus c_i - 1.
$$

(2)
The computation of the carries $c_i$ can be performed in parallel for each bit position $0 \leq i \leq n - 1$ using the formula

$$c_i = g_i + K_i$$

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

$$c_i = g_i + K_i$$

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, as follows,

$$(g, p) \circ (g', p') = (g + p \cdot g', p \cdot p').$$

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

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

Fig1: The architecture of the proposed Carry Tree adders

The computation of the bits $K_i$, instead of the normal carries $c_i$, complicates the derivation of the final sum bits $s_i$ since in this case

$$s_i = h_i \oplus c_{i-1} = h_i \oplus (g_{i-1} + K_{i-1})$$

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

$$s_i = K_{i-1} \cdot (h_i \oplus g_{i-1}) + K_{i-1} \cdot h_i$$

Equation (8) can be implemented using a multiplexer that selects either $h_i$ or $(g_{i-1} \oplus h_i)$ according to the value of $K_{i-1}$. 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 $K_{i-1}$, then no extra delay is imposed by the use of the bits $h_i$ 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

$$c_{n-1} = g_{n-1} \lor K_{n-1}.$$

III. DESIGN SUMMARIZATION:

The architecture of the proposed adders is shown in Figure1.

The computation of $K_i$ can be transformed to a parallel-prefix problem by introducing the variable $G^*_i$, which is defined as follows:

Assuming that

$$G_i = g_i + g_{i-1}$$

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

$$G^*_i = P_i \cdot G_{i-1}$$

The bits $K_i$ of the odd and the even-indexed bit positions can be expressed as

$$K_{2k} = (G^*_{2k+1}, P_{2k}) \circ (G^*_{2k}, P_{2k-2}) \circ \ldots \circ (G^*_{2}, P_0)$$

$$K_{2k+1} = (G^*_{2k+1}, P_{2k+1}) \circ (G^*_{2k}, P_{2k-1}) \circ \ldots \circ (G^*_{1}, P_1)$$

At first the number of terms $(G^*_{2k}, P_0)$ 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 $K_i$, which directly leads to a reduction of 2 logic levels in the final implementation. Taking into account that the new pre-processing 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 $K_i$ of the odd and the even-indexed bit positions can be computed independently, thus directly reducing the fan out requirements of the parallel-prefix structures.

Their design is summarized in the following steps.

1. Calculate the carry generate/propagate pairs $(g_i, p_i)$ according to (1).

2. Combine the bits $g_i, p_i, g_{i-1}$, and $p_{i-1}$ in order to produce the intermediate pairs $(G^*_{2k}, P_0)$, based on the definitions (9) & (10).

3. Produce two separate prefix-trees, one for the even and one for the odd indexed bit positions that compute the terms $K_{2k}$ and $K_{2k+1}$ of equations (11) and (12), using the pairs $(G^*_{2k}, P_0)$. Any of the already known parallel-prefix structures can be
employed for the generation of the bits $K_i$, in $\log_2 n - 1$ prefix levels.

In Figure 2 several architectures for the case of a 16-bit adder are presented, each one having different area, delay and fan out requirements. It can be easily verified that the proposed adders maintain all the benefits of the parallel-prefix structures, while at the same time offer reduced fan out requirements. Figure 2(a) presents a Kogge-Stone-like parallel-prefix structure, while in Figure 2(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 $K_i$ of the odd-indexed bit positions are computed. The remaining terms of the even-indexed bit positions are computed using the bits $K_i$ according to

$$K_{i+1} = p_i + 1 \cdot (g_i + K_i)$$

Which are implemented by the grey cells of the last prefix level

Fig.2: Novel 16-bit parallel-prefix Carry computation units

IV.RESULTS:

The proposed adders are compared against the parallel-prefix structures proposed by Kogge-Stone and Han-Carlson 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 0.25μm VST-25 technology library under typical conditions (2.5V, 25oC), 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.

Fig.3: The area and delay estimates for the traditional and the proposed 64-bit adders using a Kogge-Stone and Han-Carlson parallel-prefix structures.
Figure 3 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 – 130ps, under typical conditions. Thus the obtained results fully agree with the results given in (7), where the delay of different adder topologies is calculated using the method of Logical Effort.

In Figure 4 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 Prime Power 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 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 5.

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 dissipation. 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.

V. CONCLUSIONS AND FUTURE WORK:

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 fan out requirements. Hence, high-speed data paths of modern microprocessors can truly benefit from the adoption of the proposed adder architecture. It may desire to extend our fan out to fan out $f$.

VI. REFERENCES:


