INTEGRATION, the VLSI journal 44 (2011) 62–74

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Optimized design of parallel carry-select adders

Massimo Alioto [a,*], Gaetano Palumbo [b], Massimo Poli [a,1]

[a] DII (Dipartimento di Ingegneria dell'Informazione), UNIVERSITÀ DI SIENA, Via Roma 56, I-53100 SIENA, Italy
[b] DIEES (Dipartimento di Ingegneria Elettrica Elettronica e dei Sistemi), UNIVERSITÀ DI CATANIA, Viale Andrea Doria 6, I-95125 CATANIA, Italy

ARTICLE INFO

Article history:
Received 6 February 2009
Received in revised form
7 August 2010
Accepted 23 August 2010

Keywords:
Adders
Carry Select
Design
VLSI
Computer arithmetic

ABSTRACT

In this paper, a novel gate-level strategy for designing Carry-Select adders is proposed. The strategy is more general than the previously proposed techniques, and accounts for the dependence of multiplexer delay on its fan-out. Moreover the strategy is simple and systematic, and is helpful for designing Carry-Select adders with a pencil-and-paper approach. An approximate expression of the minimum delay achievable is derived to estimate performance before carrying out the design.

The proposed strategy is validated in more than 1000 adders. Analysis confirms that the strategy leads to a delay which is minimal in most cases, and always within 5.7%.

1. Introduction

The adder is the fundamental block in datapaths of VLSI digital circuits, such as microprocessors and DSPs, because addition is also used to perform a number of operations, such as subtraction, multiplication, division and address computation [1–4]. Moreover the speed of a digital integrated circuit is mainly limited by that of the adder circuit, which usually competes for silicon area and power consumption [5].

Until now, many adder architectures have been proposed to improve speed performance, while satisfying specific trade-offs with area and power consumption. In particular, Carry-Select adders (hereinafter referred to as CSLs [6]) allow a favorable performance-area-power trade-off [7–14], especially when reduced-area schemes are used [14–17]. Indeed, CSL represents a nice performance-area-power tradeoff that lies somewhat between the low power consumption of Ripple Carry adders ($O(n)$ area, $O(n)$ delay) and the high performance of Carry Look-Ahead ($O(n\log(n))$ area, $O(\log(n))$ delay) [12]. Thanks to their high performance at a reasonably low power consumption, Carry-Select adders are widely employed in mobile applications [11], [18], [19].

The Carry-Select adder is based on the consideration that the carry propagation (and sum evaluation) in a chain of m full adders (FAs) can be significantly sped-up, if the result is computed without waiting for the incoming carry input signal, C_{in}. To be more specific, as depicted in Fig. 1, carry and sum outputs are evaluated in parallel by assuming C_{in} is equal to 0 and 1, respectively, and then selecting the correct result through a multiplexer (MUX) according to the value of C_{in} [8]. It is worth noting that, although Fig. 1 includes two different full adder chains, they can be rearranged to share most of their logic circuitry, thereby greatly simplifying circuit implementation [15–17]. In practical CSLs, full adders are divided into Q groups of M_j bits for $j=1, \ldots, Q$, each having the structure shown in Fig. 1, thus leading to the architecture in Fig. 2.

Here, the MUX in the first group is omitted because its carry input is equal to that of the adder, available along with the two operands X and Y at the beginning of the computation [3], [7]. It is well known that CSL speed performance is greatly affected by the number of full adders chosen for each group, once full adders and MUXes are designed at the transistor level in terms of topology and sizing. Accordingly, the gate-level design aims at optimally grouping full adders into blocks to minimize the overall worst-case delay, which is not an easy task.

Until now, several techniques have been proposed to find the optimum distribution of full adders into blocks, thereby minimizing the delay [20–25]. However, most of these strategies have been developed in the specific case, where a CSL is used as the final Carry Propagate adder in a multiplier circuit, in which different input arrival times are exploited to reduce the adder delay. Regarding the design of parallel CSL adders, where all input signals are generated at the same time, to the best of the authors' knowledge, only [5], [13], [23] and [24] introduce a design strategy which optimally sets group sizes to minimize the overall delay. In this case, however, analysis is oversimplified through approximations which are not satisfied in practical cases (as in the case of [5], [23] and [24]) or is based on an intuition rather than a

* Corresponding author. Tel.: +39 0577 234632; fax: +39 0577 233602.
E-mail addresses: malioto@dii.unisi.it, poli@dii.unisi.it (M. Alioto), gpalumbo@diees.unict.it (M. Poli).
[1] Tel.: +39 095 738231; fax: +39 095 330793.

0167-9260/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2010.08.003

rigorous analysis (as in the case of [13]), hence they can easily lead to CSL configuration that is far from being optimum. For example, delay is assumed to be equal for all multiplexers regardless of their fan-out, which instead varies greatly in actual implementations.

In this paper, a gate-level design strategy to minimize the delay of a parallel Carry-Select adder by optimally sizing full adder blocks is proposed. The strategy starts from an exact timing analysis of the CSL adder, while accounting for the dependence of the MUX delay on its fan-out. The timing analysis is inspired by the intuitive approach described in [13] of the same authors, which is herein generalized and rigorously justified and supported by theorems. From the exact timing analysis, the criteria to minimize the overall delay are derived. As in the optimization technique for Carry Skip adders proposed by the same authors in [26], the resulting design procedure is based on two steps, the first of which leads to a nearly-optimum CSL adder with a lower number of bits than the required value. In the second step, the nearly-optimum adder is completed by adding bits in proper groups, while minimizing the delay increase due to the insertion of further bits.

As in the Carry Skip adder optimization technique in [26], the strategy proposed is simple enough to be used in pencil-and-paper design. Being based on a rigorous timing analysis, the strategy is systematic and provides insight into an optimal CSL. In addition, an approximate closed-form delay expression as a function of the full adder and MUX delay is provided to estimate the minimum achievable delay before the design. In this manner, the gate-level design is related to the transistor-level design, allowing for better optimization at both levels. The proposed strategy has been validated by comparing the resulting delay to the optimum block sizing obtained through exhaustive analysis. Results confirm that this strategy leads to an optimum adder in typical cases, and only in a very few cases does it have a delay greater than the minimum by at most 5.7%.

The timing analysis and the optimization of CSL are addressed in Section 2, and a practical design strategy is derived in Section 3. This design procedure is then clarified through an example and validated for a wide range of cases in Section 4. Finally, conclusions are reported in Section 5, while routine calculations have been placed in three appendices to improve the readability of the paper.

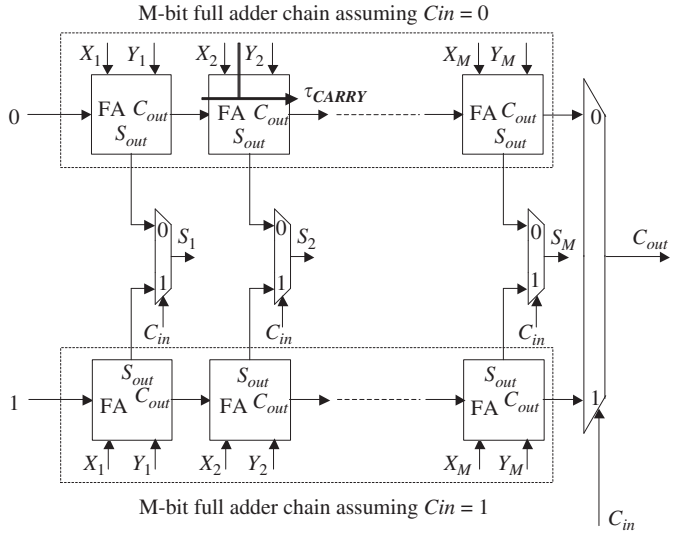

Fig. 1. Structure of an *M*-bit full adder chain in a carry-select adder.

2. Timing analysis of carry-select adders

Let us consider a parallel *N*-bit adder, whose operands $X=X_N$, X_{N-1}, …, X_1 and $Y=Y_N$, Y_{N-1}, …, Y_1 are applied at time $t=0$. The i-th digit S_i of the sum output is given by

$$S_i = X_i \oplus Y_i \oplus C_{i-1} \tag{1}$$

with C_{i-1} being the carry output generated by the $(i-1)$-th digits, or equivalently the carry input that affects the evaluation of the i-th digit, which is recursively defined by the following relationship

$$C_i = X_i Y_i + C_{i-1}(X_i \oplus Y_i). \tag{2}$$

In a Carry-Select adder, relationships (1)–(2) are implemented by cascaded full adder gates grouped into *Q* blocks. As depicted in Fig. 2, the generic j-th block evaluates M_j digits of the sum output (with $j=1$, …, Q) and its carry output $C_{out,j}$ (i.e., the carry output associated with its last digit).

In the generic j-th block, after multiplexer MUX_j selects its correct carry output, it receives input signals from the carry output of the two full adder chains within time $t_{in,j}$ equal to [5]

$$t_{in,j} = M_j \tau_{CARRY} \tag{3}$$

where the full adder carry delay τ_{CARRY} is defined as the time needed to generate its carry output once all its inputs have been applied (assuming the worst case where the carry propagates

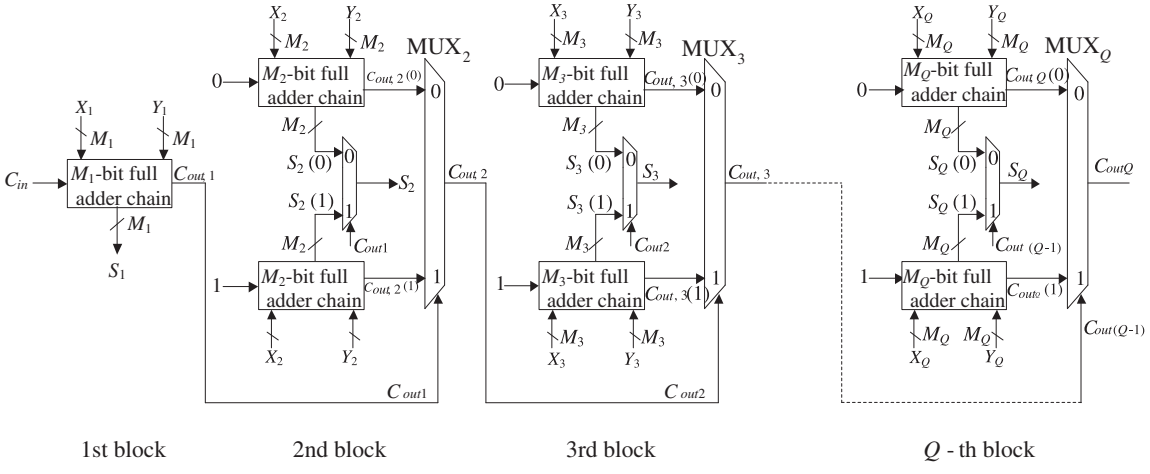

Fig. 2. Architecture of an *N*-bit carry-select adder with *Q* blocks.