INTEGRATION, the VLSI journal 48 (2015) 109–115

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

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

Energy efficient hybrid adder architecture

Shmuel Wimer [a,b,*], Amnon Stanislavsky [a,c]

[a] Technion, EE Faculty, Haifa, Israel
[b] Bar-Ilan University, Engineering Faculty, Ramat-Gan, Israel
[c] Intel Corporation, Haifa, Israel

ARTICLE INFO

Article history:
Received 21 January 2014
Received in revised form
17 June 2014
Accepted 17 June 2014
Available online 30 June 2014

Keywords:
Adders
Hybrid adders
Low-energy
VLSI design

ABSTRACT

An energy efficient adder design based on a hybrid carry computation is proposed. Addition takes place by considering the carry as propagating forwards from the LSB and backwards from the MSB. The incidence at a midpoint significantly accelerates the addition. This acceleration together with combining low-cost ripple-carry and carry-chain circuits, yields energy efficiency compared to other adder architectures. The optimal midpoint is analytically formulated and its closed-form expression is derived. To avoid the quadratic *RC* delay growth in a long carry chain, it is optimally repeated. The adder is enhanced in a tree-like structure for further acceleration. 32, 64 and 128-bit adders targeting 500 MHz and 1 GHz clock frequencies were designed in 65 nm technology. They consumed 11–18% less energy compared to adders generated by state-of-the-art EDA synthesis tool.

1. Introduction

With the explosion of mobile computers and other portable devices, low-power and low-energy design became a must. Power and energy go hand in hand; power reduction leads to lower energy consumption over a fixed time span. Arithmetic circuits are considerable contributors of power and energy in computation intensive applications and require therefore a careful power-delay design tradeoff [1; Ch. 26].

Addition is a fundamental arithmetic operation for which a wide variety of algorithms and methods exist [2]. Many alternatives for adder architectures have been invented [1 Ch. 5–8] with emphasis on their VLSI circuit implementation [3]. *Carry-lookahead* (CLA) [1 Ch. 6], *carry-skip* [4], and *carry-select* [5] adder architectures, among many others, present different area-delay-power tradeoffs. Several works studied energy-efficient adders. While in [6,7] basic full-adder cells were proposed, in [8,9] *carry-propagate* adders were compared. It was noted in [9] that faster arithmetic circuits can be more energy efficient, a direction taken by our work.

This paper proposes a hybrid adder where addition takes place by considering the carry as propagating forwards from the least significant bit (LSB) and backwards from the most significant bit (MSB). The incidence at a midpoint significantly accelerates the addition. The acceleration together with combining low-cost ripple-carry and carry-chain circuits, yields energy efficiency compared to other adder architectures.

Early knowledge of the most significant carry (MSC) is very useful for addition acceleration. Given a target clock cycle (delay constraint), speeding-up MSC computation enables power and energy reduction by transistors downsizing, usage of high threshold voltage and voltage scaling. The authors in [10–12] proposed a method to obtain the MSC based on its dependency only on the most significant bits (MSBs). Though it yielded similar time complexity as conventional CLA, the resulting delay was improved, hence consuming less energy. In [13] a method for sign detection in a Binary Signed-Digit (BSD) number system based on optimized reverse tree structure was proposed. It focused on time-area (and hence energy) efficient generation of the carry-out and was shown to be advantageous compared to BSD CLA implementation. Computation of MSC was also proposed for address generation of FFT circuits [14,15]. All the methods in [10–15] did not use the availability of MSC for accelerating the computation of the sum bits of the adder, which this work does, enabling to tradeoff timing for power reduction and energy efficiency.

The worst-case length of the MSC chain is n bits and it occurs when the addend and augend are complementary of each other. The above methods proposed to accelerate MSC computation by considering the entire n bits. This paper considers the MSC with less than n bits. It does so by a hybrid combination of LSB low-cost ripple-carry and MSB carry-chain in a balanced manner, yielding about $2\times$ reduction of the worst-case delay. Such delay reduction enables to save energy by using small and low-power transistors.

Hybrid adder architecture was also proposed in [19]. It improved the area-delay curve by offering a wider range of

* Corresponding author at: Bar-Ilan University, Engineering Faculty, Israel. Tel.: +972 3 5317208; fax: +972 3 7384051
E-mail address: wimers@biu.ac.il (S. Wimer).

http://dx.doi.org/10.1016/j.vlsi.2014.06.002
0167-9260/

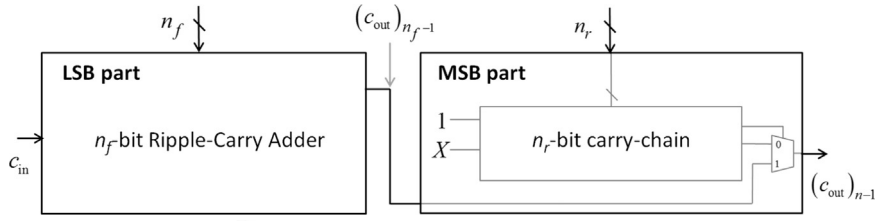

Fig. 1. The architecture of a complete n-bit hybrid adder.

power-delay optimization, obtained by concatenating several sub-groups of various, independent, adder architectures. Sub-groups optimal size was set by solving an appropriate ILP optimization. Our approach is different. The LSB and MSB parts are working in parallel, where the sums and MSC of the MSB part consider the LSB part. Another advantage is the scalability, allowing repetition of the hybrid LSB-MSB mix in wide adders, whereas in [19] the mix is fixed, determined by the adder size, and non-repetitive.

The rest of the paper is organized as follows. Section 2 describes the basic structure of the hybrid adder and its underlying logic. Section 3 discusses its circuit implementation, yielding the minimum worst-case delay. Section 4 proposes speeding-up the addition by using the hybrid adder in a tree-like structure. A complete VLSI design is described in Section 5 and compared with carry-skip adders and adders synthesized by commercial EDA tools. We conclude in Section 6.

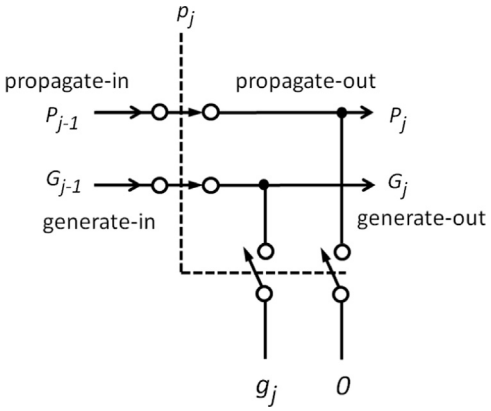

Fig. 2. A basic chain bit.

2. Carry propagation line

Shown in Fig. 1, the idea of hybrid addition is to divide the carry computation into two parts working independently in parallel, thus shortening the critical path. The carry is propagating from bit 0 and from a midpoint bit. The incidence of the paths at the midpoint is used to validate the midpoint-to-MSB sum bits.

Small area and power efficiency is achieved by using the hybrid architecture in Fig. 1. While the LSB-to-midpoint part is implemented by an ordinary low-cost ripple-carry, the MSB-to-midpoint is implemented by a carry-chain. This mix is shown to speed-up computation with a small circuitry overhead, thus yielding energy efficiency. Pure ripple carry would result in smaller area but long carry propagation delay, whereas pure carry-chain would speed up carry propagation on the expense of area growth due to the extra chain circuitry.

The above idea is subsequently elaborated. Consider the addition of two n-bit numbers $A=(a_{n-1},\ldots,a_0)$ and $B=(b_{n-1},\ldots,b_0)$. Let $p_i=a_i\oplus b_i$ and $g_i=a_i\cdot b_i$, $0\le i\le n-1$, be their propagate and generate signals, respectively. Propagate and generate signals spanned across several bits are recursively defined by $P_i\triangleq p_iP_{i-1}$ and $G_i\triangleq p_iG_{i-1}+\overline{p}_ig_i$, where $s\le i\le t$ is the bit range of interest. The initialization of P_{s-1} and G_{s-1} is subsequently discussed.

A basic chain cell is first proposed. It is then combined in the MSB part of a chain comprising n_f LSBs and n_r MSBs, $n=n_f+n_r$. Let P_j assert that the MSC is propagating along $j-n_f+1$ bits. Defining $P_{n_f-1}\triangleq 1$, there exists $P_j\triangleq p_jP_{j-1}=\prod_{i=n_f}^{j}p_i$, $n_f\le j\le n-1$. The basic bit of the chain is shown in Fig. 2. It has propagation and generation tracks, implemented with CMOS pass-gate switches (the transistor schematic is shown in Fig. 11).

The MSB part comprises a carry-chain formed by connecting n_r cells as shown in Fig. 3, similar to Manchester carry-chain [1]. The role of the upper track is to propagate P_j. In case of $p_j=1$ the pass-gates are closed and P_{j-1} is transferred. The role of the lower track is to pass the carry in case it was killed or generated at a former bit. The cumulative generate signal is $G_j\triangleq p_jG_{j-1}+\overline{p}_jg_j$, where $G_{n_f-1}\triangleq X$ (do not care). The selection of $(c_{out})_{n-1}$ is made as in a carry-skip adder with a 2:1 MUX controlled by P_{n-1}. If $P_{n-1}=0$ it happens that a carry was killed or generated somewhere between bit n_f and bit $n-1$, so $(c_{out})_{n-1}$ is selected from the generation track. If $P_{n-1}=1$, then $(c_{out})_{n-1}=(c_{out})_{n_f-1}$.

Though the carry-out is quickly computed and available for further use, we would like the internal sums to be valid no later than the carry-out. To this end we use the signals $(c_{out})_{n_f-1}$ and P_{j-1} of the upper track in Fig. 4. If $P_{j-1}=0$, $(c_{out})_{j-1}\triangleq(c_{in})_j$ has been properly computed, regardless of $(c_{out})_{n_f-1}$, and consequently bit j properly produced its sum. For $P_{j-1}=1$ the value of $(c_{out})_{n_f-1}$ is the proper $(c_{in})_j$ value. Buffering may be required to drive the high load incurred by the n_r bits.

The architecture of a complete n-bit hybrid adder is illustrated in Fig. 1. The n_f LSBs compute ordinary ripple-carry, while the n_r MSBs compute the carry by using carry-chain, and include the circuit of Fig. 4.

$(c_{out})_{n-1}$ in Fig. 1 is either the outcome of the first n_f LSBs or the outcome of the last n_r MSBs. It is reminiscent of a 2-block carry-skip adder. In ordinary carry-skip adder the optimal block sizes are determined by bit counting arguments [1], whereas the proposed hybrid adder determines the size of the blocks by accurately modeling and equating the propagation delay of the underlying blocks. The implied delay equations are then solved to find the optimal combination of the two circuit types, shown experimentally to consume less energy compared to other adders.

3. Finding the optimal midpoint

In the sequel the midpoint n_f is determined to yield minimum worst-case critical path delay. We first consider a small adder. A wider, more complex adder will also be discussed. It requires a long carry-chain in its MSB part, and buffers must therefore be inserted to avoid the quadratic delay growth occurring in pass-gate chain. The critical path in Fig. 1 passes either in the LSB or the MSB part, which are independent of each other. The final MUX which selects c_{out} from either of the two is common to both and therefore does not affect the optimal midpoint.