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/© 2014 Elsevier B.V. All rights reserved.

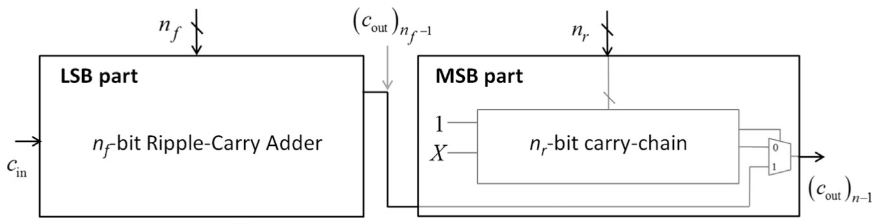


Fig. 1. The architecture of a complete n-bit hybrid adder.

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.

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 $A=(a_{n-1},\dots,a_0)$ and $B=(b_{n-1},\dots,b_0)$. 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 $(c_{out})_{n-1}=(c_{out})_{n_f-1}$.

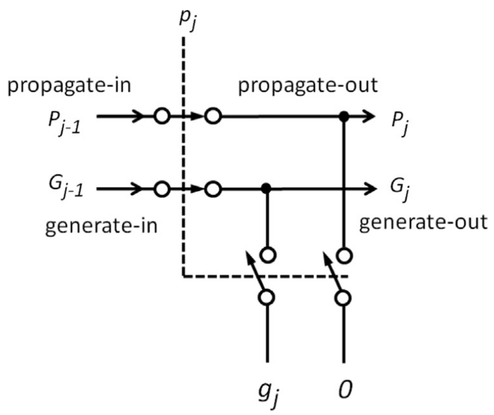


Fig. 2. A basic chain bit.

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.

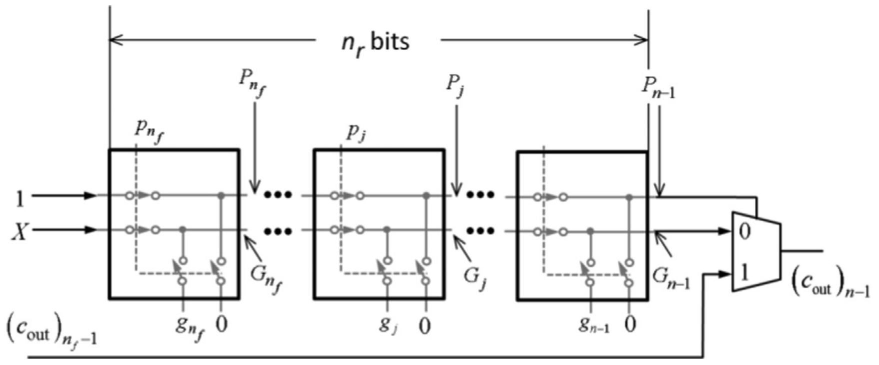


Fig. 3. A complete chain comprising n_r bits.

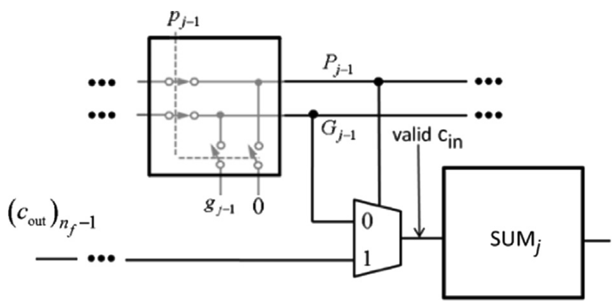


Fig. 4. Computation of the sum of bit j.

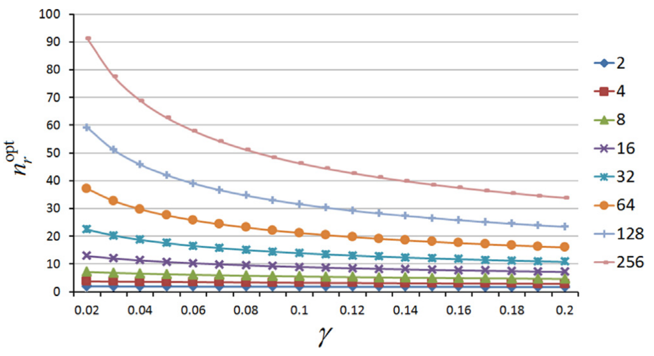


Fig. 5. Length of chain as function of adder size n and γ.

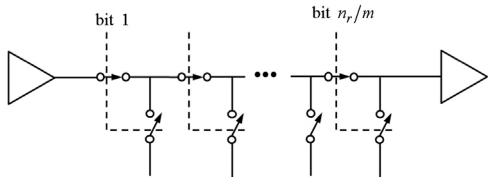


Fig. 6. Inserting repeaters into carry-chain.

The MSB part has potentially two critical paths of $(c_{out})_{n-1}$ illustrated in Fig. 3, one along the upper track and the other along the lower track, each comprising n_r pass-gates. The delay through a pass-gate is smaller than the internal carry delay in a full-adder, so $n_r > n_f$ is expected. However, while the LSB delay grows linearly with n_f, the MSB has quadratic growth with n_r. The optimal midpoint occurs therefore when the two delays are equal. Let α denote the c_{in}-to-c_{out} delay of a full-adder, r be the pass-gate intrinsic resistance and c its diffusion capacitance in all the pass-gates. Using Elmore model [16], the propagation delay of the MSB critical path is given by the following expression

$$0.7\sum_{i=1}^{n_r} r\sum_{j=i}^{n_r} c = 0.7rc\frac{n_r(n_r-1)}{2}. \quad (1)$$

Equating the LSB and the MSB delays, we obtain

$$\alpha n_f = \frac{0.7rc}{2}n_r(n_r-1). \quad (2)$$

Let $\gamma = 0.7rc/2\alpha$ be the ratio of the pass-gate intrinsic RC delay to the full-adder c_{in}-to-c_{out} delay. Substitution of $n_f = n - n_r$ in (2) yields the quadratic equation

$$\gamma n_r^2 + (1-\gamma)\, n_r - n = 0. \quad (3)$$

The optimal mid-point in (4) is the positive root obtained by solving the quadratic equation in (3).

$$n_r^{opt} = \frac{1}{2\gamma}\left[\gamma - 1 + \sqrt{(\gamma-1)^2 + 4\gamma n}\right]. \quad (4)$$

Fig. 5 shows the length n_r^{opt} of the optimal carry-chain as a function of adder size n and γ. It approaches $\sqrt{n/\gamma}$ with the increase of n. The advantage of hybrid carry computation becomes marginal since $n_r^{opt}/n \to 0$ while $n_f/n \to 1$, thus the adder mostly behaves like ordinary ripple-carry. To mitigate this problem, the carry-chain must be repeated.

Let the carry-chain be divided into m buffered sub-chains as shown in Fig. 6. To save hardware and shorten delay, buffering inverters are used, but then in every second sub-chain the signals 0 and g_j in Fig. 2 are replaced by 1 and $\bar{g}_j$, respectively. To find the number of repeaters minimizing the MSB delay, let d be the repeater's intrinsic delay, r' its driving resistance, and c' its input capacitance. Using Elmore model, the input-to-input delay of a sub-chain is

$$d + 0.7\left[r'\left(c' + \sum_{i=1}^{n_r/m} c\right) + \Sigma_{i=1}^{n_r/m} r\left(c' + \sum_{j=i}^{n_r/m} c\right)\right]. \quad (5)$$

Reasonably assuming that $r' \cong r$ and $c' \cong c$ turns (5) into

$$d + \frac{0.7rc}{2}\frac{n_r}{m}\left(\frac{n_r}{m}+1\right). \quad (6)$$

Equating the ripple-carry and carry-chain delays, we obtain

$$\alpha n_f = m\left[d + \frac{0.7rc}{2}\frac{n_r}{m}\left(\frac{n_r}{m}+1\right)\right], \quad (7)$$

which after substitution of $\gamma = 0.7rc/2\alpha$ and $n_f = n - n_r$ in (7) yields the following quadratic equation

$$\frac{\gamma}{m^2}n_r^2 + \left(1 + \frac{\gamma}{m}\right)n_r + \left(\frac{d}{\alpha}m - n\right) = 0. \quad (8)$$

With some reordering, the positive root of (8) is

$$n_r^{opt} = \frac{1}{2\gamma}\left[-(m^2+\gamma m) + \sqrt{m^4 + 2\gamma\left(1 - \frac{2d}{\alpha}\right)m^3 + (\gamma^2 + 4\gamma n)m^2}\right]. \quad (9)$$

Examining several CMOS technologies, and in particular the 65 nm used in this work, we found that $d/\alpha \cong 0.1$. Recalling that $\gamma = 0.7rc/2\alpha$ is the ratio of the pass-gate intrinsic RC delay to the full-adder c_{in}-to-c_{out} delay, and hence $\gamma < 1$, it follows that $\gamma^2 \ll 4\gamma n$. Eq. (9) can therefore be simplified to

$$n_r^{opt} = \frac{1}{2\gamma}\left[-(m^2+\gamma m)+\sqrt{m^4+1.6\gamma m^3+4\gamma n m^2}\right]. \qquad (10)$$

The value of n_r^{opt} obtained by (10) ensures that the LSB and MSB parts have the same delay $t = \alpha n_f = \alpha(n - n_r^{opt})$. To find the optimal number of repeaters we derivate t by m and solve the equation

$$\frac{dt}{dm} = \frac{\alpha}{2\gamma}\left[2m+\gamma-\frac{2m^2+2.4\gamma m+4\gamma n}{\sqrt{m^2+1.6\gamma m+4\gamma n}}\right]=0,$$

whose solution is the number m^{opt} of repeaters that minimizes the delay. Squaring and reordering yields the following cubic equation for m^{opt}

$$m^3+2.05\gamma m^2+2\gamma(\gamma-2n)m+5\gamma(n\gamma-4n^2)=0. \qquad (11)$$

Fig. 7 illustrates the solution of (11) for various adder sizes and $0.02 \le \gamma \le 0.2$.

The value of m^{opt} can be substituted in (10) to obtain the corresponding optimal midpoint n_f as summarized in Table 1 for γ=0.1. For a repeated chain, n_r weakly depends on γ. The chain delay $\alpha n_f = \alpha(n-n_r)$ remains small for very wide adders as shown in Table 1. The adders implemented and experimented in Section 5 shows similar behavior to that in Fig. 7 and Table 1.

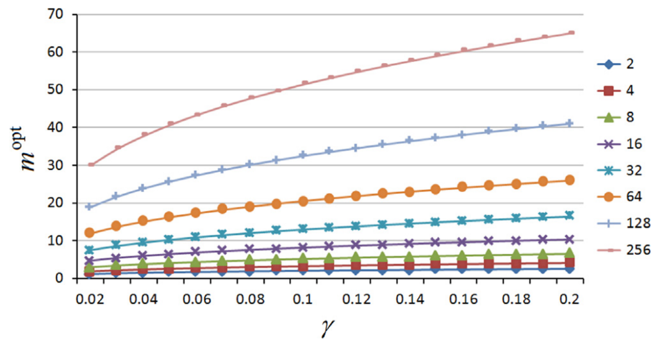


Fig. 7. The optimal number of repeaters in a carry-chain.

Table 1
Dependence of optimally repeated carry-chain on adder size.

Adder size n	32	64	128	256
Repeater count m^{opt}	12	19	31	50
Ripple-carry size n_f	4	6	8	12
Carry-chain size n_r	28	58	120	244

4. Logarithmic time hybrid addition

Hybrid addition can be sped-up by dividing the n bits into two halves and then applying computation in parallel for each part. Setting $n'=n/2$, the n'-bit adder is designed for minimum delay and its corresponding optimal midpoint n'_f, $n'=n'_f+n'_r$, and repeaters (if applies) are found according to the equations in Section 3. Its architecture is illustrated in Fig. 8. As shown below, it requires the addition of carry-chain logic at the LSB part of the $n/2$ MSBs group for the production of propagate and generate signals.

Since the two halves work in parallel, the carry-in of the $n/2$ MSBs group is unknown, as it is valid only after the computation of the $n/2$ LSBs group is done. To exploit the parallelism we make the hypotheses that all unknown carry-in are 1, as shown in Fig. 8. Comparison of $(c_{in})_{n/2}=1$ with $(c_{out})_{n/2-1}$ validates the hypothesis.

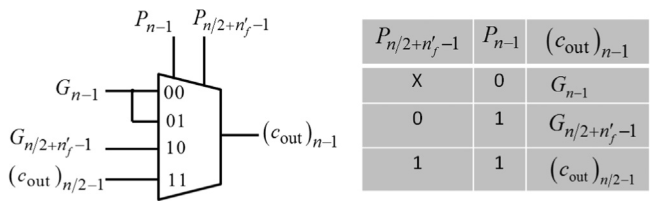
It may unfortunately happen that $(c_{out})_{n/2-1}=0$, making $(c_{out})_{n-1}$ invalid. To properly determine $(c_{out})_{n-1}$, a carry-chain of the n'_f LSBs in the $n/2$ MSB part of Fig. 8 is implemented on top of its underlying ripple-carry adder, obtaining the signals $P_{n/2+n'_f-1}$ and $G_{n/2+n'_f-1}$. This enables to properly calculate $(c_{out})_{n-1}$ as shown in Fig. 9. The 2:1 MUX producing $(c_{out})_{n-1}$ in Fig. 2 is replaced by an appropriate 4:1 MUX. To obtain the proper LSBs sums in the MSB group, their corresponding carry-in is selected with a 2:1 MUX, similarly to Fig. 3.

The above division can repeat itself. While the computation time within each $n/2^q$-bit group, $0 \le q \le \log_2 n$, is decreasing, the logic for validating all the hypotheses becomes more complex and hence more time and power consuming. A further division for q=2 is shown in Fig. 10, where $n'_f+n'_r=n/4$.

The n'_f midpoint is determined by first optimizing a $n/4$-bit group as described in Section 3. Since all the groups except the first make the hypothesis that carry-in is 1, the determination of the value of $(c_{out})_{n-1}$ must validate all those hypotheses. Table 2 extends the table in Fig. 9. The construction of appropriate selection logic is straightforward. We can deduce from Table 2 the expression of $(c_{out})_{n-1}$ based on each of the 2^q-group hybrid adders.

5. Experimental results

Hybrid adders of 32, 64 and 128 bits, targeting 500 MHz and 1 GHz, have been implemented. Those were compared with low-



$P_{n/2+n'_f-1}$	P_{n-1}	$(c_{out})_{n-1}$
X	0	G_{n-1}
0	1	$G_{n/2+n'_f-1}$
1	1	$(c_{out})_{n/2-1}$

Fig. 9. Determination of $(c_{out})_{n-1}$.

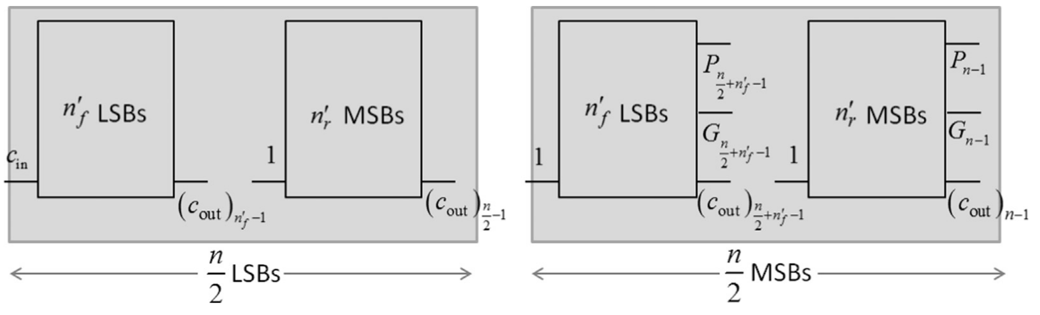


Fig. 8. Acceleration by parallel hybrid adders.

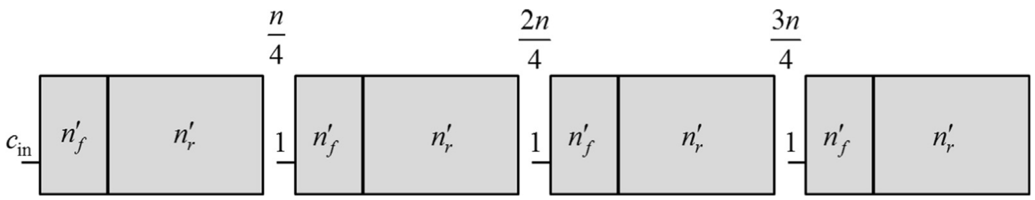


Fig. 10. A 4-group hybrid adder.

Table 2
The expression of $(c_{out})_{n-1}$ for a 4-group hybrid adder.

$P_{n/4+n'_f-1}$	$P_{2n/4-1}$	$P_{2n/4+n'_f-1}$	$P_{3n/4-1}$	$P_{3n/4+n'_f-1}$	P_{n-1}	$(c_{out})_{n-1}$
×	×	×	×	×	0	G_{n-1}
×	×	×	×	0	1	$G_{3n/4+n'_f-1}$
×	×	×	0	1	1	$G_{3n/4-1}$
×	×	0	1	1	1	$G_{2n/4+n'_f-1}$
×	0	1	1	1	1	$G_{2n/4-1}$
0	1	1	1	1	1	$G_{n/4+n'_f-1}$
1	1	1	1	1	1	$(c_{out})_{n/4-1}$

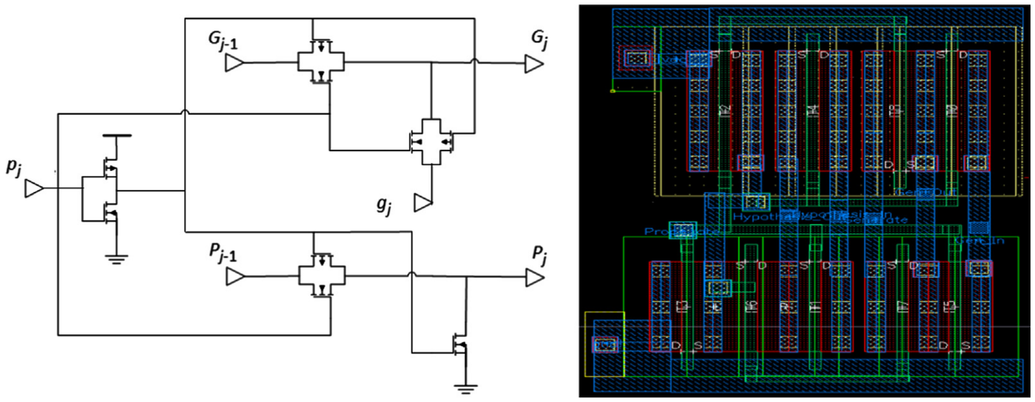


Fig. 11. The chain-cell schematics and its corresponding layout.

Table 3
Performance of 16-bit hybrid adder for various midpoints.

	Midpoint n_f	2	3	4	5	6	7	8	9	10	11	12	13	14
500 MHz	Delay [ns]	2.18	1.96	1.86	1.64	1.57	1.65	1.75	1.95	2.15	2.36	2.58	2.81	3.00
	Area [sq. μ]	231	225	221	214	208	203	197	191	186	179	174	168	162
	Energy [norm.]	1.54	1.35	1.26	1.07	1.0	1.03	1.06	1.14	1.22	1.30	1.37	1.45	1.49
1.0 GHz	Delay [ns]	1.07	1.03	0.99	0.95	0.91	0.87	0.91	1.01	1.12	1.18	1.3	1.38	1.47
	Area [sq. μ]	330	304	267	258	260	256	277	414	423	450	454	483	502
	Energy [norm.]	1.58	1.4	1.19	1.1	1.06	1.0	1.13	1.87	2.12	2.38	2.65	2.99	3.13

power adders generated by Synopsys Design Compiler EDA tool, widely used by industry [17]. Other adders have been designed for reference. We used carry-skip and Kogge–Stone adders, shown in [9] being energy efficient at 130 nm technology and 1 GHz clock frequency. The Design Compiler selects the best adder architecture and the size of its underlying logic cells, such that the resulting circuits meet the target clock cycle with minimum power (and hence energy).

The hybrid, carry-skip and Kogge–Stone adders were implemented with a semi-custom design flow, using a standard CMOS cell library of Virage Logic, designed in IBM 65 nm technology. The pass-gate cell in Fig. 2 was designed and characterized in the typical corner, and then appended to the cell library. The usage of pass-gates and their robustness in the design of low-energy addition circuits has been discussed in [18]. Its schematics and layout are shown in Fig. 11.

Table 4
Comparison of various adders.

		32-bit		64-bit		128-bit	
		Delay [ns]	**Energy [norm.]**	**Delay [ns]**	**Energy [norm.]**	**Delay [ns]**	**Energy [norm.]**
500 MHz	Hybrid	**1.71**	**0.87**	**1.81**	**0.817**	**1.85**	**0.849**
	Design Compiler	**1.82**	**1.0**	**1.845**	**1.0**	**1.795**	**1.0**
	Carry-skip	**1.68**	**1.19**	**1.8**	**1.14**	**2.3**	**1.43**
	Kogge–Stone	**1.41**	**1.05**	**1.50**	**0.976**	**1.61**	**1.02**
1 GHz	Hybrid	**0.91**	**0.844**	**0.94**	**0.834**	**1.02**	**0.845**
	Design Compiler	**0.88**	**1.0**	**0.9**	**1.0**	**0.987**	**1.0**
	Carry-skip	**0.86**	**1.4**	**1.2**	**1.98**	**1.6**	**2.42**
	Kogge–Stone	**0.81**	**1.02**	**0.83**	**0.946**	**0.87**	**0.954**

The hybrid adders were implemented with 16-bit groups (non-repeated carry-chain), as in Fig. 10. The real design matches the optimal midpoint $n_f = n - n_r^{opt}$ obtained by Eq. (4). Groups were generated and simulated for all possible midpoints. Fig. 5 shows that the theoretical optimal midpoint for $n=16$ is $n_f=7$. Based on their physical layout, Table 3 presents the delay, area and energy of the adders built with various midpoints. Energy was obtained by power-delay product and was normalized with respect to the smallest value obtained. The minima are colored in red and nicely agree with the theory. Considering area, the underlying sum bits used full-adder library cells, whereas the carry-chain was implemented separately with the pass-gate switches shown in Fig. 11. Notice that delay constraints were met for $3 \le n_f \le 9$.

Hybrid adders of 32, 64 and 128 bits were designed for 500 MHz and 1 GHz, based on Fig. 10 with its supplementary logic derived from Table 2. The optimal 16-bit groups of Table 3 were used. Table 4 compares the energy of the hybrid adder with the best adder synthesized by the Design Compiler EDA tool and the reference carry-skip and Kogge–Stone adders. It is important to note that the clock speed was selected independently of the adder types. Each adder was designed to meet 2.0 ns and 1.0 ns delay constraints with minimum power. Table 4 shows that the hybrid, Design Compiler and Kogge–Stone adders met those timing constraints, while the carry-skip adder failed for 64-bit and 128-bit 1 GHz.

To measure their propagation delay and energy, all adders were simulated by SPICE, with loads similar to the test bench in Fig. 2 of [9]. To ensure that the full carry path from LSB to MSB is exercised, the addend and augend have been selected complementary of each other.

The energy was obtained by power-delay product and was normalized to that of the synthesized adder. The comparison in 500 MHz shows that the 32-bit hybrid adder outperforms the synthesized adder and Kogge–Stone by 13% and 18% less energy, respectively. For 64-bit the improvements are 18% and 16%, respectively, and for 128-bit the improvements are 15% and 17%, respectively. Similar improvements in the range of 11–16% are shown for 1 GHz.

6. Conclusions

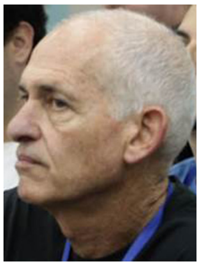
An energy efficient hybrid adder was studied. Analytical expressions of its optimal midpoint and repeaters have been derived, which agreed with the experimental results. Comparison of 32, 64 and 128-bit adders targeting 500 MHz and 1 GHz showed 11–18% less energy consumption compared to state-of-the-art EDA synthesized adders.

Acknowledgments

The authors wish to thank S. Goel of the Technion VLSI Lab for his support in the adder designs, and Prof. A. Kolodny for helpful discussions. They are also grateful for the useful comments made by the anonymous referees, which helped in improving the manuscript.

References

[1] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, Oxford University Press, Chapter 26, 2009.
[2] I. Koren, Computer Arithmetic Algorithms, A.K. Peters, 2nd Edition 2002, Chapter 5, 2001.
[3] N. Weste, D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, Pearson, 4th Edition 2009, Chapter 11, 2005.
[4] A. Guyot, B. Hochet, J.-M. Muller, A way to build efficient carry-skip adders, IEEE Trans. Comput. 36 (10) (1987) 1144–1152.
[5] A. Tyagi, A reduced-area scheme for carry-select adders, IEEE Trans. Comput. 42 (10) (1993) 1163–1170.
[6] J.-F. Lin, Y.-Y. Hwang, M.-H. Sheu, C.-C. Ho, A novel high-speed and energy-efficient 10-transistoe full-adder design, IEEE Trans. Circuits Syst. I 54 (5) (2007) 1050.
[7] A. Saydeh, H. Al-Asaad, Survey and evaluation of low-power full-adders cells, in: Proceedings of the IEEE International Conference on VLSI, 2004, pp. 332–338.
[8] C. Nagendra, M.J. Irwin, R.M. Owens, Area–time–power tradeoffs in parallel adders, IEEE Trans. Circuits Syst. II 43 (10) (1996) 689–702.
[9] M. Vratonjic, R.B. Zeydel, V.G. Oklobdzija, Low- and ultra low-power arithmetic units: design and comparison, in: Proceedings of the International Conference on Computer Design (ICCD05), 2005, pp. 249–252.
[10] D.R. Lutz, D.N. Jayasima, The half-adder form and early branch condition result, in: Proceedings of the 13th IEEE Symposium on Computer Arithmetic (ARITH13), 1997, pp. 266–273.
[11] T. Lang, J.D. Bruguera, Multilevel reverse-carry computation for comparison and for sign and overflow detection in addition, in: Proceedings of the International Conference on Computer Design (ICCD-99), 1999, pp. 73–79.
[12] T. Lang, J.D. Bruguera, Multilevel reverse-carry addition: single and dual adders, J. VLSI Signal Process. 33 (2003) 55–74.
[13] T. Srikanthan, S.K. Lam, Suman Mishra, Area–time efficient sign detection technique for binary signed-digit number system, IEEE Trans. Comput. 53 (1) (2004) 69–72.
[14] Adder capable of usual addition and reverse carry addition, US Patent no. 4,897,808, 1990.
[15] Address sequence generation by means of reverse carry addition, US Patent no. 4,974,188, 1990.
[16] A.B. Kahng, K. Masuko, S. Andmuddu Analytical delay models for VLSI interconnects under ramp input, in: Proceedings of the IEEE International Conference on Computer-Aided Design (ICCAD1996), 1996, pp. 30–36.
[17] Synopsys Design Compiler, version E-2010.12-SP2.
[18] S. Goel, A. Kumar, M.A. Bayoumi, Design of robust, energy-efficient full adders for deep-submicrometer design using hybrid-CMOS logic style, IEEE Trans. Very Large Scale Integr. Syst. 14 (12) (2006) 1309–1321.
[19] Jeong-Gun Lee, Jeong-A. Lee, Deok-Young Lee, Better area-time tradeoffs in an expanded design space of adder architecture by parameterizing bit-width of various carry propagated sub-adder-blocks, in: Proceedings of the IEEE 6th International Workshop on System-on-Chip for Real-Time Applications, 2006, pp. 10–14.



Shmuel Wimer received the B.Sc. and M.Sc. degrees in mathematics from Tel-Aviv University, Tel-Aviv, Israel, and the D.Sc. degree in electrical engineering from the Technion-Israel Institute of Technology, Haifa, Israel, in 1978, 1981 and 1988, respectively . He worked for 32 years at industry in R&D, engineering and managerial positions, for Intel from 1999 to 2009, and prior to that for IBM, National Semiconductor and Israeli Aerospace Industry (IAI). He is presently an Associate Professor with the Engineering Faculty of Bar-Ilan University, and an Associate Visiting Professor with the Electrical Engineering Faculty, Technion. He is interested in VLSI circuits and systems design optimization and combinatorial optimization.

Amnon Stanislavsky received his B.Sc. degrees in electrical engineering and physics from the Technion-Israel Institute of Technology in 2008, and M.Sc. degree in electrical engineering from the Technion-Israel Institute of Technology in 2013. He works for Intel Corporation since 2012 on physical implementation of digital circuits in the Communications and Storage Infrastructure Group (CSIG). From 2008 to 2012 he worked for Marvell on physical implementation of digital circuits in the Controllers and SoC division. He is also supervising undergraduate projects in the VLSI lab of the Electrical Engineering Faculty at the Technion. The projects include software development for SoC, RTL designs, functional verification, physical implementation and CAD algorithms.