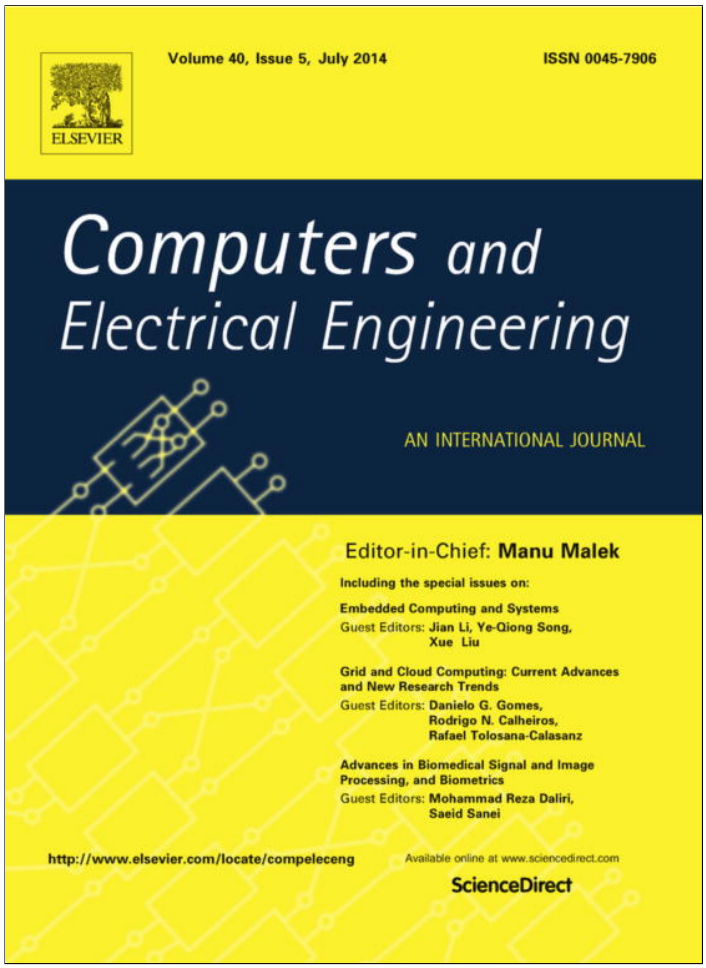


Computers and Electrical Engineering 40 (2014) 1524–1537

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

A low energy dual-mode adder ☆

Shmuel Wimer [a,b,*], Amir Albeck [a], Israel Koren [c]

[a] Bar-Ilan University, Engineering Faculty, Israel
[b] Technion - Israel Institute of Technology, EE Faculty, Israel
[c] University of Massachusetts, ECE Department, United States

ARTICLE INFO

Article history:
Received 21 December 2013
Received in revised form 9 April 2014
Accepted 10 April 2014
Available online 9 May 2014

ABSTRACT

VLSI designs are typically data-independent and as such, they must produce the correct result even for the worst-case inputs. Adders in particular assume that addition must be completed within prescribed number of clock cycles, independently of the operands. While the longest carry propagation of an n-bit adder is n bits, its expected length is only $O(\log_2 n)$ bits. We present a novel dual-mode adder architecture that reduces the average energy consumption in up to 50%. In normal mode the adder targets the $O(\log_2 n)$-bit average worst-case carry propagation chains, while in extended mode it accommodates the less frequent $O(n)$-bit chain. We prove that minimum energy is achieved when the adder is designed for $O(\log_2 n)$ carry propagation, and present a circuit implementation. Dual-mode adders enable voltage scaling of the entire system, potentially supporting further overall energy reduction. The energy-time tradeoff obtained when incorporating such adders in ordinary microprocessor's pipeline and other architectures is discussed.

1. Introduction

Common VLSI synchronous designs are usually data-independent and as such, must target worst-case scenarios, even those that occur with very low probability. Adders, in particular, assume that the execution of an operation must be accurately completed within a prescribed time period, e.g., a single clock cycle, independently of the input operands. To ensure correct results for any input, the hardware involved may significantly grow if high performance (speed, throughput) is desired. Often, adders targeting clock frequencies of several GHz use prefix and carry look-ahead (CLA) architectures which consume large area, power and energy [1–3]. Many other arithmetic circuits (e.g., multipliers) use internally adders and must also consider the worst case carry propagation.

The focus of this paper is on energy minimization. We present an architecture for building adders, called *dual-mode*. The addition completes within a single clock cycle with minimum energy for most operands, and with a very low probability, few additional cycles are required to complete the addition. The proposed architecture incorporates some of the design principles of known synchronous and asynchronous adders. A theoretical energy consumption model is presented, based on carry propagation probabilities. The optimal design point is analytically derived and shown to significantly reduce the energy consumption of the adder and to enable considerable energy reduction of the entire system by voltage scaling.

The interest in designing better adders has existed ever since. Until two decades ago the focus was on speed and area. With the spread of mobile and green computing the focus shifted to power and energy savings. Adder optimization can take

☆ Reviews processed and approved for publication by Editor-in-Chief Dr. Manu Malek.
* 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.compeleceng.2014.04.012
0045-7906/

place at the architecture level [1,2] or at the gate and transistor levels [3]. The most important factor in adder optimization is its *carry propagation probabilities*. While early works considered it when reducing the compute time, more recent publications emphasized the potential power and energy savings. Carry probabilities can also be used to estimate the adder's energy consumption [4,5]. We subsequently review several asynchronous and synchronous adder designs targeting performance enhancement and power reduction, two objectives that can usually be traded off.

In designs where the adder is on the critical delay path, the relaxation of its timing constraints opens many opportunities for energy savings and performance improvement as described in this paper. We show how the power consumed by the adder can be significantly reduced by relaxing its carry propagation delay constraints. In Digital Signal Processors (DSPs) and image processors comprising hundreds of adders performing transformations and filtering, such relaxation can yield significant energy savings with marginal performance degradation.

The rest of the paper is organized as follows. Section 2 overviews prior work related to the adder proposed in this paper. In Section 3 the carry propagation probability, that is the basis of the analysis, is presented. Section 4 describes the proposed dual-mode adder's architecture, and its energy consumption is analyzed in Section 5. Section 6 discusses the circuit implementation and energy-time tradeoff, showing that minimum circuit energy is achieved when the adder is designed such that its delay meets the expected longest carry propagation. Section 7 describes how the dual-mode adder enables voltage scaling and the implied energy reduction potential. Section 8 presents experimental results obtained from an industrial image processor. It also compares the power consumed by the dual-mode architecture to that another variable-latency adder. Section 9 concludes the discussion.

2. Prior works

While the data-dependent adder presented in this paper is proposed for synchronous designs, asynchronous adders by their very definition are data-dependent, consuming only the required computation time, determined by the adder's arguments. Still, some asynchronous adders follow a data-independent approach. A method known as *bundled data* indirectly detects when the computation completes, by using a worst-case delay model. It is designed to exceed the longest path through the computation circuit (which for addition is the carry propagation path) [6,7]. This delay may be emulated by an inverter chain. Its main advantage is that a standard, low power and small area robust implementation can be used. Its disadvantage though, is that the completion time is determined by the worst-case computation, regardless of the actual data inputs.

One type of a data-independent asynchronous adder uses *completion detection*, which directly detects when the carry propagation completes. Its carry-path is typically implemented as a *dual-rail*, where each bit is mapped to a pair of wires, encoding both the value and validity of the carry [8]. It is advantageous over bundled data as the data-path itself directly indicates when the computation completes, so no time is wasted. Its disadvantage though is that a *completion detection network* is required, adding several gate delays between operation completion and its detection. Another disadvantage is the extra wiring and switching activity that increase the area and power consumption. An alternative method, called *current-sensing completion detection*, avoids the detection network, but requires special current sensors. The latter still introduces some gate delays overhead [9] and required considerable area (and hence power) overhead.

In [10] a method for designing asynchronous data-path components, called *speculative completion*, was described. It is based on combining a worst-case approach with early completion detection. A 15–30% speedup of 32-bit and 64-bit for Brent–Kung carry look-ahead and carry-skip adders were reported (compared to the corresponding synchronous designs). The design includes an abortion detection network, which indicates that addition must take place within the full time period required for the worst-case. The abortion conditions imposed in [10] are, unfortunately, only sufficient, but not necessary, thus cases that could benefit from early completion are missed, using longer than necessary computation time. An analysis of the speculative completion method showed that 60–90% of additions could benefit from an early completion, depending on the complexity of the detection network, which grows with its hit accuracy. To improve performance, the speculative completion adder of [10] which used static logic, was implemented with dynamic logic in [11]. This, however, resulted in further power and energy inefficiencies.

Although this work also uses a detection circuit, its overhead is smaller and independent of the required accuracy, while achieving 99% hit rate. Previous works did not analyze the tradeoffs between performance, power, energy, and computation accuracy. Rather, they were based on intuition and simulations of a few specific cases [10] that do not provide a quantitative analysis of the tradeoff among the above objectives. In contrast, this paper presents a probabilistic model combined with circuit parameters, formulating the tradeoff of the design objectives. The optimal design point is shown to correlate with the $O(\log_2 n)$ expected worst-case carry-chain occurring in n-bit addition. The work in [20] took advantage of the $O(\log_2 n)$ expected worst-case and proposed a fast CLA self-timed adder yielding area-time product $\theta(n \log_2 \log_2 n)$.

Accuracy, performance and energy were also traded off in synchronous designs. Energy savings in adders can be achieved by sacrificing accuracy in applications that can tolerate it. In image processing, for instance, the image quality can sometimes be traded off for lower power. In [12] it was proposed to reduce the logic complexity of a full-adder at the transistor level and relax the numerical accuracy in a design of multi-bit adders. In addition to the inherent reduction in switched capacitance, the technique resulted in significant shortening of the critical paths, thus enabling voltage scaling. The approximate adder was used for video and image compression algorithms, achieving up to 69% power savings compared to accurate adders. In [13] a multiplier architecture using inaccurate building blocks was presented. It achieved average power savings of 32–45% over corresponding accurate multiplier designs, with average errors of 1.4–3.3%.

A method for energy savings called *Razor Design* was proposed in [14]. Unlike [12,13] which pay in accuracy, the *razor* method reduces energy at the expense of performance. Voltage is scaled down so that most of the time the underlying logic safely completes its computation. The cases where the clock cycle was insufficient for a safe completion are detected during the execution. For those cases the computation is then repeated either by allocating more clock cycles or by reducing the clock frequency.

A carry-save adder operating in two modes was described in [15], taking advantage of the very low probability of long carry propagation chains to occur. The authors devised an ad-hoc technique to detect whether the operands comply with a "short" operation mode (most often) or a "long" operation mode must take place. The detection was achieved by dividing the adder into two shorter parts with some overlap. The overlapping portion was used to detect the mode and it dictates the probability of the long mode. The shorter worst-case critical path of the *short mode* enabled voltage scaling resulting in power reduction. To compensate for the longer worst-case critical path that may occur in the *long mode*, two clock cycles were allocated for the addition to properly compute. While our adder also has two addition modes, it has a larger power reduction potential as it targets the expected worst-case carry propagation, which is logarithmic rather than the square root as in carry-skip adders. The longest carry-chain reduction of [15] is by a factor smaller than 2, while in ours it is $n/\log n$. Furthermore, [15] is tailored to the specific carry-skip architecture, whereas our method is independent of the adder's architecture.

In an attempt to improve the overall throughput, a technique called *telescopic unit* took advantage of data dependency and its implication on the probability of worst-cases to occur [16]. The improvement is obtained by speeding up the clock signal, such that its cycle suffices for common input cases. Longer computations are split over several cycles. As most of the methods described before, a telescopic unit produces two outputs: the result of the computation and a handshaking hold signal which is activated when the computation requires additional clock cycles to properly complete. Being general and based on synthesis, it results in throughput improvement over a wide range of circuits. Further improvements can, however, be achieved by taking advantage of the specific carry logic, as done in some of the above works.

3. Carry propagation probabilities

n-Bit adders are typically designed to complete the addition in a prescribed time period, regardless of their inputs. This is a worst-case design, resulting in a $O(f(n))$ delay, where $f(n)$ depends on the adder architecture. For ripple-carry adders $f(n) = n$, for carry-skip adders $f(n) = \sqrt{n}$, and for CLA and tree adders $f(n) = \log_2 n$ [1,2]. Often, the faster an adder is, the more area and energy it consumes.

The probability q of a carry to propagate through a single bit is 1/2 and is 2^{-k} through successive k bits. In the following we use the term *k-bit group* for successive k bits. Alternatively, the probability of a carry being either generated or killed in a k-bit group is $1–2^{-k}$. Let $A = (a_{n-1},\ldots,a_0)$ be the addend and $B = (b_{n-1},\ldots,b_0)$ be the augend of an n-bit adder. Assume that these operands are random, independent and uniformly distributed in the range $[0,2^n - 1]$. Let $p_i = a_i \oplus b_i$, $0 \leqslant i \leqslant n-1$, be the propagate signal of a bit. The probability q_k that a carry propagates through $k-1$ successive bits and then stops at bit k is

$$q_k = \Pr\left(\prod_{i=0}^{k-2} p_i = 1\right) \times \Pr(p_{k-1} = 0) = 2^{-k}. \tag{1}$$

The expected length L of the carry propagation chain is therefore

$$L = \sum_{k=1}^{\infty} k q_k = \sum_{k=1}^{\infty} k 2^{-k} < \int_0^{\infty} x 2^{-x} dx = \frac{1}{\ln^2 2} = 2.08. \tag{2}$$

Eq. (2) shows that the expected carry propagation (carry-chain) length is very small. It hints that the longest carry-chain that would be experienced will also be short. This was observed in the early days of digital computers and a formal proof that it is bounded by $O(\log_2 n)$ can be found in [2]. Fig. 1 illustrates the distribution of the longest carry-chain for adders of 16–256 bits. Those have been simulated with randomly drawn input arguments. It is clearly seen that the expected longest carry-chain is nearly $\log_2 n$, while the probability that its length will exceed $2\log_2 n$ is practically zero.

The above observations raise the question of whether it pays off to design adders to meet the worst-case of n-bit carry propagation. We could instead design adders for carry-chains of $O(\log_2 n)$ length. We subsequently show that the infrequent cases of excessive carry propagation can be handled without sacrificing addition correctness, with only a small performance penalty (average latency) at the system level. We can leverage the carry propagation relaxation in various ways. Adder-intensive applications such as DSP and image processors can significantly reduce their power and energy consumption. High-end microprocessors where an adder may be on a critical architectural delay path, can also benefit from the dual-mode approach, for example, the addition step within a Fused Multiply Add operation (FMA) in a floating-point unit. There, due to the very wide operands, several clock cycles are required for addition. Hence, adders designed for shorter carry propagation could accomplish FMA operation in fewer cycles.

4. Dual-mode adder architecture and circuits

The proposed adder is designed for $O(\log_2 n)$ carry-chains length, but must still handle properly longer carry-chains. To this end we need to detect whether or not the longest carry-chain occurring in an addition exceeds a certain limit of k bits.

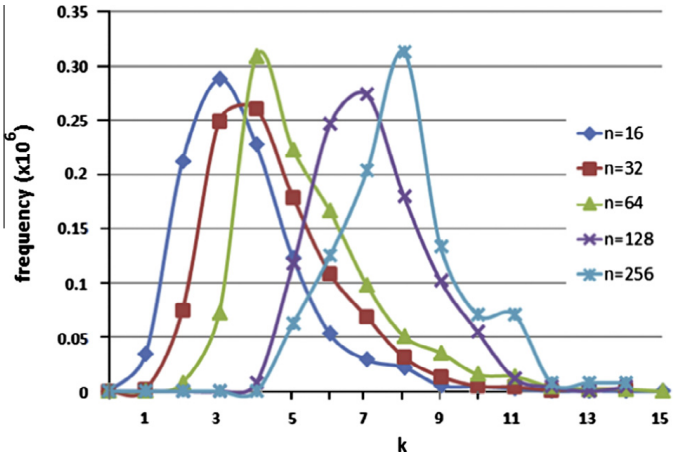


Fig. 1. Distribution of the longest carry-chain in addition.

If it does, the adder must modify its operation and allocate more time so that a correct result is produced. We subsequently address these issues in our proposed n-bit adder architecture, called *dual-mode adder*. It will target k-bit carry propagation in its most likely operation mode called *normal*, while the other mode where additional clock cycles are required to properly complete the addition, is called *extended*.

Let $n = 2^N$ and $n = mk$ where m and k are powers of 2. We divide the n bits into $m = 2^M$ groups of $k = 2^K$ bits each, so that $K = N - M$. The dual-mode adder targets the delay of a k-bit group rather than the worst-case of $n = mk$ bits, which ordinary adders do. In its most likely normal mode it will consume low power and energy. The extended mode will take place in those few cases where the carry propagates through more than k bits, where it will consume more time, power and energy.

A block diagram of a dual-mode adder embedded in a pipelined system is illustrated in Fig. 2. The addition SUM = A + B starts when the arguments are stored into the registers A and B. The detection of whether a normal mode will suffice for proper addition takes place simultaneously with the addition. If a normal node is validated (most likely), the clock gater producing the pipelined clock follows the global clock, and the sum will be loaded into the register after one cycle. If, however, an extended mode occurs (rarely), the pipelined clock signal is delayed by an appropriate number of (global clock) cycles that allows the adder to properly complete. The circuit details are subsequently elaborated and analyzed for Manchester carry-chain adder implementation [1–3], but the idea can be adapted to other adder types.

4.1. A k-bit group design

Consider the addition of two k-bit groups, and let $p_i = a_i \oplus b_i$ and $g_i = a_i \cdot b_i$, $0 \leqslant i \leqslant k-1$, be the propagate and generate signals, respectively. We define a group-propagate signal P_j asserting that the carry propagates from bit 0 to bit j, $0 \leqslant j \leqslant k-1$. The group-propagate signal is given by $P_j = \prod_{i=0}^{j} p_i = p_j P_{j-1}$, where $P_{-1} = 1$ by definition. Carry propagation and its assertion signal P_j can be implemented within a Manchester carry-chain adder. Fig. 3 illustrates one bit of the chain, comprising two tracks implemented by CMOS pass-gate switches [3].

A chain of k bits is shown in Fig. 4. The role of the upper track is to propagate P_j. When $p_j = 1$, the pass-gate transfers P_{j-1}. The role of the lower track is to pass the value of the carry. The group-generate signal is given by $G_j = p_j G_{j-1} + \bar{p}_j g_j$, where

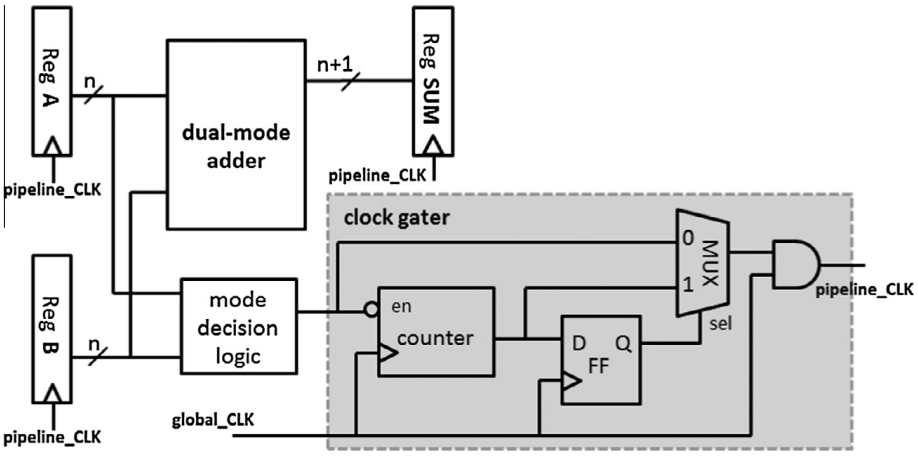


Fig. 2. Block diagram of a dual-mode adder embedded in a pipelined system.

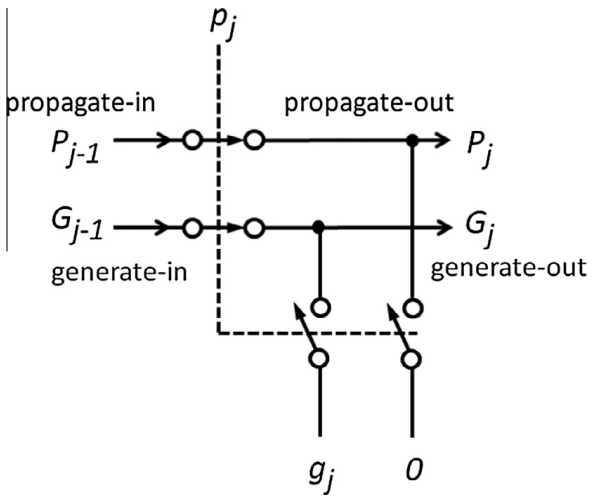


Fig. 3. A basic carry chain bit.

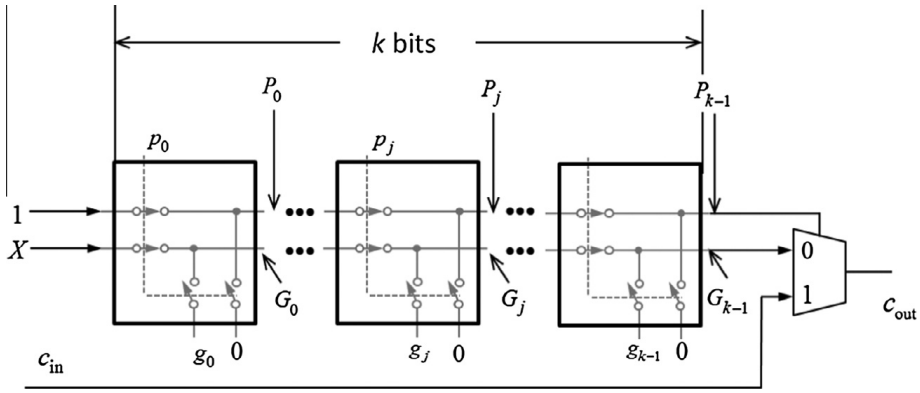


Fig. 4. A k-bit carry-chain.

$G_{-1} = X$ (do not care) by definition, as shown in Fig. 4. Note that $p_j = 1$ turns off the lower switches and both P_{j-1} and G_{j-1} inputs are transferred to the outputs. When $p_j = 0$, the propagate and generate tracks are disconnected, and new values are put on the tracks instead. A 0 value is written onto the upper track, thus setting $P_j = 0$, which in turn, enforces $P_i = 0$, $j \leqslant i \leqslant k - 1$. The carry value g_j (either generated or killed) is written onto the lower track and propagates through as long as the p-value of successive bits is 1. Once the p-value is 0, a new carry value is produced. The selection of c_{out} in the k-bit group is made with a 2:1 MUX controlled by P_{k-1}, similarly to a carry-skip adder.

We require the internal sums to be valid no later than the carry-out. Let. $P_{l-1} = 1$ and $P_l = 0$, $0 \leqslant l \leqslant k - 1$. If follows that from bit 0 to bit $l - 1$ the sum is determined by c_{in}, while from bit l to bit $k - 1$ it is determined by a carry generated or killed within the group. Fig. 5 illustrates the implementation of the sum computation.

4.2. Determining the adder's operation mode

Since n is divisible by k, as $n = mk$, the n-bit adder consists of m serially connected k-bit groups shown in Fig. 4. In its normal mode, each k-bit group either generates or kills the carry at some of its bit positions. To determine whether for specific operands the addition complies with the normal operation mode, the propagate signals of all m groups are ANDed and the results are NORed as illustrated in Fig. 6. To save hardware, the signal $P_{(j+1)k-1}$ produced at group j, $0 \leqslant j \leqslant m - 1$, can be used and NORed instead of the AND gates shown in Fig. 6.

If the *Normal_Mode* signal is asserted, it can be guaranteed that the addition properly completes in a single clock cycle. This requires appropriate circuit design of a group such that the clock cycle delay constraint is met. If, however, the *Normal_Mode* signal is de-asserted, it means that at some group the carry propagates through, and thus the carry delay exceeds the clock cycle. Addition can still complete properly if enough time is allotted by allowing the addition to use several extra clock cycles as shown in Fig. 2. In the worst case, the carry propagates through $n = mk$ bits. Since a single clock cycle suffices for k-bit carry propagation, allotting m cycles ensures proper addition computation. This is called *extended mode* and it can be handled by stopping the clock signal of the registers storing the adder's operands and output for $m - 1$ cycles.

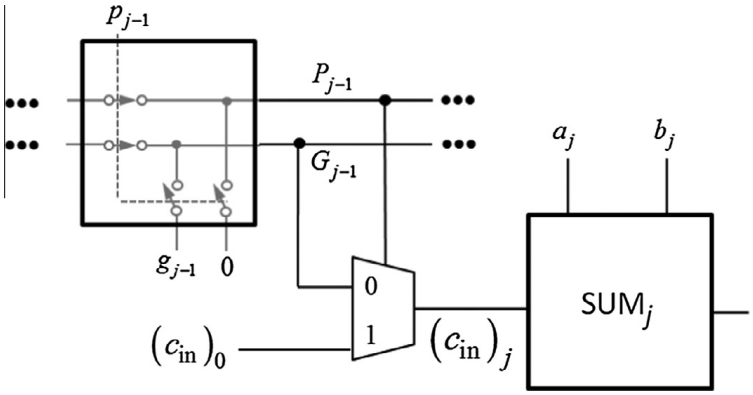


Fig. 5. Computation of the sum of bit j.

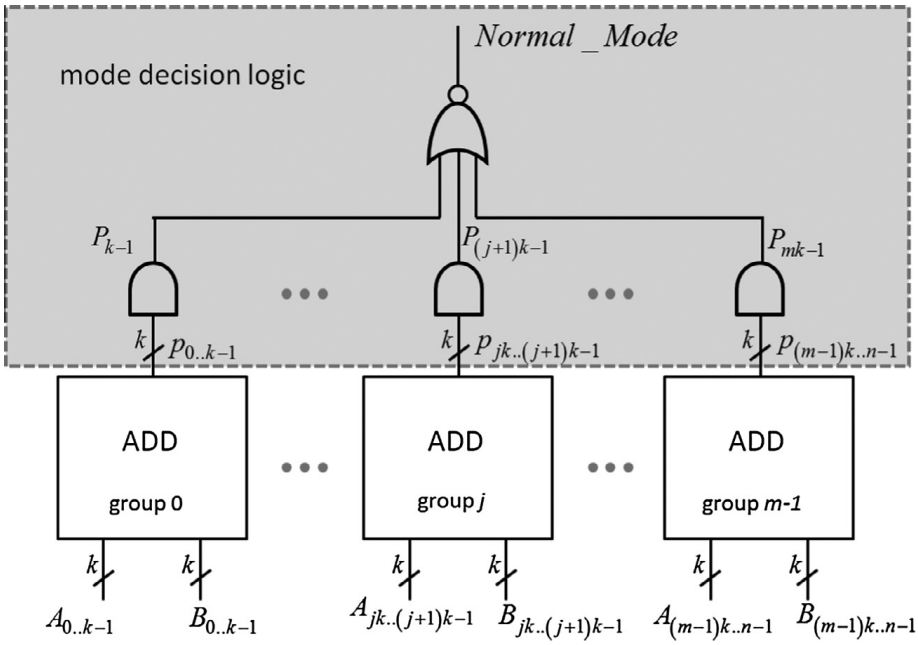


Fig. 6. Determining the adder's operation mode. Each k-bit group produces its propagate signal $P_{(j+1)k-1}$, $0 \leqslant j \leqslant m-1$.

The integration of a dual-mode adder into a design depends on the nature of the system. In a pipelined processor the system must be aware that an extended mode operation takes place, and an appropriate stall of the computation pipeline for $m-1$ cycles must take place. In DSP and image processors, a different treatment of the extended mode is required. Such a detailed implementation at the system level is beyond the scope of this paper.

The mode decision is made simultaneously with the addition. If at the end of the clock cycle the *Normal_Mode* is 1, the addition result is valid. If, however, *Normal_Mode* is 0, more time is required to complete the addition and the extended mode is followed. In the normal mode, the critical path delay of the carry in Fig. 4 comprises the k-bit chain and the MUX producing the group's carry-out that is used as carry-in of the successive group. The signals P_j and G_j in Fig. 4 have quadratic delay growth with k due to the pass-gate carry-chain implementation. To avoid the load impact on the successive groups, a buffered MUX is used. The delay of the mode prediction circuit is $O(\log_2 n) = O(\log_2 m) + O(\log_2 k)$ time units due to the fan-in limit of CMOS gates. The critical path delay in the normal mode, denoted by t_{norm}, is therefore

$$t_{\text{norm}} \propto \max\{\alpha \log_2 n, \gamma + \beta k^2\}, \tag{3}$$

where α, β and γ are technology and cell library dependent delay parameters. The parameter α is inversely proportional to the size of a device in the gate used for the mode detection logic in Fig. 6, γ relates to the size of the transistors in the MUX (see Fig. 4) and β relates to the size of a pass-gate transistor in the carry-chain, assuming that all pass-gates are similar.

5. Energy consumption

The advantage of the dual-mode adder stems from its shorter critical path delay. It is designed for an $f(k)$-bit critical path, rather than $f(n)$-bit as ordinary adders do. This enables power reduction by using weaker devices or voltage scaling as

subsequently discussed. The probability of a carry to propagate through a k-bit group is 2^{-k}. In the normal mode each of the m k-bit groups must internally kill the propagation of a carry or generate a new carry, and the corresponding probability q_{norm} is therefore

$$q_{\text{norm}}(k,m) = (1-2^{-k})^m = 1 - m2^{-k} + O(m^2 2^{-2k}) > 1 - m2^{-k}. \tag{4}$$

Eq. (4) provides a lower bound for the normal mode probability, and since the extended mode (requiring multi clock cycles) consumes more energy than the normal mode, this lower bound can safely be used for evaluating the energy savings in dual-mode addition. It follows from (4) that the probability q_{ext} of the extended mode satisfies

$$q_{\text{ext}}(k,m) < m2^{-k}. \tag{5}$$

Fig. 7 shows the probability that the adder will operate in the extended mode for various adder and group sizes. Adders of 16–256 bits have been simulated with randomly selected operands. It is clearly shown that for group size equal to or larger than $2\log_2 n$ this probability is practically zero.

Let P_{dyn} and P_{stat} denote the adder's dynamic and static (leakage) power consumption, respectively. The switching activity of the adder's internal nodes is determined by its inputs, regardless of how many clock cycles are allotted for the combinational circuit to perform the addition. Consequently, the component P_{dyn} of the total energy is independent of the adder's operating mode (normal or extended). The static power P_{stat} however, grows in the extended mode by the factor m since the addition now lasts m clock cycles. Let T be the clock period. It follows from the probability bounds in (4) and (5) that the expected energy E_{add} spent during an addition is

$$E_{\text{add}} = q_{\text{norm}}T(P_{\text{dyn}} + P_{\text{stat}}) + q_{\text{ext}}T(P_{\text{dyn}} + mP_{\text{stat}}) = T\{P_{\text{dyn}} + P_{\text{stat}}[1 + m(m-1)2^{-k}]\}. \tag{6}$$

Denoting $\delta(k, m) = m(m-1)2^{-k}$, we obtain

$$E_{\text{add}} = T\{P_{\text{dyn}} + P_{\text{stat}}[1 + \delta(k,m)]\}. \tag{7}$$

The expected time T_{add} required for an addition is

$$T_{\text{add}} = q_{\text{norm}}T + q_{\text{ext}}mT = T[1 + m(m-1)2^{-k}] = T[1 + \delta(k,m)]. \tag{8}$$

The benefit of a dual-mode adder stems from the short critical path delay occurring in its normal mode. To meet an aggressive clock cycle T, an ordinary adder designed for an n-bit worst-case carry propagation must pay in high drive strength devices, high voltage, dynamic circuits and complex architecture, all causing high power consumption. The dual-mode adder, in contrast, can use a simpler architecture for a group, static circuits, and transistors with low drive strength. This can significantly reduce the adder's area and power. Both expressions for the expected energy (7) and addition time (8) contain the factor $\delta(k, m)$, representing the overhead due to the extended mode. Table 1 shows the value of $\delta(k, m)$ for 64- and 128-bit adders and various group sizes.

6. Optimizing group size for energy minimization

There is a clear tradeoff between the group size k, the expected energy E_{add} and the expected addition time T_{add}. A smaller k shortens the critical path delay of a group, enabling energy reduction. A smaller k however, rapidly increases $\delta(k, m)$, a factor multiplying the static power, thus increasing the energy consumption. Eq. (8) shows that smaller groups increase the expected addition time. We subsequently derive the optimal k that minimizes the adder's energy for a given clock cycle T so that the delay t_{norm} of the normal mode addition given in (3) is satisfied. Given an architecture of an n-bit adder, the design degrees of freedom are its group size k, the driving strength and speed of its logic and its underlying transistors. We assume that the carry-chain circuits are of pass-gate logic as shown in Fig. 4, and all the transistors have the same size and threshold voltage, which is a common design practice.

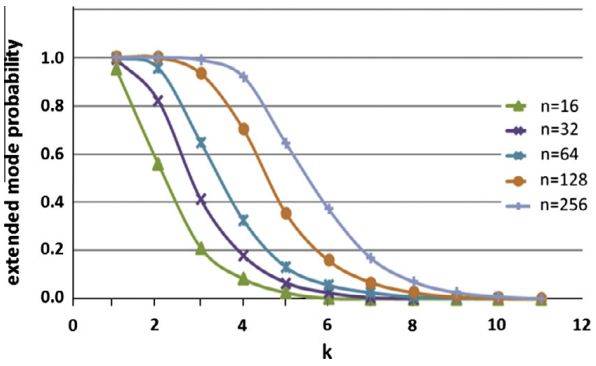


Fig. 7. The probability of the extended mode as a function of the group size.

Table 1
$\delta(k, m)$ for 64- and 128-bit adders.

	$k=2$	$k=4$	$k=8$	$k=16$	$k=32$	$k=64$
64-Bit	$m=32$	$m=16$	$m=8$	$m=4$	$m=2$	NA
	248	15	0.219	1.83×10^{-4}	0	NA
128-Bit	$m=64$	$m=32$	$m=16$	$m=8$	$m=4$	$m=2$
	1008	62	0.937	8.54×10^{-4}	0	0

For a correct addition, the output of the mode prediction circuit shown in Fig. 6 must stabilize within the clock cycle T. The mode prediction logic is independent of k since its delay is $\alpha\log n$, which by appropriate device sizing can always satisfy $\alpha\log n \leqslant T$. Finding the optimal group size thus involves the constraint

$$\gamma + \beta k^2 \leqslant T. \tag{9}$$

The parameter γ is the delay of the MUX in Fig. 4, which can also be considered as a given and thus is not a subject of the optimization. Denoting $T' = T - \gamma$, the problem is to find a group size k and transistors delay β, minimizing the energy consumed by the addition, while satisfying the delay constraint

$$\beta k^2 \leqslant T'. \tag{10}$$

Equality in (10) can always be assumed since for inequality the transistors could be further downsized, thus yielding lower energy.

Energy minimization calls for smaller transistors which have an increased delay β. This in turn will reduce k, which will increase $\delta(k, m)$ and thus the expected energy in (7) and the addition time in (8). The group size k and transistors delay β are therefore conflicting with each other and an optimal tradeoff is sought. To distinguish between the contributions of a transistor to the dynamic and static power components, we use a parameter λ denoting the amount of static (leakage) power consumption of a unit transistor width compared to its dynamic (capacitance) power. For today's process technologies this ratio falls in the range of $0.2 \leqslant \lambda \leqslant 0.4$ [17].

The power consumed by the adder, both dynamic and leakage, is proportional to the total size of the devices involved. Under the assumption that all the transistors have the same delay β and size, the size is proportional to $1/\beta$. The power P_{total} is therefore proportional to

$$P_{\text{total}} \propto \frac{1}{\beta}km = \frac{n}{\beta}. \tag{11}$$

Substituting $P \propto n/\beta$, $\delta(k, m) = m(m-1)2^{-k}$, $m = nk^{-1}$ and $T' = \beta k^2$ in (7) yields

$$E_{\text{add}} = T'\{P_{\text{dyn}} + P_{\text{stat}}[1 + \delta(k,m)]\} \propto (1+\lambda)nk^2 + \lambda n^3 2^{-k} - \lambda n^2 k 2^{-k}. \tag{12}$$

Fig. 8 shows how the energy consumption of a dual-mode adder depends on n and k for $\lambda = 0.3$, from which the group size k_{opt} minimizing the dual-mode adder energy can be derived by solving the equation $dE_{\text{add}}/dk = 0$ in (13).

$$\frac{dE_{\text{add}}}{dk} = 2(1+\lambda)nk - \lambda\ln 2 n^3 2^{-k} - \lambda n^2 2^{-k} + \lambda \ln 2 n^2 k 2^{-k}. \tag{13}$$

The blue[1] curve in Fig. 9 shows the group size yielding minimum energy for various adder sizes, obtained by solving (13). It is interestingly well matched with the expression $k = 1.6\log_2 n - 3.4$, concluding that the minimum energy of a dual-mode adder is achieved when its group size is close to the $O(\log_2 n)$ expected longest carry-chain, shown in red.

A comment is in order. The structure of a dual-mode adder is reminiscent of carry-select and carry-skip adders [1,2], where its groups correspond to the groups of those adders. It is different however, with respect to the optimization goal. While the conventional adders target minimum worst-case delay, the dual-mode adder targets minimum energy. In carry-select and carry-skip adders the optimal group size yielding minimum delay is $O(\sqrt{n})$, while in dual-mode the optimal group size yielding minimum energy is $O(\log_2 n)$.

6.1. Trading off energy and computation time

The above analysis yielded the group size minimizing the energy consumption of a dual-mode adder. The energy E_{add} in (12) has a clearly identifiable minimum as shown in Fig. 8. For a small group size, the probability of the extended-mode (multi clock cycles) is high, resulting in excessive static power dissipation. In contrast, for a large group size the probability of extended-mode is very low, but the adder consumes high dynamic power due to the underlying circuits having long critical carry paths, which requires strong drive of transistors. The expected addition time, expressed by T_{add} in (8), decreases

[1] For interpretation of color in Figs. 9, 13 and 14, the reader is referred to the web version of this article.

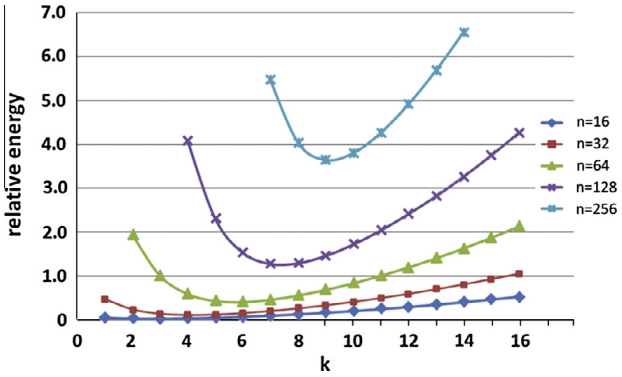


Fig. 8. The energy consumed by a dual-mode adder for $\lambda = 0.3$.

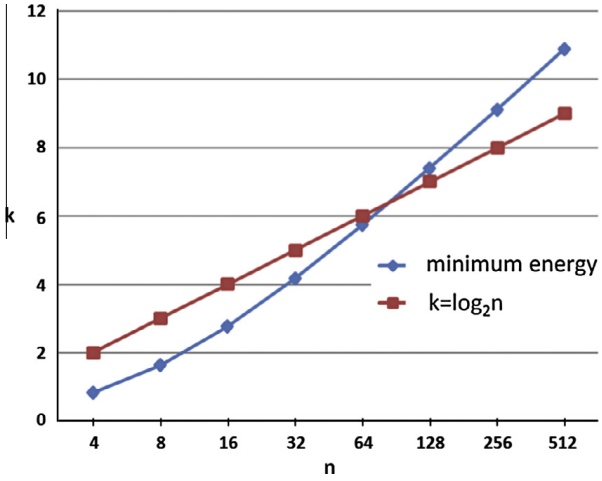


Fig. 9. The group size yielding minimum energy.

with an increased group size. Thus, the tradeoff between the energy consumption and the computation time is worth analyzing.

To this end, we can capture the energy and the computation time in a single objective by considering their product $E_{add} \times T_{add}$, which relates to the well-known AT^2 metric, used to measure computational complexity [18]. The similarity between $E_{add} \times T_{add}$ and AT^2 follows from the power that is usually proportional to the area A of a circuit, so AT can express its energy consumption. Fig. 10 plots $E_{add} \times T_{add}$ for various adder sizes. The curves were obtained for a static to dynamic power ratio $\lambda = 0.3$, as for Fig. 8.

To study the tradeoff between the desirable increase in the normal mode probability and the undesirable energy increase, we compare the group size minimizing the energy to that minimizing the energy-time product. Table 2 summarizes the results for various adder sizes. It shows that the probability of operating in the normal mode (single clock cycle) is significantly increased when the metric $E_{add} \times T_{add}$, rather than E_{add}, is minimized. This however, comes at the expense of 25–36% energy increase, denoted by ΔE_{add}, above the minimum energy.

One may also be interested in setting the group size such that the probability q_{norm} of an addition to properly complete within a single clock cycle is guaranteed to meet a certain value q. This may be important, for example, in a pipelined design where an increase in the addition time to several cycles may stall other operations. The corresponding group size can be obtained by setting $q_{norm} = q$ in (4) and solving for k_q. Substituting k_q in (12) yields the corresponding expected energy. The row $k_{0.99}$ in Table 2 shows the group sizes which guarantee $q_{norm} = 0.99$ for various adder sizes. The results show that the increase ΔE_{add} above the minimum energy is getting smaller for larger adder sizes. This is not surprising since the group size yielding minimum energy in wide adders corresponds to higher values of q_{norm}. For instance, while for a 64-bit adder the increase of q_{norm} from 0.83 to 0.99 costs ΔE_{add} of 67%, in a 256-bit adder the increase of q_{norm} from 0.95 to 0.99 requires a ΔE_{add} of only 17%.

7. Reducing the power of the entire system by voltage scaling

The energy consumed by adders in a typical microprocessor constitutes only a small portion of the entire energy. The situation however, is different in special processors such as DSP or image processor comprising hundreds of adders. It is

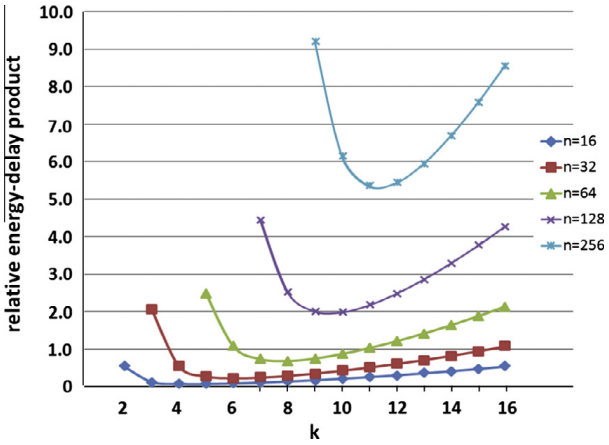


Fig. 10. $E_{add} \times T_{add}$ for various adder sizes.

Table 2
Trading off energy and expected addition time.

	Size	$n = 16$	$n = 32$	$n = 64$	$n = 128$	$n = 256$
Minimize E_{add}	k_{opt}	3	4	6	7	9
	q_{norm}	0.33	0.50	0.83	0.86	0.95
Minimize $E_{add} \times T_{add}$	k_{opt}	4	6	8	9	11
	q_{norm}	0.75	0.92	0.97	0.98	0.99
	ΔE_{add} (%)	25	34	36	27	28
	$k_{0.99}$	7	8	9	10	11
	ΔE_{add} (%)	228	123	67	34	17

subsequently shown that the incorporation of dual-mode adders in such processors can enable voltage scaling, yielding substantial overall energy reduction.

The dynamic power consumed by a system is proportional to

$$C_{sys} f V_{dd}^2, \tag{14}$$

where C_{sys} is the system's capacitive load, f is its clock frequency and V_{dd} is the power supply voltage. It is well-known that the delay t_{pd} of CMOS transistors depends on V_{dd} as follows

$$t_{pd} \propto \frac{C_{tr} V_{dd}}{I_{DS}} \simeq \frac{C_{tr} V_{dd}}{A(V_{dd} - V_{th})^2}, \tag{15}$$

where C_{tr} is the MOS transistor's capacitive load, I_{DS} is the drain current in saturation, V_{th} is its threshold voltage and A is a constant [3]. We subsequently show that the supply voltage V_{dd} can be reduced without degrading the clock speed, which in turn reduces the power of the entire system, provided that the processor's critical path delay is determined by the adders and no other path becomes critical due to the V_{dd} reduction.

It is shown in Fig. 9 that the minimum energy of an n-bit dual-mode adder is achieved when the size of its group is nearly $k = 1.6 \log_2 n - 3.4$. Consider an ordinary adder designed for $O(\sqrt{n})$ carry propagation delay (e.g., carry-skip or carry-select architecture). Since the clock cycle is set to meet the $O(\sqrt{n})$ delay requirement, the usage of a dual-mode adder could reduce V_{dd} by some factor ϕ to meet the $k = 1.6 \log_2 n - 3.4$ delay requirement of its optimal group size, without slowing down the clock speed. Equating the worst-case propagation delays $t_{pd}^{\text{dual-mode}}$ and t_{pd}^{ordinary} of the two adders, and using (15), we obtain

$$t_{pd}^{\text{dual-mode}} = (1.6 \log_2 n - 3.4) \frac{C_{tr} \phi V_{dd}}{A(\phi V_{dd} - V_{th})^2} = \sqrt{n} \frac{C_{tr} V_{dd}}{A(V_{dd} - V_{th})^2} = t_{pd}^{\text{ordinary}}. \tag{16}$$

Substituting $V_{th} = \eta V_{dd}$ in (16), where η depends on the process technology, turns it into the following quadratic equation in ϕ

$$(\phi - \eta)^2 \sqrt{n} - \phi(1 - \eta)^2 (1.6 \log_2 n - 3.4) = 0. \tag{17}$$

Fig. 11 shows the solution of (17) for various adder sizes and threshold voltages. Point (a) for instance, shows that for a 64-bit adder and a technology where $V_{th} = 0.4 V_{dd}$, using a minimal energy dual-mode adder allows a voltage scaling factor of 0.834, which based on (14), translates to $1–0.834^2 = 0.304$ (30%) potential power reduction. Similarly, point (b) represents a potential of $1–0.677^2 = 0.542$ (54%) power reduction of the entire system. It is important to note that such power reduction by voltage scaling can be achieved provided that no other paths in the system become critical.

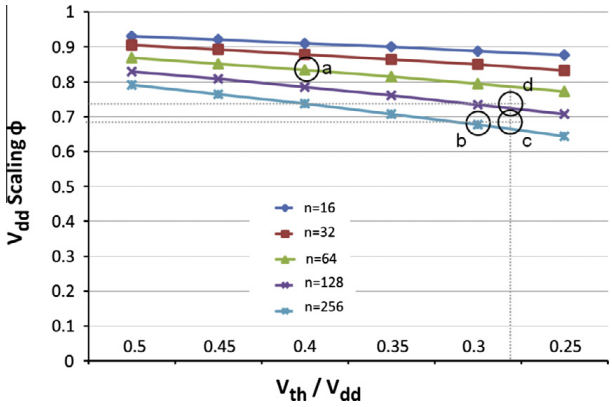


Fig. 11. Potential voltage scaling enabled by dual-mode ALU.

8. Experimental results and comparisons

This section shows first that the experimental power reduction obtained by voltage scaling of a dual-mode adder nicely matches the analysis in Section 7. The power reduction is then demonstrated with an experiment on the design of an industrial image processor [19]. We then compare the power reduction to what can be obtained by a variable latency adder [15].

Fig. 12 illustrates the results of a SPICE simulation obtained for a 256-bit adder implemented in 180 nm process technology. The SPICE setup is such that the two addition operands A and B are complementary of each other, yielding the worst-case carry propagation. The red wave form is the carry-in signal, switching from 0 to 1 and back to 0. The outcome is the carry out, showing the worst-case delay. The nominal power supply voltage is 1.8 V, where the threshold voltage is 0.5 V. Their ratio is therefore $\eta = V_{th}/V_{dd} = 0.28$. Case (a) shows the delay in a carry-skip adder implementation with a group size of $\sqrt{256} = 16$ bits. The propagation delay measured for that group when the adder was operated in nominal power supply voltage was 7.5 nS, dictating 133 MHz clock frequency. A dual-mode adder was then designed with a group size minimizing the energy according to what is shown in Fig. 9, dictating a group size of $1.6\log_2 256 - 3.6 \simeq 8$ bits (to maintain design uniformity 8-bit groups were used rather than 9-bit). This enabled to scale the voltage down to 1.25 V while satisfying the timing constraints, as shown in case (b). The ratio $\phi = 1.25/1.8 = 0.69$ shows a power reduction of more than 50%. The intersection of the lines $\eta = 0.28$ and $\phi = 0.69$ is shown in point (c) of Fig. 11 and is very close to that obtained by the analysis in Eq. (17). A similar experiment of a 128-bit adder was performed. It yielded a voltage scaling $\phi = 1.32/1.8 = 0.74$, shown in point (d) of Fig. 11, which is also well match the analysis. The 0.74 voltage scaling is translated to nearly 40% power reduction.

To show the applicability of voltage scaling for dual-mode addition, the image processor in [19] was experimented. It is used for digital still cameras to process the pixels obtained from an image sensor. It is implemented in 40 nm process technology. Processing include color filtering, noise elimination, image enhancement, color space conversion and gamma correction. The operations are performed by carry save trees, where the final carry propagate adder (CPA) is subsequently shown to be the timing critical path. The processor contains about three hundred CPAs whose width varies from 10 to 21 bits, depending on the number of neighboring pixels used by the filtering operation.

The processor has been simulated with an extensive benchmark of 36 k clock cycles. Fig. 13 plots the average longest carry-chain of each adder. As can be observed, the average longest carry-chain scatters around the minimum energy group size derived analytically in Section 6 by the expression $1.6\log_2 n - 3.4$, shown by the red line.

To find out whether the design can tolerate and benefit of voltage scaling without changing its clock cycle, static timing analysis was run first. The delays of the critical paths (dictating the clock cycle) are shown in Fig. 14, where their portion due

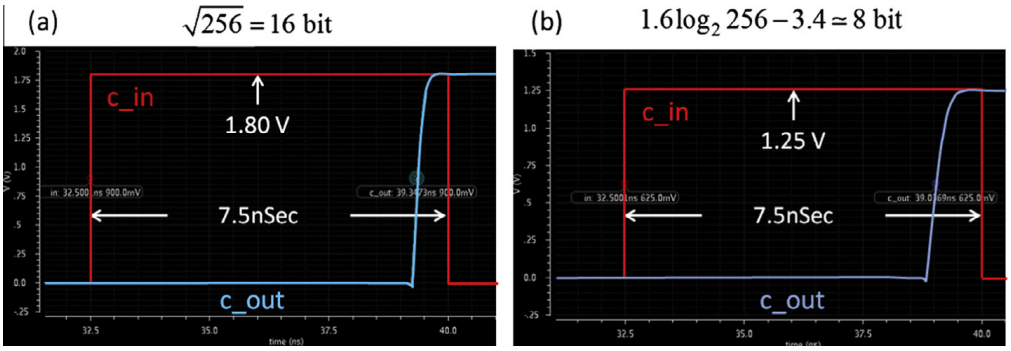


Fig. 12. Voltage scaling of a 256 bit adder.

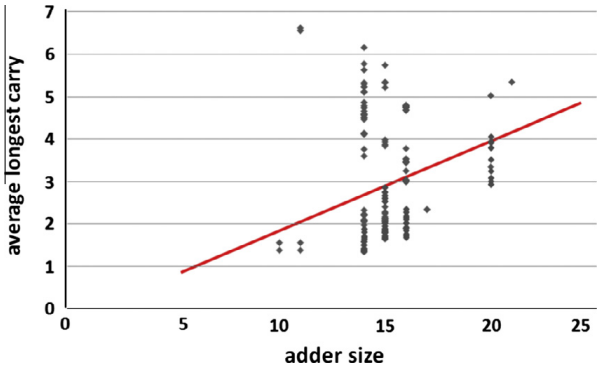


Fig. 13. Average longest carry-chain of adders in an image processor, scattered around the minimum energy group size line.

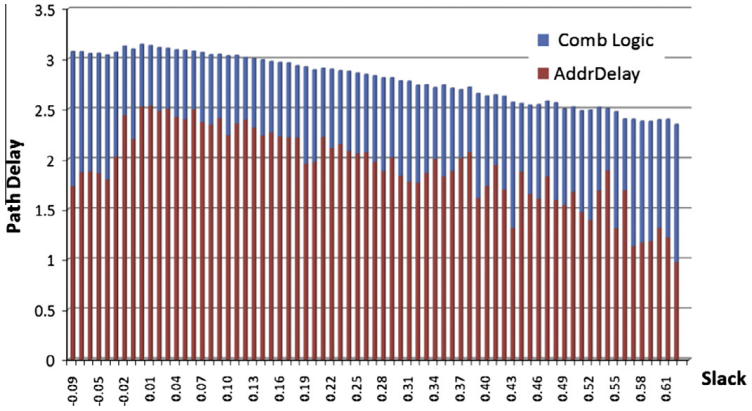


Fig. 14. The impact of the adders on delay paths criticality.

to carry paths passing through the CPAs is colored in red. As shown, most of the delay is contributed by the CPAs. Reducing their delay will therefore enable the voltage scaling as done in the first experiment.

The tradeoff between image quality and the energy reduction of the adders can be fully controlled by the analysis in Section 5. If perfect images are required, the system must be aware of the various addition modes and employ appropriate control flow to support the single-cycle and multi-cycles addition modes, similar to the pipelined processor shown in Fig. 2, with the mode decision logic in Fig. 6. Detailed discussion of this is beyond the scope of this paper. For an image processor the option of reducing the overall power (through voltage scaling) at the expense of a small precision loss is more attractive, since the processor's architecture needs not any changes and design simplicity is maintained.

To obtain the power reduction, group sizes of 3–6 were used. The size of the groups for adder width n were determined so as to achieve a low error probability and have a small number of different group sizes for cases where n is not divisible by the desired group size. Table 3 summarizes the power reduction and the corresponding error of pixel calculations. As expected, the larger the adder is, the more power reduction is possible (see next experiment).

The power consumed by the dual-mode architecture was compared to that of the variable-latency carry-select adder in [15]. Though the shorter latency of the dual-mode adder could be used to downsize its underlying transistors, we preferred

Table 3
Trading off power reduction and errors in pixel calculations.

Adder size	$n = 10$	$n = 11$	$n = 13$	$n = 14$	$n = 15$	$n = 17$	$n = 20$	$n = 21$
Group size	3–4	3–4	4–5	4–5	5	5–6	5	5–6
Power reduction	0.86	0.86	0.84	0.83	0.81	0.81	0.79	0.78
Error (%)	4	4	3	3	2	2	2	2

Table 4
The power ratio of the dual-mode adder to that of a variable-latency carry-select adder [15].

	Adder size	$n = 16$	$n = 32$	$n = 64$	$n = 128$	$n = 256$
V_{th}/V_{dd}	0.3	0.79	0.72	0.63	0.54	0.46
	0.4	0.83	0.77	0.69	0.62	0.55
	0.5	0.87	0.82	0.76	0.69	0.63

to save energy by voltage scaling since other parts of a system that are not on the critical paths will also benefit of it. Table 4 shows the power ratio of the two adders. Due to the shorter latency of our architecture, we used voltage scaling to stretch the clock cycle to equal that used by the variable-latency carry-select adder. The power ratio ranges from 0.46 (highest improvement) to 0.87 (smallest improvement). The improvement increases with an increase in the adder size and a decrease in V_{th}/V_{dd}.

9. Conclusions and further research

This paper has presented a novel adder design that targets the expected longest carry propagation rather than the worst-case. It can yield substantial power and energy savings. We described its architecture and showed an example of a circuit implementation of the proposed adder. It was shown that a dual-mode adder enables voltage scaling in DSP and image processors that has the potential of up to 50% energy savings. Taking advantage of dual-mode adder at system-level is a matter of further research. It can be considered to replace the existing adders in an ordinary microprocessor's pipeline, and in special-purpose processors. The idea of optimizing computational systems for the expected cases rather than for worst-cases should be further explored. It may change the traditional design approach towards better energy-performance tradeoffs.

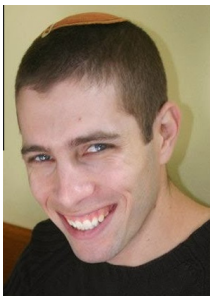
Acknowledgements

This research was partially funded by the Israel Science Foundation (ISF) under Grant Number 1678/13 and by the MAGNET Program of Israel Ministry of Industry (Alpha consortium). The authors wish to thank Mickey Geftler and Ophir Turbovich of CSR (formerly Zoran) for helpful discussions.

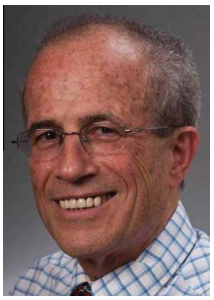
References

[1] Koren I. Computer arithmetic algorithms. A.K. Peters; 2002.
[2] Parhami B. Computer arithmetic: algorithms and hardware designs. Oxford University Press; 2009.
[3] Weste N, Harris D. CMOS VLSI design: a circuits and systems perspective. Pearson; 2005.
[4] Montalvo LA, Parhi KK, Janardhan H. Estimation of average energy consumption of ripple-carry adder based on average length carry chains. In: VLSI signal processing workshop IX. p. 189–98.
[5] Freking RA, Parhi KK. Theoretical estimation of power consumption in binary adders. In: Proc. of the 1998 IEEE international symposium on circuits and systems – ISCAS 98, vol. 2. p. 453–7.
[6] Sproull RF, Sutherland IE, Molnar CE. The counterflow pipeline processor architecture. IEEE Des Test of Comput 1994;11(3):48–59.
[7] Van Berkel Kees, Burgess Ronan, Kessels Joep, Roncken Marly, Schalij Frits, Peeters Ad. Asynchronous circuits for low power: a DCC error corrector. IEEE Des Test Comput 1994;11(2):22–32.
[8] Martin AJ. Asynchronous datapaths and the design of an asynchronous adder. Form Meth Syst Des 1992;1(1):119–37.
[9] Dean ME, Dill DL, Horowitz M. Self-timed logic using current-sensing completion detection (CSCD). In: Proceedings of ICCD; October 1991. p. 187–91.
[10] Nowick SM. Design of a low-latency asynchronous adder using speculative completion. IEE Proc Comput Digit Tech Sept. 1996;143(5):301–7.
[11] Nowick SM, Yum KY, Beerel PA, Dooply AE. Speculative completion for the design of high-performance asynchronous dynamic adders. In: 3rd International symposium on advanced research in asynchronous circuits and systems; 1997. p. 210–23.
[12] Gupta Vaibhav, Mohapatra Debabrata, Raghunathan Anand, Roy Kaushik. Low-power digital signal processing using approximate adders. IEEE Trans Comput-Aid Des Integr Circ Syst 2013;32(1):124–37.
[13] Kulkarni P, Gupta P, Ercegovac M. Trading accuracy for power with an underdesigned multiplier architecture. In: IEEE 24th International conference on VLSI design; 2011. p. 346–51.
[14] Dan Ernst, Blaauw David, Austin Todd, Das Shidhartha, Lee Seokwoo, Mudge Trevor, et al. Razor: circuit-level correction of timing errors for low-power operation. IEEE Micro 2004;24(6):10–20.
[15] Chen Yiran, Li Hai, Koh Cheng-Kok, Sun Guangyu, Li Jing, Xie Yuan, et al. Variable-latency adder (VL-adder) design for low power and NTBI tolerance. IEEE Trans VLSI Syst 2010;18(11):1621–4.
[16] Benini L, Macii E, Poncino M, De Micheli G. Telescopic units: a new paradigm for performance optimization of VLSI designs. IEEE Trans Comput-Aid Des Integr Circ Syst 1998;17(3):220–32.
[17] International Technology Roadmap for Semiconductors. Design Chapter; 2011 Edition. p. 45–6. <http://www.itrs.net/Links/2011ITRS/2011Chapters/2011Design.pdf>.
[18] Ullman JD. Computational aspects of VLSI, vol. 11. Rockville, MD: Computer Science Press; 1984.
[19] Zoran COACH 12-13™ Digital Camera Processor. <http://www.csr.com/products/128/coach-12-13>.
[20] Fu-Chiung c, Unger SH, Theobald M. Self-timed carry-lookahead adders. IEEE Trans Comput 2000;49(7):659–72.

Shmuel Wimer is an Associate Professor with the Engineering Faculty of Bar-Ilan University, and with the Electrical Engineering Faculty of the Technion. He received M.Sc. in mathematics from Tel-Aviv University, and D.Sc. in EE from the Technion. Prior to joining the academia he worked for 32 years at the industry for Intel, IBM, National Semiconductors and the Israeli Aerospace Industry.



Amir Albeck received his B.Sc. degree in EE from Bar-Ilan University on 2012, and he is currently pursuing his M.Sc. in Computer Engineering. Since 2011 he is working at Orbotech. He is interested in VLSI design optimization.



Israel Koren is a Professor at the University of Massachusetts, Amherst and a Fellow of IEEE. He has been a consultant to numerous companies including IBM, Analog Devices, Intel and AMD. His interests include Fault-Tolerant systems, Computer Architecture, Secure Cryptographic system and Computer Arithmetic. He has over 250 technical publications and two textbooks: ''Computer Arithmetic Algorithms,'' and ''Fault Tolerant Systems.''