A power-delay efficient hybrid carry-lookahead/carry-select based redundant binary to two’s complement converter

He, Yajuan; Chang, Chip Hong

2008

He, Y., & Chang, C. H. (2008). A power-delay efficient hybrid carry-lookahead/carry-select based redundant binary to two’s complement converter. IEEE Transactions on Circuits and Systems Part 1 Regular Papers. 55(1), 336-346.

https://hdl.handle.net/10356/93571

https://doi.org/10.1109/TCSI.2007.913610

A Power-Delay Efficient Hybrid Carry-Lookahead/Carry-Select Based Redundant Binary to Two's Complement Converter

Yajuan He, *Student Member, IEEE*, and Chip-Hong Chang, *Senior Member, IEEE*

***Abstract*—This paper presents an efficient reverse converter for transforming the redundant binary (RB) representation into two's complement form. The hierarchical expansion of the carry equation for the reverse conversion algorithm creates a regular multilevel structure, from which a high-speed hybrid carry-lookahead/carry-select (CLA/CSL) architecture is proposed to fully exploit the redundancy of RB encoding for VLSI efficient implementation. The optimally designed CSL sections interleaved evenly in the mixed-radix CLA network to boost the performance of the reverse converter well above those designed based on a homogeneous type of carry propagation adder. The logical effort characterization captures the effect of circuit's fan-in, fan-out and transistor sizing on performance, and the evaluation shows that our proposed architecture leads to the fastest design. A 64-bit transistor-level circuit implementation of our proposed reverse converter and that of its most competitive contender were simulated to validate the logical effort delay model. The pre- and post-layout HSPICE simulation results reveal that our new converter expends at least two times less energy (power–delay product) than the competitor circuit and is capable of completing a 64-bit conversion in 829 ps and dissipates merely 5.84 mW at a data rate of 1 GHz and a supply voltage of 1.8 V in TSMC 0.18-μm CMOS technology.**

***Index Terms*—Carry-lookahead adder (CLA), carry-select adder (CSL), redundant binary (RB) multiplier, reverse converter, ripple-carry adder.**

I. Introduction

DIGITAL multipliers are an indispensable component in general-purpose microprocessors, digital signal processors (DSPs), and multimedia application accelerators. Over the last two decades, we have witnessed an apparent paradigm shift and a surge of interest to explore alternative number representations for digital multiplier design [1]–[3], among which the redundant binary (RB) number [4], [5] has emerged as a key internal format to speed up the partial product accumulation of fast tree-structured parallel multipliers. The carry-free addition allows the partial products to be reduced at a rate of 2:1 using the RB adders as opposed to the 3:2 reduction rate with a full adder in the normal binary (NB) multiplier. In addition, the regular structure of the RB summing tree also makes RB multipliers amendable to area-efficient VLSI layout.

Manuscript received February 24, 2005; revised February 7, 2006. This paper was recommended by Associate Editor S.-G. Chen.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail: heyajuan@pmail. ntu. edu.sg; echchang@ntu.edu.sg).

Digital Object Identifier 10.1109/TCSI.2007.913610

As the accustomed bus architectures of DSPs and standard peripheral devices are still based on the two's complement number representation, an additional reverse conversion step is required in the final stage of the RB multiplier to convert the summation result in RB form to two's complement number. Unfortunately, this stage appears to be the major performance bottleneck in the entire RB multiplier architecture [4]. It has been proven that absolutely carry-free RB multiplier does not exist, and any such proposal ever claimed with superior performance has been invalidated [6]–[9].

Traditionally, the reverse converter can be implemented in a straightforward way by a chain of serially connected full adders [10]. Therefore, the fast RB-to-NB conversion problem can be traced back to the origin of fast carry propagation adder (CPA) logic. A fast converter based on carry-lookahead (CLA) mode was proposed in [10]. To simplify the carry generation logic, a new variable was defined to detect and signal carry propagation. In [11], a specialized carry propagation circuit is implemented with serial transmission gates (TGs) to gain speed. However, numerous inverters must be inserted as buffers to avoid performance degradation due to decaying drivability of cascaded TGs. A grouped carry-select method was proposed in [4] in which the carry generation circuit was implemented with carry-select adders and grouped in such a way that the number of digits in the group increases by one progressively.

Among the two operand parallel adders, hybrid carry-lookahead and carry-select adder is widely known as the most efficient adder and has been employed for the design of various fast adders in the two's complement regime [12]–[14]. A general architecture for its design has also been proposed by Wang *et al.* [15]. Motivated by the lack of dedication of such a fast addition technique to the bottleneck operation for the advancement of RB arithmetic, this paper brings together new circuit design strategies and insights to the RB-to-NB converter.

The remainder of this paper is organized as follows. A brief introduction to the RB multiplier and the inter-conversions between RB number and two's complement number representations are given in Section II. Section III describes our proposed hybrid carry-lookahead/carry-select (CLA/CSL)-based RB-to-NB converter architecture and variants of circuit topologies for the parallel-prefix carry generation with uniform and nonuniform block factors. An optimal implementation of a 64-bit reverse converter with the novel CSL circuit is detailed in Section IV to elaborate the design concept. The performance evaluation of the proposed converter and previous work using a logical effort method are presented in Section V, along with the

prelayout HSPICE simulation results of two competitive 64-bit reverse converters implemented in 0.18-μm CMOS technology. The post-layout simulation results of our proposed converter are also reported. We conclude this paper in Section VI.

II. Preliminaries

A RB number is a subset of a more generalized set of numbers known as signed digit number representation. It consists of digits from the set $\{-1, 0, 1\}$. In RB representation, the decimal value of an n-digit RB number $F = (f_{n-1} f_{n-2} \dots f_1 f_0)$, where $f_i \in \{-1, 0, 1\}$, is given by

$$F = \sum_{i=0}^{n-1} f_i \times 2^i. \tag{1}$$

To implement RB arithmetic with standard logic elements, the RB number needs to be encoded into normal binary bit stream. A commonly used binary coding, which is adopted in this study is based on binary subtraction of two binary bits f_i^+ and f_i^- as

$$f_i = f_i^+ + f_i^-. \tag{2}$$

Analogously to the signed multiplication in two's complement domain, an RB multiplier can be divided into three major building blocks, namely: 1) the Booth encoder and partial product generator; 2) the tree-structured array of RB adders (RBAs); and 3) the RB-to-NB reverse converter. Here the input operands and outputs are assumed to be in two's complement form. Therefore, the inter-conversions between the two's complement number and the RB number are overheads associated with the compatibility in data transfer through standard peripheral interfaces. The decimal value of an n-bit two's complement binary number $Z = (z_{n-1} z_{n-2} \dots z_1 z_0)$ is given by

$$Z = -z_{n-1} \times 2^{n-1} + \sum_{i=0}^{n-2} z_i \times 2^i. \tag{3}$$

From (1) and (3), it is observed that the forward conversion from an n-bit two's complement number representation into its RB number representation involves only the change of the most significant bit (MSB). Thus, the time required by this conversion is independent of the operand length. Therefore, the main overhead of the RB multiplication process lies in the conversion of the final partial product summation result from the RB form back to its two's complement representation. Since each RB digit can be decomposed into two binary bits, conventionally, the reverse conversion algorithm has been derived from the basic RB number definition

$$Z = F^+ - F^- \tag{4}$$

where Z represents the final result and F^+ and F^- are the two binary numbers in two's complement form decomposed from the RB number F as

$$F^+ = \sum_{\forall f_i \in F, f_i = 1} f_i \times 2^i$$
$$F^- = \sum_{\forall f_i \in F, f_i = -1} (-f_i) \times 2^i.$$

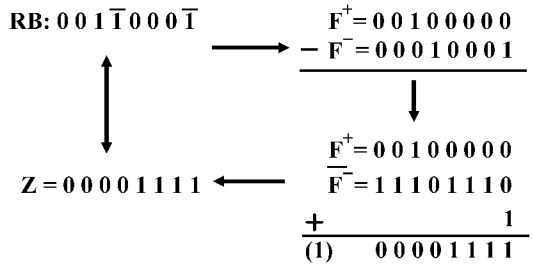


Fig. 1. Example of a reverse conversion from $001\bar{1}000\bar{1}$ to 00001111.

A two's complement subtraction is what it takes to obtain the result of (4) as shown in

$$Z = F^+ - F^- = F^+ + \overline{F^-} + 1 \tag{5}$$

Fig. 1 shows an example of a conventional RB-to-NB conversion method.

This implies that the reverse converter can be implemented in a straightforward way by a chain of serially connected full adders. It should be noted that the carry out of the most significant digit (MSD) is equal to the MSB of F^-. For the correct result, this carry out shall be ignored instead of being generated from the addition itself. This differentiates between the reverse converter and CPA as the sign bit of the converted two's complement number is located at the MSD position of the RB number. Let (f_i^+, f_i^-) denote a digit of the final RB partial product (F^+, F^-) and c_{i-1} be the carry-in from the next lower order digit, and then the sum output of the partial products z_i and the carry-out signal c_i can be derived from (5) as follows:

$$\begin{cases} z_i = f_i^+ \oplus \overline{f_i^-} \oplus c_{i-1} \\ c_i = f_i^+ \cdot \overline{f_i^-} + f_i^+ \cdot c_{i-1} + \overline{f_i^-} \cdot c_{i-1}, \quad c_{-1} = 1 \end{cases} \tag{6}$$

where $i = 0, 1, \dots, N-1$, and N is the number of digits of the final RB partial product to be converted to NB number. c_{-1} represents the first carry-in to the least significant digit (LSD) of the RB number.

III. Exploration of Hybrid CLA/CSL Architecture for RB-to-NB Conversion

Many adder types in the NB regime exist, and each has its own advantages and disadvantages. By far, the most comprehensive study from the very large-scale integration (VLSI) perspective of parallel adders has been provided in [16]. Among the two operand parallel adders, the carry-lookahead adder (CLA), with ELM [17] and B&K [18] adders being special variants, is widely known as the fastest adder with huge hardware cost, while the ripple-carry adder (RCA) consumes the least chip area but has the longest delay time, and the carry-select adder (CSL) is intermediate in performance between speed and area. To speed up CSL computation, CLA architecture is used to generate the select signals of CSL, leading to the hybrid CLA/CSL adder [12]–[15]. In this section, we will further explore the parallel and unidirectional generation of carry-select signals by leveraging the dedicated RB number encoding for the RB-to-NB reverse conversion algorithm.

A. Hybrid CLA/CSL-Based Reverse Conversion Algorithm

In a hybrid CLA/CSL circuit, the selected carry signals and sum bits are generated simultaneously by the cooperative execution of CLA and CSL networks. The selected carries are generated by a CLA tree without back propagation. The number of carry outputs to be generated is significantly reduced with regular interleaves of CSL sections. The sum bits are computed in sections by the CSL. Conventionally, each section of a CSL is implemented with dual RCA blocks with the constant carry-in of 0 and 1, respectively. Due to the anticipatory parallel computation, once the carry signal for the local section is generated by the CLA network, the corresponding carry-select adders will choose the correct sum and produce the output directly.

We make use of the fact that the binary pair representing each RB digit can never become "1" simultaneously. This is because (1, 1) has been converted to (0, 0) before the RBA tree stage to eliminate the inconsistent representations of "0" in order to simplify the design of RBA as described in [4]. The inherent redundancy of the RB coding format gives rise to the following simplifications for the generation of carry-generate bits g_i, carry-propagate bits p_i, and half-sum bits d_i:

$$\begin{cases} g_i = f_i^+ \cdot \overline{f_i^-} = f_i^+ \\ p_i = f_i^+ + \overline{f_i^-} = \overline{f_i^-} \\ d_i = f_i^+ \oplus f_i^- = \overline{f_i^+ + f_i^-}. \end{cases} \tag{7}$$

With (7), (6) can be simplified as follows:

$$\begin{cases} z_i = d_i \oplus c_{i-1} = \left(\overline{f_i^+ + f_i^-}\right) \oplus c_{i-1} \\ c_i = g_i + p_i \cdot c_{i-1} = f_i^+ + \overline{f_i^-} \cdot c_{i-1}, \quad c_{-1} = 1. \end{cases} \tag{8}$$

By unrolling the c_i recursion, we have

$$\begin{aligned} c_0 &= f_0^+ + \overline{f_0^-} \cdot c_{-1} = f_0^- \\ c_1 &= f_1^+ + \overline{f_1^-} \cdot c_0 = f_1^+ + \overline{f_1^-} \cdot \overline{f_0^-} \\ &\vdots \\ c_i &= f_i^+ + \overline{f_i^-} \cdot f_{i-1}^+ + \overline{f_i^-} \cdot \overline{f_{i-1}^-} \cdot f_{i-2}^+ + \cdots \\ &\quad + \overline{f_i^-} \cdot \overline{f_{i-1}^-} \cdots \overline{f_2^-} \cdot \overline{f_1^-} \cdot \overline{f_0^-} \\ &= \overline{f_i^-} \cdot \Big\{ f_i^+ + f_{i-1}^+ + \overline{f_{i-1}^-} \cdot f_{i-2}^+ + \ldots + \\ &\qquad \overline{f_{i-1}^-} \cdot \overline{f_{i-2}^-} \cdots \overline{f_{i-r}^-} \cdot f_{i-r-1}^+ + \ldots \\ &\qquad + \overline{f_{i-1}^-} \cdot \overline{f_{i-2}^-} \cdots \overline{f_1^-} \cdot \overline{f_0^-} \Big\} \\ &= \overline{f_i^-} \cdot H_i \end{aligned} \tag{9}$$

where $H_i = f_i^+ + f_{i-1}^+ + \bigvee_{j=1}^{i-2} \left(\bigwedge_{r=j+1}^{i-1} \overline{f_r^-} \right) \cdot f_j^+ + \bigwedge_{r=0}^{i-1} \overline{f_r^-}$ and the operators $\vee$ and $\wedge$ denote Boolean sum and product, respectively. Based on (9), H_i can be expanded by iteratively expanding the recursion as follows:

$$\begin{aligned} H_i &= G_k + P_k H_{i-1} = G_k + P_k G_{k-1} + P_k P_{k-1} H_{i-2} \\ &= G_k + P_k \cdot G_{k-1} + P_k \cdot P_{k-1} \cdot G_{k-2} + P_k \cdot P_{k-1} \cdots \\ &\quad P_{k-r} G_{k-r-1} + \cdots + P_k \cdot P_{k-1} \cdots P_2 \cdot G_1 \end{aligned} \tag{10}$$

where $P_k = \overline{f_{i-1}^-} \cdot \overline{f_{i-2}^-} \cdot \overline{f_{i-3}^-}$, $G_k = f_i^+ + f_{i-1}^+ + \overline{f_{i-1}^-} \cdot f_{i-2}^+$, and $G_1 = f_2^+ + f_1^+ + \overline{f_1^-} \cdot \overline{f_0^-}$.

For ease of exposition, a block factor of three is assumed in the above derivation, i.e., P_k and G_k are Boolean product and sum of three terms, respectively. The decomposition of H_{i-1} in the derivation of (10) shows that only some instead of all of the carries need to be generated. The integer index i signifies the bit position of the selected carry signal to be generated from the carry-lookahead unit. The number of carry generation units is dependent on the operand length. The integer index k is used to uniquely identify each carry generation unit. If N is the length of the RB operand to be converted to two's complement number, then the range of all integer values of k and the positions of all the carry signals i can be determined as follows:

$$1 \leqslant k \leqslant \lfloor N/3 \rfloor, \quad i = 3 \cdot k - 1 \tag{11}$$

where $\lfloor a \rfloor$ denotes the largest integer value not exceeding a.

More levels of decomposition are also possible following the similar approach of derivation according to the operand length N. H_i, and hence c_i, can be generated in a hierarchy of homogeneous carry-lookahead units. These signals are the inputs to the CSL blocks at the leaves of the lookahead tree. Such a conversion algorithm provides a similar advantageous structure as parallel-prefix Ling's carry generation algorithm [19]. Instead of generating all of the carry propagation signals like a traditional parallel prefix adder, our hybrid CLA and CSL conversion method can make use of pseudocarries to create the selected carry propagation signals.

In the above derivation, the block factor is assumed to be uniform and equal to 3. It is noted that, in hybrid CLA/CSL configuration, the choice of carries in the carry network varies with the block factor of CLA and the block length of CSL. It also affects the internal load distribution of the lookahead logic and the depth of the carry tree. For a fixed word length of RB operand, more than one solution is available to implement the hybrid CLA/CSL-based reverse converter. Typically, the block factor and block length are chosen to equalize the critical carry generation chain and the carry-select chain. The timings of these two chains are highly dependent on the logic styles and fan-out factor per gate for a given technology of implementation. In what follows, we will probe on this structural optimization with reference to CMOS and branch-based logic design style [20], whereby the logic cells involved are made of parallel branches, each with a limited number of serially connected transistors.

B. Parallel-Prefix Carry-Lookahead With Uniform and Nonuniform Block Factors

Motivated by the design space of hybrid CLA/CSL architectures for RB-to-NB reverse conversion, our aim in this section is to find an optimal point to combine the CLA and CSL blocks for better performance by analyzing the consequential carry generation schemes with uniform and nonuniform block factors. The block factor refers to the number of binary terms in the Boolean product P_i and Boolean sum G_i in the generalized expression of (10). For layout regularity and balanced multiplexer load, it makes good sense to interleave the carry-select signals generated by the CLA network evenly to all CSL sections. While keeping the block lengths of CSL sections identical, the block factors at different stages of the carry generation network can

vary to minimize the difference between the arrival time of the carry-select signal and the critical delay of RCA in the CSL section.

For an N-bit RB operand, let l indicate the depth of the carry generation tree, i.e., the maximum number of lookahead cells traversed from any input to the final carry generation unit. Further, let b_i denote the block factor of the cells at stage i where $i = 1, 2, \ldots, l$. When nonuniform block factor is used, it is important to use an interconnection structure that is regular to ease its implementation. We adopt the Kogge–Stone liked tree [21] for our study as the fan-out of each cell throughout the network can be made fairly constant, especially for those that lie on the critical paths. To account for the number of cells with varying fan-ins and the carries they generated for a given operand length, we define a transitive stage $t \in (1, l]$ as a stage in the carry generation network where the outputs of all cells in this stage are separated by exactly the block length m of the CSL section, i.e., $m = \prod_{i=1}^{t} b_i$.

Then, the positions of carries $D(i, j)$ generated by the jth cell located at the ith stage of the carry generation network with block factor b_i for $i = 1, 2, \ldots, t$ can be determined by the following equation:

$$\begin{aligned} D(i,j) &= j \prod_{r=1}^{i} b_r - 1 \\ \forall i &= 1, 2, \cdots, t \\ j &= 1, 2, \cdots, \left\lfloor N / \prod_{r=1}^{i} b_r \right\rfloor . \end{aligned} \tag{12}$$

For stages beyond t, the positions of the carries generated are given by the following equation:

$$\begin{aligned} D(i,j) &= \left(j + \prod_{r=t+1}^{i} b_r \right) \prod_{r=1}^{t} b_r - 1 \\ \forall i &= t+1, t+2, \cdots, l \\ j &= 1, 2, \cdots, \left\lfloor \left(N - \prod_{r=1}^{i} b_r \right) / \prod_{r=1}^{t} b_r \right\rfloor . \end{aligned} \tag{13}$$

In (11) and (12), j is the index of the lookahead cell enumerated from the right of the tree. The number of stages l of the carry network is bounded by

$$\lceil \log_{b_{\max}} N \rceil \leqslant l \leqslant \lceil \log_{b_{\min}} N \rceil \tag{14}$$

where $b_{\max}$ and $b_{\min}$ are the maximum and minimum block factors, respectively.

All cells in the carry generation network can be built using a single complex gate in static CMOS logic design style since it offers high noise margins and robustness to device and voltage scalings [22]. However, it should be noted that, in practice, complex CMOS gates are used for a maximum fan-in of 5 to 6 [23]. To avoid chaining many p-channel devices in series for low-voltage and deep-submicrometer technology, we restrict the maximum number of MOSFET in series from V_{DD} to ground to no more than 6, i.e., $b_i \leqslant 3$ for all i to investigate the variants of mixed binary and ternary radix carry-lookahead trees using the prefix notation [18], [21].

In prefix notation, the prefix operator, denoted by "•," operates on pairs of binary generate and propagate signals. The initial generate and propagate pairs are represented by (g_i, p_i). From (7), the setup time of individual generate and propagate signals for prefix addition have been substantially reduced due to the exploitation of the redundancy of RB encoding. The binary pair $(G_{i:j}, P_{i:j})$ is used to denote the group generate and propagate terms produced from bits $i, i-1, \ldots, j+1$ to j. From [14], we have: $(G_{i:j}, P_{i:j}) = (g_i, p_i) \bullet (g_{i-1}, p_{i-1}) \bullet \ldots \bullet (g_{j+1}, p_{j+1}) \bullet (g_j, p_j)$, where $i > j (g_i, p_i) \bullet (g_{i-1}, p_{i-1}) = (g_i + p_i g_{i-1}, p_i p_{i-1}) (b = 2) (g_i, p_i) \bullet (g_{i-1}, p_{i-1}) \bullet (g_{i-2}, p_{i-2}) = (g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2}, p_i p_{i-1} p_{i-2}) (b = 3)$

As the prefix operator is idempotent, $(G_{i:j}, P_{i:j})$ can be derived by the association of two overlapping terms as follows [14]: $(G_{i:j}, P_{i:j}) = (G_{i:n}, P_{j:n}) \bullet (G_{m:j}, P_{m:j})$, where $i > m \geqslant n > j$

Fig. 2 illustrates the possible implementations of an 18-bit parallel-prefix carry computation with the uniform block factor (fixed radix) of 2 or 3, and nonuniform block factor (mixed radix) of 2 and 3. In this figure, the solid node • represents the prefix operator, and the white hollow node o represents flow-through node or buffer. Fig. 2(a)–(c) depicts several carry generation schemes for different block lengths of CSL, m of 2, 4, and 8 bits using a fixed block factor of $b = 2$. Similarly, Fig. 2(d) and 2(e) show two different schemes with $m = 3$ and 9 using a fixed block factor of 3. Various combinations of m and b for mixed radix schemes are illustrated in Fig. 2(f)–(l). It is worth noting that, in the mixed radix schemes, the block factor is uniform within the same stage but nonuniform across stages. Completely mixed-radix carry generation trees are analytically obscure to unification with the CSL sections for hybrid CLA/CSL due to the immense number of complex formations that could be enumerated. The use of mixed radix cells within the same stage can easily annihilate the regularity and unnecessarily complicate circuit and layout optimizations. Therefore, it will only be considered on a sparse number of leave cells after the optimal regular mixed radix tree has been established and provided its use will reduce the depth of the tree with minimal impediment on layout regularity.

The speed of the hybrid CLA/CSL architecture is minimized when the critical delay of the CLA network is commensurate with that of the CSL sections. Therefore, for speed optimization, the block factor for the CLA complex gates at each stage of the hierarchy can be optimized to tailor to the optimal block length of CSL based on the implementations of the RCA and multiplexer logics. The RB encoding has been beneficially exploited in the RCA for the CSL circuit. Transistor-level circuit design techniques have been applied to devise a new area- and power-efficient add-one circuit for the CSL adder. The details of the novel CSL circuit for the hybrid CLA/CSL reverse converter is best demonstrated with a 64-bit RB-to-NB converter to be presented in Section IV. The critical path for both the CLA network and CSL sections are best evaluated with a good technology-independent model that account for loading at transistor-level implementation to determine an optimized mixed-radix CLA structure for various operand lengths. This will be carried out in Section V.

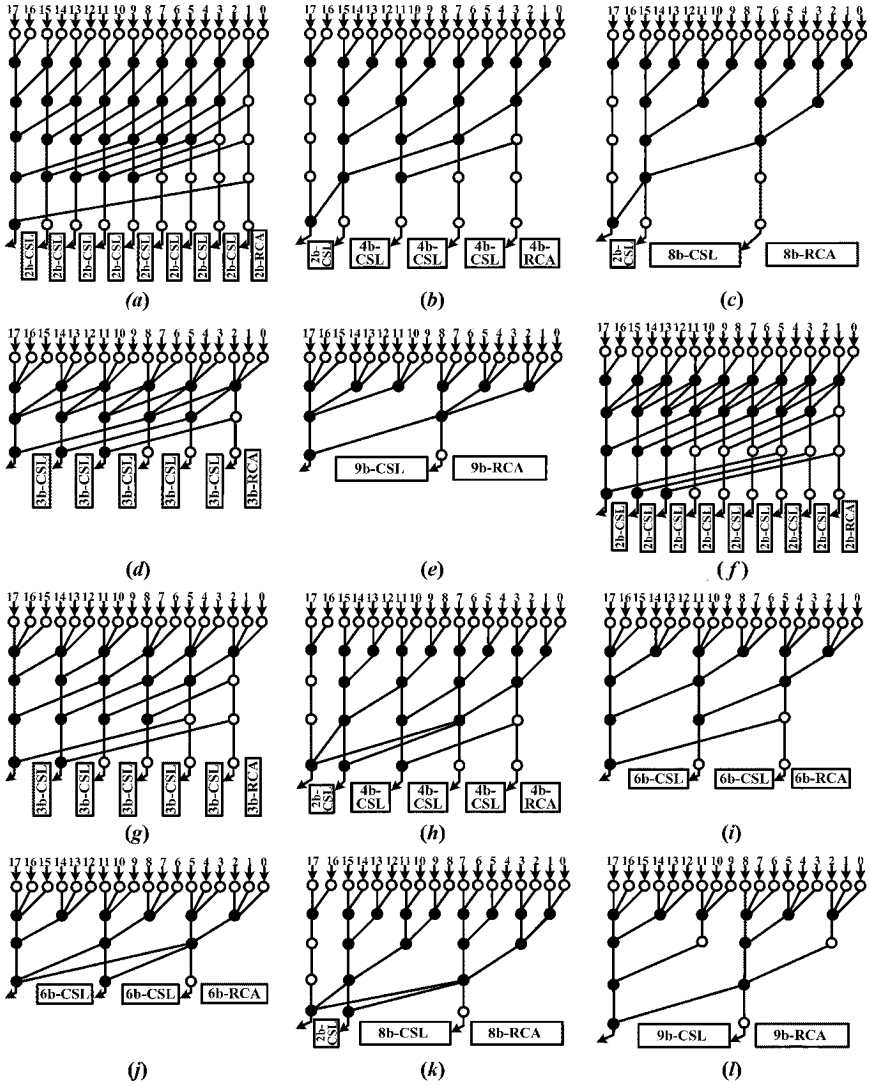


Fig. 2. 18-bit parallel-prefix carry generation with various block factors and block lengths of CSL. (a) $b = 2$ and $m = 2$. (b) $b = 2$ and $m = 4$. (c) $b = 2$ and $m = 8$. (d) $b = 3$ and $m = 3$. (e) $b = 3$ and $m = 9$. (f) $b \in \{2, 3\}$ and $m = 2$. (g) $b \in \{2, 3\}$ and $m = 3$. (h) $b \in \{2, 3\}$ and $m = 4$. (i) $b \in \{2, 3\}$ and $m = 6$. (j) $b \in \{2, 3\}$ and $m = 6$. (k) $b \in \{2, 3\}$ and $m = 8$. (l) $b \in \{2, 3\}$ and $m = 9$.

IV. Implementation of a 64-b Reverse Converter

A. Architecture of 64-bit Reverse Converter

Here, a 64-bit RB-to-NB reverse converter is implemented with our proposed CLA/CSL-based architecture. In principle, the block length of CSL can take any discrete value for any combination of the product of b_1 through b_t at the transitive stage. In our implementation, we have limited the fan-in and fan-out of every circuit node to no more than 3 to avoid chaining more than three p-channel/n-channel devices in series. Therefore, the length of the CSL section will be a power of 2 or 3 or a multiple of 6 since the value of b_i is confined to 2 or 3. We have selected an optimal block length of CSL, $m = 6$, to match a mixed-radix carry generation network from among the plausible topologies presented in Section III based on an objective evaluation through a credible logical effort model. The model of evaluation and the results will be discussed in detail in the next section.

For CSL sections of constant block length $m = 6$, the implementation of a 64-bit reverse converter is not unique as the depth of the carry generation tree can be either 5 or 6 according to the block factors selected for each stage. An efficient implementation of a 64-bit reverse converter architecture with $m = 6$ is shown in Fig. 3. The numerical indices of i and j are used to identify the stage numbers and the positions of the CLA cells at each stage, respectively.

From Fig. 3, the regular carry generation network of the hybrid CLA/CSL-based 64-bit reverse converter is optimized with $b_1 = 3$ and $b_2 = b_3 = b_4 = b_5 = b_6 = 2$. For $m = 6$, the transitive stage occurs at stage 2 but b_1 instead of b_2 is selected to be of radix-3 to take advantage of the carry equation (10) with direct RB input in the first stage. From (12) and (13), the ten carry outputs generated by the carry-lookahead network are C_5, C_{11}, C_{17}, C_{23}, C_{29}, C_{35}, C_{41}, C_{47}, C_{53}, and C_{59}, each of which is to be fed into one of the ten CSL sections excluding the first section. The first section has a constant input, which can be implemented directly as a modified RCA without the add-one circuit. For the end section, a 4-bit CSL suffices since the total number of sum bits is 64 b. From the topological diagram, it is observed that the two cells H_{53} and H_{59} that generate the carries C_{53} and C_{59} at the last stage can be ganged with their preceding cells in Stage 5 to save one stage. The two radix-3 prefix operators

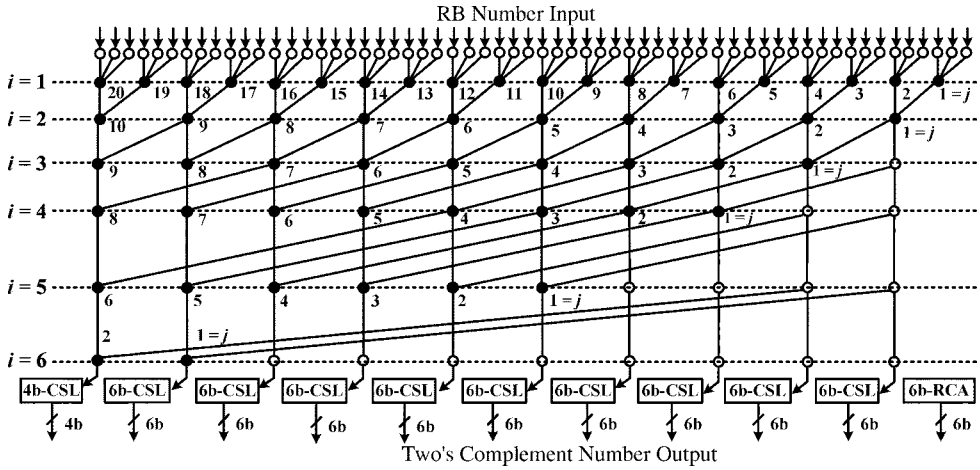


Fig. 3. Block diagram of the 64-bit reverse converter.

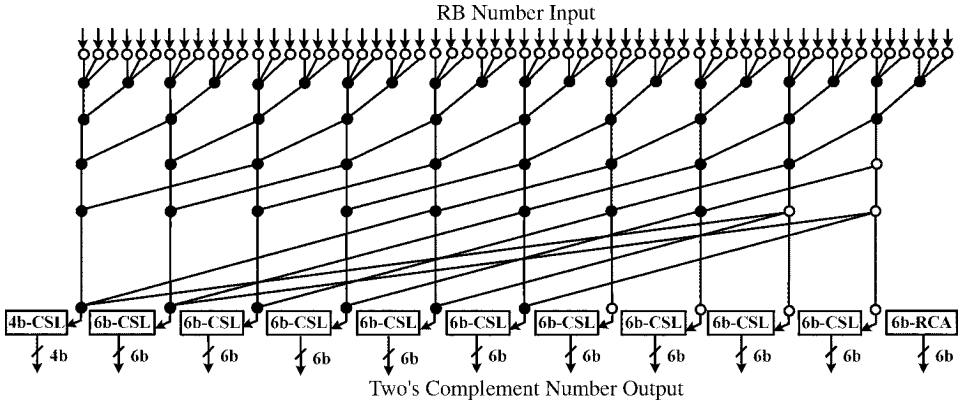


Fig. 4. Block diagram of the modified 64-bit reverse converter.

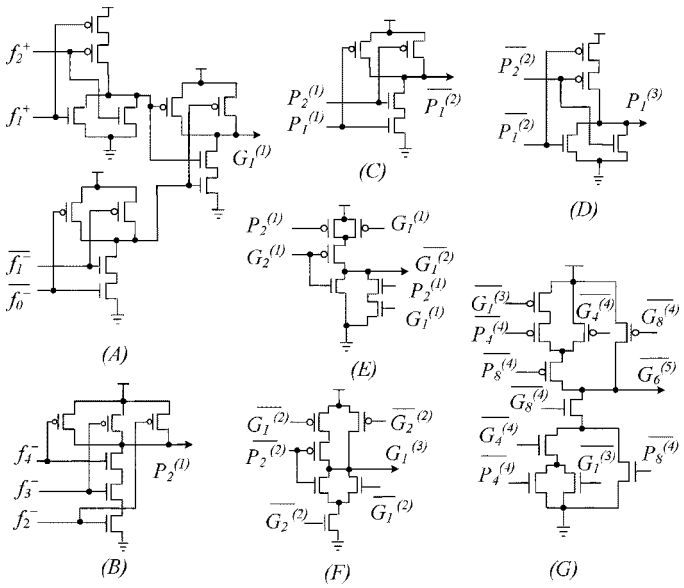


Fig. 5. Circuit implementation of G and P cells in the five-stage CLA network.

formed have the additional inputs stemmed from the first two buffer nodes at Stage 4. Therefore, none of the operators has a fan-out that exceeds three, and there is no change in the critical path except a slight perturbation to the wiring. Fig. 4 shows the block diagram of the final five-stage reverse converter.

Let $P_j^{(i)}$ and $G_j^{(i)}$ represent the carry propagate and generate signals produced by the jth prefix operator at the ith stage. Then, using the expansion of (10), the mixed-radix lookahead cells of the CLA network are constructed with the following logical expressions after appropriate resubstitution of terms:

$$\begin{aligned}
H_5 &= G_2^{(1)} + P_2^{(1)} \cdot G_1^{(1)} \\
H_{11} &= G_2^{(2)} + P_2^{(2)} \cdot G_1^{(2)} \\
H_{17} &= G_2^{(3)} + P_2^{(3)} \cdot G_1^{(2)} \\
H_{23} &= G_3^{(3)} + P_3^{(3)} \cdot G_1^{(3)} \\
H_{29} &= G_3^{(4)} + P_3^{(4)} \cdot G_1^{(2)} \\
H_{35} &= G_4^{(4)} + P_4^{(4)} \cdot G_1^{(3)} \\
H_{41} &= G_5^{(4)} + P_5^{(4)} \cdot G_1^{(4)} \\
H_{47} &= G_6^{(4)} + P_6^{(4)} \cdot G_2^{(4)} \\
H_{53} &= G_5^{(5)} + P_5^{(5)} \cdot G_1^{(2)} \\
&= G_7^{(4)} + P_7^{(4)} \cdot G_3^{(4)} + P_7^{(4)} \cdot P_3^{(4)} \cdot G_1^{(2)} \\
H_{59} &= G_6^{(5)} + P_6^{(5)} \cdot G_1^{(3)} \\
&= G_8^{(4)} + P_8^{(4)} \cdot G_4^{(4)} + P_8^{(4)} \cdot P_4^{(4)} \cdot G_1^{(3)}. \qquad (15)
\end{aligned}$$

Only seven different types of cells are required for the carry generation network. These schematics are shown in Fig. 5. In Fig. 5, type-A and -B cells are used to generate the G signals in the first stage and P signals for all radix-3 prefix operators, respectively. Type-C and -D cells are NAND and NOR gate used

to generate the P signals for all radix-2 prefix operators. Type-E and -F cells are complex gates used to generate the G signals for all radix-2 prefix operators. A type-G cell is a complex gate used merely to generate H_{53} and H_{59} at Stage 5. It should be noted that all of the P and G outputs alternate in polarities in the odd and even stages to unify the cells used. This unification not only simplifies and modularizes the circuit of the carry network but also reduces its delay.

B. Design Considerations: Modified Add-One CSL Scheme

Among the arithmetic circuit design techniques, CSL has emerged as an eminent approach to address the area–time tradeoff of CPA design. It exhibits the advantage of logarithmic gate depth as in any structure of the distant-carry adder family. When it is used together with CLA in the proposed RB-to-NB reverse converter, higher speed can be achieved at the expense of increased hardware cost. Our approach to hybrid CLA/CSL design differs from others [13], [16], [20] in that we combine the logic structure with a circuit technique to minimize the number of transistors used in the CSL section without degrading its performance. Besides, we have fully tapped on the redundancy of RB encoding to halve the logics in the mandatory copy of the RCA of each CSL section, which further speeds up its sum and carry generation.

From (8), two separate carry chains with block carry-in of 0 and 1 can be derived for the CSL network. The benefit of having two carry chains is the saving of certain hardware resources. However, it also has the potential to increase the overall latency of the hybrid CLA/CSL circuit. A possible solution to improve the speed is to use two copies of RCA blocks and select the correct sum value according to the final block carry-in signals. Although it improves the speed significantly, the amount of transistors used is also sizable. To circumvent this problem, an add-one circuit was proposed by Chang in [24]. As opposed to using dual RCAs in the conventional CSL, the architecture of contemporary CSL adder comprises a single RCA, a first zero detection and selective complement add-one circuit, and a carry-select multiplexer circuit as shown in Fig. 6 [24]. It achieves a 29.2% area reduction at the expense of 5.9% speed penalty for a 64-bit CSL over the conventional dual-RCA design. The circuit was further modified by Kim [25] to achieve even better performance. Unfortunately, an omission of a multiplexer in the MSB position of the add-one block was found in the design depicted in the circuit architecture schematic of [25].

We improve the add-one CSL circuit to further minimize the transistor count without any speed penalty. Since the add-one circuit is based on a "first" zero detection logic, it generates Z^1 by inverting each bit in Z^0 starting from the LSB until the first zero is encountered, where Z^0 and Z^1 are the sum outputs of the two copies of RCA with block carry-in 0 and 1, respectively. Fig. 7 depicts the proposed add-one circuit using buffers with only one inverter, and we have proved that the add-one circuit with single inverter buffers performs exactly the same function as that shown in Fig. 6 [26].

In Fig. 6, the complements of the sum bits are generated from the internal nodes of a pMOS-nMOS chain. Before the first zero bit is detected, each pMOS and nMOS pair functions as an inverter. Once the first zero bit has occurred, the pMOS and nMOS pair acts as a multiplexer, which selects either $V_{\rm DD}$ or Z^*_{i-1} as described by

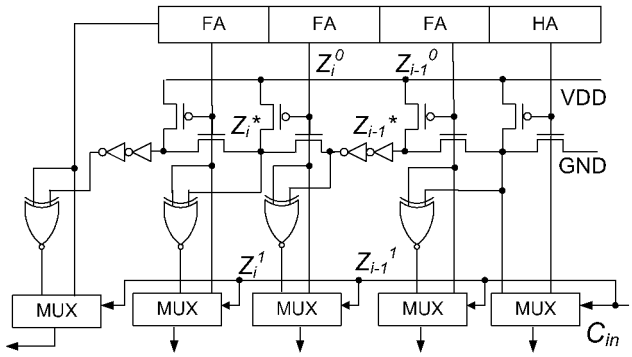


Fig. 6. CSL with a single RCA and an add-one circuit [24].

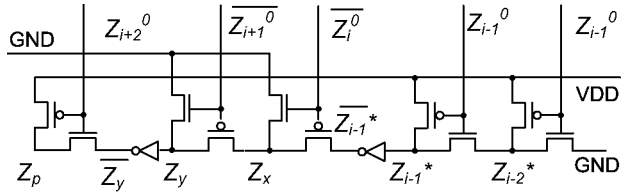


Fig. 7. Modified add-one scheme.

$$Z_i^* = Z_i^0 \cdot Z_{i-1}^* + \overline{Z_i^0}. \tag{16}$$

To select the correct sum

$$Z_i = Z_i^1 \cdot C_{\rm in} + Z_i^0 \cdot \overline{C_{\rm in}} = \left(Z_i^0 \odot Z_{i-1}^*\right) \cdot C_{\rm in} + Z_i^0 \cdot \overline{C_{\rm in}}. \tag{17}$$

In our proposed add-one circuit of Fig. 7, there is no change in the output logic before the buffer is inserted. From (16) and (17), we have

$$\begin{aligned} \overline{Z_i^*} &= \overline{Z_i^0 \cdot Z_{i-1}^* + \overline{Z_i^0}} = \overline{Z_{i-1}^*} \cdot Z_i^0 \\ Z_x &= \overline{\overline{Z_i^0}} \cdot \overline{Z_{i-1}^*} + \overline{Z_i^0} \cdot 0 = Z_i^0 \cdot \overline{Z_{i-1}^*} = \overline{Z_i^*}. \end{aligned} \tag{18}$$

Similarly

$$\begin{aligned} Z_y &= Z_{i+1}^0 \cdot Z_x = Z_{i+1}^0 \cdot \overline{Z_i^*} = \overline{Z_{i+1}^*} \\ Z_p &= Z_{i+2}^0 \cdot \overline{Z_y} + \overline{Z_{i+2}^0} = Z_{i+2}^0 \cdot Z_{i+1}^* + \overline{Z_{i+2}^0} = Z_{i+2}^*. \end{aligned} \tag{19}$$

This verifies that our proposed add-one circuit is functionally equivalent to the circuit of Fig. 6, but the total number of inverters has been reduced by half. There is no speed penalty since the block carry-in signals are generated in parallel by the CLA network rather than locally generated from the CSL sections sequentially. In fact, it is envisaged that, with the shorter chain and potential reduction of internal signal togglings, power dissipation will be lowered. The internal carry chain of RCA can also be shortened by leveraging on the special property of RB numbers from (7). This is further elaborated as follows.

From the original input f_i^+ and f_i^- of the RB number to be converted, the internal carry signal of a RCA can be simplified as follows:

$$\begin{aligned} c_0^0 &= f_0^+ \\ \overline{c_1^0} &= \overline{f_1^+ + \overline{f_1^-} \cdot c_0^0} = \overline{\overline{f_1^-} \cdot \left(f_1^+ + c_0^0\right)} \\ c_2^0 &= \overline{f_2^- + \overline{f_2^+} \cdot \overline{c_1^0}} = \overline{\overline{f_2^+} \cdot \left(f_2^- + \overline{c_1^0}\right)} \\ &\vdots \end{aligned} \tag{20}$$

Therefore, the carry generation chain in the proposed RCA can be readily implemented in branch-based logic style to minimize the number of internal connections [20]. Branch-based circuits possess high noise margins and robustness to voltage and device scaling reminiscent of classical static CMOS design style. From (7), by exploiting the RB encoding, both the sum bits and its complement can be generated simultaneously using the new XOR/XNOR circuit [27] to enhance their driving capability. Comparing with the full adder of Fig. 6, which requires two XOR gates for the sum generation, the propagation delay has been reduced. With the cogeneration of complementary sum bits, the threshold voltage ($V_{\rm th}$) drop problem of pass transistors [28] in the add-one circuit of Fig. 6 can also be overcome by replacing them with transmission gates.

The modified 6-bit ripple-carry chain and the new add-one circuit are integrated into a 6-bit CSL as shown in Fig. 8. Identical circuit topology for the odd and even carry-generation cells are used to implement the carry signals with alternating polarities of inputs and outputs in the modified ripple-carry chain. At the bottom, the final sum of CSL can be generated from the add-one circuit by using only a group of NAND gates and multiplexers. It realizes the following logic equation derived from (17):

$$\begin{aligned} Z_i &= Z_i^0 \cdot Z_{i-1}^* \cdot C_{\rm in} + \overline{Z_i^0} \cdot \overline{Z_{i-1}^*} \cdot C_{\rm in} + Z_i^0 \cdot \overline{C_{\rm in}} \\ &= Z_i^0 \cdot \left(\overline{Z_{i-1}^* \cdot C_{\rm in}}\right) + \overline{Z_i^0} \cdot \left(\overline{Z_{i-1}^* \cdot C_{\rm in}}\right). \end{aligned} \tag{21}$$

The output Z_i is selected from the two data input signals Z_i^0 and its complement. The select signal $\overline{Z_{i-1}^* \cdot C_{\rm in}}$ is generated by a NAND gate from the carry-in $C_{\rm in}$ and the complement of Z_{i-1}^*. Thus, we can eliminate one inverter in each buffer from the corresponding block of Fig. 6 without violating its functionality. The NAND gates also function as buffers to improve the driving capability.

V. Performance Evaluation

This section evaluates the performance of our newly proposed converter architecture and compares it with three competitive converters [4], [10], [11]. According to Section III-B, a fixed-size operand can have several ramifications of architecture depending on the block factors and block lengths of the CSL sections chosen for the implementation. Therefore, we are interested to find an optimum realization for a given operand size from among the feasible solutions of our base converter architecture.

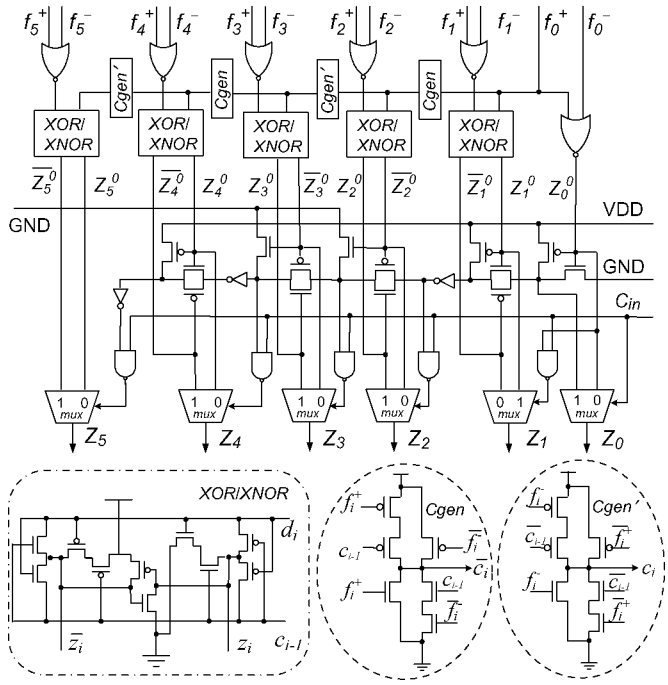


Fig. 8. 6-bit carry-select block with modified add-one scheme.

A critical path is the worst-case delay from any input transition to the latest output transition over all possible input patterns. As pointed out in [12], comparison of complex gates based on a unit gate delay model in CMOS digital circuits can be misleading because the delays are largely influenced by their circuit topology (fan-in) and loading (fan-out). Recently, a popular method called logical effort (LE) based on the *RC* delay model provides a convenient shorthand for more realistic speed estimation of CMOS digital circuits [29], [30]. It captures reasonably well the effect of transistor sizing for different transistor-level implementations according to the critical paths of the corresponding circuit architectures and their loading. The accuracy of LE estimation has been attested by reliable circuit simulation tool HSPICE by several researchers [30]–[32]. It therefore provides a fast and consistent means to evaluate the potential performance of a new reverse converter architecture as well as the relative performances among different converters.

Using the logic effort model, we perform the complexity analysis of our proposed reverse converter for operand lengths of 8, 16, 32, 64, and 128 with various block factors of CLA and block lengths of CSL. The results are shown in Table I. Note that the delay time is normalized to that of a fanout-of-4 (FO4) inverter delay [29]. Although for certain operand lengths there exist more than one implementations correspond to certain block factor and block length of CSL, only the fastest solution is presented in Table I. Since the transistor count of a circuit has an indirect correlation to the VLSI area, the number of transistors in each circuit is accounted for and summarized in Table II. The area-delay products (ATs) are also provided in Table III, where the area is measured in terms of transistor count, the AT values are normalized by the AT value of the case of $b \in \{2, 3\}$ and $m = 6$, and the obtained ratios are expressed

TABLE I
COMPARISON OF DELAY FOR DIFFERENT COMBINATIONS OF BLOCK FACTORS OF CLA AND BLOCK LENGTHS OF CSL

N	Delay (FO4)										
	b=2			b=3		$b \in \{2,3\}$					
	m=2	m=4	m=8	m=3	m=9	m=2	m=3	m=4	m=6	m=8	m=9
8	4.60	4.17	—	4.58	—	4.64	4.58	4.17	6.11	—	—
16	6.35	6.20	8.23	6.48	8.57	6.49	5.57	5.59	6.89	8.23	8.57
32	7.51	7.46	8.23	7.44	9.24	7.71	7.25	7.07	7.09	8.23	9.24
64	8.63	8.59	8.51	8.87	9.92	8.92	8.66	8.43	8.34	8.48	9.46
128	9.83	9.79	9.66	10.83	10.32	10.45	9.84	9.70	9.62	9.64	9.81

TABLE II
COMPARISON OF TRANSISTOR COUNT FOR DIFFERENT COMBINATIONS OF BLOCK FACTORS OF CLA AND BLOCK LENGTHS OF CSL

N	Number of Transistors										
	b=2			b=3		$b \in \{2,3\}$					
	m=2	m=4	m=8	m=3	m=9	m=2	m=3	m=4	m=6	m=8	m=9
8	180	178	—	149	—	178	149	178	154	—	—
16	468	476	416	496	400	448	500	466	456	416	400
32	1142	1126	1070	1121	1064	1138	1216	1108	1120	1062	1064
64	2664	2520	2502	2594	2274	2732	2579	2472	2486	2480	2478
128	6056	5482	5270	5539	4972	6503	5658	5238	5140	5094	5196

TABLE III
COMPARISON OF AREA–DELAY PRODUCT FOR DIFFERENT COMBINATIONS OF BLOCK FACTORS OF CLA AND BLOCK LENGTHS OF CSL

N	Area-Delay Product (%)										
	b=2			b=3		$b \in \{2,3\}$					
	m=2	m=4	m=8	m=3	m=9	m=2	m=3	m=4	m=6	m=8	m=9
8	88.0	78.9	—	72.5	—	87.8	72.5	78.9	100	—	—
16	94.6	93.9	109.0	102.3	109.1	92.5	88.6	82.9	100	109.0	109.1
32	108.0	105.8	110.9	105.0	123.8	110.5	111.0	98.7	100	110.1	123.8
64	110.9	104.4	102.7	111.0	108.8	117.5	107.7	100.5	100	101.4	113.1
128	120.4	108.5	103.0	121.3	103.8	137.4	112.6	102.8	100	99.3	103.1

in percentage. From these results, the most optimum reverse converter with the minimum AT for each operand length is selected to compare against other fast reverse converters.

For a fair comparison, the contender circuits CONV1 [4], CONV2 [10], and CONV3 [11] were replicated as reported in the literature. For high-speed operation, CONV1 is optimized using CSL with minimized critical path. CONV2 is a simplified CSL, which uses inverters instead of complex CMOS circuits to produce the "generate signal G" and the "propagate signal P" according to the consideration presented in [10]. CONV3 uses pass transistor logic. Hence, buffers are inserted in every two transmission gates to prevent the signal from decaying.

The delays of different reverse converters based on the LE model for various operand lengths are tabulated in Table IV. The number of transistors required by each converter is shown in Table V. The ATs are also provided in Table VI to compare the combined criterion of cost and performance of different reverse converters. In Table VI, the AT cost of CONV3 is used

TABLE IV
COMPARISON OF DELAY OF DIFFERENT CONVERTERS

Word Length	Delay (FO4)			
	This Work	CONV1 [4]	CONV2 [10]	CONV3 [11]
8	4.17	6.37	6.21	10.00
16	5.59	8.21	8.58	10.95
32	7.07	10.00	10.26	14.74
64	8.34	12.74	13.58	15.47
128	9.64	15.93	17.05	19.71

TABLE V
COMPARISON OF TRANSISTOR COUNT OF DIFFERENT CONVERTERS

Word Length	Number of Transistors			
	This Work	CONV1 [4]	CONV2 [10]	CONV3 [11]
8	178	264	228	262
16	466	536	564	604
32	1108	1048	1268	1312
64	2486	1840	2676	2936
128	5094	3906	5874	6430

TABLE VI
COMPARISON OF AREA–DELAY PRODUCT OF DIFFERENT CONVERTERS

Word Length	Area-Delay Product (%)			
	This Work	CONV1 [4]	CONV2 [10]	CONV3 [11]
8	28.3	64.2	54.0	100
16	39.4	66.5	73.2	100
32	40.5	54.2	67.2	100
64	45.6	51.6	80.0	100
128	38.7	49.1	79.0	100

as a reference to normalize the AT cost of all other converters. From the above tables, it is evident that our newly proposed reverse converter has the minimum AT for any operand length in comparison with other converters. More importantly, the delay of our proposed converter does not escalate with the increase of operand length as badly as other converters do. As the word length increases, the improvement in the combined area–time performance becomes more prominent.

To validate and reinforce the results estimated by the LE model, we implement two 64-bit reverse converters. One is our proposed converter and the other is CONV1 since it is the most competitive one according to its performance evaluated earlier in Tables IV–VI. As power consumption has become an increasingly important performance criterion, "using a design that is fast enough and consumes the least power" might be a better rule of thumb than "using the fastest design" [16]. Therefore, besides the worst case delay, the converters are also simulated for their average power consumptions. For a fair comparison, both circuits are optimized in speed according to the critical paths estimated from the architectures. The optimization process was carried out recursively until all transistor sizes converged [28]. All of the circuits are simulated using HSPICE based on the TSMC 0.18-μm CMOS process model. For each simulation, HSPICE

TABLE VII
COMPARISONS OF 64-bit REVERSE CONVERTERS

64-bit Reverse Converter	Delay (ps)	Power (mW)	PDP (pJ)
This Work	636	0.38	0.242
CONV1 [4]	979	0.64	0.627

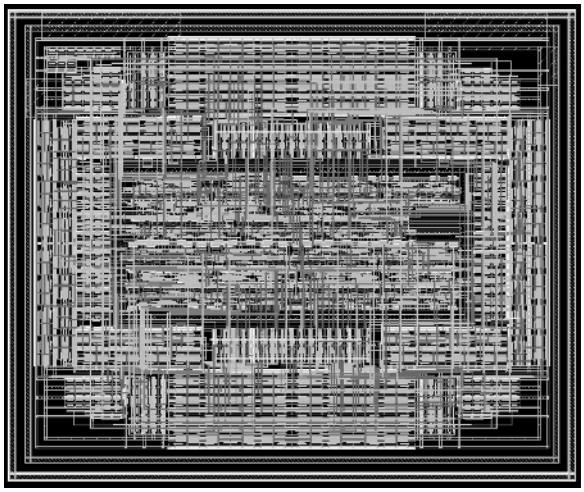


Fig. 9. Full-custom layout of the proposed 64-bit reverse converter.

will generate an average power consumption value. As the dynamic power dissipation increases linearly with frequency and quadratic with supply voltage, both circuits are simulated at the same data rate of 100 MHz and the same supply voltage of 1.8 V with 4096 randomly generated input data. A comparison of these two converters in terms of the worst case delay, average power dissipation, and their product are listed in Table VII.

From Table VII, our proposed 64-bit reverse converter outperforms CONV1. It runs 1.5 times faster than CONV1 and consumes 40% less power. This simulation result is well correlated to the relative performance difference between our converter and CONV1 in Table I for the 64-bit word length. The gate delay of an FO4 inverter for TSMC 0.18-μm CMOS process technology at 1.8 V is simulated to be 70 ps. Therefore, the deviation between HSPICE prelayout simulation and LE estimation is less than 10%. This validates the legitimacy of the rapid performance evaluation based on the LE model.

A full-custom layout of the proposed 64-bit reverse converter circuit is carried out using the TSMC 0.18-μm CMOS process, which features six metal and one poly layers. The layout pattern of the converter is shown in Fig. 9 and the postlayout simulation results are summarized in Table VIII. Table VIII presents a more accurate delay and power consumption evaluation of our proposed converter as the parasitics attributed to wires have been back annotated for the postlayout simulation.

VI. CONCLUSION

In this paper, we have shown that the inherent redundancy of RB encoding can be fully exploited to simplify and speed up the reverse conversion through an elegant amalgamation of mixed-radix carry-lookahead network and novel carry-select adder. A hybrid CLA/CSL adder realization is well suited to the proposed formulation of the reverse conversion problem. The carries of the CLA network are selected to equalize the critical path of the optimally designed CSL sections for a given operand length. The carry generation network is implemented with heterogeneous CMOS cells, and the CSL block is simplified without jeopardizing the critical path delay by making use of the group carry-in signals generated by the multilevel CLA network. To further reduce the cost of implementing the CSL, the ripple-carry adder chain is modified and incorporated with a new add-one circuit. We have shown by means of logical effort technique that the proposed reverse converter outperforms three other competitive converters in terms of delay, transistor count and their products for operand lengths vary from 8 to 128 b. The speed improvement over other converters is more provident with increased operand length. HSPICE simulation results of a 64-bit transistor-level implementation of our proposed converter and the best contender obtained from the LE delay model proved the superiority of our proposed converter.

TABLE VIII
POSTLAYOUT FIGURE-OF-MERIT OF PROPOSED 64-bit REVERSE CONVERTER

Area (mm^2)		0.08		
Supply Voltage (V)		3.3	1.8	1.1
Delay Time (ps)		598	829	1618
Average Power Dissipation (mW)	50MHz	0.79	0.239	0.089
	100MHz	1.67	0.492	0.187
	500MHz	8.85	2.61	0.959
	1GHz	19.3	5.84	—

REFERENCES

[1] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "A high-speed multiplier using a redundant binary adder tree," *IEEE J. Solid-State Circuits*, vol. SC-22, no. 1, pp. 28–34, Feb. 1987.

[2] K. K. Primlani and J. L. Meador, "A nonredundant-radix-4 serial multiplier," *IEEE J. Solid-State Circuits*, vol. 24, no. 6, pp. 1729–1736, Dec. 1989.

[3] Y. Wang, "Residue-to-binary converters based on new Chinese remainder theorems," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 3, pp. 192–206, Mar. 2000.

[4] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Mashiko, "An 8.8 ns 54 × 54-bit multiplier with high speed redundant binary architecture," *IEEE J. Solid-State Circuits*, vol. 31, no. 6, pp. 773–783, Jun. 1996.

[5] Y. Kim, B.-S. Song, J. Grosspietsch, and S. F. Gillig, "A carry-free 54b × 54b multiplier using equivalent bit conversion algorithm," *IEEE J. Solid-State Circuits*, vol. 36, no. 10, pp. 1538–1545, Oct. 2001.

[6] Y. Kim, B.-S. Song, J. Grosspietsch, and S. F. Gillig, "Correction to 'a carry-free 54b × 54b multiplier using equivalent bit conversion algorithm'," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, p. 159, Jan. 2003.

[7] W. Rulling, "A remark on carry-free binary multiplication," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 159–160, Jan. 2003.

[8] M. D. Ercegovac and T. Lang, "Comments on 'a carry-free 54b × 54b multiplier using equivalent bit conversion algorithm'," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 160–161, Jan. 2003.

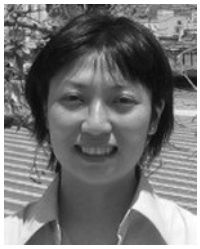
[9] I. Choo and R. G. Deshmukh, "A novel conversion scheme from a redundant binary number to two's complement binary number for parallel architectures," in *Proc. IEEE Southeast Conf.*, Clemson, SC, Apr. 2001, vol. 2, pp. 196–201.

[10] S. M. Yen, C. S. Laih, C. H. Chen, and J. Y. Lee, "An efficient redundant-binary number to binary number converter," *IEEE J. Solid-State Circuits*, vol. 27, no. 1, pp. 109–112, Jan. 1992.

[11] H. R. Srinivas and K. K. Parhi, "A fast VLSI adder architecture," *IEEE J. Solid-State Circuits*, vol. 27, no. 5, pp. 761–767, May 1992.

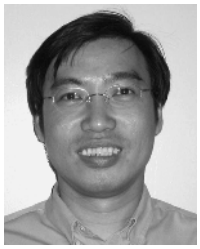
[12] N. Quach and M. J. Flynn, "High speed addition in CMOS," *IEEE Trans. Comput.*, vol. 41, no. 12, pp. 1612–1615, Dec. 1992.

[13] T. Lynch and E. E. Swartzlander, Jr., "A spanning tree carry lookahead adder," *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 931–939, Aug. 1992.
[14] V. Kantabutra, "A recursive carry-lookahead/carry-select hybrid adder," *IEEE Trans. Comput.*, vol. 42, no. 12, pp. 1495–1499, Dec. 1993.
[15] Y. Wang, C. Pai, and X. Song, "The design of hybrid carry-lookahead/carry-select adders," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 49, no. 1, pp. 16–24, Jan. 2002.
[16] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 10, pp. 689–702, Oct. 1996.
[17] T. P. Kelliher, R. M. Owens, M. J. Irwin, and T.-T. Hwang, "ELM—A fast addition algorithm discovered by a program," *IBM J. Res. Dev.*, vol. 41, no. 9, pp. 1181–1184, Sep. 1992.
[18] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Trans. Comput.*, vol. C-31, no. 3, pp. 260–264, Mar. 1982.
[19] H. Ling, "High speed binary adder," *IBM J. Res. Dev.*, pp. 156–166, May 1981.
[20] A. Neve, H. Schettler, T. Ludwig, and D. Flandre, "Power-delay product minimization in high-performance 64-bit carry-select adders," *IEEE Trans. VLSI Syst.*, vol. 12, no. 3, pp. 235–244, Mar. 2004.
[21] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Comput.*, vol. 22, no. 8, pp. 786–793, Aug. 1973.
[22] M. Alioto and G. Palumbo, "Analysis and comparison on full adder block in submicron technology," *IEEE Trans. VLSI Syst.*, vol. 10, no. 6, pp. 806–823, Dec. 2002.
[23] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design: Circuits and Systems*. Dordrectht, The Netherlands: Kluwer, 1995.
[24] T. Y. Chang and M. J. Hsiao, "Carry-select adder using single ripple-carry adder," *Electron. Lett.*, vol. 34, no. 22, pp. 2101–2103, Oct. 1998.
[25] Y. Kim and L. S. Kim, "64-bit carry-select adder with reduced area," *Electron. Lett.*, vol. 37, no. 10, pp. 614–615, May 2001.
[26] Y. He, C. H. Chang, and J. Gu, "An area efficient 64-bit square root carry-select adder for low power applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'2005)*, Kobe, Japan, May 2005, vol. 4, pp. 4082–4085.
[27] C. H. Chang, J. Gu, and M. Zhang, "Ultra low voltage, low power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 10, pp. 1985–1997, Oct. 2004.
[28] C. H. Chang, J. Gu, and M. Zhang, "A review of 0.18- μm full adder performances for tree structured arithmetic circuits," *IEEE Trans. VLSI Syst.*, vol. 13, no. 6, pp. 686–695, Jun. 2005.
[29] I. Sutherland, R. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Mateo, CA: Morgan Kaufmann, 1999.
[30] D. Harris and I. Sutherland, "Logical effort of carry propagate adders," in *Proc. 37th IEEE Asilomar Conf. Signals, Syst., Comput. (ACSSC)*, Pacific Grove, CA, Nov. 2003, vol. 1, pp. 873–878.
[31] H. Q. Dao and V. Oklobdzija, "Application of logical effort on delay analysis of 64-bit static carry-lookahead adder," in *Proc. 35th IEEE Asilomar Conf. Signals, Syst., Comput. (ACSSC)*, Pacific Grove, CA, Nov. 2001, vol. 2, pp. 1322–1324.
[32] V. G. Oklobdzija, B. R. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, "Energy-delay estimation technique for high-performance microprocessor VLSI adders," in *Proc. 16th IEEE Symp. Comput. Arithmetic (ARITH)*, Santiago de Compostela, Spain, Jun. 2003, pp. 272–279.



Yajuan He (S'05) received the B.S. degree in electronic science and technology from the East China Normal University (ECNU), Shanghai, China, in 2001. She is currently working toward the Ph.D. degree at the Nanyang Technological University (NTU), Singapore.

From 2001 to 2002, she was an RTP Digital IC Designer with the Institute of Microelectronics (IME), Singapore. In 2007, she joined STMicroelectronics, Singapore, as a Digital IC Design Engineer in the Smartcards Division, Asia-Pacific Design Center (APDC). Her research interests include computer arithmetic circuits and low-power IC design.



Chip-Hong Chang (S'92–M'98–SM'03) received the B.Eng. (Hons) degree from National University of Singapore, Singapore, in 1989, and the M.Eng. and Ph.D. degrees from Nanyang Technological University (NTU), Singapore, in 1993 and 1998, respectively.

He served as a Technical Consultant in industry prior to joining the School of Electrical and Electronic Engineering, NTU, in 1999, where he is now an Associate Professor. He holds joint appointments at the university as Deputy Director of the Centre for High Performance Embedded Systems (CHiPES) since 2000 and Program Director of the Centre for Integrated Circuits and Systems (CICS) since 2003. His current research interests include low-power arithmetic circuits, constrained driven architectures for digital signal processing, and digital watermarking for IP protection. He has published three book chapters and more than 120 research papers in international refereed journals and conferences.

Dr. Chang is a Fellow of IET.