Hindawi Publishing Corporation
Advances in Electronics
Volume 2015, Article ID 713843, 13 pages
http://dx.doi.org/10.1155/2015/713843

Research Article

FPGA-Based Synthesis of High-Speed Hybrid Carry Select Adders

V. Kokilavani,[1] K. Preethi,[1] and P. Balasubramanian[2]

[1]*Department of PG Studies in Engineering, S. A. Engineering College (Affiliated to Anna University), Poonamallee-Avadi Road, Veeraraghavapuram, Chennai, Tamil Nadu 600 077, India*
[2]*Department of Computer Science and Engineering, S. A. Engineering College (Affiliated to Anna University), Poonamallee-Avadi Road, Veeraraghavapuram, Chennai, Tamil Nadu 600 077, India*

Correspondence should be addressed to P. Balasubramanian; spbalan04@gmail.com

Received 30 September 2014; Revised 21 April 2015; Accepted 4 May 2015

Academic Editor: Gianluca Traversi

Carry select adder is a square-root time high-speed adder. In this paper, FPGA-based synthesis of conventional and hybrid carry select adders are described with a focus on high speed. Conventionally, carry select adders are realized using the following: (i) full adders and 2 : 1 multiplexers, (ii) full adders, binary to excess 1 code converters, and 2 : 1 multiplexers, and (iii) sharing of common Boolean logic. On the other hand, hybrid carry select adders involve a combination of carry select and carry lookahead adders with/without the use of binary to excess 1 code converters. In this work, two new hybrid carry select adders are proposed involving the carry select and section-carry based carry lookahead subadders with/without binary to excess 1 converters. Seven different carry select adders were implemented in Verilog HDL and their performances were analyzed under two scenarios, dual-operand addition and multioperand addition, where individual operands are of sizes 32 and 64-bits. In the case of dual-operand additions, the hybrid carry select adder comprising the proposed carry select and section-carry based carry lookahead configurations is the fastest. With respect to multioperand additions, the hybrid carry select adder containing the carry select and conventional carry lookahead or section-carry based carry lookahead structures produce similar optimized performance.

1. Introduction

Carry select adder (CSLA) belongs to the family of high-speed square-root time adders [1, 2] and provides a good compromise between the low area occupancy of ripple carry adders (RCAs) and the high-speed performance of carry lookahead adders (CLAs) [2, 3]. In the existing literature, many flavors of carry select addition have been realized on both ASIC and FPGA platforms with ASIC implementations being predominant. CSLAs usually involve duplication of RCA structures with presumed carry inputs of binary 0 and binary 1 to enable parallel addition and thereby speed up the addition process [3, 4]. To minimize the area metric of CSLAs owing to replication of the RCA structures, an add-one circuit (also called, binary to excess-1 converter, viz. BEC) is introduced [5–7]. Carry select addition can also be performed by utilizing the common Boolean logic (CBL) [8] shared between the sum and carry outputs of a full adder [9]. Nevertheless, due to the serial cascading of full adder modules, the delay metric would not decrease although the area parameter would reduce. Further, optimizations at the device, gate levels [10–15], and realization styles [16, 17] have been carried out to reduce area, improve speed, and minimize the power-delay product of CSLAs on the basis of semicustom and full-custom ASIC-style synthesis. Rather than realizing pure CSLAs, hybrid architectures incorporating carry select and carry lookahead structures have also been proposed [18–21] to improve the design efficiency of CSLAs. Moreover, some FPGA implementations of CSLAs have been attempted [21–23]. Overall, a survey of published literature reveals that CSLAs have been widely implemented using the following topologies and computational elements:

(i) (Conventional) CSLA – full adders and 2 : 1 multiplexers (MUXes)

(ii) CSLA with BEC – Full adders, BECs, and 2 : 1 MUXes

(iii) CSLA based on CBL sharing

(iv) Hybrid CSLA and CLA structures

(v) Hybrid CSLA and CLA including BECs.

In general, CSLAs are composed using a carry select architecture with/without BECs or may consist of a mix of carry select and carry lookahead configurations with/without BECs. CSLAs constructed using pure carry select structures are called "homogeneous CSLAs" and CSLAs realized using a combination of carry select and carry lookahead structures are labeled as "heterogeneous/hybrid CSLAs." The interest behind hybrid CSLAs is supported by the fact that heterogeneous adders tend to better optimize the design metrics compared to homogeneous adders [24]. In a recent work [25], section-carry based CLAs (SCBCLAs) were proposed as an alternative to conventional CLAs; for a 32-bit addition operation, the SCBCLA was found to exhibit reduced propagation delay than the conventional CLA by 15.2%. Motivated by this result, two new hybrid CSLA architectures are proposed in this work, a hybrid CSLA incorporating CSLA and SCBCLA and another hybrid CSLA embedding CSLA, SCBCLA, and BECs. This paper builds upon our prior work [21] by analyzing the performance of different CSLA architectures with respect to diverse input partitions for different addition widths for the case of dual-operand addition and further evaluates the efficacy of the conventional and proposed CSLAs with respect to multioperand additions.

The remaining part of this paper is organized as follows. With 8-bit addition as a running example, Section 2 describes the conventional CSLA topologies with and without BEC logic and also the CSLA based on sharing of CBL. Section 3 presents the architectures of hybrid CSLAs incorporating CLAs and SCBCLAs with/without BEC logic. In Section 4, the performance of different CSLA topologies is evaluated for dual-operand and multioperand additions with operand sizes of 32 and 64-bits. Finally, the conclusions follow in Section 5.

2. Homogeneous CSLA Architectures

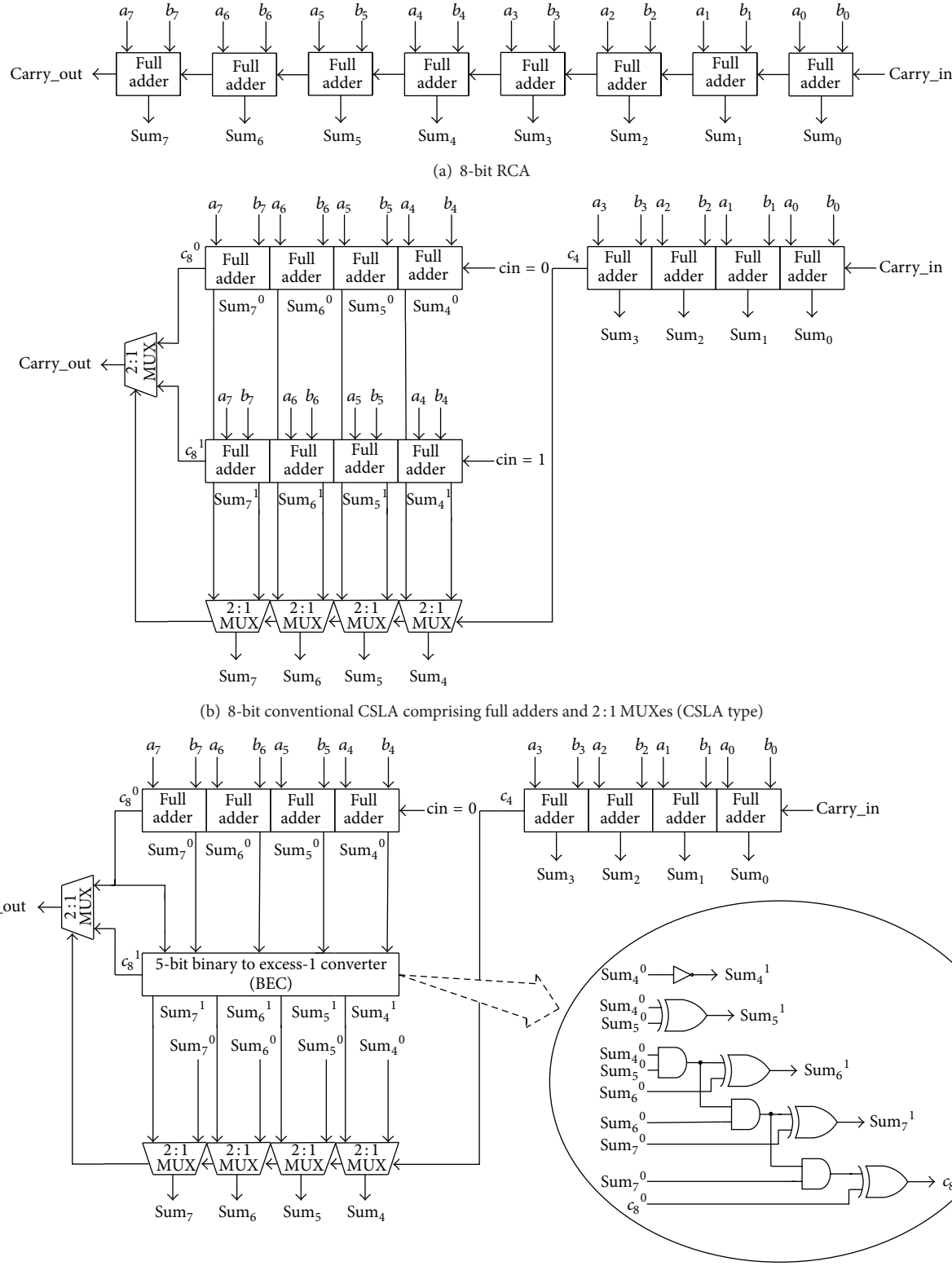
The RCA and homogeneous CSLA architectures are shown in Figure 1 for an example case of 8-bit addition. Figure 1(a) depicts an 8-bit RCA, which is formed by a cascade of full adder modules; the full adder [9] is an arithmetic building block that adds an augend and addend bit (say, a and b) along with any carry input (cin) and produces two outputs, namely, sum (Sum) and carry overflow (Cout). Since there is a rippling of carry from one full adder stage to another, the propagation delay of the RCA varies linearly in proportion to the adder width. The CSLA basically partitions the input data into groups and addition within the groups is carried out in parallel; that is, the CSLA is composed of partitioned and duplicated RCAs. It can be seen from Figure 1 that the least significant 4-bit adder stages of RCA and CSLAs are identical. However, the carry produced by the least significant nibble is simply propagated through the more significant nibble in the case of the RCA bit-by-bit, while the carry corresponding to the least significant nibble serves as the selection input for MUXes present in the more significant position in the case of CSLAs.

Figure 1(b) shows the 8-bit conventional CSLA comprising full adders and 2 : 1 MUXes, henceforth referred to as simply "CSLA." In the case of CSLA shown in Figure 1(b), the full adders present in the most significant nibble position are duplicated with carry inputs (cin) of 0 and 1 assumed; that is, one 4-bit RCA with a carry input ("cin") of 0 and another 4-bit RCA with a carry input ("cin") of 1 are used. Notice that both these RCAs have the same augend and addend inputs. While the least significant 4-bit RCA would be adding the augend inputs (a_3 to a_0) with the addend inputs (b_3 to b_0), the more significant 4-bit RCAs would be simultaneously adding up the augend inputs (a_7 to a_4) with the addend inputs (b_7 to b_4), with presumed carry inputs (cin) of 0 and 1. Due to two addition sets, two sets of sum and carry outputs are produced, one based on 0 as the carry input and another based on 1 as the carry input, which are in turn fed as inputs to the 2 : 1 MUXes. The number of MUXes used depends on the size of the RCA duplicated. To determine the true sum outputs and the real value of carry overflow pertaining to the most significant nibble position, the carry output (c_4) from the least significant 4-bit RCA is used as the common select input for all the MUXes; thereby the correct result corresponding to either the RCA with 0 as the carry input or the RCA with 1 as the carry input is displayed as output.

Figure 1(c) portrays the 8-bit CSLA containing full adders, 2 : 1 MUXes, and BEC logic, henceforth identified as "CSLA_BEC". Figure 1(c) also shows the internals of the 5-bit BEC, which is depicted by the circuit shown within the oval. The CSLA_BEC is rather different from the CSLA in that instead of having an RCA with a presumed carry input of 1 in the more significant nibble position, the BEC circuit is introduced. The BEC logic adds binary 1 to the least significant bit of its binary inputs and produces the resultant sum and carry as output. As seen in Figure 1(c), the BEC accepts as inputs the sum and carry outputs of the RCA having a presumed carry input of 0, adds binary 1 to this input, and produces the resulting sum and carry overflow as output. Now the correct result exists between choosing the output of the RCA with a presumed carry input of 0 and the output of the BEC logic. The carry output c_4 of the least significant RCA is used to determine the correct set of the most significant nibble position sum and carry outputs. The logic equations governing the 5-bit BEC are given below. In the equations, $\sim$ signifies logical inversion, $\oplus$ implies logical XOR, and $\bullet$ represents logical conjunction. Consider

$$\begin{aligned}
\mathrm{Sum}_4^{\,1} &= \sim \mathrm{Sum}_4^{\,0} \\
\mathrm{Sum}_5^{\,1} &= \mathrm{Sum}_5^{\,0} \oplus \mathrm{Sum}_4^{\,0} \\
\mathrm{Sum}_6^{\,1} &= \mathrm{Sum}_6^{\,0} \oplus \left(\mathrm{Sum}_5^{\,0} \bullet \mathrm{Sum}_4^{\,0}\right) \\
\mathrm{Sum}_7^{\,1} &= \mathrm{Sum}_7^{\,0} \oplus \left(\mathrm{Sum}_6^{\,0} \bullet \mathrm{Sum}_5^{\,0} \bullet \mathrm{Sum}_4^{\,0}\right) \\
c_8^{\,1} &= c_8^{\,0} \oplus \left(\mathrm{Sum}_7^{\,0} \bullet \mathrm{Sum}_6^{\,0} \bullet \mathrm{Sum}_5^{\,0} \bullet \mathrm{Sum}_4^{\,0}\right).
\end{aligned} \tag{1}$$

The CSLA constructed on the basis of sharing of CBL is depicted through Figure 2, which will be referred to as "CSLA_CBL" henceforth. The CSLA_CBL adder is founded



(a) 8-bit RCA

(b) 8-bit conventional CSLA comprising full adders and 2 : 1 MUXes (CSLA type)

(c) 8-bit conventional CSLA comprising full adders, 2 : 1 MUXes, and BEC logic (CSLA_BEC type)

Figure 1: (a) 8-bit RCA, (b) representative 8-bit homogeneous CSLA, and (c) representative 8-bit homogeneous CSLA with BEC logic.

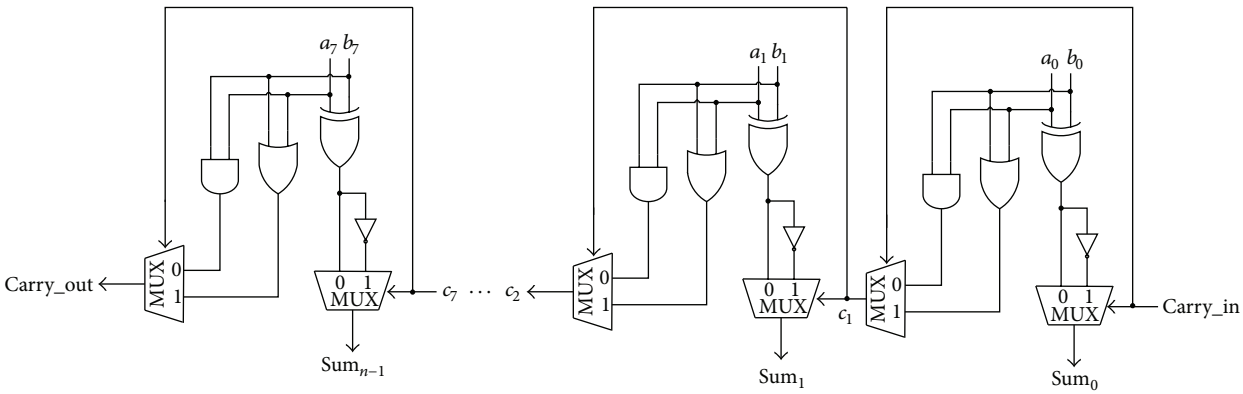


Figure 2: 8-bit homogeneous CSLA utilizing shared CBL (CSLA_CBL architecture).

upon utilizing the full adder logic, whose underlying equations are given below with a, b, and cin being the primary inputs, and Sum and Cout being the primary outputs. In (3), "+" implies logical disjunction:

$$\text{Sum} = a \oplus b \oplus \text{cin} \tag{2}$$

$$\text{Cout} = (a + b) \bullet \text{cin} + (a \bullet b)\,(\sim \text{cin})\,. \tag{3}$$

Referring to (2) and (3), it may be understood that, for a carry input (cin) of 0, (2) and (3) reduce to Sum $= (a \oplus b)$ and Cout $= (a \bullet b)$. With cin $= 1$, (2) and (3) become Sum $=\sim (a \oplus b)$ and Cout $= (a + b)$. Based on this principle, sum and carry outputs for both possible values of input carries are generated simultaneously and fed as inputs to two 2 : 1 MUXes. The correct sum and carry outputs are determined by the carry input, serving as the select input for the two MUXes. Though the exorbitant duplicated RCA and RCA with BEC logic structures are eliminated through this approach, leading to savings in terms of area, nevertheless, since the carry propagates from stage-to-stage, the critical data path delay tends to be proportional to the size of the full adders cascade. As a consequence, the delay of the CSLA_CBL adder may be close to that of the RCA which is confirmed by the simulation results given in Section 4.

3. Heterogeneous/Hybrid CSLA Architectures

Apart from synthesizing basic CSLA topologies viz. CSLA, CSLA_BEC, and CSLA_CBL, hybrid CSLA architectures involving CSLA and CLA/SCBCLA were also implemented with the intention of minimizing the maximum propagation path delay. It is well known that a CLA is faster than a RCA, and hence it may be worthwhile to have a CLA as a replacement for the least significant RCA in the CSLA structure. Although the concept of carry lookahead is widely understood, the concept of section-carry based carry lookahead may not be that well known, and hence to explain the distinction between the two, sample 4-bit lookahead logic realized using these two approaches is portrayed in Figure 3 for an illustration. For details on different section-carry based carry lookahead structures and SCBCLA constructions using them, an avid reader is directed to references [25–27], which constitute prior works in the realm of synchronous and asynchronous designs.

The section-carry based carry lookahead generator shown enclosed within the circle in Figure 3 produces a single lookahead carry signal corresponding to a "section" or "group" of the adder inputs (hence the term "section-carry"), while the conventional carry lookahead generator encapsulated within the rectangle produces multiple lookahead carry signals corresponding to each pair of augend and addend primary inputs. The section-carry based carry lookahead generator differs from the traditional carry lookahead generator in that bit-wise lookahead carry signals are not required to be computed for the former. The XOR and AND gates used for producing the necessary propagate and generate signals (P_3 to P_0 and G_3 to G_0) are highlighted using dotted lines in Figure 3; these constitute the propagate-generate logic referred to in Figures 4 and 5.

8-bit hybrid CSLAs with/without BEC logic and comprising a CLA in the least significant stage viz. "CSLA-CLA" and "CSLA_BEC-CLA" adder types are shown in Figure 4. On the other hand, 8-bit hybrid CSLAs with/without BEC logic and incorporating a SCBCLA in the least significant stage viz. "CSLA-SCBCLA" and "CSLA_BEC-SCBCLA" adder varieties are portrayed in Figure 5. Both the conventional CLA and SCBCLA constitute three functional blocks: propagate-generate logic, lookahead carry generator, and the sum producing logic. Not only is the carry lookahead generator different for CLA and SCBCLA adders, but the sum producing logic is also different; in case of CLA, the sum producing logic comprises only XOR gates, whereas in the SCBCLA, the sum producing logic consists of full adders and an XOR gate, with the XOR gate providing the sum of the primary inputs a_3, b_3, and c_3. While rippling of carries occurs internally within the carry-propagate adder constituting the SCBCLA and producing the requisite sums, the lookahead carry signal corresponding to an adder section is generated independently (in parallel) and serves as the lookahead carry input for the successive CSLA stage.

4. Results and Discussion

Three homogeneous CSLA architectures viz. CSLA, CSLA_BEC, and CSLA_CBL and four heterogeneous CSLA architectures

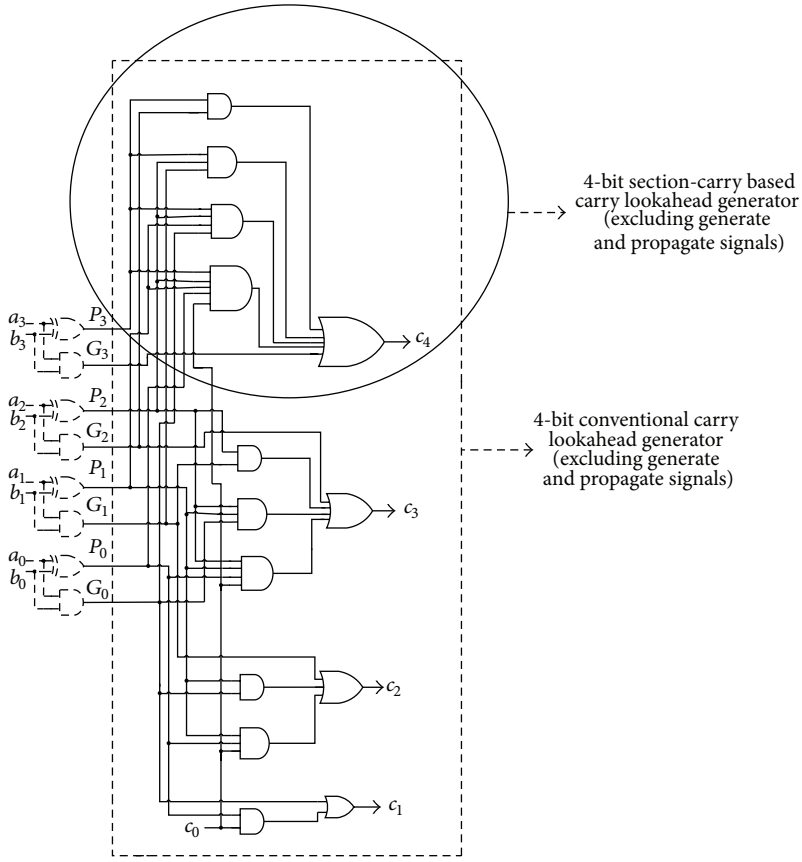
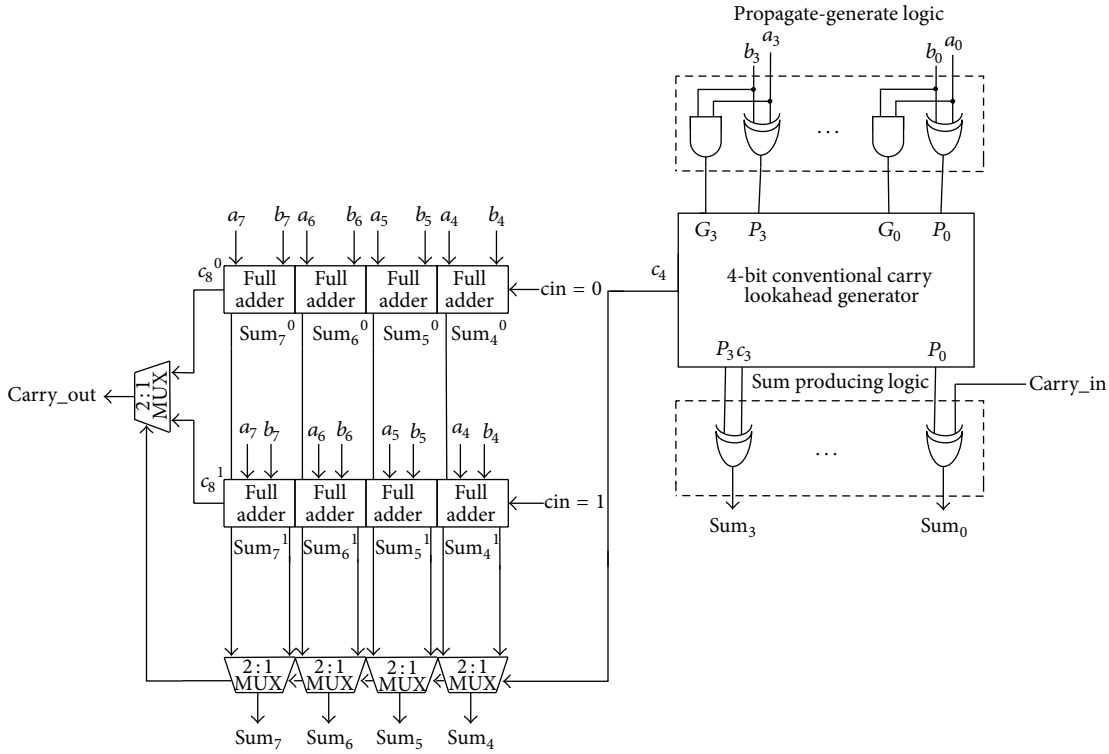


Figure 3: 4-bit conventional and section-carry based carry lookahead generators.

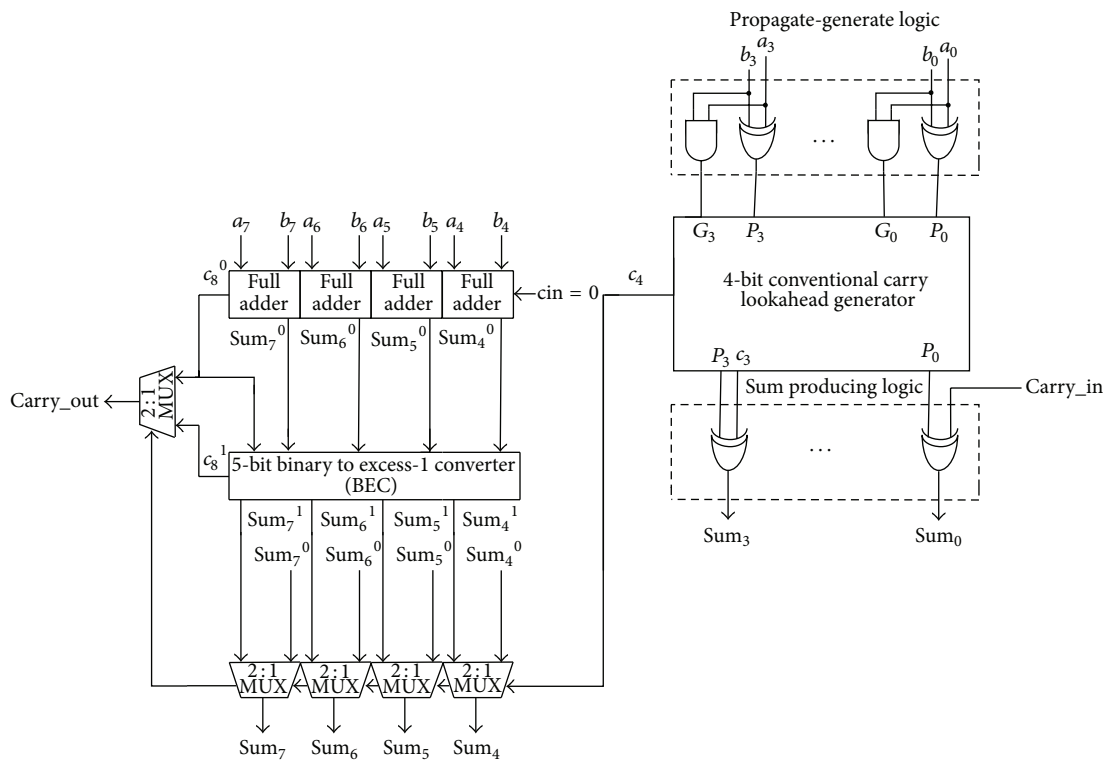
viz. CSLA-CLA, CSLA_BEC-CLA, CSLA-SCBCLA, and CSLA_BEC-SCBCLA were described topologically in Verilog HDL similar to previous works [16, 21–23, 25] to perform two kinds of addition operations viz. dual-operand addition and multioperand addition. For dual-operand addition, two binary operands having corresponding sizes of 32-bits and 64-bits were considered. For multioperand addition, addition of four binary operands, each of size 32-bits, and another multioperand addition involving four binary operands with each having size of 64-bits were considered. Moreover, two types of multioperand additions were performed based on (i) carry save adder (CSA) topology, and (ii) bit-partitioned addition scheme. All the adders were synthesized using a 90 nm FPGA (XC3S1600E) [28], with speed optimization specified as the design goal in the Xilinx 9.1i ISE design suite. The critical path delay and area values (in terms of number of basic logic elements viz. BELs) were ascertained after automatic place-and-route. The results of dual-operand additions shall be presented first, followed by the results obtained for multioperand additions.

4.1. Dual-Operand Addition. CSLAs can be implemented on the basis of uniform or nonuniform primary input partitions; accordingly they are labeled as "uniform" or "non-uniform" CSLAs, in a structural sense. "Input partitioning" basically means splitting up of the primary inputs into groups of inputs so as to pave the way for addition to be done in parallel within the partitions; it should be noted that input partitioning is inherent to all CSLAs except the CSLA_CBL type (shown in Figure 2) which has a regular carry select structure and hence is void of input partitions. Referring to Figure 1(b), it can be seen that 8 pairs of inputs have been split into two uniform or equal-sized groups of 4-input pairs; thus it can be said that the 8-bit CSLA is realized according to a 4-4 input partition.

For synthesis, 3 uniform input partitions (4-4-4-4-4-4-4-4, 8-8-8-8, and 16-16) and 2 optimum nonuniform input partitions (3-7-6-5-4-3-2-2 [29] and 8-7-6-4-3-2-2 [15]) were considered for realizing the 32-bit CSLAs. Figure 6 visually portrays the variations in propagation delay corresponding to different primary input partitions for the six CSLA types. On the other hand, 4 uniform input partitions viz. 4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4, 8-8-8-8-8-8-8-8, and 16-16-16-16, 32-32, and a nonuniform input partition viz. 8-10-9-8-7-6-5-4-3-2-2 [29] were considered for realizing the 64-bit CSLAs. Figure 7 depicts the propagation delay variations subject to different primary input partitions for the six CSLA architectures. The trend line highlighted in Figure 6 shows that the uniform 8-8-8-8 input partition consistently paves the way for least propagation delay (varying from 17 ns to 20 ns) with respect to various 32-bit homogeneous and heterogeneous CSLAs. Similarly the trend line indicated in black in Figure 7 conveys that the uniform 16-16-16-16 input partition results in the least data path delay (varying from 27 ns to 29 ns) for the different homogeneous and heterogeneous 64-bit CSLAs.

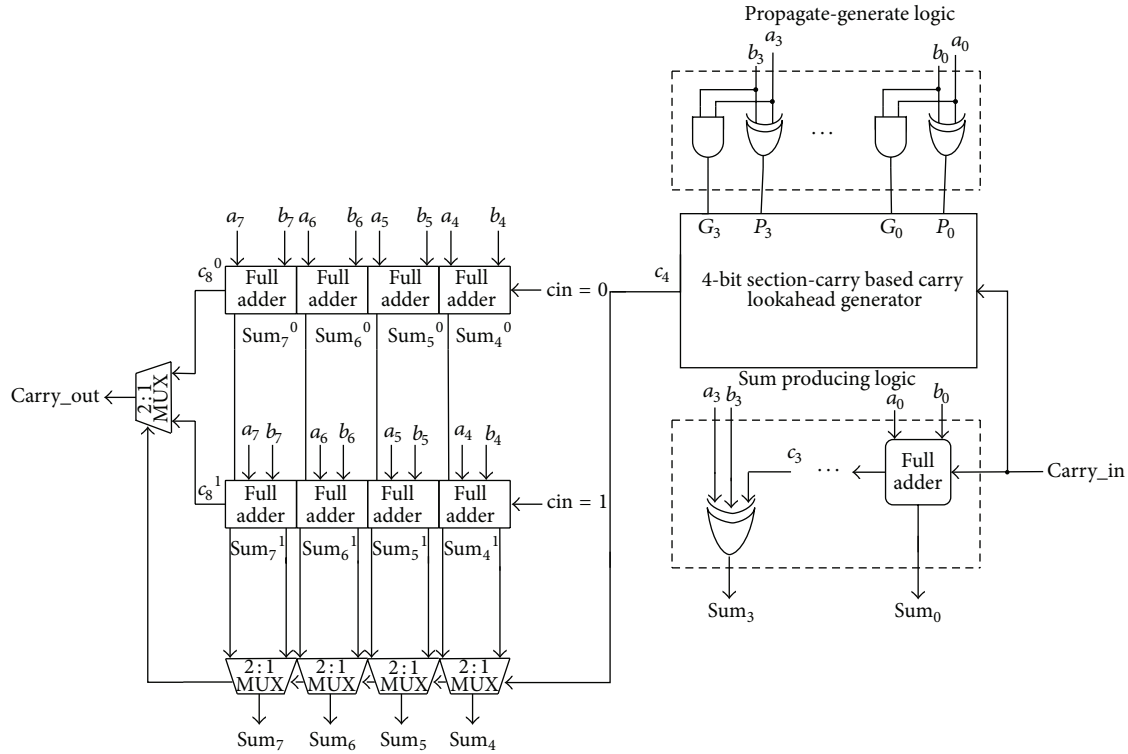


(a) 8-bit hybrid CSLA with a conventional CLA in the least significant stage

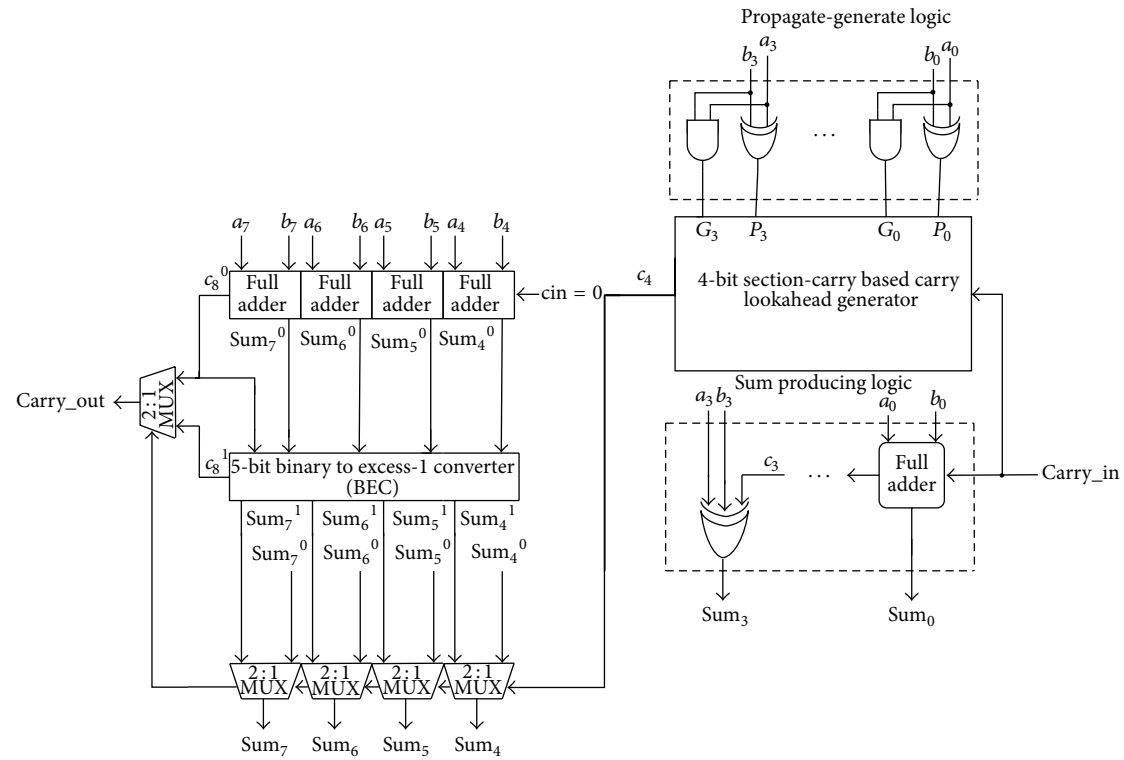


(b) 8-bit hybrid CSLA featuring BEC with a least significant CLA stage

FIGURE 4: Hybrid CSLAs without/with BEC logic comprising a CLA: (a) CSLA-CLA type and (b) CSLA_BEC-CLA type.



(a) 8-bit hybrid CSLA with 4-bit SCBCLA in the least significant stage



(b) 8-bit hybrid CSLA incorporating BEC with a least significant SCBCLA stage

Figure 5: Hybrid CSLAs without/with BEC logic comprising a SCBCLA: (a) CSLA-SCBCLA type and (b) CSLA_BEC-SCBCLA type.

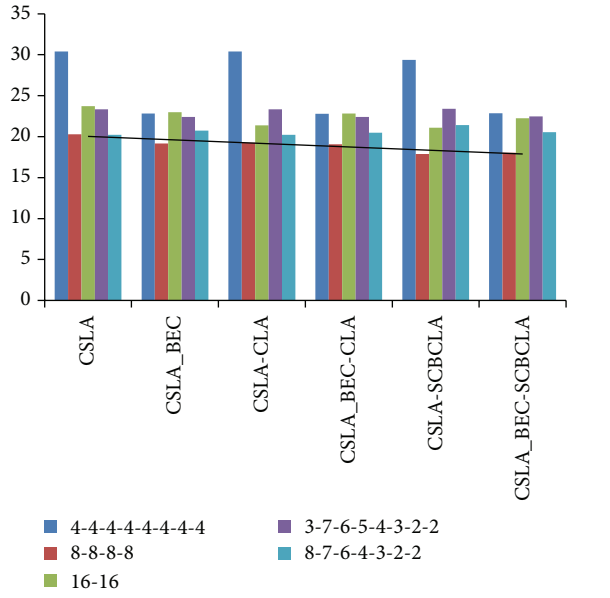


Figure 6: Capturing worst-case delay variations of 32-bit homogeneous and heterogeneous CSLAs for different input partitions. *X*-axis: CSLA type; *Y*-axis: Delay in ns.

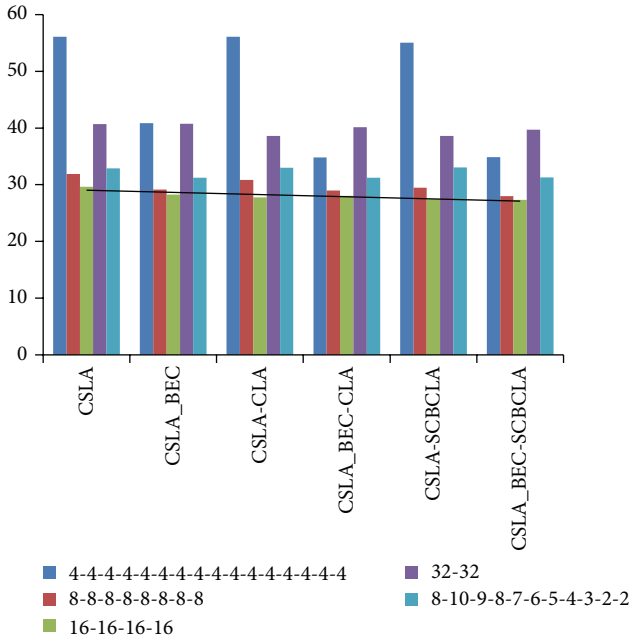


Figure 7: Portraying critical path delay variations of 64-bit homogeneous and heterogeneous CSLAs for different input partitions. *X*-axis: CSLA type; *Y*-axis: Delay in ns.

Table 1: Maximum propagation delay and area (# BELs) of 32-bit homogeneous and heterogeneous CSLAs corresponding to diverse input partitions.

Input partition	Type of CSLA architecture	Critical path delay (ns)	Area (# BELs)
Not applicable	RCA	30.604	**63**
Not applicable	CSLA_CBL	37.604	**63**
4-4-4-4-4-4-4-4	CSLA	30.388	105
	CSLA_BEC	22.820	106
	CSLA-CLA	30.398	106
	CSLA_BEC-CLA	22.781	106
	CSLA-SCBCLA*	29.359	108
	CSLA_BEC-SCBCLA*	22.864	108
8-8-8-8	CSLA	20.280	117
	CSLA_BEC	19.176	104
	CSLA-CLA	19.260	121
	CSLA_BEC-CLA	19.059	104
	CSLA-SCBCLA*	**17.897**	123
	CSLA_BEC-SCBCLA*	18.052	110
16-16	CSLA	23.722	105
	CSLA_BEC	22.986	91
	CSLA-CLA	21.384	114
	CSLA_BEC-CLA	22.835	91
	CSLA-SCBCLA*	21.097	119
	CSLA_BEC-SCBCLA*	22.255	106
3-7-6-5-4-3-2-2	CSLA	23.337	110
	CSLA_BEC	22.411	108
	CSLA-CLA	23.337	110
	CSLA_BEC-CLA	22.411	108
	CSLA-SCBCLA*	23.408	110
	CSLA_BEC-SCBCLA*	22.482	108
8-7-6-4-3-2-2	CSLA	20.218	118
	CSLA_BEC	20.743	111
	CSLA-CLA	20.218	118
	CSLA_BEC-CLA	20.473	111
	CSLA-SCBCLA*	21.403	117
	CSLA_BEC-SCBCLA*	20.544	111

The maximum combinational path delay (also called, "critical path delay") encountered and the total number of BELs consumed by different homogeneous and heterogeneous CSLAs to perform the addition of two 32-bit operands and two 64-bit operands separately is shown in Tables 1 and 2, respectively. The optimum delay and area values are in bold font in the tables. Note that the symbol ∗ signifies the proposed hybrid CSLA architectures in the tables.

From Table 1, it is evident that the CSLA-SCBCLA hybrid adder based on the 8-8-8-8 input partition features the least propagation delay (17.897 ns) amongst all homogeneous and hybrid CSLAs, and hence the 8-8-8-8 input partition is deemed to be optimum. The 32-bit RCA has critical path delay of 30.604 ns, while the 32-bit CSLA_CBL adder is

found to have the longest path delay of 37.604 ns. Compared to the maximum delay of the hybrid CSLA-SCBCLA, the hybrid CSLA_BEC-SCBCLA adder which is another proposed hybrid CSLA topology has a comparable speed performance of 18.052 ns. However with respect to area, the RCA and CSLA_CBL structures require less number of BELs than all the CSLAs. Hence it is inferred from Figure 6 and Table 1 that for the addition of two input operands having sizes of 32-bits the hybrid CSLA-SCBCLA adder is preferable over all other homogeneous and heterogeneous CSLAs and the favorable input data partition is 8-8-8-8.

Based on a similar observation, by referring to Figure 7 and Table 2, it can be seen that the 16-16-16-16 input partition is found to be optimum from a delay (i.e., speed) perspective for 64-bit dual-operand addition. The proposed CSLA_BEC-SCBCLA constructed using the 16-16-16-16 input data partition leads to the least latency amongst all other adder topologies; however, the other proposed CSLA viz. CSLA-SCBCLA based on a similar input partition features almost a similar delay metric. In terms of area occupancy though, the 64-bit RCA is optimized. Nevertheless, the RCA encounters considerably more data path delay by 1.6× in comparison with the proposed CSLA_BEC-SCBCLA based on a 16-16-16-16 input partition.

4.2. Multioperand Addition. The performance of different homogeneous and heterogeneous CSLAs is evaluated based on the case studies of multioperand addition involving 4 binary operands, with respective sizes of 32-bits and 64-bits. Two multioperand addition schemes are considered, one involving the carry save adder (CSA) topology, and another involving the bit-partitioning method.

4.2.1. CSA Based Multioperand Addition. The structure of an example CSA used to add four n-bit binary numbers is shown in Figure 8. Here, a_{n-1} to a_0, b_{n-1} to b_0, c_{n-1} to c_0, and d_{n-1} to d_0 represent the primary inputs and the sum bits and Sum_{n+1} to Sum_0 represents the primary outputs. The subscript 0 denotes the LSB and the subscript $(n-1)$ denotes the MSB. As shown in Figure 8, there are three adders in three levels to perform the addition of four input operands. In each CSA, the carry output signal of the current bit at a level is not transferred to the next bit adder of the same level as the carry input; instead, the carry output is transferred to the next bit adder in the lower level as the carry input. In the top-level adder, three numbers (a, b, and c) are added simultaneously; that is, the bits corresponding to any number could act as input carries for the full adders of the first level CSA. In the next lower level, an extra number (d) is added. The adder in the bottom level, shown within the ellipse in Figure 8, is a simple RCA which is what portrayed here but it may be any dual-operand adder that can be used to compute the final sum.

Experimentation was performed by having different dual-operand adders viz. RCA and various homogeneous and heterogeneous CSLAs in the final adder stage of the CSA, shown in Figure 8, to analyze their relative performance for two different addition scenarios: (i) addition of four binary operands, each of size 32-bits, and (ii) addition of four binary operands with each having size of 64-bits.

Table 2: Maximum propagation delay and area (# BELs) of 64-bit homogeneous and heterogeneous CSLAs corresponding to different input partitions.

Input partition	Type of CSLA architecture	Critical path delay (ns)	Area (# BELs)
Not applicable	RCA	71.555	**127**
Not applicable	CSLA_CBL	70.525	129
4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4	CSLA	56.091	217
	CSLA_BEC	40.870	209
	CSLA-CLA	56.101	218
	CSLA_BEC-CLA	34.799	215
	CSLA-SCBCLA*	55.062	220
	CSLA_BEC-SCBCLA*	34.882	217
8-8-8-8-8-8-8-8	CSLA	31.866	251
	CSLA_BEC	29.119	224
	CSLA-CLA	30.846	255
	CSLA_BEC-CLA	29.002	224
	CSLA-SCBCLA*	29.483	257
	CSLA_BEC-SCBCLA*	27.995	230
16-16-16-16	CSLA	29.625	252
	CSLA_BEC	28.259	212
	CSLA-CLA	27.759	261
	CSLA_BEC-CLA	28.029	213
	CSLA-SCBCLA*	27.427	266
	CSLA_BEC-SCBCLA*	**27.322**	227
32-32	CSLA	40.705	217
	CSLA_BEC	40.742	189
	CSLA-CLA	38.591	215
	CSLA_BEC-CLA	40.157	189
	CSLA-SCBCLA*	38.591	247
	CSLA_BEC-SCBCLA*	39.682	219
8-10-9-8-7-6-5-4-3-2-2	CSLA	32.983	251
	CSLA_BEC	31.204	226
	CSLA-CLA	32.983	251
	CSLA_BEC-CLA	31.204	226
	CSLA-SCBCLA*	33.054	251
	CSLA_BEC-SCBCLA*	31.276	226

The FPGA-based synthesis results viz. delay and area obtained for the addition of four binary operands, each having size of 32-bits, are given in Table 3 with the optimized values in bold font. Since the 8-8-8-8 primary input partition was found to yield the least data path delay, as evident from Figure 6 and Table 1, it was preferred for the various CSLA realizations. It can be seen from Table 3 that the hybrid CSLA_BEC-CLA when used in the final adder stage of the CSA encounters the least propagation delay, with

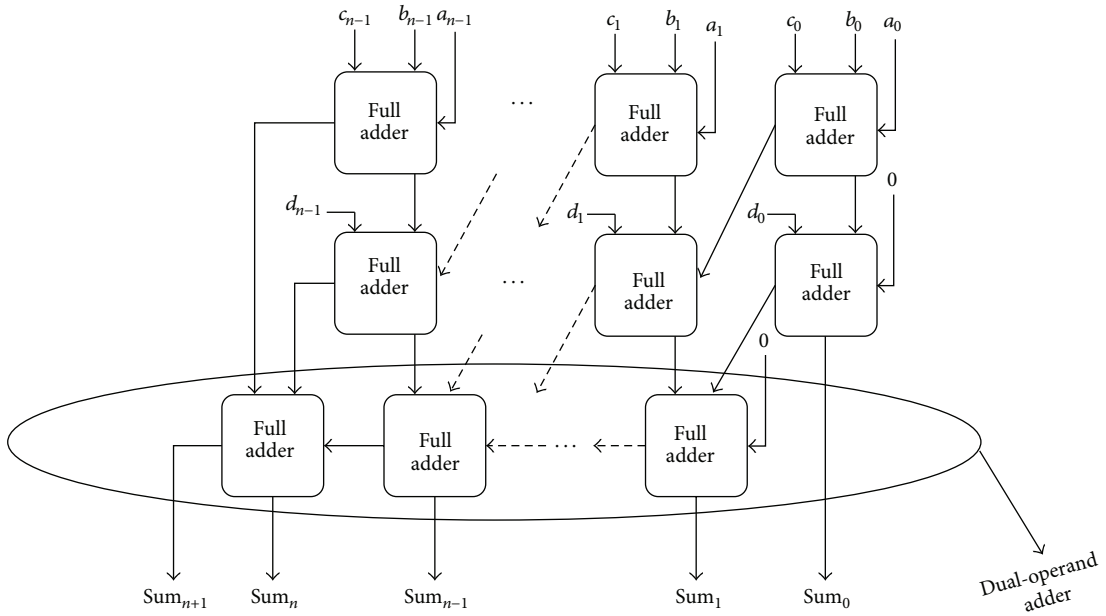


Figure 8: CSA topology for addition of four n-bit binary operands.

Table 3: Critical path delay and area figures for CSA-based multioperand addition of four 32-bit operands, with RCA/homogeneous/heterogeneous CSLAs used in the final adder stage.

Input partition	Type of adder architecture	Critical path delay (ns)	Area (# BELs)
Not applicable	RCA	39.842	**190**
Not applicable	CSLA_CBL	39.842	**190**
8-8-8-8	CSLA	27.383	229
	CSLA_BEC	22.455	229
	CSLA-CLA	25.053	229
	CSLA_BEC-CLA	**21.326**	232
	CSLA-SCBCLA*	23.378	227
	CSLA_BEC-SCBCLA*	21.684	233

Table 4: Critical path delay and area for CSA-based multioperand addition of four 64-bit operands, with RCA/homogeneous/heterogeneous CSLAs used in the final adder stage.

Input partition	Type of adder architecture	Critical path delay (ns)	Area (# BELs)
Not applicable	RCA	73.792	**382**
Not applicable	CSLA_CBL	71.667	383
16-16-16-16	CSLA	37.034	472
	CSLA_BEC	31.307	462
	CSLA-CLA	33.363	476
	CSLA_BEC-CLA	**30.428**	471
	CSLA-SCBCLA*	32.008	473
	CSLA_BEC-SCBCLA*	30.732	470

the proposed CSLA_BEC-SCBCLA adder closely following it with just a 1.7% delay difference. The conventional CLA, when used in the final adder stage of the CSA as a "homogeneous adder," reports a critical path delay of 34.306 ns. On the contrary, when the conventional CLA is used along with the CSLA inclusive of the BEC as a "heterogeneous adder" (CSLA_BEC-CLA), it enables considerable decrease in maximum data path delay by 37.8% vindicating the observation made in [24] that heterogeneous adders are preferable over homogeneous adders for delay optimization. Although the use of RCA and CSLA_CBL adders in the final adder stage of the CSA helps to minimize the area occupancy compared to their counterparts, they suffer from an exacerbated increase in delay of about 87% over the CSLA_BEC-CLA type.

The synthesis results obtained for the addition of four binary operands, each having sizes of 64-bits, is shown in Table 4 and the optimized values are in bold font. Since the 16-16-16-16 uniform input partition was found to be delay optimal (refer to Figure 7 and Table 2), it was adopted for implementing all the CSLAs. Again, the CSLA_BEC-CLA variant reports the least propagation delay compared to others as in the previous case, with the proposed CSLA_BEC-SCBCLA reporting almost a similar performance. However due to less logic complexity, the usage of RCA or CSLA_CBL in the final adder stage of the CSA results in the least area occupancy in comparison with the rest, albeit at the expense of a considerable increase in delay by about 1.4x.

4.2.2. Bit-Partitioned Multioperand Addition. In CSAs, row-wise parallel addition is performed where the tree height (i.e., number of adder levels) grows with an increase in

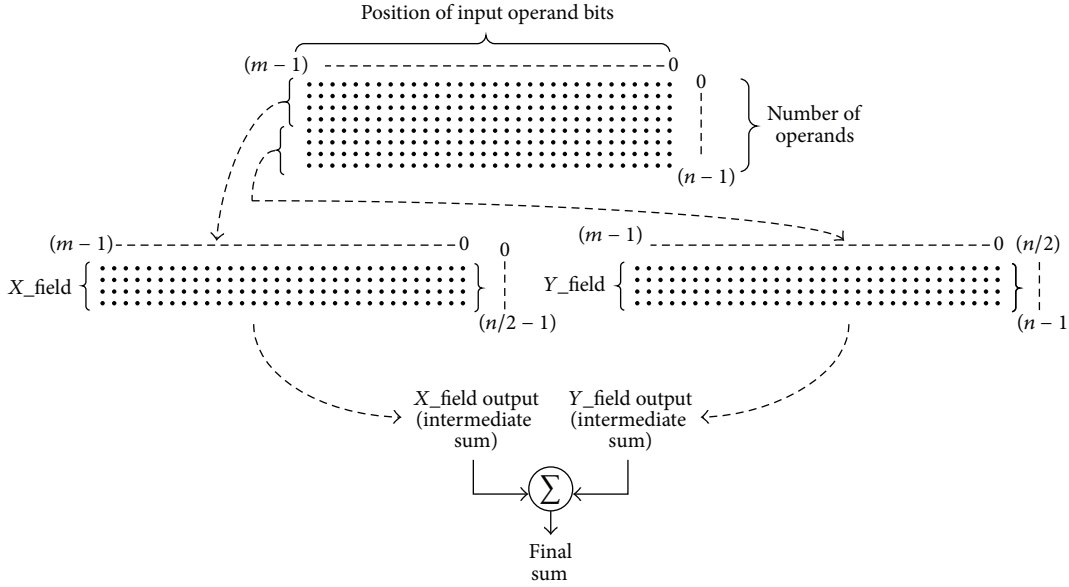


Figure 9: Bit-partitioned multioperand addition scheme.

the number of input operands by an approximate linear order. To reduce the logic depth of the adder tree, a bit-partitioning strategy was presented in [30] in the context of self-timed multioperand addition, which involved splitting up of the entire group of data operands into a desired number of subgroups, and the intermediate addition results of the subgroups are finally added to produce the final sum. The bit-partitioning approach basically parallelizes the multioperand addition and is illustrated through Figure 9 for an example scenario where addition of "n" binary operands with each operand having a size of "m" bits is considered whilst assuming "n" to be even. A "dot" represents a bit position in Figure 9.

The entire set of input operands from bit position 0 to bit position $(n-1)$ is divided into two equal-sized groups (for an example) as X_field, which comprises inputs from bit positions 0 to $(n/2-1)$ and the Y_field consisting of inputs from bit positions $(n/2)$ to $(n-1)$. Addition within the individual fields (i.e., X_field and Y_field) is performed simultaneously and the sum bits generated as intermediate outputs from these individual fields (X_field and Y_field) are then added together using a final dual-operand adder to produce the required sum. The bit-partitioning scheme might help to speed-up the addition, especially when several operands have to be added by way of performing parallel column-wise addition of row-wise partitions. For example, considering the addition of 32 data operands, each of size 32 bits, the CSA topology would encounter thirty full adder delays plus the delay associated with the final dual-operand adder. On the other hand, based on the bit-partitioning technique, considering eight partitions with each partition comprising four data operands, the bit-partitioned multioperand adder based upon the CSA topology could encounter a reduced propagation delay of about four full adder delays plus the delay of a dual-operand adder, depending upon the implementation. Also, a high regularity would be implicit within the overall architecture as the gate-level hardware is being duplicated.

In this work, the bit-partitioning scheme was employed to partition the set of four inputs into two input groups (X_field and Y_field, as shown in Figure 9) and the outputs of X and Y fields were then added to produce the final sum. Several dual-operand adders were used to realize the bit-partitioned addition separately viz. RCA, CSLA_CBL, CSLA, CSLA_BEC, CSLA-CLA, CSLA_BEC-CLA, CSLA-SCBCLA, and CSLA_BEC-SCBCLA. The different bit-partitioned addition structures were individually synthesized using the same FPGA (XC3S1600E). It should be noted that the focus here is only on evaluating the performance of the RCA and different CSLAs as employed for multioperand addition and not to comment upon the efficacy of the bit-partitioning scheme as such (i.e., no comparison with the results of the previous subsection). This is because, as mentioned in the preceding discussions, the bit-partitioning technique is scalable, can be custom-defined, and could potentially benefit in terms of latency reduction primarily for additions involving typically higher dimensions as compared with conventional combinational tree structures.

Table 5 presents the timing and area results obtained for the synthesis of bit-partitioned multi-input addition of 4 binary operands, each of size 32-bits, on the basis of RCA and various homogeneous and heterogeneous CSLAs. Since the 8-8-8-8 uniform input partition was found to be delay-optimum for realizing the 32-bit CSLAs (refer to Figure 6 and Table 1), only this uniform input partition has been considered for implementing the various homogeneous and hybrid CSLAs corresponding to X-field and Y_field of the bit-partitioned multioperand addition. To sum up the outputs of X-field and Y_field, a 33-bit dual-operand adder would be required in which case an extra bit has been added to the most significant position of various CSLA input partitions.

Table 5: Critical path delay and area metrics for bit-partitioned multioperand addition of four 32-bit operands, with RCA and various homogeneous/hybrid CSLA architectures used.

Input partition	Type of adder architecture	Critical path delay (ns)	Area (# BELs)
Not applicable	RCA	39.928	**190**
Not applicable	CSLA_CBL	42.241	195
8-8-8-8	CSLA	32.303	458
	CSLA_BEC	29.278	311
	CSLA-CLA	31.727	359
	CSLA_BEC-CLA	28.207	325
	CSLA-SCBCLA*	27.628	365
	CSLA_BEC-SCBCLA*	**27.056**	328

Table 6: Critical path delay and area parameters for bit-partitioned multioperand addition of four 64-bit operands, with RCA and various homogeneous/hybrid CSLA architectures used.

Input partition	Type of adder architecture	Critical path delay (ns)	Area (# BELs)
Not applicable	RCA	73.840	**382**
Not applicable	CSLA_CBL	77.946	388
16-16-16-16	CSLA	50.957	748
	CSLA_BEC	46.559	637
	CSLA-CLA	50.426	781
	CSLA_BEC-CLA	45.679	648
	CSLA-SCBCLA*	**45.608**	800
	CSLA_BEC-SCBCLA*	45.665	691

The optimum synthesis metrics obtained for the example multi-input addition are in bold font in Table 5. It can be seen that the proposed CSLA_BEC-SCBCLA paves the way for least computation time (27.056 ns) amongst all. In comparison, the undesirable increases in delay values for other bit-partitioned multioperand adders incorporating RCA, CSLA_CBL, CSLA, CSLA_BEC, CSLA-CLA, CSLA_BEC-CLA, and CSLA-SCBCLA types are found to be 47.6%, 56.1%, 15.9%, 3%, 15.9%, 3%, and 2.1%, respectively. However, the RCA results in the lowest area occupancy (190 BELs) and the CSLA_CBL adder occupies nearly the same area with just 5 more BELs. Nevertheless, the bit-partitioned multioperand adder based upon the RCA pays a 47.6% delay penalty in comparison with that utilizing the CSLA_BEC-SCBCLA.

Table 6 shows the delay and area values obtained for the synthesis of bit-partitioned addition of four input operands of sizes 64 bits, corresponding to different adder architectures, with the CSLAs utilizing the 16-16-16-16 uniform input partition since this partition was found to be delay optimal (refer to Figure 7 and Table 2). With respect to less area, the RCA is found to be the optimum architecture. However, in terms of less critical path delay, the proposed CSLA-SCBCLA benefits by achieving a good delay reduction of 38.2% compared to the maximum path delay of the RCA based bit-partitioned multioperand adder.

5. Conclusions

CSLA is an important member of the high-speed adder family. In this paper, existing CSLA architectures viz. homogeneous and heterogeneous have been described and two new hybrid CSLA topologies were put forward: (i) carry select-cum-section-carry based carry lookahead adder (CSLA-SCBCLA) and (ii) carry select-cum-section-carry based carry lookahead adder including BEC logic (CSLA_BEC-SCBCLA). The speed performances of the various CSLA structures have been analyzed based on the case studies of 32-bit and 64-bit dual-operand and multioperand additions. Both uniform and nonuniform input data partitions were considered for the various CSLA implementations and FPGA-based synthesis was performed. It has been found for dual-operand additions; the proposed CSLA-SCBCLA/CSLA_BEC-SCBCLA architecture is faster and outperforms all other homogeneous and heterogeneous CSLAs. For bit-partitioned multi-input additions, the proposed CSLA-SCBCLA/CSLA_BEC-SCBCLA architecture promises high speed. Nevertheless, for multioperand addition based on the CSA topology, the conventional CSLA_BEC-CLA and the proposed CSLA_BEC-SCBCLA architectures were found to exhibit an optimized and comparable speed performance. From the inferences derived through this work, it is likely that the proposed hybrid CSLA architectures could achieve enhanced performance over conventional CSLAs for ASIC-based synthesis as well.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors thank the constructive comments of the reviewers, especially the pointing out of some typos in the initial submitted version by a reviewer, which has helped to improve this paper's presentation.

References

[1] O. J. Bedrij, "Carry-select adder," *IRE Transactions on Electronic Computers*, vol. 11, no. 3, pp. 340–346, 1962.

[2] A. R. Omondi, *Computer Arithmetic Systems: Algorithms, Architecture and Implementation*, Prentice Hall, 1994.

[3] I. Koren, *Computer Arithmetic Algorithms*, A K Peeters/CRC Press, 2nd edition, 2001.

[4] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, New York, NY, USA, 2nd edition, 2010.

[5] T.-Y. Chang and M.-J. Hsiao, "Carry-select adder using single ripple-carry adder," *Electronics Letters*, vol. 34, no. 22, pp. 2101–2103, 1998.

[6] Y. Kim and L.-S. Kim, "64-bit carry-select adder with reduced area," *Electronics Letters*, vol. 37, no. 10, pp. 614–615, 2001.

[7] B. Ramkumar and H. M. Kittur, "Low-power and area-efficient carry select adder," *IEEE Transactions on VLSI Systems*, vol. 20, no. 2, pp. 371–375, 2012.

[8] I.-C. Wey, C.-C. Ho, Y.-S. Lin, and C.-C. Peng, "An area-efficient carry select adder design by sharing the common boolean logic term," in *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS '12)*, vol. 2, pp. 1091–1094, March 2012.

[9] P. Balasubramanian and N. E. Mastorakis, "High speed gate level synchronous full adder designs," *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 2, pp. 290–300, 2009.

[10] W. Jeong and K. Roy, "Robust high-performance low-power carry select adder," in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 503–506, Kitakyushu, Japan, January 2003.

[11] M. Alioto, G. Palumbo, and M. Poli, "A gate-level strategy to design carry select adders," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 465–468, IEEE, May 2004.

[12] M. Alioto, G. Palumbo, and M. Poli, "Optimized design of parallel carry-select adders," *Integration, the VLSI Journal*, vol. 44, no. 1, pp. 62–74, 2011.

[13] A. Nève, H. Schettler, T. Ludwig, and D. Flandre, "Power-delay product minimization in high-performance 64-bit carry select adders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 235–244, 2004.

[14] Y. He, C.-H. Chang, and J. Gu, "An area efficient 64-bit square root carry-select adder for low power applications," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 4, pp. 4082–4085, May 2005.

[15] B. K. Mohanty and S. K. Patel, "Area-delay-power efficient carry select adder," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 6, pp. 418–422, 2014.

[16] J. Monteiro, J. L. Güntzel, and L. Agostini, "A1CSA: an energy-efficient fast adder architecture for cell-based VLSI design," in *Proceedings of the 18th IEEE International Conference on Electronics, Circuits and Systems (ICECS '11)*, pp. 442–445, Beirut, Lebanon, December 2011.

[17] Y. Chen, H. Li, K. Roy, and C.-K. Koh, "Cascaded carry-select adder (C^2SA): a new structure for low-power CSA design," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 115–118, August 2005.

[18] Y. Wang, C. Pai, and X. Song, "The design of hybrid carry-lookahead/carry-select adders," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 1, pp. 16–24, 2002.

[19] G. A. Ruiz and M. Granda, "An area-efficient static CMOS carry-select adder based on a compact carry look-ahead unit," *Microelectronics Journal*, vol. 35, no. 12, pp. 939–944, 2004.

[20] H. G. Tamar, A. G. Tamar, K. Hadidi, A. Khoei, and P. Hoseini, "High speed area reduced 64-bit static hybrid carry-lookahead/carry-select adder," in *Proceedings of the 18th IEEE International Conference on Electronics, Circuits and Systems (ICECS' 11)*, pp. 460–463, December 2011.

[21] V. Kokilavani, P. Balasubramanian, and H. R. Arabnia, "FPGA realization of hybrid carry select-cum-section-carry based carry lookahead adders," in *Proceedings of the 12th International Conference on Embedded Systems and Applications*, pp. 81–85, 2014.

[22] R. Yousuf and Najeeb-ud-din, "Synthesis of carry select adder in 65nm FPGA," in *Proceedings of the IEEE Region 10 Conference (TENCON '08)*, pp. 1–6, November 2008.

[23] U. Sajesh Kumar and K. K. Mohamed Salih, "Efficient carry select adder design for FPGA implementation," *Procedia Engineering*, vol. 30, pp. 449–456, 2012.

[24] J.-G. Lee, J.-A. Lee, B.-S. Lee, and M. D. Ercegovac, "A design method for heterogeneous adders," in *Embedded Software and Systems*, vol. 4523 of *Lecture Notes in Computer Science*, pp. 121–132, Springer, 2007.

[25] K. Preethi and P. Balasubramanian, "FPGA implementation of synchronous section-carry based carry look-ahead adders," in *Proceedings of the IEEE 2nd International Conference on Devices, Circuits and Systems (ICDCS '14)*, pp. 1–4, IEEE, Combiatore, India, March 2014.

[26] P. Balasubramanian, D. A. Edwards, and W. B. Toms, "Self-timed section-carry based carry lookahead adders and the concept of alias logic," *Journal of Circuits, Systems and Computers*, vol. 22, no. 4, Article ID 1350028, 2013.

[27] P. Balasubramanian, D. A. Edwards, and H. R. Arabnia, "Robust asynchronous carry lookahead adders," in *Proceedings of the 11th International Conference on Computer Design*, pp. 119–124, 2011.

[28] Xilinx, http://www.xilinx.com.

[29] K. K. Parhi, "Low-energy CSMT carry generators and binary adders," *IEEE Transactions on VLSI Systems*, vol. 7, no. 4, pp. 450–462, 1999.

[30] P. Balasubramanian, D. A. Edwards, and W. B. Toms, "Self-timed multi-operand addition," *International Journal of Circuits, Systems and Signal Processing*, vol. 6, no. 1, pp. 1–11, 2012.