

Asynchronous Early Output Section-Carry Based Carry Lookahead Adder with Alias Carry Logic

P. Balasubramanian, C. Dang, D.L. Maskell, and K. Prasad

Abstract - A new asynchronous early output section-carry based carry lookahead adder (SCBCLA) with alias carry output logic is presented in this paper. To evaluate the proposed SCBCLA with alias carry logic and to make a comparison with other CLAs, a 32-bit addition operation is considered. Compared to the weak-indication SCBCLA with alias logic, the proposed early output SCBCLA with alias logic reports 13.2% reduction in area without any increases in latency and power dissipation. On the other hand, in comparison with the early output recursive CLA (RCLA), the proposed early output SCBCLA with alias logic reports 16% reduction in latency while occupying almost the same area and dissipating almost the same average power. All the asynchronous CLAs are quasi-delay-insensitive designs which incorporate the delay-insensitive dual-rail data encoding and adhere to the 4-phase return-to-zero handshaking. The adders were realized and the simulations were performed based on a 32/28nm CMOS process.

I. INTRODUCTION

Quasi-delay-insensitive (QDI) asynchronous circuits adopt an unbounded delay model for gates and wires with the exception of isochronic forks [1], which form the weakest compromise to delay-insensitivity. The signal transitions on all the isochronic forks, whether they are up-going or down-going, are assumed to happen concurrently. QDI circuits are the practically realizable delay-insensitive circuits which are robust to variations in process, supply and threshold voltages, and the operating temperature. Besides being adaptive and modular [2], QDI circuits are self-checking [3] and are naturally resistant to side channel attacks in the case of secure applications [4 – 7].

The main reasons for the robustness of QDI circuits are: i) delay-insensitive encoding for binary data representation and processing, and ii) adoption of a 4-phase handshake protocol for data communication. For delay-insensitive data encoding, the dual-rail or 1-of-2 code is common and is widely used. The dual-rail or 1-of-2 code is

the simplest member of the family of delay-insensitive m -of- n codes [8]. According to the dual-rail code, a data wire X is represented using 2 wires say, $X1$ and $X0$ as shown in Figure 1. $X = 1$ is encoded as $X1 = 1$ and $X0 = 0$, and $X = 0$ is encoded as $X1 = 0$ and $X0 = 1$. These two combinations represent the data. $X1 = X0 = 0$ is referred to as the spacer, and $X1 = X0 = 1$ is said to be invalid since the coding scheme is unordered [9]. In this work, we employ the dual-rail code for binary data encoding. The representation of 1 and 0 by respectively assigning a 1 to $X1$ and $X0$ on a mutually exclusive basis, and the usage of the zero spacer to denote the return-to-zero of all the encoded data wires defines the 4-phase return-to-zero handshake protocol.

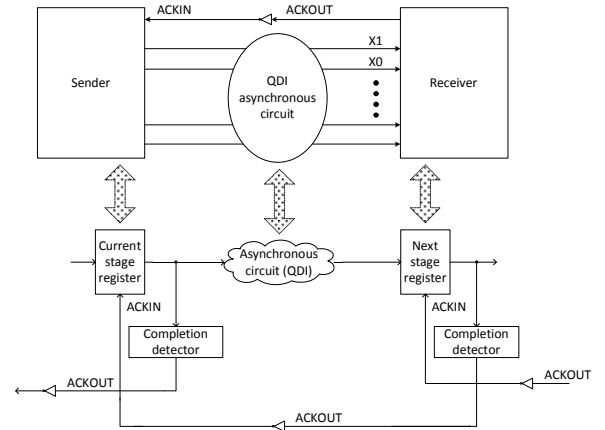


Fig. 1. A QDI asynchronous circuit stage correlated with the data sender and data receiver analogy for illustration

In general, the 4-phase handshake protocol used for data communication in a QDI asynchronous circuit may involve either an intermediate return-to-zero [10] or a return-to-one [11] phase. In this work, we consider the 4-phase return-to-zero handshaking which is explained with reference to Figure 1 through the following steps. The completion of the following four steps signifies the completion of one data transaction based on the 4-phase return-to-zero handshake protocol, which then paves the way for further data transaction(s). In short, the application of primary inputs follows the input sequence: data-spacer-data-spacer, and so forth.

- The dual-rail data bus shown in Figure 1 is initially in the spacer state and ACKIN is 1. The sender can now transmit a data which would give rise to up-

P. Balasubramanian and C. Dang are with the School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, E-mails: balasubramanian@ntu.edu.sg; hcdang@ntu.edu.sg

D.L. Maskell is with the School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, E-mail: asdouglas@ntu.edu.sg

K. Prasad is with the Department of Electrical and Electronic Engineering, Auckland University of Technology, Auckland 1142, New Zealand, E-mail: krishnamachar.prasad@aut.ac.nz

going signal transitions on any one of the corresponding dual rails of the dual-rail data bus

- Next the receiver, after receiving the data sent by the sender, would drive ACKOUT to 1
- Then the sender would wait for ACKIN to assume 0, after which it would drive the entire dual-rail data bus to the spacer state
- Finally, after an unbounded but a finite and positive time duration, the receiver would drive ACKOUT to 0 which implies ACKIN would eventually assume 1, and this signals the end of one data transaction

II. QDI ASYNCHRONOUS CIRCUIT TYPES

QDI asynchronous logic circuits are classified as strongly indicating [12], weakly indicating [12] or early output type [13]. The input-output timing relation of strong-indication, weak-indication and early output QDI asynchronous circuits is portrayed by Figure 2. A strong-indication asynchronous circuit [14] will start to process the data or spacer to produce the corresponding primary outputs only after receiving all the primary inputs. A weak-indication asynchronous circuit [15] would tend to process the data or spacer after receiving just a subset of the primary inputs and may produce all but one of the corresponding primary outputs. However, a weak-indication asynchronous circuit would produce all the primary outputs only after receiving all the primary inputs whether they be data or spacer. An early output asynchronous circuit [16] would process the data or spacer after receiving just a subset of the primary inputs and can produce all the corresponding primary outputs. Supposing an early output asynchronous circuit produces the spacer on all the primary outputs after receiving the spacer on only a subset of the primary inputs, it is said to be of early reset type. On the other hand, if an early output asynchronous circuit produces all the primary output data after receiving only a subset of the primary input data, it is said to be of early set type. The early set and reset behaviors of an early output asynchronous circuit are depicted through the portion encapsulated within the blue and red dotted ovals in Figure 2.

It is important that a QDI asynchronous circuit should be devoid of circuit orphans viz. wire and gate orphans [17] [18]. Any unacknowledged signal transition on a wire is called wire orphan and any unacknowledged signal transition on a gate output is called gate orphan [19]. The signal transitions should be monotonic throughout the circuit i.e. either monotonically increasing or monotonically decreasing in a QDI asynchronous circuit to ensure proper signal acknowledgment from the first logic level up to the last logic level [20] of the QDI circuit. Imposition of the isochronic fork assumption on the primary inputs to a QDI asynchronous circuit would help to avoid the problem of wire orphan(s). This is because the

completion detector that is present in each asynchronous circuit stage, as shown in Figure 1, will guarantee the complete arrival of the data and spacer into a QDI asynchronous circuit during the respective phases. Gate orphans are complicated to resolve than wire orphans and may necessitate imposing additional timing assumptions into a QDI circuit. Hence, the logic decomposition and physical realization of QDI asynchronous circuits have to be performed carefully by following safe QDI logic decomposition principles as outlined in [21 – 23]. In the next section, we present and describe an asynchronous early output SCBCLA architecture without and with the alias logic.

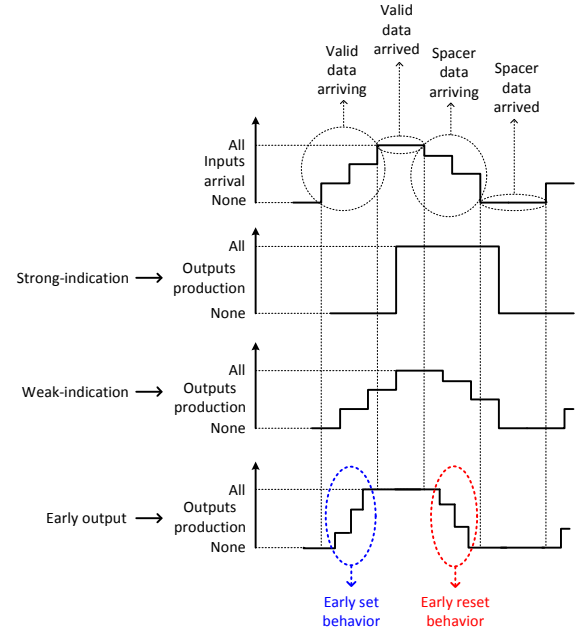


Fig. 2. Input-output timing relation of strong-indication, weak-indication and early output type QDI asynchronous circuits

III. PROPOSED ASYNCHRONOUS EARLY OUTPUT SCBCLA WITHOUT/WITH ALIAS CARRY LOGIC

The SCBCLA is based on the concept of dividing an n -bit binary adder into k sub-adder sections (i.e. k sub-SCBCLA modules) where the size of each adder section is m -bits [24]. Mathematically, $k = n/m$ where k , m and n are positive integers and are even. Here we consider $n = 32$ and $m = 4$. Hence a 32-bit SCBCLA is constructed using eight 4-bit sub-SCBCLA modules as shown in Figures 3a and 3b. In Figures 3a, 3b, 3f and 3g, (A311, A310) and (B311, B310) represent the most significant dual-rail augend and addend inputs, and (A01, A00) and (B01, B00) represent the least significant dual-rail augend and addend inputs. (C01, C00) denotes the dual-rail carry input and (C321, C320) denotes the dual-rail carry output. As seen in Figure 3, carry ripples within an adder section to produce the sum outputs, and the lookahead carry generated from an adder section is passed to the next section as the carry input.

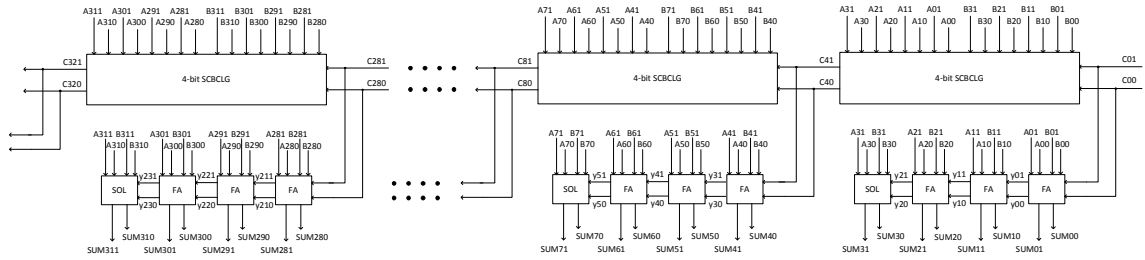


Fig. 3a. 32-bit early output SCBCLA without alias logic

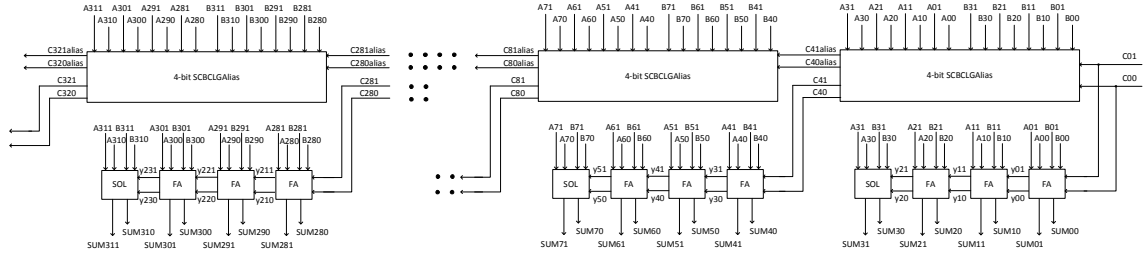


Fig. 3b. 32-bit early output SCBCLA with alias logic

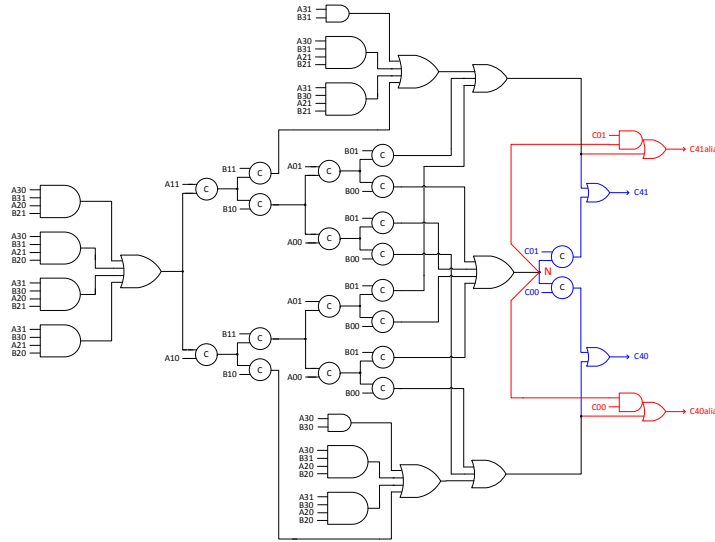


Fig. 3c. 4-bit SCBCLG without/with alias logic

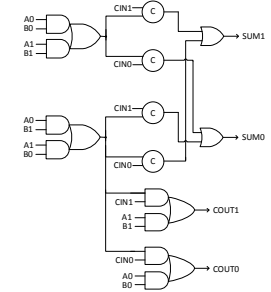


Fig. 3d. Early output full adder (FA)

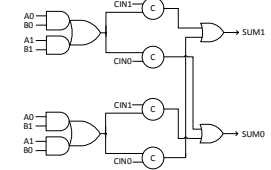


Fig. 3e. Early output sum only logic (SOL)

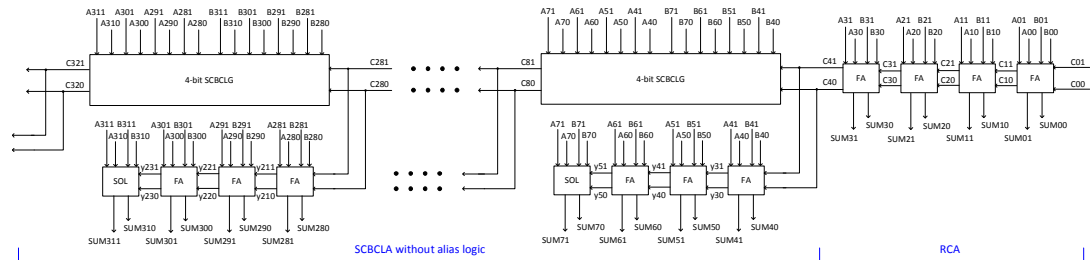


Fig. 3f. 32-bit early output SCBCLA (without alias logic) and RCA hybrid

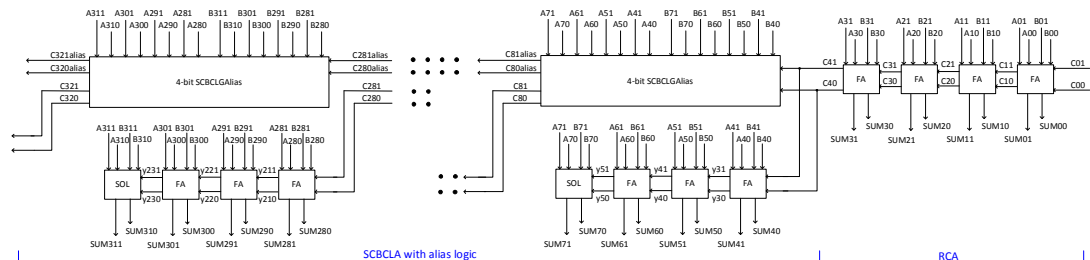


Fig. 3g. 32-bit early output SCBCLA (with alias logic) and RCA hybrid

In Figures 3a, 3b, 3f and 3g, it can be seen that there is an inter-section propagation of the carry signal based on lookahead, and an intra-section propagation of the carry signal based on a simple rippling and both these happen simultaneously. An SCBCLA uses the SCBCLG, the full adder (FA), and the sum only logic (SOL) as the circuit building blocks.

Figure 3c shows the gate-level detail of the 4-bit section-carry based carry lookahead generator (SCBCLG) without/with the alias logic. The SCBCLG is different from a conventional CLG in that only one lookahead carry output is produced. If the circuit portion shown in red is removed from Figure 3c, then the 4-bit SCBCLG produces only the lookahead carry output (C41, C40). However, if the circuit portion shown in red is retained, then the 4-bit SCBCLG produces two pairs of lookahead carry outputs, (C41, C40) and (C41alias, C40alias). Note that these two dual-rail carry output pairs are logically equivalent. Figures 3d and 3e show the gate-level details of the early output FA and the early output SOL based on [16]. The SOL is identical to the FA but does not have a carry output.

The 4-bit SCBCLG shown in Figure 3c does not contain a redundant carry logic when it produces only the dual-rail carry output (C41, C40) and not the alias dual-rail carry output (C41alias, C40alias). However, when the alias dual-rail carry output is also produced by the 4-bit SCBCLG, then it is said to contain explicit logic redundancy [25]. The logic used to produce (C41, C40) is synthesized directly [26] based on deriving the disjoint sum-of-products form [27] [28] followed by QDI logic decomposition [23]. The carry output logic shown in blue in Figure 3e cannot be discarded and (C41alias, C40alias) cannot replace the dual-rail carry output (C41, C40) of the 4-bit SCBCLG due to the gate orphan problem. To explain this, let us assume that node N, shown in red in Figure 3c, and C01 are 1 during a data phase. In the following return-to-zero phase if C01 assumes 0 before N could assume 0, then C41alias may become 0 before N becomes 0. In this case, the late assumption of 0 by N would not be subsequently acknowledged by C41alias which results in a gate orphan. It may be noted at this juncture that when the 4-bit SCBCLG produces either (C41, C40) or (C41, C40) and (C41alias, C40alias), the presence of a C-element and an OR gate with respect to C41 and C40 eliminates the problem of gate orphan(s). For example, if C01 and N were 1 during a data phase, and if C01 assumes 0 before N assumes 0 then C41 would not become 0. This is because the C-element which has C01 and N as its inputs in Figure 3c would wait for the arrival of 0 on N. Only after N becomes 0, C41 would become 0. However, since node N is considered to be isochronic, the arrival of 0 on N would be deemed to be acknowledged by both C41 and C41alias.

When (C41, C40) is sufficient to serve as the lookahead carry output of the 4-bit SCBCLG shown in Figure 3c, what is the utility of the alias lookahead carry output (C41alias, C40alias)? In Figure 3c, it can be seen

that between C01 and C41, a 2-input C-element¹ and a 2-input OR gate are present, which is the same with respect to C00 and C40. However, just a single complex gate viz. AO21 is used to connect C01 with C41alias, and likewise C00 with C40alias. If A, B, C and D are the inputs to an AO21 gate, the output of an AO21 gate, say $Y = AB + CD$. The AO21 gate requires just 10 transistors whereas the 2-input C-element and 2-input OR gate require 18 transistors. The propagation delay of an AO21 gate is less than the sum of the propagation delays of the 2-input C-element and the 2-input OR gate. Excepting the least significant 4-bit sub-SCBCLA, in the remainder of the sub-SCBCLAs, the inter-section carry propagation will be governed by the sum of the propagation delays of the 2-input C-element and the 2-input OR gate in the case of Figure 3a (32-bit SCBCLA without alias carry output logic). But in Figure 3b (32-bit SCBCLA with alias carry output logic), the inter-section carry propagation will be dictated by the propagation delay of just the AO21 gate. Hence a faster inter-section carry propagation is feasible in the case of Figure 3b compared to Figure 3a. This is an advantage of the alias carry output logic. Also, a faster return-to-zero can be facilitated by the alias carry output logic in the case of Figure 3b, while the return-to-zero in the case of Figure 3a would encounter the worst-case latency as for data processing.

Figures 3f and 3g portray two example hybrid 32-bit adders involving the SCBCLA and the ripple carry adder (RCA). Figure 3f shows an example 28-bit early output SCBCLA without alias logic which is combined together with a 4-bit early output RCA that is present in the least significant nibble position. Figure 3g shows an example 28-bit early output SCBCLA with alias logic that is joined to a less significant 4-bit early output RCA. The introduction of a RCA to replace the sub-SCBCLA or the sub-RCLA in the less significant adder positions was found to reduce the latency, area, and average power dissipation of a CLA and RCA hybrid in [26] and [29]. This shall be discussed in conjunction with the simulation results which shall be presented in the next section. The hybrid SCBCLA-RCA's shown in Figures 3f and 3g are only examples which are considered here to demonstrate their relative merits over the regular CLA counterparts.

IV. SIMULATION RESULTS AND DISCUSSION

32-bit SCBCLAs without and with alias carry output logic and 32-bit hybrid SCBCLA-RCA's without and with alias carry output logic which correspond to weak-indication and early output types, and a 32-bit early output RCLA and a RCLA-RCA hybrid were all physically realized in semi-custom ASIC style using the standard cells of a 32/28nm CMOS process [30]. The 2-input C-element was alone custom designed using 12 transistors, and was

¹ The C-element outputs 1 if all its inputs are 1, and outputs 0 if all its inputs are 0. If its inputs are different, the C-element would maintain its existing steady-state. The C-element is symbolized by the circle with the marking 'C'.

made available to realize the various asynchronous CLAs. The size of the carry lookahead generator used in all the CLAs is 4-bits. For the hybrid SCBCLAs and RCLAs, a 4-bit least significant RCA was used. Only the minimum size cells of a standard digital cell library [30] were used for the physical synthesis to enable a straightforward comparison between the synthesis results of different CLAs. Note that all the CLAs mentioned in Table I are QDI designs.

TABLE I
AVERAGE POWER DISSIPATION, (WORST-CASE) LATENCY, AND
AREA PARAMETERS OF VARIOUS 32-BIT ASYNCHRONOUS CLAs,
ESTIMATED USING A 32/28NM CMOS PROCESS

Results group	CLA or CLA-RCA hybrid adder type	Power (μ W)	Latency (ns)	Area (μ m ²)
<i>References [26] [31]: Weak-indication</i>				
Group1	SCBCLA (Without alias logic)	2191	3.31	2951.88
	SCBCLA-RCA hybrid (Without alias logic)	2189	3.08	2845.14
	SCBCLA (With alias logic)	2192	2.46	2992.55
	SCBCLA-RCA hybrid (With alias logic)	2190	2.38	2880.72
<i>References [26] [32]: Weak-indication</i>				
Group2	SCBCLA (Without alias logic)	2188	3.14	2915.29
	SCBCLA-RCA hybrid (Without alias logic)	2186	2.93	2807.02
	SCBCLA (With alias logic)	2190	2.32	2955.95
	SCBCLA-RCA hybrid (With alias logic)	2187	2.25	2842.60
<i>References [29] [16]: Early output</i>				
Group3	RCLA	2177	2.75	2569.65
	RCLA-RCA hybrid	2175	2.53	2455.80
<i>Proposed: Early output</i>				
Group4	SCBCLA (Without alias logic)	2178	3.13	2524.92
	SCBCLA-RCA hybrid (Without alias logic)	2175	2.92	2416.66
	SCBCLA (With alias logic)	2179	2.31	2565.58
	SCBCLA-RCA hybrid (With alias logic)	2177	2.23	2452.24

Approximately 1000 random input vectors were identically supplied to all the CLAs through a test bench at time intervals of 20ns to perform the functional simulations and also to capture their respective switching activities. The value change dump files generated through the functional simulations were then used for average power estimation using Synopsys tools. The worst-case latency

i.e. the critical path delay and the area occupancy of the CLAs were also estimated using Synopsys tools. An appropriate wire load model (parasitic) was included while estimating the design metrics, which are given in Table I. The optimized design parameters are highlighted in bold-face in Table I. Since the input registers and completion detector of the various CLAs are identical, the differences between their design metrics is attributable to the differences between their respective function blocks.

The simulation results corresponding to various CLAs are split into four groups, labeled as Group1 to Group4 in Table I, for the sake of discussion. Group1 corresponds to regular and hybrid SCBCLAs without and with the alias carry output logic, which are weakly indicating. With respect to Group1, the 4-bit SCBCLG was realized based on a direct synthesis [26], and the FA and SOL are realized based on [31]. Since the FA and SOL of [31] are weakly indicating, the regular and hybrid SCBCLAs corresponding to Group1 also conform to weak-indication. Reference [32] presented a latency optimized weakly indicating FA design. This FA and the associated SOL was used to replace the FA and SOL components of the regular and hybrid SCBCLAs in Group 1, which yielded the Group2 results. Since the FA of [32] is more optimized compared to the FA of [31], therefore Group2 results are better compared to Group1 results as seen in Table I. The weak-indication FA of [31] occupies 41.17 μ m² of silicon, and the weak-indication FA of [31] occupies a reduced area of 39.65 μ m². The SOL based on [31] or [32] is the same and occupies 34.56 μ m² of silicon.

Group3 comprises a regular RCLA based on [29] and a RCLA-RCA hybrid based on [29] and [16]. The 4-bit RCA employed in the RCLA-RCA hybrid is composed of 4 FA modules, and the FA is based on [16]. Since the sub-RCLAs and the RCA are early output type, therefore the RCLA and the RCLA-RCA hybrid also correspond to early output type. There is no possibility for introducing an alias carry output logic in the case of the RCLA or the RCLA-RCA hybrid. This is because the lookahead carry output of one sub-RCLA directly serves as the carry input for the successive sub-RCLA. In the regular or hybrid SCBCLAs however, the lookahead carry output generated from one sub-SCBCLA serves as the carry input for the next SCBCLG and also as the carry input for the sub-RCA embedded within the successive sub-SCBCLA. Due to the supply of two dual-rail carry inputs to a sub-SCBCLA, the alias carry output logic was able to be introduced to achieve significant optimization in the latency at the expense of meagre increases in area and average power dissipation due to the redundant carry output logic.

Group4 comprises the proposed regular and hybrid SCBCLAs without/with the alias carry output logic, which corresponds to early output type. This results from the use of the early output 4-bit SCBCLG without/with the alias carry output logic, the early output FA, and the early output SOL. The early output plain 4-bit SCBCLG requires 113.35 μ m² of silicon, and the silicon requirement increases

to $118.43\mu\text{m}^2$ with the introduction of the alias carry output logic. The early output FA and SOL require reduced areas compared to the weak-indication FA and SOL of [32] of just $27.45\mu\text{m}^2$ and $22.36\mu\text{m}^2$ of silicon respectively. Since the early output asynchronous circuits are more relaxed compared to their strong- and weak-indication circuit counterparts, simple and complex logic gates of a digital cell library can be widely used compared to the C-element. As a result, the early output asynchronous circuits generally facilitate optimizations in the design metrics compared to the strong- and weak-indication asynchronous circuits. This is the primary reason for the Group4 results being more optimized compared to the synthesis results of Group1, Group2 and Group3.

Three important observations can be made from Table I. Firstly, the SCBCLAs with alias logic report a substantial reduction in latency compared to the SCBCLAs without alias logic, and due to the redundant logic introduced in the case of the former their area and power metrics are marginally more expensive compared to the latter. On average, the SCBCLAs with alias carry output logic which correspond to Group1, Group2 and Group4, whether they are regular or hybrid variants, report 24.6% reduction in latency and 1.4% increase in cells area with negligible power increase (0.1%) compared to the averaged design metrics of the regular and hybrid SCBCLAs which have no alias carry output logic. This implies the SCBCLAs featuring the alias carry output logic achieve significant reduction in latency with almost no increase in the area and power metrics. In general, the power dissipation values of the CLAs mentioned in Table I do not vary significantly. This is because the function blocks of all the CLAs satisfy the monotonic cover constraint [10], which signifies the activation of a unique signal path from a primary input to a primary output. In general, the monotonic cover constraint is inherent in QDI circuit designs.

Secondly, the SCBCLA-RCAs hybrid and the RCLA-RCA hybrid enable additional optimizations in the design metrics compared to the regular SCBCLAs and RCLA. On average, the SCBCLA-RCAs hybrid without alias logic and the RCLA-RCA hybrid report a 7% reduction in latency and a 4% reduction in area with no power increase compared to the regular SCBCLAs without alias logic and the regular RCLA. Likewise, the SCBCLA-RCAs hybrid with alias carry logic, on average, report a 3% reduction in latency and a 4% reduction in area without any power increase compared to the regular SCBCLAs with alias logic. The area reduction is mainly because a sub-SCBCLA without/with the alias carry logic and a sub-RCLA are more area expensive than a similar size RCA. For example, the areas of the proposed early output 4-bit SCBCLAs without and with the alias carry logic and the 4-bit RCLA [29] are $218.06\mu\text{m}^2$, $223.14\mu\text{m}^2$ and $223.65\mu\text{m}^2$ respectively. In contrast, the area of the early output 4-bit RCA is just $109.8\mu\text{m}^2$. The critical path delay of the least significant 4-bit SCBCLA with alias carry logic (Figure 3b) is governed by the sum of the propagation delays of a 4-

input AND gate, two 4-input OR gates, four 2-input C-elements and an AO21 gate. On the other hand, the least significant 4-bit RCA shown in Figure 3g encounters the sum of the propagation delays of five AO22 gates. Hence, Figures 3f and 3g will exhibit reduced latencies than Figures 3a and 3b. Thus using a small RCA to replace the sub-SCBCLAs or the sub-RCLA in the least significant positions is beneficial for reducing the area, latency and power parameters, which is substantiated by the results given in Table I. Nevertheless, the optimum size of the least significant RCA which may be cascaded along with a CLA (i.e. SCBCLA or RCLA) to effect optimizations in the design metrics should be determined based on static timing analysis since the use of a larger size RCA may negatively impact the latency metric.

Thirdly, it is clear from Table I that the proposed 32-bit SCBCLA-RCA hybrid incorporating the alias carry output logic features the least latency and is preferable. It is slightly more expensive in area than the 32-bit SCBCLA-RCA hybrid with no alias carry output logic by just 1.5% and the power increase is negligible (0.1%).

V. CONCLUSIONS

This paper has presented a new asynchronous early output SCBCLA architecture without/with the alias carry output logic. The 32-bit binary addition was considered as the case study and the proposed SCBCLA with alias carry output logic reports optimized design metrics compared to the other SCBCLAs and RCLA proposed earlier. It was shown that further optimization in the design metrics could be achieved by opting for a SCBCLA-RCLA hybrid.

REFERENCES

- [1] A.J. Martin, "The limitation to delay-insensitivity in asynchronous circuits," *Proc. 6th MIT Conference on Advanced Research in VLSI*, 1990, pp. 263-278.
- [2] C.H. van Berkel, M.B. Josephs, S.M. Nowick, "Applications of asynchronous circuits," *Proc. of the IEEE*, 1999, vol. 87, no. 2, pp. 223-233.
- [3] I. David, R. Ginosar, M. Yoeli, "Self-timed is self-checking," *Journal of Electronic Testing: Theory and Applications*, 1995, vol. 6, no. 2, pp. 219-228.
- [4] L.A. Plana, P.A. Riocreux, W.J. Bainbridge, A. Bardsley, S. Temple, J.D. Garside, Z.C. Yu, "SPA – a secure Amulet core for smartcard applications," *Microprocessors and Microsystems*, 2003, vol. 27, no. 9, pp. 431-446.
- [5] D. Sokolov, J. Murphy, A. Bystrov, A. Yakovlev, "Design and analysis of dual-rail circuits for security applications," *IEEE Transactions on Computers*, 2005, vol. 54, no. 4, pp. 449-460.
- [6] F. Burns, A. Bystrov, A. Koelmans, A. Yakovlev, "Design and security evaluation of balanced 1-of-n circuits," *IET Computers and Digital Techniques*, 2012, vol. 6, no. 2, pp. 125-135.
- [7] W. Cilio, M. Linder, C. Porter, J. Di, D.R. Thompson, S.C. Smith, "Mitigating power- and timing-based side-channel attacks using dual-spacer dual-rail delay-insensitive

- asynchronous logic," *Microelectronics Journal*, 2013, vol. 44, no. 3, pp. 258-269.
- [8] T. Verhoeff, "Delay-insensitive codes – an overview," *Distributed Computing*, 1988, vol. 3, no. 1, pp. 1-8.
 - [9] S.J. Piestrak, T. Nanya, "Towards totally self-checking delay-insensitive systems," *Proc. 25th International Symposium on Fault-Tolerant Computing*, 1995, pp. 228-237.
 - [10] J. Sparsø, S. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*, Kluwer Academic, Boston, MA, USA, 2001.
 - [11] M.T. Moreira, R.A. Guazzelli, N.L.V. Calazans, "Return-to-one protocol for reducing static power in C-elements of QDI circuits employing m-of-n codes," *Proc. 25th Symposium on Integrated Circuits and Systems Design*, 2012, pp. 1-6.
 - [12] C.L. Seitz, "System Timing," in *Introduction to VLSI Systems*, C. Mead and L. Conway (Eds.), pp. 218-262, Addison-Wesley, MA, USA, 1980.
 - [13] C.F. Brej, J.D. Garside, "Early output logic using anti-tokens," *Proc. 12th International Workshop on Logic and Synthesis*, 2003, pp. 302-309.
 - [14] P. Balasubramanian, D.A. Edwards, "Efficient realization of strongly indicating function blocks," *Proc. IEEE Computer Society Annual Symposium on VLSI*, 2008, pp. 429-432.
 - [15] P. Balasubramanian, D.A. Edwards, "A new design technique for weakly indicating function blocks," *Proc. 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, 2008, pp. 116-121.
 - [16] P. Balasubramanian, "A robust asynchronous early output full adder," *WSEAS Transactions on Circuits and Systems*, 2011, vol. 10, no. 7, pp. 221-230.
 - [17] C. Jeong, S.M. Nowick, "Block-level relaxation for timing-robust asynchronous circuits based on eager evaluation," *Proc. 14th IEEE International Symposium on Asynchronous Circuits and Systems*, 2008, pp. 95-104.
 - [18] P. Balasubramanian, K. Prasad, N.E. Mastorakis, "Robust asynchronous implementation of Boolean functions on the basis of duality," *Proc. 14th WSEAS International Conference on Circuits*, 2010, pp. 37-43.
 - [19] P. Balasubramanian, "Comments on "Dual-rail asynchronous logic multi-level implementation"," *Integration, the VLSI Journal*, 2016, vol. 52, no. 1, pp. 34-40.
 - [20] V.I. Varshavsky (Ed.), *Self-Timed Control of Concurrent Processes: The Design of Aperiodic Logical Circuits in Computers and Discrete Systems*, Chapter 4: Aperiodic Circuits, pp. 77-85, (Translated from the Russian by A.V. Yakovlev), Kluwer Academic, 1990.
 - [21] W.B. Toms, "Synthesis of quasi-delay-insensitive datapath circuits," *PhD thesis*, The University of Manchester, 2006.
 - [22] P. Balasubramanian, *Self-Timed Logic and the Design of Self-Timed Adders*, PhD thesis, The University of Manchester, 2010.
 - [23] P. Balasubramanian, N.E. Mastorakis, "QDI decomposed DIMS method featuring homogeneous/heterogeneous data encoding," *Proc. International Conference on Computers, Digital Communications and Computing*, 2011, pp. 93-101.
 - [24] A.R. Omondi, *Computer Arithmetic Systems: Algorithms, Architecture and Implementations*, Prentice Hall International (UK), 1994.
 - [25] P. Balasubramanian, D.A. Edwards, W.B. Toms, "Redundant logic insertion and latency reduction in self-timed adders," *VLSI Design*, vol. 2012, Article ID 575389, 2012, pages 13.
 - [26] P. Balasubramanian, D.A. Edwards, W.B. Toms, "Self-timed section-carry based carry lookahead adders and the concept of alias logic," *Journal of Circuits, Systems, and Computers*, 2013, vol. 22, no. 4, pp. 1350028-1 – 1350028-24.
 - [27] P. Balasubramanian, R. Arisaka, H.R. Arabnia, "RB_DSOP: A rule based disjoint sum of products synthesis method," *Proc. 12th International Conference on Computer Design*, 2012, pp. 39-43.
 - [28] P. Balasubramanian, N.E. Mastorakis, "A set theory based method to derive network reliability expressions of complex system topologies," *Proc. Applied Computing Conference*, 2010, pp. 108-114.
 - [29] P. Balasubramanian, D. Dhivya, J.P. Jayakirthika, P. Kaviyarasi, K. Prasad, "Low power self-timed carry lookahead adders," *Proc. 56th IEEE International Midwest Symposium on Circuits and Systems*, 2013, pp. 457-460.
 - [30] Synopsys *SAED_EDK32/28_CORE Databook*, Revision 1.0.0, 2012.
 - [31] P. Balasubramanian, D.A. Edwards, "A delay efficient robust self-timed full adder," *Proc. IEEE 3rd International Design and Test Workshop*, 2008, pp. 129-134.
 - [32] P. Balasubramanian, "A latency optimized biased implementation style weak-indication self-timed full adder," *Facta Universitatis, Series: Electronics and Energetics*, 2015, vol. 28, no. 4, pp. 657-671.