electronics

MDPI

Article

Monotonic Asynchronous Two-Bit Full Adder

Padmanabhan Balasubramanian and Douglas L. Maskell *

College of Computing and Data Science, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore; balasubramanian@ntu.edu.sg
* Correspondence: asdouglas@ntu.edu.sg; Tel.: +65-6790-6259

Abstract: Monotonic circuits are a class of input–output mode (IOM) asynchronous circuits that are relaxed compared to quasi-delay-insensitive (QDI) IOM asynchronous circuits in terms of signaling the completion of internal processing. Some recent works have demonstrated the superiority of monotonic logic over QDI logic for arithmetic circuits such as adders and multipliers. This paper presents a new monotonic asynchronous two-bit full adder (TFA) that can be duplicated and cascaded to form a ripple-carry adder (RCA). While an RCA is a slow adder with respect to synchronous design, with respect to IOM asynchronous design an RCA is a noteworthy adder since it has perhaps the least reverse latency that is not attainable through other IOM asynchronous adders. Conventionally, an RCA is constructed via a cascade of one-bit full adders (OFAs). An OFA adds two input bits along with any carry input and produces a sum bit and any carry output. On the other hand, a TFA simultaneously adds two pairs of input bits along with any carry input and produces two sum bits and any carry output. Using our proposed monotonic TFA, we realized an RCA to compare its performance with RCAs constructed using different asynchronous OFAs, and RCAs constructed using existing TFAs. We considered the popular delay-insensitive dual-rail scheme for encoding the adder inputs and outputs, and two 4-phase handshake protocols, namely return-to-zero handshaking (R0H) and return-to-one handshaking (R1H) for communication separately. We used a 28 nm CMOS process for implementation and considered a 32-bit addition as an example. Based on the design metrics estimated, the following inferences were derived: (i) compared to the RCA using the state-of-the-art monotonic OFA, the RCA incorporating the proposed TFA achieved a 26% reduction in cycle time for R0H and a 28.5% reduction in cycle time for R1H while dissipating almost the same power; the cycle time governs the data application rate in an IOM asynchronous circuit, and (ii) compared to the RCA comprising an early output QDI TFA, the RCA incorporating the proposed TFA achieved a 22.3% reduction in cycle time for R0H and a 25.4% reduction in cycle time for R1H while dissipating moderately less power. Also, compared to the existing early output QDI TFA, the proposed TFA occupies 40.9% less area for R0H and 42% less area for R1H.

Keywords: asynchronous circuits; arithmetic circuits; logic design; digital circuits; high-speed; low power; CMOS

Citation: Balasubramanian, P.; Maskell, D.L. Monotonic Asynchronous Two-Bit Full Adder. *Electronics* **2024**, *13*, 1717. https://doi.org/10.3390/electronics13091717

Academic Editors: Graça Minas, Paolo Stefano Crovetti, João Paulo Pereira do Carmo and Manuel Fernando Silva

Received: 7 March 2024
Revised: 9 April 2024
Accepted: 25 April 2024
Published: 29 April 2024

1. Introduction

Adders are widely used in various computing units, serving several purposes. In computer arithmetic, adders are used to perform addition on binary numbers, essential for arithmetic calculations in digital systems. In computer architecture, adders form key components of arithmetic logic units (ALUs), enabling the execution of arithmetic and logical operations in central processing units. They are also utilized in graphics processing units (GPUs) for rendering and image processing tasks. In image processing, adders are utilized for tasks like pixel manipulation and enhancement. Adders are crucial in cryptography and are employed in algorithms like the Advanced Encryption Standard for encryption and decryption processes. Adders play a vital role in signal processing, facilitating tasks such as filtering, modulation, and demodulation. Adders are integral to error correction

codes, where they compute checksums and parity bits for data integrity verification. In communication systems, adders are used for channel coding and modulation schemes. In neuromorphic computing, adders are essential for computing weighted sums of input signals, mimicking synaptic integration in biological neurons. They enable neural networks to perform mathematical operations required for learning and inference tasks. By summing up weighted inputs, adders facilitate the implementation of various neural network architectures, including spiking neural networks. In robotics and control systems, adders are employed for real-time computations, enabling tasks such as path planning and feedback control. Overall, adders are indispensable components in practical computing applications.

Arithmetic operations like additions and multiplications are identified as the primary contributors to power consumption in computing systems. For instance, over 70% of power utilized by a GPU is ascribed to arithmetic operations [1]. Fast Fourier Transform (FFT) and inverse FFT operations are widespread in orthogonal frequency division multiplexing transceivers employed in wireless communication systems, and approximately, 80% of power expended by a FFT processor is attributed to adders and multipliers [2]. Moreover, in digital image processing applications with JPEG encoding or decoding, and multimedia involving MPEG encoding or decoding for digital video processing, the discrete cosine transform and its inverse variants are commonplace, which involve substantial additions and multiplications. Computer arithmetic plays a significant role in digital signal processing, with adders and multipliers being the primary components found in the data path of a digital signal processing unit. Addition emerged as the most executed operation among a collection of real-time digital signal processing benchmarks [3]. Approximately 72% of instructions executed by a prototype RISC viz. DLX was associated with addition or subtraction operations [4]. The examination of an ARM processor's ALU showed that additions accounted for nearly 80% of the operations [5].

A one-bit full adder (OFA) serves as a fundamental computing element within arithmetic circuits utilized across microprocessors, microcontrollers, and digital signal processors. Essentially, it is employed to add a pair of binary input bits along with a carry input from a preceding stage, resulting in two binary output bits: sum and carry (overflow). This component can be implemented in synchronous [6] or asynchronous design styles [7]. As an alternative to OFA, a two-bit full adder (TFA) can be used, leveraging both synchronous and asynchronous design approaches. The TFA operates by adding two pairs of input bits along with a carry input, generating two sum bits as well as a carry overflow. This paper presents a new monotonic input–output mode (IOM) asynchronous TFA.

IOM asynchronous circuits employ delay-insensitive encoding techniques for data representation and a 4-phase handshake protocol for data transmission. Unlike synchronous circuits which rely on global/local clock signals, IOM asynchronous circuits function based on events, rendering them intrinsically more resilient. The event-triggered characteristic endows IOM asynchronous circuits with heightened resistance to fluctuations in voltage, process, and temperature, enhancing their adaptability [8]. Furthermore, they boast modularity and heightened immunity to electromagnetic interference, distinguishing them favorably from synchronous circuits and so they are well-suited for security-related tasks [9].

IOM asynchronous circuits are divided into two main categories: QDI and non-QDI. QDI circuits utilize isochronic forks [10] which are electrical junctions where two or more wires may emerge, and the signal transitions on the wires are assumed to happen simultaneously—this premise has been validated in micro- and nano-electronics [11]. Quasi-delay-insensitivity mandates that a circuit's outputs are generated after all the inputs are received and the internal processing has been completed. While bolstering the resilience of QDI circuits, this characteristic also escalates the implementation costs such as latency, area, and power compared to non-QDI circuits. Diverse variations of QDI circuits exist, including strong indication and weak indication types [12] and early output QDI circuits [13]. Strong indication circuits require the receipt of all the primary inputs to commence the processing and produce all the primary outputs. Weak indication circuits tend to commence the

processing after receiving a subset of primary inputs to produce some primary outputs. However, only after receiving the last primary input, a weak indication circuit can fully complete the processing to produce the last primary output. The weak indication may be 'distributed' or 'biased'—in the former, the responsibility to indicate the primary inputs is distributed between the primary outputs; in the latter, the responsibility to indicate the primary inputs is delegated to one or more primary outputs and thus at least one primary output is exempt from indication. Early output circuits can produce all primary outputs by processing a subset of primary inputs, particularly for the processing of the spacer. Associating isochronic forks with primary inputs takes care of the late-arriving primary inputs in early output QDI circuits.

IOM asynchronous circuits outside the QDI realm include relative timed circuits [14] and monotonic circuits [15,16]. Relative timed circuits include sophisticated timing assumptions made internally to sequence the inputs and produce the outputs, whereas monotonic circuits only ensure the consistency of signal transitions within the circuit [15]. Therefore, monotonic circuits are more relaxed compared to QDI and relative timed circuits. Monotonicity means that rising signal transitions (binary 0 to 1) on a circuit's inputs lead to similar rising signal transitions on the circuit's outputs, and falling signal transitions (binary 1 to 0) on a circuit's inputs lead to similar falling signal transitions on the circuit's outputs. Monotonic circuits may exhibit monotonically rising, monotonically falling, or monotonically rising and falling behaviors. Henceforth, in this paper, 'monotonic circuits' refer to IOM asynchronous circuits that exhibit both monotonically rising and falling attributes.

QDI circuits mandate the full completion of internal processing before producing all primary outputs. This is usually achieved by avoiding the occurrence of wire and gate orphans. A wire orphan is a non-acknowledged signal transition on an input wire. A gate orphan is a non-acknowledged signal transition on an intermediate gate output in a QDI circuit. In QDI circuits, wire orphans can be overcome by associating isochronic forks [17] with the primary inputs and/or some internal outputs while gate orphans can be overcome by utilizing safe QDI logic decomposition techniques [18]. Examples of wire and gate orphans are given in [19]—an interested reader may refer to this literature for details. Monotonic circuits operate similarly to early output QDI circuits but they are not QDI circuits since they do not mandate the completion of internal processing before producing the primary outputs. Therefore, monotonic circuits discard gate orphans. However, in monotonic circuits, the isochronic fork assumption is associated with the primary inputs to avoid the occurrence of any wire orphan. Unlike QDI circuits, non-QDI circuits are rather relaxed in general, enabling a decrease in circuit complexity and implementation costs, and consequently, they are likely to attain superior performance metrics.

The rest of this paper has five sections. The fundamentals of IOM asynchronous circuits are described in Section 2. Next, a survey of IOM asynchronous OFAs pertaining to QDI and non-QDI types is given in Section 3. In Section 4, we briefly discuss existing asynchronous QDI TFAs and then present the proposed monotonic TFA. The results of the implementation covering IOM asynchronous RCAs incorporating different OFAs and TFAs including the proposed design are given in Section 5. Lastly, Section 6 highlights some conclusions drawn from this research.

2. Basic Architecture of IOM Asynchronous Circuits

Figure 1a presents a block diagram illustrating one IOM asynchronous pipeline stage. A pipeline stage consists of an asynchronous circuit placed between a set of input-side and output-side registers. The input-side registers would function as output-side registers for any previous stage, and the output-side registers would function as input-side registers for any subsequent stage. The input-side registers facilitate processing by furnishing inputs to the asynchronous circuit. The completion detector conveys the completion of output generation by the asynchronous circuit. Figure 1b,c depict example completion detectors pertaining to return-to-zero handshaking (R0H) and return-to-one handshaking (R1H), respectively—these handshaking schemes will be discussed at a later stage in

this section. The completion detector linked to the output registers provides an output acknowledgment signal (Ackout) that is inverted, resulting in the input acknowledgment signal (Ackin). Ackin enables the input registers to furnish new inputs (data/spacer) to the IOM asynchronous circuit. The communication protocol between input-side and output-side registers in IOM asynchronous circuits is called handshaking.

The C-element [20] serves as a register in an IOM asynchronous circuit. The C-element outputs binary 0 (1) when all its inputs are binary 0 (1). On the contrary, if the inputs of a C-element are different, the C-element would maintain its existing steady state. Assuming J and K are a C-element's inputs, the C-element output denoted by L is given by the expression L = JK + (J + K) L. The transistor-level realization of a 2-input C-element depicted in Figure 1d is obtained by introducing feedback in the static CMOS realization of an AO222 gate [21]. Many full-custom implementations of the C-element were presented and compared in [22,23]. However, in this paper, we utilize a semi-custom implementation of the C-element [21], depicted in Figure 1d. In the input-side registers shown in Figure 1a, each C-element has one input joined to Ackin, while the other input is joined to an encoded input rail.

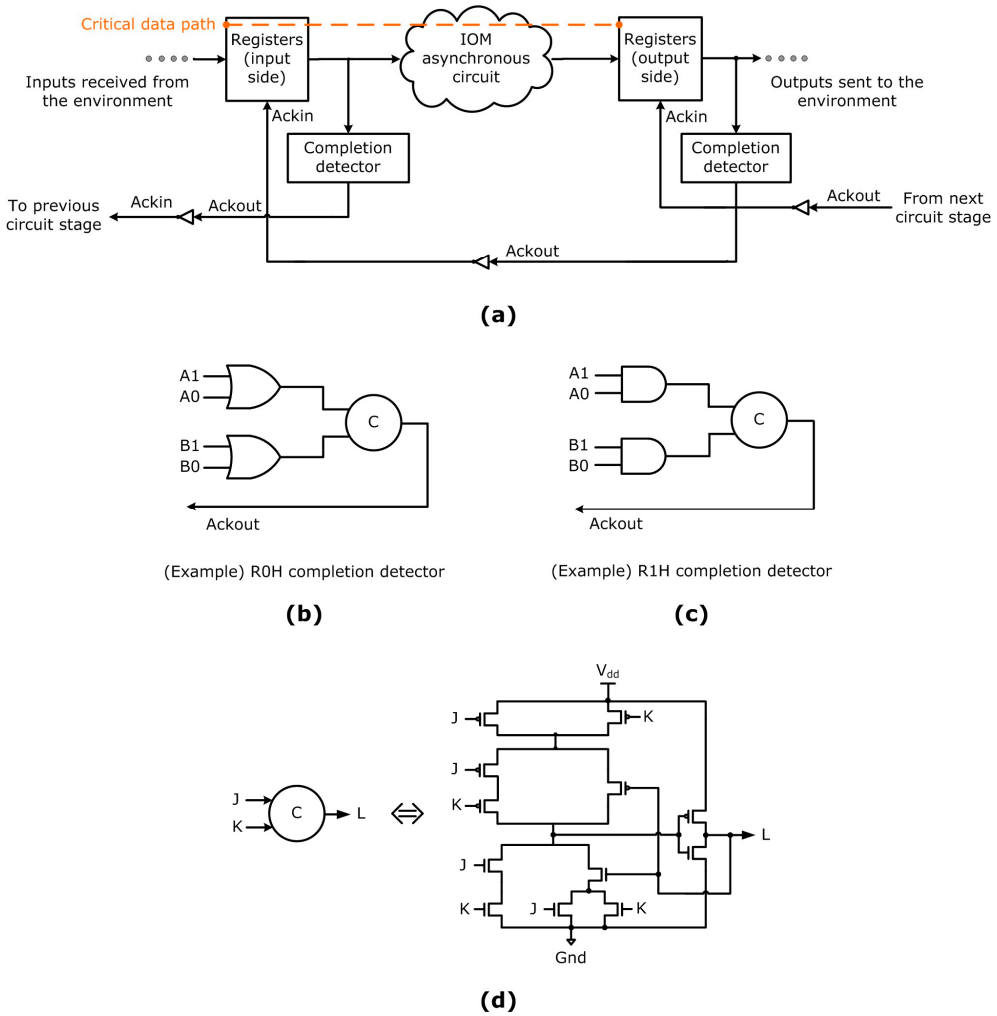

Figure 1. (**a**) An IOM asynchronous circuit pipeline stage; sample completion detectors corresponding to (**b**) return-to-zero handshaking (R0H), and (**c**) return-to-one handshaking (R1H). (**d**) Logic symbol and static CMOS realization of a 2-input C-element, obtained by introducing feedback in an AO222 complex gate [21].

A delay-insensitive code [24], such as the dual-rail code (also known as 1-of-2 code) is commonly used to encode inputs and outputs of an IOM asynchronous circuit. We shall

explain how inputs and outputs are encoded based on dual-rail encoding with respect to R0H and R1H before discussing the handshake protocols. As per dual-rail encoding corresponding to R0H handshaking [25], an input, denoted as T, is represented by two wires or rails, designated as T1 and T0 here. If T equals 1, it is represented by the encoding T1 = 1 and T0 = 0; if T equals 0, it is represented by the encoding T0 = 1 and T1 = 0. These two encodings are termed 'data' with respect to R0H. T1 = T0 = 0 signifies the '(zeroes) spacer' inserted between successive data in R0H. T1 = T0 = 1 is not legal as the coding scheme must be unordered [26]. As per dual-rail encoding corresponding to R1H handshaking [27], an input signal T is once again represented by two wires or rails as say, T1 and T0. If T equals 1, it is represented by the encoding T1 = 0 and T0 = 1, whereas if T equals 0, it is represented by the encoding T0 = 0 and T1 = 1. These two encodings are termed 'data' with respect to R1H. T1 = T0 = 1 denotes the '(ones) spacer' inserted between consecutive data in R1H. T1 = T0 = 0 is not legal since the coding scheme must be unordered.

Figure 1b,c exemplify dual-rail encoded inputs (A1, A0) and (B1, B0). Figure 1b shows a sample completion detector associated with R0H. In this configuration, the completion detector comprises OR gates in the first level, and each 2-input OR gate combines the two rails of each encoded input. The outputs of those OR gates are given to a C-element or a series of C-elements to generate Ackout. Figure 1c shows a sample completion detector related to R1H. In this configuration, the completion detector comprises AND gates in the first level, and each 2-input AND gate combines the two rails of each encoded input. The outputs of those AND gates are given to a C-element or a series of C-elements to generate Ackout.

We shall now explain the handshake protocols R0H and R1H. Regarding R0H, the first phase involves driving Ackin to 1 while Ackout is 0. This prompts the input-side registers to supply data to the asynchronous circuit. Throughout this phase, one of the dual rails of the entire data bus will assume 1 implying that data is sent to the asynchronous circuit for processing. In the second phase, the outputs produced using the asynchronous circuits are received by the output-side registers, and the completion detector associated with them issues an Ackout of 1. Moving on to the third phase, the input-side registers wait for the transition of Ackin to 0 and then deliver the (zeroes) spacer to the asynchronous circuit for processing. In the final phase, the spacer output produced using the asynchronous circuit is received by the output-side registers, and the completion detector associated with them will subsequently issue an Ackout of 0. This process defines the completion of a data transaction and signals the readiness of the asynchronous circuit to initiate the next data transaction upon the transition of Ackin to 1. Therefore, in R0H, the inputs will be supplied according to the sequence: data–spacer–data–spacer, and so forth.

Regarding R1H, the first phase involves driving Ackin to 1 while Ackout is 0. This prompts the input-side registers to send the (ones) spacer to the asynchronous circuit for processing. Throughout this phase, all the encoded input rails will be set to 1, implying that the spacer is conveyed to the asynchronous circuit for processing. Moving on to the second phase, the spacer output produced using the asynchronous circuit is received by the output-side registers and the completion detector associated with them transmits an Ackout of 1. In the subsequent phase, the input registers will wait for the transition of Ackin to 0 after which data are sent to the asynchronous circuit for processing—this implies that one of the dual rails of the entire data bus is driven to 0. In the final phase, the data output via the asynchronous circuit is received by the output-side registers and the completion detector associated with them will subsequently issue an Ackout 0. This process defines the completion of a data transaction and signals the readiness of the asynchronous circuit to initiate the next data transaction upon the transition of Ackin to 1. Therefore, in R1H, the inputs will be supplied according to the sequence: spacer–data–spacer–data, and so forth.

Figure 2a,b show representative input–output timing relations in strong indication, weak indication, and early output QDI/relative-timed/monotonic circuits according to R0H and R1H, respectively. The arrival of data is highlighted by the dotted blue circles and ovals, and the arrival of the spacer is highlighted by the dotted red circles and ovals. The

early set and reset phenomena are highlighted by dotted brown and green ovals. Some of the gates used in an IOM asynchronous circuit may exhibit the early set property while some of the gates may exhibit the early reset property. For example, an OR gate exhibits an early set and this is because it can output 1 even if one of its inputs assumes 1; and an AND gate exhibits an early reset and this is because it can output 0 even if one of its inputs assumes 0. On the contrary, the inverter and the Muller C-element do not exhibit early set or reset property—this is because both these gates require all the input(s) to effect a change in their output. Concerning the OFA and the TFA, to be discussed in this paper, some of those are strongly indicating while the rest are weakly indicating or early output type which includes early output QDI and monotonic types. In early output OFAs and TFAs, the early reset (set) property is manifested for the spacer processing in the case of R0H (R1H).

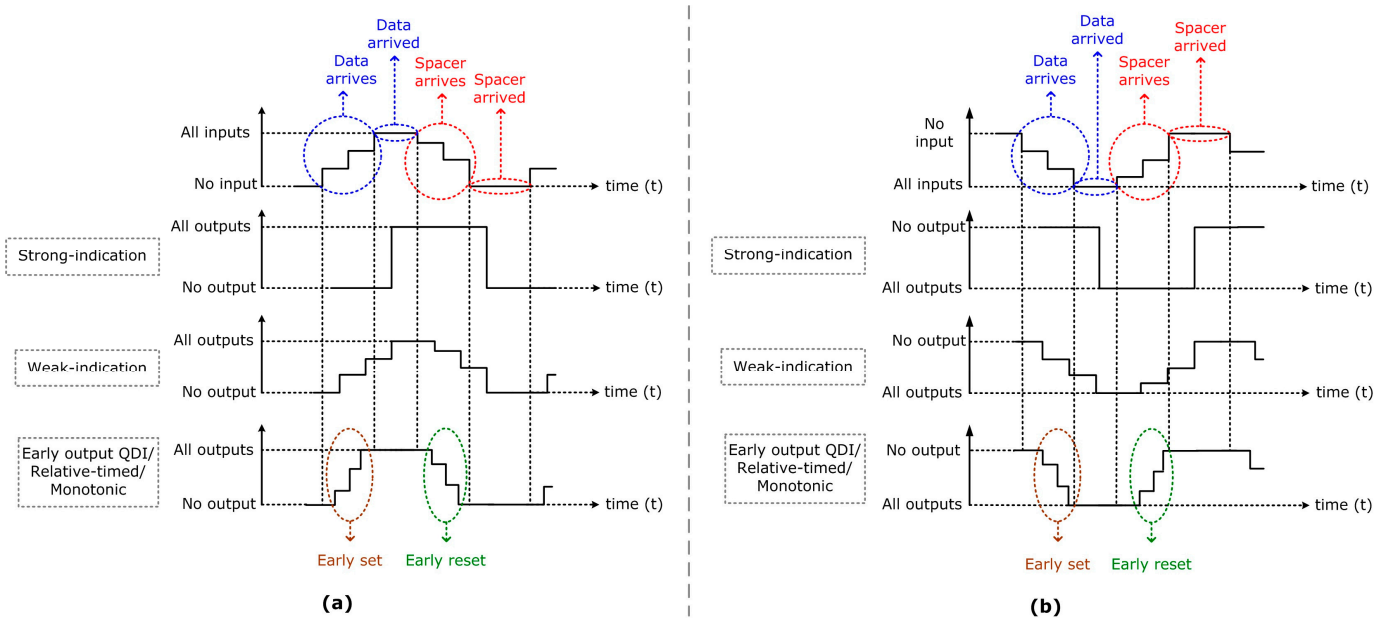

Figure 2. Input–output timing relation in strong indication, weak indication, and early output QDI/relative-timed/monotonic circuits corresponding to (**a**) R0H and (**b**) R1H.

To ensure the delay insensitivity in QDI circuits, the spacer is inserted between successive input data. In a monotonic circuit, introducing the spacer between successive input data enables achieving delay insensitivity externally for handshaking purposes, preventing collisions between data and the spacer. In the IOM asynchronous pipeline shown in Figure 1a, the main timing parameter is the 'cycle time', which denotes the time taken to complete one data transaction. The maximum processing time taken for data is called forward latency and the maximum processing time taken for the spacer is termed reverse latency. Whether the forward latency equals the reverse latency or not in an IOM asynchronous circuit depends on the underlying logic of an asynchronous circuit. The cycle time of an IOM asynchronous circuit is defined as the sum of forward latency and reverse latency. The critical path dictating the latency of an IOM asynchronous circuit involves an input register bank and the asynchronous circuit, highlighted by the orange dashed line in Figure 1a.

3. IOM Asynchronous RCAs with One-Bit Full Adders

Cascading N OFAs allow for the creation of an N-bit ripple carry adder (RCA), as shown in Figure 3, which is dual-rail encoded. In Figure 3, A and B signify the adder inputs, and SUM signifies the adder output. C denotes the carries, generated internally. Input bits A(N–1) and B(N–1), and sum bit SUM(N) are the most significant while input bits A(0) and B(0) and sum bit SUM(0) are the least significant. While the RCA offers

advantages such as a reduced area and low power dissipation compared to other high-speed adders, it offers a slower speed for synchronous design. Nevertheless, the RCA architecture holds significance for an IOM asynchronous design, particularly because certain RCA architectures demonstrate minimal reverse latency that cannot be attained by other adder architectures. Moreover, the RCA boasts the least silicon footprint among its counterparts and dissipates less power. In this context, we regard the RCA architecture as a platform to analyze the performance of several IOM asynchronous OFAs and TFAs including the proposed TFA.

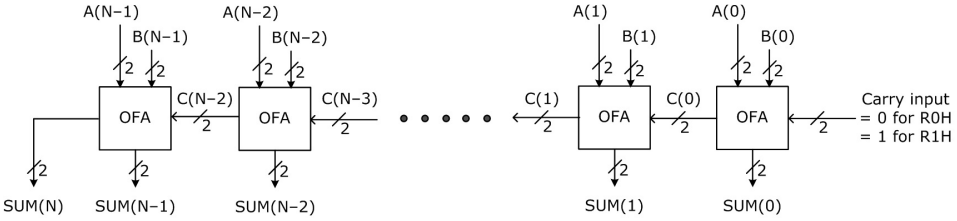

Figure 3. Generic architecture of a dual-rail encoded N-bit asynchronous RCA, realized using cascading one-bit full adders (OFAs).

An OFA computes the sum of two input bits along with any carry input, yielding a sum bit and any carry overflow. The output expressions of the dual-rail encoded OFA, corresponding to R0H, are given by Equations (1)–(4). In these equations, the OFA inputs are represented by (A1, A0) and (B1, B0), alongside the carry input denoted by (CI1, CI0). The sum output is labeled as (SUM1, SUM0), and the carry output, also known as the carry overflow arising from the addition, is denoted by (CO1, CO0).

$$SUM1 = A0B0CI1 + A0B1CI0 + A1B0CI0 + A1B1CI1 \quad (1)$$

$$SUM0 = A0B0CI0 + A0B1CI1 + A1B0CI1 + A1B1CI0 \quad (2)$$

$$CO1 = A0B1CI1 + A1B0CI1 + A1B1CI0 + A1B1CI1 \quad (3)$$

$$CO0 = A0B0CI0 + A0B0CI1 + A0B1CI0 + A1B0CI0 \quad (4)$$

Equations (1)–(4) appear in the sum of disjoint products form [28], where the logical conjunction of any two product terms yields null. Equations (1)–(4) also conform to the monotonic cover constraint [25] whereby only one product term becomes active for the application of data. At the circuit level, this translates into the activation of a unique signal path between a primary input and a primary output for data processing thereby avoiding unnecessary switching activity. QDI circuits generally incorporate the monotonic cover constraint in their physical realization. This work also includes the monotonic cover constraint in the monotonic asynchronous circuit realizations.

We shall now survey IOM asynchronous OFAs belonging to QDI and non-QDI types and theoretically estimate the latencies and cycle time of RCAs incorporating different OFAs. However, the theoretical modeling of latency and cycle time of different asynchronous RCAs is only approximate. This is so because the delays of fundamental components such as OFAs and TFAs are alone considered while gate, interconnect, and parasitic delays are discarded for simplicity. Further, the delay of the input register, which is small, is unaccounted for in the theoretical modeling.

The strong indication property is manifest in the OFAs of [29,30]. Further, it is possible to realize a strong indication OFA according to the delay-insensitive minterm synthesis (DIMS) method [31]. When these OFAs are duplicated and interconnected to form corresponding N-bit RCAs, with N denoting the size of the adder, these RCAs would have identical forward latency and reverse latency, given by $O[N \times D_{OFA}]$; D_{OFA} denotes the propagation delay of an OFA. RCAs incorporating strong indication OFAs have high forward and reverse latencies due to the maximum carry propagation encountered for the

processing of data and spacer. As a consequence, the cycle time of such RCAs amounts to $O[2 \times N \times D_{OFA}]$, leading to notably slow speed performance.

The OFAs referenced in [32–34] exhibit characteristics of weak indication behavior. Further, it is possible to implement a weak indication OFA based on the DIMS method [31]. By employing the weak indication OFAs discussed in [31,32] to construct N-bit RCAs, the cycle time of such RCAs would be $O[2 \times N \times D_{OFA}]$, with equal forward latency and reverse latency of $O[N \times D_{OFA}]$. This arises from the extensive carry propagation encountered during data and spacer processing. In contrast, RCAs utilizing the weak indication OFAs presented in [33,34] would have a cycle time of $O[(N + 2) \times D_{OFA}]$, which results from a forward latency of $O[N \times D_{OFA}]$ and a reverse latency of $O[2 \times D_{OFA}]$. The reverse latency is reduced due to a biased weak indication since the sum output of the OFAs is endowed with the responsibility of indicating (i.e., acknowledging the receipt of) all the adder's inputs and the carry output is exempt from the indication.

In [35], three weak indication OFAs were introduced, based on the binary sorting networks (SN) framework. These OFAs were called SN full adder, SNX full adder, and SNFC full adder. Reference [35], however, did not present any physical implementation of these OFAs. An N-bit RCA realized using the SN OFA results in a cycle time, given by $O[2 \times N \times D_{OFA}]$, indicating equality of forward latency and reverse latency. N-bit RCAs constructed with SNX and SNFC OFAs have a cycle time of $O[(N + 2) \times D_{OFA}]$, suggesting a reduction in reverse latency compared to the forward latency. The reduced reverse latency is ascribed to the biased weak indication characteristic of SNX and SNFC OFAs.

Reference [36] introduced an early output QDI OFA. Upon replication and interconnection of this OFA to construct an N-bit RCA, the RCA would have a forward latency of $O[N \times D_{OFA}]$ and a reduced reverse latency of $O[2 \times D_{OFA}]$. The reduction in reverse latency results from the early output property of the OFA manifested during spacer processing. Hence, the RCA comprising the early output QDI OFA of [36] has a cycle time, expressed by $O[(N + 2) \times D_{OFA}]$.

The early output QDI OFAs discussed in [37] can be used to realize relative timed RCAs. Among the two OFAs presented, one is tailored for area optimization, while the other is tailored for latency optimization. The resulting RCAs would feature a forward latency of $O[N \times D_{OFA}]$ while achieving a minimum reverse latency of $O[D_{OFA}]$. The optimization in reverse latency is facilitated by enabling all OFAs within the RCA to simultaneously generate the sum output without awaiting the carry input during spacer processing. Thus, the cycle time of these relative-timed RCAs is given by $O[(N + 1) \times D_{OFA}]$, representing a theoretically optimal speed performance. Recently, a monotonic OFA was presented in [38] which when replicated to realize an IOM asynchronous RCA has a forward latency of $O[N \times D_{OFA}]$ and an optimal reverse latency of $O[D_{OFA}]$, thus enabling a theoretically optimal cycle time of $O[(N + 1) \times D_{OFA}]$. Compared to [29–37], ref. [38] demonstrated reductions in all the standard design parameters such as area, cycle time, and power dissipation for both R0H and R1H.

4. IOM Asynchronous RCAs with Two-Bit Full Adders

An N-bit RCA can be realized using (N/2) TFAs instead of N OFAs, where N is even, as shown in Figure 4. Thus, the number of addition stages is halved in an RCA comprising TFAs compared to an RCA consisting of OFAs.

If N is odd, a combination of TFAs and OFAs may be used to realize an N-bit RCA. A binary TFA computes the sum of two pairs of input bits along with any carry input, resulting in two sum bits and any carry overflow. The truth table of a dual-rail encoded TFA corresponding to R0H is shown in Table 1. The truth table for R1H can be derived likewise, and this is left to the interest of the reader. In Table 1, the most significant TFA input pair is denoted by (A11, A10) and (B11, B10), and the least significant TFA input pair is denoted by (A01, A00) and (B01, B00). The carry input is represented by (C01, C00). The sum outputs are specified as (SUM11, SUM10) and (SUM01, SUM00), with the former

being the most significant and the latter being the least significant. The carry output, which is the carry overflow from the addition, is designated as (C21, C20).

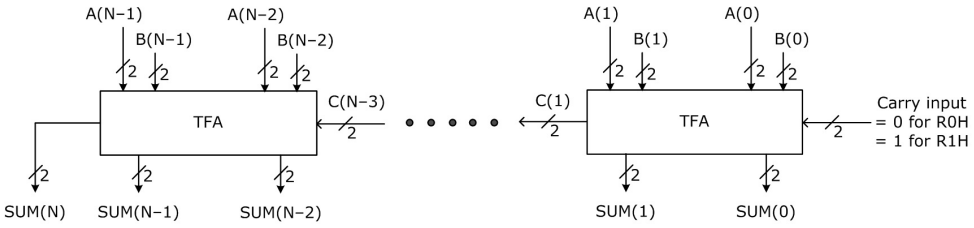

Figure 4. The generic architecture of a dual-rail encoded N-bit asynchronous RCA, constructed using two-bit full adders (TFAs).

Table 1. The truth table of the dual-rail encoded asynchronous two-bit full adder (TFA) w.r.t R0H.

TFA Inputs					TFA Outputs		
(A11, A10)	**(A01, A00)**	**(B11, B10)**	**(B01, B00)**	**(C01, C00)**	**(C21, C20)**	**(SUM11, SUM10)**	**(SUM01, SUM00)**
(0, 1)	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(0, 1)
(0, 1)	(0, 1)	(0, 1)	(0, 1)	(1, 0)	(0, 1)	(0, 1)	(1, 0)
(0, 1)	(0, 1)	(0, 1)	(1, 0)	(0, 1)	(0, 1)	(0, 1)	(1, 0)
(0, 1)	(0, 1)	(0, 1)	(1, 0)	(1, 0)	(0, 1)	(1, 0)	(0, 1)
(0, 1)	(0, 1)	(1, 0)	(0, 1)	(0, 1)	(0, 1)	(1, 0)	(0, 1)
(0, 1)	(0, 1)	(1, 0)	(0, 1)	(1, 0)	(0, 1)	(1, 0)	(1, 0)
(0, 1)	(0, 1)	(1, 0)	(1, 0)	(0, 1)	(0, 1)	(1, 0)	(1, 0)
(0, 1)	(0, 1)	(1, 0)	(1, 0)	(1, 0)	(1, 0)	(0, 1)	(0, 1)
(0, 1)	(1, 0)	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(1, 0)
(0, 1)	(1, 0)	(0, 1)	(0, 1)	(1, 0)	(0, 1)	(1, 0)	(0, 1)
(0, 1)	(1, 0)	(0, 1)	(1, 0)	(0, 1)	(0, 1)	(1, 0)	(0, 1)
(0, 1)	(1, 0)	(0, 1)	(1, 0)	(1, 0)	(0, 1)	(1, 0)	(1, 0)
(0, 1)	(1, 0)	(1, 0)	(0, 1)	(0, 1)	(0, 1)	(1, 0)	(1, 0)
(0, 1)	(1, 0)	(1, 0)	(0, 1)	(1, 0)	(1, 0)	(0, 1)	(0, 1)
(0, 1)	(1, 0)	(1, 0)	(1, 0)	(0, 1)	(1, 0)	(0, 1)	(0, 1)
(0, 1)	(1, 0)	(1, 0)	(1, 0)	(1, 0)	(1, 0)	(0, 1)	(1, 0)
(1, 0)	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(1, 0)	(0, 1)
(1, 0)	(0, 1)	(0, 1)	(0, 1)	(1, 0)	(0, 1)	(1, 0)	(1, 0)
(1, 0)	(0, 1)	(0, 1)	(1, 0)	(0, 1)	(0, 1)	(1, 0)	(1, 0)
(1, 0)	(0, 1)	(0, 1)	(1, 0)	(1, 0)	(1, 0)	(0, 1)	(0, 1)
(1, 0)	(0, 1)	(1, 0)	(0, 1)	(0, 1)	(1, 0)	(0, 1)	(0, 1)
(1, 0)	(0, 1)	(1, 0)	(0, 1)	(1, 0)	(1, 0)	(0, 1)	(1, 0)
(1, 0)	(0, 1)	(1, 0)	(1, 0)	(0, 1)	(1, 0)	(0, 1)	(1, 0)
(1, 0)	(0, 1)	(1, 0)	(1, 0)	(1, 0)	(1, 0)	(1, 0)	(0, 1)
(1, 0)	(1, 0)	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(1, 0)	(1, 0)
(1, 0)	(1, 0)	(0, 1)	(0, 1)	(1, 0)	(1, 0)	(0, 1)	(0, 1)
(1, 0)	(1, 0)	(0, 1)	(1, 0)	(0, 1)	(1, 0)	(0, 1)	(0, 1)
(1, 0)	(1, 0)	(0, 1)	(1, 0)	(1, 0)	(1, 0)	(0, 1)	(1, 0)
(1, 0)	(1, 0)	(1, 0)	(0, 1)	(0, 1)	(1, 0)	(0, 1)	(1, 0)
(1, 0)	(1, 0)	(1, 0)	(0, 1)	(1, 0)	(1, 0)	(1, 0)	(0, 1)
(1, 0)	(1, 0)	(1, 0)	(1, 0)	(0, 1)	(1, 0)	(1, 0)	(0, 1)
(1, 0)	(1, 0)	(1, 0)	(1, 0)	(1, 0)	(1, 0)	(1, 0)	(1, 0)

The output expressions of the dual-rail encoded TFA, corresponding to R0H, are given by Equations (5)–(10), which correspond to the DSOP form.

$$C21 = A10A00B11B01C01 + A11A00B10B01C01 + A10A01B11B00C01 + A11A01B10B00C01 + A10A01B11B01 + A11A01B10B01 + A11B11 \quad (5)$$

$$C20 = A11A01B10B00C00 + A10A01B11B00C00 + A11A00B10B01C00 + A10A00B11B01C00 + A11A00B10B00 + A10A00B11B00 + A10B10 \quad (6)$$

$$\begin{aligned}\text{SUM11} = {} & \text{A11A01B10B00C00} + \text{A10A01B11B00C00} + \text{A11A00B10B01C00} + \text{A10A00B11B01C00} + \\ & \text{A11A00B11B01C01} + \text{A11A01B11B00C01} + \text{A10A00B10B01C01} + \text{A10A01B10B00C01} + \text{A10A01B10B01} + \\ & \text{A11A00B10B00} + \text{A10A00B11B00} + \text{A11A01B11B01}\end{aligned} \quad (7)$$

$$\begin{aligned}\text{SUM10} = {} & \text{A11A01B10B00C01} + \text{A10A01B11B00C01} + \text{A11A00B10B01C01} + \text{A10A00B11B01C01} + \\ & \text{A10A01B10B00C00} + \text{A10A00B10B01C00} + \text{A11A01B11B00C00} + \text{A11A00B11B01C00} + \text{A11A00B11B00} + \\ & \text{A11A01B10B01} + \text{A10A01B11B01} + \text{A10A00B10B00}\end{aligned} \quad (8)$$

$$\text{SUM01} = \text{A01B00C00} + \text{A00B01C00} + \text{A00B00C01} + \text{A01B01C01} \quad (9)$$

$$\text{SUM00} = \text{A00B01C01} + \text{A01B00C01} + \text{A01B01C00} + + \text{A00B00C00} \quad (10)$$

IOM asynchronous TFAs corresponding to weak indication were first introduced in [39], serving as a basis for constructing RCAs that have identical forward and reverse latencies of $O[(N/2) \times D_{TFA}]$. The forward latency and reverse latency are maximum due to the extensive carry propagation incurred given the presence of the C-element in the critical path of such TFAs. Hence, the cycle time of weak indication RCAs employing those TFAs is specified as $O[N \times D_{TFA}]$, given that an N-bit RCA constructed with TFAs (assuming N is even) requires N/2 TFAs. As an advancement over the TFA design put forward in [39], early output QDI TFAs were introduced in [40]. A couple of methods for encoding the TFA inputs and outputs were also explored in [40], including homogeneous encoding, i.e., using one type of coding scheme such as say the dual-rail (1-of-2) code, and heterogeneous encoding, i.e., using a combination of coding schemes such as 1-of-4 code and dual-rail code. The 1-of-4 code is used to simultaneously encode two input/output variables while the dual-rail (1-of-2) code is used to encode one input/output variable. Nevertheless, it was observed in [40] that homogeneous (dual-rail) encoding is preferable to heterogeneous encoding due to its ability to achieve relatively reduced latency, area, and power dissipation. The forward latency of an N-bit RCA utilizing the early output QDI TFA of [40] is given by $O[(N/2) \times D_{TFA}]$, while the reverse latency is less and given by $O[2 \times D_{TFA}]$. Consequently, the cycle time of an N-bit RCA utilizing the early output QDI TFA is given by $O[(N/2 + 2) \times D_{TFA}]$, which is less compared to the cycle time of the RCA utilizing the weak indication TFA from [39].

Given the above, there are two motivations behind this research. First, it is possible to reduce the D_{TFA} of [40] by resorting to a monotonic implementation instead of a QDI implementation, and second, it is possible to achieve a theoretically optimal cycle time of $O[(N/2 + 1) \times D_{TFA}]$ with a monotonic implementation. Hence, there arises scope for designing an IOM asynchronous RCA with TFAs that would achieve lesser cycle time than existing IOM asynchronous RCAs. In this context, the proposed monotonic implementation of an IOM asynchronous TFA is shown in Figure 5, which is dual-rail encoded and corresponds to R0H. In Figure 5, (A01, A00) and (B01, B00) represent an input bit-pair while (A11, A10) and (B11, B10) represents a more significant input bit-pair. Among the sum bits, (SUM11, SUM10) is more significant than (SUM01, SUM00). (C01, C00) represents the carry input, and (C21, C20) represents the carry output/overflow.

Interestingly, the proposed design does not contain any C-elements contrary to the counterpart designs presented in [39,40] and contains 13 simple gates and 12 complex gates. The complex gate types include AO21, AO22, and AO222. To realize an equivalent monotonic TFA that corresponds to R1H, all the gates shown in Figure 5 should be replaced by their respective duals—in other words, AND and OR gates shown in Figure 5 should be replaced by OR and AND gates, respectively; and AO21, AO22, and AO222 complex gates should be replaced by OA21, OA22, and OA222 complex gates, respectively. In general, the simple and/or complex logic gates in an asynchronous circuit (excluding the C-elements) that correspond to R0H should be replaced by their respective logic duals to obtain an equivalent circuit that corresponds to R1H, and vice versa [41].

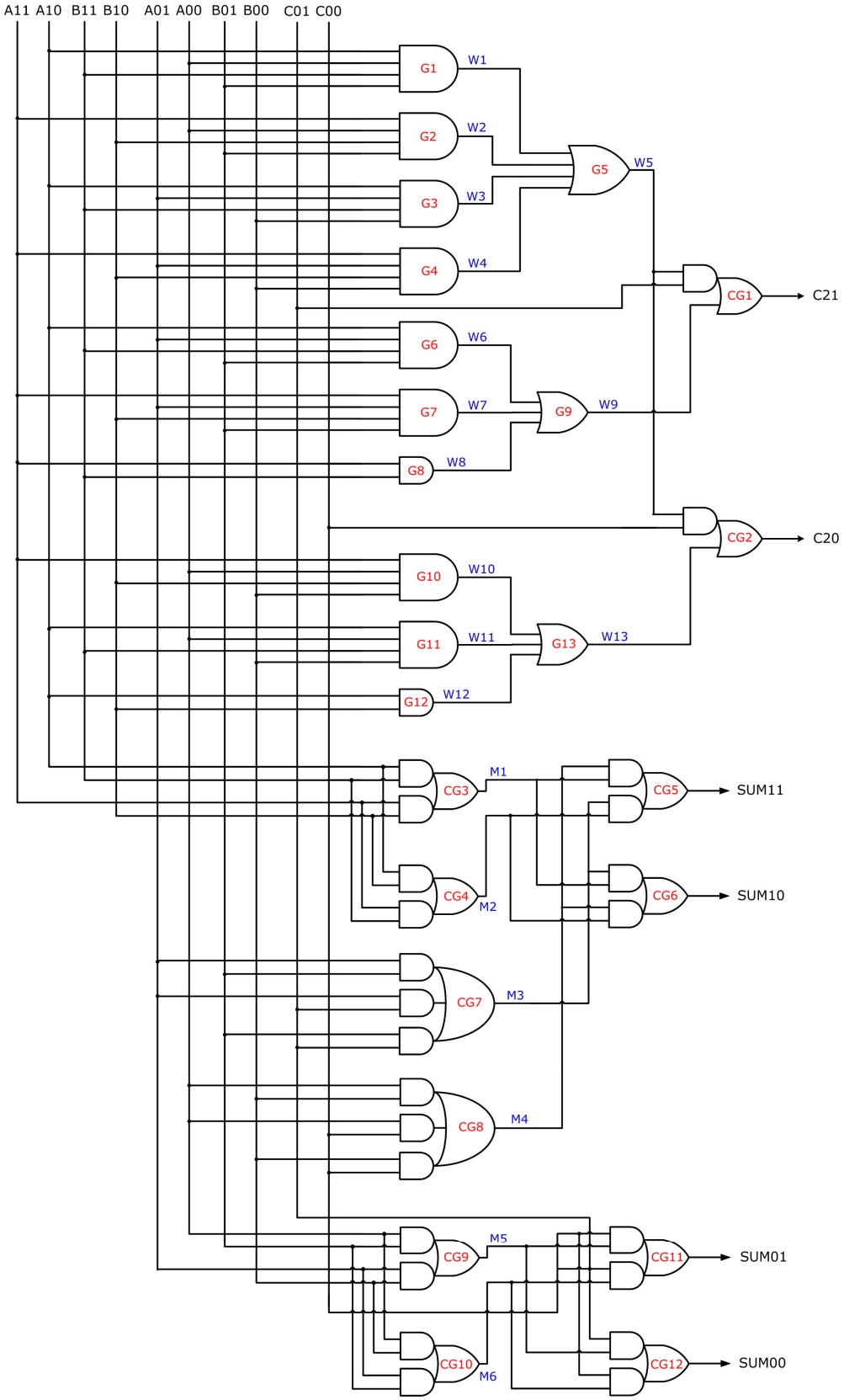

Figure 5. Proposed monotonic asynchronous two-bit full adder (TFA), corresponding to R0H. G1, G2, G3, G4, G6, G7, G8, G10, G11, G12 are AND gates; G5, G9, G13 are OR gates; CG1, CG2 are AO21 complex gates; CG3, CG4, CG5, CG6, CG9, CG10, CG11, CG12 are AO22 complex gates; and CG7, CG8 are AO222 complex gates.

Based on Figure 5, we shall discuss the operation of the proposed TFA with respect to R0H here. It would not be difficult to comprehend the TFA operation with respect to R1H and so this is left to the reader's interest. To aid with the explanation of the proposed TFA operation, in Figure 5, the simple gates are marked with labels G1 to G13 in red, and the complex gates are marked with labels CG1 to CG12 in red. W1 to W13 and M1 to M6 highlighted in blue signify the internal/intermediate TFA outputs.

We shall now consider comprehensive example scenarios for each of the primary outputs of the proposed TFA to describe its monotonic behavior and early output property, manifested during data and spacer processing.

4.1. Carry Output of TFA—(C21, C20)

- Scenario 1: When data are supplied, if A10 = A00 = B11 = B01 = 1, or if A11 = A00 = B10 = B01 = 1, or if A10 = A01 = B11 = B00 = 1, or if A11 = A01 = B10 = B00 = 1, W1 or W2 or W3 or W4 would assume 1, respectively. Consequently, W5 would assume 1. Under this condition, if the carry input C01 = 1, carry output C21 would assume 1 or if C00 = 1, C20 would assume 1. Thus, the signal transitions rise monotonically from the primary inputs to the carry output for data processing. In the next spacer phase, if A10/A00/B11/B01 assumes 0 (if these were 1 earlier), or if A11/A00/B10/B01 assumes 0 (if these were 1 earlier), or if A10/A01/B11/B00 assumes 0 (if these were 1 earlier) or if A11/A01/B10/B00 assumes 0 (if these were 1 earlier), W1 or W2 or W3 or W4 (whichever was 1 earlier) would, respectively assume 0. Consequently, W5 would assume 0. Under this condition, regardless of C01 or C00 assuming 0, C21 or C20 (whichever was 1 earlier) would assume 0 in an early output manner. Hence, the signal transitions monotonically fall from the primary inputs to the carry output for spacer processing.
- Scenario 2: When data are supplied, if A10 = A01 = B11 = B01 = 1 (A11 = A00 = B10 = B00 = 1), or if A11 = A01 = B10 = B01 = 1 (A10 = A00 = B11 = B00 = 1), or if A11 = B11 = 1 (A10 = B10 = 1), W6 or W7 or W8 (W10 or W11 or W12) would assume 1, respectively. Consequently, W9 (W13) would assume 1, and subsequently C21 (C20) would assume 1. Again, the signal transitions rise monotonically from the primary inputs to the carry output for data processing. In the next spacer phase, if A10/A01/B11/B01 (A11/A00/B10/B00) assumes 0 if either of these combinations were 1 earlier, or if A11/A01/B10/B01 (A10/A00/B11/B00) assumes 0 if either of these combinations were 1 earlier, or if A11/B11 (A10/B10) assumes 0 if either of these combinations were 1 earlier, W6 or W7 or W8 (W10 or W11 or W12) whichever was 1 earlier would, respectively assume 0 in an early output manner. Consequently, W9 (W13) would assume 0, and subsequently C21 (C20) would assume 0. Thus, the signal transitions fall monotonically from the primary inputs to the carry output for spacer processing.

4.2. Most Significant Sum Output of TFA—(SUM11, SUM10)

- Scenario 1: When data are supplied if A10 = B11 = A00 = B00 = 1, or if A11 = B10 = A00 = B00 = 1, or if A10 = B11 = A00 = C00 = 1, or if A11 = B10 = A00 = C00 = 1, or if A10 = B11 = B00 = C00 = 1, or if A11 = B10 = B00 = C00 = 1, M1 and M4 would assume 1, and therefore SUM11 would assume 1. Alternatively, if A10 = B10 = A01 = B01 = 1, or if A11 = B11 = A01 = B01 = 1, or if A10 = B10 = A01 = C01 = 1, or if A11 = B11 = A01 = C01 = 1, or if A10 = B10 = B01 = C01 = 1, or if A11 = B11 = B01 = C01 = 1, M2 and M3 would assume 1, and therefore SUM11 would assume 1. Thus, the signal transitions rise monotonically from the primary inputs to the most significant sum output for data processing. In the next spacer phase, if M1 or M4, whichever was 1 earlier, assumes 0, or if M2 or M3, whichever was 1 earlier, assumes 0, SUM11 would assume 0 in an early output manner. Thus, the signal transitions fall monotonically from the primary inputs to the most significant sum output for spacer processing.

- Scenario 2: When data are given, if A10 = B10 = A00 = B00 = 1, or if A11 = B11 = A00 = B00 = 1, or if A10 = B10 = A00 = C00 = 1, or if A11 = B11 = A00 = C00 = 1, or if A10 = B10 = B00 = C00 = 1, or if A11 = B11 = B00 = C00 = 1, M2 and M4 would assume 1, and therefore SUM10 would assume 1. Alternatively, if A10 = B11 = A01 = B01 = 1, or if A11 = B10 = A01 = B01 = 1, or if A10 = B11 = A01 = C01 = 1, or if A11 = B10 = A01 = C01 = 1, or if A10 = B11 = B01 = C01 = 1, or if A11 = B10 = B01 = C01 = 1, M1 and M3 would assume 1, and therefore SUM10 would assume 1. Thus, the signal transitions rise monotonically from the primary inputs to the most significant sum output for data processing. In the next spacer phase, if M2 or M4, whichever was 1 earlier, assumes 0, or if M1 or M3, whichever was 1 earlier, assumes 0, SUM10 would assume 0 in an early output manner. Hence, the signal transitions fall monotonically from the primary inputs to the most significant sum output for spacer processing.

4.3. Least Significant Sum Output of TFA—(SUM01, SUM00)

- Scenario 1: When data are supplied, if A01 = B00 = 1, or if A00 = B01 = 1, M5 would assume 1 and if C00 = 1, SUM01 would assume 1. Alternatively, if A00 = B00 = 1, or if A01 = B01 = 1, M6 would assume 1 and if C01 = 1, SUM01 would assume 1. Thus, the signal transitions rise monotonically from the primary inputs to the least significant sum output for data processing. In the next spacer phase, if A01/B00 assumes 0 (if these were 1 earlier), or if A00/B01 assumes 0 (if these were 1 earlier), M5 would assume 0, and regardless of C00 assuming 0 (if it was 1 earlier), SUM01 would assume 0 in an early output manner. Alternatively, if A00/B00 assumes 0 (if these were 1 earlier), or if A01/B01 assumes 0 (if these were 1 earlier), M6 would assume 0, and regardless of C01 assuming 0 (if it was 1 earlier), SUM01 would assume 0 in an early output manner. Hence, the signal transitions fall monotonically from the primary inputs to the least significant sum output for spacer processing.
- Scenario 2: When data are given, if A00 = B01 = 1, or if A01 = B00 = 1, M5 would assume 1 and if C01 = 1, SUM00 would assume 1. Alternatively, if A00 = B00 = 1, or if A01 = B01 = 1, M6 would assume 1 and if C00 = 1, SUM00 would assume 1. Thus, the signal transitions rise monotonically from the primary inputs to the least significant sum output for data processing. In the next spacer phase, if A00/B01 assumes 0 (if these were 1 earlier), or if A01/B00 assumes 0 (if these were 1 earlier), M5 would assume 0, and regardless of C01 assuming 0 (if it was 1 earlier), SUM00 would assume 0 in an early output manner. Alternatively, if A00/B00 assumes 0 (if these were 1 earlier), or if A01/B01 assumes 0 (if these were 1 earlier), M6 would assume 0, and regardless of C00 assuming 0 (if it was 1 earlier), SUM00 would assume 0 in an early output manner. Hence, the signal transitions fall monotonically from the primary inputs to the least significant sum output for spacer processing.

The proposed TFA was replicated and linked in series to construct a 32-bit RCA corresponding to R0H and R1H individually, which were implemented physically using a 28 nm CMOS standard digital cell library [42]. One stage of the architecture depicted in Figure 1a was considered, comprising input registers, a completion detector, and a 32-bit RCA representing the IOM asynchronous circuit. Random input vectors were supplied to perform functional simulations, and representative screenshots of a segment of simulation waveforms corresponding to R0H and R1H are illustrated in Figures 6 and 7, respectively.

Figure 6 illustrates a 32-bit addition in hexadecimal format, corresponding to R0H. Here, one set of dual-rail encoded inputs ranges from (X311, X310) to (X01, X00), while the other set ranges from (Y311, Y310) to (Y01, Y00). The resulting dual-rail encoded 33-bit sum output is denoted by (SUM321, SUM320) to (SUM01, SUM00). A_R0H represents a 32-bit input bus containing dual-rail encoded input rails (X311, X301, X291, . . ., X01), and B_R0H denotes another 32-bit input bus consisting of dual-rail encoded input rails (Y311, Y301, Y291, . . ., Y01). SUM_R0H signifies the 33-bit sum bus containing dual-rail encoded output rails (SUM321, SUM311, SUM301, . . ., SUM01). As depicted in Figure 6, a zero spacer is

observed between two input data sets. Consequently, the spacer sum of (0 0000 0000) is displayed between two other sums in the figure.

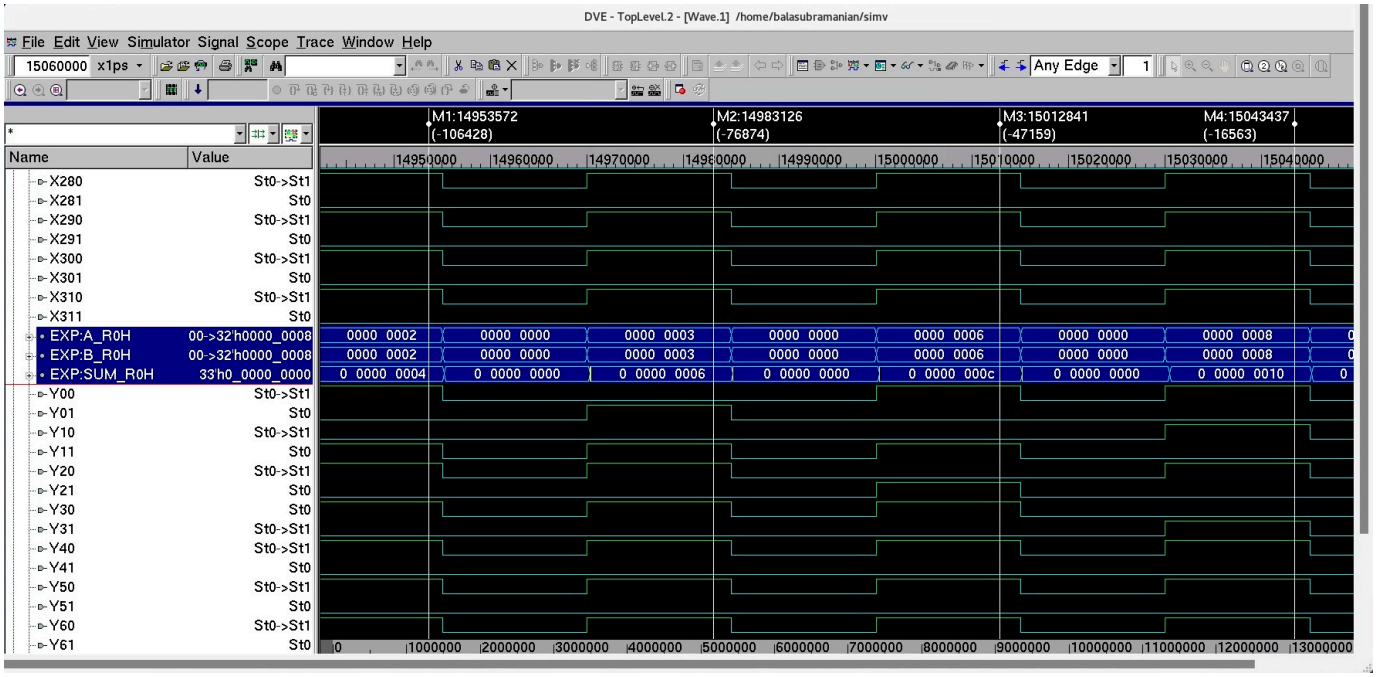

Figure 6. Screenshot of a segment of simulation waveforms corresponding to 32-bit addition, following R0H. Markers M1 to M4 point to different addition instances. A_R0H and B_R0H represent the 32-bit input buses and SUM_R0H represents the 33-bit sum output bus, given in hexadecimal.

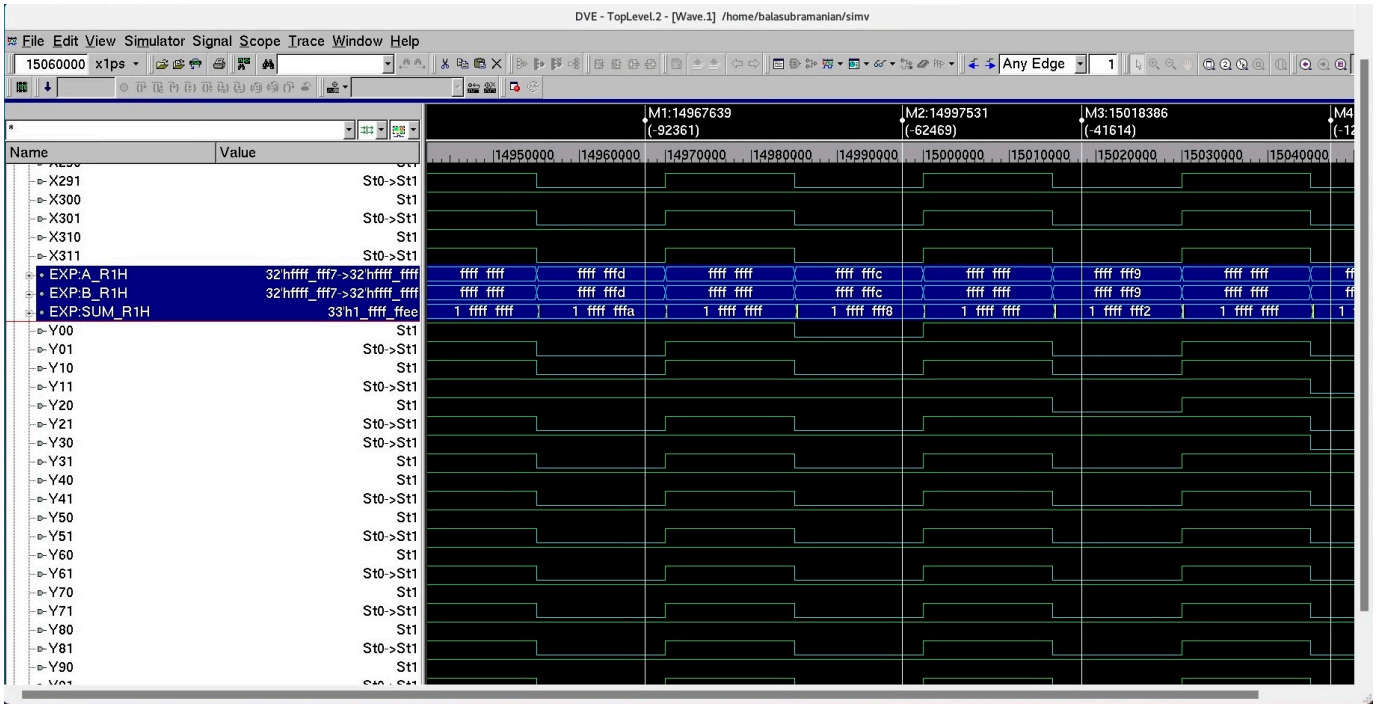

Figure 7. Screenshot of a segment of simulation waveforms corresponding to 32-bit addition, following R1H. Markers M1 to M3 point to different addition instances. A_R1H and B_R1H represent the 32-bit input buses and SUM_R1H represents the 33-bit sum output bus, given in hexadecimal.

Figure 7 depicts a 32-bit addition in hexadecimal format, corresponding to R1H. A_R1H represents a 32-bit input bus containing dual-rail encoded input rails (X310, X300, X290, . . ., X00), while B_R1H denotes another 32-bit input bus consisting of dual-rail encoded input rails (Y310, Y300, Y290, . . ., Y00). SUM_R1H signifies the 33-bit sum bus comprising dual-rail encoded output rails (SUM320, SUM310, SUM300, . . ., SUM00). As observed in Figure 7, the one's spacer is visible between two input data. Consequently, the spacer sum of (1 FFFF FFFF) is displayed between two other sums in the figure.

5. Results

For implementation, one IOM asynchronous pipeline stage was constructed, comprising input-side registers and an asynchronous circuit (i.e., a 32-bit RCA), as illustrated in Figure 1a. Ackin was presumed to be provided by the environment. The RCA serves as a basis for assessing the efficiency of different OFAs and TFAs including the proposed TFA. Thus, several RCAs were implemented by independently duplicating and linking various OFAs and TFAs, corresponding to R0H and R1H separately. The carry input to the least significant OFA and TFA in the RCAs is set to 0. This was carried out for R0H by joining one encoded rail of the carry input (C1) to a tie-to-low library cell and joining the other encoded rail of the carry input (C0) to Ackin. Consequently, when Ackin = 1, C1 = 0, and C0 = 1, implying that a zero-carry input is provided to the least significant OFA or TFA in the RCA. Conversely, when Ackin = 0, C1 = C0 = 0 signifies the provision of the spacer as the carry input to the least significant OFA or TFA in the RCA. For R1H, the encoded rail C1 of the carry input was joined to a tie-to-high library cell, and the encoded rail C0 of the carry input was joined to Ackin. As a result, when Ackin = 1, C1 = C0 = 1 implies the provision of the (one's) spacer as the carry input to the least significant OFA or TFA in the RCA. When Ackin = 0, C1 = 1, and C0 = 0 implies that a carry input of 0 is given to the least significant OFA or TFA in the RCA.

To implement OFAs, TFAs, and RCAs, the gates of a 28 nm CMOS standard digital cell library [42] were used. Since the cell library does not include the C-element, the 2-input C-element (as depicted in Figure 1d) was realized to implement various OFAs, TFAs, RCAs, registers, and completion detectors. The asynchronous RCAs comprising OFAs and TFAs were described structurally in Verilog HDL and simulated, and their design parameters were subsequently evaluated.

To evaluate the design characteristics of various asynchronous RCAs, we considered a standard low-leakage typical cell library specification having a supply voltage of 1.05 V and an operating junction temperature of 25 °C. Synopsys EDA tools were utilized for simulations and to gauge the design metrics. The functionality of different RCAs was simulated by VCS by using a test bench that consisted of approximately 1000 random data inputs along with an identical number of spacer inputs. Two test benches were utilized, one for R0H and an equivalent one for R1H. The test bench was supplied at a latency of 15 ns to accommodate the slowest RCA. The (total) power dissipation was estimated by considering the switching activity captured through functional simulations. Power dissipation was estimated through PrimePower. The default wire load model was used and a fanout-of-4 drive strength was assigned to all the output ports, i.e., the sum bits of the adders. Timing analysis was performed using PrimeTime (version: vO-2018-06-SP5-2). A virtual clock was used to constrain the adder inputs and outputs. The clock, being virtual, was not physically used, and thus it does not account for the design metrics. The forward latency of IOM asynchronous RCAs, which is equivalent to the critical path delay of synchronous RCAs, was directly estimated while the reverse latency was estimated based on the corresponding gate and net delays specified in the timing reports. As mentioned earlier, the cycle time, signifying the time taken to complete one data transaction, was estimated as the sum of forward latency and reverse latency.

Table 2 presents the design metrics of IOM asynchronous RCAs implemented using diverse OFAs and TFAs. The design parameters estimated include forward latency, reverse latency, cycle time, area, and power dissipation. The input-side registers and completion detector are consistent across all the RCAs for R0H and R1H, with the sole difference attributed to the underlying adder logic. Hence, disparities in the design parameters of RCAs are entirely ascribed to variations in OFA/TFA logic. For clarity and discussion of results, adder legends AZ1 to AZ17 for R0H and AO1 to AO17 for R1H are assigned to the RCAs referenced in Table 2. AZ1 to AZ14 (AO1 to AO14) signify RCAs realized using OFAs corresponding to R0H (R1H), and AZ15 to AZ17 (AO15 to AO17) signify RCAs realized using TFAs corresponding to R0H (R1H).

Table 2. Design parameters of 32-bit IOM asynchronous RCAs comprising different OFAs/TFAs corresponding to R0H and R1H implemented using a 28 nm CMOS process.

Reference for OFA/TFA; RCA Legend	Timing Parameters (ns)			Area (μm^2)	Power (μW)
	Forward Latency	Reverse Latency	Cycle Time		
Based on return-to-zero handshaking (R0H)					
[29]; AZ1 [a]	14.70	14.70	29.40	2518.32	1446
[31]; AZ2 [a]	9.34	9.34	18.68	2493.93	1449
[31]; AZ3 [b]	8.31	8.31	16.62	2412.60	1445
[30]; AZ4 [a]	9.12	9.12	18.24	2282.47	1429
[32]; AZ5 [b]	7.07	7.07	14.14	2005.96	1415
[33]; AZ6 [b]	4.52	0.74	5.26	2087.28	1431
[34]; AZ7 [b]	3.40	0.82	4.22	2038.49	1421
[36]; AZ8 [c]	3.19	0.70	3.89	1648.12	1405
[37]; AZ9 [c,d]	3.14	0.73	3.87	1534.27	1396
[37]; AZ10 [c,e]	3.02	0.72	3.74	1648.12	1403
[35]; AZ11 [b,f]	8.97	8.97	17.94	2103.55	1424
[35]; AZ12 [b,g]	6.64	1.42	8.06	2282.47	1451
[35]; AZ13 [b,h]	6.20	1.04	7.24	2339.40	1437
[38]; AZ14 [i]	2.87	0.48	3.35	1387.88	1381
[39]; AZ15 [j]	4.29	4.29	8.58	2855.84	1480
[40]; AZ16 [k]	2.31	0.88	3.19	2477.66	1446
AZ17 (Proposed)	1.95	0.53	2.48	1778.24	1399
Based on return-to-one handshaking (R1H)					
[29]; AO1 [a]	14.24	14.24	28.48	2518.32	1445
[31]; AO2 [a]	8.84	8.84	17.68	2363.80	1443
[31]; AO3 [b]	8.12	8.12	16.24	2347.53	1442
[30]; AO4 [a]	8.97	8.97	17.94	2282.47	1429
[32]; AO5 [b]	7.04	7.04	14.08	2005.96	1415
[33]; AO6 [b]	3.88	0.73	4.61	2087.28	1431
[34]; AO7 [b]	3.39	0.81	4.20	2038.49	1421
[36]; AO8 [c]	3.02	0.70	3.72	1648.12	1404
[37]; AO9 [c,d]	3.16	0.72	3.88	1534.27	1395
[37]; AO10 [c,e]	2.99	0.70	3.69	1648.12	1402
[35]; AO11 [b,f]	9.05	9.05	18.10	2103.55	1424
[35]; AO12 [b,g]	6.75	1.19	7.94	2282.47	1456
[35]; AO13 [b,h]	6.31	1.03	7.34	2339.40	1437
[38]; AO14 [i]	2.85	0.48	3.33	1387.88	1380
[39]; AO15 [j]	4.19	4.19	8.38	3083.55	1485
[40]; AO16 [k]	2.31	0.88	3.19	2705.38	1451
AO17 (Proposed)	1.84	0.54	2.38	1892.10	1400

[a] Uses strong-indication OFA; [b] Uses weak-indication OFA. [c] Uses early output OFA ([d] AOPT_EO_FA, [e] LOPT_EO_FA of [37]) to realize a relative-timed RCA. [f] Uses SN full adder, [g] SNX full adder, and [h] SNFC full adder of [35]. [i] Uses monotonic OFA. [j] Uses weak-indication TFA. [k] Uses early output QDI TFA.

From Table 2, it is seen that the RCAs incorporating the proposed TFA (AZ17 and AO17) have less forward latency and reverse latency and thus a reduced cycle time compared to RCAs incorporating OFAs or other TFAs for R0H and R1H. The reasons for this shall be explained now. AZ1, AZ2, and AZ4 (AO1, AO2, and AO4 w.r.t R1H) comprise strong-indication OFAs and so their forward and reverse latencies shall be equally high which contributes to a greater cycle time. AZ3, AZ5, and AZ11 (AO3, AO5, and AO11 w.r.t R1H) consist of weak-indication OFAs but their forward and reverse latencies are still equal and rather high, and this is due to the non-distributed and unbiased nature of the weak indication phenomenon present in them. AZ6, AZ7, AZ12, and AZ13 (AO6, AO7, AO12, and AO13 w.r.t R1H) are realized using OFAs which feature a distributed/biased weak indication. As a result, their reverse latency is significantly less than their forward latency. However, due to many logic elements present in their critical path and/or due to the presence of N stages in the RCA (here, N = 32), their cycle time is rather high compared to AZ17 (AO17) which has only N/2 stages. The same reason is partly responsible for the moderately high cycle time of AZ9 (AO9 w.r.t R1H) which comprises an early output QDI OFA. AZ9 and AZ10 (AO9 and AO10 w.r.t R1H) are relative-timed RCAs realized using area-optimized and latency-optimized early output QDI OFAs. On the contrary, AZ14 (AO14 w.r.t R1H) is a monotonic RCA realized using a monotonic OFA. Given these, although AZ9 and AZ10 (AO9 and AO10) tend to enable a theoretically optimal cycle as AZ14 (AO14), the latter achieved an edge in terms of all the design metrics. This has been possible because AZ9 and AZ10 (AO9 and AO10) incorporate C-elements in their logic while AZ14 (AO14) does not. Moreover, AZ14 (AO14) requires fewer logic elements and a relatively shorter critical path compared to AZ9 and AZ10 (AO9 and AO10). From Table 2, it is observed that compared to the best of the existing RCAs constructed using OFAs viz. AZ14, AZ17 incorporating the proposed TFA achieves a 26% reduction in cycle time for R0H, and compared to AO14, AO17 achieves a 28.5% reduction in cycle time for R1H. This is mainly because although AZ14 and AZ17 (AO14 and AO17 w.r.t R1H) are monotonic implementations, AZ17 has half the number of individual adder stages compared to AZ14.

AZ15 and AZ16 (AO15 and AO16 w.r.t R1H) are RCAs constructed using TFAs. AZ15 (AO15 w.r.t R1H) has a weak indication TFA which causes equal forward and reverse latencies and thus a high cycle time. Consequently, despite halving the number of RCA stages compared to an RCA containing OFAs, the cycle time of AZ15 (AO15) is greater than the cycle time of AZ6, AZ7, AZ8, AZ9, AZ10, AZ12, AZ13, and AZ14 (AO6, AO7, AO8, AO9, AO10, AO12, AO13, and AO14 w.r.t R1H), and it occupies more area, and dissipates more power in comparison. On the other hand, AZ16 (AO16 w.r.t R1H) contains an early output QDI TFA which causes a reduced reverse latency compared to the forward latency and results in a reduced cycle time. Hence, AZ16 (AO16) is preferable to AZ15 (AO15). Though AZ16 (AO16) comprising TFAs features a reduced cycle time compared to RCAs incorporating OFAs, it occupies more silicon and dissipates more power in comparison. Nevertheless, there exists scope for designing a better TFA by opting for a monotonic implementation rather than an indicating or early output QDI implementation. In this context, the proposed TFA is found to be beneficial. Particularly, AZ17 (AO17 w.r.t R1H) incorporating the proposed TFA results in forward and reverse latencies that are less than the forward and reverse latencies of AZ1 to AZ16 (AO1 to AO16 w.r.t R1H). This has been possible due to four reasons: (i) AZ17 has half the number of RCA stages (here, 16) compared to a conventional RCA (32 stages), (ii) no C-elements in the proposed TFA logic (shown in Figure 5) compared to AZ15 and AZ16 which have many C-elements, (iii) the monotonic implementation permits the simultaneous production of the spacer by all the TFAs comprising the RCA, resulting in an optimal reverse latency, and (iv) fewer logic elements in its critical path.

The forward latency and reverse latency of RCAs comprising different OFAs have already been (approximately) modeled theoretically in [38]—an interested reader may refer to the same for details, as it is not repeated here. To specifically comment on the reduced cycle time achieved by the RCA incorporating the proposed TFA compared to

RCAs incorporating other TFAs, we provide theoretical modeling of their forward and reverse latencies concerning R0H for an N-bit addition. The theoretical modeling of forward and reverse latencies for R1H can be carried out likewise since only the duals of the gates should be substituted and so it is left to a reader's interest.

The forward and reverse latencies of AZ15 are expressed by Equation (11) since they are the same. The forward and reverse latencies of AZ16 and AZ17 are expressed by Equations (12), (13), and (14), (15), respectively. Equations (11)–(15) are approximate because interconnect and parasitic delays were not accounted for for simplicity. In the equations, $D_{register}$, D_{AND4}, D_{OR4}, D_{OR3}, D_{AO21}, D_{CE2}, D_{AO222}, and D_{AO22} denote the typical propagation delay of a register (which is a 2-input C-element), a 4-input AND gate, a 4-input OR gate, a 3-input OR gate, an AO21 complex gate, a 2-input C-element, an AO222 complex gate, and an AO22 complex gate, respectively.

$$AZ15_{forward\_latency/reverse\_latency} = D_{register} + (D_{AND4} + D_{OR4} + D_{CE2} + D_{OR2}) + [\{(N/2) - 2\} \times (D_{CE2} + D_{OR2})] + (2 \times D_{CE2} + D_{OR2}) \quad (11)$$

$$AZ16_{forward\_latency} = D_{register} + (D_{AND4} + D_{OR4} + D_{AO21}) + [\{(N/2) - 2\} \times D_{AO21}] + (D_{CE2} + D_{OR3}) \quad (12)$$

$$AZ16_{reverse\_latency} = D_{register} + (D_{AND4} + D_{OR4} + D_{AO21}) + (D_{CE2} + D_{OR3}) \quad (13)$$

$$AZ17_{forward\_latency} = D_{register} + (D_{AND4} + D_{OR4} + D_{AO21}) + [\{(N/2) - 2\} \times D_{AO21}] + (D_{AO222} + D_{AO22}) \quad (14)$$

$$AZ17_{reverse\_latency} = D_{register} + (D_{AND4} + D_{OR4} + D_{AO21}) \quad (15)$$

In Equations (11), (12), and (14), on the right side, the first term denotes an input register delay; the second term represents the delay encountered in the least significant TFA to generate the carry output; the third term denotes the delay incurred in carry propagation through $\{(N/2) - 2\}$ TFAs; and the fourth term denotes the delay encountered in the most significant TFA. In Equation (13), on the right side, the first term denotes an input register delay; the second term denotes the delay incurred in generating the spacer as the carry output; and the third term denotes the delay incurred in producing the spacer as the sum output in all the TFAs. In Equation (14), on the right side, the first term denotes an input register delay while the second term denotes the delay incurred in generating the spacer as the carry output, which subsumes the delay incurred in generating the spacer as the sum output in all the TFAs.

Substituting the typical propagation delays of gates from [42], we find that the theoretical forward and reverse latency of AZ15 is 3.038 ns, and hence its cycle time equals 6.076 ns. The theoretical forward and reverse latencies of AZ16 are calculated to be 1.4 ns and 0.518 ns, implying a cycle time of 1.918 ns. The theoretical forward and reverse latencies of the RCA incorporating the proposed TFA viz. AZ17 are 1.399 ns and 0.339 ns, implying a cycle time of 1.738 ns. The theoretical delay modeling predicts a lesser cycle time for AZ17 compared to AZ15 and AZ16, which is confirmed by the practical cycle time estimates given in Table 2. Compared to the best of the existing RCAs comprising TFAs viz. AZ16 (AO16 w.r.t R1H), AZ17 incorporating the proposed TFA (AO17 w.r.t R1H) achieves a 22.3% (25.4%) reduction for R0H (R1H).

Now, analyzing the area, generally, OFAs have substantially reduced logic than TFAs. Hence, an RCA with N OFAs would require less area than an RCA with (N/2) TFAs, where N is even. This is confirmed by the area estimates given in Table 2 for R0H and R1H. For example, the area of the OFA comprising AZ14 is 19.31 μm^2 whereas the area of the proposed TFA comprising AZ17 is 63.03 μm^2, which is more than double the area of the OFA comprising AZ14. Hence, from an area perspective, RCAs comprising OFAs tend to have the upper hand compared to RCAs comprising TFAs. Among the RCAs comprising TFAs, the area of the weak indication TFA comprising AZ15 (AO15) is 130.38 μm^2 (144.61 μm^2), and the area of the early output QDI TFA comprising AZ16 (AO16) is 106.74 μm^2 (120.97 μm^2). In contrast, the area of the proposed monotonic TFA comprising AZ17 (AO17) is less, which is 63.03 μm^2 (70.14 μm^2). The proposed TFA requires fewer gates than the existing weak

indication and early output QDI TFAs and does not include the C-element—this is the reason behind its relatively reduced area.

In modern VLSI designs, cycle time and power dissipation merit greater consideration than the area. Generally, power dissipation does not vary proportionally to the area between two logically equivalent IOM asynchronous circuits if they incorporate the monotonic cover constraint. As mentioned in Section 3, in an IOM asynchronous circuit, one signal path gets activated from a primary input to a primary output satisfying the monotonic cover constraint so unnecessary switching activity is avoided. As a result, despite any significant differences in area, the variations in power dissipation between logically equivalent but structurally different RCAs in Table 2 are not proportionately high. This is one important reason why AZ17 (incorporating the proposed TFA) has only a moderately greater power dissipation than AZ14 (incorporating the better-optimized OFA). From Table 2, it is found that compared to AZ14, AZ17 dissipates just 1.3% more power for R0H while having a 26% reduced cycle time, and compared to AO14, AO17 dissipates just 1.4% more power for R1H while having a 28.5% reduced cycle time. Given the above, from Table 2, considering cycle time and power dissipation, AZ17 (AO17) comprising the proposed TFA is found to be better optimized than RCAs comprising other OFAs and TFAs concerning R0H (R1H). Among AZ17 and AO17, the former requires 6.4% less area while the latter has a 4% reduced cycle time although both these dissipate almost the same power.

In IOM asynchronous circuits, the energy metric, a crucial measure for low-power design [43], is derived from the product of power and cycle time. Minimizing power and cycle time are desirable. Hence, it is desirable to minimize the power-cycle time product (PCTP). The power and cycle time values given in Table 2 were multiplied and then normalized. The resulting normalized PCTP for asynchronous RCAs is illustrated in Figure 8a,b, corresponding to R0H and R1H, respectively. Normalization is achieved by dividing the actual PCTP of each RCA by the highest PCTP corresponding to an RCA for R0H and R1H separately.

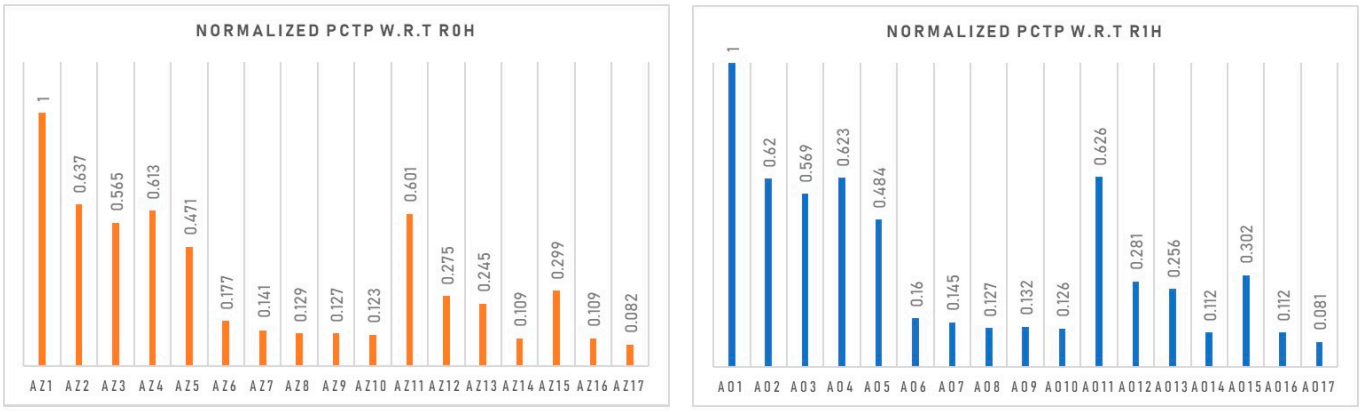

Figure 8. Normalized values of power-cycle time product (PCTP) corresponding to 32-bit asynchronous RCAs comprising different OFAs and TFAs with respect to (**a**) R0H and (**b**) R1H.

In the case of R0H, AZ1 exhibits the highest PCTP, while in the case of R1H, AO1 does. In Figure 8a,b, a normalized PCTP value of 1 indicates a suboptimal design concerning energy. Hence, a small value of normalized PCTP is preferable, as it is representative of an energy-efficient design. Given this, from an energy perspective, AZ17 and AO17, featuring the proposed TFA, are preferable for R0H and R1H, respectively. Compared to AZ14 (representing the best among RCAs comprising OFAs) and AZ16 (representing the best among RCAs comprising TFAs), both of which have the same normalized PCTP (after rounding), AZ17 reports a 24.8% reduction in PCTP. Likewise, compared to AO14 (representing the best among RCAs comprising OFAs) and AO16 (representing the best among RCAs comprising TFAs), both of which have the same normalized PCTP (after

rounding), AO17 reports a 27.7% reduction in PCTP. Between AZ17 and AO17, the latter reports a 4% reduction in PCTP. Hence, from the combined viewpoint of cycle time, power dissipation, and PCTP, it is inferred that R1H is slightly preferable to R0H.

6. Conclusions

This paper introduced a novel monotonic asynchronous TFA that can be replicated and linked together to form an RCA. While RCAs are regarded as slow adders in synchronous design, in the realm of IOM asynchronous design, they stand out due to their minimal reverse latency, unmatched by other asynchronous adders. Traditionally, RCAs are built by cascading OFAs. In this work, we considered TFAs for constructing RCAs. Utilizing our proposed monotonic TFA, we constructed an RCA to evaluate its performance against RCAs realized using various asynchronous OFAs, as well as RCAs constructed using existing asynchronous TFAs. We implemented the adders using a 28-nm CMOS technology, focusing on a 32-bit addition as an example. From the design metrics estimated, the following conclusions were drawn: (i) In comparison to an RCA utilizing a state-of-the-art monotonic OFA, the RCA involving our proposed TFA achieved a 26% reduction in cycle time and a 24.8% reduction in PCTP for R0H, and a 28.5% reduction in cycle time and a 27.7% reduction in PCTP for R1H; (ii) when compared to an RCA incorporating an early output QDI TFA, the RCA employing our proposed TFA attained a 22.3% reduction in cycle time and a 24.8% reduction in PCTP for R0H, and a 25.4% reduction in cycle time and a 27.7% reduction in PCTP for R1H.

Author Contributions: Conceptualization, P.B.; methodology, P.B.; validation, P.B.; formal analysis, P.B.; investigation, P.B. and D.L.M.; resources, D.L.M.; data curation, P.B.; writing—original draft preparation, P.B.; writing—review and editing, P.B.; visualization, P.B.; supervision, D.L.M.; project administration, P.B. and D.L.M.; funding acquisition, D.L.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the Singapore Ministry of Education (MOE), Academic Research Fund under grant number Tier-1 RG127/22.

Data Availability Statement: All data are within the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Zhang, H.; Putic, M.; Lach, J. Low power GPGPU computation with imprecise hardware. In Proceedings of the 51st Annual Design Automation Conference, San Francisco, CA, USA, 1–5 June 2014.
2. Wanhammar, L. *DSP Integrated Circuits*; Academic Press: Cambridge, MA, USA, 1999.
3. Chen, D.C.; Guerra, L.M.; Ng, E.H.; Potkonjak, M.; Schultz, D.P.; Rabaey, J.M. An integrated system for rapid prototyping of high performance algorithm specific data paths. In Proceedings of the International Conference on Application Specific Array Processors, Berkeley, CA, USA, 4–7 August 1992.
4. Hennessy, J.L.; Patterson, D.A. *Computer Architecture: A Quantitative Approach*; Morgan Kaufmann Publishers: Burlington, MA, USA, 1990.
5. Garside, J.D. A CMOS VLSI implementation of an asynchronous ALU. In Proceedings of the IFIP Working Conference on Asynchronous Design Methodologies, Manchester, UK, 31 March–2 April 1993.
6. Shams, A.M.; Darwish, T.K.; Bayoumi, M.A. Performance analysis of low power 1-bit CMOS full adder cells. *IEEE Trans. VLSI Syst.* **2002**, *10*, 20–29. [CrossRef] [PubMed]
7. Martin, A.J. Asynchronous datapaths and the design of an asynchronous adder. *Form. Methods Syst. Des.* **1992**, *1*, 117–137. [CrossRef]
8. Nowick, S.M.; Singh, M. Asynchronous design—Part 1: Overview and recent advances. *IEEE Des. Test* **2015**, *32*, 5–18. [CrossRef]
9. Martin, A.J.; Nystrom, M. Asynchronous techniques for system-on-chip design. *Proc. IEEE* **2006**, *94*, 1089–1120. [CrossRef]
10. Martin, A.J. The limitation to delay-insensitivity in asynchronous circuits. In *Beauty Is Our Business*; Texts and Monographs in Computer Science; Feijen, W.H.J., van Gasteren, A.J.M., Gries, D., Misra, J., Eds.; Springer: New York, NY, USA, 1990; pp. 302–311.

11. Martin, A.J.; Prakash, P. Asynchronous nano-electronics: Preliminary investigation. In Proceedings of the 14th IEEE International Symposium on Asynchronous Circuits and Systems, Newcastle upon Tyne, UK, 7–11 April 2008.
12. Seitz, C.L. System Timing. In *Introduction to VLSI Systems*; Mead, C., Conway, L., Eds.; Addison-Wesley: Reading, MA, USA, 1980; pp. 218–262.
13. Brej, C. Early-output Logic and Anti-Tokens. Ph.D. Thesis, The University of Manchester, Manchester, UK, September 2005.
14. Stevens, K.S.; Ginosar, R.; Rotem, S. Relative timing. *IEEE Trans. VLSI Syst.* **2003**, *11*, 129–140. [CrossRef]
15. Varshavsky, V.I. Aperiodic Circuits. In *Self-Timed Control of Concurrent Processes: The Design of Aperiodic Logical Circuits in Computers and Discrete Systems*; Varshavsky, V.I., Ed.; Translated from the Russian by Yakovlev, A.V.; Kluwer Academic Publishers: New York, NY, USA, 1990; pp. 77–85.
16. Cortadella, J.; Kondratyev, A.; Lavagno, L.; Sotiriou, C. Coping with the variability of combinational logic delays. In Proceedings of the IEEE International Conference on Computer Design, San Jose, CA, USA, 11–13 October 2004.
17. Jeong, C.; Nowick, S.M. Optimal technology mapping and cell merger for asynchronous threshold networks. In Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems, Grenoble, France, 13–15 March 2006.
18. Toms, W.B.; Edwards, D.A. Indicating combinational logic decomposition. *IET Comput. Digit. Tech.* **2011**, *5*, 331–341. [CrossRef]
19. Jeong, C.; Nowick, S.M. Optimization of robust asynchronous circuits by local input completeness relaxation. In Proceedings of the Asia and South Pacific Design Automation Conference, Yokohama, Japan, 23–26 January 2007.
20. Muller, D.E.; Bartky, S. A theory of asynchronous circuits. In Proceedings of the International Symposium on the Theory of Switching (Part I), Cambridge, MA, USA, 2–5 April 1957.
21. Beerel, P.A.; Ozdag, R.O.; Ferretti, M. *A Designer's Guide to Asynchronous VLSI*; Cambridge University Press: Cambridge, UK, 2010.
22. Shams, M.; Ebergen, J.C.; Elmasry, M.I. A comparison of CMOS implementations of an asynchronous circuits primitive: The C-element. In Proceedings of the 1996 International Symposium on Low Power Electronics and Design, Monterey, CA, USA, 12–14 August 1996.
23. Heck, L.S.; Moreira, M.T.; Calazans, N.L.V. Hardening C-elements against metastability. In Proceedings of the 24th IEEE International Conference on Electronics, Circuits and Systems, Batumi, Georgia, 5–8 December 2017.
24. Verhoeff, T. Delay-insensitive codes—An overview. *Distrib. Comput.* **1988**, *3*, 1–8. [CrossRef]
25. Sparsø, J.; Furber, S.B. *Principles of Asynchronous Circuit Design: A Systems Perspective*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2001; pp. 9–28.
26. Bose, B. On unordered codes. *IEEE Trans. Comput.* **1991**, *40*, 125–131. [CrossRef]
27. Moreira, M.T.; Guazzelli, R.A.; Calazans, N.L.V. Return-to-one protocol for reducing static power in C-elements of QDI circuits employing m-of-n codes. In Proceedings of the 25th Symposium on Integrated Circuits and Systems Design, Brasilia, Brazil, 30 August–2 September 2012.
28. Sasao, T. *Switching Theory for Logic Synthesis*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1999.
29. Singh, N.P. A Design Methodology for Self-Timed Systems. Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, February 1981.
30. Toms, W.B. Synthesis of Quasi-Delay-Insensitive Datapath Circuits. Ph.D. Thesis, The University of Manchester, Manchester, UK, February 2006.
31. Sparsø, J.; Staunstrup, J. Delay-insensitive multi-ring structures. *Integration* **1993**, *15*, 313–340. [CrossRef]
32. Folco, B.; Bregier, V.; Fesquet, L.; Renaudin, M. Technology mapping for area optimized quasi delay insensitive circuits. In Proceedings of the IFIP 13th International Conference on Very Large Scale Integration, Perth, Australia, 17–19 October 2005.
33. Balasubramanian, P.; Edwards, D.A. A delay efficient robust self-timed full adder. In Proceedings of the IEEE 3rd International Design and Test Workshop, Monastir, Tunisia, 20–22 December 2008.
34. Balasubramanian, P. A latency optimized biased implementation style weak-indication self-timed full adder. *Facta Univ. Ser. Electron. Energetics* **2015**, *28*, 657–671. [CrossRef]
35. Huemer, F.; Steininger, A. Sorting network based full adders for QDI circuits. In Proceedings of the Austrochip Workshop on Microelectronics, Vienna, Austria, 7 October 2020.
36. Balasubramanian, P. A robust asynchronous early-output full adder. *WSEAS Trans. Circuits Syst.* **2011**, *10*, 221–230.
37. Balasubramanian, P.; Yamashita, S. Area/latency optimized early-output asynchronous full adders and relative-timed ripple carry adders. *SpringerPlus* **2016**, *5*, 440. [CrossRef] [PubMed]
38. Balasubramanian, P.; Maskell, D.L. A monotonic early output asynchronous full adder. *Technologies* **2023**, *11*, 126. [CrossRef]
39. Balasubramanian, P.; Edwards, D.A. Dual-sum single-carry self-timed adder designs. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI, Tampa, FL, USA, 13–15 May 2009.
40. Balasubramanian, P.; Prasad, K. Asynchronous early-output dual-bit full adders based on homogeneous and heterogeneous delay-insensitive data encoding. *WSEAS Trans. Circuits Syst.* **2017**, *16*, 64–73.
41. Moreira, M.T.; Guazzelli, R.A.; Calazans, N.L.V. Return-to-One DIMS logic on 4-phase m-of-n asynchronous circuits. In Proceedings of the 19th IEEE International Conference on Electronics, Circuits and Systems, Seville, Spain, 9–12 December 2012.

42. Synopsys SAED_EDK32/28_CORE Databook, Revision 1.0.0. January 2012. Available online: https://www.synopsys.com/community/university-program/teaching-resources.html (accessed on 7 September 2023).
43. Rabaey, J.M.; Chandrakasan, A.; Nikolic, B. *Digital Integrated Circuits: A Design Perspective*, 2nd ed.; Pearson Education: London, UK, 2003.