FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 38, Nº 2, June 2025, pp. 239 - 261
https://doi.org/10.2298/FUEE2502239B

Original scientific paper

ASYNCHRONOUS MONOTONIC MULTIPLEXER

Padmanabhan Balasubramanian[1], Nikos E. Mastorakis[2]

[1]College of Computing and Data Science, Nanyang Technological University, Singapore
[2]Technical University of Sofia, English Language Faculty of Engineering (ELFE), Department of Industrial Engineering, Sofia, Bulgaria

ORCID iDs: Padmanabhan Balasubramanian https://orcid.org/0000-0001-9412-4773
Nikos E. Mastorakis N/A

Abstract. *Among input-output (IO) mode asynchronous circuits, indicating circuits are more popular and robust. However, they may not be efficient in design metrics such as cycle time, area, and power dissipation. In contrast, monotonic circuits, which are also IO mode asynchronous circuits but less explored, can potentially optimize the design metrics better than indicating circuits. Recent studies have demonstrated that monotonic circuits outperform indicating circuits in arithmetic operations like addition and multiplication. While monotonic circuits may be labeled theoretically less robust than indicating circuits, their operation is similar in practice. This article presents a novel, compact monotonic IO mode asynchronous multiplexer. The multiplexer is significant in digital circuits as it has applications across various domains including communication systems, digital signal processing, memory addressing, etc. We considered dual-rail encoding for the multiplexer and employed four-phase handshaking. Two four-phase handshaking schemes are available namely, return-to-zero (RtZ) handshaking and return-to-one (RtO) handshaking, and we considered both for this work. Compared to an optimized early output quasi-delay-insensitive multiplexer, which is derived by modifying a strong-indication multiplexer and represents the best among the existing designs, the proposed monotonic multiplexer achieves a 67% (69%) reduction in latency, an 84% (84%) reduction in area, and a 66% (67%) reduction in power for RtZ (RtO) handshaking. Since the multiplexer is a small component, its effectiveness should be evaluated by integrating it into a circuit setup. In this context, we used existing multiplexers and the proposed multiplexer to realize IO mode asynchronous 32-bit carry select adders (CSLAs) while keeping the compute element, namely the full adder consistent. We estimated the design metrics of CSLAs incorporating different multiplexers, implemented using a 28-nm bulk CMOS process technology. The CSLA utilizing the proposed monotonic multiplexer achieved a 43% (44%) reduction in cycle time and a*

Received September 17, 2024; revised November 16, 2024 and January 07, 2025; accepted January 14, 2025
Corresponding author: Padmanabhan Balasubramanian
College of Computing and Data Science, Nanyang Technological University, Singapore
E-mail: balasubramanian@ntu.edu.sg

28% (28%) reduction in area compared to the CSLA utilizing an early output quasi-delay-insensitive multiplexer for RtZ (RtO) handshaking with no power penalty.

Key words: *asynchronous circuits, digital logic design, low power, high-speed, CMOS*

1. INTRODUCTION

The multiplexer is a basic and important component that has practical applications across various domains in electronics, telecommunications, and digital systems. In communication systems, multiplexers combine multiple data streams into a single stream for transmission over a shared medium like a cable or a wireless channel. This is known as multiplexing, and it allows for more efficient use of a communication channel. In analog-to-digital conversion, where analog signals should be converted into digital format for processing by digital systems, multiplexers are used to select one of many analog input signals for conversion by an analog-to-digital converter. In digital signal processing, multiplexers are used to select different data streams or inputs for processing by digital signal processing algorithms or hardware. In memory systems, multiplexers are used for memory address decoding where they are used to select an appropriate memory location based on the address provided by a central processing unit. In control systems, multiplexers are used to select different control signals or inputs to control various processes or devices. Multiplexers are also used in testing and measurement equipment to select different input signals for measurement by a single instrument. In digital video broadcasting systems, multiplexers are used to combine multiple digital video streams, audio streams, and other data into one transport stream for transmission. In telecommunication networks, multiplexers play a crucial role in combining multiple voice or data channels into higher-order transmission links, optimizing bandwidth usage. In sensor networks, multiplexers can be used to aggregate data from multiple sensors before transmitting the data over to a communication network. In industrial automation systems, multiplexers select different sensors, actuators, or control signals in manufacturing processes or automated systems.

This paper introduces a multiplexer that falls within the monotonic category of input-output (IO) mode asynchronous circuits. These circuits typically encode data using delay-insensitive codes and follow a four-phase handshake protocol for data exchange. In contrast to synchronous circuits that depend on a clock signal, IO mode asynchronous circuits function based on events, which enhances their robustness. The event-driven approach makes IO mode asynchronous circuits more resilient to variations in process, voltage, and temperature, thus improving their adaptability [1,2]. Furthermore, IO mode asynchronous circuits are modular [3,4], self-checking [5], less affected by electromagnetic interference than synchronous designs [6], and naturally resistant to side-channel attacks [7], making them particularly suitable for security-sensitive applications [8].

Asynchronous circuits used in IO systems are commonly classified into quasi-delay-insensitive (QDI) and non-QDI categories. QDI circuits rely on a concept called isochronic forks [9], which refers to electrical nodes with multiple outgoing wires, functioning under the premise that signals on these wires transition at the same time. This assumption generally applies to both microelectronics and nanoelectronics [10]. QDI circuits ensure that outputs are generated only after all inputs have been received and processed, and internal computations are completed. While this feature improves the reliability of QDI circuits, it also results in greater implementation costs, as these circuits tend to require more area, have higher latency and cycle times, and dissipate more power than non-QDI circuits.

QDI circuits are categorized into several types: strong indication [11], weak indication [11], and early output QDI (EOQDI) [12]. In strong indication circuits, outputs are not generated until all primary inputs have been received and processed. Weak indication circuits, on the other hand, allow some outputs to be produced once certain inputs have been processed, though the final output will appear only after all inputs have been processed. EOQDI circuits are designed to generate outputs as soon as some inputs are processed, especially when a spacer is introduced. In EOQDI circuits, isochronic forks assumed for the primary inputs ensure that any delayed inputs are duly acknowledged.

Non-QDI IO mode asynchronous circuits consist of relative-timed circuits [13] and monotonic circuits [14,15]. Relative-timed circuits rely on certain timing constraints to properly sequence input signals for output generation. Monotonic circuits, however, maintain consistent signal transitions throughout the circuit. In a monotonically rising circuit, a rising transition (e.g., binary 0 to 1) at the primary inputs triggers corresponding rising transitions at both intermediate and primary outputs. Conversely, in a monotonically falling circuit, a falling transition (e.g., binary 1 to 0) at the inputs results in falling transitions at the outputs. A circuit may exhibit monotonic behavior with rising, falling, both, or neither transition. While synchronous circuits typically do not show monotonicity, IO mode asynchronous circuits are generally monotonic. In this discussion, "monotonic circuits" specifically refer to IO mode asynchronous circuits that support both rising and falling behaviors. Although monotonic circuits often produce early outputs, they are not deemed QDI because they do not require the completion of all internal computations before generating outputs. Non-QDI circuits, which impose fewer constraints than QDI circuits, provide greater flexibility and contribute to reduced circuit complexity, thereby achieving more efficient designs. Recent studies have shown that monotonic circuits outperform QDI circuits in computer arithmetic tasks such as addition [16] and multiplication [17].

The organization of this article is as follows: Section 2 describes the foundational concepts of IO mode asynchronous circuit design. Section 3 reviews traditional indicating multiplexers along with an EOQDI multiplexer. Section 4 presents the design of the proposed monotonic multiplexer. Section 5 compares the design metrics of various multiplexers and 32-bit asynchronous carry select adders implemented using these multiplexers. Finally, Section 6 concludes this article with a summary.

2. Fundamentals of IO mode Asynchronous Circuit Design

Fig. 1a presents a block diagram of an IO mode asynchronous circuit pipeline [1]. Each stage in this pipeline consists of an asynchronous circuit placed between sets of input and output registers. The input registers may function as output registers for the previous stage, while the output registers can serve as input registers for the following stage. The input registers supply inputs to the asynchronous circuit for processing. A completion detector on the input side indicates the reception of all inputs, and on the output side, it indicates the completion of all outputs. Figs. 1b and 1c show examples of completion detectors for Return-to-Zero (RtZ) and Return-to-One (RtO) handshaking schemes respectively, which will be discussed further in this section. The completion detector associated with the output registers sends an output acknowledgment signal (Ack_O), which, after a Boolean inversion, becomes the input acknowledgment signal (Ack_I). Ack_I allows the input registers to supply new inputs to the asynchronous

circuit. In an IO mode asynchronous circuit, “handshaking” refers to the communication protocol followed between input and output registers.

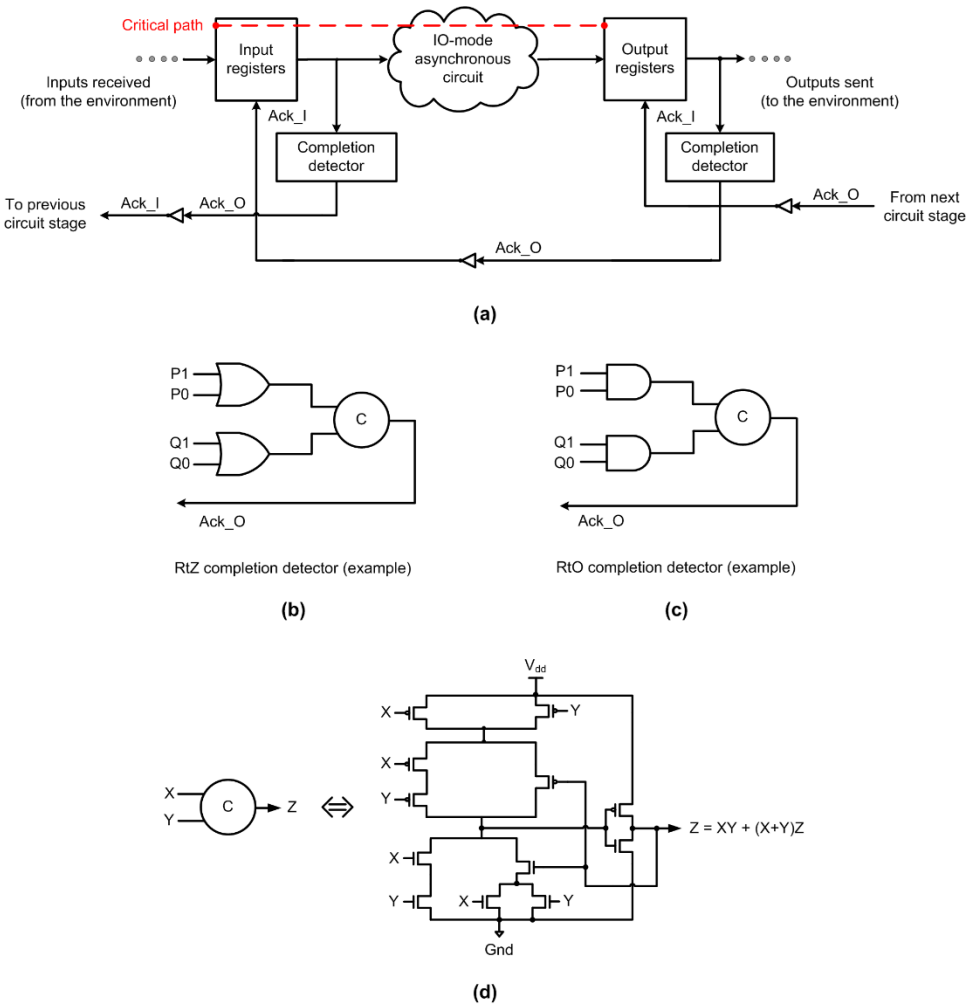


Fig. 1 (a) Block diagram of a single pipeline stage in an IO mode asynchronous circuit. Sample completion detector design for (b) RtZ handshaking and (c) RtO handshaking. (d) Logic symbol and static CMOS transistor-level design for a 2-input C-element, created by adding feedback to an AO222 complex gate

In an IO mode asynchronous circuit, the Muller C-element [18] functions as a register. When all inputs to the C-element are either 0 or 1, the output will match this value, producing a 0 or 1 accordingly; if the inputs are mixed, the output retains its current state. Fig. 1d shows inputs X and Y feeding into the C-element, with Z as the output. This figure provides the logic symbol, transistor-level structure, and output equation for the C-element. A 2-input C-element can be realized at the transistor level by

adding feedback to a static CMOS AO222 complex gate [19]. Multiple custom designs for the C-element have been developed and evaluated in [20,21]; however, this study employs a semi-custom design based on the approach followed in [19]. In the input registers of Fig. 1a, each C-element connects one input to Ack_I and the other to an encoded input signal rail.

In IO mode asynchronous circuits, delay-insensitive codes [22], such as dual-rail (also known as two-rail, double-rail, or 1-of-2 code), are frequently used for encoding inputs and outputs. We will now explain how dual-rail encoding is applied to inputs and outputs within the context of RtZ and RtO handshaking, followed by an overview of each handshaking scheme. For dual-rail encoding under the RtZ handshaking scheme [1], an input signal, denoted by I, is represented with two wires say, I1 and I0. When I = 1, the encoding is set to I1 = 1 and I0 = 0; when I = 0, it is encoded as I1 = 0 and I0 = 1. These configurations represent 'data' according to the RtZ handshake scheme. The combination I1 = I0 = 0 acts as the 'zero spacer' that separates successive data in RtZ handshaking. The encoding I1 = I0 = 1 is considered invalid in RtZ handshaking, as the encoding is designed to remain unordered [23]. For RtO handshaking [24], the dual-rail encoding uses two wires say, I1 and I0, to represent the input signal I. Here, when I = 1, the encoding is I1 = 0 and I0 = 1; when I = 0, the encoding is I1 = 1 and I0 = 0. These two configurations indicate 'data' according to the RtO handshake scheme. The combination I1 = I0 = 1 serves as the 'one spacer' that separates successive data in RtO handshaking. The combination I1 = I0 = 0 is invalid for RtO handshaking, as unordered encoding is required [23].

Figs. 1b and 1c depict examples of dual-rail encoded inputs, denoted as (P1, P0) and (Q1, Q0). In Fig. 1b, a completion detector for RtZ handshaking is shown, consisting of OR gates at the initial logic level. Each two-input OR gate combines the two rails of a particular encoded input. The outputs from these OR gates are then routed to a C-element, or a chain of C-elements, to produce the output acknowledgment signal (Ack_O). In contrast, Fig. 1c illustrates a completion detector for RtO handshaking, which utilizes AND gates at the first logic level. Each two-input AND gate combines the two rails of the encoded inputs, and the resulting outputs are subsequently directed to a C-element or a series of C-elements to generate Ack_O.

We shall now describe RtZ and RtO handshaking protocols. In RtZ handshaking, the process begins with the first phase, where Ack_I is set to 1 and Ack_O is at 0. This signals the input registers to send data to the asynchronous circuit. During this phase, one of the two rails of each encoded input is set to 1, indicating that the data is ready for processing. In the second phase, the output registers receive the processed data from the asynchronous circuit, and the completion detector sets Ack_O to 1. In the third phase, the input registers wait for Ack_I to return to 0 before sending the spacer to the asynchronous circuit for processing. In the final phase, the output registers receive the spacer, and the completion detector sets Ack_O back to 0. This concludes one data transaction and signals that the asynchronous circuit is ready for the next data transaction when Ack_I is set to 1 again. As a result, in RtZ handshaking, the input sequence follows the pattern of 'data–spacer–data–spacer', and so on.

In RtO handshaking, the process starts with the first phase, where Ack_I is set to 1 and Ack_O is at 0. This signals the input registers to send the spacer to the asynchronous circuit for processing. During this phase, all rails of the encoded inputs are set to 1, indicating that the spacer is ready to be processed by the asynchronous circuit. In the second phase, the asynchronous circuit generates the spacer, which is then received by

the output registers. The completion detector for the output registers sets Ack_O to 1. In the third phase, the input registers wait for Ack_I to return to 0 before sending the data to the asynchronous circuit, with one of the two rails of each encoded input set to 0. In the fourth phase, the asynchronous circuit processes the data and produces the output, which is then received by the output registers. The completion detector sets Ack_O back to 0. This marks the end of a data transaction and indicates that the asynchronous circuit is ready for the next data transaction when Ack_I is set to 1 again. Thus, in RtO handshaking, the input sequence follows the pattern of 'spacer–data–spacer–data' and so forth.

To maintain delay insensitivity in IO mode asynchronous circuits, a spacer is placed between successive input data. In a monotonic circuit, this spacer ensures that data and spacer do not conflict, thus preserving external delay insensitivity throughout the handshaking process. In the IO mode asynchronous pipeline depicted in Fig. 1a, the key timing metric is the 'cycle time', which represents the total duration required to complete a single data transaction. The maximum time taken to process data is known as forward latency, while the maximum time to process the spacer is referred to as reverse latency. Forward and reverse latency may vary based on the circuit's design and underlying logic. The overall cycle time of an IO mode asynchronous circuit is the sum of the forward and reverse latencies. The critical path that determines the latency of the circuit, which includes the input register bank and the asynchronous circuit, is shown by the red dashed line in Fig. 1a.

3. Conventional Asynchronous Multiplexer Designs

Existing IO mode multiplexer designs follow the strong indication, as the multiplexer features a single primary output that is dual-rail encoded. Hence, multiplexers cannot be designed as weak indication circuits since weak indication circuits require a minimum of two encoded primary outputs – one generated after processing a subset of the primary inputs, and the other produced after processing the remaining or all the inputs. This section will discuss strong indication and EOQDI multiplexers.

3.1. DIMS multiplexer

The delay-insensitive minterm synthesis, also called DIMS [25], requires listing a function's distinct product terms that include all the support variables. Let us consider (A1, A0) and (B1, B0) as the dual-rail encoded inputs, (S1, S0) as the dual-rail encoded select signal, and (M1, M0) as the dual-rail encoded output of a 2-to-1 multiplexer. In this case, the output expressions for the multiplexer are given by (1) and (2), which conform to RtZ handshaking. Equations (1) and (2) imply that when the select signal is binary 0 (i.e., S1 = 0, S0 = 1), input A is selected and forwarded as the output. Conversely, input B is selected and forwarded as the output when the select signal is binary 1 (i.e., S1 = 1, S0 = 0).

$$M1 = A0B1S1 + A1B0S0 + A1B1S0 + A1B1S1 \quad (1)$$

$$M0 = A0B0S0 + A0B0S1 + A0B1S0 + A1B0S1 \quad (2)$$

Equations (1) and (2) are represented as sums of disjoint products, where each product term is independent or orthogonal to the others [26]. In the case of RtZ handshaking, only one product term in (1) or (2) evaluates to 1 when data is input. Conversely, for RtO handshaking, only one product term evaluates to 0 under the same conditions. These

conditions, in which only a single product term evaluates to either 1 or 0 for RtZ or RtO handshaking, are known as the monotonic cover constraint [1]. From the perspective of physical implementation, this constraint guarantees that one signal path is activated from the primary input to the primary output, thereby avoiding unnecessary transitions on the intermediate gate outputs within the circuit. The monotonic cover constraint is commonly incorporated in IO mode asynchronous circuits.

Fig. 2 shows the logic implementation of a strongly indicating 2-to-1 multiplexer using the DIMS method, which utilizes C-elements and OR gates and is designed for RtZ handshaking. If the OR gates in this design were replaced with AND gates, the resulting logic would be suitable for RtO handshaking. In general, by replacing all gates (except the C-elements) in an IO mode asynchronous circuit with their Boolean duals, a circuit can be transformed from one that corresponds to RtZ handshaking to one that corresponds to RtO handshaking and vice versa. This principle has been proved through logical induction in [27].

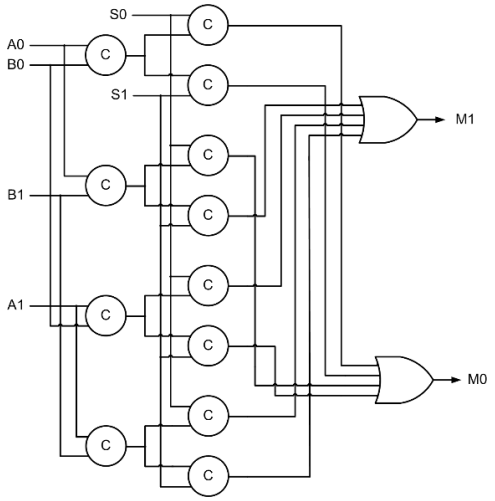


Fig. 2 Logic realization of a 2-to-1 multiplexer based on the DIMS method (corresponding to RtZ handshaking)

3.2. Toms' multiplexer

Toms' method for strongly indicating logic synthesis of combinational functions (post-encoding) [28,29] employs standard multi-level logic synthesis techniques [30]. These techniques involve extracting one or more product terms (a sum of multiple products) by solving the rectangle covering problem, which is efficiently tackled using a sparse-matrix approach developed by Rudell [31]. Once the product terms are identified, they are substituted as intermediate variables into the original logic expressions. The extraction phase involves finding and creating common sub-functions and variables, while the substitution phase inserts an intermediate function X into a larger function Y, where Y is expressed in terms of its original inputs and X. These processes are similar to Boolean division and multiplication, respectively.

Fig. 3 displays the logical implementation of a strongly indicating Toms 2-to-1 multiplexer for RtZ handshaking, which is composed of C-elements and OR gates. To obtain the corresponding multiplexer circuit for RtO handshaking, the OR gates should be replaced with AND gates.

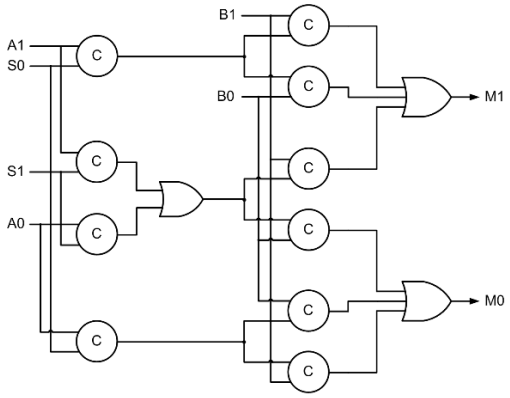


Fig. 3 2-to-1 multiplexer based on Toms' method (corresponding to RtZ handshaking)

3.3. Early output QDI multiplexer

Equations (1) and (2) given for the DIMS multiplexer can be simplified to remove logical redundancy. In (1), the sum of products A1B1S1+A1B1S0 simplifies to A1B1, and similarly, in (2), A0B0S1+A0B0S0 simplifies to A0B0. As a result, the strong indication multiplexer based on the DIMS method can be modified to create an EOQDI multiplexer that synthesizes (3) and (4). Fig. 4, derived based on (3) and (4), represents the circuit for RtZ handshaking. To convert this to an equivalent circuit for RtO handshaking, the OR gates in Fig. 4 should be replaced with AND gates.

$$M1 = A0B1S1 + A1B0S0 + A1B1 \quad (3)$$

$$M0 = A0B1S0 + A1B0S1 + A0B0 \quad (4)$$

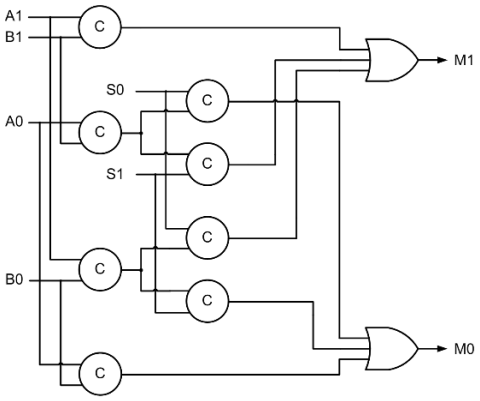


Fig. 4 Early output QDI multiplexer (corresponding to RtZ handshaking)

3.4. SIDCO and SIDCAO multiplexers

In [32], two strongly indicating multiplexer designs were proposed: SIDCO, which uses C-elements and OR gates, and SIDCAO, which incorporates C-elements, AND gates, and OR gates. These designs are illustrated in Figs. 5a and 5b, both of which are intended for RtZ handshaking. To adapt them for RtO handshaking, the OR gates in Figs. 5a and 5b should be swapped with AND gates and the AND gates should be replaced by OR gates. The SIDCO and SIDCAO multiplexers implement (5) and (6), given below.

$$M1 = A1S0 + B1S1 \tag{5}$$

$$M0 = A0S0 + B0S1 \tag{6}$$

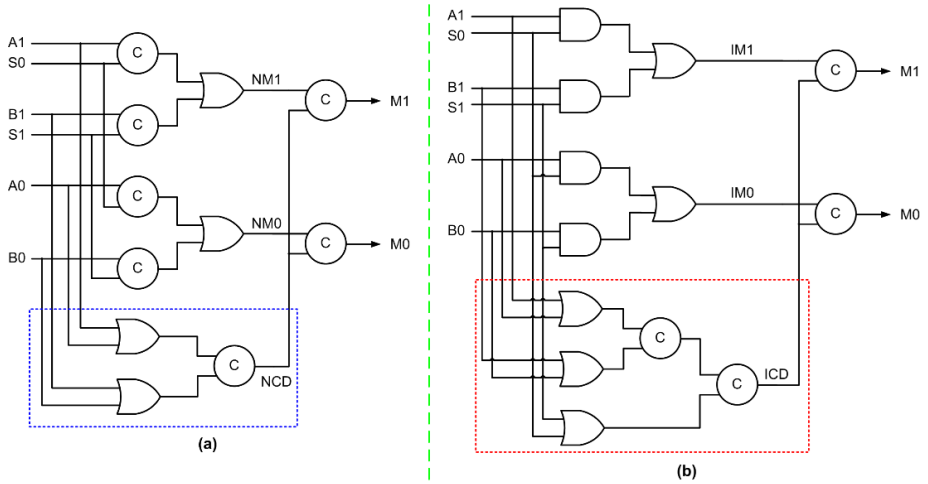


Fig. 5 (a) SIDCO multiplexer and (b) SIDCAO multiplexer, corresponding to RtZ handshaking.

In Fig. 5a, the intermediate output (NM1, NM0) is functionally the same as the primary output (M1, M0) of the multiplexer. However, to ensure a strong indication, an internal completion detector is used to verify the arrival of the multiplexer inputs (A1, A0) and (B1, B0). This internal completion detector, highlighted within the blue dotted box in Fig. 5a, generates an output labeled NCD. NCD is synchronized separately with NM1 and NM0 to yield the final primary output (M1, M0).

In Fig. 5b, the C-elements that generate the products A1S0, B1S1, A0S0, and B0S1 in Fig. 5a are replaced with AND gates. To ensure a strong indication, an internal completion detector is introduced to verify the arrival of both the multiplexer inputs and the select signal. This completion detector, highlighted within the red dotted box in Fig. 5b, produces an output labeled ICD. The intermediate output (IM1, IM0) behaves identically to the primary output (M1, M0). However, IM1 and IM0 are synchronized individually with ICD to produce the final primary output (M1, M0).

4. Proposed Asynchronous Monotonic Multiplexer

The multiplexers discussed till now are either strongly indicating or EOQDI. In contrast, the multiplexer introduced here is a monotonic circuit. The circuit diagrams of the proposed multiplexer are presented in Figs. 6a and 6b for RtZ and RtO handshaking.

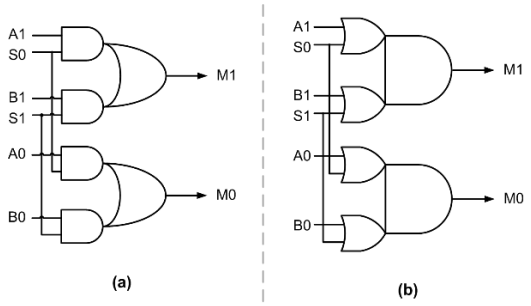


Fig. 6 Proposed multiplexer corresponding to handshake schemes: (a) RtZ and (b) RtO

The proposed multiplexer, designed for RtZ handshaking, uses two AO22 complex gates, which implement (5) and (6). The dual of the AO22 complex gate is the OA22 complex gate, so for RtO handshaking, two OA22 gates are used. In comparison to Figs. 2 through 5, Fig. 6 shows that the proposed multiplexer does not incorporate any C-elements, unlike the other designs. Additionally, this proposed design represents an optimized gate-level realization of a 2-to-1 multiplexer in IO mode asynchronous circuit type, requiring only ten transistors for a static CMOS implementation. As mentioned in Section 1, monotonic circuits are more flexible than QDI circuits and only need to ensure the monotonicity of signal transitions for data and spacer.

We shall explain the monotonic and early output operation of the proposed multiplexer based on RtZ and RtO handshake schemes by considering example scenarios. Typically, for RtZ handshaking, when data is applied, the signal transitions will rise monotonically (from 0 to 1) through the primary inputs and any intermediate outputs to the primary outputs. For the spacer, the signal transitions will fall monotonically (from 1 to 0) from the primary inputs through the intermediate outputs to the primary outputs. In Fig. 6a, when data is input, if A1 = S0 = 1 or B1 = S1 = 1, the output M1 could assume 1 early without waiting for B1/B0 to assume 1 in the former case and A1/A0 to assume 1 in the latter case. Given this, any late transition of B1/B0 to 1 (in the former case) and A1/A0 to 1 (in the latter case) will be acknowledged by the completion detector. Since the isochronic fork assumption is applied to the primary inputs, the acknowledgment provided by the completion detector applies to the multiplexer as well. When the spacer is applied, M1 can transition to 0 early, even if A1 or S0 (if they were 1 previously), or B1 or S1 (if they were 1 previously), change to 0 without waiting for all inputs to reach 0. However, the next data will only be supplied to the monotonic multiplexer after all primary inputs have been reset to 0, after indication by the completion detector. Since the isochronic fork assumption applies to all primary inputs, the RtZ condition of all inputs, as confirmed by the completion detector, is also applicable to the multiplexer. These example scenarios considered show that both monotonic and early output operation manifests in the proposed multiplexer for the application of data and spacer with respect to RtZ handshaking.

Typically, for RtO handshaking, when the spacer is applied, the signal transitions rise monotonically (from 0 to 1) through the primary inputs and any intermediate outputs to the primary outputs. In contrast, when data is applied, the signal transitions fall monotonically (from 1 to 0) from the primary inputs through any intermediate outputs to the primary outputs. Referring to Fig. 6b, when the spacer is provided, all inputs (A1, A0, B1, B0, S1, and S0) will be set to 1, which causes M1 and M0 to assume 1. Due to the properties of OR logic, if A1, B1, A0, and B0 are all 1, M1 and M0 can be set to 1 early, without waiting for S1 and S0 to change to 1. However, if S1 and S0 take longer to reach 1, this will be confirmed by the completion detector before the data is supplied to the multiplexer. As the isochronic fork assumption holds for the primary inputs, the multiplexer is said to acknowledge that S1 and S0 have reached 1 once the completion detector confirms this. Now, let us consider an example scenario when data is supplied after the application of the spacer. If A1 and S0 assume 0 or if B1 and S1 assume 0, M1 could switch to 0 early, without waiting for B1/B0 to assume 0 in the former case and A1/A0 to assume 0 in the latter case. Given this, any late transition of B1/B0 to 0 (in the former case) or A1/A0 to 0 (in the latter case) will be acknowledged by the completion detector. Since the isochronic fork assumption applies to all primary inputs, the acknowledgment provided by the completion detector is said to apply to the multiplexer as well. Thus, the example scenarios considered imply that both monotonic and early output operation manifests in the proposed multiplexer for the application of both data and spacer with respect to RtO handshaking.

Fig. 7 shows a screenshot of the input-output simulation waveforms of the proposed multiplexer based on RtZ handshaking. Simulations were performed using Synopsys VCS by supplying all distinct inputs at a latency of 1ns. The (zero) spacer was inserted between two data inputs. In Fig. 7, (A1, A0) and (B1, B0) represent the multiplexer inputs, (S1, S0) represents the multiplexer select input, and (Z1, Z0) represents the multiplexer output.

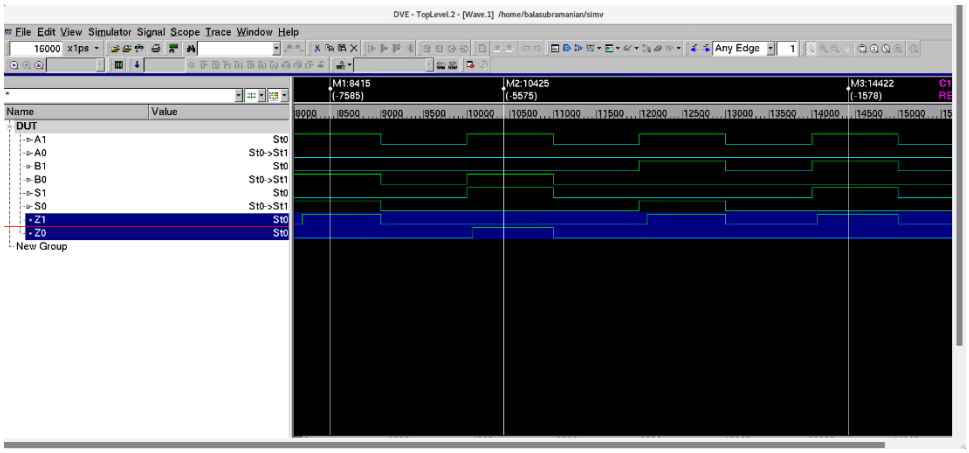


Fig. 7 A screenshot of simulation waveforms of the proposed multiplexer corresponding to RtZ handshaking

Three sample markers, namely M1, M2, and M3 are shown in Fig. 7 which highlight specific instances of input-output and they are mentioned below:

- Marker M1: A1 = B0 = S0 = 1, and Z1 = 1
- Marker M2: A1 = B0 = S1 = 1, and Z0 = 1
- Marker M3: A1 = B1 = S1 = 1, and Z1 = 1

Fig. 8 shows a screenshot of the input-output simulation waveforms corresponding to the proposed multiplexer based on RtO handshaking. Again, simulations were performed using Synopsys VCS by supplying all distinct inputs at a latency of 1ns. The (one) spacer was inserted between two data inputs. The multiplexer input and output naming convention was maintained the same in Fig. 8 as in Fig. 7. Three sample markers viz. M1, M2, and M3 are shown in Fig. 8 which highlights specific instances of input-output and they are mentioned below:

- Marker M1: A1 = B0 = S0 = 0, and Z1 = 0
- Marker M2: A1 = B0 = S1 = 0, and Z0 = 0
- Marker M3: A1 = B1 = S0 = 0, and Z1 = 0

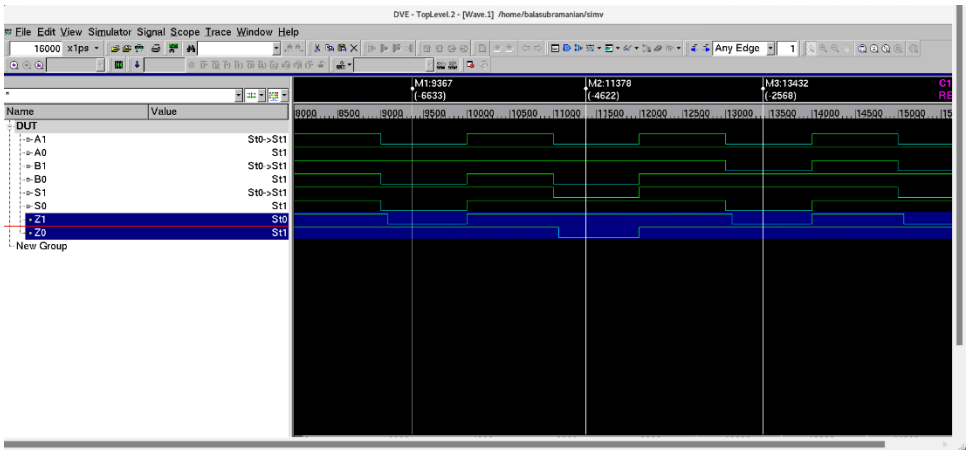


Fig. 8 A screenshot of simulation waveforms of the proposed multiplexer corresponding to RtO handshaking

5. Design Metrics

This section will first present the design metrics of various multiplexers for RtZ and RtO handshaking. After that, the design metrics of 32-bit carry-select adders (CSLAs) implemented with conventional and proposed multiplexers will be presented. The CSLA is used as a case study to illustrate how various multiplexers affect the cycle time, which in turn influences the throughput in an IO mode asynchronous design.

The IO mode asynchronous multiplexers were designed using gates from a 28-nm bulk CMOS standard digital cell library [33], with separate implementations for RtZ and RtO handshaking. The strong indication and EOQDI multiplexers discussed in Section 3 utilize the C-element in their logic design. Further, the completion detectors require C-

elements for their implementation. Since the C-element is not typically available in standard cell libraries, it was manually implemented in static CMOS, as illustrated in Fig. 1d. A low-leakage typical case cell library was used that operates at a supply voltage of 1.05 V and a junction temperature of 25°C. Functional simulations and design metric evaluations, including latency, area, and total (average) power dissipation, were carried out using Synopsys EDA tools viz. VCS, PrimeTime and PrimePower. The default wire load model was applied, and all output ports (i.e., multiplexer outputs) were assigned a fanout-of-4 drive strength. A virtual clock was specified just to constrain the inputs and outputs of the adder, though it was not included in the physical design. Functional simulations for the multiplexers were conducted using separate test benches for RtZ and RtO handshaking, with inputs provided at a latency of 1 ns. Nevertheless, the test benches corresponding to RtZ and RtO handshaking are logically equivalent. The design metrics estimated for the multiplexers are given in Table 1.

Table 1 Design metrics of multiplexers realized using a 28-nm bulk CMOS process.

Multiplexer	Latency (ns)	Area (μm^2)	Power Dissipation (μW)
Corresponding to RtZ handshaking			
DIMS	0.49	46.76	14.37
Toms	0.51	39.65	13.61
EOQDI	0.45	31.01	10.15
SIDCO	0.44	31.26	20.40
SIDCAO	0.57	31.51	24.89
Proposed (monotonic)	0.15	5.08	3.43
Corresponding to RtO handshaking			
DIMS	0.47	44.73	13.15
Toms	0.50	39.65	13.79
EOQDI	0.45	31.01	10.39
SIDCO	0.38	31.26	19.55
SIDCAO	0.57	31.51	24.91
Proposed (monotonic)	0.14	5.08	3.40

In contrast to the proposed multiplexer depicted in Fig. 6, the other multiplexers shown in Figs. 2 to 5 require more gates and additional logic levels. As a result, the conventional multiplexers consume more area, have higher power dissipation, and exhibit greater latency, as evident from Table 1. Among the conventional designs, the EOQDI multiplexer, which has been derived from the DIMS strong indication multiplexer, is more efficient. Its latency is close to the latency of the SIDCO multiplexer while occupying slightly less area and dissipating approximately half the power. The EOQDI multiplexer's lower power dissipation compared to the SIDCO multiplexer can be attributed to the absence of an internal completion detector in the former compared to the latter. Generally, an internal completion detector in an IO mode asynchronous circuit experiences significant switching activity, as all gates within the detector transition during both data and spacer application. When compared to the EOQDI multiplexer, the proposed monotonic multiplexer delivers a 66.7% (68.9%) reduction in latency, occupies 83.6% (83.6%) less area, and dissipates 66.2% (67.3%) less power for RtZ (RtO) handshaking.

To assess the performance of the proposed multiplexer, a 32-bit carry-select adder (CSLA) was considered as a demonstration platform, where multiplexers are employed to determine the correct sum output in the final stage. The CSLA [34] typically involves two parallel adders: one operates with the assumption of a zero carry input, and the other assumes a carry input of one. This method accelerates the addition process by evaluating both addition possibilities simultaneously and then selecting the appropriate result. Once both adders have finished their calculations, multiplexers are used to select the correct sum based on the actual carry input.

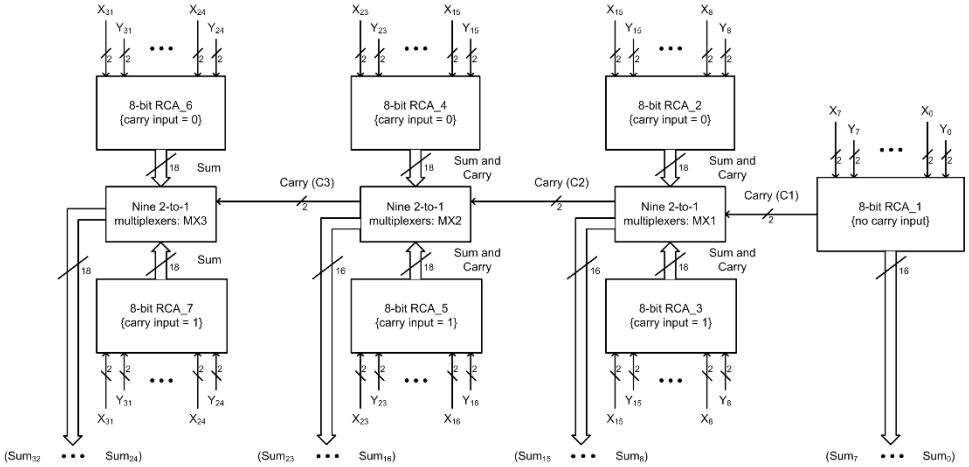


Fig. 9 32-bit IO mode dual-rail encoded asynchronous CSLA featuring an 8-8-8-8 input partition

In Fig. 9, a 32-bit IO mode asynchronous CSLA featuring an 8-8-8-8 input partition is shown [35]. The primary inputs and outputs, and intermediate outputs of the CSLA are dual-rail encoded. X and Y represent the inputs of the CSLA while Sum represents its output. Given an 8-8-8-8 input partition, a least significant 8-bit ripple carry adder (RCA) viz. RCA_1 is used to add input bits X_7 to X_0 with corresponding input bits Y_7 to Y_0. 8-bit RCA_2 and RCA_3 are used to add input bits X_{15} to X_8 with corresponding input bits Y_{15} to Y_8 separately assuming carry inputs of 0 and 1 respectively. The two sets of sum and carry outputs produced corresponding to these additions are forwarded as inputs to a multiplexer logic labeled MX1 which consists of nine 2-bit multiplexers to issue the correct sum bits Sum_{15} up to Sum_8 and the carry output signal C2 using a common select signal, which is the carry output C1 produced by RCA_1. Likewise, RCA_4 and RCA_5 are used to add input bits X_{23} to X_{16} with corresponding input bits Y_{23} to Y_{16} separately assuming carry inputs of 0 and 1 respectively. The two sets of sum and carry outputs produced corresponding to these additions are forwarded as inputs to a multiplexer logic labeled MX2 which consists of nine 2-bit multiplexers to issue the correct sum bits Sum_{23} up to Sum_{16} and the carry output signal C3 using a common select signal, which is the carry output C2 from MX1. Similarly, RCA_6 and RCA_7 are used to add input bits X_{31} to X_{24} with corresponding input bits Y_{31} to Y_{24} separately assuming carry inputs of 0 and 1 respectively. The two sets of sum and carry outputs produced corresponding to these

additions are forwarded as inputs to a multiplexer logic labeled MX3 which consists of nine 2-bit multiplexers to issue the correct sum bits Sum_{32} up to Sum_{24} using a common select signal, which is the carry output C3 from MX2.

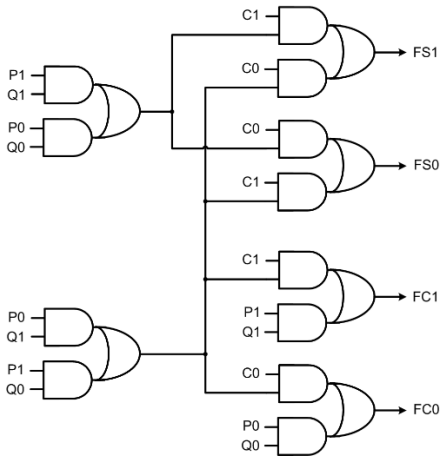


Fig. 10 Asynchronous monotonic full adder (corresponding to RtZ handshaking)

As illustrated in Fig. 9, by maintaining the same RCAs, different multiplexers discussed in Sections 3 and 4 can be used individually to implement 32-bit CSLAs, allowing for their performance evaluation. The RCAs are efficiently implemented using a monotonic full adder described in [16], shown in Fig. 10, which has been designed for RtZ handshaking. In Fig. 10, (P1, P0) and (Q1, Q0) represent the full adder's inputs, while (C1, C0) denotes the carry input. The sum output is denoted as (FS1, FS0), and the carry output is represented by (FC1, FC0). This monotonic full adder is characterized by its early output property. To obtain the logical equivalent of Fig. 10 pertaining to RtO handshaking, the AO22 complex gates should be replaced by their Boolean duals viz. the OA22 complex gates. In [35], an EOQDI full adder was used. Compared to this, the full adder of [16] occupies 44.4% less area for both handshake schemes, and hence it was considered for this work.

We designed several 32-bit IO mode asynchronous CSLAs utilizing traditional and newly proposed multiplexers individually based on dual-rail encoding, and adhering to RtZ and RtO handshaking. Specifically, one stage of the IO mode asynchronous circuit, as shown in Fig. 1a, was constructed, including a register set for the adder inputs, a completion detector, and the 32-bit CSLA functioning as the IO mode asynchronous circuit. The design metrics for the 32-bit CSLAs, incorporating various multiplexers, were evaluated for both RtZ and RtO handshaking, and they are presented in Table 2. The design parameters were determined using Synopsys EDA tools mentioned earlier, following the same low-leakage 28-nm CMOS cell library PVT specification and the methodology followed for the multiplexers. A test bench with over 1000 random inputs was supplied to the CSLAs at a latency of 10 ns for performing functional simulations and tracking the switching activity. The test benches used for RtZ and RtO handshaking are logically equivalent. The design metrics estimated for the CSLAs include forward latency, reverse latency, cycle time (which is the sum of forward and reverse latencies), area, and total power dissipation.

Table 1 presented the forward latency (since multiplexers were implemented independently) alone and not the cycle time. This was because the completion detector's latency was found to exceed that of certain multiplexers when the multiplexers were implemented as a single IO mode asynchronous circuit stage, complicating the performance comparison. However, when the 32-bit CSLA was implemented as an IO mode asynchronous stage, its latency exceeded the completion detector's latency, enabling accurate performance comparisons of various multiplexers. Table 2 presents all three timing parameters (forward latency, reverse latency, and cycle time) highlighting that the latencies in the forward and reverse directions for some IO mode asynchronous CSLAs may be different. This issue has been discussed in previous research [37]. Furthermore, different multiplexers impact the CSLA's timing in distinct ways. The forward latency of the CSLAs was directly measured, while the reverse latency was computed based on the respective gate and net delays specified in the timing reports. The cycle time, representing the duration of a single data transaction, was obtained by summing the forward and reverse latencies.

Table 2 Design parameters of asynchronous 32-bit CSLAs with different multiplexers realized using a 28-nm bulk CMOS process. In the table, FL denotes forward latency, RL denotes reverse latency, and CT denotes cycle time.

Multiplexer used in CSLA	FL (ns)	RL (ns)	CT (ns)	Area (μm^2)	Power (μW)
Corresponding to RtZ handshaking					
DIMS	2.51	1.80	4.31	2893.94	2279
Toms	2.71	2.07	4.78	2701.81	2269
Early output QDI	2.14	1.58	3.72	2468.50	2251
SIDCO	2.53	1.89	4.42	2475.36	2284
SIDCAO	2.56	2.00	4.56	2482.23	2297
Proposed (monotonic)	1.51	0.61	2.12	1768.59	2218
Corresponding to RtO handshaking					
DIMS	2.50	1.78	4.28	2839.04	2264
Toms	2.70	2.05	4.75	2701.81	2264
Early output QDI	2.12	1.55	3.67	2468.50	2239
SIDCO	2.52	1.87	4.39	2475.36	2273
SIDCAO	2.54	1.97	4.51	2482.23	2286
Proposed (monotonic)	1.48	0.59	2.07	1768.59	2205

As shown in Fig. 9, the forward latency of the CSLAs is determined by the total propagation delays of RCA_1, MX1, MX2, and MX3. Within MX1, a delay is introduced by a 2-to-1 multiplexer that is responsible for passing the carry signal to MX2. Similarly, in MX2, another 2-to-1 multiplexer introduces a delay as it forwards the carry signal to MX3. In MX3, a final 2-to-1 multiplexer delay occurs, which generates the sum bit. Therefore, the forward latency of the CSLA in Fig. 9 can be represented by (7), where D_{Reg} stands for the propagation delay of an input register, D_{RCA_1} is the delay of the 8-bit RCA_1, D_{FA} denotes the propagation delay of a full adder, and D_{MUX21} is the delay of a 2-to-1 multiplexer. In (7), the first two terms on the right-hand side are constants, as the same full adder (depicted in Fig. 10) was used in all CSLAs. However, the third term may vary depending on the specific multiplexer employed.

$$FL_{CSLA} = D_{Reg} + D_{RCA\_1} + (3\times D_{MUX21}) = D_{Reg} + (8\times D_{FA}) + (3\times D_{MUX21}) \quad (7)$$

To examine how different multiplexers affect the forward and reverse latencies of CSLAs, let us focus on RtZ handshaking as a test case here. A similar analysis can be conducted for RtO handshaking, which we leave to an interested reader. However, we will present the results of the theoretical delay modeling for both RtZ and RtO handshaking.

By substituting the appropriate multiplexer delays into (7), the forward latency of CSLAs using DIMS, Toms, EOQDI, SIDCO, SIDCAO, and the proposed (monotonic) multiplexers are represented by (8), (9), (10), (11), (12), and (13) respectively. These equations serve as approximate latency models, as they omit the delays associated with interconnects and parasitics for simplifying the theoretical analysis. In the equations, D_{CE2} and D_{AO22} represent the typical propagation delays of a 2-input C-element and an AO22 complex gate, while D_{OR2}, D_{OR3}, and D_{OR4} represent the propagation delays of 2-input, 3-input, and 4-input OR gates, respectively.

$$FL_{CSLA}{}^{DIMS} = D_{Reg} + (8\times D_{FA}) + 3\times(D_{CE2} + D_{OR4}) \tag{8}$$

$$FL_{CSLA}{}^{Toms} = D_{Reg} + (8\times D_{FA}) + 3\times(2\times D_{CE2} + D_{OR2} + D_{OR3}) \tag{9}$$

$$FL_{CSLA}{}^{EOQDI} = D_{Reg} + (8\times D_{FA}) + 3\times(D_{CE2} + D_{OR3}) \tag{10}$$

$$FL_{CSLA}{}^{SIDCO} = D_{Reg} + (8\times D_{FA}) + 3\times(2\times D_{CE2} + D_{OR2}) \tag{11}$$

$$FL_{CSLA}{}^{SIDCAO} = D_{Reg} + (8\times D_{FA}) + (7\times D_{CE2} + 3\times D_{OR2}) \tag{12}$$

$$FL_{CSLA}{}^{Proposed} = D_{Reg} + (8\times D_{FA}) + 3\times D_{AO22} \tag{13}$$

The reverse latency can also be theoretically modeled to point out the delay variations across CSLAs that use different multiplexers. With the inclusion of a monotonic full adder (as shown in Fig. 10), the reverse latency of the CSLA (depicted in Fig. 9) using strong indication or EOQDI multiplexers is generally represented by (14). However, the reverse latency expression for the CSLA utilizing the proposed multiplexer will differ and this will be addressed later in this section.

$$RL_{CSLA} = D_{Reg} + D_{RCA\_1} + (3\times D_{MUX21}) = D_{Reg} + D_{FA} + (3\times D_{MUX21}) \tag{14}$$

By comparing (14) and (7), it becomes evident that the reverse latency is smaller than the forward latency, primarily because only the delay of a single full adder is considered in (14), whereas (7) accounts for the combined delays of eight full adders. This difference arises due to the monotonic full adder, which can be reset early based on the adder inputs, regardless of the carry input. For RtZ handshaking, as seen in Fig. 10, once the signals P1/P0 and Q1/Q0, whichever were 1 earlier, transition to the spacer, both the sum (FS1/FS0) and the carry output (FC1/FC0) of the full adder can also assume the spacer state, regardless of whether C1/C0 assumes the spacer. This means that all the full adders in RCA_1 through RCA_7 can transition to the spacer state at the same time. As a result, when the spacer is applied, the delay of the RCA effectively reduces to the delay of just one full adder, as indicated by the second term in (14). Since conventional multiplexers contain C-elements in their design, the third term in (14) remains identical to that in (7). Therefore, the reverse latency of CSLAs using DIMS, Toms, EOQDI, SIDCO, and SIDCAO multiplexers are given by (15), (16), (17), (18), and (19), respectively. These equations are also approximate models of latency, as they omit the delays associated with interconnects and parasitics for simplicity.

$$RL_{CSLA}{}^{DIMS} = D_{Reg} + D_{FA} + 3\times(D_{CE2} + D_{OR4}) \quad (15)$$

$$RL_{CSLA}{}^{Toms} = D_{Reg} + D_{FA} + 3\times(2\times D_{CE2} + D_{OR2} + D_{OR3}) \quad (16)$$

$$RL_{CSLA}{}^{EOQDI} = D_{Reg} + D_{FA} + 3\times(D_{CE2} + D_{OR3}) \quad (17)$$

$$RL_{CSLA}{}^{SIDCO} = D_{Reg} + D_{FA} + 3\times(2\times D_{CE2} + D_{OR2}) \quad (18)$$

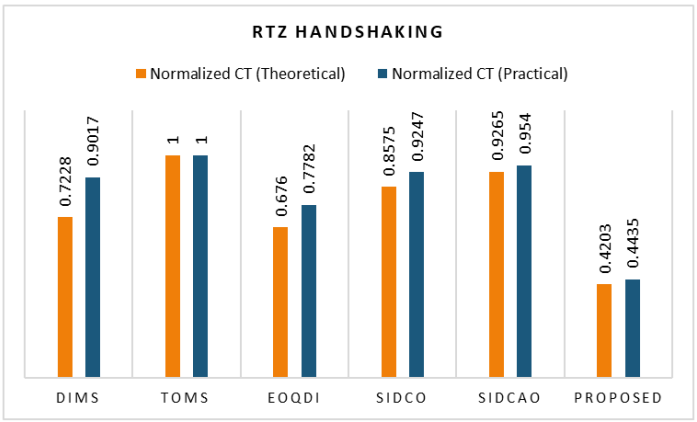
$$RL_{CSLA}{}^{SIDCAO} = D_{Reg} + D_{FA} + (7\times D_{CE2} + 3\times D_{OR2}) \quad (19)$$

The reverse latency of the CSLA incorporating the proposed multiplexer is theoretically given by (20). Similar to (15) through (19), the second term on the right-hand side of (20) represents the delay of a full adder corresponding to RCA_1. Referring to Fig. 6a, since the proposed multiplexer is monotonic, once A1/B1 (whichever was 1 earlier) or A0/B0 (whichever was 1 earlier) transitions to the spacer state, M1/M0 (whichever was 1 earlier) will also transition to the spacer state, regardless of the state of the select signal (S1/S0). It is important to note that (A1, A0) and (B1, B0), which represent the multiplexer inputs, combine the corresponding output pairs of the full adders in RCA_2 through RCA_7. Consequently, all the multiplexers in MX1, MX2, and MX3 can simultaneously transition to the spacer state based on the outputs from RCA_2 to RCA_7, resulting in a delay that is equivalent to a single multiplexer, as indicated by the third term in (20).

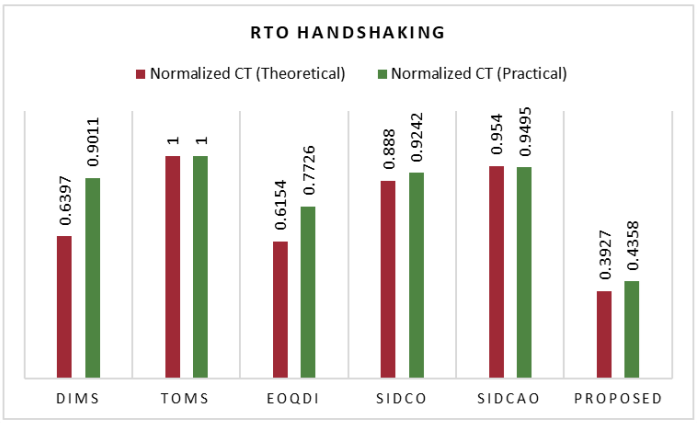
$$RL_{CSLA}{}^{Proposed} = D_{Reg} + D_{FA} + D_{AO22} \quad (20)$$

Equations (8) to (13) and (14) to (20) provide insights into the variations in cycle time of CSLAs that incorporate different multiplexers. By substituting the average propagation delays of the gates from the cell library [33] into these equations, we calculated the theoretical cycle time for CSLAs utilizing various multiplexers, specifically for RtZ handshaking. For CSLAs corresponding to RtO handshaking, we calculated the cycle time after considering the dual versions of the gates described in (8) to (13) and (15) to (20), excluding the C-element. To normalize the theoretical cycle time, we divided the cycle time of each CSLA by the maximum cycle time within the group, doing so separately for both RtZ and RtO handshaking. A similar normalization procedure was applied to the practical cycle times of CSLAs, using the estimates provided in Table 2 for RtZ and RtO handshaking. Figs. 11a and 11b portray a comparison between the theoretical and practical (normalized) cycle times of CSLAs incorporating different multiplexers based on RtZ and RtO handshaking, respectively.

Figs. 11a and b demonstrate that while there are noticeable variations between the theoretical and practical cycle times for certain CSLAs, primarily due to the simplified theoretical delay models, a strong correlation is seen between the theoretical and practical delays. This validates the accuracy of our theoretical delay modeling. Additionally, Figs. 11a and b show that the CSLA utilizing the proposed monotonic multiplexer achieves a substantial reduction in cycle time when compared to those using strong indication or EOQDI multiplexers. As shown in Table 2, while the CSLA with the EOQDI multiplexer has a lower cycle time than the one comprising the strong indication multiplexer, the CSLA employing the monotonic multiplexer achieves a 43% (43.6%) reduction in cycle time relative to the CSLA incorporating the EOQDI multiplexer for RtZ (RtO) handshaking.



(a)



(b)

Fig. 11 Comparison of the theoretical and practical (normalized) cycle time of CSLAs incorporating different multiplexers corresponding to the handshake schemes: (a) RtZ and (b) RtO

Because the same full adder (shown in Fig. 10) was utilized in constructing the CSLAs, and both the input registers and the completion detector external to the CSLA remain unchanged, the area variations between the CSLAs listed in Table 2 are solely due to the differences in the areas of the multiplexers present in the CSLAs. Fig. 12 depicts the areas of various multiplexers for RtZ and RtO handshaking protocols.

The area occupied by several multiplexers is observed to be identical for both RtZ and RtO handshaking. This similarity arises from the fact that certain gate pairs in the digital cell library [33] have equivalent areas – such as 2-input and 3-input OR gates having the same area as 2-input and 2-input AND gates for a normal drive strength. Additionally, AO22 and OA22 complex gates with normal drive strength have identical areas. This explains why, in Table 2, many CSLAs have the same areas for both handshaking types.

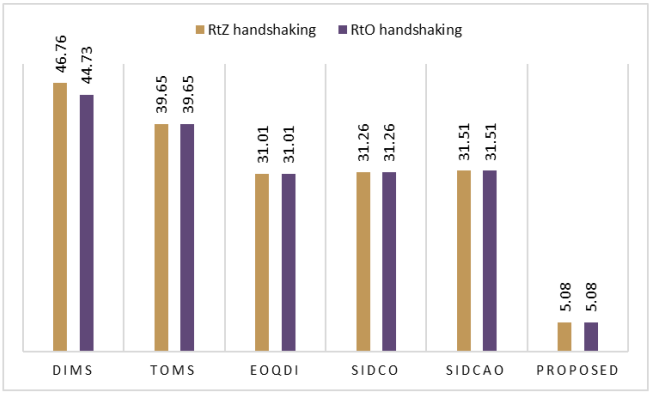


Fig. 12 Area (in μm^2) of different multiplexers corresponding to the handshake schemes: (a) RtZ and (b) RtO

Compared to strong indication and EOQDI multiplexers illustrated in Figs. 2 through 5, the proposed monotonic multiplexer in Fig. 6 requires significantly fewer gates and logic components, resulting in a much smaller area, as seen in Fig. 12. Specifically, the proposed multiplexer occupies 83.6% less area than the EOQDI multiplexer, and the CSLA with the proposed multiplexer occupies 28.4% less area compared to the CSLA with the EOQDI multiplexer. However, despite this reduction in area, the power dissipation difference between the two CSLAs is rather minimal, with only a 1.5% decrease for RtZ and RtO handshaking. The reason for this shall be explained next. As observed in Table 2, the variation in power dissipation across various CSLAs is not much. This is due to the design of the multiplexers and full adders within the CSLAs, which adhere to the monotonic cover constraint. As explained in Section 3.1, the monotonic cover constraint activates a single, unique signal path from a primary input to a primary output, reducing unnecessary switching and signal transitions in IO mode asynchronous circuits, thus minimizing power variation between logically similar but structurally distinct CSLAs. Minor differences in power do exist, and this is due to the differences in multiplexers' logic. The power dissipation of input registers, the completion detector, and full adders remain almost the same since they are shared across all CSLAs. These elements contain more logic than the multiplexers in the CSLA, which causes them to dominate the total power dissipation. Consequently, the variation in power dissipation among different CSLAs in Table 2 is rather minor.

In contrast, Table 1 revealed a noticeable variation in power dissipation, as it focused specifically on the design metrics of individual multiplexers. The overall conclusion from Table 2 is that the CSLA with the proposed multiplexer significantly reduced both cycle time and area compared to traditional multiplexers, without an increase in power dissipation. This implies the proposed multiplexer enables more optimization when considering all the design metrics together. Furthermore, Table 2 shows that the proposed multiplexer when used for RtO handshaking enables a slight decrease in both cycle time and power when used for RtZ handshaking, and this is due to differences in the gate types corresponding to the handshake schemes.

In synchronous design, the power and delay product commonly serves as a figure of merit for low-power or low-energy evaluation [38]. The equivalent metric in IO mode asynchronous design is the power and cycle time product (PCTP). Hence, we computed

the PCTP for each CSLA, using the design metrics listed in Table 2 for both RtZ and RtO handshaking. The calculated PCTPs were then normalized by dividing them by the highest PCTP for RtZ and RtO handshaking separately. The normalized PCTPs corresponding to RtZ and RtO handshake schemes are presented in Figs. 13a and 13b, respectively. As seen in Fig. 13, the CSLA with the proposed multiplexer has a substantially lower PCTP compared to CSLAs using other multiplexers. Specifically, the CSLA with the proposed multiplexer achieves a 43.9% (44.5%) reduction in PCTP for RtZ (RtO) handshaking compared to the CSLA featuring the EOQDI multiplexer.

Fig. 13 Normalized PCTP of CSLAs incorporating different multiplexers corresponding to the handshake schemes: (a) RtZ and (b) RtO

An optimized IO mode asynchronous CSLA was presented in [35], which utilized the EOQDI full adder of [36] and the SIDCO multiplexer of [32]. We implemented this CSLA as well and estimated its design metrics for the two handshake schemes, given below. These metrics were estimated by following the same design methodology discussed previously.

- RtZ handshaking: Forward latency = Reverse latency = 2.54 ns, and Cycle time = 5.08 ns; Area = 3158.50 μm^2; Power = 2444 µW
- RtO handshaking: Forward latency = Reverse latency = 2.50 ns, and Cycle time = 5 ns; Area = 3158.50 μm^2; Power = 2438 µW

Compared to these design parameters, the CSLA realized using the monotonic full adder of [16] and the proposed monotonic multiplexer achieved the following reductions in design metrics: (i) 58.3% less cycle time, 44% less area, 9.2% less power, and 62.1% less PCTP for RtZ handshaking, and (ii) 58.6% less cycle time, 44% less area, 9.6% less power, and 62.6% less PCTP for RtO handshaking.

6. CONCLUSION

This article introduced a new IO mode multiplexer design that falls under the monotonic class. The proposed multiplexer is notable for its gate-level efficiency, requiring only ten transistors for a static CMOS implementation. Compared to an existing optimized EOQDI multiplexer, the proposed monotonic multiplexer demonstrates a latency reduction of 67% (69%), an area reduction of 84% (84%), and a power reduction of 66% (67%) for RtZ (RtO) handshaking, based on implementation a 28-nm CMOS technology. Since the multiplexer is a relatively small component, its performance is best evaluated in a broader circuit context. To do this, we incorporated conventional multiplexers and the proposed multiplexer individually into IO mode asynchronous 32-bit CSLAs, keeping the compute element (the full adder) consistent. The performance evaluation revealed that the CSLA utilizing the proposed multiplexer achieves a 43% (44%) reduction in cycle time, a 28% (28%) reduction in area, and a 44% (45%) reduction in PCTP compared to the CSLA with an EOQDI multiplexer for RtZ (RtO) handshaking.

REFERENCES

[1] J. Sparsø and S. B. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*. Dordrecht: Kluwer Academic Publishers, 2001.

[2] S. M. Nowick and M. Singh, "Asynchronous Design – Part 1: Overview and Recent Advances", *IEEE Design & Test*, vol. 32, pp. 5-18, June 2015.

[3] C. H. Van Berkel, M. B. Josephs and S. M. Nowick, "Applications of Asynchronous Circuits", *Proc. IEEE*, vol. 87, pp. 223-233, Feb. 1999.

[4] A. J. Martin and M. Nystrom, "Asynchronous Techniques for System-on-Chip Design", *Proc. IEEE*, vol. 94, pp. 1089-1120, June 2006.

[5] I. David, R. Ginosar and M. Yoeli, "Self-timed is Self-Checking", *J. Electron. Test.: Theory Appl.*, vol. 6, pp. 219-228, Apr. 1995.

[6] G. F. Bouesse, G. Sicard, A. Baixas and M. Renaudin, "Quasi Delay Insensitive Asynchronous Circuits for Low EMI", In Proceedings of the 4th International Workshop on Electromagnetic Compatibility of Integrated Circuits, 2004, pp. 27-31.

[7] Z. Yu, S. B. Furber and L. A. Plana, "An Investigation into the Security of Self-Timed Circuits", In Proceedings of the 9th International Symposium on Advanced Research in Asynchronous Circuits and Systems, 2003, pp. 206-215.

[8] L. A. Plana, P. A. Riocreux, W. J. Bainbridge, A. Bardsley, S. Temple, J. D. Garside and Z. C. Yu, "SPA – A Secure Amulet Core for Smartcard Applications", *Microprocess. Microsyst.*, vol. 27, pp. 431-446, Oct. 2003.

[9] A. J. Martin, "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, USA, 1990.

[10] A. J. Martin and P. Prakash, "Asynchronous Nano-Electronics: Preliminary Investigation", In Proceedings of the 14th IEEE International Symposium on Asynchronous Circuits and Systems, 2008, pp. 58-68.

[11] C. L. Seitz, "System Timing" In *Introduction to VLSI Systems*; Mead, C., Conway, L., Eds.; Addison-Wesley: Reading, MA, USA, 1980.

[12] C. Brej, *Early Output Logic and Anti-Tokens*. Ph.D. Thesis, The University of Manchester, UK, September 2005.
[13] K. S. Stevens, R. Ginosar and S. Rotem, "Relative Timing", *IEEE Trans. VLSI Syst.*, vol. 11, pp. 129-140, Feb. 2003.
[14] J. Cortadella, A. Kondratyev, L. Lavagno and C. Sotiriou, "Coping with the Variability of Combinational Logic Delays", In Proceedings of the IEEE International Conference on Computer Design, 2004, pp. 1-4.
[15] V. I. Varshavsky, "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 A.V. Yakovlev), Kluwer Academic Publishers: New York, USA, 1990.
[16] P. Balasubramanian and W. Liu, "High-Speed and Energy-Efficient Asynchronous Carry Look-Ahead Adder", *PLOS ONE*, vol. 18, p. e0289569, Oct. 2023.
[17] P. Balasubramanian and N.E. Mastorakis, "Speed, Power and Area Optimized Monotonic Asynchronous Array Multipliers", *J. Low Power Electron. Appl.*, vol. 14, p. 1, Jan. 2024.
[18] D. E. Muller and S. Bartky, "A Theory of Asynchronous Circuits", In Proceedings of the International Symposium on the Theory of Switching (Part I), 1957, pp. 204-243.
[19] P. A. Beerel, R. O. Ozdag and M. A. Ferretti, *A Designer's Guide to Asynchronous VLSI*, Cambridge: Cambridge University Press, 2010.
[20] M. Shams, J. C. Ebergen and M. I. Elmasry, "A Comparison of CMOS Implementations of an Asynchronous Circuits Primitive: The C-element", In Proceedings of the International Symposium on Low Power Electronics and Design, 1996, pp. 93-96.
[21] L. S. Heck, M. T. Moreira and N. L. V. Calazans, "Hardening C-elements against Metastability", In Proceedings of the 24th IEEE International Conference on Electronics, Circuits and Systems, 2017, pp. 314-317.
[22] T. Verhoeff, "Delay-Insensitive Codes – An Overview", *Distrib. Comput.*, vol. 3, pp. 1-8, Mar. 1988.
[23] B. Bose, "On Unordered Codes", *IEEE Trans. Comput.*, vol. 40, pp. 125-131, Feb. 1991.
[24] M. T. Moreira, R. A. Guazzelli and N. L. V. Calazans, "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, 2012, pp. 1-6.
[25] J. Sparsø and J. Staunstrup, "Delay-Insensitive Multi-Ring Structures", *Integr. VLSI J.*, vol. 15, pp. 313-340, Oct. 1993.
[26] T. Sasao, *Switching Theory for Logic Synthesis*, Dordrecht: Kluwer Academic Publishers, 1999.
[27] P. Balasubramanian, "Comparative Evaluation of Quasi-Delay-Insensitive Asynchronous Adders Corresponding to Return-to-zero and Return-to-one Handshaking", *FU: Elec. Ener.*, vol. 31, no. 1, pp. 25-39, 2018.
[28] W. B. Toms, *Synthesis of Quasi-Delay-Insensitive Datapath Circuits*, Ph.D. Thesis, The University of Manchester, UK, February 2006.
[29] W. B. Toms and D. A. Edwards, "Indicating Combinational Logic Decomposition", *IET Comput. Digit. Tech.*, vol. 5, pp. 331-341, July 2011.
[30] R. K. Brayton, G. D. Hachtel, C. T. McMullen and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Springer: New York, 1984.
[31] R. Rudell, *Logic Synthesis for VLSI Design*, Ph.D. Thesis, University of California, Berkeley, USA, 1989.
[32] P. Balasubramanian and D.A. Edwards, "Power, Delay and Area Efficient Self-Timed Multiplexer and Demultiplexer Designs", In Proceedings of the 4th International Conference on Design & Technology of Integrated Systems in Nanoscale Era, 2009, pp. 173-178.
[33] Synopsys Databook, Synopsys SAED_EDK32/28_CORE Databook, Revision 1.0.0. January 2012.
[34] O. J. Bedrij, "Carry-Select Adder", *IRE Trans. Electron. Comput.*, vol. EC-11, pp. 340-346, June 1962.
[35] P. Balasubramanian, "Asynchronous Carry Select Adders", *Eng. Sci. Technol. Int. J.*, vol. 20, pp. 1066-1074, June 2017.
[36] P. Balasubramanian, "A Robust Asynchronous Early Output Full Adder", *WSEAS Trans. Circ. Syst.*, vol. 10, pp. 221-230, July 2011.
[37] P. Balasubramanian and S. Yamashita, "Area/Latency Optimized Early Output Asynchronous Full Adders and Relative-Timed Ripple Carry Adders", *SpringerPlus*, vol. 5, p. 440, Apr. 2016.
[38] J. M. Rabaey, A. Chandrakasan and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, 2nd ed.; London: Pearson Education, 2003.