Quasi Delay Insensitive FIFOs: Design Choices Exploration and Comparison

Taciano A. Rodolfo[1], Marcos L. L. Sartori[1], Matheus T. Moreira[2], Ney L. V. Calazans[1]
[1]PUCRS - School of Technology - Ipiranga Ave, 6681 - Porto Alegre - RS - Brazil, 90619-900
[2]Chronos Tech LLC - 9444 Waples Street - San Diego - CA - USA, 92121
{taciano.rodolfo, marcos.sartori}@acad.pucrs.br, matheus.trev@gmail.com, ney.calazans@pucrs.br

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401566

Abstract— **This paper explores asynchronous FIFOs design choices, more specifically FIFOs from the quasi-delay insensitive (QDI) template family. It proposes eight different asynchronous FIFO structures on a CMOS 45nm technology, using a QDI standard cell library. Structures are exercised through analog-mixed-signal simulation, ranging from nominal to subthreshold supply voltages. Follows a comparison of area, throughput and power efficiency. The experimental results allow inferring a technique for designers to select the most adequate QDI FIFO flavor for specific circuits. Insight on the experiments assesses the beneficial and/or limiting effects of using the specific cell library.**

I. Introduction

Order is one of the most basic concepts in computing. For some applications, treating inputs in the inverse order of their arrival may be mandatory [1]. Specific applications can even require a random or seemingly random treatment of input data [2]. But the overwhelming majority of computations demand a first-come, first-served strategy for dealing with inputs. First-in-first-out (FIFO) circuits provide structure, behavior and storage to enable in-order data treatment.

A FIFO contains a data storage block, added with input and output ports. The input port controls the writing of new data to the storage block and the output port controls the reading of new data from this block. Reading from and writing to a FIFO are independent operations which may occur concurrently, as long as access is not to a same storage position, which is often enforced by the FIFO structure. Also, reading from an empty FIFO and storing new data to a full FIFO are invalid operations, and the FIFO structure enforces the avoidance of these as well.

Ports and storage can share a single clock, which characterizes a *synchronous FIFO*. If a FIFO implementation is clock-based, but the input and output clocks are allowed to be distinct (in frequency and/or in phase and/or duty cycle etc.), the resulting circuit is called a *bi-synchronous FIFO* or *asynchronous FIFO*. The last denomination is inadequate and will not be used herein. Note that a bi-synchronous FIFO is in fact a globally asynchronous, locally synchronous (GALS) circuit. Bi-synchronous FIFOs are important and much used to match systems with widely different speed characteristics and/or that employ distinct clock domains. If a FIFO does not use clock(s) to synchronize its operations it is a truly *asynchronous FIFO* (AFIFO). AFIFOs operate with local handshake protocols, both at its ports and internally. As any asynchronous circuit [3], AFIFOs can be further classified according to the handshake protocol they use, their data encoding scheme etc., into one of two encompassing design template families, bundled-data (BD) and quasi-delay insensitive (QDI).

Besides the previous synchronization criterion-based FIFO taxonomy, there is another taxonomy orthogonal to the first, based on the criterion on where in the internal storage lies the FIFO input and output. If the hardware structure uses fixed storage positions for both input and output ports, the FIFO is called *flow-through* [4], meaning that data entering the FIFO must evolve along all or some storage positions until it reaches the output [4]. Another organization, *pointer-based FIFOs*, manipulate internal storage addresses in a circular way, with internal fixed-module counters. This dynamically defines the FIFO input and output positions on the storage. Clearly, there is a design trade-off implied when choosing one of these organizations [5]: flow-through FIFOs typically have high throughput and poor latency, while pointer-based FIFOs have larger control complexity. Often, synchronous FIFOs and bi-synchronous FIFOs are built with pointer-based organizations while AFIFOs usually rely on flow-through organizations.

This work explores the characteristics and trade-offs of the little-known class of QDI AFIFOs. Before addressing the details of QDI AFIFOs it is beneficial to provide a set of motivations to better dominate such hardware structures, their use and design process. There is an increasing demand for QDI design in general, and for QDI AFIFOs in particular, arising from several developments on current semiconductor technology. First, the wire form factor in advanced technologies creates increasing problems for routing parallel wires of buses inside an integrated circuit (IC). Thus, designers tend to employ serialized communication more often, and QDI links are an attractive option due to their performance and robustness to variations, as detailed e.g. by Tse et al. in [6]. More QDI links imply more (QDI) FIFOs to adapt serial/parallel IC parts. Second, since the start of the 21st century, complex IC design growingly substitute bus-based for network-on-chip(NoC)-based intra-chip communication. NoC routers are hardware parts formed basically by multiple FIFOs, which dominate the router design, and a few control circuits. Synchronous NoCs do not scale well, and many researchers and industries today rely on GALS (which can be inefficient as well) or asynchronous QDI NoCs as largely justified e.g. by Beigné et al. [7]. Third, 3D IC integration is a path to dominate system complexity and efficiently integrate heterogeneous ICs. Communication between *chiplets* in 3D IC stacks is better achieved using asynchronous communication, and QDI design is used here, as described e.g. by Vivet et al. [8]. Complex ICs are often a collection of synchronous intellectual property modules (IPs), either acquired or developed inhouse, involving multiple teams and different design cultures. Integrating a large number of independently developed IPs can be quite challenging, and QDI communication can reduce this complexity. In fact, companies specialized in providing intra-chip QDI communication to third-party GALS designs are today a reality [9].

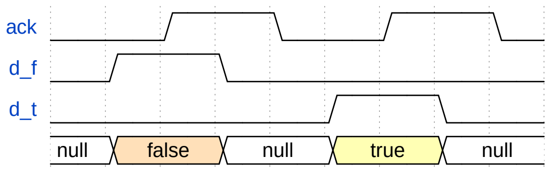
II. Basics and The Proposed Approach

Synchronous circuits rely on a global clock signal to provide a discrete common time reference. Typically, the clock is a wave with a period greater than the worst combinational logic delay in any path in the circuit between two consecutive registers. All synchronous circuit registers simultaneously capture data (within a certain time window, computed considering clock propagation time, skew and jitter) as determined by clock transitions at registers. These characteristics guarantee registers capture data only after all combinational logic has finished computing. Asynchronous circuits have no such single common time reference. To ensure correct operation, asynchronous logic blocks communicate with each other using essentially local *handshake channels* [10]. This approach eliminates the need for distributing a global clock. It also produces circuits that operate based on the average delay of combinational blocks, not on the worst-case circuit path.

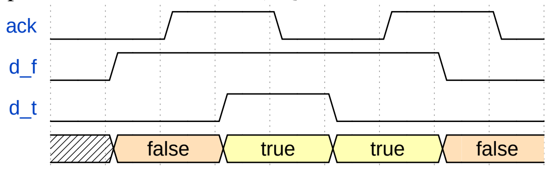
Handshake protocols comprise two distinct steps: (i) *request*, when an entity announces (or requires) data availability; (ii) data *acknowledgment*, when another entity acknowledges (or grants) data, enabling subsequent transmissions. The implementation of these steps are protocol-dependent, and such protocols can be categorized in two classes: (i) 2-phase (2ph) protocols; and (ii) 4-phase (4ph) protocols. A 2ph protocol implements the handshake steps with a single transition in each control signal, allowing transmission of new data immediately after acknowledgment. A 4ph protocol in turn requires that request and acknowledgment signals return to a neutral state prior to the transmission of new data.

The use of dedicated request/acknowledge signals separated from data lines characterizes what is known as the *bundled data* (BD) design style. BD allows simpler, close to synchronous, data path implementations, at the expense of more complex timing assumptions. Since combinational logic data transformation must be transparent to the local handshake protocol [10], requests must arrive at the consumer only after all computations on channel data are concluded and results are ready at the consumer inputs, otherwise the latter can capture incorrect data. This poses a design challenge, and the control signals often require delay lines to match its propagation delay to that of the data path. As an alternative, the request information is embedded within the data itself, by using delay-insensitive (DI) codes. Circuits using DI codes follow either Delay Insensitive (DI) or Quasi Delay Insensitive (QDI) design *templates*. The DI template class is ideal for maximum robustness, but it was demonstrated to be of little practical use [11]. Besides using DI codes, QDI templates also assume that selected wire forks within the design are *isochronic* (i.e. the delays from the wire "input" to both ends of the fork are identical, or differ by a negligible amount). The QDI assumption is considered the least compromise between robustness and practicality. A QDI circuit design requires less restrictive timing assumptions than BD circuits. This makes QDI circuits less sensitive to process, voltage and temperature (PVT) variations and aging. QDI circuits rely on DI codes and on *completion detection* circuits to recognize data availability.

Figure 1 depicts the QDI handshake push protocols used in this work. Here *push* protocols stand as those where requests follow the same sense as data, by opposition to *pull* protocols, where requests follow the opposite sense of data. In pull protocols the control wire is accordingly renamed **req**. Both examples employ the dual-rail DI code, other DI codes are possible, e.g. any m-of-n code. The *return to zero* (RTZ) protocol depicted in Figure 1(a) is a 4-phase, level-sensitive protocol. It requires the transmission of a spacer code (null) between two consecutive data transmissions. The *no return* (NR) protocol depicted in Figure 1(b) is a 2-phase, transition-signaling protocol. It can transmit twice the amount of data as RTZ with the same switching activity. However, handling this NR or other 2-phase protocols normally requires more complex circuitry than this RTZ or other 4-phase protocols.



(a) 4-phase return-to-zero (RTZ) QDI handshake, level-encoded.



(b) 2-phase no-return (NR) QDI handshake, transition-encoded.

Fig. 1: Two QDI handshake communication protocols.

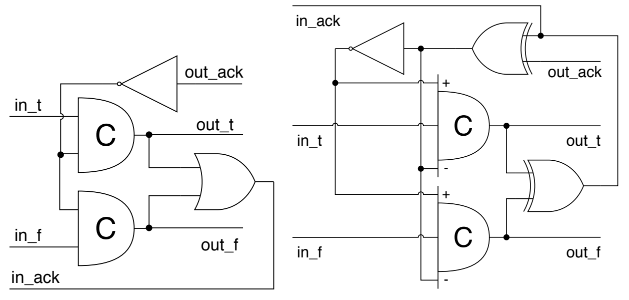
QDI circuit design often relies on the availability of a set of specific logic gates, distinct from ordinary ANDs, ORs and inverters. It benefits from gates with hysteretic functions such as C-Elements. Hysteretic gates in CMOS are network of transistors with feedback. Although their behavior can be built with ordinary gates, this is sub-optimal in area and performance. Also, feedback lines usually comprise the isochronic forks, and are thus better left inside a cell and not generated by routing tools unaware of fork constraints. This work employs the open-source ASCEnD-FreePDK45 library [12], which targets the FreePDK $45\,\mathrm{nm}$ predictive bulk

CMOS technology.

III. Investigated QDI AFIFOs Architecture

Eight 8-stage, 32-bit QDI AFIFOs were described in Verilog at the netlist level, instantiating ASCEnD-FreePDK45 cells. Each AFIFO is a combination of choosing one of two DI codes, one of two handshake protocols and one of two **ack** generation strategies. The DI code choices are dual-rail (DR) and 1-of-4 (1of4), by far the two most common in QDI asynchronous designs. Communication protocols are also chosen among the two most used ones, 4ph RTZ and 2ph transition-signaling push protocols, described in Section II. The acknowledgment generation strategy can be either one where a single **ack** wire passes between each two consecutive AFIFO stages (here called *busack*) or a *unitack* strategy, where each DI code unit in a stage generates an **ack** for the previous stage. For the DI codes used here, this implies 32 **ack** wires for DR codes and 16 **ack** wires for 1of4 codes. Remember that 32-bit values are encoded in 64 wires for both DR and 1of4 DI codes (a DR code unit is two wires for each bit and the 1of4 code unit is 4 wires, corresponding to 2 bits). Externally, each AFIFO has input and output ports with a single wire **ack** as output and input, respectively. This implies that every AFIFO first stage includes an **ack** generation tree. All implemented QDI AFIFOs are obviously register-only pipelines.

Figure 2 depicts the single-bit DR register structure employed here. The 4ph DR register in Figure 2(a) is a weak-conditioned half buffer (WCHB) register [10], similar to those employed in null-convention logic templates (NCL) [13]. The 2ph DR register of Figure 2(b)[1] is based on the LETS-style Mousetrap pipeline described in [14]. This register employs XOR gates and C-elements to control the propagation of alternating even and odd phase data through the pipeline. The 1of4 registers are respective extensions of the former two.



(a) 4ph DR. (b) 2ph DR.

Fig. 2: Single-bit QDI DR registers used in QDI AFIFOs.

All QDI AFIFOs have consistently similar interfaces, changing only how to represent data, depending on the selected DI code. The write-side interface has a 32-bit input port to receive user data and a 1-wire output port to signal **ack** when the data has been properly received by the QDI AFIFO. Similarly, the read-side interface has a 32-bit output port to deliver data and a 1-wire output port to allow the user to signal **ack** when the data has been received. The number of wires and their arrangement used to encode these 32 bits vary according the DI encoding scheme. For instance, DR uses 32 pairs of wires to represent 32-bit data, whereas 1of4 encodes the same 32-bit data using 16 4-wire bundles. Both 1of4 and DR codes require the same amount of wires, other DI codes may differ.

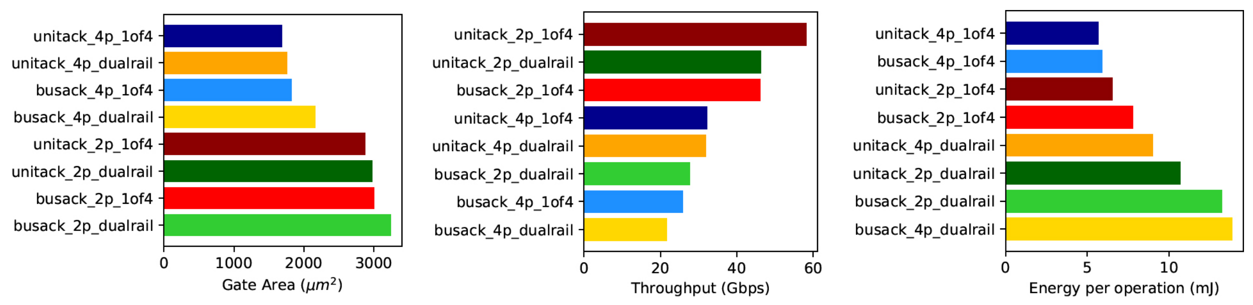
[1]Note the asymmetric inputs, marked with '+' or '-', which only contribute resp. to set or to reset the output [3].

IV. Experimental Setup and Results

Analog mixed signal (AMS) simulations allow to precisely evaluate the impact of protocol, DI encoding and acknowledgment bundling on QDI AFIFOs. Their netlist-level descriptions were placed and routed using Cadence Genus and Innovus to produce physical layouts. After logic and physical synthesis, parasitic analog (PEX) netlists extracted from layouts using Mentor Calibre were simulated using Cadence Xcelium 19, with digital testbenches.

Originally, six digital testbenches were developed to cover the combinations of protocols and DI codes. The 4ph AFIFOs require testbenches that use the RTZ protocol. The 2ph AFIFOs require the use of a protocol that alternates between odd and even phases, by changing a single wire per code transmission. Both level-encoded transition signaling (LETS) [14] and pure transition-signaling (NR) [15] are suitable choices. However, experiments revealed identical results for all metrics for LETS and NR. Therefore, all 2ph results apply to both, but there is only reference to the NR protocol, for simplicity. As a consequence, only four out of the original six testbenches were required: (i) DR RTZ; (ii) DR NR; (iii) 1of4 RTZ; (iv) 1of4 NR. These testbenches cover all eight circuits, since the acknowledgment bundling scheme choice does not change the circuit interface or the expected external behavior. The testbenches simulate ideal, zero-delay environments. They provide new random data as soon as their respective circuits are ready. They also consume data coming from the circuit immediately after these become available. This arrangement sustains each AFIFO operating at its maximum throughput capacity. All analog simulations employ transistor models at the typical corner. Furthermore, the digital testbench automatically verifies the AFIFO output correctness. This last feature is useful to determine the point where supply voltage is no longer feasible to produce correct operation in voltage scaling scenarios.

For each simulation, the test environment measures the average power and the average cycle time, a computation conducted over all measured times between every two consecutive outputs. The overall results appear in Figure 3: (a) depicts the gate area of each circuit extracted from synthesis, more realistic figures considering wire congestion are floorplan-dependent, and thus are ignored here; (b) depicts the maximum throughput, computed as the number of bits transported in a cycle divided by the average cycle time; (c) depicts the energy required to transmit one 32-bit word in each AFIFO, computed as the post-layout average power multiplied by the average cycle time. From these, it is possible to reason about the relative merits of each QDI AFIFO organization.



(a) Gate area. (b) Maximum throughput. (c) Power efficiency.

Fig. 3: Results from synthesis and simulation for all eight QDI AFIFOs at nominal supply (1 V). Best results on top.

Under all metrics, the unitack bundling scheme performed better than their busack bundling counterparts. This is mainly attributed to the lack of C-Element trees merging the acknowledge signals from multiple codes at each pipeline stage. However, the area depicted in Figure 3(a) only accounts for gate area. If the AFIFO is spread over a sufficiently long distance, wire congestion may become an issue. In this situation it might be advisable the use a busack scheme instead.

The adoption of 2ph protocols or 1of4 codes are both techniques that reduce switching activity by half when compared to 4ph DR schemes. A 2ph protocol reduces switching by eliminating the spacer, i.e. every data line toggle carries meaningful information. A 1of4 code reduces switching activity by packing two bits per unit code, sending the double amount of data per transition. Therefore, using 1of4 codes in conjunction with a 2ph protocol should yield the highest power efficiency. However, results in Figure 3(c) show the 4ph 1of4 AFIFO has the highest power efficiency. This is due to the higher complexity of 2ph registers and their reliance on XOR gates. Still, 2ph protocols are considerably faster than 4ph protocols. The lack of spacers yields a more efficient use of the channel time-wise, at additional costs in area and power.

Power efficiency improves when supply voltage reduces [16]. QDI circuits are highly tolerant to supply downscaling, as recently demonstrated [17]. Figure 4 shows the voltage scaling impact, with results of multiple simulations run over a range of supplies, from $0.3\,\mathrm{V}$ to $1.0\,\mathrm{V}$, in $100\,\mathrm{mV}$ steps. Several observations can be then inferred. E.g. the dotted horizontal lines in Figure 4 show that reducing the supply voltage by 40% to $0.6\,\mathrm{V}$, the unitack 2ph 1of4 AFIFO presents the same maximum throughput as the unitack 4ph 1of4 AFIFO at nominal voltage ($> 30Gbps$), but spends just around one third of the energy per operation ($2mJ$ versus $5.9mJ$).

V. Conclusions and Future Work

QDI AFIFOs are a quite efficient way to integrate complex IPs in both, intra-chip and extra-chip environments [8]. This work provides insights on how a set of the most used design choices of codes, protocols and **ack** generation can influence the performance of such AFIFOs.

Experiments relied on the support of ASCEnD-FreePDK45, a cell library supporting asynchronous QDI design. Not all

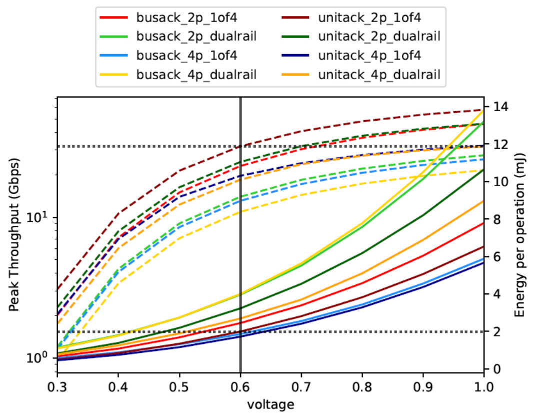


Fig. 4: The voltage scaling impact on QDI AFIFOs. Dashed lines are throughput and solid lines are energy efficiency figures.

library capabilities were used. For example, the library supports the return-to-one (RTO) protocol [18], and a comparison between the use of RTZ and RTO is an interesting work to conduct, since they use distinct cells of the library and imply distinct trade-offs. Also, some of the ASCEnD-FreePDK45 features somehow limited experiments. For example, there is only one XOR gate in this library, a 2-input minimum drive cell. This limitation is seen as leading to some of the non-intuitive results mentioned in Section IV. Adding more driving capabilities and 3-input XOR gates could change several of the obtained numerical results, maybe significantly.

ASCEnD-FreePDK45 employs a predictive design kit and the device models in this PDK are not as accurate as those in commercial technologies. For example, the experiments could not reach or go beyond the minimum energy point (MEP) in the voltage scaling experiments. An ongoing work is to use an industrial design kit to reproduce the experiments and explore deep-subthreshold effects in QDI AFIFOs.

Other comparisons may as well be interesting, such as investigating the effect of using pointer-based FIFO organizations against flow-through QDI AFIFOs proposed herein, or extending the analysis to consider variations on the data width and compute its impact on the **ack** generation scheme choice.

ACKNOWLEDGMENTS

Research partially funded by the Brazilian government funding organisms CNPq (grant no. 312917/2018-0) and CAPES. Authors acknowledge also the HP Brazil partial support.

REFERENCES

[1] P. L. Lanzi, "XCS with Stack-Based Genetic Programming," in *The Congress on Evolutionary Computation (CEC)*, May 2003, pp. 1186–1191.

[2] J. Ye, Y. Hu, and X. Li, "RPUF: Physical Unclonable Function with Randomized Challenge to Resist Modeling Attack," in *IEEE Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, Dec. 2016, pp. 1–6.

[3] P. A. Beerel, R. O. Ozdag, and M. Ferretti, *A Designer´s Guide to Asynchronous VLSI*. Cambridge University Press, 2010.

[4] E. Brunvand, "Low latency self-timed flow-through FIFOs," in *Sixteenth Conference on Advanced Research in VLSI*, May 1995, pp. 76–90.

[5] M. Jhamb, R. K. Sharma, and A. K. Gupta, "A High Level Implementation of a High Performance Data Transfer Interface for NoC," *International Journal of Computer and Information Engineering*, vol. 8, no. 10, pp. 1943–1948, Sept. 2014.

[6] J. Tse, B. Hill, and R. Manohar, "A Bit of Analysis on Self-Timed Single-Bit On-Chip Links," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, May 2013, pp. 40–49.

[7] E. Beigné, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An Asynchronous NoC Architecture Providing Low Latency Service and Its Multi-Level Design Framework," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, May 2005, pp. 54–63.

[8] P. Vivet, Y. Thonnart, R. Lemaire, C. Santos, E. Beigné, C. Bernard, F. Darve, D. Lattard, I. Miro-Panadès, D. Dutoit, F. Clermidy, S. Cheramy, A. Sheibanyrad, F. Pétrot, E. Flamand, J. Michailos, A. Arriordaz, L. Wang, and J. Schloeffel, "A 4×4×2 Homogeneous Scalable 3D Network-on-Chip Circuit With 326 MFlit/s 0.66 pJ/b Robust and Fault Tolerant Asynchronous 3D Links," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 33–49, Jan. 2017.

[9] G. Rinaldi and M. T. Moreira, "Chronos Latency - Pole Position Performance," Aug. 2019. [Online]. Available: https://img1.wsimg.com/blobby/go/8669fba0-bdc4-47c3-a1ad-28738c4bae81/downloads/Chronos%20Latency%20WP_Rev1_2.pdf?ver=1605052602867

[10] J. Sparsø, *Introduction to Asynchronous Circuit Design*. Independently published, 2020. [Online]. Available: https://orbit.dtu.dk/en/publications/introduction-to-asynchronous-circuit-design

[11] A. J. Martin, "The Limitations to Delay-Insensitivity in Asynchronous Circuits," in *6th MIT Conference in Advanced Research in VLSI (AUSCRYPT)*, Mar. 1990, pp. 263–278.

[12] M. L. L. Sartori, M. T. Moreira, and N. L. V. Calazans, "ASCEnD-FreePDK45 - A Free Standard Cell Library for SDDS-NCL Circuits," June 2020. [Online]. Available: https://github.com/marlls1989/ascend-freepdk45

[13] K. M. Fant, *Logically Determined Design: Clockless System Design with NULL Convention Logic*. Wiley, 2005. [Online]. Available: https://books.google.com.br/books?id=igVTAAAAMAAJ

[14] P. B. McGee, M. Y. Agyekum, M. A. Mohamed, and S. M. Nowick, "A Level-Encoded Transition Signaling Protocol for High-Throughput Asynchronous Global Communication," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, Apr. 2008, pp. 116–127.

[15] J. Pontes, P. Vivet, and Y. Thonnart, "Two-Phase Protocol Converters for 3D Asynchronous 1-of-N Data Links," in *Asia-Pacific Design Automation Conference (ASP-DAC)*, May 2015, pp. 154–159.

[16] H. K. O. Berge, A. Hasanbegović, and S. Aunet, "Muller C-elements based on Minority-3 Functions for Ultra Low Voltage Supplies," in *IEEE Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, Apr. 2011, pp. 195–200.

[17] M. L. L. Sartori, R. N. Wuerdig, M. T. Moreira, S. Bampi, and N. L. V. Calazans, "Leveraging QDI Robustness to Simplify the Design of IoT Circuits," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Oct. 2020, pp. 1–4.

[18] 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 *Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2012, pp. 1–6.