Tradeoffs Between RTO and RTZ in WCHB QDI Asynchronous Design

Matheus T. Moreira[1], Julian J. H. Pontes[2], Ney L. V. Calazans[1]

GAPH – FACIN – PUCRS[1]
Porto Alegre – Brazil
matheus.moreira@acad.pucrs.br, ney.calazans@pucrs.br

CEA – LETI[2]
Grenoble – France
julian.hilgembergpontes@cea.fr

Abstract — **Classically, quasi-delay-insensitive asynchronous circuits based on weak-conditioned half-buffer employ the return-to-zero, 4-phase handshake protocol. This work scrutinizes the alternative return-to-one protocol and analyzes the effects of using it in practical circuits. A pipelined shift and add multiplier serves as case study. Return-to-one and return-to-zero versions of the circuit provide ground for extensive comparison. Experimental results point to reductions in static power and in forward propagation delay of up to 35% and 12%, respectively, when using return-to-one. Also, results indicate that mixing return-to-zero and return-to-one leads to dynamic power savings.**

Keywords— return-to-one, handshake, quasi-delay-insensitive, weak-conditioned half-buffer, low power, leakage reduction.

I. Introduction and Related Work

The evolution of silicon technologies needs new approaches to cope with power problems that are increasingly constraining synchronous design., asynchronous circuits become thus relevant for an increasing number of applications. The quasi-delay-insensitive (QDI) design style [1] is attractive to asynchronous circuits, especially because it allows wire and gate delays to be ignored, given that isochronic fork [2] delay assumptions are respected. This reduces design complexity and eases timing closure and analysis [3]. Defining a QDI template requires choosing a handshake protocol and a delay-insensitive (DI) code to represent data. The most adopted protocol is the 4-phase, because it allows reducing design complexity when compared to 2-phase [4]. Also, there are many ways to encode data in a DI manner and, even though new codes are often suggested, the 1-of-n class is widespread in asynchronous VLSI design [4]. A key factor for the success of these codes is the fact that they obviate data validity tests and completion circuits require little hardware when compared to other codes.

In 1-of-n codes, data is represented using **n** wires. Data validity is identified when exactly one of **n** wires is at a given logic value and data absence can be marked by any of the $\mathbf{2^n-n}$ other code words. The value that indicates absence of data is called *spacer*, as it always separates two successive 1-of-n codes in a data channel. Spacers are classically signaled by setting all wires of a channel to 0 (all-0s) and valid data by setting a single wire to 1, defining the *return-to-zero* (RTZ) 4-phase protocol. RTZ is well accepted in research community, and alternative manners for representing spacers received little attention so far. An alternative is the return-to-one (RTO) 4-phase protocol, where spacers are encoded by all wires at 1 (all-1s) and valid data by a single wire at 0 [5].

Using alternative representations for spacers is not a novelty itself, but related works that mention the use of all-1s spacers do not employ these to define an RTO-based protocol. Sokolov et al. [6] [7] proposed *alternating spacers*, a technique that uses all-0s and all-1s spacers successively in computations. They showed their technique is adequate to build secure cryptographic processors, but it incurs in large area overhead when compared to the usual RTZ. This comes from the fact that spacers are temporally distributed in the circuit. Thus, every latch or combinational block has to deal with both types of spacers and alternate codes, which increases complexity and limits usage. Cilio et al. [8] also proposed a similar protocol, where each data value is between two different spacers: all-0s and all-1s. They claim the single spacer scheme falls short in balancing switching activity between rails, increasing vulnerability to side-channel attacks (based on e.g. power and electromagnetic emissions). Albeit the scheme improves robustness, the drawback is again increased area and power. Murphy and Yakovlev [9] presented measurements in a prototype AES cryptographic core, which employs the Sokolov et al. alternating spacers and Moore et al. [10] used the all-1s encoding as an alarm state to reach balanced implementations. For both works, results are high robustness to attacks at high area costs.

There is no specific work proposing the exploration of a protocol based only in the all-1s spacer and its delay and power tradeoffs. A first work defining an RTO protocol for exploring design improvement opportunities, such as leakage power reduction, appeared in [5]. After that, other works [11] [12] evaluated the advantages of using this protocol in two different styles for QDI combinational logic design, namely Delay-Insensitive Minterm Synthesis (DIMS) [11] and Null-Convention Logic (NCL) [12] [13], respectively. These works evaluated only simple blocks of combinatory logic and not the impact of using RTO in the design of sequential and control blocks for QDI circuits. The main contribution of the present article is to evaluate the tradeoffs associated to using RTO or RTZ in complete QDI circuits, discussing general system level effects and specific isolated effects in sequential, control and logic or arithmetic blocks. The discussion emerges from the design of a shift and add multiplier case study, based on Weak-Conditioned Half-Buffers (WCHBs) [14] modeled with Production Rules (PRs) [2] and synthesized targeting a 65nm CMOS technology in both, RTO and RTZ. Circuits use the pseudo-synchronous automated asynchronous design flow proposed by Thonnart et al. [15]. Post synthesis timing simulation and power analysis enable investigating a series of RTO and RTZ tradeoffs.

II. The Return-to-One protocol

Classically, the RTZ 4-phase protocol is used in 1-of-n DI codes, where **n** zeroes represent a spacer and valid code words are those with a single 1. Figure 1(a) shows the RTZ 1-of-2 code, which uses two wires, called D.1 and D.0, to carry a single bit of information. A '0' bit is denoted by D.0 at 1, and a '1' bit by D.1 at 1. In 1-of-n RTZ conventions, any code word with more than a wire at 1 represents no valid data.

Figure 2(a) shows data transmission in a system using the RTZ protocol. Communication starts with all wires at 0 (all-0s). Next, the sender puts data in the channel (D.0, D.1) which is acknowledged by the receiver with the ack signal. After the sender receives ack, it produces a spacer to end communication. The receiver then lowers the ack signal, after which another communication can take place.

The RTO 4-phase protocol [5] is similar to RTZ. One difference is that valid data values are reversed compared to RTZ. Figure 1(b) shows conventions for a 1-of-2 code based on RTO. Spacers are represented by **n** wires at 1 (all-1s). A '1' bit is denoted by D.1 at 0 and a '0' bit by D.0 at 0. As Figure 2(b) shows, differently from RTZ, RTO data transmission starts after the all-1s value is in the data channel. As soon as the sender puts valid data in channel (D.0, D.1) the receiver may acknowledge it, by lowering the ack signal. Next, all data wires must return to 1 to produce a spacer. When the spacer is detected by the receiver, it raises the ack signal and new data can follow. The reason for the inverted ack is to reduce transistor count in RTO systems. Also, RTO-RTZ domain interfaces for a same code requires only **n** inverters. As a generalization, an RTO D.x wire logical value can be translated from RTZ by Eq. (1).

$$\forall x, 0 \le x \le n-1 : RTO(D.x) = \neg RTZ(D.x) \qquad (1)$$

Here, expressions RTO(D.x) and RTZ(D.x) correspond to wire logic values in the RTO and RTZ domains, respectively. In this way, according to Martin [2], the conversion of data from one domain to another is DI. Throughout this work 1-of-2 codes will be employed to demonstrate the use of RTO. However, all presented techniques can be adjusted easily to any 1-of-n code.

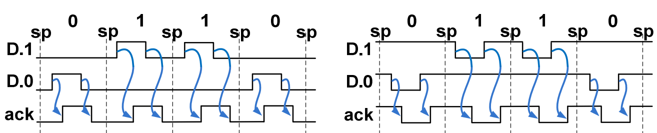
Wire Name	Spacer	Bit '0'	Bit '1'
D.1	0	0	1
D.0	0	1	0

(a)

Wire Name	Spacer	Bit '0'	Bit '1'
D.1	1	1	0
D.0	1	0	1

(b)

Figure 1 – 4-phase 1-of-2 data encoding for (a) RTZ and (b) RTO protocols.

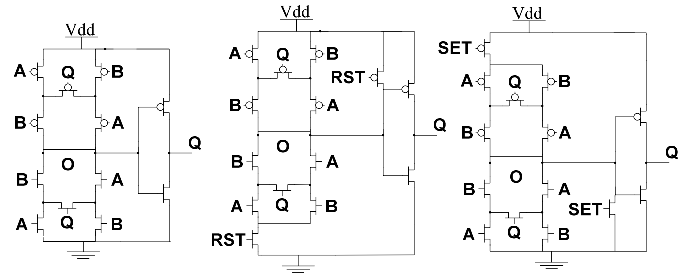


(a) (b)

Figure 2 – Example of 4-phase (a) RTZ and (b) RTO 1-of-2 data transmission, where **sp** stands for spacers.

III. RTZ AND RTO STANDARD-CELL SYNTHESIS

Several works propose standard-cell based design of RTZ QDI circuits. This usually requires only C-elements other than conventional gates. C-elements are basic components in asynchronous circuits used for event synchronization [13]. The output of a basic C-element will only switch to 1 when all inputs are at 1. Similarly, it will only switch to 0 if all inputs are at 0. For any other input combinations, the output keeps its previous value. As examples, references [16] and [17] propose techniques for designing such circuits. Throughout this work we assume the use of the van Berkel C-Element due to its better power and delay tradeoffs [17]. Figure 3(a) presents its CMOS schematic. Also, QDI circuits can be implemented using different templates and buffer types. Some common templates are: WCHB, pre-charged half buffer (PCHB) [13] and pre-charged full buffer (PCFB) [13]. This work assumes the use of WCHB template latches due to their wide adoption and the availability of an automated synthesis flow based on them [15]. This template originally requires resettable C-elements as storage cells. Figure 3(b) presents the schematic of an active-low resettable C-Element.



(a) (b) (c)

Figure 3 – Transistor topologies for van Berkel C-elements: (a) basic; (b) active-low resettable; (c) active-high settable.

According to Martin in [2], PRs are useful to define the behavior of each output of a QDI circuit using conjunction, disjunction and negation operators. Assuming that D are the inputs, Q are the outputs, req is the active-high request signal and rst is the active-low reset signal, the PRs for a j-bit RTZ 1-of-2 WCHB appear in Figure 4(a). The first two PRs correspond to the second of fourth phases of the handshake, and the last two PRs refer to the fourth phase and reset. These can be mapped to 2j active-low resettable C-elements (Figure 5(a)).

$$\begin{aligned}
&\forall i, 0 \le i \le j-1 : \\
&rst \wedge D_i.0 \wedge req \rightarrow Q_i.0 \uparrow \\
&rst \wedge D_i.1 \wedge req \rightarrow Q_i.1 \uparrow \\
&\neg rst \vee (\neg D_i.0 \wedge \neg req) \rightarrow Q_i.0 \downarrow \\
&\neg rst \vee (\neg D_i.1 \wedge \neg req) \rightarrow Q_i.1 \downarrow .
\end{aligned}$$

(a)

$$\begin{aligned}
&\forall i, 0 \le i \le j-1 : \\
&D_i.0 \vee D_i.1 \rightarrow V_i \downarrow \\
&\neg D_i.0 \wedge \neg D_i.1 \rightarrow V_i \uparrow \\
&ALL0(V) \rightarrow valid \uparrow \\
&ALL1(V) \rightarrow valid \downarrow .
\end{aligned}$$

(b)

Figure 4 –PRs for a j-bit RTZ 1-of-2 WCHB: (a) latch and (b) validity detector. Predicates ALL0 and ALL1 are true when the vector passed as argument is a binary vector V composed by respectively only 0s/only 1s.

Similarly, assume that the j-bit V signal is the vector of individual validity bits, D are the latch inputs, and valid is the active-low global validity output signal. Validity of a j-bit RTZ data channel D encoded in a 1-of-2 WCHB is then given by Figure 4(b). Predicates ALL1(x) and ALL0(x) are used to determine whether all wires of x are respectively at 1 or 0. These PRs implement the first and third phases of the protocol when driving the previous latch req input. Each data wire pair requires a 2-input NOR gate to compute its validity (the V_i's). Note that the generation of the intermediate V_i's uses NOR gates rather than ORs. If all intermediate values are at 0, data D is valid. This can be computed with a j-input NOR gate. If all intermediate values are at 1, a spacer is detected. This function can be computed with a j-input NAND gate. Conditions where some of the intermediate values are at 1 while others are at 0 cannot determine a new value for valid. Hence, outputs of the j-input NAND and NOR gates need to be synchronized with a 2-input C-element.

The resulting validity detector (VD) circuit appears in Figure 5(b). Another approach to synchronize all intermediate signals is to use a tree of C-elements. However, experience showed that these are usually more expensive in terms of silicon area, power and delay than using j-input NORs and NANDs. The latter gates can be built as trees of cells with less inputs, due to the associative nature of the non-inverted equivalent functions. Also, partial VDs can be implemented with 2-input NOR gates inside each single value WCHB and synchronized at higher hierarchical levels. This can be useful, depending on the circuit data dependencies. This work assumes that VDs are separated from WCHBs, to analyze the effects of RTO and RTZ individually in each block.

The RTO implementation of a WCHB is similar to the RTZ one. The main difference is the all-1s spacer. Because its initial state must be a spacer, there is a slight modification in the PRs for this WCHB, active-high set signal replaces the active-low rst signal of the RTZ WCHB PRs. The PRs for a j-bit RTO 1-of-2 WCHB are then given by Figure 6(a). As Figure 5(c) shows, resettable C-elements of Figure 5(a) are replaced by settable ones, enabling RTO spacers for initialization. The schematic of these C-Elements appears in Figure 3(c).

The RTO VD block is also similar to the RTZ one. The difference is that it must detect 0s rather than 1s for data validity. Resulting PRs are in Figure 6(b). When mapped to logic gates, the result is that instead of employing a NOR gate for each pair of wires to detect whether one of them is at 0, a NAND gate is used to verify if data is valid (if one of the wires is at 0). The resulting circuit appears in Figure 5(d). Similarly to what happens in RTZ, each intermediate signal is synchronized with j-input NAND and NOR gates and a 2-input C-element.

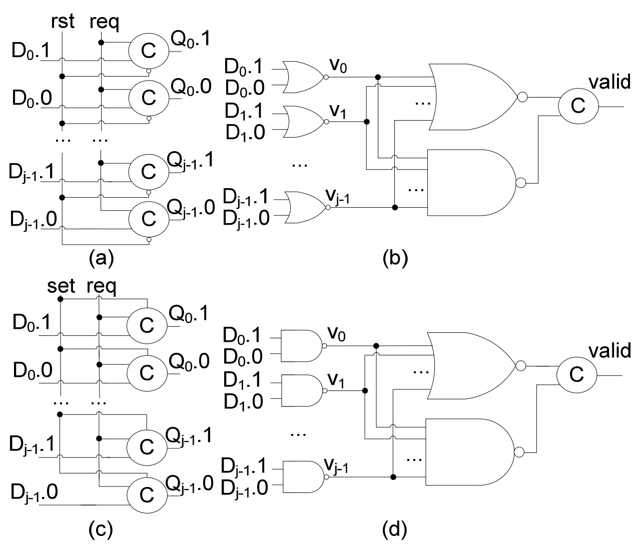


Figure 5 – WCHB latch and validity detector implementations: (a) RTZ buffer using resettable, active-low C-elements (lower input); (b) RTZ validity detector; (c) RTO latch using settable, active-high C-elements (upper input);
(d) RTO validity detector.

$$\begin{array}{ll}
\forall i, 0 \le i \le j-1: & \forall i, 0 \le i \le j-1: \\
\neg set \wedge \neg D_i.0 \wedge \neg req \rightarrow Q_i.0\downarrow & \neg D_i.0 \vee \neg D_i.1 \rightarrow v_i\downarrow \\
\neg set \wedge \neg D_i.1 \wedge \neg req \rightarrow Q_i.1\downarrow & D_i.0 \wedge D_i.1 \rightarrow v_i\uparrow \\
set \vee (D_i.0 \wedge req) \rightarrow Q_i.0\uparrow & ALL0(V) \rightarrow valid\uparrow \\
set \vee (D_i.1 \wedge req) \rightarrow Q_i.1\uparrow. & ALL1(V) \rightarrow valid\downarrow. \\
\text{(a)} & \text{(b)}
\end{array}$$

Figure 6 – PRs for a j-bit RTO 1-of-2 WCHB (a) and validity detector (b).

As for Boolean and arithmetic operations, the DIMS logic style [13] is classically employed as it allows the use of standard-cell based design without losing the DI property. For RTO, we refer to the equivalent logic style as DIMxS, standing for Delay-Insensitive Maxterm Synthesis, since RTO is based on the maxterm-based logic expressions. DIMS and DIMxS require only C-elements other than conventional standard cells. In fact, most works report the use of DIMS for implementing QDI Boolean and arithmetic blocks. Table 1 shows the truth table of an RTZ 1-of-2 half-adder with inputs A_{in} and B_{in} and outputs C_{out} and S_{out}.

Table 1 – Truth table for an RTZ 1-of-2 half-adder.

$A_{in}.1$	$A_{in}.0$	$B_{in}.1$	$B_{in}.0$	$C_{out}.1$	$C_{out}.0$	$S_{out}.1$	$S_{out}.0$
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
1	0	0	1	0	1	1	0
1	0	1	0	1	0	0	1

The first line of the table occurs when both inputs are spacers, which sets the outputs to spacers as well. Whenever one of the inputs is a spacer the outputs do not change. For that reason, such states are omitted. PRs describing functions for $C_{out}.0$, $C_{out}.1$, $S_{out}.0$ and $S_{out}.1$ appear in Figure 7.

$$\begin{array}{l}
(A_{in}.1 \wedge B_{in}.1) \rightarrow C_{out}.1\uparrow \\
(\neg A_{in}.1 \wedge \neg B_{in}.1) \rightarrow C_{out}.1\downarrow \\
(A_{in}.0 \wedge B_{in}.0) \vee (A_{in}.0 \wedge B_{in}.1) \vee (A_{in}.1 \wedge B_{in}.0) \rightarrow C_{out}.0\uparrow \\
(\neg A_{in}.0 \wedge \neg B_{in}.0) \wedge (\neg A_{in}.0 \wedge \neg B_{in}.1) \wedge (\neg A_{in}.1 \wedge \neg B_{in}.0) \rightarrow C_{out}.0\downarrow \\
(A_{in}.0 \wedge B_{in}.1) \vee (A_{in}.1 \wedge B_{in}.0) \rightarrow S_{out}.1\uparrow \\
(\neg A_{in}.0 \wedge \neg B_{in}.1) \wedge (\neg A_{in}.1 \wedge \neg B_{in}.0) \rightarrow S_{out}.1\downarrow \\
(A_{in}.1 \wedge B_{in}.1) \vee (A_{in}.0 \wedge B_{in}.0) \rightarrow S_{out}.0\downarrow \\
(\neg A_{in}.1 \wedge \neg B_{in}.1) \wedge (\neg A_{in}.0 \wedge \neg B_{in}.0) \rightarrow S_{out}.0\downarrow.
\end{array}$$

Figure 7 – PRs for an RTZ, 1-of-2 half-adder.

When mapping these to DIMS-based components, conjunctions of the pairs inside parenthesis are mapped to 2-input C-elements, to guarantee synchronization of both valid data and spacers. The disjunctions outside parenthesis in the PRs filter the C-elements that have their outputs set to 1. This models rising output transitions. The conjunctions outside parenthesis for falling-transition PRs synchronize spacers in all C-elements. These functions are mapped directly to OR gates. The equivalent gate level schematic appears in Figure 8 (a).

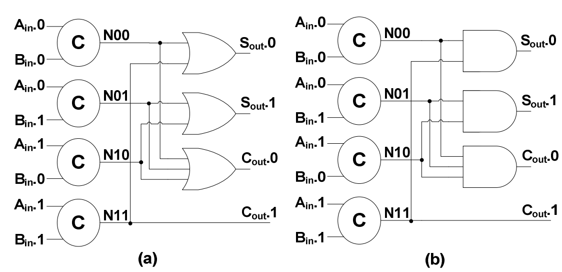


Figure 8 –DIMS/DIMxS implementation of (a) RTZ and (b) RTO half-adders.

The RTO implementation is similar. Table 2 presents the truth table of an equivalent RTO version of the 1-of-2 DIMxS half-adder, which derives from Table 1 by Eq. (1).

Table 2 – Truth table for an RTO 1-of-2 half-adder.

$A_{in}.1$	$A_{in}.0$	$B_{in}.1$	$B_{in}.0$	$C_{out}.1$	$C_{out}.0$	$S_{out}.1$	$S_{out}.0$
1	1	1	1	1	1	1	1
1	0	1	0	1	0	1	0
1	0	0	1	1	0	0	1
0	1	1	0	1	0	0	1
0	1	0	1	0	1	1	0

The equivalent PRs for outputs $C_{out}.0$, $C_{out}.1$, $S_{out}.0$ and $S_{out}.1$ for this implementation of the DIMS half-adder appear in Figure 9.

$$(A_{in}.1 \wedge B_{in}.1) \rightarrow C_{out}.1 \uparrow$$
$$(\neg A_{in}.1 \wedge \neg B_{in}.1) \rightarrow C_{out}.1 \downarrow$$
$$(A_{in}.0 \wedge B_{in}.0) \wedge (A_{in}.0 \wedge B_{in}.1) \wedge (A_{in}.1 \wedge B_{in}.0) \rightarrow C_{out}.0 \uparrow$$
$$(\neg A_{in}.0 \wedge \neg B_{in}.0) \vee (\neg A_{in}.0 \wedge \neg B_{in}.1) \vee (\neg A_{in}.1 \wedge \neg B_{in}.0) \rightarrow C_{out}.0 \downarrow$$
$$(A_{in}.0 \wedge B_{in}.1) \wedge (A_{in}.1 \wedge B_{in}.0) \rightarrow S_{out}.1 \uparrow$$
$$(\neg A_{in}.0 \wedge \neg B_{in}.1) \vee (\neg A_{in}.1 \wedge \neg B_{in}.0) \rightarrow S_{out}.1 \downarrow$$
$$(A_{in}.1 \wedge B_{in}.1) \wedge (A_{in}.0 \wedge B_{in}.0) \rightarrow S_{out}.0 \downarrow$$
$$(\neg A_{in}.1 \wedge \neg B_{in}.1) \vee (\neg A_{in}.0 \wedge \neg B_{in}.0) \rightarrow S_{out}.0. \downarrow$$

Figure 9 – PRs for an RTO, 1-of-2 half-adder.

These PRs are similar to those of the RTZ protocol. Indeed, the disjunctions inside parenthesis to compute minterms are the same. The difference is that in RTO valid data is given by 0s. Therefore, instead of computing minterms, the maxterms are the ones that need to be computed. Thus, disjunctions are used for falling transitions in PRs and conjunctions indicating when all maxterms are at 1 detect spacers. Therefore, ORs used in the RTZ classic DIMS are replaced by ANDs in the RTO version. Figure 8(b) shows the associated RTO gate level schematic. Using this approach, any RTO DIMxS logic block can be implemented. Classically, AND/NAND gates are preferred over OR and NOR gates in VLSI design. A stack of NMOS transistors is present in these gates, while ORs and NORs employ a stack of PMOS transistors. Due to the fact that electron mobility is normally three times that of holes, NAND/AND gates are expected to present better power and delay tradeoffs than NOR/OR gates for the same silicon area. As a consequence, RTO DIMxS circuits are expected to present a better power-delay tradeoff than equivalent RTZ circuits. Additionally, the use of mixed systems combining RTO and RTZ for communicating blocks can be implemented by just inverting outputs, which can lead to further optimization degrees.

IV. A Pipelined Multiplier Case Study

To analyze the tradeoffs between RTO and RTZ, a pipelined multiplier was implemented using 1-of-2 data encoding in the WCHB template based on [15]. This circuit multiplies two unsigned integer n-bit values A and B, using a shift and add algorithm. Figure 10 shows the block diagram that maps the algorithm to handshake components. All dotted lines represent single wire control signals and full lines represent 1-of-2 data channels. This circuit computes the product of A_{in} and B_{in}, and writes the result in the (2*n)-bit output Q_{out}. The algorithm was pipelined and the shift and add iteration is unrolled into **n+2** stages. The case study enables the separate analysis of RTO and RTZ protocols effects in registers, validity detectors, control circuits, arithmetic blocks, and yet allows overall circuit analysis. Registers are called REG_i, where **i** is its number in the pipeline.

All registers employ the WCHB template. Inputs are initially registered in $\mathbf{REG_0}$, with size **2n**, and a zero generator is employed to initialize the value of variable **x**. This generator is represented by the box labeled **0**, before $\mathbf{REG_1}$. The validity of the registered data is given by the first validity detector (**VD0**). Also, before the next register ($\mathbf{REG_1}$), the registered input $\mathbf{B_0}$ has its size **n** extended to **2n** through the **EXT** operator, to allow shift left operations without any bit loss. Each of the next **n** pipeline stages comprises a register $\mathbf{REG_i}$, with size **5n**, a validity detector $\mathbf{VD_i}$ and a shift and add step $\mathbf{STEP_i}$, $\mathbf{1 \leq i \leq n}$. Finally, stage **n+1** has its output registered in $\mathbf{REG_{n+1}}$, which produces the circuit output $\mathbf{Q_{out}}$ and has also size **5n**. Additionally, all Boolean and arithmetic logic blocks were implemented based on DIMS/DIMxS.

For comparison sake, RTZ- and RTO-based versions of this circuit were described in structural VHDL by interconnecting handshake components of an in-house macro blocks library. These blocks were initially modeled using PRs, as in Section II and then the required C-elements were manually mapped to the description. Boolean logic, such as the AND/OR operations required by the blocks were extracted from PRs and mapped in the description, but the synthesis tool was able to optimize their logic using basic logic gates. Also, the macro library had only RTZ components. RTO versions were added by designing new blocks according to the techniques discussed in Section II.

The STMicroelectronics CMOS 65nm technology was used for synthesis. Typical gates (such as ANDs, ORs and buffers) were mapped in the Corelib provided by the vendor and C-elements were mapped to the ASCEnD-ST65 library [18] [19]. Design optimization of the circuit was enabled by the adoption of the flow proposed in [15] coupled to industrial tools. The specific tools employed were the RTL Compiler and Encounter from Cadence. Thirteen distinct implementations of each RTO and RTZ designs were produced, each with a different maximum operating speed. Different speeds are obtained in the classical way, using gates with distinct drive strengths. The generated designs had their internal nets and gate delays annotated in a standard delay format (SDF) file, which was the source to simulate the mapped netlist. Also, similar speed RTO and RTZ circuits presented equivalent silicon area, which suggests that none of the protocols can be classified as more area efficient.

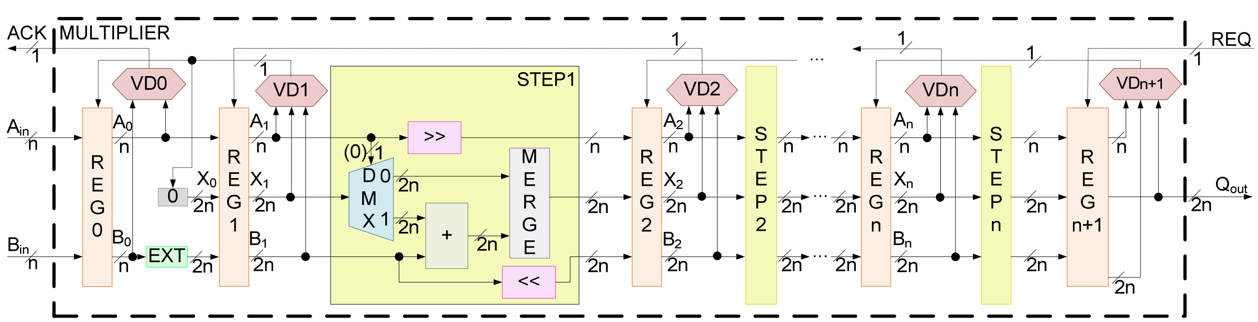


Figure 10 – Block diagram of the study case pipelined multiplier. Dotted lines are single wire control signals and full lines 1-of-2 encoded data channels.

The simulation scenario employed a producer of random data and a consumer, both described in SystemC. These were used to verify correct operation of the circuits and to measure delay values. Delay measurements considered the designs´ throughput, measured in millions of operations per second (MOPS), and average acknowledge delays, measured for each design by simulating it with random data on the inputs, during 1 ms. Internal activity of the nets was annotated and exported for power analysis.

Simulation results show that RTO fastest versions were able to compute 236 MOPS and the slowest versions 148, while for RTZ these values were 237 and 176, respectively. Figure 11 presents the average acknowledge delays for valid data and spacers for each design (note that only the intersection between these intervals is shown in the graphs). The measured values are very similar. Yet, for slow designs, under 200 MOPS, the RTO protocol presents slightly faster responses for valid data and slower responses for spacers. For the other designs, this scenario is reversed. Still, the speed-up in responses for valid data and spacers is always under 3%. Thus, neither protocol can be said to be more efficient in absolute terms for acknowledge speed.

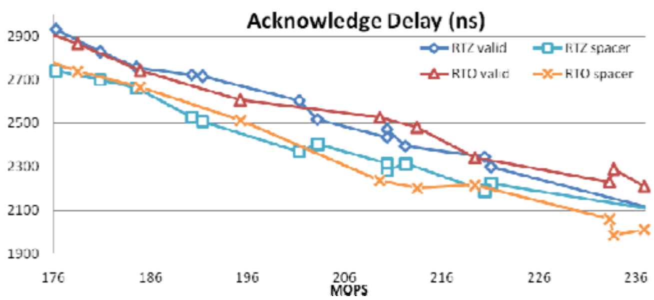


Figure 11 – Acknowledge delay of valid data and spacer for RTO and RTZ.

A similar simulation scenario was employed to measure the forward propagation delay with the same producer and consumer. However, the producer generates a single pair of random data and the consumer measures how long it takes to compute data and to generate a valid output, without considering the propagation of spacers that would eventually follow it. Then, the circuit is reset and another pair of data is sent. In this way, inter-register acknowledge does not interfere while measuring the forward propagation delay. This process simulates for 1 ms. Figure 12 presents the measured average values.

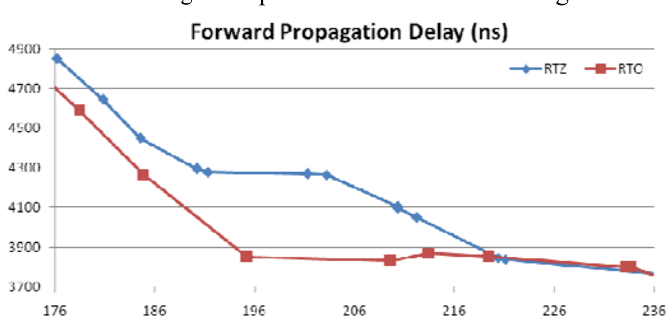


Figure 12 – Forward propagation delay results for the simulated designs.

Results show that the RTO designs present lower forward propagation delay in most cases. In the best case, this difference was around 12%. In the fastest designs, RTZ forward propagation delay equals that of RTO. In fact, for the fastest design, the RTZ version presents slightly lower forward propagation delay. Still, this result displays a clear advantage of using RTO, as forward propagation delay is a very important characteristic of handshake-based circuits. Typically, given a sender/receiver scenario, the smaller the percentage of forward propagation delay of the sender in total communication time, the faster the receiver will acknowledge. This leads to scenarios where the sender is released faster and may start a new communication earlier, which leads to overall speed up.

The activity of all internal nets was annotated and used as the source for five power analysis scenarios: an idle state, where the circuit was reset and was not fed with any data, and four dynamic scenarios, where the circuit was fed with data during 25%, 50%, 75% and 100% of the simulation time. Figure 13 presents the measured total power for idle states. The obtained results display a clear advantage of the RTO protocol concerning idle power, 25% in average and 35% in the best case. This is important for asynchronous circuits, since there is no global synchronization and while some parts of the system are operating, others may be quiescent. Albeit it was expected a close to linear growth of power in the presented charts, these are slightly distorted. This is because the used synthesis method employs tools that are designed for synchronous systems that end up optimizing the critical path delay of the circuit rather than its average delay. This explains the peaks and valleys in the power charts. Nevertheless, the qualitative aspect of the results is not jeopardized. If tools adequate for asynchronous synthesis were available, these peaks and valleys would not appear in the charts, but the correlation between the latter would remain. Experiments from [5] support this argument.

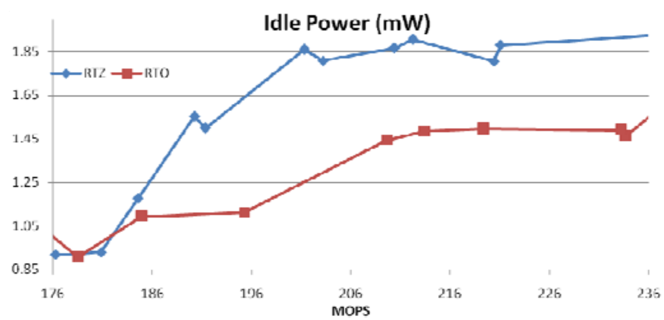
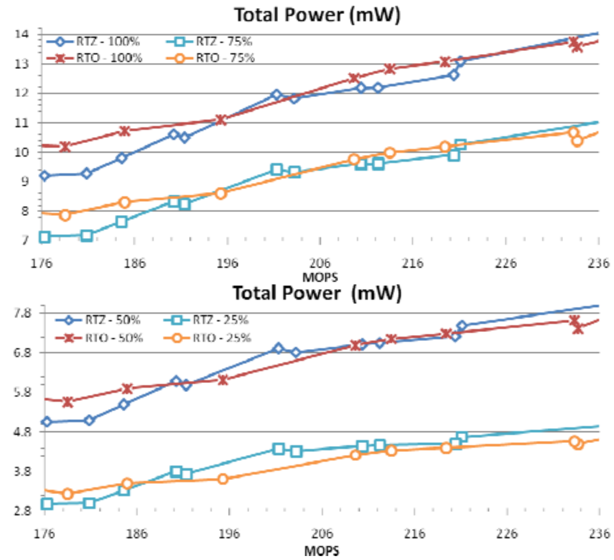


Figure 13 – Total power of the designs while at reset/set and idle states.

As for the dynamic scenarios, RTO- and RTZ-based designs present similar total power. Figure 14 presents the measured total power of the circuits for such scenarios. Figure 14(a) presents the charts for 100% and 75% operating times and Figure 14(b) does the same for 50% and 25% values. Typically, RTO and RTZ charts of a same scenario are intertwined with one another. In general, RTO presents higher total power for slower designs, (8% higher in worst case). For faster designs the situation is reversed, and in the best case RTO presents reductions of over 10%. In this way, both protocols are assumed to have similar power efficiency for dynamic scenarios. In this context, at circuit level, the advantage of RTO-based systems is the reduced power in idle states. In other words, a reduction on leakage power is observed.



(a)

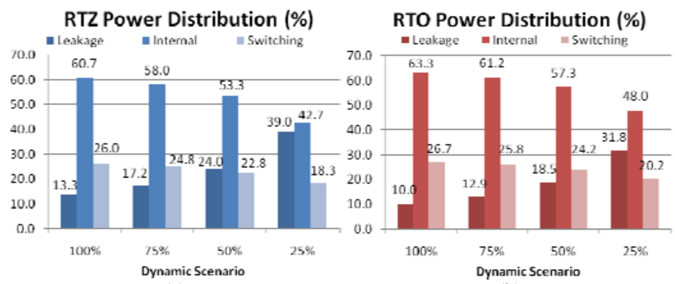
(b)

Figure 14 – Total power of the RTO and RTZ designs when: (a) computing 100% and 75% of the time and (b) computing 50% and 25% of the time.

For the case study, effects of this characteristic appear in the charts of Figure 14(b). The lower the operating time percentage is, the lower the RTO chart values become, compared to RTZ values. This characteristic of RTO can be understood by analyzing the power distribution of the simulated circuits as Figure 15 shows. Figure 15(a) and Figure 15(b) present the average power distribution of the simulated RTZ- and RTO-based circuits, respectively, for the four dynamic scenarios. In the presented charts, power is distributed in leakage, internal and dynamic power. The first is due to subthreshold leakage currents, which are a major concern for recent technology nodes [20]. Internal power is due to parasitics and short-circuit currents when switching standard-cells inputs. Switching power comes from charge and discharge of capacitances when switching standard-cells outputs. The charts show leakage power represents a smaller portion of the total power in RTO, when compared to RTZ. These results point to good potential for RTO to cope with leakage power constraints in future technologies. Measured power for dynamic scenarios did not meet expectations at the system level. With the use of AND/NAND gates, rather than OR/NOR gates, and results presented in [5], a better power efficiency was expected for RTO in the dynamic scenarios as well, i.e. RTO-based designs were expected to provide the same throughput as RTZ designs with lower power.

A more precise circuit power analysis reveals that the total power of the pipeline is the sum of the power from three block types (Figure 10): VDs, REGs and STEPs power. It is possible to measure the isolated effects of RTO and RTZ in logic arithmetic blocks (steps), sequential blocks and validity detectors measuring the percentage of total power accountable for each block. Figure 16 presents the obtained results of this power analysis for the idle scenario (parts (a), (b) (c)) and for dynamic scenarios with 100% operation, in (d), (e), (f). The 100% operation scenario represents the worst case dynamic scenario for RTO justifying its choice. Note that the presented values are not absolute and, rather, charts show the percentage of total power for each block of each circuit for both simulation scenarios. Also, note that REGs and VDs portion of power decreases as designs get faster, while the STEPs portion increases. This is mostly due to critical path optimizations in synthesis.

As charts in Figure 16(a),(b) and (c) show, the distribution of RTO and RTZ total power is similar in idle state. This demonstrates that reduction in leakage power, when using RTO, reflects at the circuit level. However, for dynamic state results, RTO and RTZ display significant discrepancies. For **VDs** and **STEPs** power, significant savings occur when employing RTO. This agrees with expected results from [5] and [11], since these blocks employ AND/NAND gates rather than OR/NOR gates, which is classically preferable in VLSI synthesis. Thus, the obtained results point to design optimizations when using RTO for arithmetic DIMxS-based circuits and validity detectors. On the other hand, as Figure 16(e) shows, the RTZ **REGs** blocks present a significantly lower part of the total power, when compared to RTO.



(a) (b)

Figure 15 – Average power distribution of designs: (a) RTZ and (b) RTO.

These blocks are composed exclusively of resettable or settable C-elements. The former are used in RTZ and the latter in RTO. Results suggest that, albeit settable C-elements of RTO present lower static power than resettable C-elements of RTZ, their dynamic power is bigger. In fact, an accurate analysis of the characterized timing and power models of the ASCEnD-ST65 library confirmed these conclusions. One explanation is the fact that the settable C-element employs a stack of three PMOS transistors, while in the resettable C-elements this stack uses NMOS transistors, as Figure 3 details. This discussion leads to the use of smaller transistors in resettable C-elements. The issue is independent of C-element topology. Thus, results point to power reductions when using RTZ-based WCHB registers. Case study data suggest the RTO protocol is more power efficient in dynamic scenarios for validity detectors and logic arithmetic blocks, while RTZ is more suited for buffers.

Therefore, a system that employs both protocols is expected to provide further optimizations. Differently from the works presented in [6]-[10], which propose the use of two distinct spacers that are temporally distributed, mixing RTZ and RTO as described here implies the use of two spacers spatially distributed. In this way, avoidance of the hardware complexity and area overheads observed in [6] is possible. Also, translation of RTZ signals into RTO and vice-versa is very simple (See Eq. (1)). In the worst case, it requires one inverter per wire. However, resettable C-elements of RTZ registers already have inverters in their outputs, required by their memory scheme (see Figure 3). Then, the output conversion of these blocks to RTO just requires using as output of resettable C-elements wire **O**, just before the inverter, which maintains the DI property. Also, AND gates in arithmetic RTO blocks right before an RTZ block can be changed to NAND gates, implying further optimization.

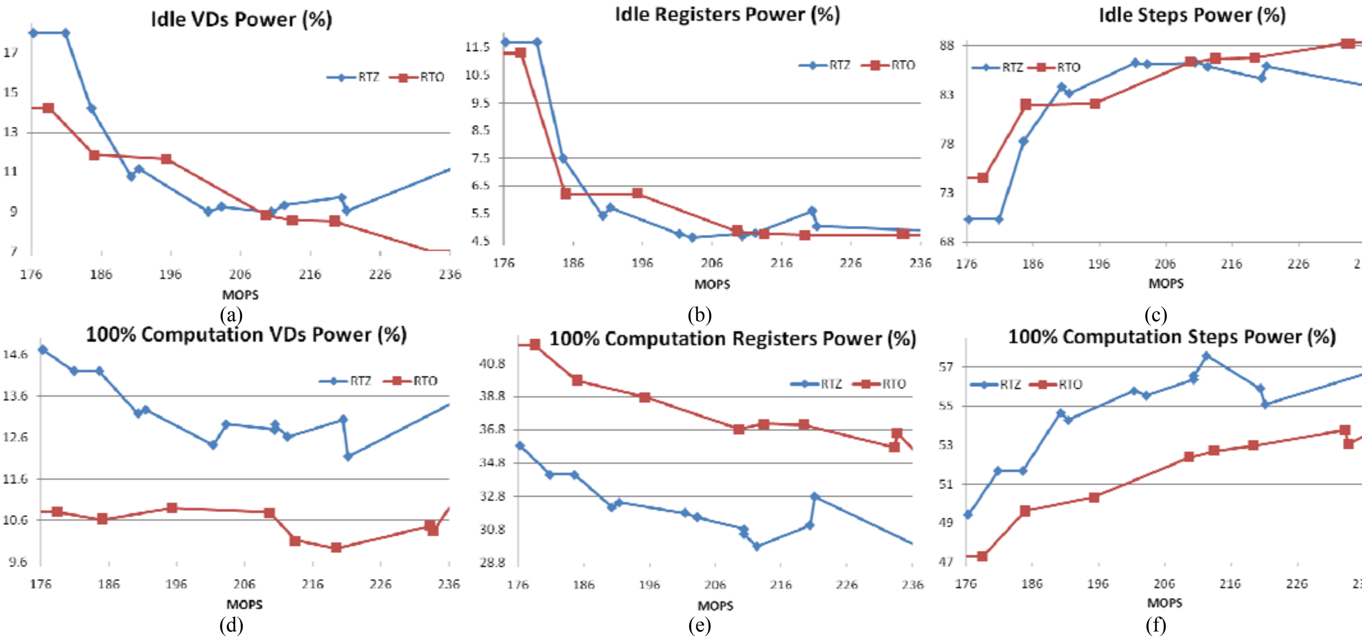


Figure 16 – Power distribution in the multiplier components for a scenario where the circuit is computing 100% of the time. VD stands for validity detector.

Such a system is expected to provide much lower total power, given that, as Figure 16(e) shows, **REGs** are responsible for a minimum of 35% (in the worst case, over 40%) of the total power of RTO-only circuits. Because the portion of idle power of **REGs** for both protocols is usually lower than 6.5% (over 10% in a few cases), as Figure 16(b) shows, the use of RTZ **REGs** is not expected to cause significant increase in circuits´ idle power. Moreover, mixed RTO/RTZ circuits can take advantage of the benefits of each protocol for each different block, leading to an enlarged design optimization space. For instance, according to Figure 16(d) chart, faster RTO **VDs** can be obtained for a same power budget of their RTZ version. These circuits acknowledge inputs faster, enabling better communication at system level, and potentially leading to overall circuit speed-up. Finally, as mentioned before, Equation (1) satisfies the conditions stated in [2] for maintaining the circuit DI property. In this way, mixing RTO and RTZ does not jeopardize the functionality of a QDI circuit.

V. Conclusions

Classically, the RTZ protocol is employed for 4-phase communication in WCHB QDI circuits. This work demonstrates that using the RTO protocol or a mix of the two leads to better choices. Results showed that RTO-based blocks present lower idle power, which is beneficial for asynchronous circuits. This can be employed in modules like networks-on-chip, which typically have low average activation rates along time. Also, DIMS/DIMxS logic blocks and validity detectors are more power efficient, in terms of idle and dynamic operation, when implemented using RTO. Future work includes the analysis of a mixed system, with some blocks implemented using RTO and others with RTZ, to provide a better understanding of protocol choice tradeoffs. To do so, a set of inverted C-elements is under design to enhance the ASCEnD-ST65 library. Also, a study evaluating other schemes for implementing set and reset signals in C-elements is under way, which may lead to optimizations in RTO registers. Another future work is an evaluation of the effects of the RTO protocol on other logic styles. Finally, analyzing the effects of the RTO protocol in additional DI codes is also an interesting future work.

Acknowledgements

This work was partially supported by the CAPES-PROSUP (under grant 11/0455-5) and FAPERGS (under grant 11/1445-0). Authors acknowledge the support of CNPq under grants 310864/2011-9 (N. Calazans) and 142079/2013-8 (M. Moreira).

References

[1] R. Manohar and A. J. Martin. Quasi-delay-insensitive circuits are turing-complete. In: International Symposium on Advanced Research in Asynchronous Circuits and Systems, 1996.

[2] A. J. Martin. The limitations to delay-insensitivity in asynchronous circuits. In: Proceedings of the sixth MIT conference on Advanced research in VLSI, 1990, pp. 263-278.

[3] W. J. Bainbridge, W. B. Toms, D. A. Edwards and S. B. Furber. Delay-insensitive, point-to-point interconnect using m-of-n codes. In: International Symposium on Advanced Research in Asynchronous Circuits and Systems, 2003, pp. 132-140.

[4] A. J. Martin and M. Nyström. Asynchronous Techniques for System-on-Chip Design. In: Proceedings of the IEEE, 94(6), June 2006, pp. 1089-1020.

[5] 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, 2012. 6p.

[6] D. Sokolov. Automated synthesis of asynchronous circuits using direct mapping for control and data paths. PhD Thesis, SEECE, Univeristy of Newcastle upon Tyne, NCL-EECE-MSD-TR-2006-111, 2006.

[7] D. Sokolov, J. Murphy, A. Bystrov and A. Yakovlev. Design and analysis of dual-rail circuits for security applications. In: IEEE Transactions on Computets, vol. 54(4), April 2005, pp. 449–460.

[8] W. Cilio, M. Linder, C. Porter, J. Di S. Smith and D. Thompson. Side-channel attack mitigation using dual-spacer dual-rail delay-insensitive logic (D3L). In: SoutheastCon, 2010, pp. 471-474.

[9] J. Murphy and A. Yakovlev. An alternating spacer AES crypto-processor. In: European Solid-State Circuits Conference, 2006, pp. 126-129.

[10] S. Moore, R. Anderson, R. Mullins, G. Taylor and J. J. A. Fournier. Balanced self-checking asynchronous logic for smart card applications. In: Microprocessors and Microsystems 27, 2003, pp. 421-430.

[11] M. Moreira, R. Guazzelli and N. L. V. Calazans. Return-to-One DIMS Logic on 4-phase m-of-n Asynchronous Circuits. In: International Conference on Electronics, Circuits and Systems, 2012, pp. 669-672.

[12] M. Moreira and N. L. V. Calazans. NCL+: Return-to-One Null Convention Logic. In: International Midwest Symposium on Circuits and Systems, 2013, 4p.

[13] K. M. Fant and S. A. Brandt. NULL convention logic: a complete and consistent logic for asynchronous digital circuit synthesis. In: International Conference on Application Specific Systems, Architectures and Processors, 1996, pp. 261-273.

[14] P. A. Beerel, R. O. Ozdag and M. Ferretti. A Designer's Guide to Asynchronous VLSI. Cambridge University Press, 2010, 337 p.

[15] Y. Thonnart, E. Beigné and P. Vivet. A Pseudo-Synchronous Implementation Flow for WCHB QDI Asynchronous Circuits. In: International Symposium on Advanced Research in Asynchronous Circuits and Systems, 2012, pp. 73-80.

[16] E. Yahya and M. Renaudin. QDI Latches Characteristics and Asynchronous Linear-Pipeline Performance Analysis. TIMA Technical Report TR 06/06-03, 2006, 11 p.

[17] M. T. Moreira, B. S. Oliveira, F. G. Moraes and N. L. V. Calazans. Impact of C-Elements in Asynchronous Circuits. In: International Symposium on Quality Electronics Design, 2012, pp. 438-444.

[18] M. T. Moreira, B. S. Oliveira, J. J. H. Pontes and N. L. V. Calazans. A 65nm Standard Cell Set and Flow Dedicated to Automated Asynchronous Circuits Design. In: International SOC Conference, 2011, pp. 99-104.

[19] M. T. Moreira, B. Oliveira, J. Pontes, F. Moraes and N. L. V. Calazans. Adapting a C-Element Design Flow for Low Power. In: International Conference on Electronics, Circuits and Systems, 2011.

[20] N. Ekekwe. Power dissipation and interconnect noise challenges in nanometer CMOS technologies. In: IEEE Potentials 29(3), 2010, pp. 26-31.