Design of an Improved and Robust Asynchronous Wrapper (AW) for FPGA Applications

Duarte L. Oliveira[1], Lester A. Faria[1] and Eduardo Lussari[1,2]

[1]Electronic Engineer Division Technological Institute of Aeronautics – ITA - SJC – São Paulo – Brazil
email: duarte@ita.br, lester@ita.br
[2]Electronic Division Mectron Engenharia S.A. - SJC – São Paulo – Brazil
email: lussari@gmail.com.br

ABSTRACT

Contemporary digital systems must be based on the "System-on-Chip – SoC" concept. An interesting style for SoC design is the GALS paradigm (Globally Asynchronous, Locally Synchronous), which can be used to implement circuits in FPGAs (Field Programmable Gate Arrays). However the implementation of asynchronous interfaces (asynchronous wrapper – AW) constitutes a major drawback for this kind of devices. Although there is a typical AW design style, which is based on asynchronous controllers providing communication between modules (called ports), port controllers are subject to essential-hazard when implemented in FPGAs. In this context, this paper proposes a new asynchronous GALS wrapper architecture, suitable for implementations in any kind of FPGAs. The proposed port controllers showed to be essential-hazard-free, not needing any special cares in implementation concerning to LUTs choice. Additional advantages of the proposed architecture are: total autonomy that synchronous modules achieve when interacting with the asynchronous wrapper; the ports can be synthesized in the direct mapping style (so without knowledge of asynchronous logic synthesis); and the ports interact with environment in Ib/Ob Mode, not needing a timing analysis. Simulation results show the applicability of the proposed architecture and lead to its practical implementations in FPGAs.

Keywords: asynchronous logic; essential hazard; FPGA; interface; finite state machine; XBM specification

I. INTRODUCTION

Contemporary digital systems must necessarily be based on the "System-on-Chip – SoC" concept. The main reason for that is the need for satisfying the ever-growing demand for higher performance, re-usability and low-power requirements [1,2]. SoC circuits are generally composed by functional modules, which can be IP-cores (intellectual property cores), which are developed by many different vendors. These IP-cores are pre-designed, verified, tested and optimized for high-performance, allowing also cost reduction and shorter development time. However, SoC circuits, when implemented using a global clock signal, are subject to speed and power penalties (clock skew, distribution networks, etc.), leading to a very difficult timing analysis [3]. The need for the implementation of SoC circuits in FPGAs, leads to an even worse clock skew problem, once delays between macro-cells can be very representative.

A natural choice for these problems is the asynchronous project methodology [3,6], which can eliminate the previously mentioned challenges by removing the clock signal from the design. But, once they are built with asynchronous modules, some drawbacks can be highlighted focusing on a trustable implementation: *a)* the lack of reliable tools for asynchronous design; *b)* difficulties found in hazard-free designing and testing; *c)* limited culture on asynchronous design; and *d)* lack of asynchronous IPs [7].

Concerning to asynchronous controllers design in FPGAs [8-9], the drawbacks become even worse, once the internal routing process between macro-cells introduce significant delays that can result in essential-hazard [6]. The more accepted solutions found in literature are related to the circuit class, but the work-around are limited to delay-element insertion, or LUT placement, both solutions presenting difficulties of implementation in commercial FPGAs.

Therefore, intermediate solutions were previously proposed between totally synchronous and totally asynchronous designs, such as the GALS methodology (*Globally Asynchronous Locally Synchronous*), consisting of many synchronous functional modules that communicate to each other in an asynchronous form. In order to handle this asynchronous communication between modules, an interface circuit is added around each synchronous module, what is called an Asynchronous Wrapper (AW). In the AW, the communication is performed by ports (asynchronous controllers), which show to be subject to inherent problems. Considering all the previous works found in literature, different Asynchronous Wrapper Ports (AW_P) focusing on FPGAs [17-23] showed some kind of essential-hazard, or did not satisfy the isochronic fork concepts.

In this context, this paper proposes an AW_P that, making use of a pausible local-clock, showed to be completely free of essential-hazard, improving all the previous works found in literature and allowing its implementation in any kind of FPGA. Additional advantages of the proposed architecture are: total autonomy that synchronous modules achieve when interacting with the asynchronous wrapper; the ports can be synthesized without any knowledge of asynchronous logic synthesis, once it is used the direct mapping style; and ports interact with environment in Ib/Ob Mode, not needing a timing analysis.

In order to achieve this goal, this paper is divided in 7 sections. While section 2 presents a theoretical background on the related topics discussed on this paper, section 3 describes the architecture of the different blocks that compose the final structure, section 4 elucidates the Mapping-Direct XBM methodology and presents, as well, the Essential-Hazard-Free specification for the ports design. Section 5 shows, step-by-step the design of the hazard-free ports; and section 6 relates to simulation results and further discussions. Finally, section 7 presents some concluding remarks and future works.

II. THEORECTICAL BACKGROUND

FPGAs have become a popular solution for implementing digital circuits because of their low-cost and short development time benefits. This technology has been growing considerably in the last years, presenting million-gates FPGAs that allow the implementation of complex digital systems [4,5].

On the other hand, the term GALS was first used by Chapiro in [10], having been successfully used in many implementations, including FPGAs [15] and ASIC (Application Specific Integrated Circuit) [13,14]. A GALS system consists of many synchronous functional modules (that may be IPs), which carries its own individual clock signals and communicate to each other in an asynchronous form. To handle the asynchronous communication between modules, an interface circuit is added around each synchronous module, what is called an Asynchronous Wrapper (AW). This term was first used in [11], by Bormann and can be built with local clocks, gated-clocks, FIFO, communication asynchronous controller (Input *Ports*, Output *Ports*), etc. Techan et al. [12] shows the different styles of asynchronous interfaces dedicated to GALS systems. In this context, Asynchronous wrappers that make use of Communication Ports (AW_P) are of main interest because they allow removing the asynchronous handshake scheme from the synchronous module, allowing the design of each one of the synchronous modules by using standard techniques of synchronous design. Figure 1 shows a generic interface with a synchronous module.

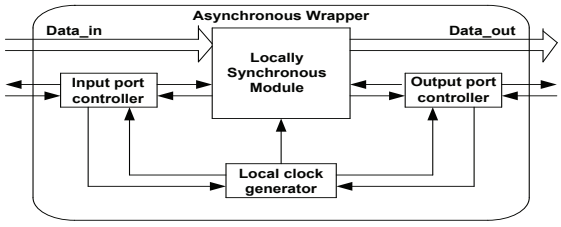

Figure 1. Asynchronous wrapper architecture, providing locally synchronous modules to communicate to others in an asynchronous way, avoiding clock skew problems in FPGA.

Although GALS is able to solve the problems related to the global clock signal, the communication between modules is already performed in the asynchronous paradigm. The communication ports are asynchronous controllers, and subject to its inherent problems. Jia [16] shows the advantages of implementing GALS in FPGA, highlighting the elimination of the clock skew problem. On the other hand, the biggest problem of GALS approach for FPGAs is the asynchronous communication, which must be performed by the controllers (ports) [17-23], showing some kind of essential-hazard or not satisfying the isochronic fork concepts. The *Speed-independent* or *quasi delay-insensitive* circuit class, equivalent to asynchronous controllers, cannot be implemented in commercial FPGAs because the lines between cells present significant delay. Therefore, applying the isochronic fork concept [6] in FPGAs is unnatural.

In order to evaluate previous trials of dealing with this problem, some works were found in literature, where different kinds of ports have been implemented in FPGA, and were synthesized in the logic synthesis style [6]. The ports proposed in [17-20] were specified in STG (Signal Transition Graph), which is a specification *Petri-net-like*, having been synthesized in the Petrify tool [6]. These ports are controllers of quasi-delay-insensitive (QDI) class, but the implementations in FPGA required a very complicated choice of LUTs in order to meet the requirement *isochronic fork* [6].

On the other hand, the ports proposed in [21-23] were specified in XBM (Extended Burst-Mode) and BM (Burst-Mode). These ports were implemented, respectively, in 3D [25] and Minimalist [26] tools, on Huffman architecture with output feedback. These ports are controllers of bounded gate and wire delay (BGWD) class, interacting with the environment in the generalized fundamental mode (GFM), leading to a high necessity of timing analysis. They are subject to essential hazard, therefore requiring the insertion of delay elements.

In order to implement the FPGA ports, Pontes et al. [22] inserted delays to meet GFM (Fig. 7 - delay-1) and used the hard macros technique to choose specific LUTs and to avoid essential hazard. In the same way, Farouk et al. [20] had to choose specific LUTs in order to meet isochronic fork restrictions [6].

As could be seen in [21], the implementation of the AW_P in FPGAs presents two main problems: firstly, the two different ports (input and output) operate at GFM, leading to the need of a timing analysis and requiring that the interaction with the environment must be slow enough to allow the stabilization of the ports; and secondly, the input port presents essential-hazard. Figure 2 shows a simulation of the input port in platform ALTERA, showing the existence of essential-hazard.

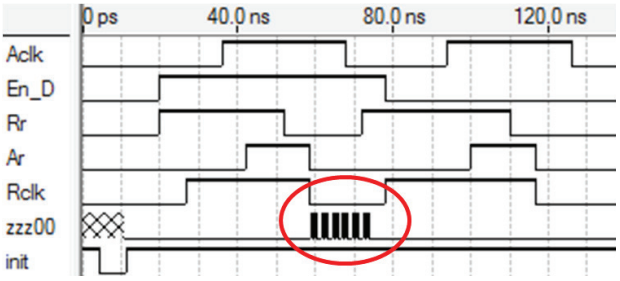

Figure 2. Simulation: input port of [21] presenting essencial hazard

In this paper we show that, through our proposal of implementation (Direct mapping - MAP_DIR_XBM) [24]), it is possible to achieve an improvement on this performance, leading not only to a more robust controller (considering technology changes), but also not needing any time analysis, what leads to FPGA implementation independent of the mapping and routing. It is also introduced the "necessary and sufficient" condition for synthesizing the hazard-free ports by MAP_DIR_XBM method.

III. ARCHITECTURE

A. Asynchronous Wrapper

The main objective of the proposed architecture is to provide a weak interface interaction between the locally synchronous module (LSM) and asynchronous interface. It is an upgrade of the architecture previously shown in [23]. Figure 3 shows the variables "data available" and "data accept" as the only ones used for the communication between the LSM and the interface [23]. It is possible to see that, when *data available* = 1, data is ready to be transmitted, and when *data accept* = 1, data will be received.

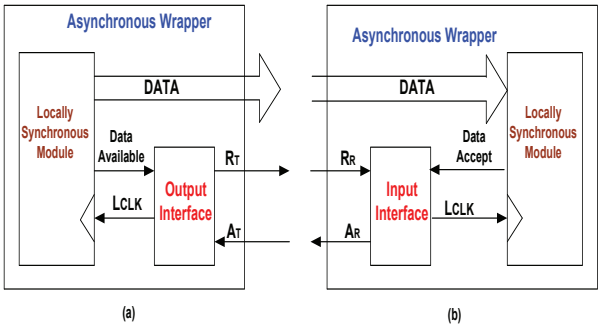

Figure 3. Locally synchronous module presenting weak interface interaction between the locally synchronous module (LSM) and asynchronous interface: a) output; b) input [23].

On the other hand, Fig. 4 and 5 show the architectures of our proposed input/output communication control ports (interfaces), which implements the weak interaction between the interface and the LSM, through the insertion of a pausible clock generator. This new architecture works perfectly and shows to be an alternative option of implementation to the previous ones, considering the advantages of a Pausible Clock Generator.

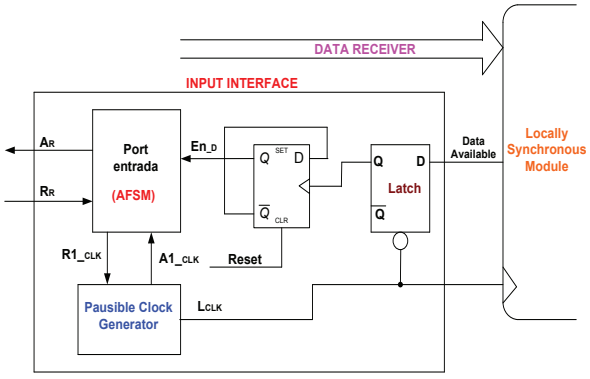

Figure 4. Proposed input interface/control.

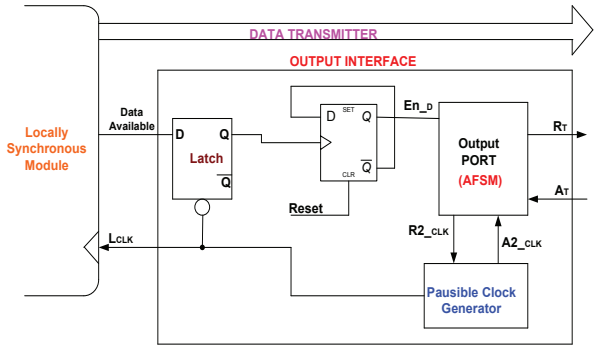

Figure 5. Proposed output interface/control.

Considering the architectures presented in Fig.4 and 5, Fig.6 summarizes the proposed asynchronous wrapper that receives and transmits data and can be compared to the one presented in Fig.3.

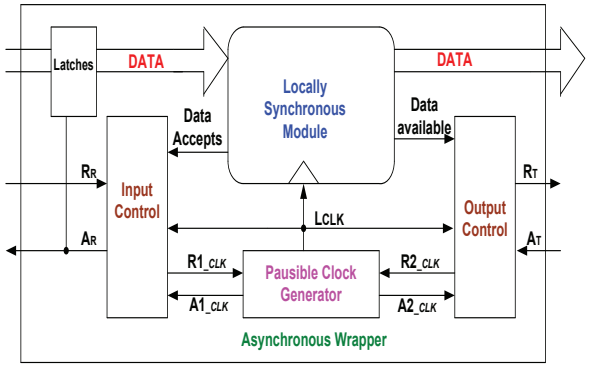

Figure 6. Proposed I/O Asynchronous Wrapper (AW) – Full Architecture.

As can be seen in Fig.6, the proposed AW comprises five blocks: locally synchronous module, two different controls (for input and output of data – Fig.4 and 5), a pausible clock generator (which can generate the signal for both controllers together) and a register based on transparent latches. In order to implement this kind of architecture using PCG, some efforts have to be done, once the target architecture may be focused in a FPGA implementation. It is described below.

B. Pausible Clock Generator

Different kinds of Pausible Clock Generators (PCG) have been proposed and can be found in literature. Many ones are based on arbitrators (MUTEX), being inappropriate for FPGA. In [22, 27] some PCGs are proposed by using only logic gates and *C* latches. In order to illustrate and compare these architectures with the new one proposed in this paper, Fig. 7 and 8 show different kinds of PCGs, with the arbitrators based on basic gates and using the C-latches, while Fig. 9 shows the improved proposed pausible clock generator that shows to be full suitable for FPGA implementation and achieves a performance not shown by the previous ones.

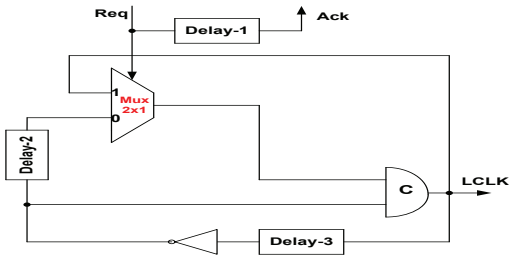

Figure 7. Pausible clock generator of [22]

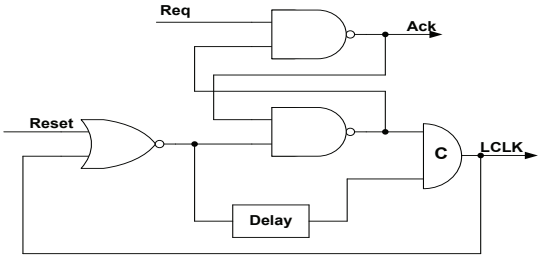

Figure 8. Pausible clock generator of [27].

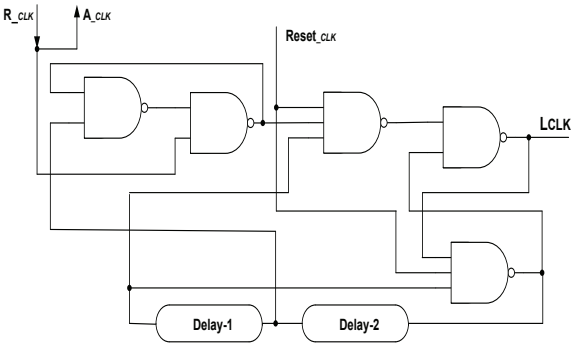

Figure 9. Proposed pausible clock generator that fullfills the requirements for FPGA implementation.

Once our AW architecture depends on a functional PCG, Fig.9 shows the developed architecture that makes our PCG not only a new one, but the one that works correctly for all FPGAs. In this design, the pause occurs when R_{CLK} goes 0→1 and the clock LCLK goes 1→0. Activation of the clock LCLK 0→1 occurs when R_{CLK} 1→0. Figure 10 shows the timing diagram of the proposed PCG. Figure 11 shows a symmetrical delay based on FFs, being full accepted by the compilers of FPGAs and, finally, Fig. 12 shows the proposed PCG with two possible pause entrances.

These architectures were developed focusing on FPGA implementation and showed to be fully efficient. Specific details on the design of them are out of the scope of this work, once it is part of the major proposed block AW that was previous described. Blocks with similar functionalities were not found in literature.

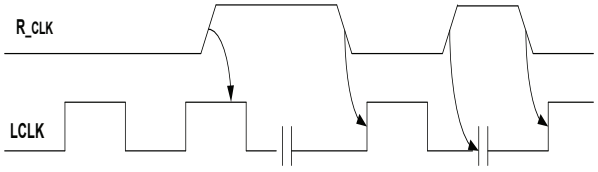

Figure 10. Timing diagram: pausible clock generator.

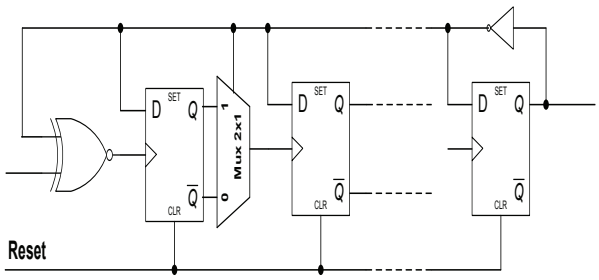

Figure 11. Symmetric delay.

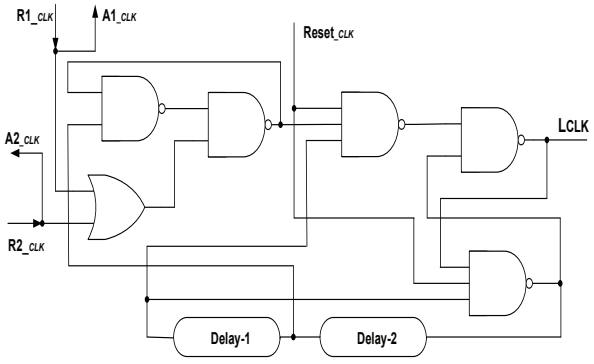

Figure 12. Proposed PCG: two input for stopping.

C. Ports: Essential-Hazard-Free specification

The so-called essential-hazard problem was first studied by Unger in [28]. He showed that essential-hazard is related to the specifications, considering BGWD asynchronous controllers. The proposed solution was the insertion of delay elements in the feedback lines. Unfortunately, it leads to a reduction of the reliability, modularity, technology migration and raising timing analysis problems. Oliveira et al. in [29] proposed a method in logic-synthesis style, which parts from the burst-mode specification (BM), synthesizing essential-hazard free asynchronous controllers without inserting any delay elements. As an initial condition for the implementation of this method, it is required that BM specification satisfies the "essential signal condition" [29]. It can be shown that "essential signal condition" [29] can be extended also to the extended burst-mode (XBM) specification. The BM specification, proposed by Nowick [30] and later extended by Yun in [25] as XBM specification, are quite popular to describe asynchronous controllers once they are familiar to designers of synchronous paradigm.

As an example, Fig.13 shows an XBM specification with inputs [*a, b, c, d*] and output [*x, y, z*], where the signal *a* represents a level sensitive signal (LSS) and the signals [*b, c, d*] are transition sensitive signals (TSS). TSS signals can be classified as "direct-don't-care" or "terminating" signals. The state transitions are labeled with the input burst/output burst (Ib/Ob) signals, where it is possible to find $Ob = Æ$. In the state transition from state 3→4, the signal b^* is a "direct don't-care signal" and the signal *c* is a "terminating edge signal". For more information on the XBM specification, refer to [25].

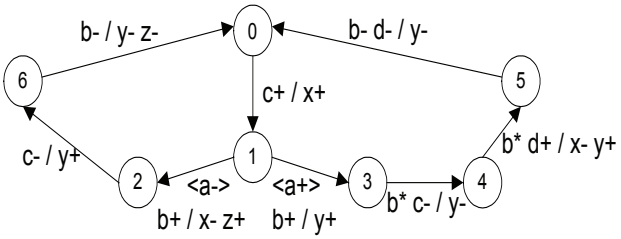

Figure 13. XBM specification.

The essential-hazard-free condition is based on the concepts of essential signals, proposed by Oliveira et al. in [29], for the Burst-Mode specification. In the XBM specification, during the state transitions, an input signal is considered a context signal only if it doesn't change its value during this transition (the signal must not be present in the label). On the other hand, an input signal is considered a trigger signal if it is labeled for this transition. As an example, consider 5→0 state transition, in Fig. 13 above. The signals b and d are considered trigger signals while signal c is a context signal (its value is 0).

Definition

Let A and B be a pair of states in an XBM specification and Ib/Ob be the input/output burst for the A→B transition, being Es an input signal (Es ∈ Ib) classified as terminating TSS. Es will be an essential signal only if it is a context signal in all transitions that are incident on state A and it is a trigger signal on the transition A→B.

For instance (see Fig. 13), b and d are not essential on transition 5→0 because they are trigger signals on transition 4→5. Signal b is essential on transitions 1→2 and 1→3, because it is a context signal on transition 0→1.

Lemma

(*Essential Signal Condition – ESC*) – The XBM specification is essential-hazard-free (XBM_EHF) if, and only if, for each state transition $t_r \in$ XBM that is labeled by the Ib_r/Ob_r and where $Ob_r \# Ø$, there must be, at least, one essential input signal. The proof of this lemma is similar to that one found in [29].

Figures 14 and 15 show, respectively, the XBM and BM specifications of the input and output ports of the AW_P [21]. Their descriptions are already adapted for use in the proposed AW_P. Applying the lemma 3.3.1, it can be concluded that these specifications are XBM_EHF specifications. Although being a EHF specification, it was shown that the implementation of this input port by the 3D tool generates essential hazard, leading to the conclusion that lemma 3.3.1 is necessary but not sufficient.

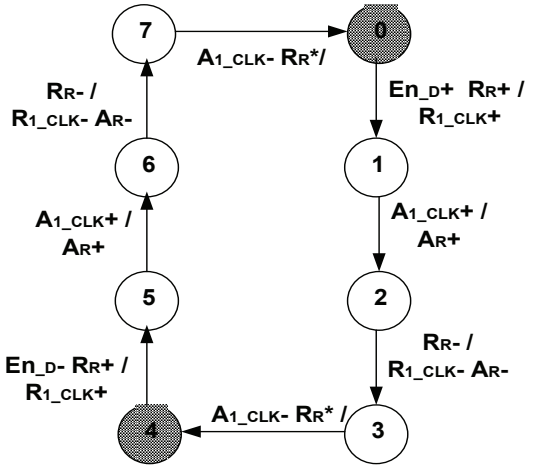

Figure 14. XBM specification: input port.

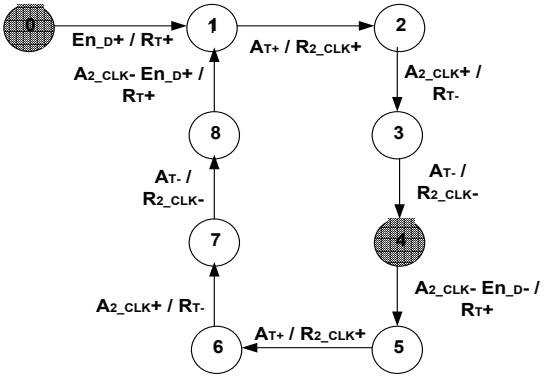

Figure 15. BM specification: output port

IV. MAPPING_DIRECT_XBM : METHODOLOGY

MAP_DIR_XBM method, proposed and described in [24], uses the direct mapping style for the synthesis of XBM asynchronous controllers. In this section, besides presenting the MAP_DIR_XBM method, it is introduced the sufficient condition that guarantees implementability of XBM controllers, free of essential hazard, improving lemma 3.3.1.

Figure 16 shows the logic structure of the target architecture for states (I,j) of the port controllers, built from the CC (control cell) and LO (logic output – standard RS architecture) blocks. Each state of the XBM is associated to a control cell (CC), which consists of a memory element + a block of combinational logic (condition logic – CL), as presented in Fig. 17.

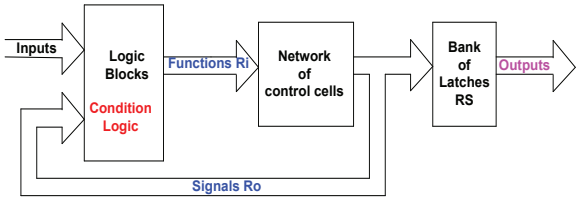

Figure 16. Structure: target architecture proposed and described by [24].

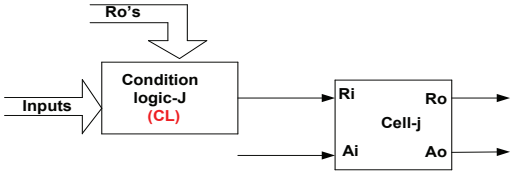

Figure 17. Control Cell

A. Memory element

As can be seen in Fig. 18, and was previously described in [24], the Memory Element presents two input signals [*Ri, Ai*] and two output signals [*Ro, Ao*]. Figure 19 shows the behavior of the memory element used in the MAP_DIR_XBM method. The input signal *Ri* enables the present state, while the output signal *Ro* triggers the enabling of the next state. On the other hand, the output signal *Ao* triggers the disabling process of the previous state, while the input signal *Ai* disables the state.

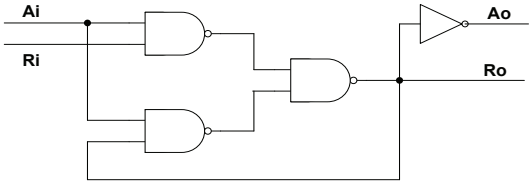

Figure 18. Memory element (Cell) of [24].

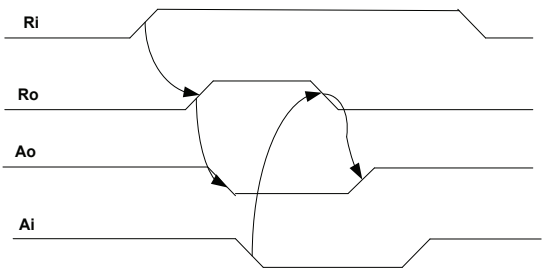

Figure 19. Timing diagram of the memory element.

B. Synthesis methodology

The behavior of the Communication Port (AFSM) of a GALS system is captured initially by the XBM specification, as illustrated in Fig.14. The synthesis procedure for the *MAP_DIR_XBM* method consists of seven steps:

1. Each state of the XBM specification will be represented by a CC. Paths with two state transitions must have a support CC.
2. For each CC, there is a CL (Condition Logic – excitation equation) block. The output of the CL is connected to the Req-in (memory elements). Support CCs do not need CL blocks.
3. Connect *Ro-i→CL-j* for each state transition, where signal *Ro* belongs to the initial state cell, and the CL block is related to the final state.
4. Connect *Ao-j→Ai-i* for each state transition on the backward direction. When there are two or more connections arriving at the *Ai* signal (decision state), a junction (JOIN) must be generated.
5. For each state *j* of the XBM that corresponds to the CL_j block, extract the sum-of-products Boolean function. Each state transition that comes from state *j* generates a product with *trigger signals of type terminating edge* that belongs to the input burst.
6. In each connection junction (JOIN), replace it by an OR gate.
7. Using the *Ro* signals from the CCs, extract the sum-of-product minimized Boolean functions F_{SET} and F_{RESET} (standard RS architecture) of the output signals.

C. Condition for the Implementation of EHF Ports

An essential hazard occurs in a XBM_AFSM when there is a race between input signals (belonging to a burst input) and the state variables, considering that this race is won by some of the state variable. The circuits obtained by the *MAP_DIR_XBM* method interact with the environment on the Ib/Ob mode. If the circuit satisfies the Lemma 4.3.1 below, it satisfies the unbounded gate and wire delay model. Therefore, the circuit is not sensible to the effects of cell mapping in FPGAs and to variations on the temperature and on the power source. Therefore, the synthesized communication ports are EHF. The implementation of ports that satisfies lemma 4.3.1 makes robust ports and, as consequence, we obtain a robust asynchronous wrapper.

Lemma

The resulting circuit based on a net of cells is considered EHF if, and only if, it is synthesized by the MAP_DIR_XBM method and the XBM specification is EHF.

Proof: Be any state transitions $S_i \rightarrow S_j \rightarrow S_k$ Î XBM specification, being labeled, respectively, by Ib_j and Ib_k. In the network of cells, the state S_j is activated by the signal Ro-i (which behaviors as a state variable) and by the input burst Ib_j. The activation state S_j leads to the activation signal Ro-j that activates the next state S_k. Like $Ib_k \not\subset Ib_j$, once the essential signal $E_{s\text{-}k} \notin Ib_j$, because $E_{s\text{-}k}$ belongs to Ib_k and was not activated because the XBM is EHF. Therefore the state S_k is not activated, what means that there was not any race that could activate the state S_k.

V. DESIGN OF THE HAZARD-FREE PORTS

The lemma 4.3.1 ensures the necessary and sufficient condition for the Input/Output ports of [21] to be used in the proposed asynchronous wrapper, improving it and making it more robust. As an illustration of the *Map_Dir_XBM* method, let us apply it on the input port, as follows.

As first and second steps of the methodology, one must replace each XBM state (see Fig 20) by a control cell (Fig. 18).

In the third step, one must perform the connections based on the handshaking protocol between control cells, in this case, the Request signals (see Fig. 21).

The fourth step performs the connections based on the handshaking protocol between control cells, in this case, the Acknowledge signal (see Fig. 22).

The fifth step of methodology extracts the Boolean equations of the CL blocks (see Fig. 22). For each one of the CLs there is a sum of product equation, where the product is the burst input of the state transition that focuses on the CL state.

The step six is not applied in this project.

The seventh step extracts the output Boolean equations for Fig. 22. The output logic is implemented in standard RS architecture, where the inputs are Ro's. Figure 23 shows the logic circuit of the input port.

Figure 24 shows the logic circuit of the output port synthesized by the *Map_Dir_XBM* method.

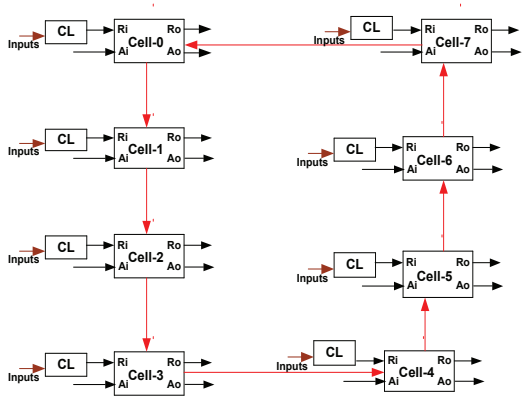

Figure 20. Net of cells: first and second steps.

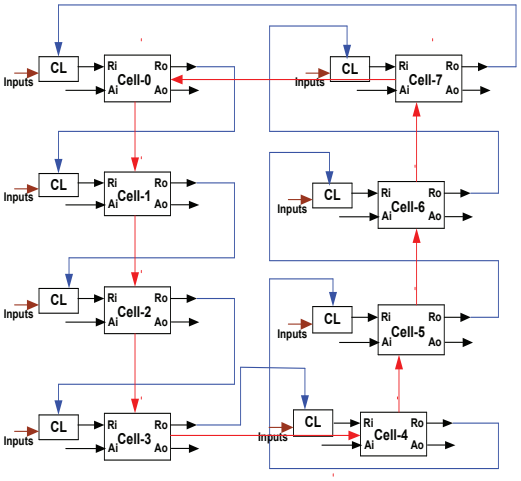

Figure 21. Net of cells: third step.

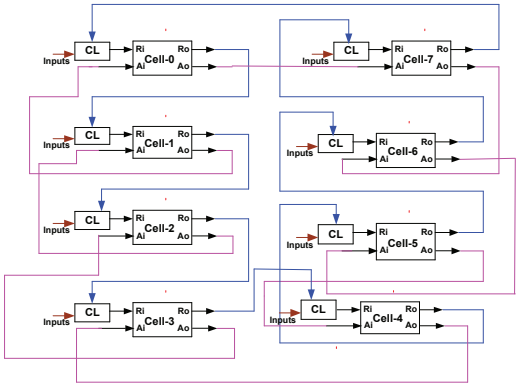

Figure 22. Net of cells: Input Port (fourth step).

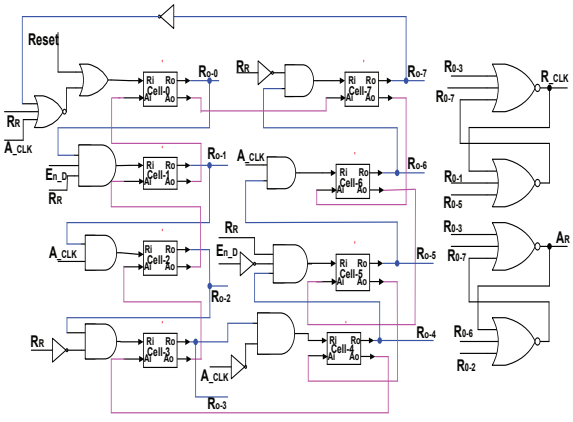

Figure 23. Logic circuit: Input Port (seventh step).

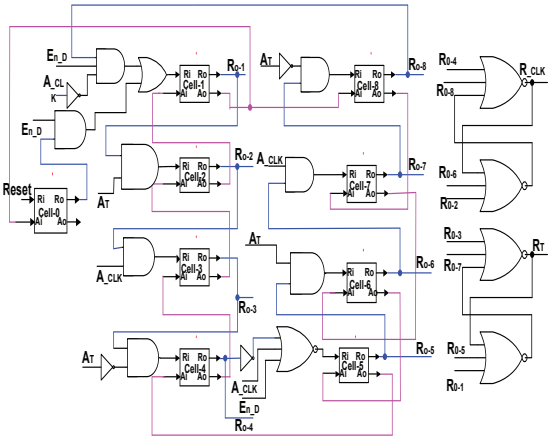

Figure 24. Logic circuit: Input Port (seventh step).

VI. SIMULATION RESULTS & FURTHER DISCUSSIONS

Firstly, it is important to list the advantages of the GALS systems: *a)* reuse of previously design IPs; *b)* the IPs work on their original frequency; *c)* use of standard synchronous tools for the design and verification of new IPs; *d)* dramatic reduction in the timing analysis efforts; *e)* reduction in electromagnetic interference; *f)* potential power reduction; and *g)* elimination of the clock skew issue.

This list of advantages of GALS system leads to a conclusion that GALS design can meet a relevant role in the future of digital design, but a major drawback to this potential use is the asynchronous interface, especially when implemented in FPGA.

Focusing on this kind of applications, this paper presented an asynchronous hazard-free oriented interface for FPGA implementations, which shows to be robust to any kind of mapping. When a XBM controller is EHF, being implemented in the architecture of Fig. 16, we say that it satisfies the "delay insensitive" model (DI) [6] and interacts with the environment in Ib/Ob mode. This delay model does not need to insert any element of delay or meet the requirements of isochronic fork. It was not found in literature any work presenting an interface with similar properties.

During the previous sections, it was shown the detailed design of ports and PCG of a new asynchronous GALS wrapper architecture, in order to be implemented in FPGAs. These designed structures were simulated in Quartus II version 9.0 of ALTERA, family Cyclone III, in the device EP3C40F780C6 [31]. Figure 25 shows the simulation result of the proposed asynchronous wrapper, which satisfies the XBM specifications. It required 58 LUTS with 14ns of latency time.

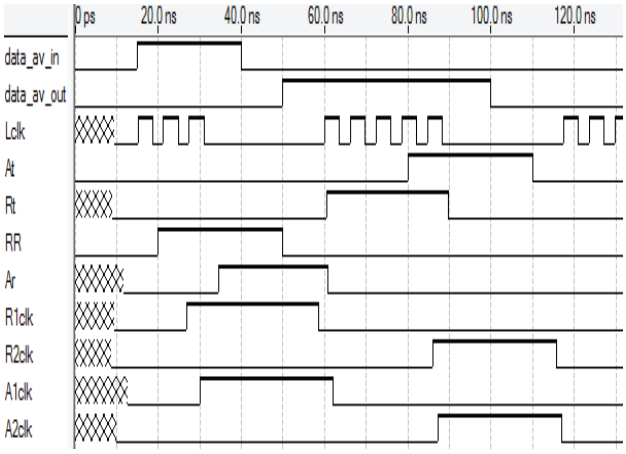

Figure 25. Simulation of asynchronous wrapper.

Figure 26 shows the simulation of the proposed PCG, showing the stopping and starting of the clock. Figure 27 shows the results of the three PCGs, involving six parameters. The delay elements inserted in the three PCGs are equal. Comparing the proposed PCG with the PCGs presented in [22,27], our proposal presents a reduction in three of these parameters.

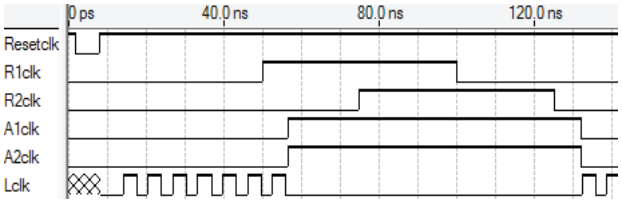

Figure 26. Simulation: PCG.

PCGs	Time Req→Ack	Stop time of clock	Start time of clock	Time of cycle	Power consumed	Nu. of LUTS
Figure 7	9,4ns	**6,3ns**	8,5ns	5,3ns	106,8mw	8
Figure 8	8,3ns	7,3ns	7,9ns	**2,4ns**	106,8mw	**5**
Figure 12	**6,5ns**	8,3ns	**7,2ns**	3,5ns	**106,7mw**	11

Figure 27. Results: PCG's parameters.

VII. CONCLUSION

GALS systems implemented in FPGAs showed to be an interesting design style. Through the results of this work, it was possible to discuss typical problems concerning to asynchronous interface design, dedicated for FPGAs, especially with the asynchronous wrapper. It was proposed a new architecture to an asynchronous wrapper that is able to overcome the previously discussed drawbacks, showing to be a good option for those designers who need to implement GALS in FPGA. In this context the proposed architecture (controllers) showed to be essential-hazard-free, not needing any special cares in implementation concerning to LUTs choice and being fully compatible with FPGA. As additional advantages of the proposed architecture we showed: total autonomy that synchronous modules achieve when interacting with the asynchronous wrapper; ports can be synthesized in the direct mapping style (so without knowledge of asynchronous logic synthesis); and ports interact with environment in *Ib/Ob* Mode, not needing a timing analysis. Simulation results showed to be correct, agreeing to the expected ones and proving the high potentialities of the new structures proposed. Finally, it was presented the logic and essential-hazard free synthesis of input/output ports. Future works lead to a robust asynchronous interface for FPGA GALS implementation, involving FIFO.

REFERENCES

[1] G. DE Micheli, "An Outlook on Design Technologies for Future Integrated Systems," IEEE Trans. on CAD of Integrated Circuits and Systems, vol. 28, no.6, pp. 777-789, June 2009.

[2] K. D. Muller-Glaser, et. al. "Multiparadigm Modeling in Embedded Systems Design", *IEEE Trans. on Control Systems Technology*, vol. 12, no. 2, March 2004.

[3] A. J. Martin and M. Nystrom, "Asynchronous Techniques for System-on-Chip Design*," Proc. of the IEEE*, vol.94, no. 6, pp.1089-1120, June 2006.

[1] J. J. Rodriguez, et. Al., "Features, Design Tools, and Applications Domains of FPGAs", *IEEE Trans. on Industrial Electronics*, vol. 54, No. 4, pp.1810-1823, August 2007.

[2] P. P. Czapski and A. Sluzek, "A Survey on System-Level Techniques for Power Reduction in Field Programmable Gate Array (FPGA)-Based Devices", The Second Int. Conf. on Sensor Technologies and Applications, pp.319-327, 2008.

[3] C. J., Myers, "*Asynchronous Circuit Design*", Wiley & Sons, Inc., 2004, 2a edition.

[4] W. Hardt, et. al., "Architecture Level Optimization for Asynchronous IPs", Proc. 13th Annual IEEE Int. Conf. ASIC/SOC, pp.158-162, 2000.

[5] M. Tranchero and L. M. Ryneri, "Implementation of Self-Timed Circuits onto FPGAs Using Commercial Tools", 11th Euromicro Conf. on Digital System Design Architectures, Methods and Tools, pp.373-380, 2008.

[6] N. Huot, et. al., "FPGA architecture for multi-style asynchronous logic," Proc. of the Design, Automation and Test in Europe Conference and Exhibition, pp. 32-33, 2005.

[7] D. M. Chapiro, *Globally-Asynchronous Locally-Synchronous Systems*, PhD thesis, Stanford University, October 1984.

[8] D. S. Bormann and P. Y. K., "Asynchronous Wrappers for Heterogeneous Systems," Proc. Int. Conf. Computer Design (ICCD), pp.307-314, October 1997.

[9] P. Techan, M. Greenstreet, and G. Lemieux, "A Survey and Taxonomy of GALS Design Styles," *IEEE Design & Test of Computers*, vol. 24, pp.418-428, September-October 2007.

[10] F. K. Gurkaynak, et al., "GALS at ETH Zurich: Success or Failure ?," , Proc. 12th IEEE Int. Symposium on Asynchronous Circuits and Systems, pp. 150-159, 2006.

[11] M. Krstic´, et al., "Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook," *IEEE Design & Test of Computers*, vol. 24, pp. 430-441, September-October 2007.

[12] A. Kumala, et al., "Reliable GALS Implementation of MPEG-4 Encoder with Mixed Clock FIFO on Standard FPGA," Int.

Conf. on Field Programmable Logic and Application, pp. 1-6, 2006.

[13] X. Jia and R. Vemuri, "Using GALS Architecture to Reduce the Impact of Long Wire Delay on FPGA Performance," Proc. IEEE ASP-DAC, pp. 1260-1263, 2005.

[14] M. Najibi, et al., "Prototyping Globally Asynchronous Locally Synchronous Circuits on Commercial Synchronous FPGAs," Proc. IEEE 16th RSP, pp.63-69, 2005.

[15] E. Amini, M. Najibi and H. Pedram, "Globally Asynchronous Locally Synchronous Wrapper Circuit based on Clock Gating," Proc. Emerging VLSI Technologies and Architectures, pp.193-199, 2006.

[16] E. Amini, et. al., "FPGA Implementation of Gated Clock based Globally Asynchronous Locally Synchronous Wrapper Circuits," Proc. IEEE Int. Symposium Systems, Circuits and Signal, pp. 1-4, 2007.

[17] H. A. Farouk, M. T. El-Hadidi, "Implementing Globally Asynchronous Locally Synchronous Processor Pipeline on Commercial Synchronous FPGAs," Proc. IEEE 17th International Conference on Telecommunications, pp.989-994, 2009.

[18] J. Muttersbach, T. Villiger, and W. Fichtner, "Practical Design of Globally-asynchronous Locally-synchronous System," Proc. IEEE 6th Int. Symposium Advanced Research in Asynchronous Circuits and Systems, pp.52-59, 2000.

[19] J. Pontes, et al., "SCAFFI: an Intrachip FPGA asynchronous interface based on hard macros," 25th Int. Conf. on Computer Design, pp.541-546, 2007.

[20] A. R. Ravi, "Globally-Asynchronous, Locally-Synchronous Wrapper Configurations for Point-to-Point and Multi-Point Data Communications," Master of science, University of Central Florida, 2001.

[21] D. L. Oliveira, et al. "A New Memory Element for Synthesis by Direct Mapping of Asynchronous FSMs from XBM Specification," Proc. XIII SIGE, SJC, Brazil, 2011.

[22] K. Y. Yun and D. L. Dill, "Automatic Synthesis of Extended *Burst*-Mode Circuits: Part I (Specification and Hazard-Free Implementation) and Part II (Automatic Synthesis)," *IEEE Trans. on CAD of Integrated Circuit and Systems*, vol. 18:2, February, pp. 101-132, 1999.

[23] R. M. Fuhrer, et al., "Minimalist: An environment for the Synthesis, verification and testability of burst-mode machines," Technical Report, Columbia University, TR-CUCS-020-99, 1999.

[24] R. Gagné, J. Belzile, and C. Thibeaut, "Asynchronous Component Implementation Methodology for GALS Design in FPGAs," Proc. IEEE Conf. NEWCAS-TAISA, pp.1-4, 2009.

[25] S. H. Unger, "Hazards and Delays in Asynchronous Sequential Switching Circuits," *IRE Trans. on Circuits Theory*, vol. 6, pp.12-25, March, 1959.

[26] D. L. Oliveira, et al., "Burst-Mode Asynchronous Controllers on FPGA," *Int. Journal of Reconfigurable Computing*, vol. 2008, pp.1-9, 2008.

[27] S. M. Nowick, "Automatic synthesis of burst-mode asynchronous controllers," *Ph.D. Thesis*, Technical report CSL-TR-95-686, 1995.

[28] Altera Corporation, 2011, www.altera.com.