doi: 10.5028/jatm.v5i1.191

An Asynchronous Interface with Robust Control for Globally-Asynchronous Locally-Synchronous Systems

Duarte Lopes de Oliveira[1], Eduardo Lussari[1], Sandro Shoiti Sato[2], Lester de Abreu Faria[1]

ABSTRACT: Contemporary digital systems must necessarily be based on the "System-on-Chip" (SoC) concept. Especially in relation to the aerospace industry, these systems must overcome some additional engineering challenges concerning reliability, safety and low power. An interesting style for aerospace SoC design is the GALS (Globally Asynchronous, Locally Synchronous) paradigm, which can be used for Very Large Scale Integration – Deep-Sub-Micron (VLSI_DSM) design. Currently, the major drawback in the design of a GALS system is the asynchronous interface (asynchronous wrapper – AW) when being implemented in VLSI_DSM. There is a typical AW design style based on asynchronous controllers that provides communication between modules (called ports), but the port controllers are generally subjected to essential hazard, what decreases the reliability and safety of the full system. Concerning to this main drawback, this paper proposes an AW with robust port controller that shows to be free of essential hazard, besides allowing full autonomy for the locally synchronous modules, creating fault tolerant systems as much as possible. It follows the Delay Insensitive (DI) model interacting with the environment in the Generalized Fundamental Mode (GFM) without the need to insert any delay elements. Additional delay elements, although proposed by some previous work found in literature, are not desirable in aerospace applications. The proposed interface allows working on Ib/Ob mode, showing the DI model is more robust than the QDI model and, therefore, it does not need to meet isochronic fork requirements nor timing analysis. Once an interface presenting similar properties was not found in literature, the proposed architecture proved to have great potential of implementation in practical VLSI_DSM designs, including the aerospace ones, once it overcomes the main engineering challenges of this kind of industry.

KEYWORDS: Aerospace systems, Reliability, Low power, Asynchronous controllers, GALS.

INTRODUCTION

Contemporary digital systems are usually implemented on Very Large Scale Integration (VLSI) and must necessarily be based on the "System-on-Chip" (SoC) concept. The reason for that is to satisfy the ever-growing demand for higher performance, reusability and low-power requirements (De Micheli, 2009; Muller-Glaser *et al.*, 2004). Especially in relation to the aerospace industry, these systems must overcome some additional engineering challenges concerning reliability, safety, high complexity and the unavailability of component failure data, generating fault tolerant systems as much as possible (Sues, *et al.*, 2005; Bertuccelli, 2008). SoC circuits are composed of functional modules, which can be the intellectual property cores (IP-cores) from many different vendors. These IP-cores are pre-designed, verified, tested and optimized for high-performance, providing both cost and development time reduction. Once SoC circuits are implemented in deep-sub-micron (DSM) technologies (VLSI_DSM) (for example, 70 nm, 500M transistors for chip and f=2,5 GHz), delays caused by wires prove to be big when compared to the gate timing, and the difference between minimal and maximum delays in the gates is significant (Jain *et al.*, 2001; Martin *et al.*, 2006). Therefore, when SoC circuits are implemented using only a global clock signal, they are subjected to speed and power penalties (clock skew, distribution networks etc.), thus making timing analysis very complex (Friedman, 2001). Besides that, the harsh environment found in aerospace applications, with high temperature variations, can make this time analysis even more difficult.

1.Instituto Tecnológico de Aeronáutica – São José dos Campos/SP – Brazil **2.**ETE Ferraz de Vasconcelos – São Paulo/SP – Brazil
Author for correspondence: Lester A. Faria | Praça Marechal Eduardo Gomes, 50 – Vila das Acácias | CEP 12228-901 São José dos Campos/SP – Brazil | E-mail: lesteraf@gmail.com
Received: 14/11/12 | **Accepted:** 18/01/13

Asynchronous project methodologies (Martin *et al.*, 2006; Myers, 2004) can naturally eliminate such challenges by removing the clock signal from the design. Different classes of asynchronous circuits may be used to implement SoCs, which can be built from completely asynchronous modules, but these kinds of circuits are not a widely accepted solution. The main reasons for that refusal are: a) lack of reliable tools for asynchronous design; b) difficulties from hazard-free designing and testing; c) limited culture on asynchronous design; and d) lack of asynchronous IPs (Hardt *et al.*, 2000).

The aerospace industry imposes many additional challenges to the design of dedicated systems, such as the high complexity of systems; main power generation systems; missions' profiles and environment; high demand for new technologies; high reliability and safety requirements; unavailability of component failure data; component sizes; and especially tight schedules, what leaves no room for errors. Any problem in an aerospace system leads to big losses of aircrafts (or spacecraft), crews, missions and revenues. In this context, reliability and robustness are important, leading to lower maintenance cost and lower failure frequency. The objective is always to maximize system performance, while satisfying constraints that ensure a reliable operation (Sues *et al.*, 2005; Bertuccelli, 2008).

Concerning to this special situation and the features of both synchronous and asynchronous systems, intermediate solutions were proposed between "totally synchronous" and "totally asynchronous", such as the Globally Asynchronous, Locally Synchronous methodology (GALS). The term GALS was first used by Chapiro (1984), in his PhD thesis. A GALS system consists of many synchronous functional modules that communicate in the asynchronous form. In this paper, we refer to the GALS systems as digital systems partitioned in functional modules (that may be IPs), which carry their own individual clock signals and are unrelated between modules. An asynchronous communication scheme is provided for the communication between different modules with different clock domains. In order to handle the asynchronous communication between these modules, an interface circuit has to be added around each one of the synchronous modules, which is called an asynchronous wrapper (AW). The AW term was first used by Bormann *et al.* (1997). This local interface may be built by using local clocks, FIFOs, asynchronous controllers (Input *Ports*, Output *Ports*) etc. Techan *et al.* (2007) show different styles for asynchronous interfaces dedicated to GALS systems. Figure 1 shows a generic interface with a synchronous module as an example.

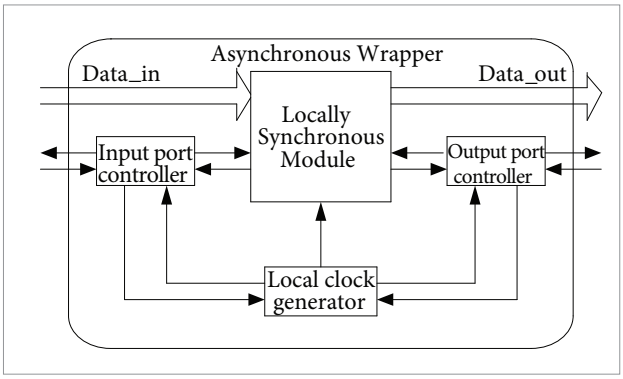

Figure 1. Asynchronous wrapper.

GALS systems have been successfully used in many implementations, including the Application Specific Integrated Circuit (ASIC) (Gurkaynak *et al.*, 2006; Amini *et al.*, 2006; Miller *et al.*, 2005) and Field Programmable Gate Array (FPGA) (Jia *et al.*, 2005; Kumala *et al.*, 2006; Yuan *et al.*, 2005). Currently, FPGA devices have shown to be a common choice for implementing digital circuits (Muller-Glaser, 2004), growing considerably in recent years. High-performance FPGAs, with up to 50 million gates, can be easily found nowadays, therefore allowing complex digital systems, such as GALS, to be programmed on them (De Micheli, 2009) and to be implemented in CMOS technology, DSM.

Asynchronous interfaces that use communication ports are of main interest, once they allow removing the asynchronous handshake scheme from the synchronous modules, allowing the synchronous module to be developed using standard techniques of synchronous design. Although the GALS methodology has solved problems related to the global clock signal, the communication between modules is already performed in the asynchronous paradigm, therefore being subjected to all its inherent problems.

IMPLEMENTATIONS OF PORTS: DIFFERENT APPROACHS

Different kinds of ports have been synthesized in the logic synthesis style (Myers, 2004). As an example, the ports proposed by Amini *et al.* (2006) have been specified in Signal Transition Graph (STG), which is a Petri-net-like speficification (Chu, 1987), being synthesized

in the Petrify tool (Cortadella *et al.*, 1997). These ports must meet the isochronic fork requirement (Myers, 2004), but the realization of this requirement in VLSI_DSM presents a high level of difficulty. Furthermore, the STG specification, as well as its synthesis method, is not familiar to synchronous paradigm designers.

The ports proposed by Muttersbach *et al.* (2000), Muttersbach (2001), Reddy Ravi (2001) and Pontes *et al.* (2007) were specified in Extended Burst-Mode (XBM) and Burst Mode (BM). These ports were implemented, respectively, in 3D (Yun *et al.*, 1999) and minimalist (Fuhrer *et al.*, 1999) tools. They interact with the environment in the generalized fundamental mode (GFM), requiring a timing analysis and being subjected to essential hazard, especially in the DSM technology. Concerning to this last drawback, the insertion of delay elements may be a possible solution (VLSI_DSM), but it degrades the testability and cycle-time of the system. The insertion of delay elements is not adequate when implementing GALS in FPGA as well, because these devices (FPGAs) are not designed to favor the insertion of delay elements.

AVOIDING ESSENTIAL HAZARD IN PORTS CONTROLLERS: INCREASING THE SYSTEM'S RELIABILITY

The XBM specification is quite interesting when describing port controllers, once it is not only "familiar" to synchronous paradigm designers, but also because the method that synthesizes ports described by XBM shows to be simpler when compared to the synthesis by STG (Myers, 2004). Yun *et al.* (1999) and Nowick (1993) proposed the insertion of delay elements on the feedback wires in order to avoid essential hazard in burst-mode controllers. Oliveira *et al.* (2008) proposed a sufficient condition that guarantees essential hazard-free operation on burst-mode controller without the need for extra delay elements, when mapped on VLSI_DSM or any type of LUT-based FPGA. The absence of delay elements is highly desirable when considering FPGA devices (difficulties in implementing this kind of elements) and, furthermore, in aerospace applications, in which the harsh environment must change the behavior of electronic components.

This paper proposes robust port controllers for asynchronous interfaces used in GALS style. They are completely free of essential hazard and are described in the XBM specification. The robust controller design for asynchronous interfaces is proposed as a solution to the increasing demand for high reliablility aerospace electronic systems. The paper also shows that the method proposed by Oliveira *et* al. (2011) to synthesize BM controllers free of essential hazard is improved for XBM controllers. These proposed ports are implemented in the following architectures: "Huffman machine with feedback output" and "standard RS". The use of both architectures enables a better performance of the system, besides being more reliable and providing safer operation for aerospace applications. A new AW for GALS with robust ports is also proposed. Once it is known that a major drawback in the design of asynchronous wrapper is the synthesis of these ports, the proposed AW proved to be very important and robust. These ports are easily implemented both in VLSI_DSM and LUT-based FPGA. Other advantages of this wrapper are: 1) total autonomy to the locally synchronous modules, when interacting with the proposed AW; and 2) its ports interact with the environment in the mode I_b/O_b, thus not requiring timing analysis and being more robust than the GFM mode. In this mode, a new input burst is immediately accepted when all signals of output burst change their values. All of these achieved features make the proposed architecture a good option for aerospace implementations, once it increases the reliability of the full system, overcoming some of the main challenges in this kind of industry.

DIFFERENT STYLES OF GALS DESIGN

Once the synchronous modules of a GALS system operate at different frequencies and/or different phases, the communication between them is subjected to metastability (Ginosar, 2003). Metastability occurs when a specific signal violates the setup time or the hold time of the memory element, and during any time the output voltage assumes an intermediate value that leads the circuit to achieve a random logic value. Metastability may occur in a timing window defined by the sum of "setup" and "hold" times. So, the GALS design style is determined according to the treatment of metastability, since there different ones in literature. Techan *et al.* (2007) propose specific taxonomy to classify these styles, in which they basically can be classified into three main styles: a) weak synchronous interface; b) pausible clock interface; and c) asynchronous interface.

WEAK SYNCHRONOUS INTERFACE

This style has three variants: a) heterochronous; b) mesochronous; and c) plesiochronous. In the heterochronous style (footer), the clocks of the synchronous modules run on different nominal frequencies (Techan *et al.*, 2007). On the other hand, in the mesochronous style (from Greek, meso means average), the clocks show the same average frequency, but with different unknown phases, which are generated by the same oscillator (Techan *et al.*, 2007). Finally, in the plesiochronous style (from Greek, plesio means "almost equal"), the clocks operate with equal nominal frequency, but being generated by different oscillators (Techan *et al.*, 2007). These styles always require timing analysis, starting from the knowledge of the clocks and using FIFO as a base, phase adjusters and, sometimes, synchronizers. The advantage of these styles is to enable low latency and high frequency clocks. On the other hand, there is the need for a rigorous timing analysis. Figure 2 shows a mesochronous interface that uses a phase adjuster (timing recovery circuit – TRC).

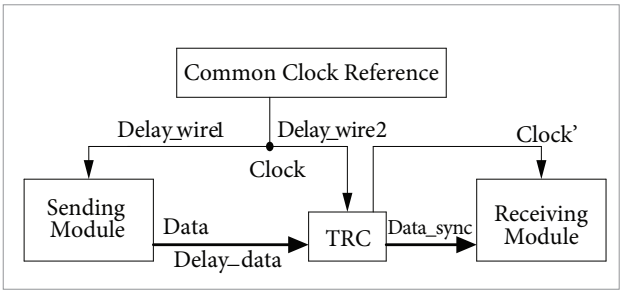

Figure 2. Mesochronous with TRC.

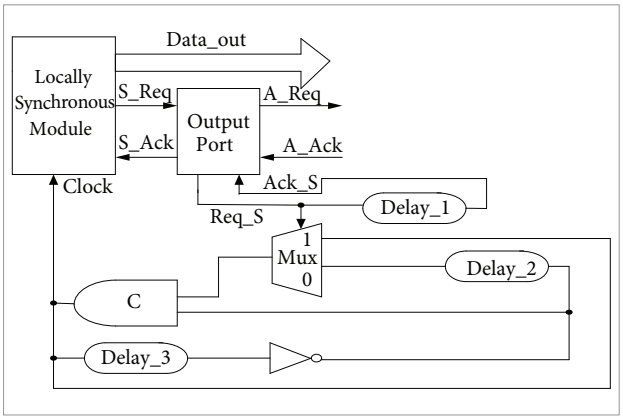

Figure 3. Pausible clock for FPGA described by Techan *et al.* (2007).

PAUSIBLE CLOCK INTERFACE

This style, firstly proposed by Chapiro (1984), tackles the problem of metastability by interrupting the clock signal. When data are ready for transmission, the clock is interrupted, enabling data synchronization. The synchronous modules have pausible clock signals. Most often, these clocks are locally generated using a ring oscillator and a mutual-exclusion circuit, or arbitrator, which properly generates the pause and restart of the clock (Yun *et al.*, 1999). The potential advantages of this style are the robustness in the treatment of metastability and power reduction. On the other hand, the weakness of this style is the possibility of "deadlock" and "jitter" (Mullins *et al.*, 2007). Different architectures have been proposed for pausible clocks, for example, the one involving FIFO (Techan *et al.*, 2007). Figure 3 shows an architecture involving pausible clock as an example.

ASYNCHRONOUS INTERFACE

This style uses circuits known as synchronizers and handshaking signals. The synchronous modules have clocks running freely at different frequencies, without any prior knowledge about their timing. Data are synchronized from one clock domain to another. Some examples of data synchronizers are the well known "two registers", or "double latches" (Mullins *et al.*, 2007), or some other more elaborated synchronization schemes, such as the "synchronization pipeline" (Sjogren *et al.*, 2000) and "FIFOs" (Dobkin *et al.*, 2006). The proposed synchronizers do not totally eliminate failure due to metastability, once the probability of failure different of zero percent remains (Dobkin *et al.*, 2006). The "two register" synchronizer presents as advantages its simplicity and robustness, but as a disadvantage there is an increasing area, power, and especially a high penalty in latency times, which leads to an increase of two clock cycles. Figure 4 shows the architecture of asynchronous interface as an example.

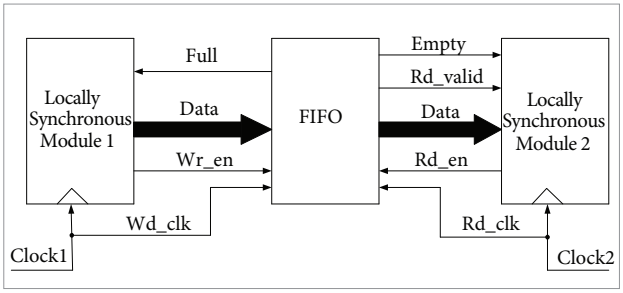

Figure 4. Asynchronous interface based on FIFO.

COMMUNICATION CONTROLLERS (PORTS)

GALS systems require asynchronous communication links, which can require two kinds of communication protocols: two or four stages handshaking. The ports can work as active (generating the "request" signal) or passive (generating the "acknowledge" signal). In GALS design there are two types of communication controllers: a) port of "demand", b) port of "poll" (inquiry). In the port of demand, the data being transferred are immediately required after the communication. Therefore, in this type of controller the clock must be immediately stopped (paused) and reactivated (restarted) when communication is done. In the port of poll, the clock is not stopped immediately. It defines when it is "safe" to send the data. The clock is stopped (paused) only in cases when there is the need for additional time, in order to troubleshoot eventual metastability.

XBM-EHF SPECIFICATION: CONDITION

BM is a kind of specification based on a state transition graph which was first proposed by Davis *et al.*, (1979), later formalized by Nowick (1993), and improved by Yun *et al.* (1999) as XBM. It allows multiple input changes and is usually used to describe Mealy Asynchronous Finite State Machines (FSM). These machines interact with the environment in GFM. In GFM, a new input burst can only occur if the controller is stable (with no activity in the ports or in the lines). The XBM specification supports the BM specification, introducing two kinds of input signals: a) conditional signal that is sensitive to level, showing non-monotonic behavior; and b) "directed don't care signals" that can activated concurrently with the output signals.

In this paper, the XBM specification is illustrated with the benchmark Biufifo2dma of the HP (see Fig. 5), with four inputs *(cntgt1,dackn, fain,ok*), two outputs (*dreq,frout*) and initial state *0*. The description *fain-dackn+/ frout+* in transition 4→3 means that the output (frout: 0→1) will follow the input burst (*fain:* 1→0 *AND dackn:* 0→1). Signals not enclosed in angle brackets and ending with + or — are "terminating". Signals enclosed in angle brackets are "conditionals", which are level sensitive with non-monotonic behavior. The input signals *dackn*, *fain* and *ok* are transition sensitive signals (TSS). The level sensitive signal *cntgt1* is used to describe the mutual exclusion between transitions 2→5 and 2→4. The "directed don't care signal" *fain** in transition 2→4 means that *fain* may either change its value or remain in its previous value. All state transition should have at least one signal called "compulsory". A compulsory signal is an input signal that, in the previous state transition, is not directed to don't care.

A TSS input signal in a XBM specification is considered as a context signal in a transition $A \rightarrow B$ if it does not change its value during such transition (it is not on the label). On the other hand, it is considered as a trigger signal if it is labeled during this transition. The input burst of each state transition can be represented by an input transition cube (ITC). For example, the ITC in state transition 0→1 on Fig. 5 is *cntgt1, dackn, fain, ok=2102* (the number *2* means"don't care"). In this example, *ok is* a trigger signal, while *dackn* and *fain* are context signals (whose values are *1* and *0,* respectively).

Definition 1.1: Let A and B be a pair of total states in a XBM specification, and I_b/O_b be the input/output burst for the $A \rightarrow B$ transition. Let E_s be one "terminating" input ($E_s \in I_b$). E_s is considered as an essential signal if it is a context signal on all transitions that address state A and is a trigger signal on the transition $A \rightarrow B$.

For instance (see Fig. 5), there is not an essential signal in state transitions 0→1, 4→3 and 3→2 because they are trigger signals on transitions 5→0, 2→4 and 2→5. Signal ok is essential on transition 1→2, because it is a context signal on transition 0→1. On transitions 2→4 and 2→5, *dackn* is essential signal.

Lemma 1.1 – (proof is presented by Oliveira *et al.* (2008)) A XBM specification is essential hazard-free (XBM-EHF) only if for each state transition labeled by I_b/O_b, if $O_b \neq \varnothing$, there must be, at least, one essential signal.

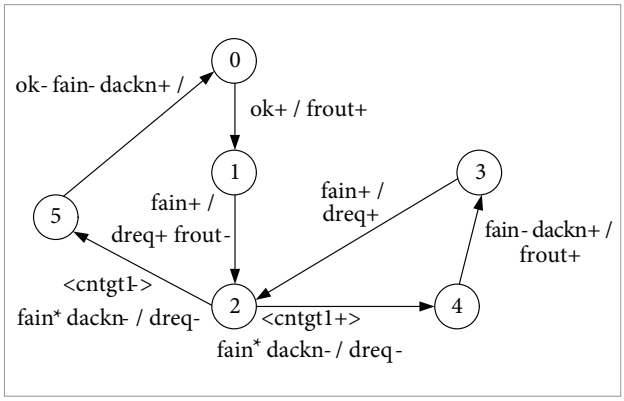

Figure 5. Extended burst-mode specification of Biufifo2dma.

As an example, Fig. 6 shows the *HP-mp-for-pkt* benchmark described by a BM specification. On all transition labels there is at least one essential signal. Therefore, it is a BM-EHF specification. Figure 7 shows the state flow map of HP-mp-for-pkt. As shown by Oliveira *et al.* (2008), which applies the rule generalized Ungle to check for essential hazard, this states flow map is subjected to essential hazard. The essential hazard depends on the code that the don't-care assumes, when held the logic coverage free of logic hazard.

ESSENTIAL CUBE CONDITION

Lemma 1.1 is a necessary and sufficient condition for an essential-hazard-free specification, but not for hazard-free implementation. The super-state concept will guarantee the latter condition. According to Oliveira *et al.* (2008), the concept of super-state is presented. It is used to obtain an implementation EHF. To simplify the implementation EHF, in this article we generalize the concept of super-state introducing the idea of essential cube.

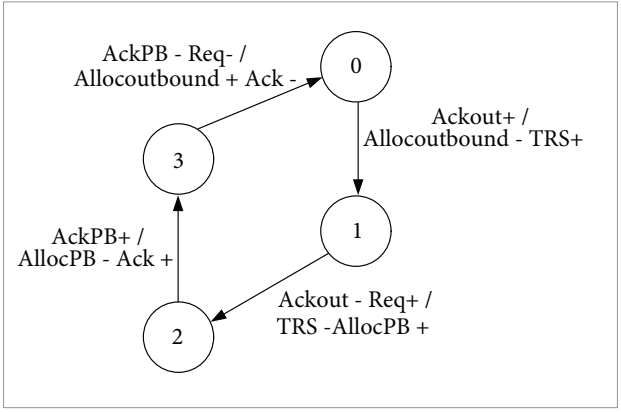

Figure 6. BM-EHF Specification.

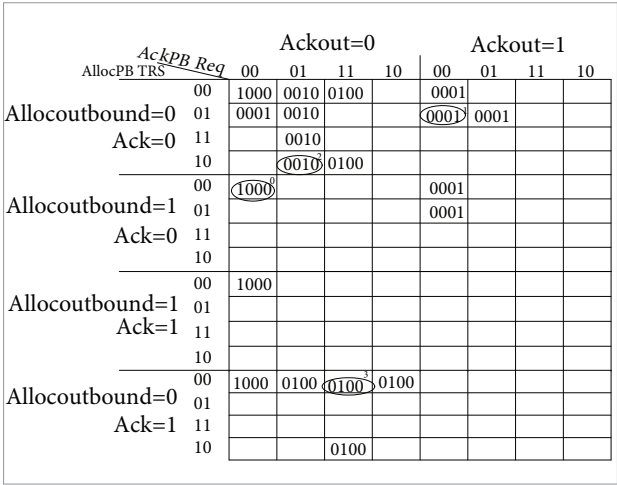

Figure 7. State flow map: BM spec. subject to essential hazard.

Definition 1.2: Consider an input burst I_b *(a, b, ..n)* and an output burst O_b *(x, y,..m).* We call a super-state the set of single total states defined by all 0/1 combinations of a subset S_{Ib} of the input burst signals, keeping all the remaining input signals and all the output signals constant.

Definition 1.3: Consider a XBM-EHF specification and a super-state *F* of the state transition *T*, so that *FÎ XBM-EH,* whereas *T* is labeled by *Ib/Ob*. We call essential cube of transition *T* all the total states related to the 0/1 combinations of input burst and output burst (*Ib/Ob).* Whereas the states not reachable in the cube *T* are encoded with the value of context signals, and trigger signals are don't-care.

A super-state XBM flow map is derived from a XBM-EHF specification by applying definition 1.2 to all total states. The essential cube is composed of 2^N states, in which N is the total number of input signals plus the output signals that are labeled in a state transition. Figures 8a-d are part of the flow map for the BM specification described in Fig. 6. Cells in blue are used to compose super-states and essential cubes (applying the definition 1.3). For example, the *0⟶1* transition (see Fig. 8a,b and 9a,b) creates super-state *1* composed of two total states: *AckPB Req Ackout Allockoutbound Ack AllocPB RTS=[0010001, 0000001].* State *0010001* is the final total state. Figure 9b shows the essential cube of the state transition 0→1, in which the next total states in blue are not reachable and belong to the essential cube. Due to the delays of gates and wires, the state totals which not are reachable can become reachable. For example, the 1⟶2 transition (see Fig. 8c,d and 9c,d) creates super-state *2*, composed of four total states: *AckPB Req Ackout Allockoutbound Ack AllocPB RTS= [1000010, 1100010 ,0100010,0000010].* State *0100010* is the final total state. Figures 9b,d respectively, show the essential cubes of the transitions 0→1 and 1→2. Lemma 1.2 and theorem 1.1 show the robustness of our controls.

Lemma 1.2 – Let T (B→A) be a state transition of XBM-EHF specification labeled by *Ib/Ob* and let an input signal any *Is* ∈ *Ib* and an output signal any *Os* ∈ *Ob*. If T is described by an essential cube, then in whatever order and whatever the time of arrival of *Is* and *Os* activation the total generated states belong to the essential cube T, and they all lead to the final total state A.

Proof: As a cube essential, T consists of 2^N total states, in which N is the sum of the signals that compose the input burst (*Ib*) and the output burst (*Ob*). As the next states not reachable in the transition B→A are encoded in the way in which the signals of *Ib* and *Ob* are don't-care, then whatever combination of activations of the signals *Is* and *Os* in T, the total states generated will belong to the cube essential T and lead to final state A, therefore the cube essential is free of essential hazard.

Theorem 1: The XBM-EHF specification has an EHF implementation in the "Huffman machine architectures with feedback output" or "standard RS" if ∀ state transition T (B→A) ∈ XBM-EHF, all your activation is covered by the cube essential T.

Proof: Lemma 1.2 says that if the XBM specification is EHF, then whatever the state transition T ∈ XBM has a cube essential T. As the context signals in T transition remains as a constant value in all the next states, which are reachable and not reachable in the cube essential, then regardless of the delays of gates and wires of the architectures, the activation of the next state belongs to the cube essential. As essential cube is EHF according to lemma 1.2, then the implementations on both architectures are EHF.

ASYNCHRONOUS WRAPPERS: ARCHITECTURE

The main objective of the proposed architecture is to provide a weak interface interaction between the locally synchronous module (LSM) and the asynchronous interface. Figure 10 shows the two different variables, "data available" and "data accept", as the only ones used for communication between LSM and the interface. When data available='1', data is ready to be transmitted, while when data accept='1' the data was received. Our architecture is based on the architecture proposal described by Reddy Ravi (2001).

Figure 11 shows the architecture of the proposed output communication control, which implements the weak interaction between the interface and the LSM, while Figs. 12 and 13 show the proposed input and output asynchronous wrapper, respectively, with the insertion of a gated clock generator. Finally, Fig. 14 shows the full proposed AW that receives and transmits data.

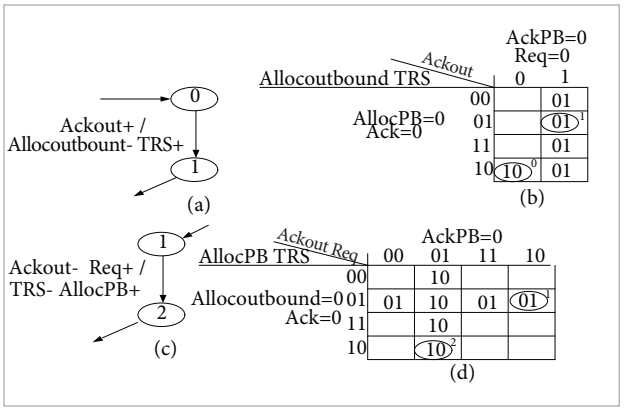

Figure 8. Part of BM specification: a) transition 0→1; b) flow map; c) transition 1→2;d) flow map.

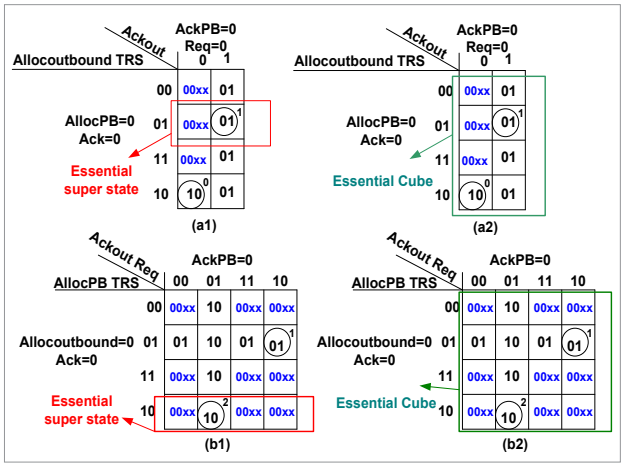

Figure 9. Part of BM flow map with ESS and EC: a) 0→1; b) 1→2.

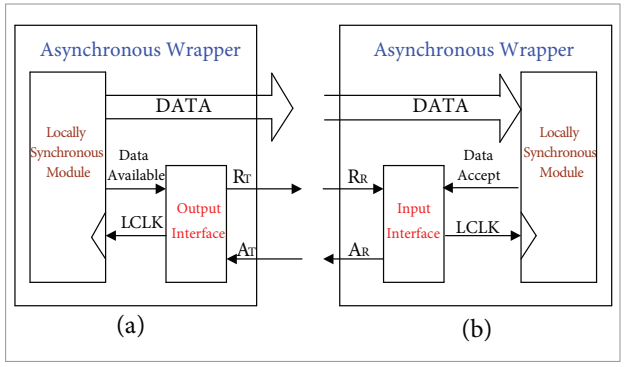

Figure 10. Locally synchronous module with weak interface: a) output; b) input.

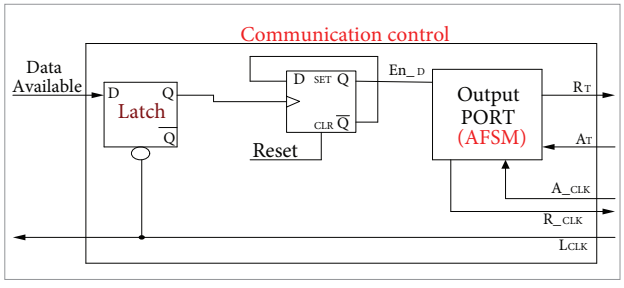

Figure 11. Communication control of output.

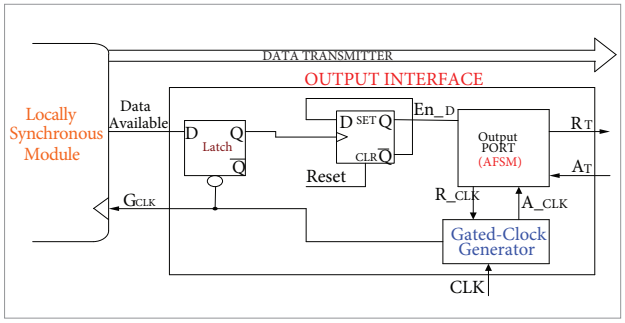

Figure 12. Proposed output asynchronous wrapper.

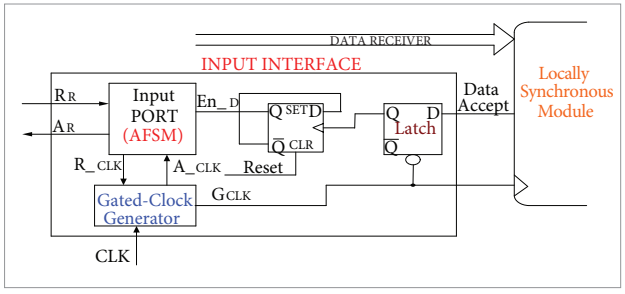

Figure 13. Proposed input asynchronous wrapper.

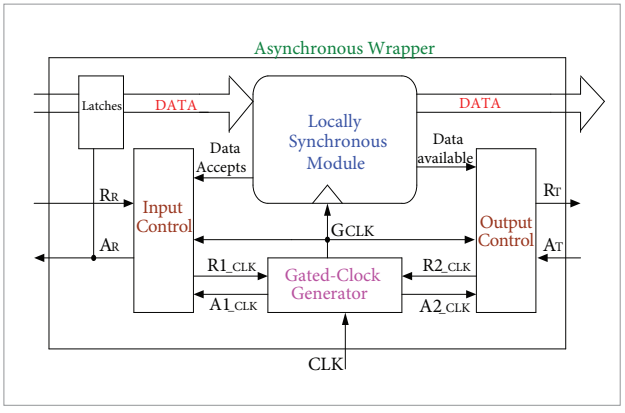

Figure 14. Proposed asynchronous wrapper: with I/O.

GATED-CLOCK GENERATOR

In this paper, a gated-clock generator (GCG) composed basically by two synchronizers and a gated-clock is also proposed. Figure 15 shows the timing diagram of the proposed GCG with the activation and deactivation of signal GCLK. While Fig. 16 shows the architecture of GCG, Fig. 17 shows the topology of the gated-clock and Fig. 18 shows the topology of its synchronizer. The stopping (pause) of the GCLK signal occurs when R_{CLK} switches 0→1 and after two clock cycles the signal "Stop" switches 0→1, thus determining the stopping (interruption) of signal GCLK.

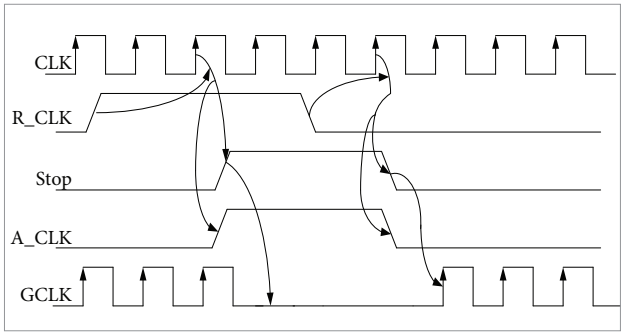

Figure 15. Timing diagram: gated-clock generator.

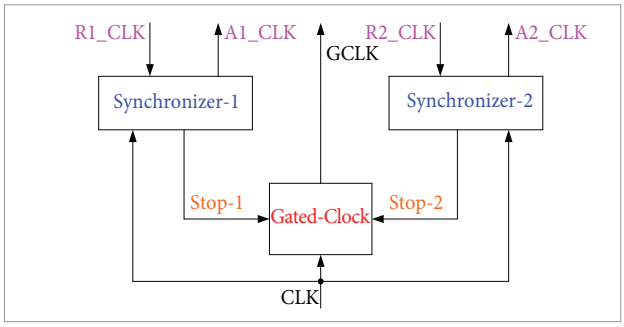

Figure 16. Architecture of the proposed gated-clock generator.

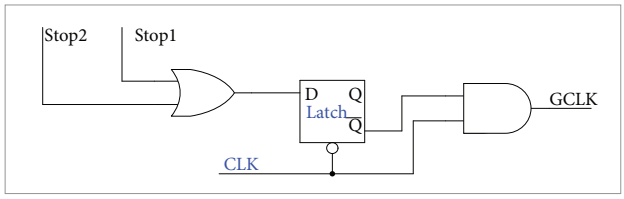

Figure 17. Topology of the gated-clock

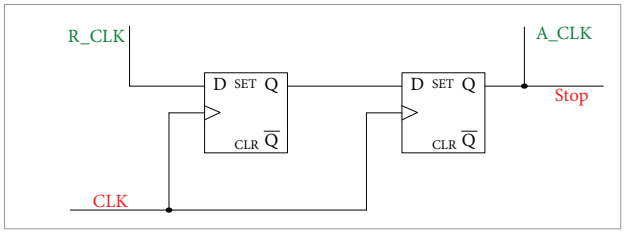

Figure 18. Topology of the synchronizer.

DESIGN: PORTS (AFSM)

The input/output ports used in the proposed AW were previously proposed by Muttersbach *et al.* (2000) and Muttersbach (2001). They are described in the XBM specification (as shown in Figs. 19 and 20). The XBM specification of the input/output ports meets the essential signal concept, therefore XBM_EHF.

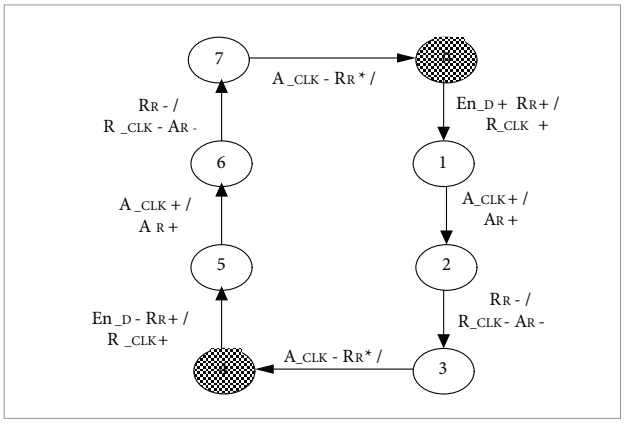

Figure 19. XBM Specification: input port described by Muttersbach et al. (2000) and Muttersbach (2001).

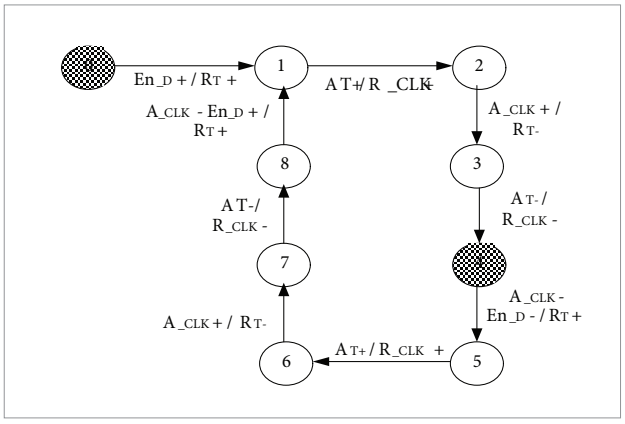

Figure 20. BM Specification: output port described by Muttersbach et al. (2000) and Muttersbach (2001).

PROCEDURE: SYNTHESIS OF PORTS

The ports designing method starts from the XBM description and is synthesized in four steps:

- Use the algorithm of Yun *et al.* (1999) and derive the minimum set of XBM flow charts;
- Encode XBM flow tables using the adjacency diagram (Unger, 1969);
- For each coded XBM flow table, insert the essential super-states, as seen in the previous section;
- Perform the logic minimization, logic-hazard-free, for each "non-input" signal in the "standard RS" and "machine Huffman with output fed back" architectures (Oliveira *et al.*, 2008).

Figure 21 shows the state flow map of the output port, with the introduction of a state signal 'Z' to solve conflicts, while Fig. 22 shows all the minterms (black and blue) used

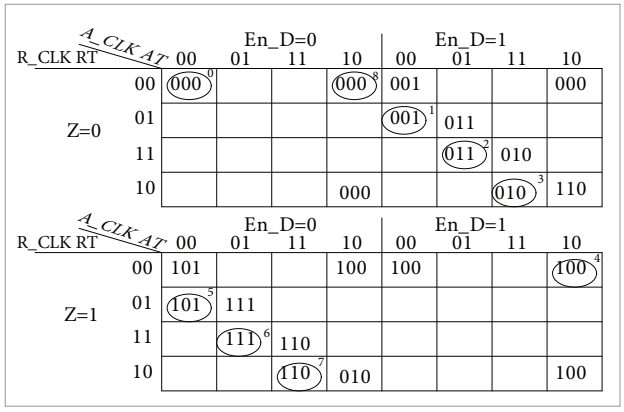

Figure 21. State flow map: output port subject to essential hazard.

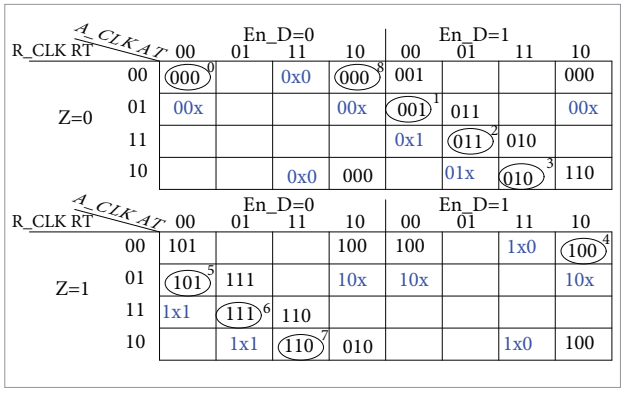

Figure 22. State flow map: output port essential hazard-free.

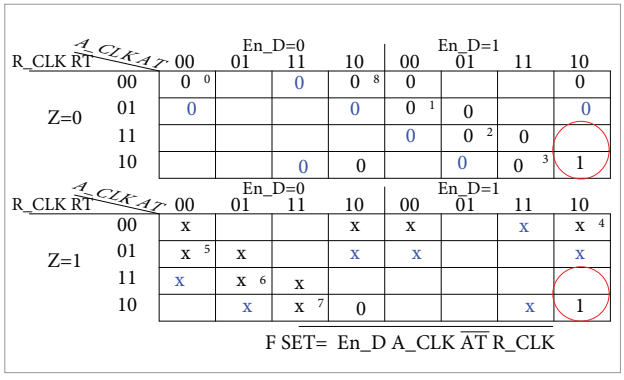

Figure 23. Karnaugh map: coverage hazard-free of signal Z (FSET).

in logic coverage, which ensure the output port to be free of essential hazard. The output port was implemented in the architectures "Huffman machine with output feedback" and "standard RS". Figures 23-26 show the logic coverage free of logic hazard using the Karnaugh maps. Finally, Figs. 27 and 28 show, respectively, the logic circuits of output and input ports.

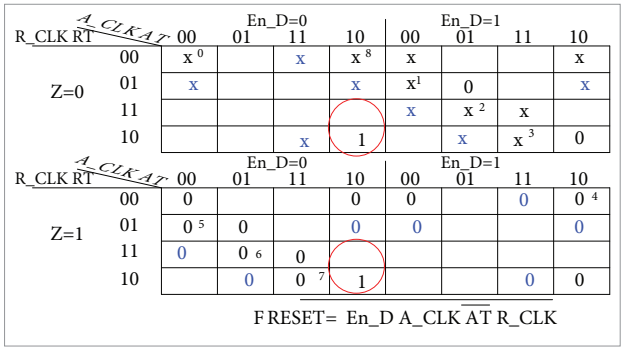

Figure 24. Karnaugh map:coverage hazard-free of signal Z (FRESET).

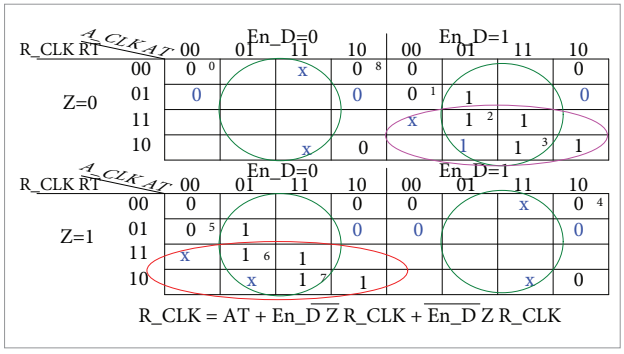

Figure 25. Karnaugh map: coverage hazard-free of signal R_CLK.

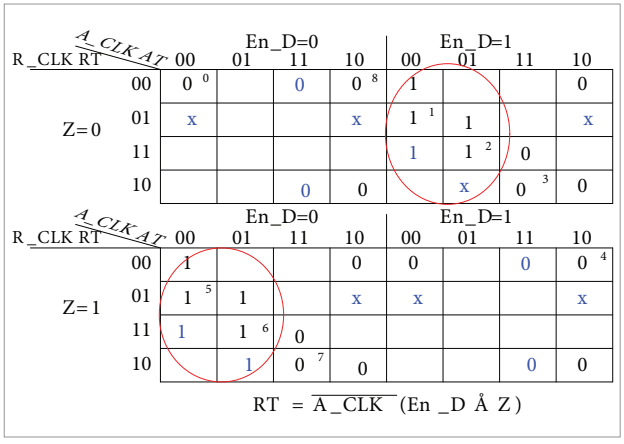

Figure 26. Karnaugh map: coverage hazard-free of signal RT.

DISCUSSION & SIMULATION

Oliveira *et al.* (2011) present a list of advantages of the GALS system, which leads to the conclusion that GALS design can play a relevant role in the future of digital design in all kind of applications, including aerospace ones. However, a major drawback to this use is the asynchronous interface.

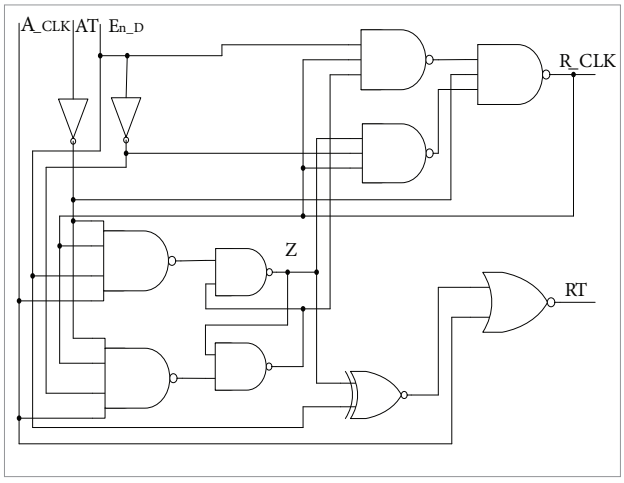

Figure 27. Logic circuit: free-hazard output port hazard-free.

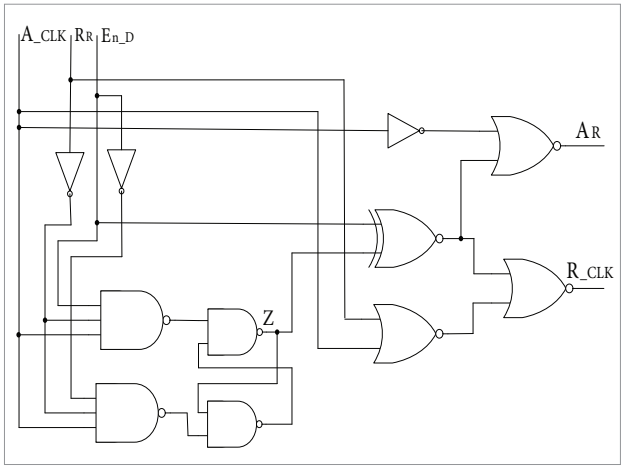

Figure 28. Logic circuit: input port hazard-free.

Focusing on this kind of application, the proposed hazard-free asynchronous interface proved to have a great potential, being highly desirable for the aerospace industry, once it overcomes the main challenges of this industry, thus increasing the reliability of the full system. In the treatment of essential hazard, our ports support any type of mapping either for VLSI_DSM or PLDs devices. It follows the Delay Insensitive model (DI) (Myers, 2004), restricted to interact with the environment in GFM, but without the insertion of any delay elements. This interface allows working in I_b/O_b mode, showing that the DI model is more robust than the QDI model, therefore not needing to meet isochronic fork requirements. An interface presenting similar properties was not found in literature. Figures 29 and 30 show simulations of I/O ports of the proposed AW, which show that the proposed architecture satisfies the XBM specification, are hazard-free and robust.

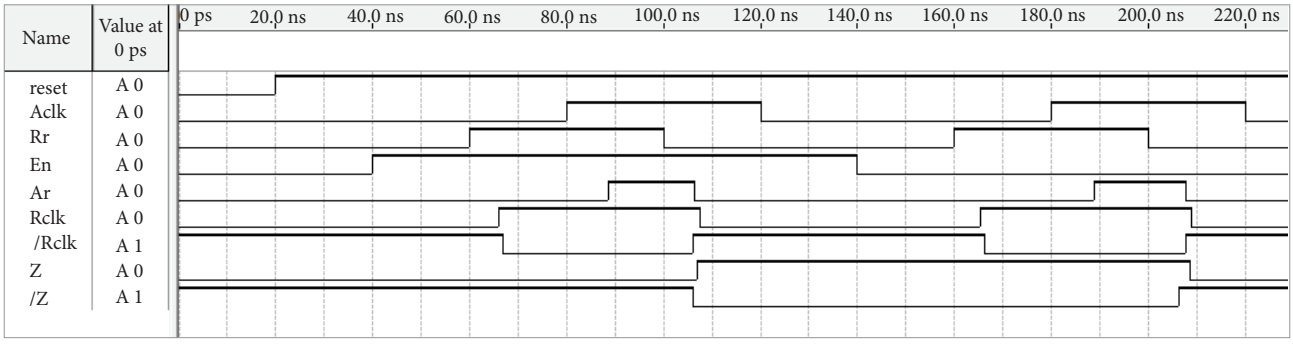

Figure 29. Simulation of input port.

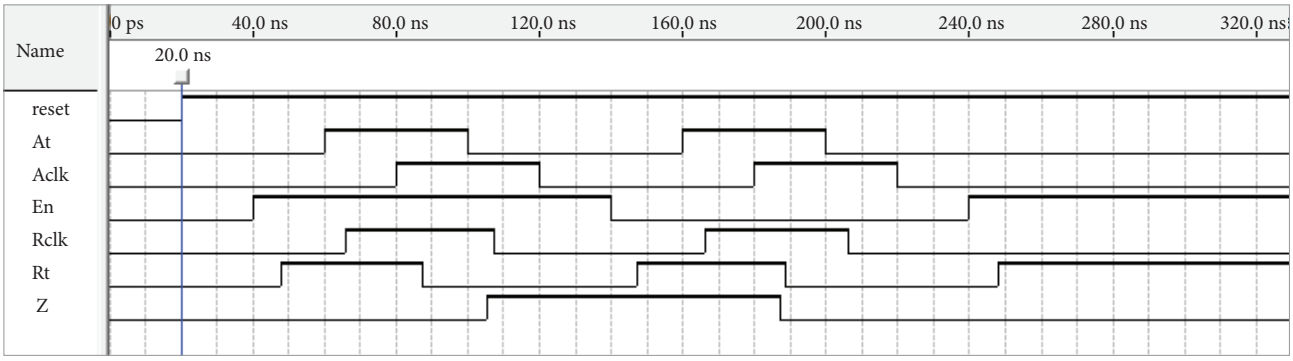

Figure 30. Simulation of output port.

CONCLUSION

GALS systems implemented in VLSI_DSM are an interesting design style for SoCs, however, typical problems concerning the asynchronous interface, especially for AW design, proves to be major drawbacks. In relation to aerospace applications, in which reliability and safety are major constraints, these drawbacks are prohibitive. Concerning this situation, a new architecture to AW was proposed in order to overcome the previously discussed problems, showing to be a good option for those designers who need to implement GALS in VLSI_DSM, including for aerospace applications, once it improves the reliability of the system, thus eliminating essential hazards. The achieved results showed that the proposed architecture is completely free of essential hazard and allows full autonomy for the locally synchronous modules. It follows the DI model, interacts with the environment in GFM without the need to insert any delay elements, as suggested by the previous papers found in literature, and allows working in Ib/Ob mode, proving to be more robust than the QDI model and, therefore, not needing to meet isochronic fork requirements nor requiring timing analysis. Since an interface presenting similar properties was not found in literature, the proposed architecture showed to have a great potential of implementation in all VLSI_DSM systems, including the aerospace ones, in which the harsh environment imposes additional challenges to the designers. Future work leads to a robust asynchronous interface for the implementation of GALS, involving FIFO and an application aimed for software-defined radio.

REFERENCES

Amini, E., Najibi, M. and Pedram, H., 2006, "Globally asynchronous locally synchronous wrapper circuit based on clock gating", Emerging VLSI Technologies and Architectures, IEEE Computer Society Annual Symposium, Vol. 00, pp. 2-3.

Bertuccelli, L.F., 2008, "Robust Decision-Making with Model Uncertainty in Aerospace Systems", Ph.D. thesis, MIT, September, 2008.

Bormann, D.S. and Cheung, P.Y.K., 1997, "Asynchronous Wrappers

for Heterogeneous Systems," Proc. Int. Conf. Computer Design (ICCD), pp. 307-314.

Chapiro, D.M., 1984, "Globally-Asynchronous Locally-Synchronous Systems", PhD thesis, Stanford University, October, 1984.

Chu, T.A., 1987, "Synthesis of Self-Timed VLSI Circuits from Graph-Theory Specifications", Ph.D. thesis, Dept. of EECS, MIT, June, 1987.

Cortadella, J. *et al.*, 1997, "Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers", IEICE Trans. Inf. Syst., Vol.E80-D, No. 3, pp. 315-325.

Davis, A.L. *et al.*, 1979, "A data-driven machine architecture suitable for VLSI implementation", In C.L. Seitz, editor, Proc. of the Caltech Conf. on Very Large Scale Integration, pp. 179-194.

De Micheli, G. 2009, "An Outlook on Design Technologies for Future Integrated Systems", CAD of Integrated Circuits and Systems, Vol. 28, No.6, pp. 777-789.

Dobkin, R., Ginosar, R. and Sotiriou, C.P., 2006, "High Rate Data Synchronization in GALS SoCs", TVLSI, Vol. 14, No. 10, pp. 1063-1074.

Friedman, E.G., 2001, "Clock distribution networks in synchronous digital integrated circuits", Proc. IEEE, Vol. 89, No. 5, pp. 665-692.

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

Ginosar, R., 2003, "Fourteen ways to fool your synchronizer", Proc. of ninth Int. Symp. On Async. Circuits and Systems, Vancouver, British Colombia, Canada, pp. 89-96.

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

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

Jain, A. *et al.*, 2001, "A 1.2 GHz alpha microprocessor with 44.8 GB/s chip pin bandwidth", IEEE Int. Solid-State Circuits Conf. Tech. Dig., pp. 240–241.

Jia, X., Vemuri, R., 2005, "Using GALS architecture to reduce the impact of long wire delay on FPGA performance", Proc. of the Asia and South Pacific Design automation Conf., pp. 1260-1263.

Krstic, M. *et al.*, 2007, "Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook", IEEE Design & Test of Computers, Vol. 24, pp. 430-441.

Kumala, A. *et al.*, 2006, "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.

Martin, A.J. and Nystrom, M., "Asynchronous Techniques for System-on-Chip Design", Proc. of the IEEE, Vol.94, No. 6, pp. 1089-1120.

Miller, S.P. *et al.*, 2005, "A Methodology for the Design and Verification of Globally Asynchronous/Locally Synchronous Architectures", NASA/CR-2005-213912, pp. 1-35.

Muller-Glaser, K.D. *et. al.*, 2004, "Multiparadigm Modeling in Embedded Systems Design", IEEE Trans. on Control Systems Technology, Vol. 12, No. 2.

Mullins, R. and Moore, S., 2007, "Demystifying Data-Driven and Pausible Clocking Schemes", Proc. of ASYNC'07, pp. 175-185.

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

Muttersbach, J. 2001, "Globally-Asynchronous Locally-Synchronous Architectures for VLSI Systems", Ph.D. Thesis, ETH, Zurich, 2001.

Myers, C.J. 2000, "Asynchronous Circuit Design", Wiley & Sons, Inc., 2004, 2nd edition.

Nowick, S.M, 1993, "Automatic Synthesis of Burst-Mode Asynchronous Controllers", PhD thesis, Stanford University, 1993.

Oliveira, D.L. *et al.*, 2008, "Burst-Mode Asynchronous Controllers on FPGA", Int. Journal of Reconfigurable Computing, Vol. 2008, pp. 1-10.

Oliveira, D.L. *et al.*, 2011, "Synthesis of Robust Conrollers for GALS_FPGA from Multi-Burst Graph Specification", Proc. IEEE VII Southern Conference on Programmable Logic (SPL), pp. 123-129.

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

Reddy Ravi, A., 2001, "Globally-Asynchronous, Locally-Synchronous Wrapper Configurations for Ponint-to-Point and Multi-Point Data Comunication", Masters of Science, University of Central Florida, 2001.

Sjogren, A.E. and Myers, C.J., 2000, "Interfacing Synchronous and Asynchronous Modules within a High-Speed Pipeline", IEEE Transactions on VLSI Systems, Vol. 8, No. 5, pp. 573-583.

Sues, R.H. *et al.*, 2005, "Reliability-Based MDO for Aerospace Systems", AIAA-2001-1521, pp. 1-8.

Techan, P., Greenstreet, M. and Lemieux,G., 2007, "A Survey and Taxonomy of GALS Design Styles", IEEE Design & Test of Computers, Vol. 24, pp. 418-428.

Unger, S.H. 1969, "Asynchronous Sequential Switching Circuits", John Wiley & Sons Inc.

Yuan, L. *et al.*, 2008, "Research on the Problems of Satellite Borne FPGA Based Finite State Machine", 2nd Int. Symposium on Systems and Control in Aerospace and Astronautics (ISSCAA), pp. 1-4.

Yun, K.Y. and Dill, D.L., 1999, "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, pp. 101-132.

Yun, K.Y. and Donohue, R.P., 1999, "Pausible clocking-based heterogeneous systems", IEEE Transactions on VLSI Systems, Vol. 7, No. 4, pp. 482-488.