FPGA Implementation of Self-Timed Asynchronous Pipeline Circuits using a Hybrid Architecture

Duarte L. Oliveira’[1], João B. Brandolin[2]
[1]Technological Institute of Aeronautics – ITA – IEEA, SJC, Brazil
[2]Federal Institute of Education, Science and Technology of São Paulo, Brazil
E-mail: duarte@ita.br, brandolin@ifsp.edu.br

Abstract— ***Digital design may be based on critical requirements (power consumption, robustness, high performances, and others) being also implemented in devices such as FPGA (Field Programmable Gate Array). Asynchronous design is an option for digital design, having several design styles in different delay models. An interesting class of the asynchronous paradigm is the QDI (Quasi Delay Insensitive) delay model, being designed in the asynchronous pipeline style. The QDI delay model has several important features in the Deep-Sub-Micron MOS (DSM-MOS) technology. In this paper, we propose hybrid architecture for self-timed asynchronous pipeline systems. This architecture satisfies the QDI delay model, but uses D flip-flops. This self-timed architecture has as a characteristic a control that is summarized in a C element and an XOR gate. Comparing with three literature controls that can be used in a self-timed linear pipeline of two stages, there was an average reduction of 55.6% in the number of LUTs (Look-Up Tables) targeting FPGAs.***

I. INTRODUCTION

An Embedded digital systems (EDS) may require high integration capacity, high speed, and low power consumption [1]. They are characterized in Deep-Sub-Micron MOS (DSM-MOS) technology, needing to operate with low noise. Besides, the difference between the maximum and minimum delays in the wires and gates is higher compared with other MOS technologies, and the delay in a wire may be greater than in a gate [2]. Conventional synchronous digital systems use a global clock signal to synchronize their operations and are quite popular due to the simplicity of design. There is also an abundant supply of commercial CAD tools for their automatic synthesis. However, a serious problem in DSM-MOS technology is managing the global clock signal, once it is a major cause of the noise, high electromagnetic emissions, and high power consumption. The clock signal distribution is also a task of increasing complexity, leading to potential clock skews. The timing analysis of high integration DSM-MOS synchronous digital circuits is extremely difficult. This limitation gets worse in the FPGA platform [3]. A common feature in many systems is the fact that they are battery-powered. Thus, the batteries should have a long life leading to the dissipated power to be a very important parameter in the design of such systems. In a digital system, registers are the main contributors to the dynamic power dissipation [3]. At the same time, those studies have shown that the clock signal is responsible for a large portion (15% to 45%) of the system power [4]. Then, an interesting alternative to digital design is the asynchronous paradigm, once it eliminates the problems caused by the clock signal, besides being highly modular and more robust.

Asynchronous digital systems operate by events and do not have a global signal for synchronizing the operations. This synchronization is carried out locally by handshaking protocols. Asynchronous digital systems can be designed in different styles and different classes of asynchronous circuits [5]. The class defines in what delay model the circuit operates and in what operating mode it communicates with the environment [5]. An interesting asynchronous circuit class is the Quasi Delay Insensitive (QDI), due to its features [6]:*a)* a high potential for better latency time; *b)* they are robust to temperature, supply voltage and process (PVT) variations; *c)* robustness to delay and Stuck-at faults (easily tested); *d)* they are highly modular, allowing reusability and the use of the intellectual property (IP) [7]; *e)* a better performance in security system designs (e.g., encryption) [8]; and f) timing analysis is simplified, it only satisfies the isochronic fork concept [6]. The QDI combinational circuits (QDI_CC) use the delay insensitive (DI) code (dual-rail, for example) for the encoding of their signals, operating mainly in the 4-phase handshake protocol. Although the latency time of the QDI circuits is defined by the average delay of the circuit and not by the highest delay, there is an overhead due dual-rail encoding, which increases area. An interesting solution to increase the throughput of QDI_CC is the pipeline style. Different proposals were made to the asynchronous pipeline control, which can be applied in QDI_CC [9-12].

However, these controllers are full custom or present an overhead in the area and latency time.

This paper proposes self-timed asynchronous linear pipeline architecture of N stages (see Fig. 1) to implement QDI_CC circuits. This architecture allows increasing the throughput of QDI_CC circuits with a low penalty in the area. As a case study, we present the design of a 1-bit QDI ALU (Arithmetic Logic Unit) found in many projects. The proposed pipeline QDI architecture, when compared with other architectures that support QDI pipeline, presents a significant reduction in the controller area. Compared to other controllers directed to QDI pipeline and based on basic gates, there is a reduction in area, obtained from simplified control. The proposed pipeline (see Fig. 1) shows that there are three different controls. In each case, there is an input control related to the input register (input detector + XOR gate), an intermediate control with XOR gate, and an output control related to the output register (XOR gate). Figure 2 shows the interaction between QDI modules and the latency-insensitive communication property [4], and that uses the acknowledge signal (Ack). The registers are based on FFs D this is due to their abundance in FPGAs devices, so there is a relaxation in the QDI condition, because FFs D operates in fundamental mode [6], therefore generating a hybrid architecture.

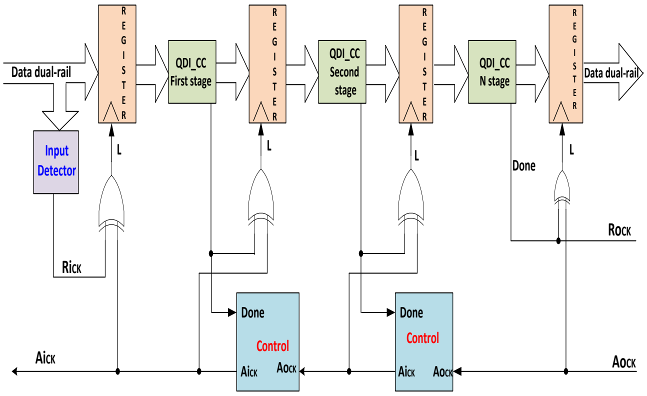


Fig. 1. *Proposed architecture: QDI Asynchronous Linear Pipeline System.*

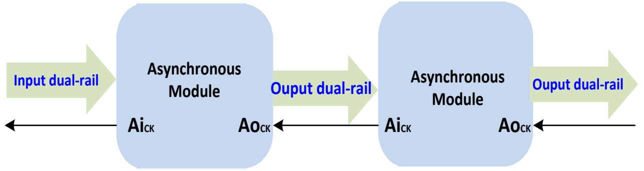


Fig. 2. *Proposed architecture: QDI Asynchronous Linear Pipeline System.*

II. Controller Synthesis: Pipeline Systems

The pipeline QDI_CC intermediate controller (control – see Fig.1) was described in burst mode (BM) specification [5], as shown in Fig. 3a, and Fig. 3b shows the table of BM flow. The specification has, as input signals {Done,AoCK} and as output signal {AiCK}. The controller operates in the intermediate stages of the pipeline, in the 4-phases handshake protocol. Figure 3c shows the Boolean equation of the AiCK signal of the proposed control, which may be implemented as one C element, the case of the AiCK signal, as shown in Fig. 4a and 4b.

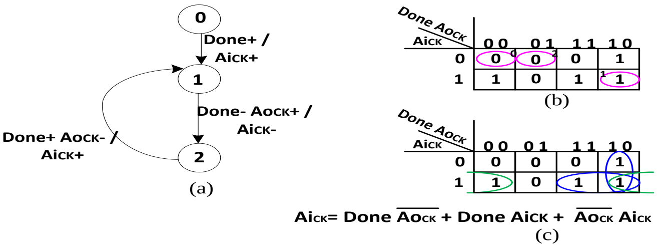


Fig. 3. *Specification of the Control: a) BM; b) table of flow BM; c) equation of Control-1: AiCK signal.*

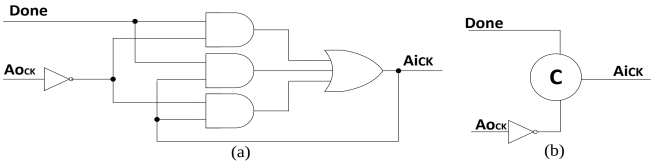


Fig. 4. *Control of Pipeline: a) equation Boolean of AiCK signal; b) equation of AiCK signal with C element and inverter.*

III. Synthesis of the QDI Basic Gates

This section explains the design procedure of a QDI_CC circuit, in the case of the basic gates being designed either in the DIMS (Delay-Insensitive Minterm System) style [13] or NCL (NULL Conventional Logic) style [14]. QDI_CC circuits are synthesized in DI codes. There are different DI codes, and in this paper, we adopted the dual-rail code. The QDI_CC circuits that will be synthesized operate in the 4-phases handshake protocol [9-11]. In the dual-rail code, each variable is encoded with two bits. For the variable *a*, we have *a0a1=00* (null - space), *a0a1=01* (1), *a0a1=10 (0)* and *a0a1=11 (never occurs).* The DI codes generate the operation conclusion signal without needing a delay element and a relatively simple circuit [15].

A. *NCL method for dual-rail gates*

An interesting style is the NULL Convention Logic (NCL) one, proposed by Kant et al. [14]. The NCL style is based on a set of 27 complex gates, implemented in CMOS transistor level. Figure 5b shows the symbol of a THmn NCL gate, where *n* is the number of inputs, and *m* is the minimum number of inputs that go to one, to set the output. The *n* variables must go to zero to reset the output. Figure 5a shows the operation table of an NCL gate. Fig. 6a shows the implementation of the THxor0 gate, whose function is Z = AB + CD in the architecture based on basic gates of [16]. For NCL XOR dual-rail gate, we have F1=a0b1 + a1b0 and F0=a1b1 + a0b0 as shown in Fig. 6b.

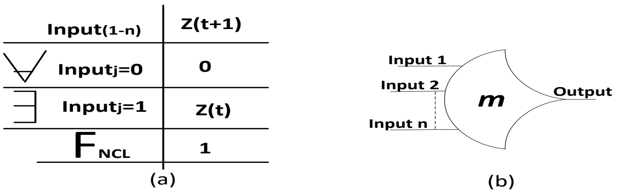


Fig. 5. *NCL gates: a) Table of operations; b) Symbol: THmn*

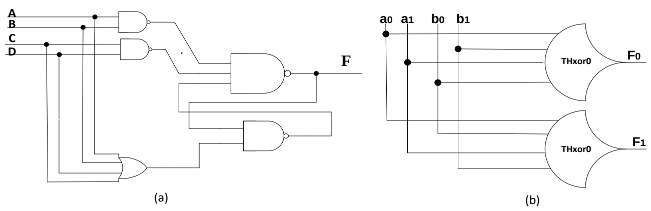


Fig.6. *a) THxor0 of [16]; b) THxor0 for an NCL XOR dual-rail gate.*

The operation complete detector (CD), like the input detector, is required in the QDI style. It consists of OR2 gates and C element of fan-in= N, where N is the number of signals. Figure 7 shows the CD circuit of a full adder.

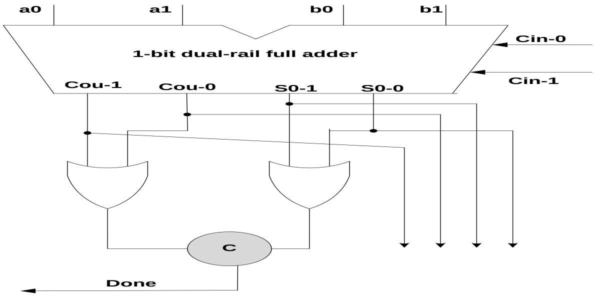


Fig. 7. *Operation completion detector: done signal.*

A simple approach to implementing Boolean functions with NCL gates starts from a minimized two-level function F_IT (independent of technology) and follows three steps:

1 Perform the conventional technology mapping of the F_IT function using only a basic gates library. The mapping is performed, for example, by the SIS [17] tool in the target library [NOT, AND2, OR2, XOR, XNOR, NAND2, NOR2, and AOI4] and to obtain the F_DT function (technology dependent).

2 Perform the dual-rail extension of each gate of the F-DT function, obtaining the F-DT-dual-rail.

3 Perform the trivial mapping of the F-DT-dual-rail, using a target library of seven dual-rail NCL gates.

IV. Case Study: Pipeline QDI ALU

Arithmetic Logic Unit (ALU) is an important functional component present in most digital systems. There is different design styles proposed for the synthesis of a single-rail ALU. To illustrate the proposed pipeline architecture, we apply it to an example found in [18], based on the 8-bit ALU of the 74181 TTL integrated circuit. Table I shows its Operations Table, with 12 operations, partitioned into three blocks (signals *M* and *C_0*).

TABLE I. TABLE OF OPERATIONS OF THE ALU

M=1		M=0	Selection
C_0=0	C_0=1	C_0=X	S_1S_0
A	A plus 1	A	00
$\overline{A}$	$\overline{A}$ plus 1	$\overline{A}$	01
A plus B	A plus B plus 1	A⊕B	10
$\overline{A}$ plus B	$\overline{A}$ plus B plus 1	A⊙B	11

Figure 8 shows the logical design of the 1-bit ALU, synthesized in [18]. Firstly, we must define the number of stages of the ULA, which in this case, will be two stages. Particularizing for the two-stage linear pipeline shown in Fig. 1, we have a single control (see Fig. 9).

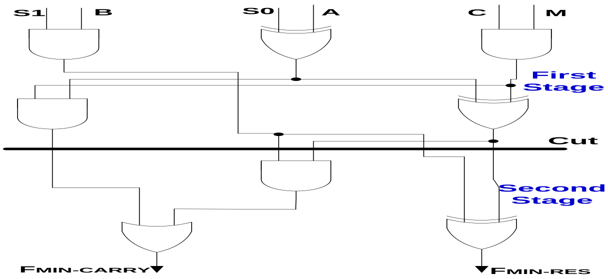


Fig. 8. *Logic Circuit: Multi-level 1-bit ALU [18].*

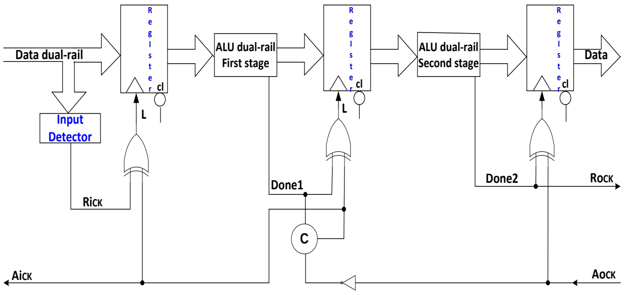


Fig. 9. *Proposed architecture: Two-stage Pipeline QDI Combinatorial System.*

A. Design of the pipeline QDI ALU

The QDI ALU design follows the approach proposed in [24]. One way to design QDI_CC circuits is to design the optimized conventional combinatorial circuit (single-rail), being dependent on the target technology (first step). The second step is to perform the mapping following a library involving gates of fan-in=2. In our example, the mapping is trivial, because all gates are already fan-in = 2, and then convert single-rail gates to dual-rail gates, which are synthesized by the DIMS or NCL, as shown in Section III, which is the third step. The next step defines the cutoff of the circuit when a balance in the critical paths is tried. Figure 10 shows the QDI ALU scheme pipeline of two stages, implemented in the target architecture, without the control, the input and output registers, and CD circuits.

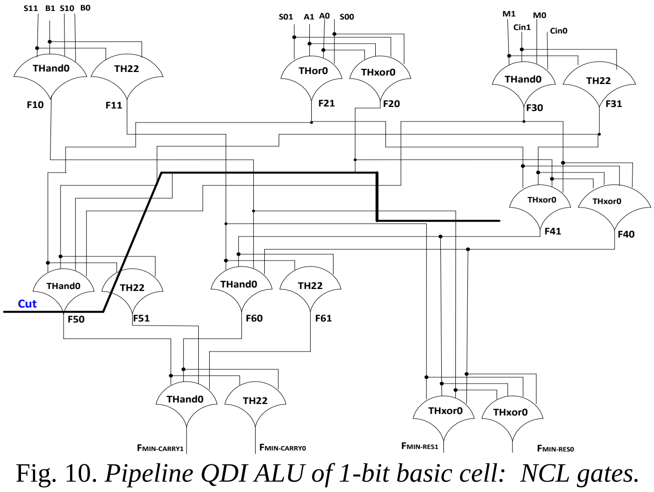


Fig. 10. *Pipeline QDI ALU of 1-bit basic cell: NCL gates.*

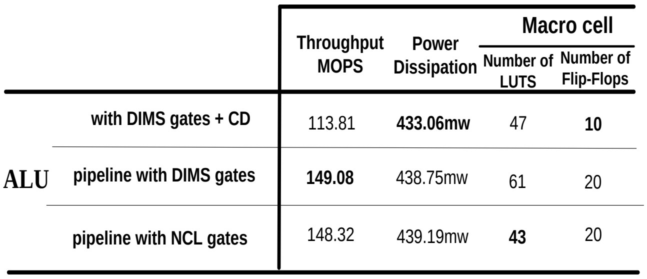
V. Experimental Results

The designs, in VHDL structural, non-pipeline QDI with DIMS gates and versions pipeline QDI with DIMS gates and NCL gates were compiled and simulated, post-mapping, in ALTERA tool, Quartus II software, version 9.0, Stratix III family, in EP3SE50F484C2 device.

A. Results of the ALUs in FPGA

Table II shows the results of area (LUTs + FFs), dissipated power, and throughput for the three projects of the ALU. Comparing the QDI DIMS ALU pipeline with QDI ALU shows an increase of 30.9% in the throughput. For the area (FFs + LUTs) and dissipated power, there is a penalty of 42.1% and 1.3%, respectively. When compared to the QDI NCL ALU pipeline with QDI ALU, there was an increase of 30.3% in the throughput. For the area (FFs + LUTs), there was a penalty of 10.5%, and for the dissipated power, there was a 1.4% penalty.

TABLE II. RESULTS OF THE ALUs

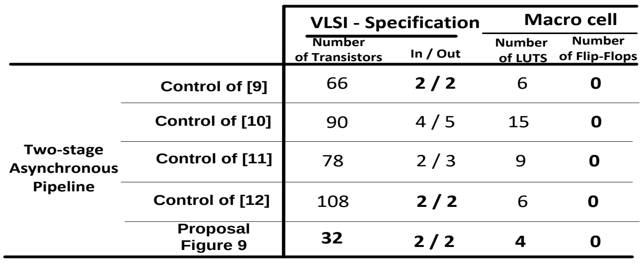


		Throughput MOPS	Power Dissipation	Macro cell	
				Number of LUTS	Number of Flip-Flops
ALU	with DIMS gates + CD	113.81	**433.06mw**	47	**10**
	pipeline with DIMS gates	**149.08**	438.75mw	61	20
	pipeline with NCL gates	148.32	439.19mw	**43**	20

B. Results of the Controls for the pipeline QDI_CC

Table III shows the results of the area obtained for three different controls found in literature, which are used in pipeline QDI_CC of two-stage. The C element uses 12 transistors in the static style [14]. Compared with those controls, the proposed control in the FPGA core obtained an average reduction in LUTs 55.6%, and in the VLSI core, it achieved an average reduction in transistors of 62.6%.

TABLE III. RESULTS OF THE CONTROLS



		VLSI - Specification		Macro cell	
		Number of Transistors	In / Out	Number of LUTS	Number of Flip-Flops
Two-stage Asynchronous Pipeline	Control of [9]	66	**2 / 2**	6	**0**
	Control of [10]	90	4 / 5	15	**0**
	Control of [11]	78	2 / 3	9	**0**
	Control of [12]	108	**2 / 2**	6	**0**
	Proposal Figure 9	**32**	**2 / 2**	**4**	**0**

VI. Conclusion

In this paper, we proposed the QDI linear pipeline architecture. Through a case study, we apply in the two-stage QDI pipeline a 1-bit ALU of 12 operations. A QDI pipeline is robust for data security applications once it hampers the analysis of the physical quantities. It is a technique used for finding the cryptographic key and therefore, to violate the data. A QDI pipeline presents other interesting properties, such as the robustness and variations in temperature and supply voltage, which often occurs in hostile environments like space and other military combat areas.

References

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

[2] D. Goldhaber-Gordon, et al., “Overview of Nanoelectronic Devices,” *Proc. of the IEEE*, vol. 85, No. 4, pp.521-540, April 1997.

[3] 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.

[4] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou, “Coping with the variability of combinational logic delays,” *ICCD*, pages 505–508, 2004.

[5] C. J., Myers, “*Asynchronous Circuit Design*”, Wiley & Sons, Inc., 2004, 2a edition.

[6] J. Martin, “Compiling Communication to Delay-Insensitive VLSI Circuits”, *Distributed Computing*, 1(4), pp.226-234, December 1986.

[7] J. Martin, “The Limitations to Delay Insensitive in Asynchronous Circuits,” 6th MIT Conference on Advanced Research in VLSI Processes, pp.263-277, 1990.

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

[9] D. Shang, et. al., “High-security asynchronous circuit implementation of AES”, *IEE Proc. Comput. Digit. Tech*. vol. 153, No. 2, pp.71-77, March, 2006.

[10] S. B. Furber and P. Day, “Four-Phase Micropipeline Latch Control Circuits,” *IEEE Trans. on VLSI Systems*, vol.4, no. 2, pp.247-253, June, 1996.

[11] R. Kol and R. Ginosar, “A Doubly-Latched Asynchronous Pipeline,” IEEE/ACM International Conference on Computer Design (ICCD’97), pp.706-713, 1997.

[12] G. S. Taylor and G. M. Blair, “Reduced Complexity Two-Phase Micropipeline Latch Controller,” *IEEE Journal of Solid-State Circuits*, vol. 33, Nro. 10, pp.1590-1593, October, 1998.

[13] D. L. Oliveira, et al., “Using FPGAs to Implement Asynchronous Pipeline,” 5th IEEE Latin American Symposium on Circuits and Systems, Santiago, Chile, 2014.

[14] 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, pp. 261-273, 1996.

[15] M. Y. Agyekum and S. M; Nowick, “An error-correcting unordered code and hardware support for robust asynchronous global communication.” In: DATE’11, pp. 765-770, 2011.

[16] D. L. Oliveira, et al. “Synthesis of QDI Combinational Circuits using Null Convention Logic Based on Basic Gates,” *Advances in Science, Technology and Engineering Systems Journal*, Vol. 3, No. 4, 308-317, 2018.

[17] E. Sentovich, et al., “SIS: System for Sequential Circuit Synthesis,” Tech. Rep. M92/41, Electronic Research Laboratory, College Engineering, University of California, Berkeley, 1992.

[18] Taub,H.“*CircuitosDigitaiseMicroprocessadores*”, Portuguese edition, McGraw-Hill, 1982.