2013-01-01

Asynchronous Logic Design With Increased Fault Tolerance And Optimized For Subthreshold Operation

Ivan Santos
University of Texas at El Paso, ivansantos1820@gmail.com

ASYNCHRONOUS LOGIC DESIGN WITH INCREASED FAULT TOLERANCE AND OPTIMIZED FOR SUBTHRESHOLD OPERATION

IVAN SANTOS

Department of Electrical and Computer Engineering

APPROVED:

Eric MacDonald, Ph.D., Chair

David A. Roberson, Ph.D.

John Moya, Ph.D.

Benjamin C. Flores, Ph.D.
Dean of the Graduate School

Dedication

This thesis is dedicated to my beloved family.

ASYNCHRONOUS LOGIC DESIGN WITH INCREASED FAULT TOLERANCE AND OPTIMIZED FOR SUBTHRESHOLD OPERATION

by

IVAN SANTOS, B. S. E. E.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

December 2013

Acknowledgements

I owe my most sincere gratitude to my graduate research advisor Dr. Eric Macdonald, who has not been only my mentor, but an appreciated model to follow. His enormous support, motivation and patience facilitated all the thesis work. I do not think there would be a better advisor for my Master's degree. His guidance and supervision represented the main key to expand my research experience, and his outstanding tutoring permitted me to acquire valuable skills that I will be implementing as a design engineer. Thank you Dr. MacDonald for all your care towards my young career! I will never forget all the things you have done for me, and I will always appreciate the opportunity of being part of your research group.

Additionally, I would like to thank Dr. David A. Roberson and Dr. John Moya for being part of my defense committee and for all the constructive feedback towards my thesis preparation. Likewise, I would like to extend my gratitude to the System Administrator of the Electrical and Computer Engineering department, Nito Gumataotao, who always helped me when I had troubles to manage all the software used in my research.

I would also like to thank my Multicore and DSP team at Texas Instruments, specially to Dr. Rama Venkatasubramanian, who gave the opportunity to expand my acknowledge in circuits design and power management under his supervision.

Finally but not the least important, I would like to recognize with my most sincere gratitude the enormous support that I received from my family throughout my academic career. Gracias papás –Lupe y Horacio– y hermano Néstor por su impecable apoyo hacia mí y por nunca dejarme caer. De la misma forma te agradezco a ti mi futura esposa Sara por aguantar mi ritmo, por ser mi inspiración y por tu soporte incondicional. ¡Los amo con todo mi corazón!

Abstract

Synchronization has always been an essential feature in electronic circuits, in which functionality is achieved by an appropriate flow of data with respect to time. An ideal concept of logic design would require every operation to happen instantly; however, the application of this principle to electronics is governed by the laws of physics, which influence the time to execute a process. In fact, the speed at which a microprocessor can perform logic operations is determined by the time that charging and discharging capacitive loads requires, and the time that electric signals spend when traveling through the circuitry. Consequently, the variation of capacitance over the circuit, and the disparity of wiring result in a disaccord at the time new data are ready. As a solution for these irregularities, a clock signal has been utilized as an element that provides synchronization on the flow of data. However, an exhaustive timing analysis is needed to determine the maximum clock frequency that can regulate the performance of each logic module in an electronic system. In addition, the clock signal typically represents around 30% of power consumption in synchronous circuits and introduces electromagnetic interference (EMI) due to the rapid fluctuation of electric current flow. For this reason, exclusion of a clock signal allows asynchronous logic to be suitable for low power electronics and circuits that demand noise stability. Moreover, this thesis explores two additional fields that complement asynchronous logic to create a more robust technology that consumes less power – more appropriate for biomedical electronics and space applications. The explored fields grasp fault tolerant schemes that decrease the vulnerability to radiation and strategies to design logic apt to operate in a subthreshold regime (0.3 V). Thus, the main objective of the presented work is to introduce a strategy that combines 1) the benefits of asynchronous circuits, 2) the reduced power

consumption achieved at a subthreshold operation, and 3) the radiation hardening that a fault tolerant scheme offers. Specifically, this strategy was implemented on an existing asynchronous system referred to as Null Convention Logic™ in order to not only reduce energy, but also to increase the fault tolerance on logic gates that constitute the library of this asynchronous approach.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The *clock* signal was introduced as an approach to establish synchronization in a logic circuit. Subsequently, electronic systems started to operate on a synchronized mode, which permits the computation of data at a fixed frequency. Furthermore, the performance of synchronous electronic systems relies on a worst-case scenario dictated by the maximum time required to process new data. This maximum time arises from the addition of 1) the critical path in the circuit, which is the longest path that data has to travel, 2) the largest clock skew, which is the variation in time of the clock signal arrival to different logic modules, and 3) an aggregated protection time, which deals with Process, Voltage, and Temperature (PVT) variations. Since an electronic circuit is constituted by different *logic modules,* which have distinct arrangements of transistors, the time for each to compute data varies. Thus, the perfect electronic circuit would have multiple logic modules that could compute data exactly at the same time. If this ideal circuit would exist, then determining the oscillation frequency of clock signals in the circuit would be uncomplicated. This frequency would be set to a greater value than the time needed to compute new data, which would be the same for all the modules. Furthermore, there would not be any discrepancy between the time at which new data is ready and the time at which this data is needed – meaning that positive slack (i.e., data is being computed and ready before the time needed) and negative slack (i.e., new data is ready after the time needed) would be avoided. Unfortunately, this optimal scenario is hard to accomplish due to the physical variations in electronic circuits [1]. Some instances of these physical variations are temperature deviations, process irregularities, and voltage fluctuation. Moreover, with the shrinking scale of current technologies, having an optimal clock distribution to spread signals uniformly among the synchronous circuit represents a big challenge [2-6].

Asynchronous logic design was introduced as an approach to compute and use data as needed [7-8]. In asynchronous circuits the global clock signal that provides synchronization in the flow of data is replaced by *handshaking signals* that communicate the request and availability

of new data. A general protocol of this asynchronous setting is based in the implementation of *request data signals*, which are sent by a logic module that needs new data to be processed, and *acknowledge data signals*, which are sent by the logic module that serves the requested data. Figure 1.1 compares an acknowledgment signals format with a traditional clock signal setup. In this way, the prevention of positive and negative slacks is accomplished without the necessity of having logic modules with the same performance. Then, an exhaust timing analysis is no longer needed to evade logical errors. Moreover, as discussed in [9] the absence of timing restrictions reduces the negative effects of PVT deviations, which alters the performance and functionality of a logic system.

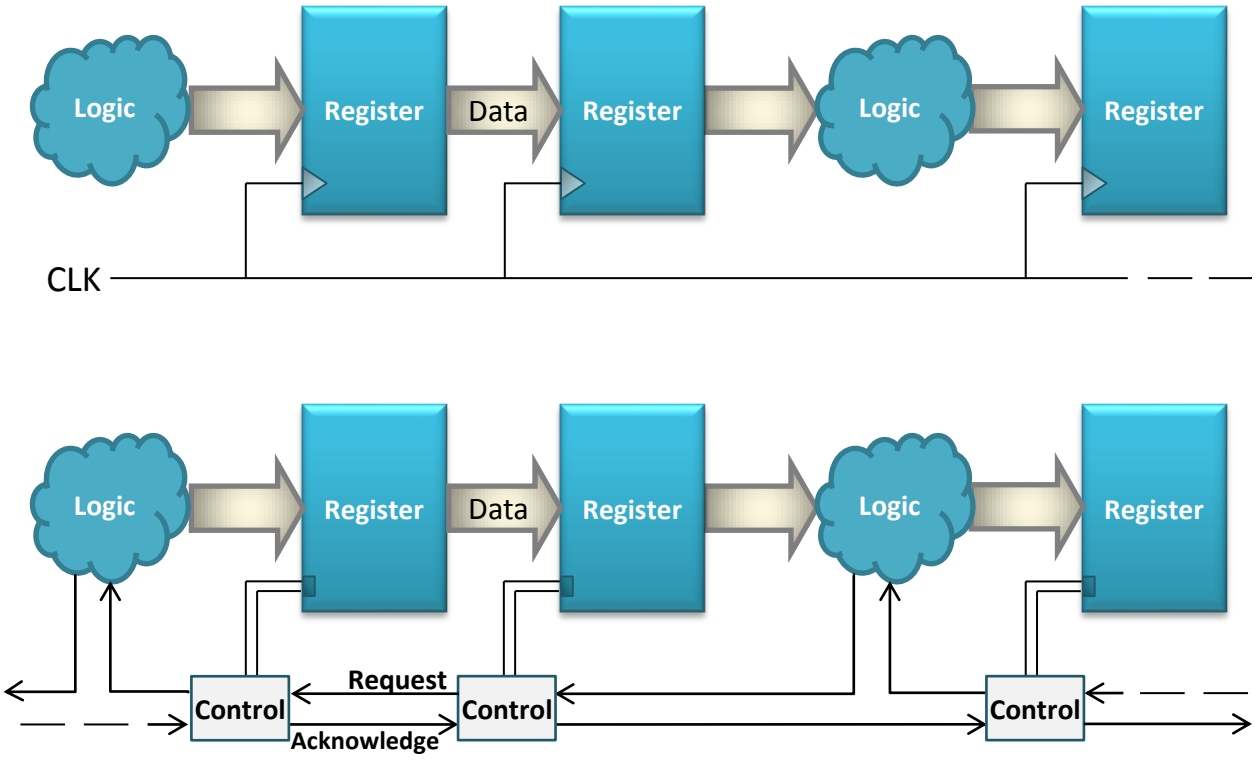


Figure 1.1: Clock signal structure (top) vs. acknowledgement signals setup (bottom).

Selected approaches for designing asynchronous systems have been successfully introduced in the last decades [10-19]. Likewise, the transfer of current synchronous circuits to an asynchronous configuration is becoming more common due to the pragmatic performance that

current asynchronous methodologies offer. Accordingly, one of the main reasons for which a circuit is considered to be designed in an asynchronous regime, beyond power consumption and speed, is the guaranteed correct functionality in presence of PVT disparities. An asynchronous circuit compromises a more accurate operation than its synchronous version while operating in unstable environments; however, this precise job does not always represent a better performance (i.e., logic operations per time), much less an improvement in the wasted energy. Thus, designing asynchronous circuits embodies the analysis of their applications when saving power and/or increasing the number of operations per time are considered primary factors. For this reason, passive applications, which are active for a short time and idle the majority of the time, are very suitable to be operated by asynchronous circuits. Regardless of this condition, an optimal approach for improving the power consumption on active applications that are operated by asynchronous circuits is the reduction of the voltage that activates the logic system. More precisely, pushing a system to work in a subthreshold regime with less than 0.45V reduces the energy consumption [20-23]. This practice is possible thanks to current technologies, such as the MIT Lincoln Labs 150nm XLP CMOS process, which are optimized to be implemented in circuits that require extremely low voltage for their operation [24]. Even with these optimized technologies, lowering the voltage comes with some negative consequences like the introduction of higher delay variations in the data flow, which characterize one of the main causes of soft errors in electronic circuits. Since asynchronous circuits are not operationally affected by delay variations, the implementation of this subthreshold practice is not a condition that might cause the asynchronous system to fail in a regular environment. Another undesirable consequence of working at a subthreshold voltage level is the increased vulnerability to radiation [25]. The amount of charge adversely induced by radiation that flips a logical value in a logical module is known as *critical charge* – Q_{crit} –. This critical charge is proportional to the voltage applied to the circuit. Thus, when the voltage is lowered the critical charge diminishes. As a result, asynchronous circuits that work in a subthreshold regime require reinforcement when they are exposed to aggressive environments with high levels of particle emissions. Even though the

resilient nature of asynchronous logic is contemplated for the conception of biomedical and space applications, asynchronous circuits must not fail in presence of particle strikes. For instance, a satellite that is exposed to high doses of radiation in space must be operated by robust electronics that can handle such high radioactivity. Similarly, a pacemaker, which is under-skin implanted, must not stop working in presence of Electromagnetic Interference (EMI) that might be produced by daily-used electronics like smart-phones, personal computers, global positioning systems, remote controllers, just to mention a few. One of the techniques to increase the fault tolerance of circuits is by design. Specifically, logical redundancy in circuits accompanied by strategic interweaving configurations increases the critical charge needed to corrupt the logic of the circuit.

The research presented in this paper explores the aforementioned areas: asynchronous logic, subthreshold operation, and increased fault tolerance design; introducing a new methodology that can be adopted by low power applications that are controlled by asynchronous circuits in order to increase its robustness while operating at a low level voltage. Respectively, this novel approach was employed for the design of a logic gate library corresponding to an asynchronous scheme called NULL Convention Logic^{TM} managing the correct operability from 1.5 V (superthreshold voltage) to 0.3 V (subthreshold voltage) and enhancing the vulnerability to radiation by increasing the Q_{crit} at 0.3 V in average 16 times better than the Q_{crit} at 1.5 V.

The remainder of this thesis is organized as follows. An overview and fundamental definitions of asynchronous logic, radiation hardening, and subthreshold operation is presented in Chapter 2. Previous approaches to increase the fault tolerance of electronic circuits, optimization of circuits to operate in low voltage levels, and advancements on asynchronous technologies to support these features are described in Chapter 3. The novel methodology for asynchronous circuit design with fault tolerance and optimized for subthreshold operation is explained in Chapter 4. The setup for simulations and results are discussed in Chapter 5. Finally, Chapter 6 provides a conclusion and future work for the presented research.

Chapter 2: Background

2.1 Asynchronous Circuits

Asynchronous design is a nascent discipline, and a straightforward way to learn this philosophy has not been instituted. In fact, due to the diversity of terminologies that different authors use to describe this subject, the comprehension of asynchronous logic results in a complicated process. Distinct methodologies that use different structures and symbolic elements diminish the possibility of establishing a single procedure to design asynchronous circuits. These approaches may seem different at first glance; however, a similar principle is enclosed by their logic configuration.

As the name implies the description by itself, asynchronous circuits lack of synchrony in data management. Every module that composes an electronic system computes data at a different rate; however, since asynchronous logic requires no time limits to compute data, timing conflicts cannot surge from this logic. Having no clock signal allows the electronic system to freely operate as rapidly as possible. Still, this might result in a positive improvement if the circuit operates faster than in a synchronous environment (e.g., an electronic circuit that has most of its logic modules producing positive slacks), or in the worst case asynchronous conditions might result in a slow circuit if this was reporting several negative slacks in synchronous operation. An asynchronous designer must comprehend completely all the properties of asynchronous logic and analyze the electrical behavior of a logic system before trying to translate the configuration to an asynchronous regime. As SparsØ and Furber said: *"asynchronous techniques may not always be the right solution to the problem"* [10]. Consequently, considering all the tradeoffs that switching to an asynchronous mode implies is crucial. In the absence of a clock signal, what would be the best approach for achieving the flow of data? Is the time to compute data not considered at all? Is asynchronous logic free of stalls or hazards? If so, how are these logic issues managed? Can asynchronous logic be applied to any electric circuit? These and more questions are considered in ongoing research for the development of asynchronous logic methodologies.

2.1.1 Classification of Asynchronous Circuits

Classifying logic circuits according to the rationality to determine their output, results in two main categories: combinational circuits and sequential circuits. The output of combinational circuits is dependent on the present combination of inputs solely. On the other hand, the output of sequential circuits depends not only on the current arrangement of inputs, but on the previous output state originating output data retention – *memory* –.

Most of the forms of asynchronous circuits fall into the category of sequential circuits. Furthermore, from a gate level perspective asynchronous circuits can be classified as *"Speed-Independent (SI)"*, *"Delay-Insensitive (DI)"*, or *"Quasi-Delay-Insensitive (QDI)"* according to delay expectations that are assumed in order to achieve logic coherence and functional operation [10, 18]. Figure 2.1 shows a diagram for the asynchronous circuits' classification.

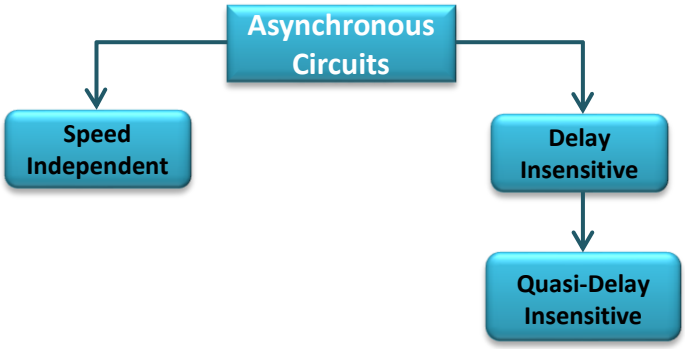


Figure 2.1: Classification of asynchronous circuits.

Speed Independent (SI) Circuits

The correct operation of a *Speed Independent (SI)* circuit is based in the assumption of unknown bounded delays in logic gates that comprise the circuit, and the assumption of zero-delay in wires that connect the logic gates. Even so, this last conjecture is not a realistic factor in existing semiconductor technologies, which are influenced by the laws of physics.

Delay Insensitive (DI) Circuits

An asynchronous circuit falls into the *Delay Insensitive (DI)* category if the functionality of the circuit is completely independent of the wiring delays and the delays of logic modules.

The pure design of this type of asynchronous circuits expresses a restrictive organization of the logic structure. In other words, the strategies to formulate the logic of these circuits look for the total avoidance of conflicts generated by delay variations and consider unbounded wire and gate delays, which exemplify an unknown data-flow scenario with respect to time. Delay-Insensitive is considered the most robust asynchronous logic because its convention does not tolerate that the functionality of asynchronous systems could be risked by assuming negligible delays. In fact, the development of asynchronous systems that are governed by a pure Delay Insensitive logic is limited to the usage of simple inverters and Muller C-elements for their design as described in [26]. C-elements not only avoid transitions to illegal states, but also ensure *observability* (more details of C-elements are presented later in this section and the property of observability is described in section 2.1.2). Nonetheless, this configuration needs to be accompanied by the execution of a rigid *handshaking protocol* (also known as *completion detection circuitry*) that ensures the correct flow of data in the system [11].

- ***Handshaking protocol***

The implementation of handshaking signals prevents confusion in asynchronous registers between old and new data. Hence, a correct flow of data is achieved at a register transfer level (RTL) [10]. Logic modules can be considered as *sender* modules or *receiver* modules depending on which task are currently processing. For instance, if logic module *"A"* is computing data that was requested by logic module *"B"*, then module *"A"* will eventually send that new data becoming the sender module and module *"B"* will receive that data becoming the receiver module. For this same scenario, if logic module *"A"*, which is the sender module, requires some data from another module *"C"*, then module *"A"* could be also considered as a receiver module because it will eventually receive data from the sender module *"C"*. Moreover, a logic module is forced to wait for an acknowledgement signal in order to proceed; i.e., a sender module that has just sent data will have to wait for a new request signal in order to compute new data and respond to the request. In the same way, a receiver module has to wait for a reply signal that will

acknowledge the receiver that new data is available and ready to be used. Figure 2.2 illustrates the basic flowchart of a logic module that can be considered either as a *sender* or a *receiver* depending on its current state of the flow (the doted arrows denotes when a logic module switches from a *sender mode* to a *receiver mode* or vice versa).

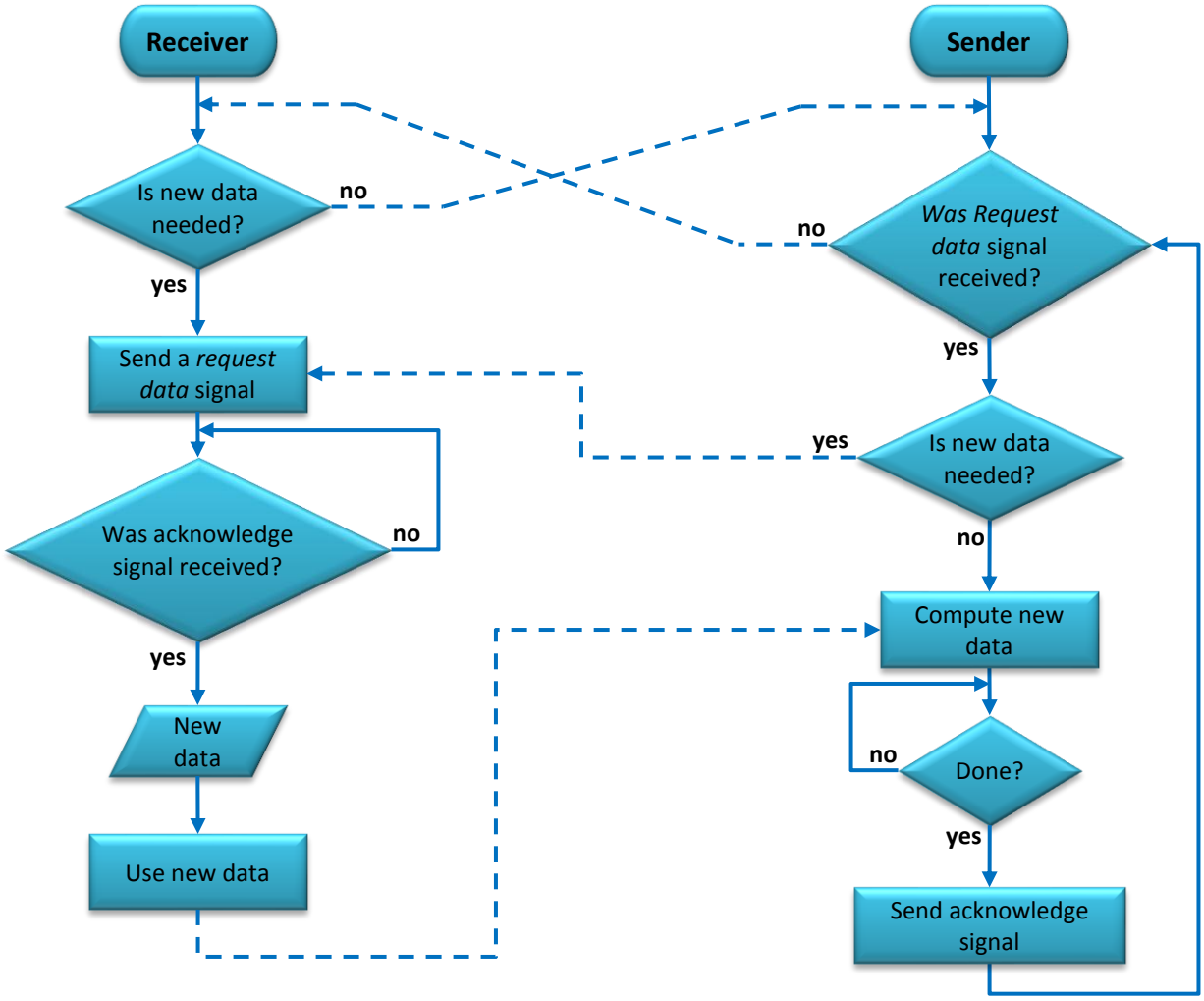


Figure 2.2: Basic flowchart of a logic module in a handshaking protocol.

When a sender module is the one that starts a communication transaction, the handshaking class is considered as *push-channel* [10]. In this category, the sender notifies the receiver about new data by asserting a request signal, then the receiver responds with the assertion of an acknowledge signal that communicates the availability to receive data and starts

collecting the new data. On the other hand, when a receiver module starts the communication the handshaking style is referred to as *pull-channel* [10]. In this case, the receiver requests the sender to compute and send new data, to which the sender responds with the assertion of an acknowledge signal as soon as new data is ready and then the sender starts sending this new data. Both handshaking categories can adopt a *four-phase* or *two-phase* progression.

- ***Four-phase Handshaking***

A handshaking protocol is referred to as *four-phase handshaking* if the communication between two logic modules flows as next described: 1) receiver sets the *request signal* high when new data is needed; 2) sender sets the *acknowledge signal* high and sends the requested data; 3) receiver consumes the transmitted data and unsets the request signal when finishes; 4) sender stops sending data when perceiving that the *request signal* has transitioned back to low, and then unsets the *acknowledge signal* [10]. This protocol is also referred to as *Return-To-Zero (RTZ)* protocol by some authors because physically the initial voltage level of the handshaking signals is usually 0, and at the end of the handshaking communication both signals return to this zero voltage level [13, 18].

- ***Two-phase Handshaking***

A handshaking protocol is considered a *two-phase handshaking* if the communication progress as next: 1) receiver notifies the sender about the necessity of new data by switching the *request signal*, which will transition from low-to-high, or from high-to-low depending on its last state, so now the sender, which was sending previously asked data, will start working on the calculation of new data; 2) sender switches the *acknowledge signal* when new data is available and send it, so the receiver simply stars consuming this new data. [10, 11]. Following this protocol, the structure of logic combinational gates that constitute recent Delay Insensitive libraries is designed in such a way that logic gates can hold a logical value – *feedback property* – (e.g., NULL Convention Logic™ library [14, 16]), causing these gates to act as sequential elements as well. For instance, the logic configuration of these gates obliges all the inputs to

transition back to a *no-data* state in order to transition their output to a *non-asserted* state too. This type of gates are usually referred to as *Muller C-elements* [7, 8], and they are fundamental to avoid logical hazards in the asynchronous design (a more detailed description of C-elements is covered later in this section). Figure 2.3 shows the waveforms of acknowledgement signals for a four-phase and a two-phase protocol [10].

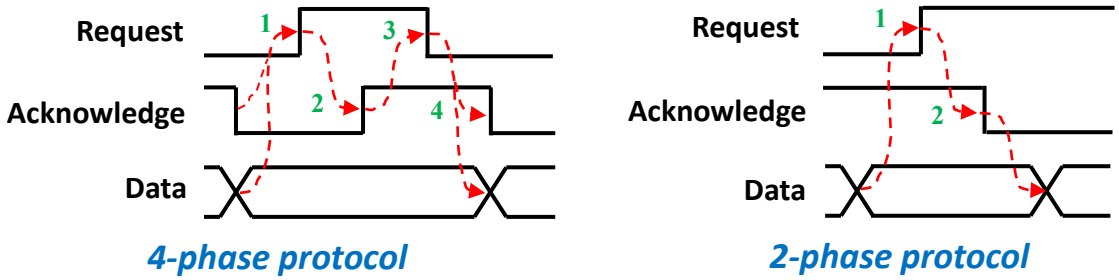


Figure 2.3: Four-phase protocol (left) vs. two-phase protocol (right) waveforms [10].

- ***Dual-rail and Quad-rail configuration***

Since an unbounded Delay Insensitive (DI) configuration system that uses a completion detection protocol cannot differentiate between an input that has been delayed and an input that has kept the same value for a consecutive calculation, a single data bit cannot be represented with a single wire [11]. Instead, two wires – *dual-rail* –, are used to represent a single data bit (e.g., d ⊢ {d.t, d.f}); or four wires – *quad-rail* –, are used to represent two data bits (e.g., {d1, d0} ⊢ {d.3, d.2, d.1, d.0}). For instance, a single bit can have only two logic values: 0 (*logical zero*) or 1 (*logical one*), so in a dual-rail configuration one wire will be used to represent the *value 0* or a *false state* (e.g., *d.f*), and another wire will be used to represent the *value 1* or a *true state* (e.g., *d.t*). In this way, when the wire that represents the value 0 is high, a logical 0 is transmitted, and if the wire is low then no 0 value is transmitted. The same principle applies for the wire that represents the value 1, when this is high then a logical 1 is transmitted. Considering that in a synchronous circuit a wire that symbolizes a logical 0 and a logical 1, cannot carry these two values at the same time, then in a dual-rail or quad rail configuration, no more than one wire can be high at the same time because the mutually exclusivity must be preserved for a correct

data flow. If two or more wires are high simultaneously, then a logical hazard will be introduced. In contrast, if none of the wires are high means that no data is being transmitted, and this case can be treated as a *spacer*, *empty* state, *NULL* state, or simply as a *no-data* state. Thus, any logic design methodology for Delay Insensitive circuits has to consider the way to detect and avoid illegal states, in which logical hazards might arise, and the mode to manage the no-data states as transitions in data. Otherwise, a simple delay in the circuitry will provoke malfunctions in the asynchronous system. Figure 2.4 shows the combination of wire states to represent data for a dual-rail and quad-rail configuration.

Dual-rail		
d.t	**d.f**	**data bit**
0	0	no-data
0	1	0
1	0	1
1	1	illegal

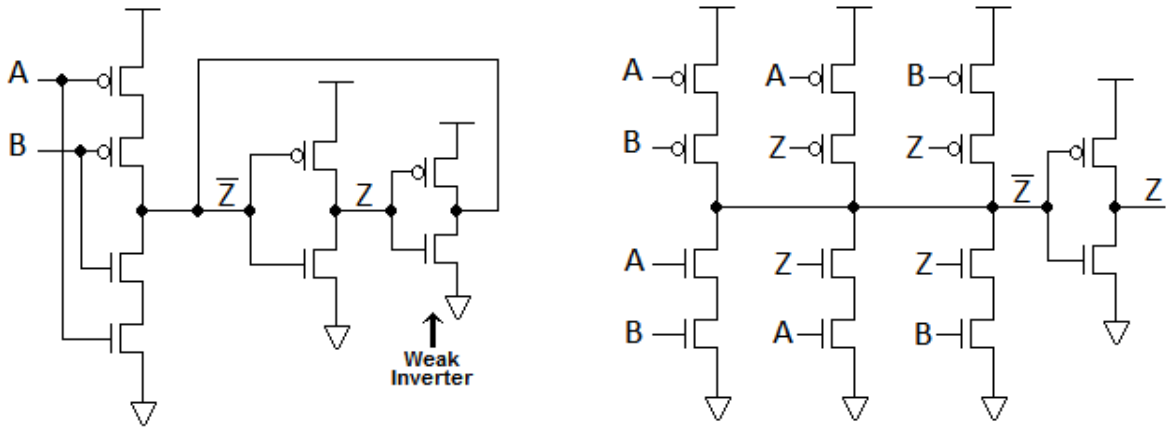
Quad-rail					
d.3	**d.2**	**d.1**	**d.0**	**Data bit 1**	**Data bit 0**
0	0	0	0	no-data	
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
1	1	1	1	illegal	

Figure 2.4: Data representation of dual-rail and quad-rail configurations.

- **Muller C-element**

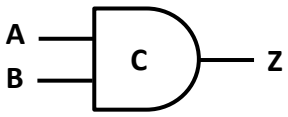
One of the most fundamental units in asynchronous architectures is the Muller C-element [7]. Due to its property of feedback (providing the ability to maintain a logical state), the C-element is considered as a sequential circuit. The main characteristic of this logic module is a comparison operation, which enhances the property of complete *observability* on logic gates (more details of observability are described in section 2.1.2). In order to distinguish the essential usage of a C-element in comparison to traditional Boolean logic gates as described in [10], consider the Boolean logic of a two-input AND gate. The output of an AND gate is asserted to a logical 1 only when both inputs are 1, otherwise the output is 0. Therefore, an *observer* (e.g., "gate B" is an observer of "gate A" if the output of "gate A" drives one of the inputs of

"gate B"), sensing the transition of this output from 0 to 1 might conclude that both inputs of that AND gate were asserted to a logical 1; however, when sensing a transition from 0 to 1, the observer cannot infer that both inputs changed to a 0 state, because this situation might be the case in which only one of the inputs provoked the transition. Thus, the AND gate is said to be solely an indicator of when two inputs are 1. In contrast, the output of an OR gate only indicates the certainty of a logical 0 in all its inputs when transitioning from 1 to 0. In distinction of these gates, the output of a C-element comprehends both indications. A basic C-element has two inputs (e.g., *"A"*, *"B"*) that when they are identical the output of this C-element (e.g., *"Z"*) is set to the agreed value. Alternatively, if both inputs have different values, then the output holds the previous state (i.e., no-change in the output). Consequently, an observer of a C-element can infer both cases: when both inputs are 0, and when both inputs are 1. Furthermore, the most basic CMOS arrangement for a C-element is implemented with a total of eight transistors, from which two P-FETs in series constitute the *pull-up* network, two N-FETs in series represent the *pull-down* network, and four additional transistors are used in a loop chain of two inverters. The output of the first inverter provides the main output of the C-element and drives the input of the second inverter as well. The second inverter provides feedback to the first inverter completing the feedback necessary to maintain the state when the inputs of the C-element are different. The second inverter must be weaker in terms of drive strength to reduce the contention when transitioning between states. Figure 2.5a illustrates the schematic of this basic C-element. Another variation of the gate exists with a larger transistor count relative to the original (14 vs. 8 transistors) and this implementation is preferred as the structure results in less contention electrical current. This structure has three pairs of series P-FETs in parallel that form the pull-up network, and the pull-down network has three pairs of series N-FETs in parallel. Figure 2.5b illustrates the CMOS configuration of this second version of the C-element, Figure 2.5c represents its truth table, and Figure 2.5d illustrates the general C-element symbol.



A	B	Output
0	0	0
0	1	Hold output
1	0	Hold output
1	1	1

c) C-element truth table



d) C-element symbol

Figure 2.5: CMOS structures, truth table, and symbol of a Muller C-element.

Quasi-Delay-Insensitive (QDI) Circuits

A variation on the logic of Delay-Insensitive circuits was introduced with the name of *Quasi-Delay-Insensitive (QDI)* logic, and adopts the exclusive assumption of equality on wire delays – *isochronic forks* –, which are explained in the next sub-section. This concept is much more feasible to be implemented for designing asynchronous circuits in real practice. QDI is usually considered a subdivision of Delay Insensitive logic because the consideration of unbounded gate and wire delays on circuits is preserved.

- **Isochronic Forks**

The output of a logic gate can drive a certain number of inputs of other gates. The term *'fanout'* of a gate denotes this number. Then, a *'wire fork'* or simply a *'fork'* is an output wire that feeds more than one logic element, so this corresponds to the physical translation of fanout, which is divided in separate wires to carry out the signal to different destinations. This endpoints are physically separated in a circuit board, thus the delay in the separated wires that compose a

fork is usually different depending on their length. In spite of this, an *isochronic fork* is assumed to have the same delay for each of its wires even the length of these are different [26]. For instance, if a fork coming from element O feds elements A, B, and C, and the element A, which is closer to the origin of the fork, senses a transition on the signal first than the rest of the elements, the isochronic assumption allows to infer that this transition was also sensed by elements B and C, even though these elements require more time to detect the signal transition. The assumption of isochronic forks in asynchronous circuits is what subdivides QDI from DI circuits. Allowing assumptions on wire-delays is a practice that debilities the robustness of DI circuits; however, isochronic forks ensure an acknowledgment event that allows an uncorrupted flow of data by forwarding the completion of signals in advance. This concept is illustrated in Figure 2.6.

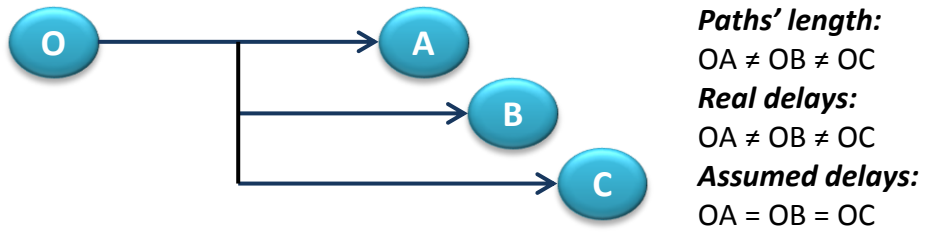


Figure 2.6: Graphical representation of a wire fork with isochronic assumption.

2.1.2 NULL Convention Logic (NCL)

A *symbolically complete expression* is a type of logic that 1) has no relationships with time, 2) is insensitive to the propagation time of data among the logic elements, and 3) expresses the transformation of data and control in a single expression [14]. Pure Boolean logic is not considered symbolically complete expressive since implementing Boolean logic in circuits involves the usage of an external control to coordinate the validation of data (i.e., knowing the moment to consider data as valid).

A mistake that existent approaches for designing Delay Insensitive circuits commit is the attempt to eliminate time dependencies by implementing a simple Boolean logic context surrounded of Muller C-elements, which might result in a complicated process and an expensive

implementation. In contrast, as stated in [14], NULL Convention Logic™ is a theoretically complete and economically feasible approach to DI circuits.

NCL avoids the usage of pure Boolean logic, but instead implements a modified symbolically complete Boolean logic. As described in [14], in order to attain a *symbolically complete expression*, both control and data conversion must be expressed in a mutually exclusive value assertion domain. For example, the primitive value assertion domain of Boolean logic relies in the mutual exclusivity of the values True and False, i.e., the output of a Boolean logic gate is either False or True, but not both simultaneously. However, these two values only represent data, but not a control value. Thus, NCL introduces a third assertion value that represents the opposite of data: no-data, which is called a *NULL* value and is used as a control value to invalidate data. Figure 2.7 shows the truth tables of a two-input Boolean logic AND gate, two-input OR gate, and a NOT gate with the addition of this third NULL state.

AND	True	False	NULL
True	True	False	NULL
False	False	False	NULL
NULL	NULL	NULL	NULL

OR	True	False	NULL
True	True	True	NULL
False	True	False	NULL
NULL	NULL	NULL	NULL

NOT	
True	False
False	True
NULL	NULL

Figure 2.7: Truth tables of AND, OR and NOT gates with NULL state.

Having a NULL state, a new completeness of input criteria is introduced to logic gates: an output value is only asserted to data when all the inputs are asserted to data, otherwise, a NULL value is asserted meaning no-data for the output. Considering Figure 2.8, which represents a conventional logic system with input signals *a*, *b*, *c*, *d*, and *e*, and output signals *j* and *k*, the completeness of input criteria is performed as follows: 1) circuit arrangement as a whole starts a logic sequence in a NULL state, which means that all the signals are transmitting *no-data*; 2) then assuming that all the signals but *'e'* changed to a valid DATA state, the gates *'A'* and *'B'* will assert a DATA state that will propagate through gates *'C'* and *'E'* allowing the output signal *'j'* to become DATA; in this case the entire circuit remains in a NULL state,

because the input signal *'e'* is not driving data, and therefore the output signal *'k'* keeps its NULL value; 3) when signal *'e'* transitions to a valid DATA state, this will be reflected at the output signal *'k'*, thus now the complete circuit will change to a DATA state. Since all the inputs were asserted to DATA in this example, the *input completeness* of the circuit was carried out, and now the outputs will be transmitting valid DATA to other logic modules in the system until all the inputs go back to a NULL state resetting the system for starting a new cycle. Hence, as Karl and Scott defined in [14], *"the circuit indicates its own completion of resolution, autonomously and purely symbolically."*

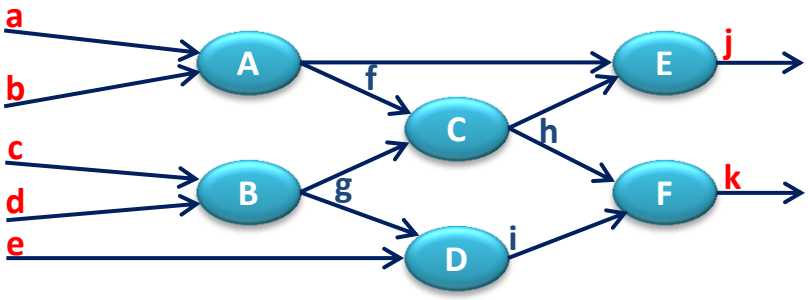


Figure 2.8: Diagram example of a logic system.

The input completeness principle is also applied to data asserted output signals, which are required to transition to NULL values only when all the inputs transition back to a NULL state. For instance, a case in which the output of a gate was recently asserted to DATA, and then all the inputs start going back to NULL with exception of one input that keeps a valid DATA state, the gate will be holding DATA strictly until the remaining input asserts a NULL value. In order to incorporate the property of input completeness into traditional Boolean logic, the Boolean logic gates need to be modified with the integration of feedback to their logic structure. Figure 2.9 represents a symbolic structure of a gate with feedback, which introduces an auto-dependency to its logic, referred to as *hysteresis*. That is, the output signal of a gate becomes an input to the gate itself simultaneously causing the next logic state of a gate to be dependent on the present state, and giving rise to the capability of holding states. Thus, this principle suggests the combination

of sequential logic with combinational logic in a single structure that adopts the coordination of data validation, as in the case of *threshold gates* or commonly referred to as *NCL gates*.

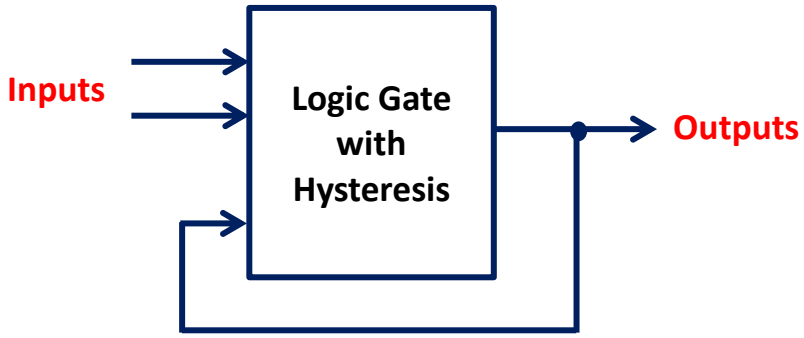


Figure 2.9: Symbolic structure of logic gate with feedback.

Table 2.1: List of the 27 fundamental NCL gates and their set equations.

	NCL gate	Boolean equation (Set equation)
1	**TH12**	A + B
2	**TH22**	AB
3	**TH13**	A + B + C
4	**TH23**	AB + AC + BC
5	**TH33**	ABC
6	**TH23w2**	A + BC
7	**TH33w2**	AB + AC
8	**TH14**	A + B + C + D
9	**TH24**	AB + AC + AD + BC + BD + CD
10	**TH34**	ABC + ABD + ACD + BCD
11	**TH44**	ABCD
12	**TH24w2**	A + BC + BD + CD
13	**TH34w2**	AB + AC + AD + BCD
14	**TH44w2**	ABC + ABD + ACD
15	**TH34w3**	A + BCD
16	**TH44w3**	AB + AC + AD
17	**TH24w22**	A + B + CD
18	**TH34w22**	AB + AC + AD + BC + BD
19	**TH44w22**	AB + ACD + BCD
20	**TH54w22**	ABC + ABD
21	**TH34w32**	A + BC + BD
22	**TH54w32**	AB + ACD
23	**TH44w322**	AB + AC + AD + BC
24	**TH54w322**	AB + AC + BCD
25	**THxor0**	AB + CD
26	**THand0**	AB + BC + AD
27	**TH24comp**	AC + BC + AD + BD

Threshold Gates

NCL circuits can be structured with 27 fundamental *threshold gates* that constitute a set of the most commonly used Boolean equations that have four or fewer variables [15, 16]. Table 2.1 shows the list of these 27 NCL gates and their respective Boolean equation.

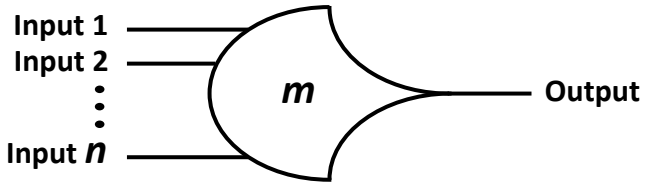


Figure 2.10: Symbol of a NCL TH*mn* threshold gate.

The main type of threshold gates is the *THmn gate*, where $1 \leq m \leq n$. Figure 2.10 represents the symbol of a THmn gate. The letter '*m'* that is written inside the symbol denotes the minimum number of inputs from a total of n inputs that need to be asserted in order to assert the output. For example, a TH23 gate has $n = 3$ inputs from which at least $m = 2$ inputs must be asserted to assert the output. Assuming that this TH23 gate has inputs A, B, C and output Z, then the corresponding set equation would be Z = AB + AC + BC. Another type of threshold gate is the *'weighted threshold gate'*, which is characterized as THmnW$w_1w_2\ldots w_R$, $1 \leq R < n$, and $1 < w_R \leq m$, where W denotes that the gate is a weighted threshold gate, and w_1, w_2, and w_R denote the weight of $input_1$, $input_2$, and $input_R$, respectively. This gate follows the same principle of a regular THmn gate; however the difference is that now some of its inputs have an integer weight $w > 1$ that mimics a repetitive number of a particular input in the gate. For instance, a TH24W22 gate has $n= 4$ inputs from which at least $m = 2$ inputs must be asserted to assert the output; however, since $w_1 = 2$ and $w_2 = 2$, then the only assertion of $input_1$ or $input_2$, which have an integer weight of 2, will meet the criteria of 2 out 4 input assertion and provoke the assertion of the output. Assuming that this TH24W22 gate has inputs A, B, C, D, and output Z, then the set equation would be Z = A + B + CD. Figure 2.11 shows the symbol of this TH24W22 gate and its corresponding set equation.

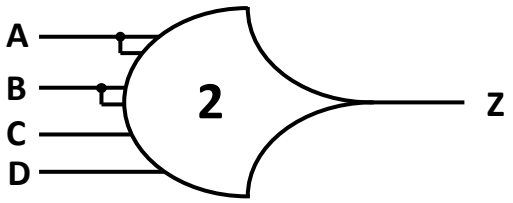


Figure 2.11: Symbol of a TH24W22 weighted threshold gate.

- **Transistor Level Structure**

Threshold gates have a similar arrangement to the structure of C-elements, which encloses the property of hysteresis. In fact, NCL gates are sometimes considered in literature as modified C-elements. The structure of NCL gates ensure a complete transition of all the inputs back to a NULL state before continuing with new output assertion corresponding to new incoming data [15, 16]. Thus, the general transistor-level structure of a threshold gate achieves the input completeness with hysteresis by enclosing these next four equations: 1) *Set equation* – this represents the traditional Boolean logic equation that dictates the main logic to assert a *true state* in the output; 2) *Reset equation* – this is used for initializing the output to a *false state*, which is denoted with the letter *'n'* in the symbol of the NCL gate, or to a *true state*, which is denoted with the letter *'d'* in the symbol; 3) *Hold-1 equation* preserves a *true state* in the output and is constituted by 'ORing' all the inputs, so the hold-1 equation characterizes the complement of the reset equation; 4) *Hold-0 equation* characterizes the complement of the set equation and preserves a *false state* in the output of the gate. Furthermore, these holding equations are complemented with a feedback signal coming from the output in order to achieve hysteresis; so the logic of these equations depends on the previous output state, which is 'ANDed' to this logic. Besides, the logic of NCL gates is *'non-inverting'*, which means that in order to counteract the *'inverting'* nature of CMOS design a simple inverter drives the output signal. For example, consider a TH23 gate with inputs A, B, C, and output Z, its set equation would be $Z = AB + AC + BC$; its reset equation would be all the inputs 'ANDed' together, i.e., ABC; its hold-1 equation would be all the inputs 'ORed' together and ANDed with the previous output state Z^{-}, i.e., $Z^{-}(A + B + C)$; finally, its hold-0 equation would be the complement of the set

equation Z' ANDed with Z^{-}, i.e., $Z^{-}(A'B' + A'C' + B'C')$. Figure 2.12 illustrates the schematic of this TH23 at a transistor level with its corresponding equations and symbol.

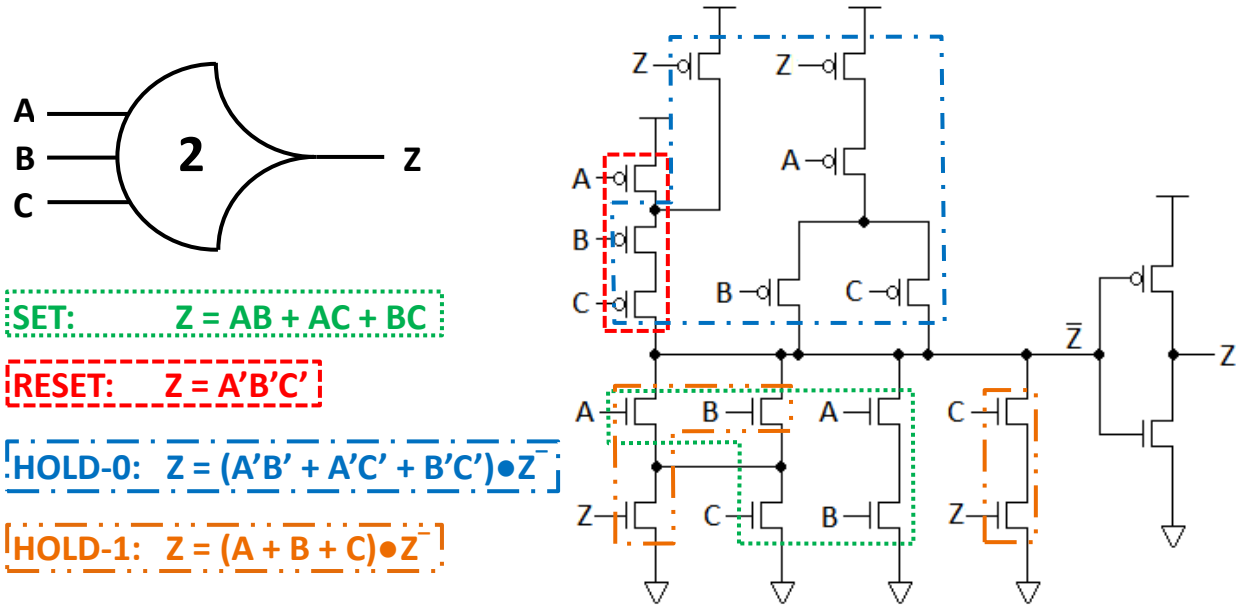


Figure 2.12: Schematic, equations, and symbol of a TH23 gate.

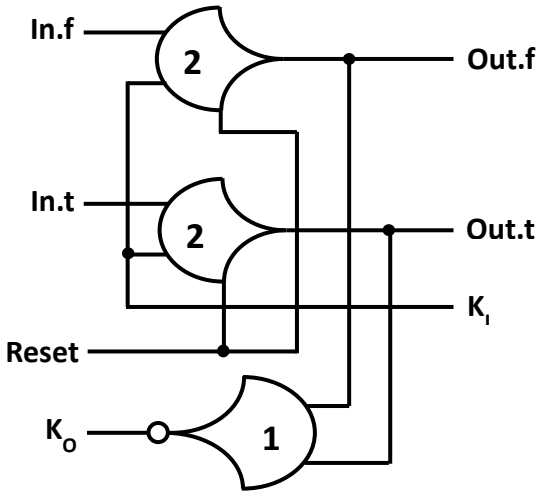


Figure 2.13: Single-bit dual-rail NCL register.

NCL Registers

The most basic register in NCL is the single bit dual rail NCL register, which is constituted by two TH22 gates and one NOR threshold gate (i.e., a TH12 gate without the output

inverter) as showed in Figure 2.13. In order to prevent a deadlock (e.g., "system A" needs an output from "system B" in order to proceed, but "system B" is waiting for "system A" to finish), a NCL circuit requires from at least three NCL registers that are alternating DATA and NULL *wavefronts* in a feedback loop. That is, adjacent NCL registers need to prevent a new DATA wavefront to overwrite the previous DATA wavefront by using their request signal *'K_I'*, and their acknowledge signal *'K_O'*, and ensuring that a NULL wavefront always separates two DATA wavefronts: … → DATA → NULL → DATA → … [16]. For example, considering the handshaking communication between registers *'A'* and *'B'*, this is established as following: DATA and NULL wavefronts flow from *'register A'* to *'register B'*; the acknowledgment signal *'B_K_O'* from *'register B'* is combined in a *Completion Detection* module to generate a request signal *'A_K_I'* that goes to *'register A'*; when *'register B'* finishes receiving DATA that is coming from *'register A'*, it will set signal B_K_O = *rfn* (*request for null*, i.e., logical 0), so *'register A'* will sense A_K_I = rfn and will start sending a NULL wavefront; then when *'register B'* finishes receiving the NULL wavefront coming from *'register A'*, it will set signal B_K_O = *rfd* (*request for data*, i.e., logical 1), so *'register A'* will sense A_K_I = rfd and it will start sending DATA again, which will correspond to a new cycle. Figure 2.14 displays the arrangement of this just described example.

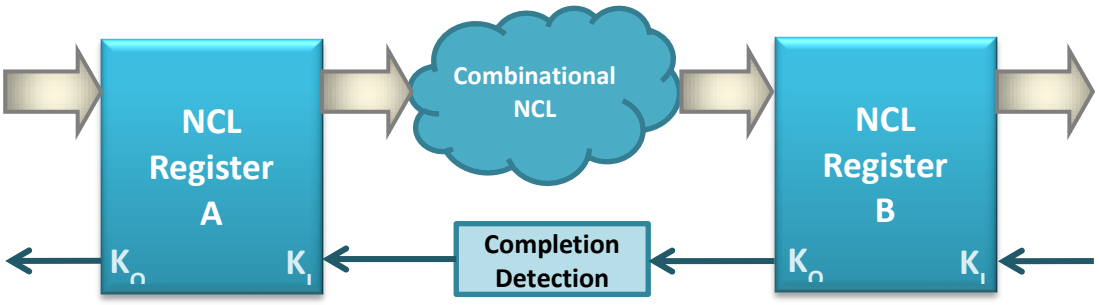


Figure 2.14: Handshaking communication between NCL registers.

Combinational NCL Design

Scott Smith and Jia Di stated in [16] that *"NCL circuit design is similar to synchronous Boolean design, where minimized equations are generated and then mapped to a set of gates;*

however, NCL circuits must be both input complete and observable in order to achieve delay-insensitivity."

- **Input Completeness**

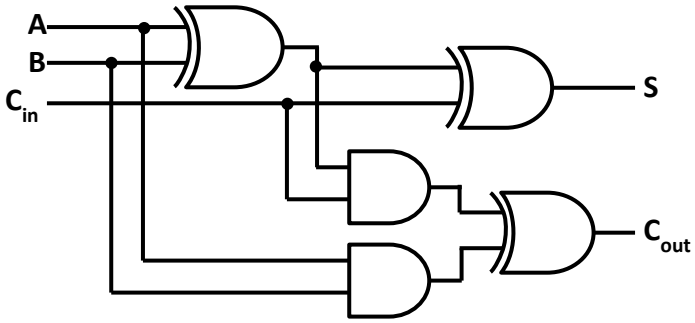
As described previously, the output of a combinational NCL circuit transition to DATA or NULL if and only if all the inputs had transitioned to DATA or NULL, respectively. In order to achieve an NCL circuit with input completeness, the output equations must enclose the input completeness with respect to all the input variables. For example, consider the circuit with output equations: Y = AC + AD, Z = BD; the output Y is input complete with respect to A because the equation of Y includes A in all the product terms (Y = **A**C + **A**D); the output Z is input complete with respect B and D because the equation of Z includes the variables B and D in all the product terms (Z = **BD**); however, none of the outputs are input complete with respect to C, so the circuit is considered in general as not input complete. In order to achieve input completeness in this circuit then any of the outputs (Y or Z) could become input complete with respect to C by ANDing 'C' to the product terms that are missing this variable, i.e., Y = AC + AD**C**, or Z = BD**C**.

- **Observability**

An *orphan* is a signal that transitions during a DATA wavefront without affecting the determination of an output [16]. The property of *observability* in a NCL system is achieved when *no orphans* propagate through an NCL gate. Thus, a NCL circuit that fulfills the observability condition has only gates that when transition at least one output transitions. Smith and Di described in [16] that *"the best way to ensure that a circuit is observable is to not divide product terms when mapping equations to their corresponding gate-level circuits."* Nonetheless, the observability condition might be not always possible to achieve in some complex circuits; however, the *isochronic fork* assumption can neglect *orphan signals*.

- ***From Boolean logic to dual-rail combinational NCL***

Designing a dual-rail combinational NCL circuit is not that complicated if having experience designing Boolean circuits. The usage of *Karnaugh maps* (*K-maps*) is also required to optimize the *Sum-Of-Products* (*SOP*) that characterize the output equations. The difference of dual rail is that both logical *0's* and *1's* from the K-maps are considered to express the *false state* rail *'d.f'* (using the logical *0's*) and the *true state* rail *'d.t'* (using the logical *1's*). Remember that in a dual rail configuration a variable is split in two rails, e.g., A ⊢ {A.t, A.f}; thus the original variable remains in the true rail, e.g., A = A.t, and the complement of the variable is expressed in the false rail, e.g., A' = A.f.



A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Figure 2.15: Logic circuit and truth table of a single-bit full adder.

In order to explain the process of dual rail combinational NCL design consider a 1 bit full adder with inputs A, B, C_{in}, and outputs C_{out}, S. Figure 2.15 shows the representation of this adder with traditional Boolean gates and its corresponding truth table. Then follow these next steps:

1) Express all the variables in a dual-rail configuration:

$$A \vdash \{A.t, A.f\} \quad \therefore \; A = A.t, \quad A' = A.f$$

$$B \vdash \{B.t, B.f\} \quad \therefore \; B = B.t, \quad B' = B.f$$

$$C_{in} \vdash \{C_{in}.t, C_{in}.f\} \quad \therefore \; C_{in}=C_{in}.t, \quad C_{in}'=C_{in}.f$$

$$C_{out} \vdash \{C_{out}.t, C_{out}.f\} \quad \therefore \; C_{out}=C_{out}.t, \quad C_{out}'=C_{out}.f$$

$$S \vdash \{S.t, S.f\} \quad \therefore \; S = S.t, \quad S' = S.f$$

2) Like in regular Boolean design use K-maps to derive the equations of the original output variables (using the logical 1's) and the equations of the complement of these outputs (using the logical 0's). Figure 2.16a shows the derivation of the logic equations for outputs $C_{out}.t$, and $C_{out}.f$ from a K-map, which yields these expressions:

$$C_{out}.t = A.tB.t + A.tC_{in}.t + B.tC_{in}.t$$

$$C_{out}.f = A.fB.f + A.fC_{in}.f + B.fC_{in}.f$$

Figure 2.16b shows the derivation of the logic equations for outputs S.t, and S.f, which yields these expressions:

$$S.t = A.tB.tC_{in}.t+A.tB.fC_{in}.f+A.fB.tC_{in}.f+A.fB.fC_{in}.t$$

$$S.f = A.fB.fC_{in}.f+A.fB.tC_{in}.t+A.tB.fC_{in}.t+A.tB.tC_{in}.f$$

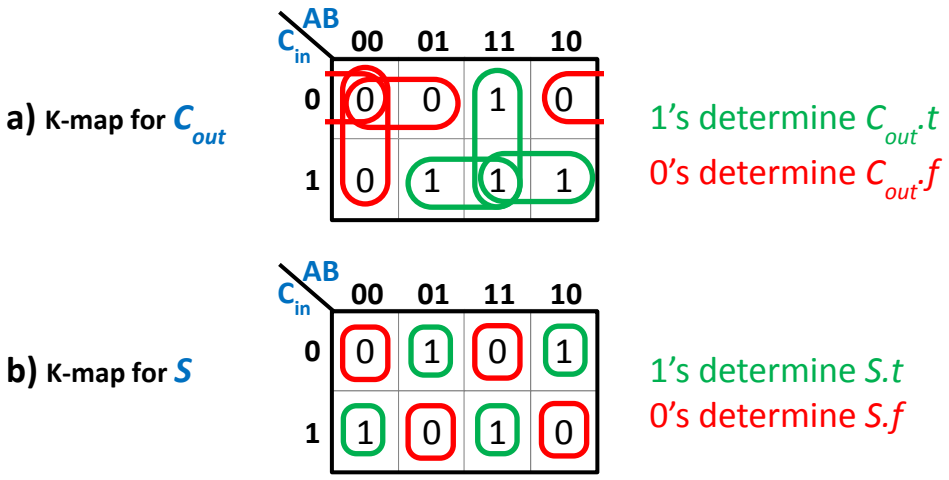


Figure 2.16: Karnaugh maps to determine C_{out} and S logic equations.

3) Check that the circuit meets the input completeness requirements by ensuring that the output equations are input complete with respect to all the inputs. For instance, the output equation *S.t* is input complete with respect to A, B, and C_{in} because these variables are included in all the SOP terms of the *S.t* output equation. Since these are the total input variables of the circuit the adder is considered to have input completeness:

TOTAL INPUTS: A, B, C. $S.t = A.tB.tC_{in}.t + A.tB.fC_{in}.f + A.fB.tC_{in}.f + A.fB.fC_{in}.t$

4) Associate sets of four or fewer logical rails from the output equations with the corresponding set equations from the 27 fundamental NCL gates. For instance, the *S.t* equation can be implemented using four TH33 gate (Z = ABC) driving one TH14 gate (Z = A + B + C + D).

5) Finally, when all the equations are represented with their corresponding NCL gates, ensure that every single gate provokes the assertion of an output when the output is asserted in order to avoid orphans and achieve observability in the circuit. Figure 2.17 shows the basic NCL gate-level circuit of a 1-bit full adder with dual rail configuration that was designed by following these steps. Nonetheless, this adder can be optimized by modifying the *S.t* and *S.f* equations in function of $C_{out}.t$ and $C_{out}.f$ (as managed using regular Boolean logic and K-maps) for reducing the number of NCL gates as illustrated in Figure 2.18 (note that observability and input completeness still needs to be ensured).

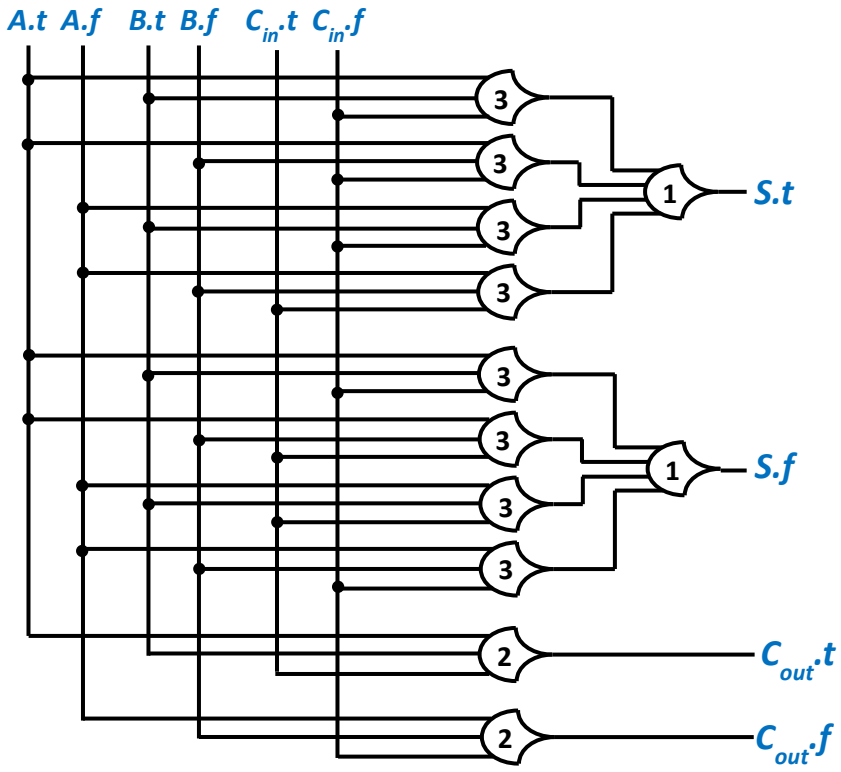


Figure 2.17: Basic NCL circuit of a 1-bit full adder.

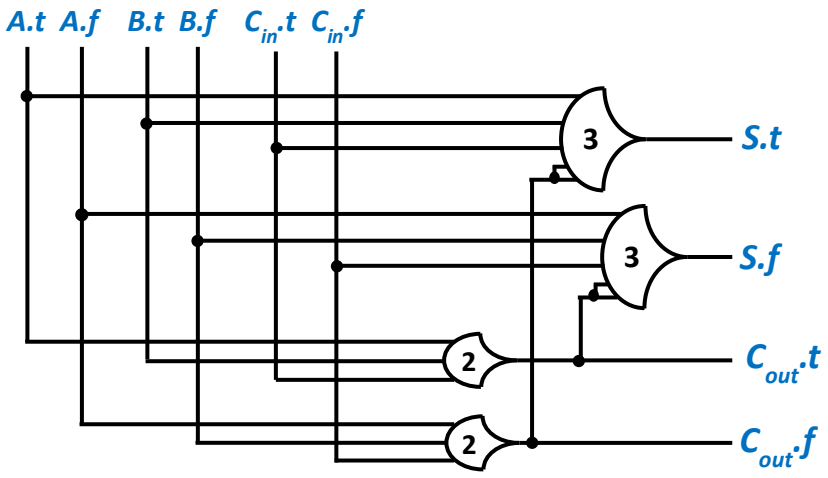


Figure 2.18: Optimized circuit version of a NCL 1-bit full adder.

2.2 Fault Tolerance

Shrinking the size and lowering the operating voltages of electronic circuits are practices that are required to meet the demand of more compact electronics and low power applications. However, these approaches might come at the expense of radiation robustness [27]. Furthermore, increasing the fault tolerance of a system is an essential requirement of applications that are exposed to radioactive environments (e.g., space, nuclear plants). In fact, selecting the level of *radiation hardness (radhard)* in a circuit is based on whether the circuit will operate in *low-radioactive* zones (e.g., low orbit, non-radioactive terrestrial environments), or in *high-radioactive* regions (e.g., high orbit, radioactive terrestrial areas) [28, 29]. For instance, a satellite orbiting the Earth is expected to function for at least 10 years, which encourages the utilization of robust electronics that can subsist along this meaningful period.

2.2.1 Radiation

For this research *'radiation'* is a general term to refer to the incident ionization of matter, which is provoked by energetic particles emitted at the time a nuclei of an atom is disintegrated. When a material is irradiated, an amount of energy is deposited. The measure of this energy per mass unit is referred to as *dose*, which can be expressed in *rad* or *Gray (Gy)*. These are the equivalences: 1 Gy = 100 rad = 1 J/Kg. Considering the radiation of Silicon (Si), a

high-dose rate in this material is considered as greater than 10 rad per second (rad/s), and a *low-dose rate* goes below 0.1 rad/s [29].

2.2.2 Single Event Effects (SEE)

The amount of charge that causes a change of state in sequential logic modules is defined as the *critical charge* – Q_{crit}, which is an important measure to determine the fault tolerance in a circuit [30].

The drain part of an OFF transistor connected to the feedback loop arrangement in a sequential element is considered a *'sensitive node'* during the occurrence of *collection of charge.* If the collected charge in a sensitive node surpasses the minimum Q_{crit}, a transient electrical current is caused, which in turn might provoke the change of state at the output of the logic module (i.e., *bit-flip*) [28].

A *'single event'* is considered in literature as the incidence of a radioactive particle striking a circuit, and the *collection of charge* is one of the effects of this phenomenon (i.e., *Single Event Effects (SEE)*). Hence, single events can introduce faults on logic circuits. A *Single Event Transient (SET)* is related to the occurrence of a particle striking a combinatorial logic circuit, in which the inputs force the output to recover its original state (the excessive charge is ultimately attenuated). In contrast, a *Single Event Upset (SEU)* is associated with the occurrence of a particle striking a sequential logic circuit, where a SET is latched (due to the feedback configuration) and propagated throughout the entire logic system. This incident causes a temporary fault, which is referred to as a '*soft error*' or simply as an *'upset'* [28, 29]. The frequency at which these *fault events* occur is denominated *soft error rate (SER)* [27]. Since an upset does not represent a permanent physical damage, a SEU will last in a system until this is reset. In extreme cases, a single event with a high dose of radiation can cause physical injuries in circuits as in the case of a *Single Event Latchup (SEL).* A *'latchup'* is an electric incidence with short-circuit aspect that results from an excessive amount of collected charge and propitiates a logic path with low resistance. Consequently, this phenomenon leads to *permanent failures*, such

as blown bond wires or metallization on the die [28]. Another consequence of disproportionate charge in a circuit, which is also considered as a physical disruption, is the increase of thermal energy, which in the worst case can cause a *junction burnout* if the heat is slowly dissipated [28].

According to [27], soft errors in semiconductor devices (e.g. CMOS devices) are induced mainly by *alpha-particles*, and secondly by *high-energy cosmic ray neutrons*.

Alpha-Particles

An *alpha particle (α)* is a positive charged nuclear particle composed of two protons and two neutrons, comparable to helium nucleus (He^{2+}), and emitted with an approximated kinetic energy range of 4 to 9 MeV when the nucleus of an unstable isotope decays to a lower energy state [27]. These radioactive particles interact with the electric charge in matter by inducing significant amounts of electron-hole pairs on junction nodes (i.e., *collection of charge*) while transferring energy. For example, a single alpha-particle striking Silicon (Si) loses an average of 3.6 eV of energy for every electron-hole pair created. Even though the *charge generation rate* is increased when traveling more distance, the velocity of an alpha-particle strike is reduced while traversing the material. Consequently, the higher density of charge in the material, the faster the alpha-particle loses its kinetic energy [27]. The rate of energy loss in a material is referred to as Linear Energy Transfer (LET) [28]. An alpha-particle generates anywhere from 4 to 16 fC/μm over its entire energy range as shown in Figure 2. 19, which considers the stopping power (eV/m) in terms of differential charge (dQ/dX) [27]. Likewise, an approximated analytic model of the collected charge induced by an *ion track* (e.g., alpha-particle penetrating silicon) was developed in [31] expressed as a double exponential current in function of time:

$$\boldsymbol{I(t) = I_O\left[e^{-\alpha t} - e^{-\beta t}\right]} \qquad (2.1)$$

where I_o is the approximately maximum induced current, $1/\alpha$ is the collection time constant of the junction (falling time), and $1/\beta$ is the time constant for establishing the rising of the ion track (rising time). Figure 2.20 displays the transient response for a 5 MeV alpha-particle with electron concentration $N_D = 1\times10^{15}cm^{-3}$ and $N_D = 7\times10^{15}cm^{-3}$ [31].

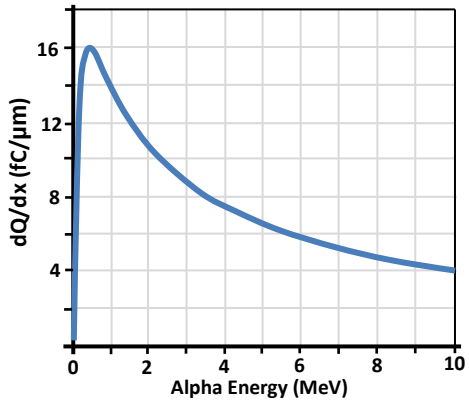


Figure 2.19: Differential charge for an alpha-particle in Si as a function of energy [27].

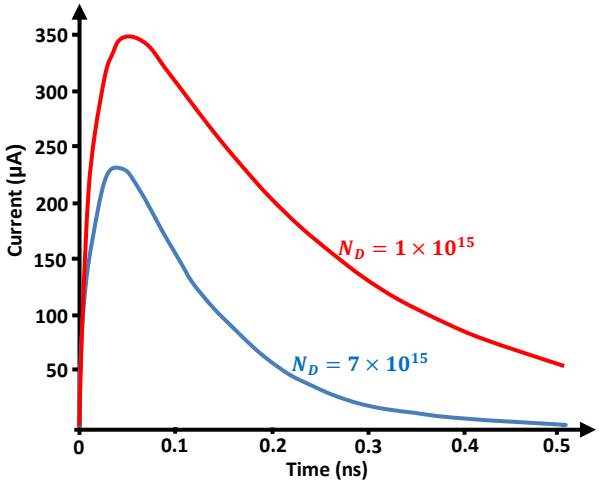


Figure 2.20: Transient response of a 5 MeV alpha-particle [31].

High-energy cosmic ray neutrons:

Cosmic rays are conformed of energetic heavy ions that irradiate uniformly the Earth and penetrate the atmosphere creating cascades of particles, from which only the neutrons are considered to cause upset in silicon-base devices at terrestrial altitudes. This upset is originated when the neutron collides with a silicon nucleus and transfers enough kinetic energy to hit the silicon from the lattice breaking into smaller fragments that generate charge. At difference from alpha-particles, the effect of cosmic ray neutrons in circuits cannot be reduced meaningfully with protective shields (e.g. lead shields, concrete walls), thus in order to reduce the SER induced by cosmic rays, the fault tolerance of circuits needs to be achieved by design or by process adjustments.

2.2.3 SER Mitigation

In order to increase the fault tolerance in a system, two main approaches are considered to mitigate soft errors in circuits: 1) *radiation hardness by process*, and 2) *radiation hardness by design (RHBD).*

The next described techniques fall under the category of radiation hardness by process:

Depleted Borophosphosilicate-glass (BPSG) — Boron (B) is an element used in the formation of BPSG, which is utilized in the fabrication of insulating layers (e.g. metal layers insulation) for semiconductor devices. This element is naturally composed of two isotopes: ^{11}B (80.1%) and ^{10}B (19.9%) [32]; even though both of these isotopes react with thermal neutrons from cosmic rays, the reaction of ^{10}B occurs nearly 1 million times more than a reaction of ^{11}B. Moreover, when ^{10}B interacts with a cosmic ray, this breaks apart (i.e. *fission*) releasing alpha particles. Consequently, the presence of ^{10}B in BPSG layers represents a hazard on semiconductor devices by augmenting the SER. In contrast, the sporadic reaction of ^{11}B to cosmic rays is not harmful. As a solution, depleted Boron, which is low in ^{10}B (e.g., 0.02 %) and rich in ^{11}B (e.g. 99.8%), has become a primordial ingredient for the creation of BPSG with augmented resistance to radiation [32].

The next described techniques fall under the category of radiation hardness by design:

Error Correction Codes (ECC) — Extra memory cells are added to the circuit as a requirement of error correction codes at a software level, where a single 'bit-flip' can be detected and corrected with the help of parity bits (i.e. *check bits* that state if the number of logical 1's in a string of bits is odd or even) that corroborate transferred data. The most used technique of ECC is the Hamming Code for protecting memory. This method can detect two 'bit-flips' simultaneously in a read operation, but not correct them; nonetheless, the chances of having two SEU in the same logic module are negligible and not considered for this research. ECC does not represent a complex modification in the design of circuits since the only required modification in a logic circuit is the addition of more memory cells. Even though the presence of SEU (i.e., data

is still corrupted) in circuits are permitted by this technique, the probabilities of the total breakdown of the system are diminished.

Triple Modular Redundancy (TMR) — This method is considered an error masking approach, where a logic circuit is triplicated, and a majority voter system applying 2-out-of-3 criteria, determines the final output of the circuit [33, 34]. The penalty of this approach is the excessive increased area, which might be an issue for shrunk technologies. For example, consider the original logic module '*A1*' with replicates '*A2*' and '*A3*'; if the correct output is supposed to be '*logical 0'*, but module A3 suffered a flip-bit, which results in a '*logical 1'*, then the majority voter will decide that the correct output is 'logical 0' because this value was received twice coming from the modules A1 and A2. Figure 2.21 represents this example.

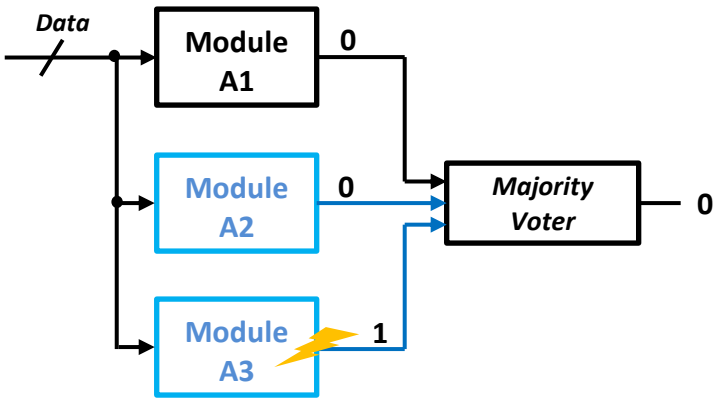


Figure 2.21: Logical system with TMR configuration.

Interwoven Duplication — This approach suggest the interconnection of two identical logic modules managing the same data, in which the feedback of the sequential structure of a module drives the combinational arrangement of the opposite module, resulting in a logical dependency. If a single event occurs at a module, the effects will be mitigated by blocking a resultant SEU in the affected node and restoring the output from the backup module (opposite unaffected module). This configuration was first introduced in [34], in which a Dual-Interlocked Storage Cell (DICE) was proposed for increasing considerably the SEU tolerance in memory elements. Figure 2.22 shows an interwoven duplicated feedback loop with SEU tolerance.

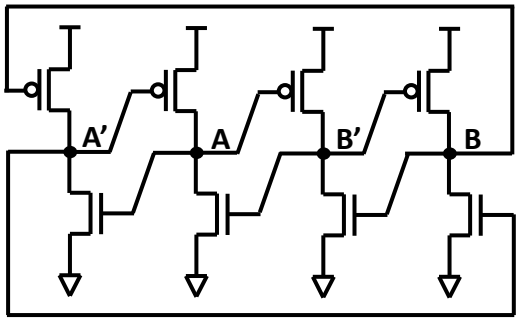


Figure 2.22: Interwoven duplicated feedback loop.

2.3 Subthreshold Operation

The total energy – E_{tot} – dissipated in CMOS circuits is obtained by adding the *dynamic energy* – E_{dyn} – plus the *leakage energy* – E_{leak} – plus the *short-circuit energy* – E_{sc} – simultaneously represented in this next equation:

$$E_{tot} = \alpha C_{load} {V_{dd}}^2 + I_{leak} V_{dd} t_{leak} + I_{peak} t_{sc} V_{dd} \qquad (2.2)$$

where E_{dyn} characterizes the consumed energy for charging load capacitance C_{load} with probability α, E_{leak} is the undesired lost energy when powering a circuit during time t_{leak} , and E_{sc} is dissipated when both NMOS and PMOS transistors are simultaneously conducting during the short time t_{sc} [23]. This clarifies that by lowering the voltage supply a quadratic reduction of power is attained. Even though the reduction of V_{dd} increases the delay, an optimum reduction of *power-delay product (PDP)* can be satisfactorily reached with a properly constructed circuit as discussed in [20, 36-38]. For this reason, subthreshold voltage regime has been seen as an optimum solution for low power applications that consider performance a secondary concern. Figure 2.23 displays the plot of power-delay product exhibiting the quadratic voltage dependence for two different circuits.

Subthreshold voltage operation of transistors refers to the situation in which the electrical potential difference from gate-to-source – V_{GS} – is less than the threshold voltage – V_{T0} – in transistors, i.e., $V_{GS} < V_{T0}$ [35]. The voltage supply of a logic circuit working under this regime is scaled down to less than V_{T0}, and the leakage current –I_{leak} – of transistors is used as the

switching current to implement logic because at that regime the total energy dissipated in a circuit is characterized mostly by the E_{leak} instead of E_{dyn} as shown in figure 2.24 [25].

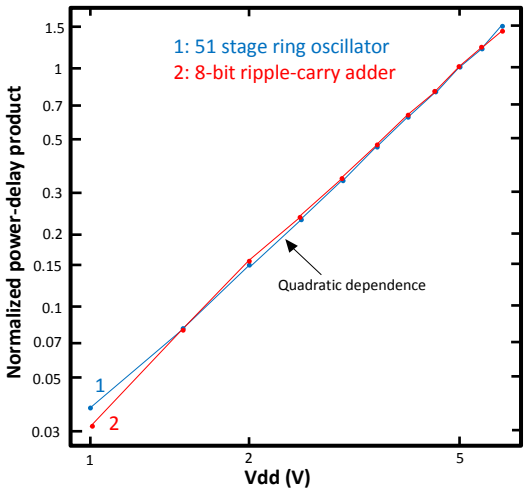


Figure 2.23: PDP with quadratic dependence on V_{dd} [20].

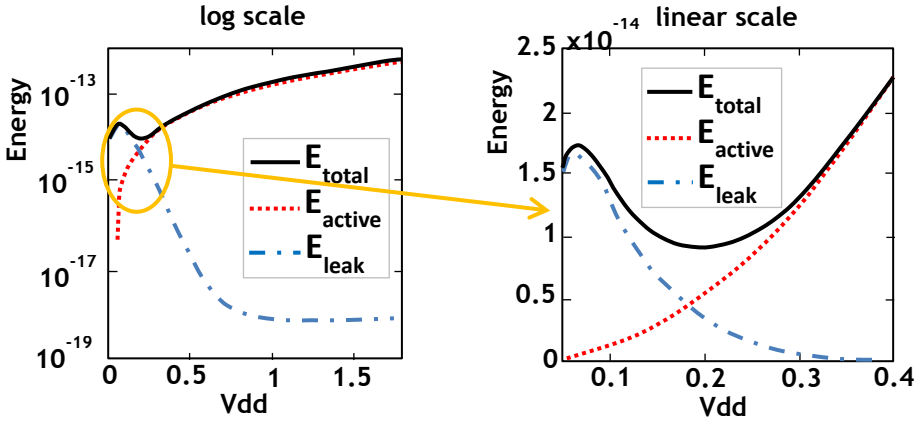


Figure 2.24: Total energy as a function of V_{dd} [21].

In a subthreshold regime a MOSFET behaves in a similar way to a bipolar transistor, in which the subthreshold current is exponentially dependent on the V_{GS} and independent of the drain-to-source voltage – V_{DS} – [20].

The basic equation for modeling the subthreshold current is:

$$I_{sub} = I_0 e^{\frac{(V_{GS}-V_T)}{nV_{th}}} \qquad (2.3)$$

where n is the subthreshold slope factor (i.e., amount of voltage required to drop V_{th} by one decade) expressed as $1 + C_d/C_{ox}$, V_{th} results from kT/q, and I_0 is the total OFF electrical current described with this equation:

$$I_0 = \mu_0 C_{ox} \frac{W}{L} (n-1) V_{th}^2 \qquad (2.4)$$

where μ_0 is the carrier mobility, C_{ox} is the gate oxide capacitance per unit area, W is the width of the transistor and L is the length of the transistor [36, 37].

2.3.1 Considerations for a Subthreshold Regime in CMOS Circuits

In order to increase the fault tolerance in a system, two main approaches are considered to mitigate soft errors The aspects to consider when designing logic circuits with subthreshold regime are enclosed by 1) transistor size scaling with balanced topologies, 2) selection of an optimal subthreshold voltage – V_{sub} – , and 3) process technology optimized for subthreshold operation.

Sizing of Transistors — Scaling the size of transistors allows the minimization of operational V_{dd} below V_{T0} [36]. The reason is because balancing the size of PMOS transistors versus the size of NMOS transistors allows the existence of an equivalent current in both types of transistor, as discussed in [40]. For instance, the PMOS/NMOS ratio that results in an equal subthreshold current for a 0.18-µm technology is 12 as proved in [36], allowing to operate with a V_{dd} of 70 mV.

Minimum Energy-Optimal Voltage – V_{min} — Since lowering V_{dd} increases exponentially the delay in CMOS circuits, this can result in a counter-effect to the improvement of the total energy savings if the E_{leak} dissipated during time delay T_d exceeds the energy that a circuit would consume during the same time with a superthreshold voltage. Thus, selecting the optimal voltage that results in a minimum energy per operation is crucial for subthreshold operation. Analytical studies have been evaluated in [21, 36] for modeling V_{min}, which represents the ideal voltage supply for energy savings in subthreshold circuits.

Subthreshold Suitable Process Technology — Modern process technologies have been emerged alleviating the exhaust analysis to find the correct V_{dd} and transistors' size that lets a logic circuit to operate in a subthreshold regime [24]. Such processes enhance the mobility degradation and the saturation velocity [40]. Likewise, since the hot carrier injection is not present at sub-voltages, the *lightly doped drain (LDD)* process is not needed anymore [40].

Chapter 3: Previous Work

3.1 Fault Tolerant Asynchronous Circuits

Even though asynchronous logic represents an efficient methodology to create robust circuits that can handle the effects of Process, Voltage, Temperature (PVT) variations; there is not an inclusive asynchronous methodology that guarantees increased tolerance to radiation in circuits. Consequently, extra procedures are needed to be applied when designing asynchronous circuits that are intended to work in hostile environments. Approaches that improved the robustness of asynchronous circuits are next referenced.

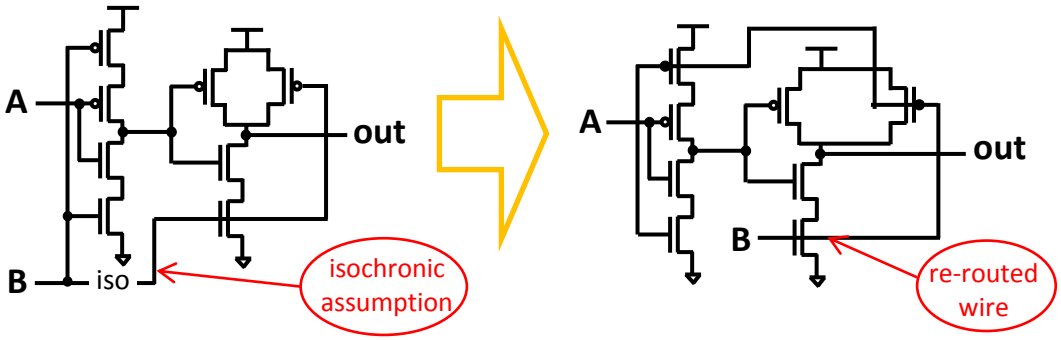


Figure 3.1: Forking avoidance with routing isochronic wires [41].

Delay faults in Quasi Delay Insensitive (QDI) circuits might result in a logical fault if they are present in isochronic forks because of the permitted time assumption. If a delay prolongs more than estimated in an assumed isochronic wire, then the expected logical behavior becomes corrupted and the probabilities of stuck-at faults (e.g., stuck-at-one, stuck-at-zero) occurrence increase. Stuck-at-faults inhibit the transition of handshaking signals resulting in deadlock. According to [41] a deadlock is a more desirable event than incorrect logical actions because by monitoring the forward process of the QDI circuit every certain time a deadlock can be detected. Thus, detecting deadlocks in QDI circuits was an approach adopted by [41] in order to isolate logical faults. Likewise, converting a fault into deadlock to later be detected is an optional

practice. On the other hand, a solution for mitigating delay fault in wires with isochronic assumptions was presented in [41]. This technique suggested the routing of a signal through the gate that originally would receive the fork signal with isochronic assumption and then rout the signal to the rest of the gates receiving the same signal. Figure 3.1 shows a circuit with isochronic fork before and after implementing this routing approach.

On the other hand, QDI circuits in [42] were represented with Production of Rule Sets (PRS), in which $G \rightarrow S$ represents a Boolean expression *'G'* with simple assignment *'S'* (e.g., if G is true, then S is executed, else S is skipped). A simple assignment can have for example the form $Z \uparrow$ (i.e., Z becomes true) or $Z \downarrow$ (i.e., Z becomes false). These PRS were used to describe the functional logic of a circuit to later be translated into a transition diagram, in which illegal states and undesirable conditions are excluded. For instance, following the principles of [42] and considering the next PRS: $X \rightarrow Z \downarrow, \neg X \rightarrow Z \uparrow, Y \rightarrow X \downarrow, \neg Y \rightarrow X \uparrow, Z \rightarrow Y \downarrow, \neg Z \rightarrow Y \uparrow$, with initial state S_o(X,Y,Z) = (101) the transition diagram in Figure 3.2 is derived. Accordingly, [42] stated that three main types of effects emerge in QDI circuit when suffering a Single Event Upset (SEU): 1) deadlock (stuck at a logic state), 2) abnormal transitions that exclude logic states, and 3) tolerant transitions that let the circuit to eventually recover. Figure 3.3 shows transition diagrams with these effects. In order to prevent these effects to happen in QDI circuits, [42] proposed the usage of doubled-checked gates, which consisted of a duplication of an original gate, and two more Muller C-elements for corroborating similarity in doubled-up PRS as pictured in Figure 3.4. In case of SEU, disparity at the outputs of the proposed gate would arise. So now the description of the QDI circuit needed to include PRS that account for equality at the redundant outputs for transitioning. Although this solution reflected an approachable mitigation of SEU effect in QDI circuits, the complexity of the QDI circuits doubled up. Moreover, area

was increased not just because of the duplication of the gates, but the requirement of additive C-elements (for each logic gate one more C-element would be added). Furthermore, the effectiveness of this approach is in part dependent on the robustness of the C-elements. This concern was taken into consideration in this thesis work and a proposed Recursive C-element with increased fault tolerance is presented in section 4.2.

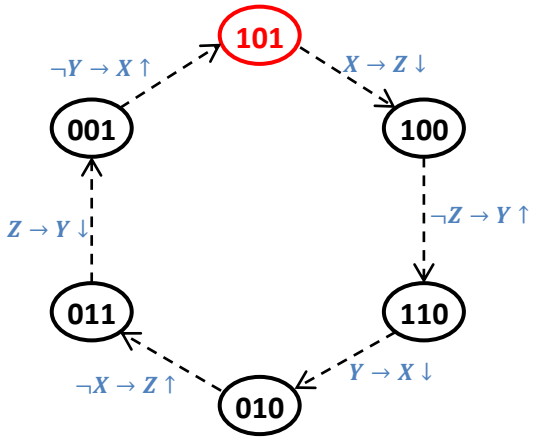


Figure 3.2: Transition diagram instance derived from determined PRS [42].

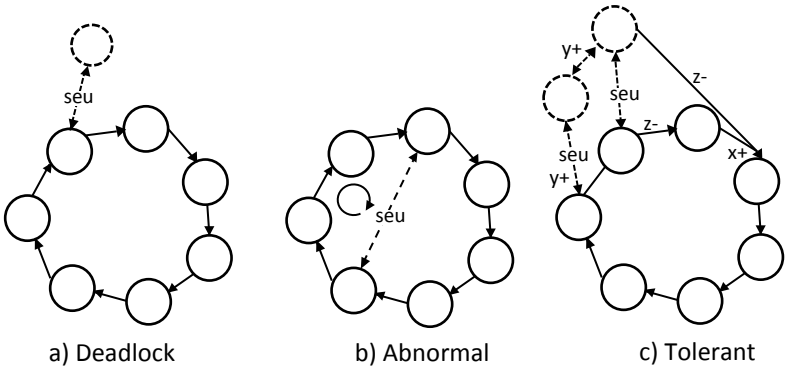


Figure 3.3: Effects of SEU in QDI transition diagrams [42].

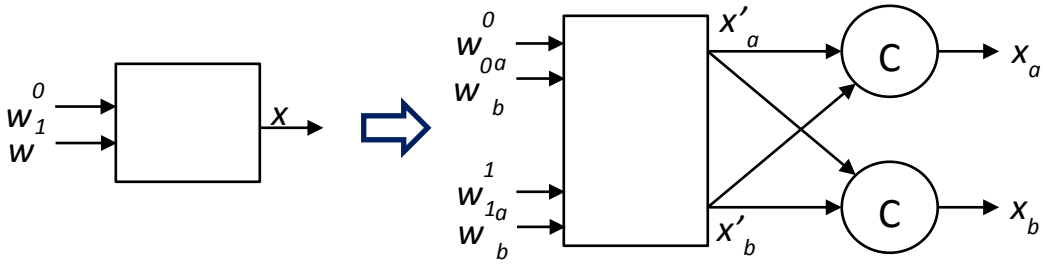


Figure 3.4: Translation of a logical gate into a double-checked gate [42].

In order to mitigate the propagation of transient faults through Muller pipeline latches (i.e., C-element structured latches) a new latch scheme was proposed in [43]. This suggested latch has an additional latch control that detects and correct possible illegal states in a dual-rail configuration (i.e., {d.t, d.f} = {1, 1}) by implementing redundancy in terms of both area and time. The technique implemented to design the latch control is the usage of *Signal Transition Graphs (STG)* with Petri Net theory foundation. The implementation of this approach resulted in a dual-rail handshaking scheme that contains two Muller latches, one with a resettable architecture and one with a regular design. Both of these latches receive the same input data (redundancy in area) and feed a NOR gate to detect NULL states and a XOR gate to detect DATA states. The outputs of both NOR and XOR gate drive a logical 1 alternately each other to characterize the request and acknowledge signals (e.g., K_I, K_O) needed in handshaking protocols. If the output of both NOR and XOR gate drive a logical 1 simultaneously, which means an illegal latched state, then the scheme is reset and asked to latch again the received input data (redundancy in time). Figure 3.5 shows the arrangement of this proposed latch. This approach resulted in a significant area penalty because the latch is duplicated and additional logical gates are required to form the controller. Furthermore, the applied redundancy in time provoked a significantly longer latency, which affected negatively the circuit performance.

According to [44] a sensitive time duration exists in which a SEU can cause a bit-flip to propagate. Consequently, an additive self-feedback register in NULL Convention Logic™ (NCL) pipelines is suggested in order to reduce that sensitive window. Although this approach enhanced fault tolerance, just one Single Event Upset (SEU) scenario was considered (a 0-1 bit-flip) leaving more SEU faults without resolution. Therefore, the area penalty resulting from this approach was not worthwhile. Nonetheless, this strategy was complemented by the

work presented in [45], which suggested the implementation of modified NCL gates in asynchronous design. The structure of the threshold gates was modified by exchanging the inverter that drives the output for the Schmitt trigger structure shown in Figure 3.6, which intended to stabilize transients at the output node. As mentioned before, the approaches of these references did not account for all possible error scenarios caused by SEU. Moreover, the simulation of a particle strike considered in this source was not that much close to a real single event regarding periods and shape.

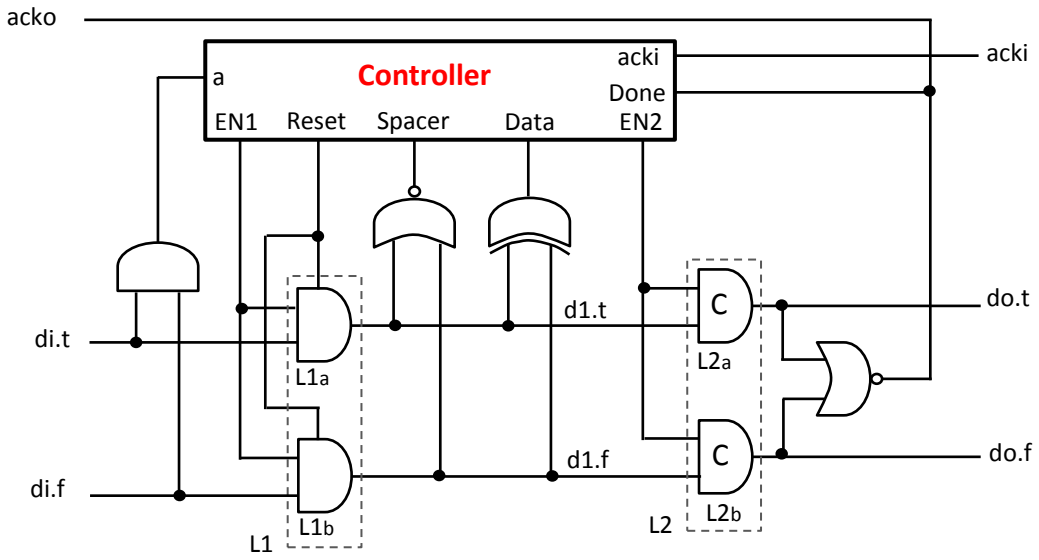


Figure 3.5: Latch with logic controller proposed in [43].

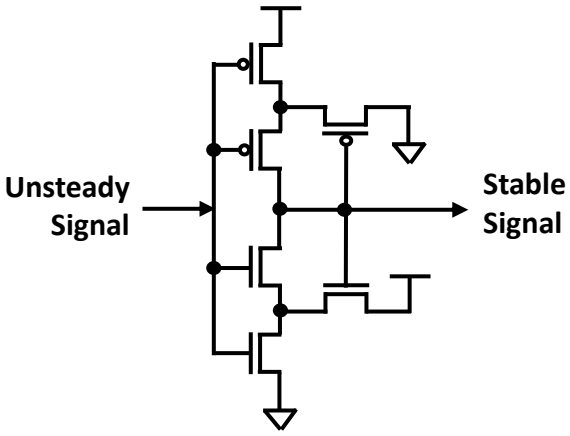


Figure 3.6: Schmitt trigger used in [45].

3.2 Asynchronous Circuits with Subthreshold Operation

When working at a subthreshold operation, logic circuits become more sensitive to PVT variations, and hence delays deviate more drastically. Such variable delays cause instability in synchronous circuits like clock skews, which generally end up with the failure of the whole system. For this reason, subthreshold approaches with asynchronous considerations, like the instances presented in the next paragraphs, resulted in a promising combination for low-power applications.

A clever idea to reduce as much energy as possible in circuits is to combine existent energy saving techniques (e.g., power gating, clock gating, dynamic voltage) with adaptions for sub-V_T powering. However, this approach is not a straight forward process because working at the subthreshold regime increases the sensitiveness of circuits to PVT variations as mentioned previously. Thus, original energy saving techniques need to be modified when using $V_{dd} < V_T$ in order to remain effective. An example of this tactic was considered in [46] for permitting the Self-Adaptive V_{dd} Scaling (SAVS) technique to be suitable for sub-V_T. The first consideration was to implement a QDI structure to handle the unpredictable delays associated with PVT variations at sub-V_T (the robustness of the SAVS circuitry is enhanced). However, this approach resulted in a power overhead, for which [46] proposed a modification in the QDI hardware named *'pseudo-QDI.'* This proposed technique required an analysis of a QDI pipeline in order to determine an additional timing requirement on the reset of a four-phase configuration, remove the entire Data-path Completion Detection (DCD), and just leave the latch responsible of sensing the last completion signals. Thus, by removing all of logic modules that were shaping the DCD, a notable amount of energy and area is reduced. This approach seemed very promising, however

not just an exhaust logical analysis was required, but the robustness of the QDI hardware was affected.

As a result of combining asynchronous fundamentals with subthreshold considerations, a 4-bit multiplier that can operate at 150 mV was developed in [47]. Since asynchronous circuits are *event-driven*, a new assignment can be expected for an undetermined period while consuming only leakage power. In the case of this subthreshold asynchronous multiplier, if a multiplication by 'zero' is detected, then a 'done' signal can be immediately triggered skipping futile calculations that would occur in a synchronous operation mode. Moreover, three to five times reduction in power was reported in [47] comparing a synchronous and an asynchronous multiplier working at a subthreshold regime. The technique adopted by [47] in the creation of this multiplier included an asynchronous communication among four 1-bit addition blocks that constituted a shift-and-add structure. Each of the addition blocks was responsible to compute delivered data when requested, store the result in a latch, and generate a 'done' signal which alerted the next block to start computing. Even though this work reported functionality of the multiplier at 150 mV with a delay of ~0.1 ms and a lowest energy per computation of 1.17 pJ, a more deep description of the considerations adopted to allow the multiplier to operate at a subthreshold operation was not revealed. On the other hand, a low-voltage 16-bit signed product multiplier with asynchronous shift-add configuration was presented in [48]. This multiplier could handle only power-of-two terms so that the reference stated that the proposed design was not only less complex than other multiplier approaches, but also had one of the smallest IC area requirements by then (2009). Moreover, the low power consumption reported in this work (5.86 μW at 1.1 V and 1 MHz) was supported by these next considerations: 1) less power-switching activity was achieved by implementing signed magnitude data representation,

2) a shifter structure with transmission gates was implemented, 3) partial product terms were truncated to reduce the hardware by 50%, 4) latch-adder combined for reducing area and power, and 5) a transparent latch was used for blocking unnecessary switching [48]. These considerations allowed a notable reduction of power consumption per operation; however, the voltage supply was just minimized to 1.1 V falling out of competition regarding present asynchronous-subthreshold approaches [48]. Moreover, the same reference stated that when compared with other multipliers in literature, the proposed showed the longest delay.

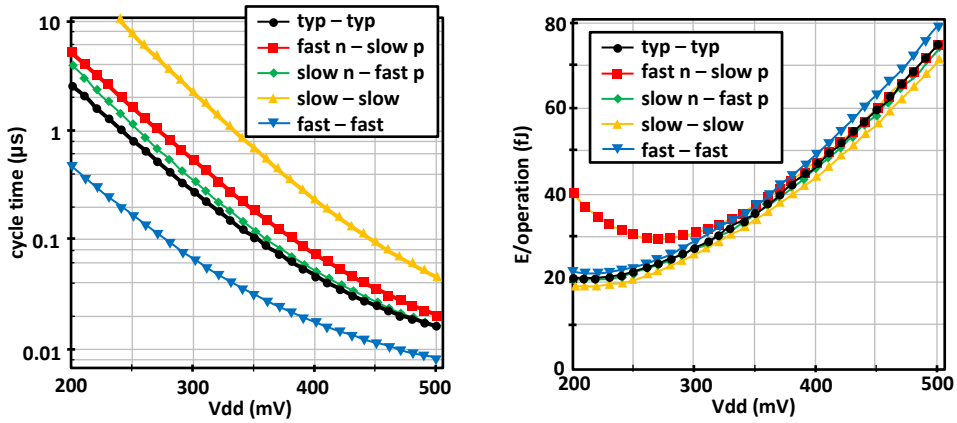


Figure 3.7: Delay vs. Sub-V_T (left) and Energy vs. Sub-V_T (right) of a ring counter [49].

Another source that supported the idea of complementing subthreshold management on circuits with asynchronous organization is [49], which realized a complete study on this combination of concepts. The main argument in this work was that existing Delay Insensitive approaches can show decent 'self-adaptability' in a sub-V_T domain when considering an optimum transistor sizing rather than straight symmetry. Likewise, this approach was complemented with an analysis that compared a voltage range below V_T, which was utilized to select the most advantageous scenario where circuits spent less energy per operation. Both of these examinations were combined to approximate an area-proportional leakage in relation to switching activity of circuits that results in a minimum energy-per-operation. For example, the

delay and energy per operation plots showed in Figure 3.7 were analyzed in [49] to approximate the most efficient setup for the circuit of a ring counter.

In order to maximize performance in asynchronous circuits that operate with Sub-V_T, an electrical current-sensing strategy was implemented in a *completion detection system* (CDS) [50]. This structure could sense driving current in the range of *pA – nA* that went through the PMOS transistors of combinational gates. This electrical current signal was then converted into a voltage signal to be first compressed in the log domain and secondly amplified. After this process, the amplified voltage signal feed a '*monostable multivibrator'* that generated a pulse with a width proportional to the calculation time of the combinational gate. Thus every time this pulse signal de-asserted (e.g., from 1 to 0) meant a completion. Figure 3.8 shows the block diagram of this approach. After simulations using a 16-bit ripple carry adder, [50] reported 32% delay optimization in the CDS with the expenses of 1 *nW* consumed by the amplifier. The penalty of this method was increased area for each combinational gate, and if NCL would be considered as an alternative, this approach would result to be obsolete because NCL has a self-implementation of completion detection.

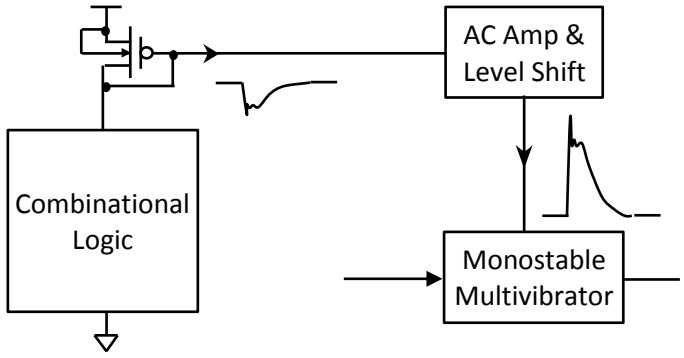


Figure 3.8: Completion detection system proposed in [50].

One of the approaches considered in this thesis for improving the functionality of circuits in a subthreshold regime is proposed in [51]. This reference combined two subthreshold

techniques that were implemented in NCL circuits: 1) threshold voltage adjustment, and 2) gate-topology balancing [52]. The first methodology implicated the usage of a gate isolation ring for body biasing placed around the NMOS transistors (pull-down network – PDN) of NCL gates. Consequently, by increasing V_T the leakage current decreased while prolonging gate delays (not a problem for asynchronous configurations). The second procedure was adopted from [52] and consisted in completing an electrical current analysis to balance the I_{on}/I_{off} ratio, where I_{on} represented the drive current when the device is active, and I_{off} represented the current when the device is idle. For example, considering a basic inverter, when the PMOS is driving current to assert a logical 1 output, the leakage current through the NMOS has to be considerably less (i.e., $I_{on}/I_{off} \gg 1$), otherwise the output might be result in a metastable state (i.e., neither logical 1 nor logical 0), or in the worst case flipped. Thus, for a particular gate, all possible input combinations were considered to come out with the most effective modification of the structure of circuits for avoiding *'metastability'*. As explained in [51], an effective way to balance the I_{on}/I_{off} ratio is by using a non-simplified product of terms when designing the logic structure of an NCL gate. This tactic allows designing a gate with an equal number of charging branches and leaking branches. The charging branches are considered as the electrical current path that goes through the devices that are ON for a certain combination of inputs. In contrast, the leaking branches are conformed by current that flows through OFF transistors for the same combinations of inputs. Figure 3.9 depicts unbalanced vs. balanced structure of circuits showing the charging and leaking branches at a particular input combination. Thus, according to [51] "*balancing the topologies allows for the logic gates to continue their operation in a region where leakage current is the dominant factor in total energy consumption."* An example of a circuit applying this methodology is an Arithmetic Logic Unit (ALU) presented in [53]. The

main design consideration for this ALU was to let the carry signal available before the sum output as a performance improvement. For the subthreshold operation the circuit arrangements from [51] were adopted.

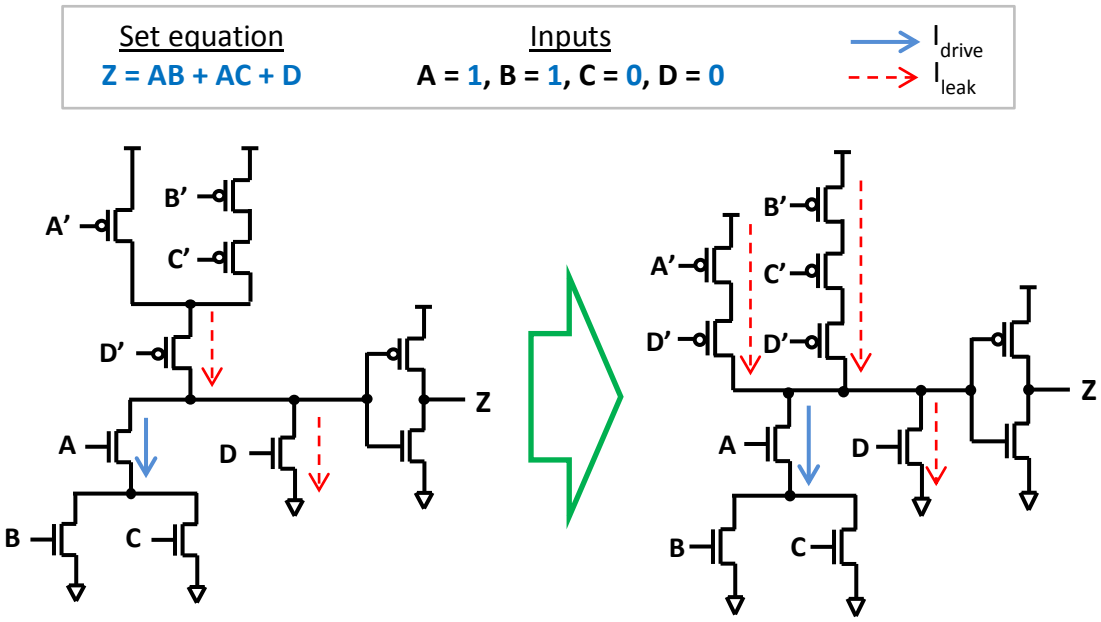


Figure 3.9: Electrical current branches in an unbalanced (left) and balanced (right) circuit.

3.3 Subthreshold-Radhard Designs

Keeping the balance between subthreshold operation and fault tolerance is a challenging requirement for low-power space applications. While subthreshold approaches are intended to keep power consumption as low as possible, the robustness is at stake. On the other hand, common fault tolerance methods implicate redundancy, which results in power increment. For the best knowledge of this thesis author, a minimum number of research contributions that focus in radiation hardening design with subthreshold considerations exist. Consequently, merging these two subjects results in not just an attractive but crucial research topic.

As stated in [54] when reducing the supply voltage to sub V_T in circuits, not just the driving current is reduced but the Q_{crit} also decreases affecting the fault tolerance of the system.

The next considerations were adopted in [54 - 56] for the design of a radiation hardened flip-flop optimized for subthreshold operation: 1) redundant data needs to be stored in two different nodes (one node from the original circuit, and the second node from the redundant copy of the original circuit), and 2) the circuit is designed with limited number of stacked transistors. The first criterion is to provide a fault tolerance to the circuit. As explained before, redundancy enforces the strength of the logic in a circuit. The second condition is considered for avoiding the reduction of drive current strength in subthreshold regimes. Figure 3.10 illustrates the radhard latch used in the proposed flip-flop design, which after testing showed robustness in terms of critical charge of 75 fC working with 250 mV.

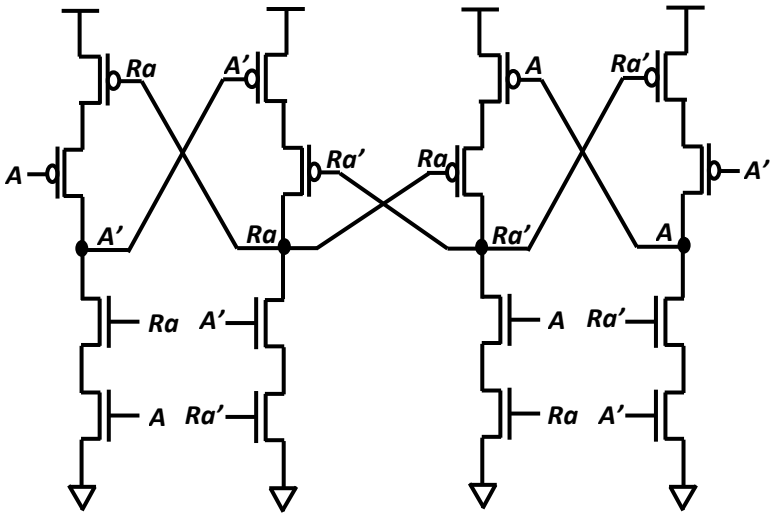


Figure 3.10: Radhard latch proposed in [54].

Another example of a circuit that combines fault tolerance with subthreshold operation was presented in [57]. Improving the structure of Dual Interlocked Storage Cells (DICE) a SRAM that operated in a range of 0.3 V to 0.6 V was designed in [57] with 90 nm CMOS technology. A logic analysis was executed in the structure of DICE in order to identify transistors that might be removed without affecting the robustness to reduce power consumption. Likewise, dummy cells were implemented for generating a sense clock based on the largest time

delay. As explained in [57], a scheme based on worst case operations lead to the elimination of negative effects on standby leakage current and increase reliability. Thus the optimization of power consumption in this radhard approach was achieved by 1) examining the situations in which the most power is consumed, then 2) re-designing the structure of the circuit to eliminate unnecessary redundant elements while keeping robustness, and lastly 3) comparing the tradeoffs between area increment and robustness according to the low-power application requirements. Having a very robust design that wastes too much energy, or having an energy-efficient circuit that is not that tolerant to radiation might not be worthwhile.

Chapter 4: Robust Subthreshold Asynchronous Logic

As described in the previous chapters, subthreshold voltage operation in CMOS circuits represents if not the best, one of the most efficient approaches to reduce dramatically the power consumption. One disadvantage is that when a circuit is restricted to operate with low voltage supply, that is set under common threshold voltages ($V_{dd} < V_T$) depending on the used process technology, this suffers extreme delay variations. Fortunately, an extensive foundation about why the philosophy of asynchronous logic design turns out to be an optimum solution for handling delays has been expounded (discussed previously in Section 2.1). The concern arises when low-power asynchronous circuits are considered to be implemented in space applications. Due to the reduced voltage, the vulnerability of circuits to SEU increases proportionally to the reduction of critical charge – Q_{crit} –. Considering this scenario, a novel methodology to increase the fault tolerance in asynchronous circuits that operate in the subthreshold domain is next described.

4.1 Methodology

The proposed methodology for designing asynchronous logic with increase fault tolerance and optimized for subthreshold operation can be summarized in the next three key steps:

1) Selecting the type of asynchronous domain (e.g., Speed-Independent, Delay-Insensitive, Quasi-Delay-Insensitive) and its correspondent logic design approach.
2) Implementing low-power techniques that let the selected asynchronous approach to operate under threshold voltage regimes.
3) Apply robustness methods that increase the fault tolerance of the asynchronous circuits in presence of radioactivity.

4.1.1 Asynchronous Configuration

Considering the portability of the asynchronous logic design methodologies presented in the background section of this thesis, NULL Convention Logic™ (NCL) was selected as the base for the design of fault tolerant asynchronous schemes. The aspects that were considered for selecting NCL are:

1) The literature about NCL is well described and simplifies the principles of asynchronous logic for a better understanding.
2) NCL follows the philosophy of Quasi Delay Insensitive (QDI) logic, which is a more feasible approach to the creation of asynchronous logic circuits compared to pure Delay Insensitive (DI) logic. This last requires a more complex analysis to be translated into circuitry.
3) NCL has a complete CMOS gate library composed of 27 fundamental threshold gates and resettable NCL registers that support the creation of any logical system that is typically designed under synchronous criteria.
4) The structured Very-Large-Scale Integration (VLSI) design of NCL gates is simple, for which no additional training is required.
5) Existing commercial and open source Hardware Descriptive Language (HDL) tools can be utilized to simulate asynchronous circuits using existent NCL netlist, as the one provided by the Unified NULL Convention Logic Environment (UNCLE) toolset [58, 59]. An example of a circuit described in Verilog language is provided in Appendix A.

The methodology explained in this thesis is applied to a TH23 threshold gate, which is the most common instance of NCL gate considered in literature for experimentation. Nonetheless, the main reason why this gate was used in this thesis for experimentation is because its logic characterizes one of the major advancements of NCL: the combination of simple Boolean logic gates (e.g., AND gates, OR gates) in a single structure. Moreover, although 22 out of 27 NCL gates have this particular feature, the TH23 gate has the less complex transistor structure, which is easy to design. The inferred input values from this gate are $m = 2$, and $n = 3$,

which means that the gate has 3 inputs from which at least 2 inputs need to be asserted to a high value (e.g., logical 1) in order to assert the output to a high value as well. Moreover, since NCL gates fit the requirements of symbolically complete expressions, or possess input completeness in their logic, the 3 inputs of a TH23 gate have to be de-asserted (e.g., transition from logical 1 to logical 0) in order to de-assert the output.

4.1.2 Subthreshold Considerations

The next procedures were adopted for the VLSI design of CMOS NCL gates in order to enhance their functionality at a subthreshold regime.

1) Keep the minimum number of stacked transistors (i.e. transistors in series) in order to not prolong the intrinsic gate delay, which dictates the maximum frequency of the transistors' operation and is defined by equation 4.1:

$$\tau_g = \frac{C_g V_{dd}}{I_{on}} \qquad (4.1)$$

where C_g is the gate capacitance, V_{dd} is the supply voltage, and I_{on} is the current through ON transistors.

2) Implement the methodology introduced in [51, 52] to balance the topology of gates. Such technique enhances the I_{on}/I_{off} ratio avoiding possible metastability and diminishing contention current when transitioning (i.e., current of pull-up network vs. current of pull-down network).

3) Size the transistors of NCL gates according to their calculated best *logical effort* – g –, which is the ratio of their input capacitance to the input capacitance of a typical inverter driving the same output current. This action enhances the structure of NCL gates to deliver as much current as possible in both super and sub threshold regimes. The electrical mobility (i.e., both electron mobility and hole mobility) of an NMOS device is approximately twice the electrical mobility of a PMOS device with the same W/L ratio (i.e., width/length ratio). In order to compensate this disparity, PMOS devices in a CMOS circuit are sized generally twice bigger than NMOS devices. For instance in a regular

logic inverter, which is formed by transistors that have minimum length L, the width of PMOS transistor – w_p – is two times the width of NMOS transistor – w_n –, this is $w_p = 2 \times w_n$ as shown in Figure 4.1. Thus, this thesis work considers sizing the pull-up network of NCL gates twice bigger than the pull-down network. Besides, the effects of having transistors in series and parallel in terms of resistivity (i.e., parallel transistors add up their width without increasing resistivity, and series transistors add up their length increasing resistivity) are considered when sizing the transistors.

4) Utilize extreme low-power process technology for the fabrication of the NCL gates. The one used for simulation in this thesis is the MIT Lincoln Labs 150nm Fully-Depleted-Silicon-On-Insulator (FD-SOI) technology, which is optimized specially for subthreshold operation [24].

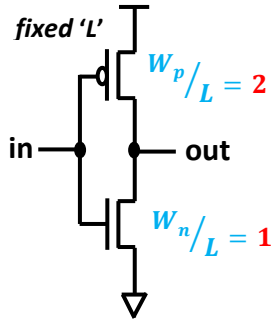


Figure 4.1: Inverter with logical effort g=1.

4.1.3 Fault Tolerant CMOS Design

The methodology to increase the fault tolerance on NCL gates is influenced by a dual-interlocked configuration to shape interwoven-duplicated CMOS structures. The next steps are followed to create radiation hardening CMOS arrangements for a NCL gate:

1) Duplicate the original schematic of the threshold gate.
2) Identify the sensitive nodes in the schematic. For the case of structures of threshold gates the node that drives the output/feedback signal is considered the main sensitive node. The reason is because if this node suffers a SET, the transient signal might be captured by a

subsequent NCL gate in the fanout producing a SEU. This node is formed at the output of the characteristic inverter that every NCL gate has as part of their non-inverted output configuration. Consequently, the node that connects the pull-down and pull-up networks and feeds the input of the mentioned inverter is also considered a sensitive node. The signal coming from this node is referred to in this thesis as the complement of the output signal – *output-bar* –. Figure 4.2 portrays the general structure for NCL threshold gates, which is formed by the *set*, *reset*, *hold-1*, and *hold-0* transistor arrangements. Notice that the hold-1 and hold-0 segments have transistors in series that are triggered by the output signal – *feedback* –. These transistors are referred in this thesis to as the *'feedback transistors'*.

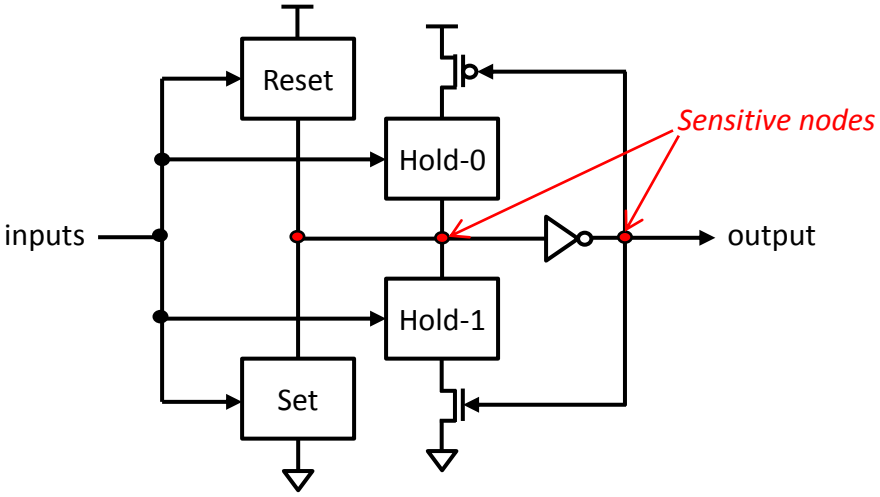


Figure 4.2: General structure of threshold gates with sensitive nodes.

3) Once these four sensitive nodes are identified (i.e., two sensitive nodes from the original schematic, and two from the redundant schematic), the interwoven configuration is implemented as next described. Consider Figure 4.3 as a reference for the explanation.
    a. The output signal of the original schematic triggers the NMOS feedback transistors of its own hold-1 segment (as originally) and the PMOS feedback transistors of the hold-0 segment of the redundant schematic. The same idea applies for the output signal from the redundant schematic.

b. The output-bar signal from the original schematic feeds the PMOS transistor of its own inverter and the NMOS transistor of the inverter that drives the output of the redundant schematic. The same principle applies for the output-bar signal from the redundant schematic.

Figure 4.3a shows the duplicated schematic before applying the interwoven configuration between the original and redundant arrangements of transistors. Then Figure 4.3b illustrates the duplicated schematic with an interwoven configuration, in which output signals Z and ZR are combined to interlace both the original and redundant threshold gates, and the output-bar signals Z' and ZR' intermingle the representative inverters.

This interwoven-duplication introduces dependency and support between the original schematic and its replica achieving a notable increase in the fault tolerance of the resultant robust NCL gate.

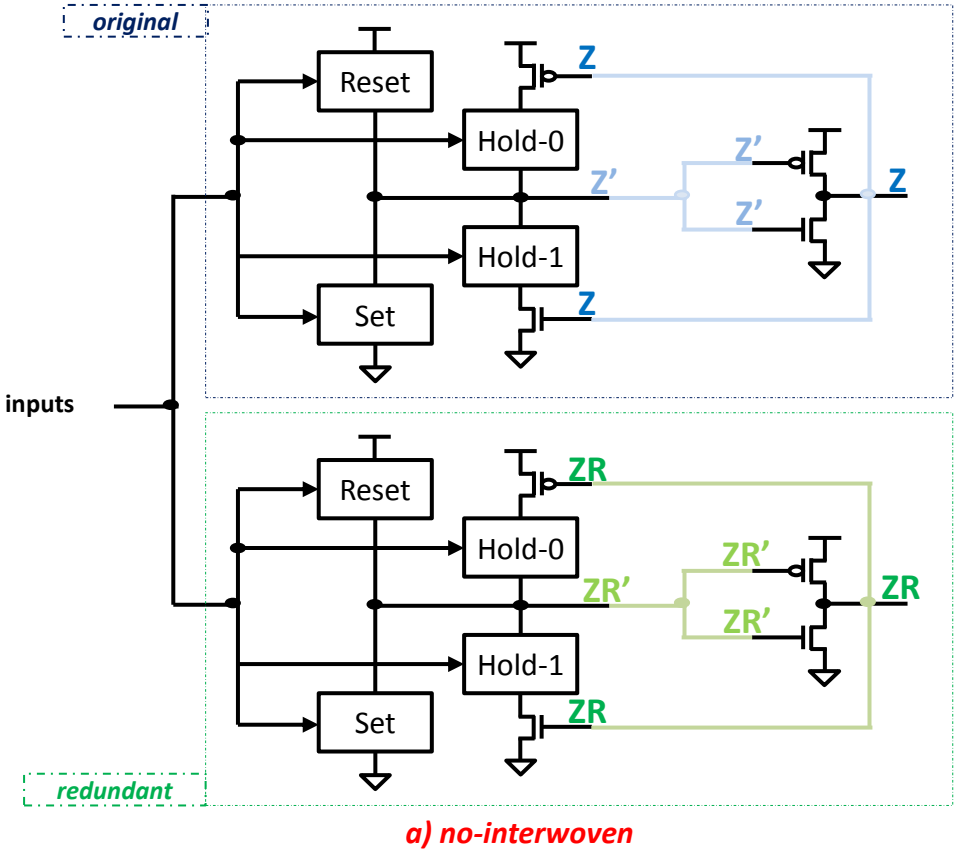


Figure 4.3a: Original NCL gate before being interwoven.

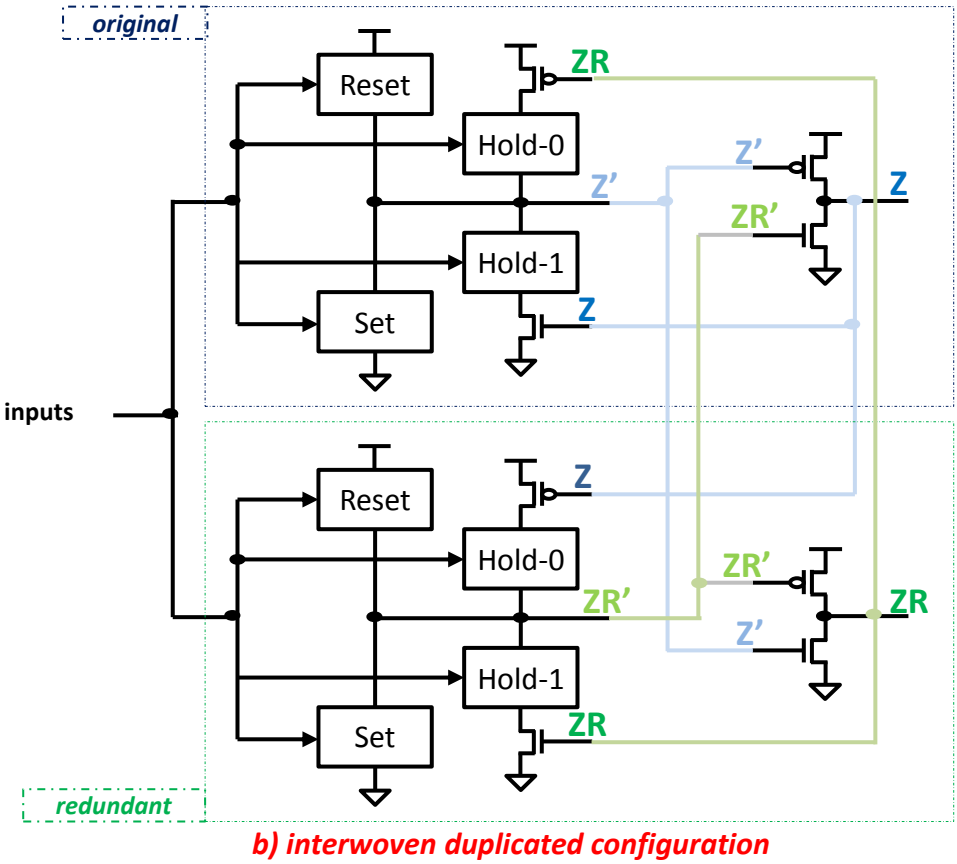


Figure 4.3b: Proposed interwoven-duplicated NCL threshold gate

4.2 Recursive C-element

Taking into consideration techniques that mitigate SEU effects in asynchronous circuits by implementing C-element units to corroborate redundant data, such as the work presented in [43], a proposed C-element is introduced in this thesis with the name of *Recursive C-element* as an effort to increase the fault tolerance in asynchronous circuits that use traditional C-elements. As previously described in section 2.1.1, a basic C-element verifies equality in the state of its inputs to resolute the output state. The difference in the proposed element is that the resolution of the output is not only dependent on the equality of inputs, but also in the equality between the output itself and a redundant output coming from a duplicated scheme. Thus, the logic of the proposed C-element results in a *recursive function*, in which the implementation references itself.

Consequently, the output of the Recursive C-element comes from a doubled logical-value-comparison, which helps to increase the fault tolerance by logical design.

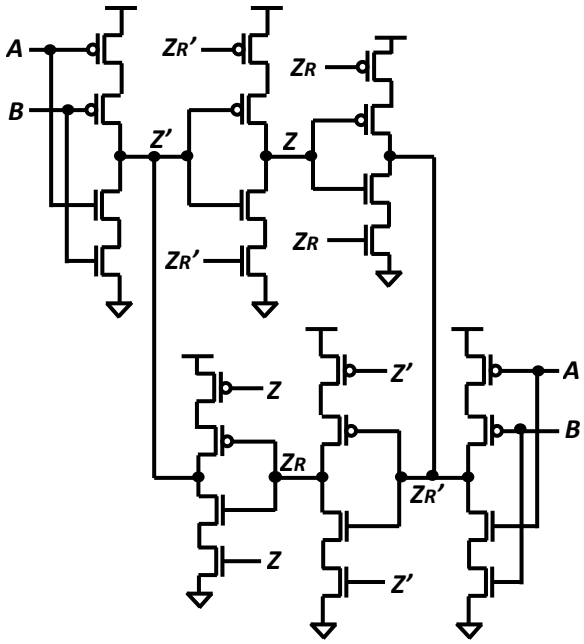


Figure 4.4: Schematic of Recursive C-element.

The schematic of the proposed element is based in the structure of a traditional eight-transistor C-element (see Figure 2.5a); however, the inverters that comprise the feedback alignment are replaced by *two-input* modules (each module is constituted by two NMOS transistors and two PMOS transistors in series), which are used to introduce a recursive logic without losing the feedback configuration. Moreover, the design combines a duplicated scheme with an interwoven configuration, resulting in a mirror robust structure composed of 24 transistors. Figure 4.4 illustrates the proposed CMOS schematic for the Recursive C-element with SEU mitigation properties. In this example, inputs A and B drive two similar transistor arrangements (upper and lower arrangements in Figure 4.4), in which output signals Z and ZR and output-bar signals Z' and ZR' are not just mingled to provide feedback of the opposite arrangement respectively, but also are compared following the same criteria of a traditional

C-element. In other words, the Recursive C-element applies the logic of a regular C-element to its own outputs. Figure 4.5 represents the truth table of the proposed Recursive C-element. With two similar logical interwoven structures, fault tolerance is achieved. If a sensitive node in one of the arrangements is hit by a particle strike, a SET might be produced but immediately mitigated because the logic signals are permanently evaluated twice by the recursive configuration. This operation results in an increased critical charge – Q_{crit} – as later proved in section 5.5.

A	B	Z'	ZR'	Z	ZR
0	0	1	1	0	0
0	1	Hold State	Hold State	Hold State	Hold State
1	0	Hold State	Hold State	Hold State	Hold State
1	1	0	0	1	1

Figure 4.5: Truth table of the Recursive C-element.

Chapter 5: Simulations and Results

The proposed technique was implemented in a TH23 threshold gate to prove the concepts of fault tolerance and subthreshold voltage operation. These properties were analyzed and compared by executing simulations of scenarios that include traditional and proposed TH23 gates working under superthreshold (1.5 V) and subthreshold (0.3 V) regimes. Additionally to the traditional and proposed TH23 gates, an expanded version of the proposed TH23 gate was also considered for testing. This model has more transistors than the other versions because the structure of this extended TH23 comprehends a no simplified configuration. That is, one logic branch (i.e., transistors in series) is required for every single term in the sum of products that characterizes the set, reset, and hold equations. For instance, instead of using only three transistors for the expression A(B+C), four transistors are used to accomplish the expanded expression AB+AC.

5.1 Software Tools

The Virtuoso® Schematic Editor tool from Cadence® was used in this thesis for the design of the gates at a transistor level. The technology that describes the physical properties of the PMOS and NMOS transistors used in the schematics is the MIT Lincoln Labs 150nm XLP CMOS process, which is intended to be used in the design of circuits that operate with extremely low power [24].

The Virtuoso Analog Design Environment (ADE) tool was utilized for setting the simulation environment and generating the input/output waveforms for the transient analysis. The selected circuit simulator in this ADE tool was Spectre®, which provides an accurate SPICE-level simulation. Likewise, for this work, the Euler integration method resulted in a better approximation to the expected results because this method has more tolerance to convergence

issues. For all the simulations in this work the running time was set to 1000 ns. A brief tutorial that shows every step followed in this work to run the simulations is described in Appendix B.

5.2 Single Event Simulation

The particle strike to produce a Single Event in the sensitive nodes was simulated with a double exponential current model approximated with a piecewise linear current source following the alignment described in [30, 54] as depicted in Figure 5.1. The first current exponential – I_1 – was gradually changed by a factor of 0.1 mA for each simulation from a negative to a positive value. The second current exponential – I_2 – also changed but in relation to the variable I_1 since $I_2 = 0.8 \times I_1$. For all the simulations in this work, the applied current was integrated in a period of 2ns and then multiplied by the magnitude of V_{dd} to calculate the total amount of induced charge. A simplified form for calculating the charge is represented in the equation 5.1:

$$Q = \left(\frac{I_1 \times 0.05ns}{2}\right) + (I_2 \times 0.005ns) + \left(\frac{(I_1 - I_2) \times 0.005ns}{2}\right) + \left(\frac{I_2 \times 1.945ns}{2}\right) \qquad (5.1)$$

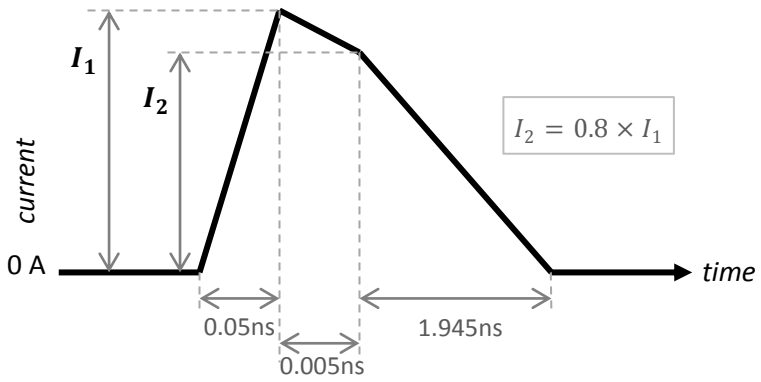


Figure 5.1: Approximation of a double-exponential current model.

Due to the fact that a particle strike can provoke a Single Event Upset (SEU) that results in either a 0-1 bit-flip or a 1-0 bit-flip, positive and negative induced currents were considered in order to determine the worst case scenario that dictates the minimum Q_{crit} value.

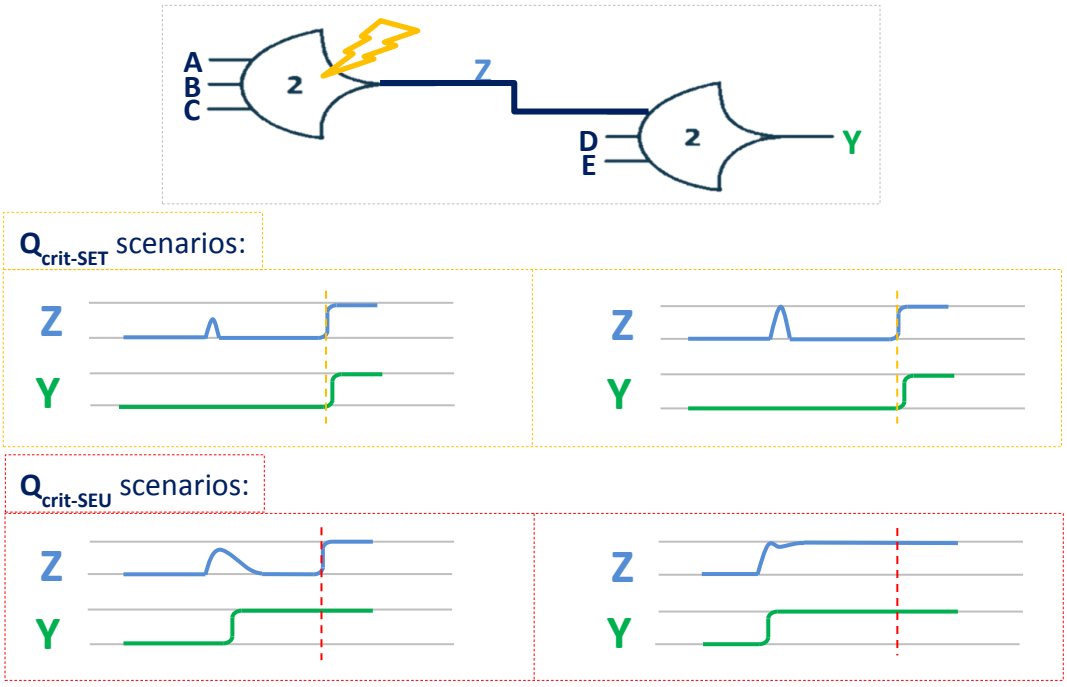


Figure 5.2: SET and SEU scenarios that influence the determination of Q_{crit}.

Determining the critical charge – Q_{crit} – in asynchronous circuits is somehow different than in synchronous configurations because asynchronous are not clock sensitive but edge sensitive. Thus, Single Event Transients (SET) might result in SEU even though their magnitude does not hit the upper voltage level (logical 1) but surpasses half V_{dd}. In view of the types of SEU in NCL circuits described in [44], two different critical charge values were considered in this thesis to determine failure in NCL gates. The first critical charge was expressed as $Q_{crit\text{-}SET}$. This limit value was measured when a threshold gate suffers a Single Event Transient (SET) that is not propagated. This means that the output signal flips for a moment and recovers its original state before the bit-flip is captured by a subsequent gate. In this case, the asynchronous system would not fail. The second critical charge was expressed as $Q_{crit\text{-}SEU}$. This charge is measured at the point that a SET in a gate is propagated. The output signal flips and even though the original state of the output might be recovered, the bit-flip is captured by a subsequent gate provoking a SEU. Consequently, the asynchronous system would fail. Figure 5.2 shows two different

scenarios depicted with waveforms and considered for the measurement of each type of critical charge corresponding to an arrangement of two TH23 gates. Every set of waveforms shows two output signals: 1) 'Z', which comes from the gate that is being tested and 2) 'Y' which comes from the subsequent gate that is used to verify if a SET is captured. Thus, the measured critical charge values would correspond in this example to the TH23 gate with output Z.

5.3 TH23 Threshold Gate Simulations

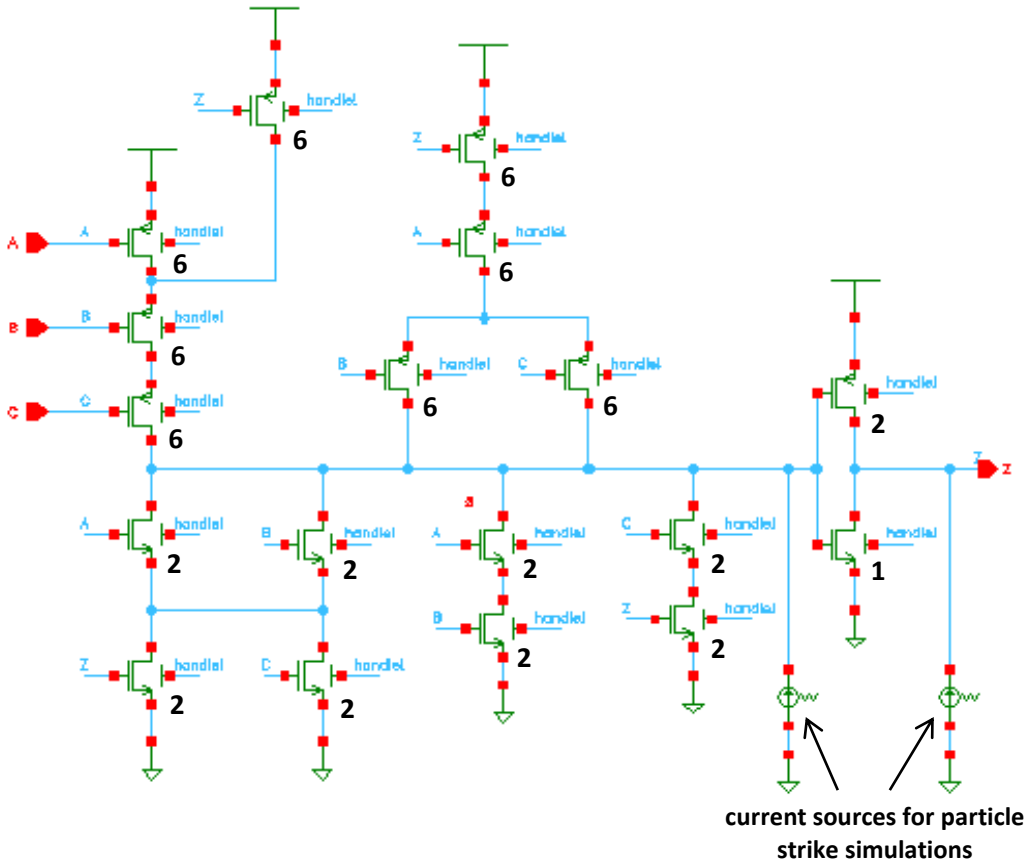


Figure 5.3: TH23 schematic designed in Virtuoso®.

The schematic of the traditional TH23 gate was connected as illustrated in Figure 5.3. The length for all the transistors was set to the minimum size: 150 nm. Since the PMOS/NMOS size ratio was kept at 2/1, the width of the PMOS transistors were calculated to result in a double

sized W/L ratio compared to the W/L ratio of NMOS transistors. In Figure 5.3 the number that is placed next to every transistor represents the factor that multiplies 150nm to determine the width of the transistor. For instance a transistor with number 2 has a width equals to 300 nm ($w = 2 \times 150nm = 300nm$). Notice that electrical current sources are connected to the sensitive nodes in order to simulate particle strikes by inducing charge.

In the same way, the schematics for the proposed TH23 gate and the expanded robust TH23 gate were arranged as shown in Figure 5.4 and Figure 5.5 respectively.

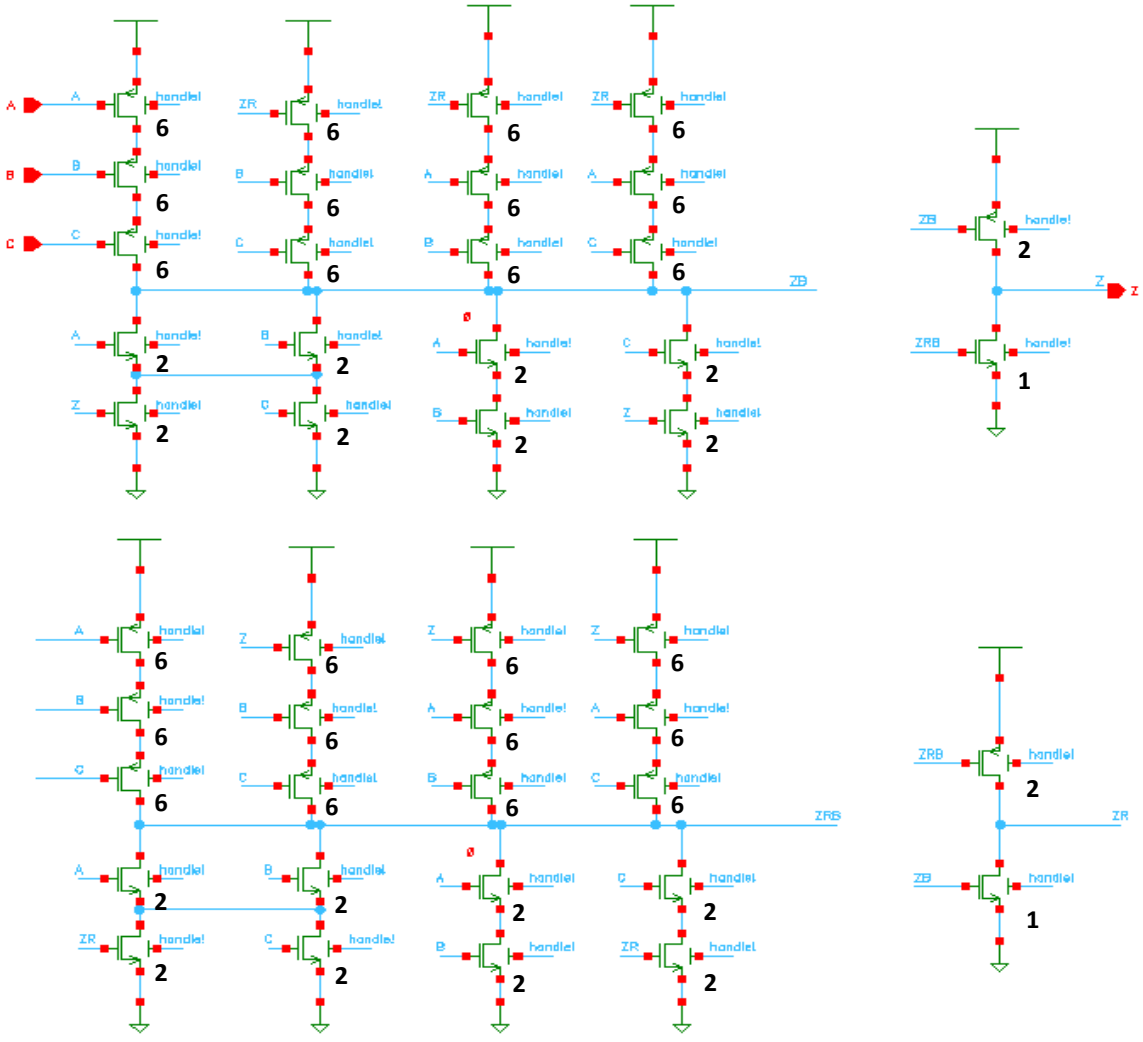


Figure 5.4: Proposed TH23 threshold gate.

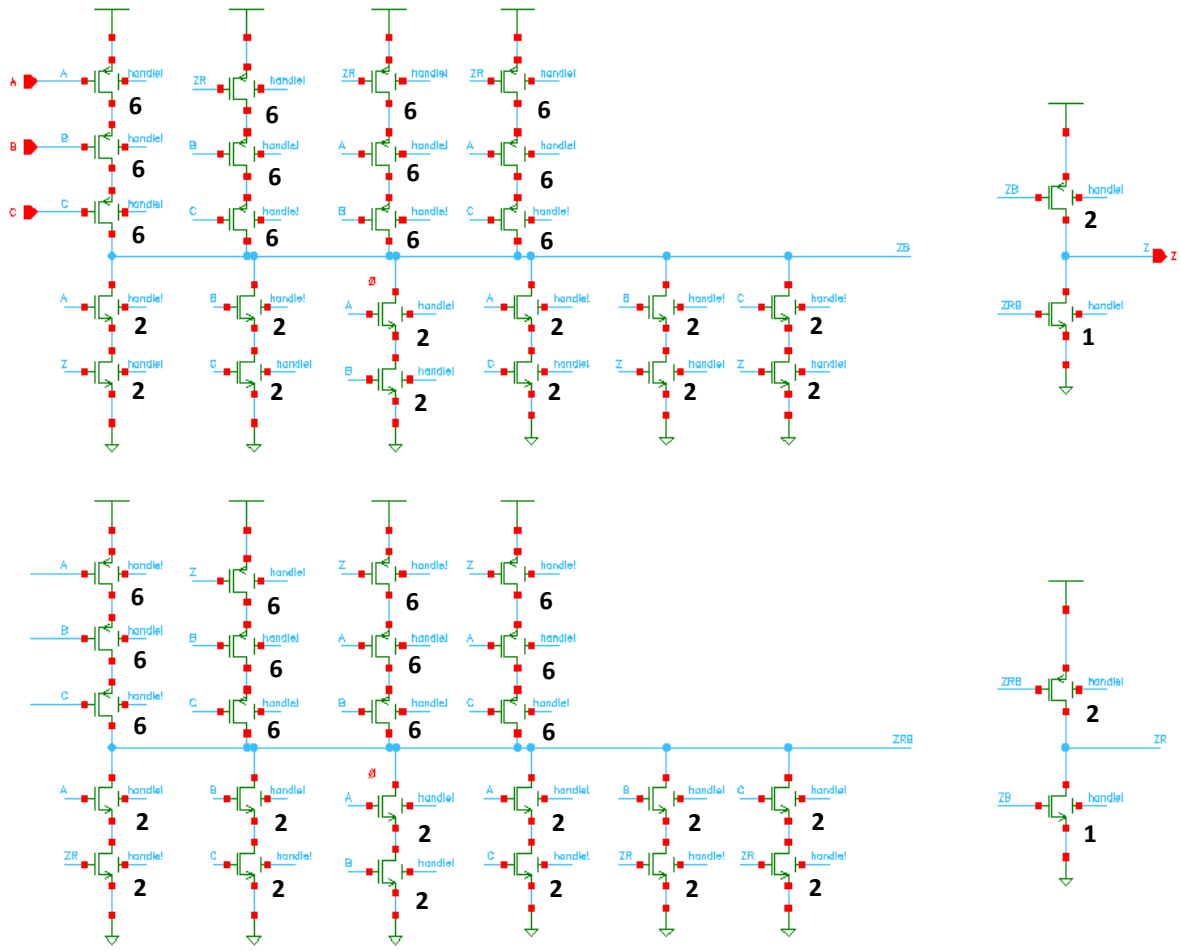


Figure 5.5: Expanded robust TH23 threshold gate.

For testing the functionality of these NCL gates a test bench was set with buffered inputs, which resemble a more realistic environment. This test bench contained the main supply voltage source that was regulated to 1.5 V when performing superthreshold simulations, and 0.3 V when executing subthreshold simulations. Likewise, a voltage source that was set to 0 V was used for driving the body contact of every CMOS transistor and avoiding floating body effects that might result in unexpected results. The signal coming from this voltage source is labeled ‘handle!’ in all the schematics. The voltage sources that were utilized for the inputs were loaded with Piecewise Linear (PWL) Files for a more precise data management. These files dictate the input voltage at specific times that are strategic to leave a control input signal that can transition

directly the output signal. For the case of the TH23 gate, three inputs result in eight different combinations that need to be analyzed to determine the series of input-vectors that represent the more vulnerable conditions in which the gate might fail. Figure 5.6 display the series of input-vectors that resulted in the more vulnerable situation for the TH23 gate in the presence of SET. The circled vectors were considered the crucial combination of inputs in which significant parasitic capacitance is collected in the sensitive nodes. Thus, resulting in the worst case scenarios for calculating the minimum charge needed to flip the output signal. Notice that input signal A was kept high letting either signal B or C to transition the output. Since both $Q_{crit\text{-}SET}$ and $Q_{crit\text{-}SEU}$ needed to be determined, the output signal of the tested gate fed one of the inputs of an extra TH23 NCL gate that helped to examine if a fault in the tested gate was captured. One of the inputs of the extra TH23 gate was set to logical 1, so if a fault was captured the output of the extra gate will be asserted to a high level. Figure 5.7 displays the test bench used for the simulations.

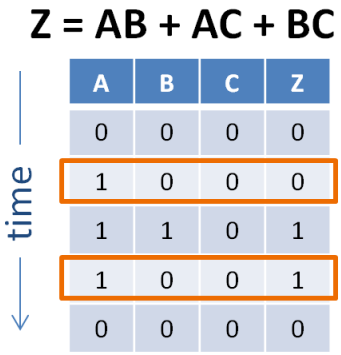


Figure 5.6: Input-vectors that increase the radiation vulnerability of the TH23 gate.

The realized simulations include these next scenarios:

1) Traditional TH23 threshold gate @ 1.5 V.
2) Traditional TH23 threshold gate @ 0.3 V.
3) Proposed TH23 threshold gate @ 1.5 V.

4) Proposed TH23 threshold gate @ 0.3 V.

5) Expanded TH23 threshold gate @ 0.3 V.

6) Expanded TH23 threshold gate @ 0.3 V.

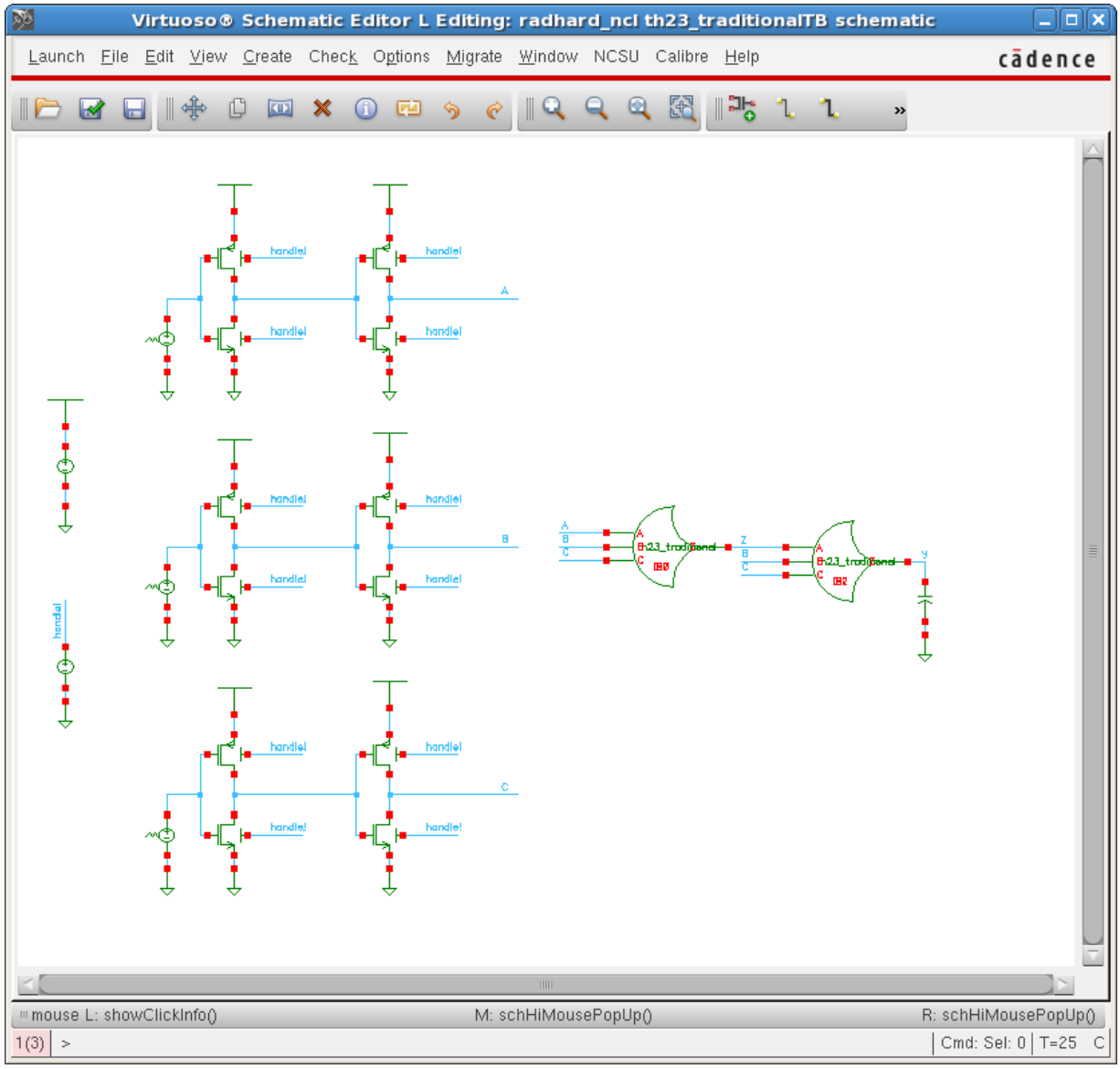


Figure 5.7: Test bench used in the simulations of threshold gates.

5.4 Results

After running every simulation, the resultant waveforms were examined to determine when the gate fails under specific injected current. Figure 5.8 shows the input and output

waveforms of the traditional vs. the proposed TH23 gates that resulted from a superthreshold voltage (1.5 V) simulation in which a particle strike was simulated. The output signal of the traditional TH23 gate transitioned from a logical 0 to a logical 1 state (i.e., 0-1 fault) at the time that the current was injected; however the output signal of the proposed TH23 resisted to this event causing no bit-flip and transitioning until the expected time. Similarly Figure 5.9 displays the failure of the traditional TH23 gate at a subthreshold voltage operation (0.3 V) competing with the proposed TH23 gate, which is more robust to Single Events.

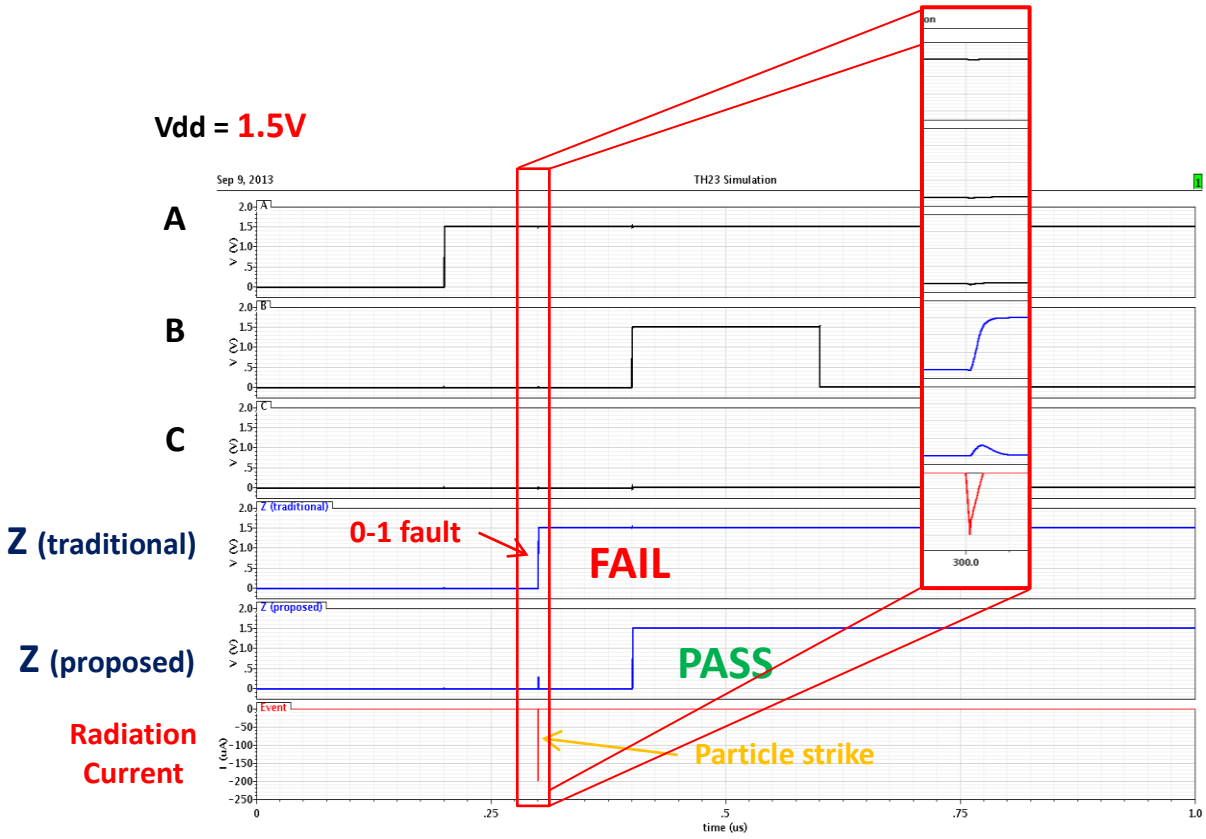


Figure 5.8: Traditional vs. proposed TH23 NCL gates @ 1.5V.

The data collected of every simulation include both values of the calculated critical charge, the energy that was dissipated in a 1000 ns period, the average delay, which was measured from the half of the transition (i.e., half V_{dd}) of the input signal that triggered the output to half the output signal transition. Likewise the size of every gate version was measured

by considering the width of every transistor. These data is presented in Table 5.1 for superthreshold scenarios and in Table 5.2 for subthreshold setups. Such data reveal that the proposed TH23 gate working at a subthreshold regime is around 16 times more robust and 60 times more energy-efficient than a traditional TH23 gate operating with 1.5 V. The penalties to pay are a 160% increase in area and a notable performance reduction; however, for low-power applications that require extreme robustness to radiation the proposed design is more than enough.

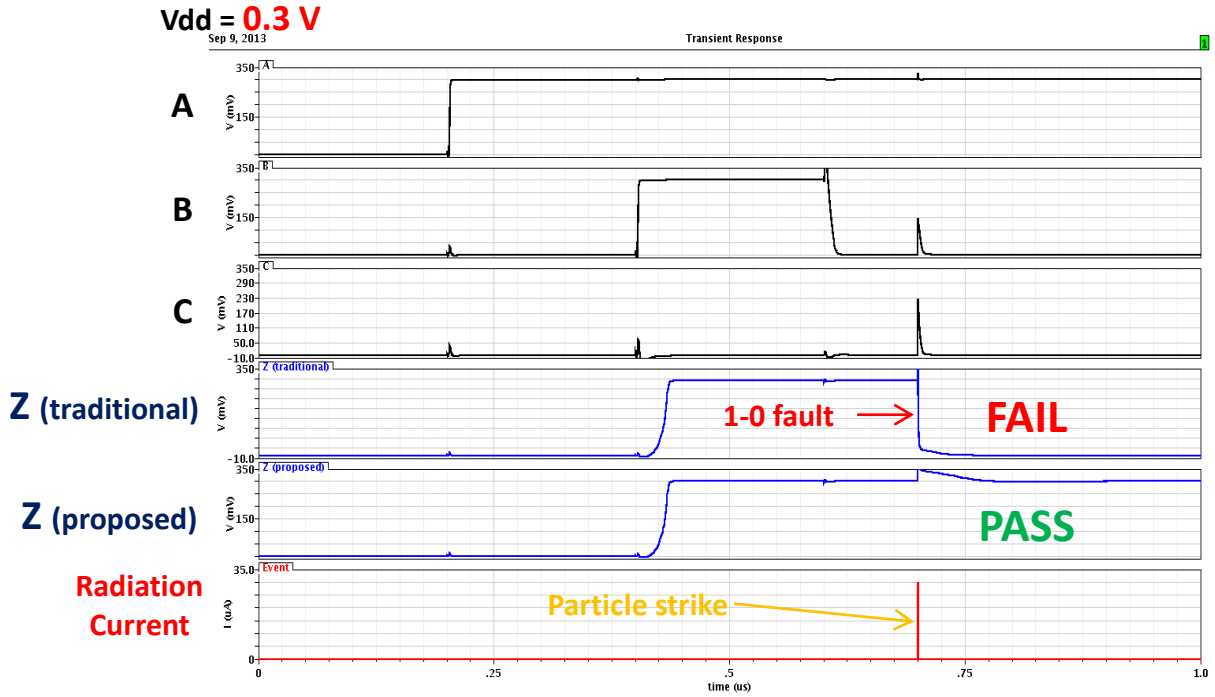


Figure 5.9: Traditional vs. proposed TH23 NCL gates @ 0.3 V.

Table 5.1: Comparison of the results of the three different TH23 gates at 1.5 V.

TH23 @ 1.5 V	$Q_{crit\text{-}SET}$ (fC)	$Q_{crit\text{-}SEU}$ (fC)	Energy (fJ)	Average Delay (ps)	Size (µm)
Traditional	161	**162**	15	891	10.5
Proposed	1760	**>5000 (no-SEU)**	18	983	27.3
Expanded	1995	**>5000 (no-SEU)**	22	1025	30.6

Table 5.2: Comparison of the results of the three different TH23 gates at 0.3 V.

TH23 @ 0.3 V	$Q_{crit\text{-}SET}$ (fC)	$Q_{crit\text{-}SEU}$ (fC)	Energy (fJ)	Average Delay (ns)	Size (µm)
Traditional	14	**15**	0.05	188	10.5
Proposed	19	**2699**	0.24	245	27.3
Expanded	20	**2703**	0.22	267	30.6

The proposed methodology was also applied for the design of these next four threshold gates: TH22, TH33, TH54w32, and THxor0. Tables 5.3 and 5.4 contain the collected results of simulations of traditional gates at a superthreshold regime versus proposed fault tolerant and energy efficient designs at subthreshold domain. This information indicates that in average the proposed designs working in a subthreshold regime offer 4 times more robustness than traditional gates working in superthreshold environments. Likewise, the average saved energy at subthreshold is 37 less than the energy dissipated at superthreshold. The penalty to pay is 120% area increase in average and a notable performance reduction. The schematics of these NCL gates are presented in Appendix C.

Table 5.3: $Q_{crit\text{-}SEU}$ comparison among TH22, TH33, TH54w32, and THxor0 gates.

Threshold Gate	Traditional $Q_{crit\text{-}SEU}$ @ 1.5V (fC)	Proposed $Q_{crit\text{-}SEU}$ @ 0.3V (fC)
TH22	148.6	320.8
TH33	106.4	309
TH54w32	140	800
THxor0	211	1173

Table 5.4: Energy and size data of TH22, TH33, TH54w32, and THxor0 gates.

Threshold Gate	*Traditional Energy @ 1.5V (fJ)*	*Proposed Energy @ 0.3V (fJ)*	*Traditional Size (µm)*	*Proposed Size (µm)*
TH22	38	0.88	4.95	10.5
TH33	38	1	5.35	11.5
TH54w32	55	1.5	8	18
THxor0	71	2	12.15	28.1

5.5 Recursive C-element Simulations and Results

Similar to the simulations of the proposed threshold gates with increased fault tolerance, a test bench that provides realistic conditions by using buffered inputs and output was used to test different versions of C-elements: 1) traditional 8T C-element, 2) 14T C-element with reduced contention current, and 3) proposed Recursive C-element. The supply voltage was set to 1.5 V and each simulation lasted 300 ns in each case in order to compare the results fairly. The width of all the PFET transistors was set to 2 µm and for the NFET transistors to 1 µm, with the exception of the transistors that constitute the feedback inverters, which were constructed with a 1 µm PFET transistor and a 0.5 µm NFET transistor to reduce contention current. The radiation-induced charge was also simulated with a double exponential current model (described in section 5.2), and the current exponentials (I_1, I_2) were gradually changed by a factor of 0.1 mA for each simulation from a negative to a positive value, allowing a comprehensive evaluation to measure the critical charge (Q_{crit}) in each critical node. Fig. 5.10 illustrates the waveforms for the traditional C-element and proposed Recursive C-element with induced charge in a common logic node. The particle strike was simulated at 100 ns in the simulation (right where there was a logic

transition in both inputs – considered as a window of vulnerability) originating in the traditional C-element (red-dotted line) a SEU, which in the counterpart was mitigated by the proposed C-element (green-dashed line). The measured data is collected in Table 5.5. The Q_{crit} of the proposed element was increased for more than 12 times showing no-SEU in the simulations (due to the tool limitations) at the only expense of an approximate 75% and 200% increase in transistor count compared to the 8T and 14T C-element versions respectively, and around 100% increase in gate delay. Thus, the proposed Recursive C-element should be restricted to applications that require a high tolerance to radioactivity without considering area and performance requirements.

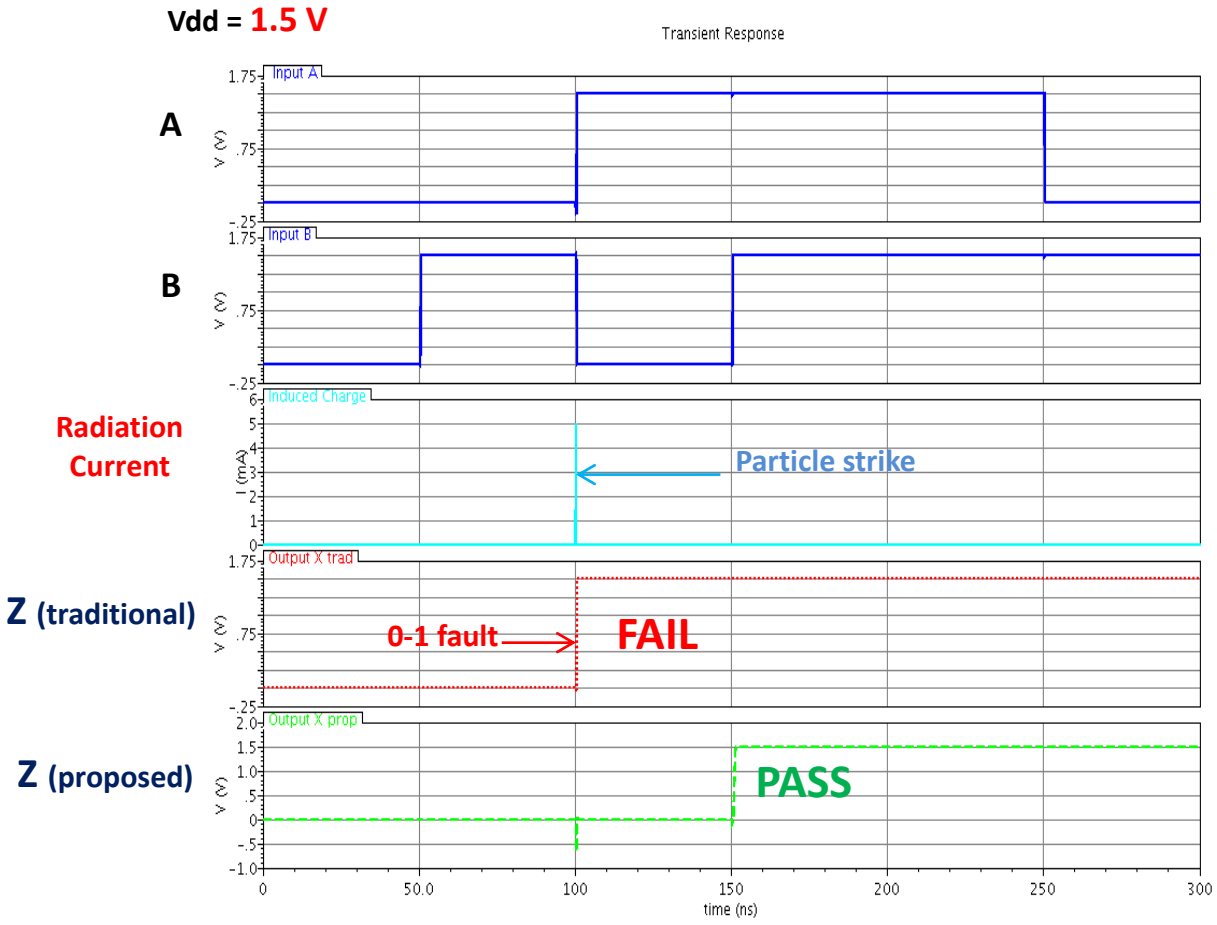


Figure 5.10: Traditional C-element vs. Recursive C-element waveforms.

Table 5.5: Comparison among traditional C-elements and Recursive C-element.

C-element	*Transistor Count*	*Size (µm)*	*Q_{crit} @ 1.5V (fC)*	*Average Delay (ps)*
Traditional	8	10.5	403	148
Reduced Contention Current	14	21	484	121
Recursive	24	48	>5000 (no-SEU)	318

Chapter 6: Conclusion and Future Work

A novel methodology for asynchronous circuit design with increased fault tolerance and optimized for subthreshold operation was presented in this thesis. This approach is intended to be utilized in low-power applications that require not just a reduced energy-dissipation, but also an increased robustness against radiation.

This proposed technique was implemented in NULL Convention Logic^{TM} threshold gates (particularly in a TH23 gate), which are used to design asynchronous circuits. The structure of the threshold gates was modified following a duplicated-interwoven configuration to operate in a subthreshold regime (0.3 V) and tolerate a notable amount of radiation-induced charge before failing (i.e. critical charge – Q_{crit} –). After comparing a traditional version of a TH23 gate powered with 1.5 V versus a proposed version working at subthreshold regime with 0.3 V, the simulations results showed an improvement of 16 times in the fault tolerance of the proposed design. Likewise, the results revealed that proposed threshold gate resulted to be 60 times more energy-efficient; however, the penalties to pay were a 160% area increment and an undesired performance reduction. Nonetheless, such proposed gates are more than suitable for applications that require extreme robustness to radiation and low energy dissipation.

A novel C-element with recursive logic configuration was also introduced in this thesis with the name of *Recursive C-element.* This proposed element is intended to increase the fault tolerance in asynchronous circuits that implement traditional C-elements. The design of this new element follows a duplicated-interwoven configuration to increase radiation robustness. The simulation results showed an increased fault tolerance in the proposed design of 12 times more than a traditional C-element. The penalties to pay were a 75 to 200 % increase in area, and 100% increase in gate delay.

6.1 Future Work

After The future extension for this thesis work is summarized in the next two points:

1) The proposed methodology was applied in 5 out of 27 NCL gates, thus in order to have a complete robust library of threshold gates, the proposed technique must be applied to the rest of the threshold gates. Likewise, the NCL registers must be enhanced to design asynchronous circuits that involve memory management.
2) The physical layout of each proposed NCL gate is needed to be designed. In this way, an instance of an asynchronous logic circuit can be fabricated in Silicon to be tested with radiation. The test can be performed by irradiating the circuit using a cyclotron (i.e., particle accelerator) to generate particle strikes.

References

[1] Borkar, S.; Karnik, T.; Narendra, S.; Tschanz, J.; Keshavarzi, A.; De, V., "Parameter variations and impact on circuits and microarchitecture," *Proceedings of Design Automation Conference, 2003*, pp. 338-342, Jun. 2003.

[2] Calhoun, B. H.; Yu, C.; Xin, L.; Ken, M.; Pileggi, L.T.; Rutenbar, R.A.; Shepard, Kenneth L., "Digital circuit design challenges and opportunities in the era of nanoscale CMOS," *Proceedings of the IEEE*, vol.96, no.2, pp.343,365, Feb. 2008.

[3] Kapoor, A.; Jayakumar, N.; Khatri, S.P., "A novel clock distribution and dynamic de-skewing methodology," *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, pp. 626-631, Nov. 2004.

[4] Friedman, E.G., "Clock distribution networks in synchronous digital integrated circuits," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 665-692, May 2001.

[5] Restle, P.J.; McNamara, T.G.; Webber, D.A.; Camporese, P.J.; Eng, K.F.; Jenkins, K.A.; Allen, D.H.; Rohn, M.J.; Quaranta, M.P.; Boerstler, D.W.; Alpert, C.J.; Carter, C.A.; Bailey, R.N.; Petrovick, J.G.; Krauter, B.L.; McCredie, B.D., "A clock distribution network for microprocessors," *Journal of Solid-State Circuits, IEEE*, vol. 36, no. 5, pp.792, 799, May 2001.

[6] Jackson, M.A.B.; Srinivasan, A.; Kuh, E.S., "Clock routing for high-performance ICs," *Proceedings of Design Automation Conference, 27th ACM/IEEE*, pp. 573-579, Jun 1990.

[7] Muller, D. E.; Bartky, W. S., "A theory of asynchronous circuits", *Proc. Int'l Symp. In Theory of Switching, Part 1*. Harvard Univ. Press: 1959, pp. 204–243. 1959.

[8] Miller, R. E., "An introduction to speed independent circuit theory," *Proceedings of the Second Annual Symposium on Switching Circuit Theory and Logical Design*, pp.87, 93, 17-20 Oct. 1961.

[9] Martin, A.J., "Asynchronous logic for high variability nano-CMOS," *16th IEEE International Conference on Electronics, Circuits, and Systems, 2009*. ICECS 2009, pp. 69-72, Dec. 2009

[10] Sparsø, J.; Furber, S., "Principles of asynchronous circuit design – a system perspective," Kluwer Academic Publishers, 2001.

[11] Hauck, S., "Asynchronous design methodologies: an overview," *Proceedings of the IEEE*, vol.83, no.1, pp.69, 93, Jan 1995.

[12] Sutherland, I. E.; Lexau, J. K., "Designing fast asynchronous circuits," *7th International Symposium on Asynchronous Circuits and Systems, 2001. ASYNC 2001*, pp.184, 193, 2001.

[13] Martin, A.J.; Nystrom, M., "Asynchronous techniques for system-on-chip design," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1089-1120, Jun. 2006.

[14] Fant, K.M.; Brandt, S.A., "NULL Convention Logic^{TM}: a complete and consistent logic for asynchronous digital circuit synthesis," *Proceedings of International Conference on*

Application Specific Systems, Architectures and Processors, 1996. ASAP 96, pp. 261-273, Aug. 1996.

[15] Sobelman, G.E.; Fant, K., "CMOS circuit design of threshold gates with hysteresis," *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, 1998. ISCAS '98*, vol.2, pp.61-64, 31 May-3 Jun. 1998.

[16] Smith, S. C.; Di, J., "Designing asynchronous circuits using NULL Convention Logic (NCL)," Morgan & Claypool, 2009.

[17] Smith, S.C.; Al-Assadi, W.K.; Di, J., "Integrating asynchronous digital design into the computer engineering curriculum," *IEEE Transactions on Education*, vol.53, no.3, pp.349, 357, Aug. 2010.

[18] Shams, M.; Ebergen, J. C.; Elmasry, M. I., "Asynchronous circuits", *John Wiley's Encyclopedia of Electrical Engineering*, pp.716-725, 1999.

[19] Manohar, R.; Martin, A. J., "Quasi-delay insensitive circuits are turing-complete," *Proc. Int. Symp. Advanced Research in Asynchronous Circuits and Systems*, 1996.

[20] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, 1992.

[21] Blaauw, D; Bo, Z., "Energy efficient design for subthreshold supply voltage operation," *IEEE International Symposium on Circuits and Systems, 2006. ISCAS 2006*, pp.4 pp.,32, 21-24 May 2006.

[22] Paul, B.C.; Raychowdhury, A.; Roy, K., "Device optimization for digital subthreshold logic operation," *IEEE Transactions on Electron Devices*, vol.52, no.2, pp.237,247, Feb. 2005.

[23] Akgun, O.C.; Rodrigues, J.; Sparsø, J., "Minimum-energy sub-threshold self-timed circuits: design methodology and a case study," *IEEE Symposium on Asynchronous Circuits and Systems (ASYNC), 2010*, pp.41, 51, 3-6 May 2010.

[24] Massachusetts Institute of Technology: Lincoln Laboratory. (2006). *Low-power FDSOI CMOS process design guide, revision 2006:1* [Online]. Available: http://www.ece.umd.edu/~dilli/research/layout/MITLL_3D_2006/3D_PDK2.3/doc/DesignGuide2006-4.pdf.

[25] Soeleman, H.; Roy, K.; Paul, B.C., "Robust subthreshold logic for ultra-low power operation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.9, no.1, pp.90-99, Feb. 2001.

[26] Martin, A. J., "The limitations to delay-insensitivity in asynchronous circuits," In, *Advanced Research in VLSI: Proceedings of the Sixth MIT Conference*, W.J. Dally, pages 263–278. MIT Press, 1990.

[27] Baumann, R.C., "Soft errors in advanced semiconductor devices-part I: the three radiation sources," *IEEE Transactions on Device and Materials Reliability,* vol. 1, no. 1, pp. 17-22, Mar. 2001.

[28] Texas Instruments Incorporated. (2011). Maher, M. (1994, January). *AN-926 Radiation design considerations using CMOS* logic [Online]. Available: http://www.ti.com/lit/an/snoa254a/ snoa254a.pdf.

[29] National Semiconductor Corporation. (1999). Texas Instruments Incorporated. (2011). Maher, M. (1999). *Radiation owner's manual* [Online]. Available: http://www.ti.com/ww/en/hirel/space/ techdocs.shtml.

[30] Heijmen, T.; Giot, D.; Roche, P., "Factors that impact the critical charge of memory elements," *12th IEEE International On-Line Testing Symposium, 2006. IOLTS 2006.* pp.6, 2006.

[31] Messenger, G. C., "Collection of charge on junction nodes from ion tracks," *IEEE Transactions on Nuclear Science,* vol.29, no.6, pp.2024-2031, Dec. 1982.

[32] Baumann, R.; Hossain, T.; Murata, S.; Kitagawa, I., "Boron compounds as a dominant source of alpha particles in semiconductor devices," *IEEE International Reliability Physics Symposium, 1995. 33rd Annual Proceedings*, pp.297, 302, 4-6 Apr. 1995.

[33] Nicolaidis, M., "Design for soft error mitigation," *IEEE Transactions on Device and Materials Reliability*, vol.5, no.3, pp.405, 418, Sept. 2005.

[34] Calin, T.; Nicolaidis, M.; Velazco, R., "Upset hardened memory design for submicron CMOS technology," *IEEE Transactions on Nuclear Science*, vol. 43, no. 6, pp. 2874-2878, Dec. 1996.

[35] S. M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits Analysis and Design.* New York: NY: McGraw-Hill, 1999.

[36] Calhoun, B.H.; Wang, A.; Chandrakasan, A., "Modeling and sizing for minimum energy operation in subthreshold circuits," *IEEE Journal of Solid-State Circuits* , vol.40, no.9, pp.1778, 1786, Sept. 2005.

[37] Dokic, B.; Pajkanovic, A., "Low power CMOS sub-threshold circuits," *36th International Convention on Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013,* pp.60, 65, 20-24 May 2013.

[38] Soeleman, H.; Roy, K., "Ultra-low power digital subthreshold logic circuits," *International Symposium on Low Power Electronics and Design, 1999*, pp.94-96, 17-17 Aug. 1999.

[39] Lin, T.; Chong, K.; Gwee, B.; Chang, J.S.; Qiu, Z., "Analytical delay variation modeling for evaluating sub-threshold synchronous/asynchronous designs," *8th IEEE International NEWCAS Conference (NEWCAS) 2010,* pp.69-72, 20-23 June 2010.

[40] Schrom, G.; Selberherr, S., "Ultra-low-power CMOS technologies," *International Semiconductor Conference, 1996*, vol.1, pp.237, 246, 9-12 Oct 1996.

[41] LaFrieda, C.; Manohar, R., "Fault detection and isolation techniques for quasi delay-insensitive circuits," *International Conference on Dependable Systems and Networks 2004*, pp. 41-50, Jun. 2004.

[42] Jang, W.; Martin, A.J., "SEU-tolerant QDI circuits," *11th IEEE International Symposium on Asynchronous Circuits and Systems, 2005. ASYNC 2005*, pp. 156-165, Mar. 2005.

[43] Gardiner, K. T.; Yakovlev, A.; Bystrov, A., "A C-element latch scheme with increased transient fault tolerance for asynchronous circuits," *13th IEEE International On-Line Testing Symposium, 2007. IOLTS 07.* pp. 223-230, Jul. 2007.

[44] Kuang, W.; Xiao, E.; Ibarra, C.M.; Peiyi Zhao, "Design asynchronous circuits for soft error tolerance," *IEEE International Conference on Integrated Circuit Design and Technology, 2007. ICICDT '07.* pp.1, 5, May 30 - June 1 2007.

[45] Kuang, W.; Ibarra, C.M.; Zhao, P., "Soft error hardening for asynchronous circuits," *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07*, pp.273-281, 26-28 Sept. 2007.

[46] Lin, T.; Chong, K.; Chang, J. S.; Gwee B.; Shu, W., "A robust asynchronous approach for realizing ultra-low power digital Self-Adaptive VDD Scaling system," *2012 IEEE Subthreshold Microelectronics Conference (SubVT),* pp.1, 3, 9-10 Oct. 2012.

[47] Crop, J.; Fairbanks, S.; Pawlowski, R.; Chiang, P., "150mV sub-threshold asynchronous multiplier for low-power sensor applications," *International Symposium on VLSI Design Automation and Test (VLSI-DAT), 2010*, pp.254, 257, 26-29 April 2010.

[48] Gwee, B.; Chang, J. S.; Shi, Y.; Chua, C.; Kwen-Siong, C., "A low-voltage micropower asynchronous multiplier with shift–add multiplication approach*," IEEE Transactions on Circuits and Systems I: Regular Papers*, vol.56, no.7, pp.1349,1359, July 2009.

[49] Lotze, N.; Ortmanns, M.; Manoli, Y., "A study on self-timed asynchronous subthreshold logic," *25th International Conference on Computer Design, 2007. ICCD 2007*, pp.533, 540, 7-10 Oct. 2007.

[50] Akgun, O.C.; Leblebici, Y.; Vittoz, E.A., "Current sensing completion detection for subthreshold asynchronous circuits," *18th European Conference on Circuit Theory and Design, 2007. ECCTD 2007***,** pp.376, 379, 27-30 Aug. 2007.

[51] Coleman, D.; Di J., "Analysis and improvement of delay-insensitive asynchronous circuits operating in subthreshold regime," *Journal of Low Power Electronics*. American Scientific Publishers, Vol. 6, No. 2. 2010.

[52] Wang, A.; Chandrakasan, A., "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *IEEE Journal of Solid-State Circuits,* vol.40, no.1, pp.310, 319, Jan. 2005.

[53] Chen, J.; Vasudevan, D.; Popovici, E.; Schellekens, M., "Design of a low power, sub-threshold, asynchronous arithmetic logic unit using a bidirectional adder," *14th Euromicro Conference on Digital System Design (DSD), 2011*, pp.301, 308, Aug. 31-Sept. 2 2011.

[54] Chavan, A.; Palakurthi, P.; MacDonald, E.; Neff, J.; Bozeman, E., "Heavy ion characterization of a radiation hardened flip-flop optimized for subthreshold operation," *Journal of Low Power Electronics and Applications*, vol. 2, pp. 1-10, 2012.

[55] Chavan, A.; Dukle, G.; Graniello, B.; MacDonald, E., "Robust ultra-low power subthreshold logic flip-flop design for reconfigurable architectures," *IEEE International Conference on Reconfigurable Computing and FPGA's, 2006. ReConFig 2006.* pp.1, 7, Sept. 2006.

[56] Chavan, A.; MacDonald, E.; Neff, J.; Bozeman, E., "Radiation hardened flip-flop design for super and sub threshold voltage operation," *IEEE Aerospace Conference, 2011*, vol., pp.1, 6, 5-12 March 2011.

[57] Kuande Wang; Li Chen; Jinsheng Yang, "An ultra low power fault tolerant SRAM design in 90nm CMOS," *Canadian Conference on Electrical and Computer Engineering, 2009. CCECE '09*. pp.1076, 1079, 3-6 May 2009.

[58] Reese, R.B., *UNCLE (Unified NCL Environment), Technical Report MSU-ECE-10-001* [Online]. November 2010. Available: http://www.ece.msstate.edu/~reese/uncle/UNCLE.pdf.

[59] Reese, R.B.; Smith, S.C.; Thornton, M.A., "Uncle - An RTL Approach to Asynchronous Design," *18th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2012*. pp.65, 72, 7-9 May 2012.

Glossary

Alpha particle: Positive charged nuclear particle composed of two protons and two neutrons, (pp. 28).

Bit-flip: Faulty change of a logical state, (pp. 27, 30).

C-element: Logic gate in which the output changes to the agreed value of all inputs, otherwise the output holds the state, (pp. 7, 10-12, 55).

Channel: Communication arrangement among logic modules, (pp. 8, 9).

Clock skew: Variation in time of the clock signal arrival to different logic modules in the same logic system (pp. 1).

Collection of charge: Induction by radiation of electron-hole pairs on junction nodes of logic modules, (pp. 27, 28).

Completion detection: Logic configuration that detects and notifies when a logic task is done in order to ensure a correct flow of data, (pp. 7, 21).

Cosmic ray: energetic heavy ions that irradiate uniformly the Earth and penetrate the atmosphere creating cascades of particles, (pp. 29).

Critical path: Longest path that data signals have to travel in a circuit, (pp. 1).

Critical charge, Q_{crit}: Amount of charge needed to cause a bit-flip (pp. 3, 27, 60).

Dose: Measure of radiation energy per mass unit deposited in circuits. Dose is expressed in *rad* or *Gray (Gy)* units, (pp. 26).

Dual-rail: Two logic signals used to represent a single data bit, (pp. 10, 23).

Error Correction Codes (ECC): Software methods implemented in logic systems to detect and correct bit-flips, (pp. 30).

Fanout: Number of inputs driven by a single output, (pp. 13).

Fault tolerance: Resistance to the effects of radiation that cause logical errors such as bit-flips, (pp. 26, 27).

Feedback: Property of a logic gate in which the output signal feeds an input from the same gate to determine the next output state. This property induces the ability to hold the value of logic states – *memory* –, which is the main characteristic of sequential logic gates, (pp. 9, 11, 17, 53).

Fork: Division of wires that originate from the same output and feed different inputs, (pp. 13).

Gray (Gy): Unit of radiation dose, (pp. 26).

Handshaking protocol: Asynchronous coordination among logic modules to ensure a correct flow of data, (pp. 7).

Hysteresis: Auto-dependency of a logic module on its own outputs to resolute its logic state. This term is sometimes used by some authors to refer to a feedback property, (pp. 16, 19).

Input completeness: Property of a threshold logic gate that is completely dependent on all its inputs to determine the output (pp. 14, 22).

Isochronic fork: Assumption of equal wire-delays on a fork, (pp. 13, 14).

Latchup: Electrical event with short-circuit aspect that results from an excessive amount of collected charge, (pp. 27).

NULL state: Logic state that represents no-data in NCL^{TM} circuits, (pp. 11, 14).

Observability: Property of NCL^{TM} circuits that do not produce orphans, (pp. 11, 22).

Orphan: Signal that transitions during a data wavefront in NCL^{TM} circuits without affecting the determination of an output, (pp. 22).

Pull-down network: Arrangement of NMOS transistors in the schematic of a logic circuit that drains the output load capacitance when strategically activated, (pp. 12).

Pull-up network: Arrangement of PMOS transistors in the schematic of a logic circuit that drains the output load capacitance when strategically activated, (pp. 12).

Q_{crit}: Amount of charge needed to cause a bit-flip, (pp. 3, 27, 60).

Quad-rail: Four logic signals used to represent a single data bit, (pp. 10).

Rad: Unit of radiation dose, (pp. 26).

Radhard: Radiation hardness, (pp. 26).

Radiation hardness: Resistance to radiation effects, (pp. 26).

Soft error: Logic fault that leads to the malfunction of a logic system (e.g., bit-flip), (pp. 27).

Threshold gates: Logic gates proposed by NCL^{TM} to design asynchronous circuits, (pp. 27).

Appendix A – Asynchronous 1-bit Full Adder

The next Register Transfer Language (RTL) code is based on a dual-rail configuration for describing a 1-bit Full Adder that is implemented by using the UNCLE library, which describes NCL threshold gates and registers [58, 59].

```
//=========================//
//        1-bit Full Adder NCL        //
//=========================//
// NCL gates' modules
module th13 (y, a, b, c);
        input a, b, c;
        output y;
        assign  #1 y =  a | b | c;
endmodule
module th14 (y, a, b, c, d);
        input a, b , c, d;
        output y;
        assign #1  y = a | b | c | d;
endmodule
module th22r (y, a, b, rsb);
        input a, b, rsb;
        output y;
        reg yi;
        always 2(a or b or rsb)
           if (rsb == 0)                    yi <=  0;
```

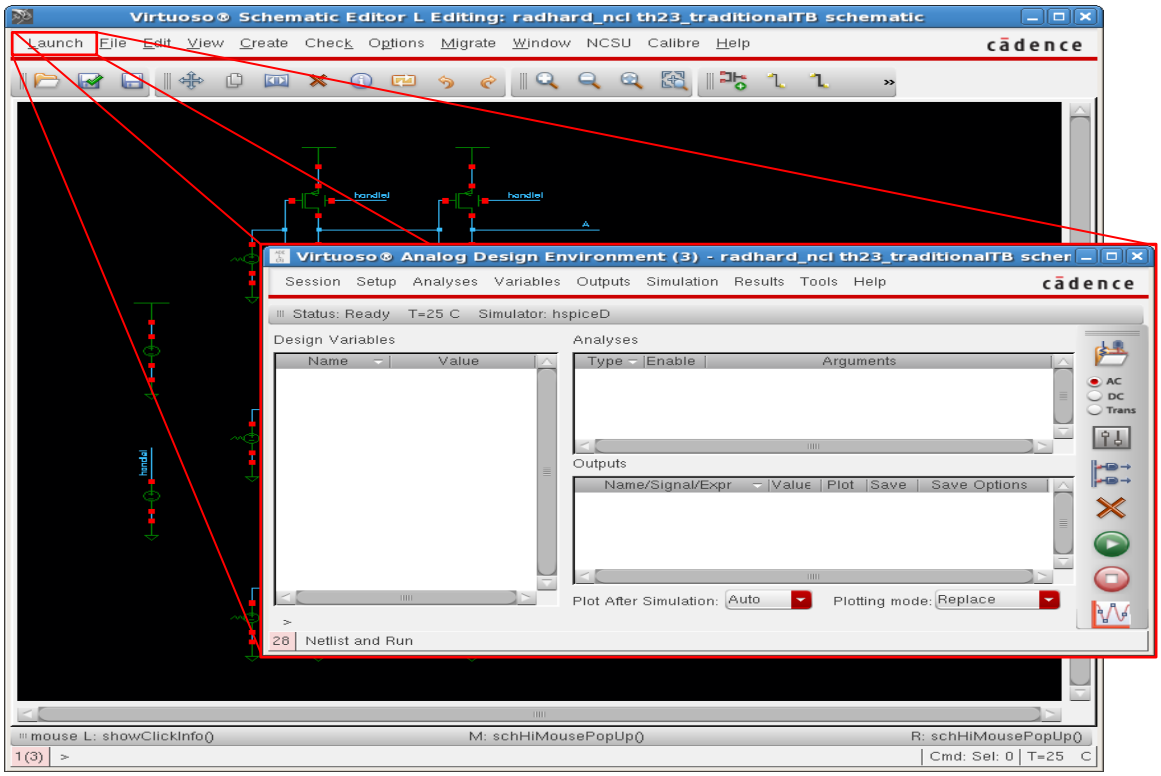
```
        else if (((a) & (b)))                yi <=  1;
        else if (((a==0) & (b==0)))          yi <=  0;
    assign #1 y = yi;
endmodule
module th33r (y, a, b, c, rsb);
    input a, b, c, rsb;
    output y;
     reg yi;
    always 2(a or b or c or rsb)
       if (rsb == 0)                              yi <=  0;
       else   if (((a) & (b) & (c)))              yi <=  1;
       else if (((a==0) & (b==0) & (c==0)))       yi <=  0;
    assign #1 y = yi;
endmodule
// 1-bit Full Adder Module
module onebitadder (A_0,A_1,B_0, B_1, Cin_0, Cin_1, Cout_0, Cout_1, Sum_0, Sum_1, reset);
    input A_0, A_1, B_0, B_1, Cin_0, Cin_1;    // A, B, and Cin dual-rail inputs
    input  reset;                              // To reset the NCL gates
    output Cout_0, Cout_1, Sum_0, Sum_1;       // Sum and Cout dual-rail outputs
    wire ws01, ws02, ws03, ws04;               // for S.1
    wire ws11, ws12, ws13, ws14;               // for S.0
    wire wc11, wc12, wc13;                     // for Cout_1
    wire wc01, wc02, wc03;                     // for Cout_0
```

```
    // Sum_1 connections
    th33r T1 (.a(A_1), .b(B_0), .c(Cin_0), .y(ws01), .rsb(reset));
    th33r T2 (.a(A_0), .b(B_1), .c(Cin_0), .y(ws02), .rsb(reset));
    th33r T3 (.a(A_0), .b(B_0), .c(Cin_1), .y(ws03), .rsb(reset));
    th33r T4 (.a(A_1), .b(B_1), .c(Cin_1), .y(ws04), .rsb(reset));
    th14 T5 (.a(ws01), .b(ws02), .c(ws03), .d(ws04), .y(Sum_1));
    // Sum_0 connections
    th33r T6 (.a(A_0), .b(B_1), .c(Cin_1), .y(ws11), .rsb(reset));
    th33r T7 (.a(A_1), .b(B_0), .c(Cin_1), .y(ws12), .rsb(reset));
    th33r T8 (.a(A_1), .b(B_1), .c(Cin_0), .y(ws13), .rsb(reset));
    th33r T9 (.a(A_0), .b(B_0), .c(Cin_0), .y(ws14), .rsb(reset));
    th14 T10 (.a(ws11), .b(ws12), .c(ws13), .d(ws14), .y(Sum_0));
    // Cout_1 connections
    th33r T11 (.a(A_0), .b(B_1), .c(Cin_1), .y(wc11), .rsb(reset));
    th33r T12 (.a(A_1), .b(B_0), .c(Cin_1), .y(wc12), .rsb(reset));
    th22r T13 (.a(A_1), .b(B_1), .y(wc13), .rsb(reset));
    th13 T14 (.a(wc11), .b(wc12), .c(wc13), .y(Cout_1));
    // Cout_0 connections
    th22r T15 (.a(A_0), .b(Cin_0), .y(wc01), .rsb(reset));
    th22r T16 (.a(A_0), .b(B_0), .y(wc02), .rsb(reset));
    th22r T17 (.a(B_0), .b(Cin_0), .y(wc03), .rsb(reset));
    th13 T18 (.a(wc01), .b(wc02), .c(wc03), .y(Cout_0));
endmodule
```

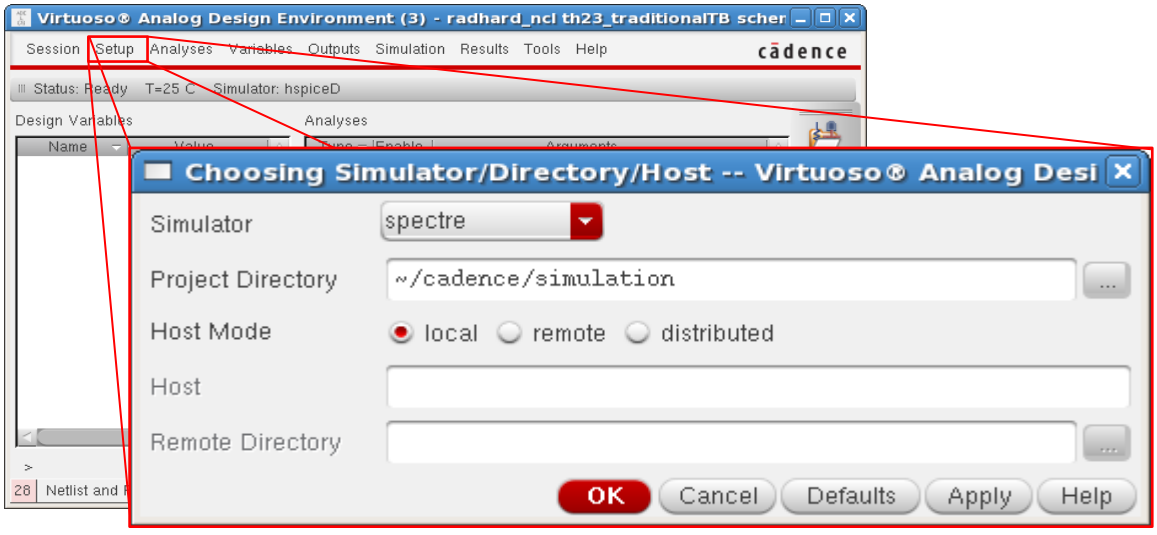
Appendix B – Simulations Setup in Virtuoso®

The next steps are followed in this thesis to setup the simulations in Virtuoso ®.

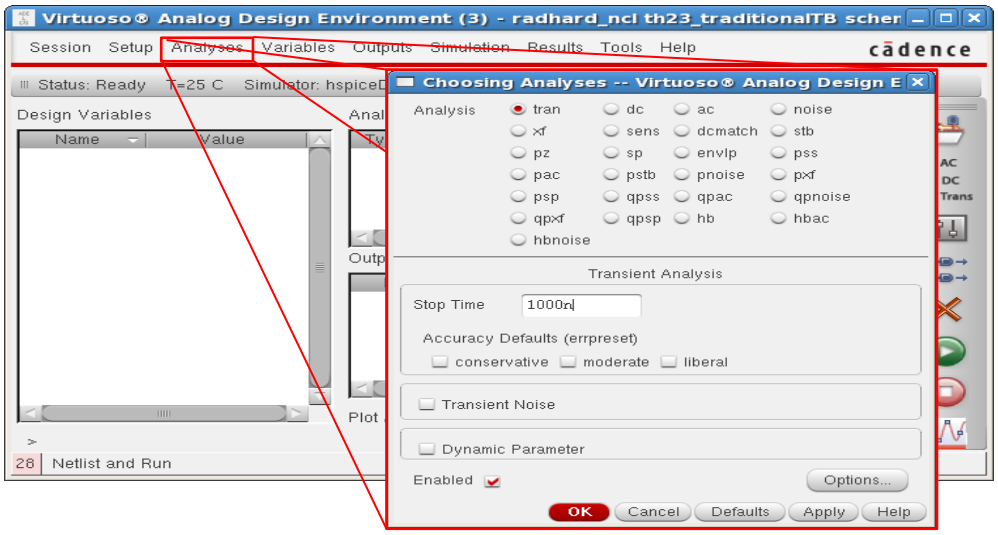
1) Open the Analog Design Environment tool from the menu bar: ***Launch*** → ***ADE L***



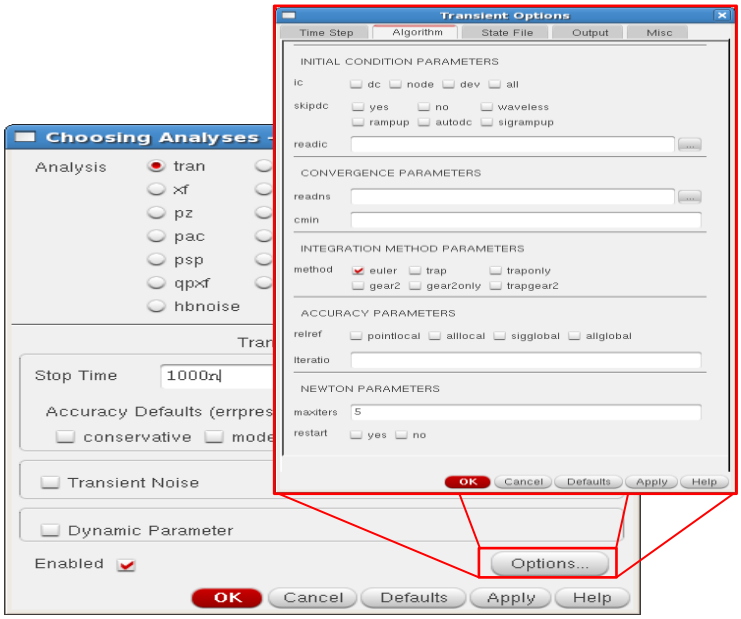
2) When the ADE window appears, select Spectre® as the circuit simulator: ***Setup*** → ***Simulator/Directory/Host…*** From simulator option select ***Spectre*** and click ***OK***.



3) Select the time for the transient analysis: ***Analyses*** → ***Choose…*** When the Choosing Analyses window appears set the stop time that dictates the simulation time (for this thesis the Stop Time is set to 1000 ns), and click ***Options…*** to select the Euler algorithm method: ***Algorithm*** → ***INTEGRATION METHOD PARAMETERS*** → ***euler*** → ***OK***

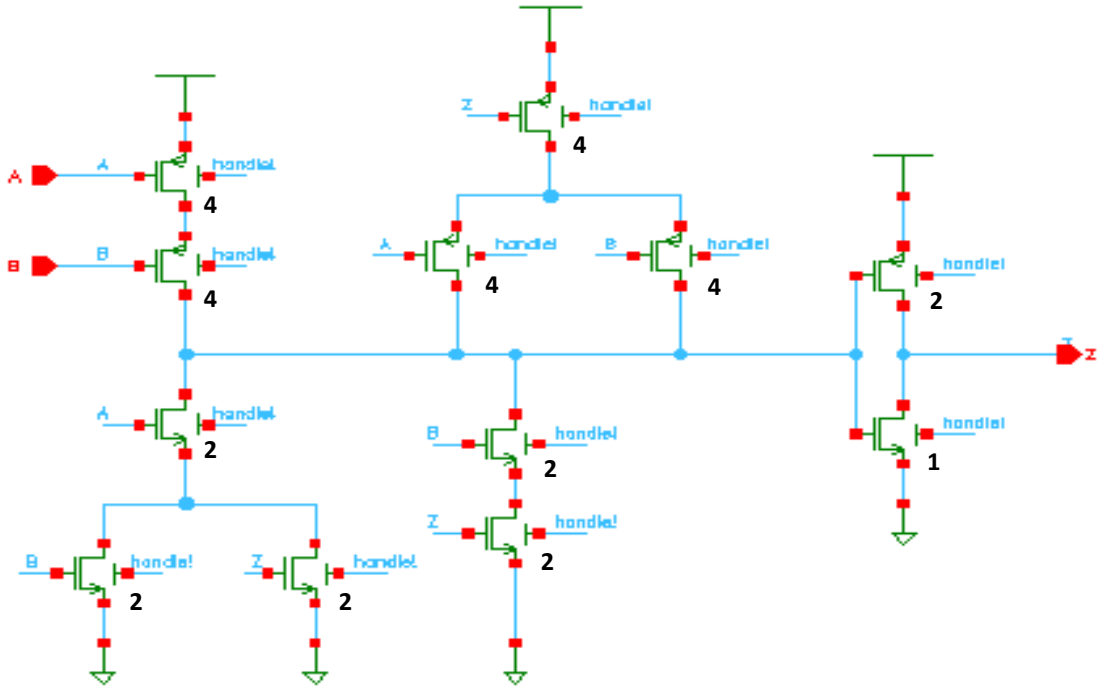


4) Lastly, select the wires that drive the signals to be analyzed during the transient simulation: ***Ouputs*** → ***To be plotted*** → ***Select On Schematic*** After this, the schematic window will come to the front in order to select the wires. Once the wires are selected click the Run icon or ***Simulation*** → ***Run***

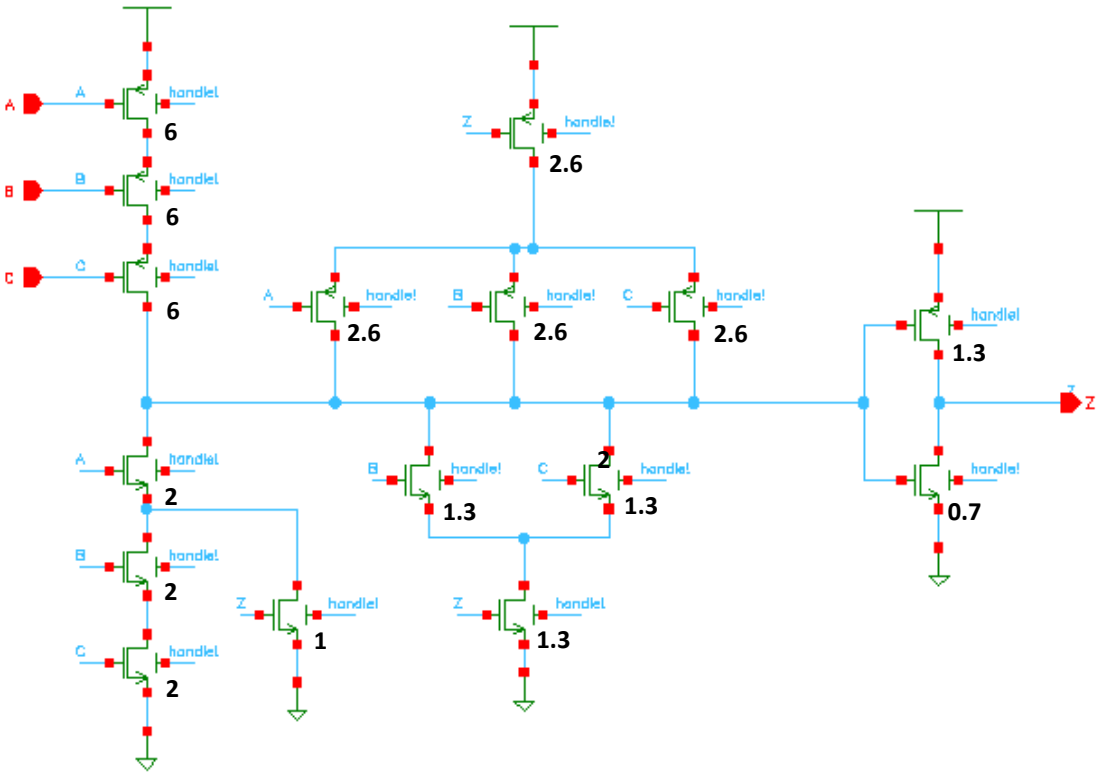


Appendix C – Schematic Instances of NCL Threshold Gates

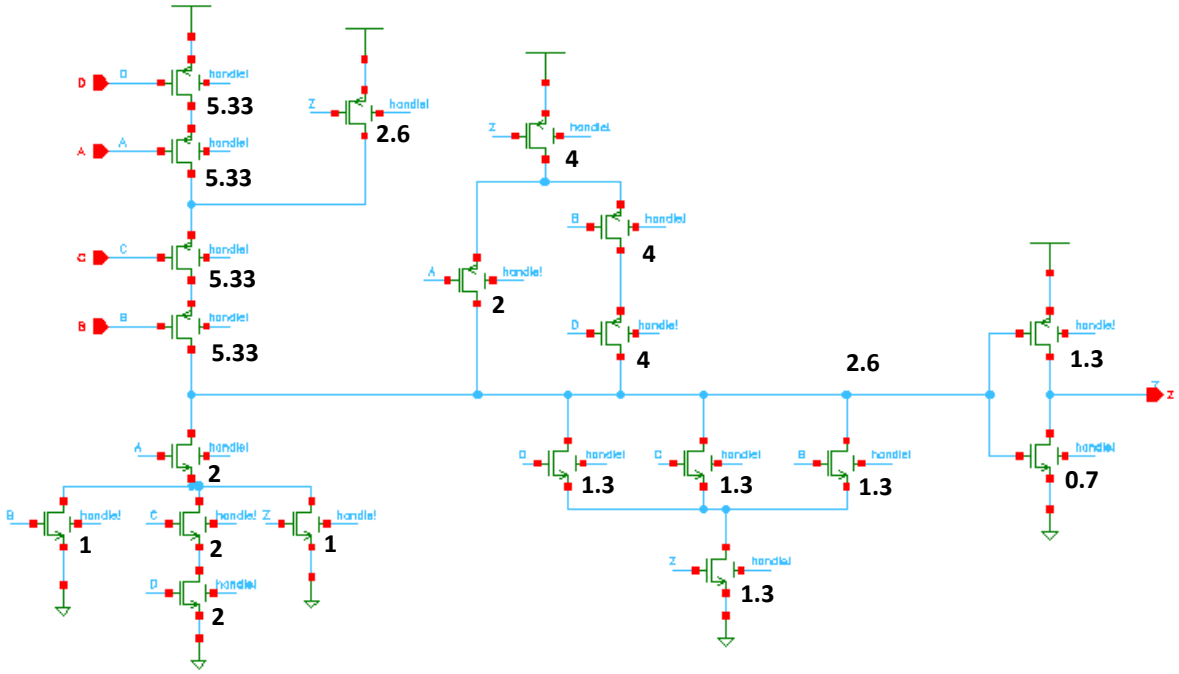
❖ Traditional TH22 threshold gate.



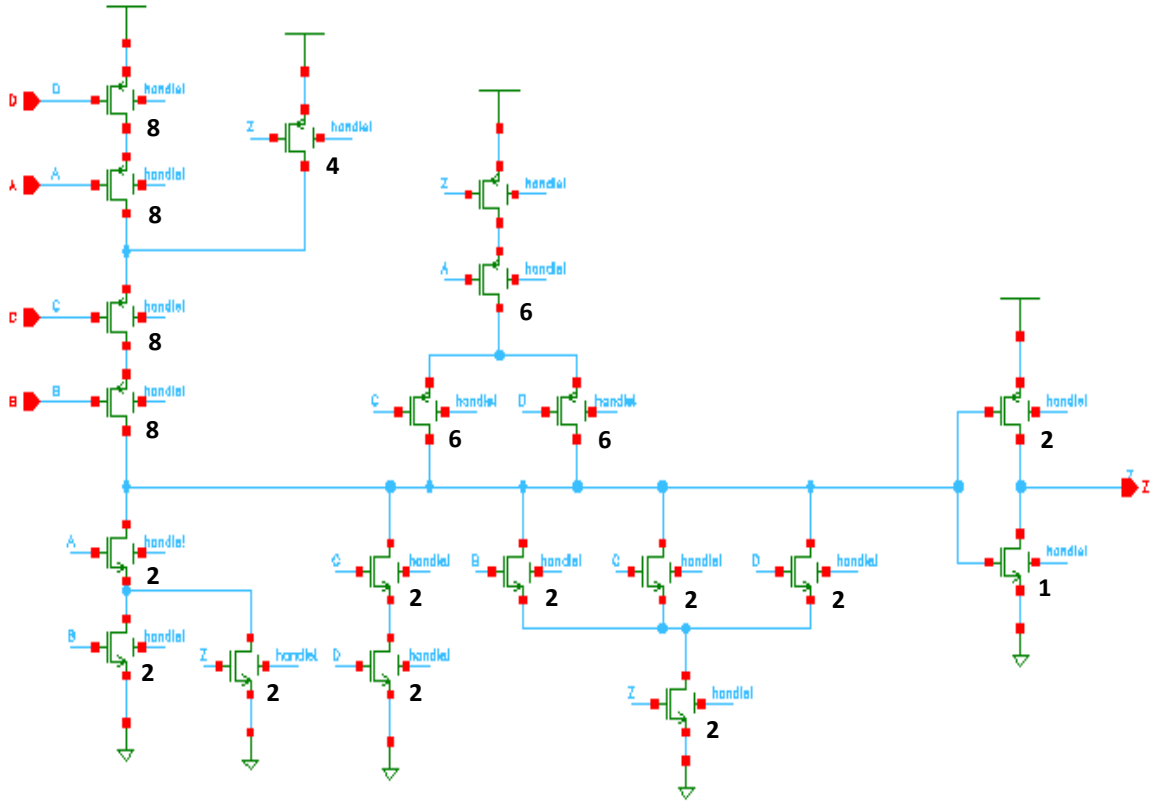
❖ Traditional TH33 threshold gate.



❖ Traditional TH54w32 threshold gate.



❖ Traditional THxor0 threshold gate.



Vita

Ivan Santos was born on Decemeber 20th, 1988 in Juárez, Chihuahua, Mexico. After completing his high school education from Colegio de Bachilleres de Chihuahua Plantel 7 (COBACH 7), Mexico in 2006, he decided to pursue his Bachelor of Science Degree in Electrical Engineering at The University of Texas at El Paso (UTEP). Just before graduating, he participated in a research project related to Micro-Electro-Mechanical Systems (MEMS) at Purdue University during the summer of 2011. He graduated with honors in the Fall of 2011, and determined to stay at UTEP to pursue a Master of Science in Computer Engineering. His research area has been Asynchronous Circuits Design with radiation hardening. Upon graduation, he will be joining a Digital Signal Processors (DSP) design group at Texas Instruments, Dallas, TX, as a Design Engineer.

Permanent address: 11643 Space Shuttle Ln
El Paso, TX, 79936

This thesis was typed by the author.