Charge Sharing Aware NCL Gates Design

Matheus T. Moreira, Bruno S. Oliveira, Fernando G. Moraes, Ney L. V. Calazans
GAPH – Faculty of Computer Science – PUCRS – Porto Alegre – RS – Brazil
{matheus.moreira, bruno.oliveira}@acad.pucrs.br, {fernando.moraes, ney.calazans}@pucrs.br

Abstract— **Interest in asynchronous circuits has increased in the VLSI research community due the growing limitations faced during the design of synchronous circuits, which often result in over constrained design and operation. For designing asynchronous circuits quasi-delay-insensitive approaches are often preferable due to their simple timing analysis and closure. Null Convention Logic (NCL) is a style that supports quasi-delay-insensitive design and enables power-, area- and speed-efficient circuits using a standard-cell methodology. However, the correct functionality of such circuits can be jeopardized by glitches caused by charge sharing effects, which can generate single event upsets. This work scrutinizes the electrical behavior of NCL gates and proposes design optimizations that improve their robustness to charge sharing glitches. Experimental results suggest that the proposed optimizations lead to more robust implementations, increasing fault avoidance and reliability in such circuits.**

Keywords— reliability, charge sharing, null convention logic.

I. INTRODUCTION

Asynchronous, or clockless, circuit design can cope better with inherent problems of current technologies that make synchronous design over constrained, like susceptibility to PVT variations and excessive power dissipation in clock trees. The quasi-delay-insensitive (QDI) clockless design style [1] is attractive for several reasons, but especially because it allows wire and gate delays to be ignored, given that the isochronic fork [1] delay assumption is respected. With QDI, design complexity can be considerably reduced, easing timing closure and analysis. The definition of a specific QDI template requires the choice of a handshake protocol [1] and a delay-insensitive (DI) data encoding [1]. According to Martin and Nyström [2], the 4-phase handshaking protocol coupled to either 1-of-2 or 1-of-4 codes comprises almost the entirety of options in practical QDI design. For such templates, various logic styles support sequential and combinational logic implementations. Of the styles proposed to date, the Null Convention Logic (NCL) [3] is one that enables power-, area- and speed-efficient design based on standard cells. Yet, NCL gates are, in fact, sequential circuits. Thus, single event transients (SETs) can generate single event upsets (SEUs) in many locations within an NCL-based circuit, which can potentially make the circuit to stall, given the local data dependent nature of QDI circuits [1].

Among the causes that may generate SETs, charge sharing appears as an inevitable electrical phenomenon that requires no external interference to occur [4]. It is caused by capacitance coupling and is a classical source of glitches that propagate as SETs. In this work we scrutinize the analog behavior of NCL gates in the presence of SETs caused by charge sharing effects and demonstrate that these gates can be vulnerable to such problems depending on the way their transistors are arranged. We distinguish arrangements that are particularly problematic and can generate SEUs, and propose topological optimizations that can alleviate or even eliminate the problem altogether.

II. NULL CONVENTION LOGIC

NCL was proposed by Theseus Logic, Inc. [3] and has been successfully employed for implementing QDI asynchronous systems on silicon. It is an alternative to other design styles like delay insensitive minterm synthesis (DIMS) [1] and was applied to cope with power problems [5] [6], to design high-speed circuits [7] [8] and to fault tolerant schemes [9], as well as other applications such as ternary logic [10]. One of its advantages is that it enables power-, area- and speed-efficient QDI design with a standard-cell-based approach, while other asynchronous templates require recourse to full-custom approaches. NCL gates are sometimes called *threshold gates*, but this is imprecise. In fact, NCL gates couple a threshold function with positive integer weights assigned to inputs to the use of a hysteresis mechanism. This is required to support DI circuit design using dual rail or 1-of-4 encoding [3].

Without loss of generality, this work restricts attention to NCL gates where all inputs have the same weight (i. e. 1). NCL gates will accordingly also be called M-of-N gates. In such NCL gates, the output will switch to logical 0 when all inputs are at logical 0 and to logical 1 when at least M of its N inputs is/are at logical 1. Otherwise, it keeps its output state. Figure 1 shows the NCL gate symbol, where N is the number of inputs and M is the gate threshold. There are different ways to design NCL gates in CMOS, such as the six approaches that Parsan and Smith discuss in [11]. Among these, classical static implementations are of particular interest, as they are simpler and constitute a good area and power compromise. Accordingly, this work assumes the use of static implementations of NCL gates.

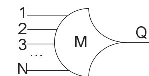


Figure 1 – Symbol of an M-of-N NCL gate.

Figure 2 shows an example, the CMOS schematic of a 2-of-3 static NCL gate, demonstrating the M-of-N general structure. Output "Q" will be logical 0 if the inputs are all at logical 0. If any two inputs are at logical 1, the output will be at logical 1. The transistors of a static NCL gate can be divided in three groups: the Logic Stack, the Feedback and the Output Inverter. The logic stack (here, P0-P2 and N0-N4) defines the NCL gate output to be logical 0 or logical 1, depending only on the input combinations (three logical 0s or two or more logical 1s). The output inverter group (P9 and N9) drives the gate load itself, while feedback transistors (P3-P8 and N5-N8) implement the memory scheme that guarantees output validity and stability for unbounded periods of time, using hysteresis. It consists of a feedback inverter powered only by selected input combinations (two logical 0s and one logical 1). Specific details on the tradeoffs of such implementation can be found in [11].

The authors proposed in [12] a design flow for obtaining NCL gates at the layout level. It automatically dimensions gate transistors and generates timing and power models according to the Synopsys Liberty format, compatible with most electronic

978-1-4799-1585-9/13/$31.00 ©2013 IEEE

design automation tools. The only manual part of the flow is the layout generation itself. Throughout this work, we assume the use of this flow for designing NCL gates.

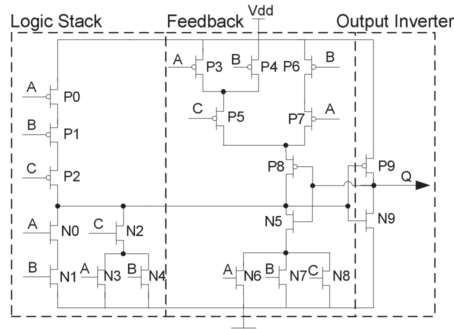


Figure 2 – Example of a 2-of-3 NCL threshold gate [2].

III. Charge Sharing and Related Work

When two electric equipotential regions at different initial voltages are connected together [13] [4], they share charges to reach a new global potential value, causing voltage variations on both regions due to charge redistribution. Such voltage changes in e.g. nodes of a transistor circuit are effects of charge sharing. Also, it is possible to discern two kinds of charge sharing: *pure charge sharing* and *charge sharing with a driven path* [13]. In NCL, the latter is not problematic, as it will only interfere in the output of a gate during output switching, which is the expected behavior. Pure charge sharing, on the other hand, can be hazardous as it will be discussed in Section IV. Therefore, this paper restricts attention to pure charge sharing only. Pure charge sharing occurs, e. g., when two non-driven RC subnets are connected through a switch that is closed. Classically, in dynamic synchronous circuits, like domino logic, charge sharing arises when either charge flows from a gate output to internal nodes capacitances previously discharged, or when charge flows from initially charged internal nodes parasitic capacitances to the output. These phenomena cause glitches on the outputs that, depending on the scenario, may trigger adjacent gates erroneously, producing SETs that can be latched and cause SEUs.

In asynchronous circuits, glitches caused by charge sharing effects can have irreversible consequences. These circuits rely on complex handshake communication protocols that can stall in the presence of a soft error, causing the whole circuit to halt. Also, in asynchronous circuits, SETs are hardly masked [14]. This is due to the fact that there is no temporal assumption and, consequently, no temporal masking. Additionally, logic and electrical masking are very restricted. This is because the majority of asynchronous templates make extensive use of sequential components, which limit the depth of logic paths and of signal regeneration effects and increase the possibility of SETs to be latched and generate SEUs.

Albeit there is a wide variety of works available in the literature that deal with soft errors problems in asynchronous circuits, most of these focus on problems other than charge sharing. In special many works approach soft errors caused by particle strikes. Moreover, most of the works encompass only a special case of NCL logic, namely sets of C-Elements, which represents the particular set of N-of-N NCL gates with $N > 1$ [12]. This is the case of the works presented in [15], [16] and [17]. In fact, as far as the authors could verify, [18] is the only work that report an analysis of the behavior of NCL gates in general under the presence of particle strikes and proposes optimizations. Here, the authors detect weak conditions for NCL gates and propose optimizations by using Schmitt-Triggers and transistor resizing. The presented results suggest that for deep submicron technology nodes, NCL gates are very sensitive to particle strikes, even with their optimizations. Yet, they also propose an efficient method to detect and correct soft errors in computational blocks, assuming that registers are error free. Yet, the use of Schmitt-Triggers and bigger transistors considerably interferes in the performance of the circuit at system level. Also, there is no reference to charge sharing problems that, as it will be demonstrated throughout this work, can also jeopardize the correct functionality of NCL circuits in current technologies. Additionally, differently from soft errors caused by particle strikes, those caused by charge sharing require no external interference and are inevitable in CMOS design [14].

Other works, like [19], address the charge sharing problem on C-Elements. However, their results are valid for this class of restricted NCL gates, which presents a very regular transistors topology, and are all based on older technologies, such as 0.35μm. Besides, the presented results rely on scenarios where C-Elements are simulated in unpractical situations. For instance, they employ C-Elements that are usually not present in any handshake component, which are the components that compose an asynchronous circuit [1], and assume no wire capacitance.

In this context, the present work stands off by performing a close analysis of the analog behavior of NCL gates in general. It also proposes techniques to build gates more robust to charge sharing problems, particularly to avoid SEU generation. This is especially important because, as previously explained, these problems are inevitable for CMOS design and, as it will be demonstrated, if an NCL gate is not properly designed, it can be very sensitive to them.

IV. Charge Sharing on NCL Gates

Recalling Figure 2 static NCL gates employ, as usual in static logic, series and parallel transistor connections. Depending on the threshold and on the number of inputs, more transistors can be required. Complexity may lead to routing congestion due to the existence of many internal nodes when implemented on silicon, which in turn increases parasitic capacitances of the internal nodes of the gate. Also, transistors of the logic stack are usually big, because they are required to drive the output inverter. In this way, their drain and source areas are made larger, which also increases parasitic capacitances. With larger internal parasitic capacitances, the gate is more susceptible to problems caused by charge sharing, as bigger glitches are generated, increasing the possibility of these glitches to be latched and to SEUs.

Albeit there are many different ways of implementing the same logic using the same number of transistors, there is no reported consensus on how to arrange the transistors of an NCL threshold gate. In fact, as far as the authors could verify, no previous work analyses the effects of different arrangements, for designing robust gates. The results showed herein suggest that the correct planning of transistor schematics in NCL gates can generate gates more robust towards charge sharing effects. This paper will employ as example, without loss of generality, the static 3-of-5 NCL threshold gate. Figure 3 shows a first schematic of this gate, called Arrangement 0 (*A0*). The choice of this gate as case study is justified for its extensive use in fundamental blocks usually found in QDI circuits, like adders [12] and for being a critical case for the analysis herein. Unless specified, all transistors in the experiments are assumed to be general purpose standard threshold devices.

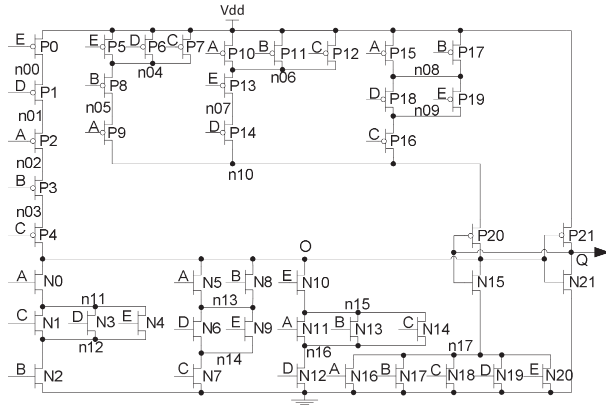


Figure 3 – A0 CMOS schematic for the 3-of-5 NCL gate.

As Figure 3 shows, the PMOS transistors in the logic stack of the 3-of-5 gate (*P0-P4*) can only be arranged in series, to guarantee that all inputs at logical 0 drive the output to logical 0. The capacitances of nodes *n00-n03*, that connect these transistors, are usually not increased by parasitics inherent to metal wiring after layout design. This is because such transistors are likely to be connected by abutting the diffusion of their respective drains and sources. In this way, it is not expected that charge sharing effects caused by these capacitances generate relevant hazardous glitches.

The NMOS transistors of the logic stack, on the other hand, can be arranged in many different forms. Each form can contribute differently to increase glitches caused by charge sharing in the pull-down region. Arrangement *A0*, specifically, is a worst case. For instance, assume the scenario presented in Table I. Initially, all inputs are at logical 0. In this way, nodes *n00-n03* and *O* are charged and the output is set to logical 0 through the output inverter. This is the typical scenario of the beginning of data transmission in QDI circuits. Next, say that inputs *B*, *C* and *D* are set to logical 1, one at a time (time instants 1-3). In this way, nodes *O* and *n11-n16* are discharged and the output is switched to logical 1. This represents data being propagated through the circuit. Now say that all inputs are set back to logical 0, one at a time, starting with *B*, then *C* and then *D* (time instants 4-6), to avoid capacitance coupling of the initially charged nodes *n00-n03* and the discharged nodes *n11-n16*. In this stage, node *O* is charged and all nodes *n11-n16* are kept discharged. This is an initial setup for the analysis of the charge sharing problem. Finally, say that inputs *E* and *A* switch to logical 1, in this order (time instants 7-8). The effect is that when input *E* switches, capacitance of the following pairs of nodes are coupled: *n11* and *n12*, *n13* and *n14* and *O* and *n15*. At this point, a glitch occurs in node *O* as part of its charge is absorbed by node *n15*. However, a larger glitch occurs when input *A* switches, because, in this case, the charge stored in *O* is partially absorbed by capacitances of nodes *n11*, *n12*, *n13*, *n14* and *n16*, which represent a much bigger value. In such a scenario, this glitch can be more easily latched, generating an SEU.

Table I – Input sequence for observing charge sharing effects in a 3-of-5 gate.

Time Instant → Inputs ↓	0	1	2	3	4	5	6	7	8
A	0	0	0	0	0	0	0	0	1
B	0	1	1	1	0	0	0	0	0
C	0	0	1	1	1	0	0	0	0
D	0	0	0	1	1	1	0	0	0
E	0	0	0	0	0	0	0	1	1

This scenario is very common in NCL applications. From the authors' experience, in adders this exact situation or an equivalent one always happens during the computation of each single bit. Additionally, even worse scenarios may occur, if more inputs are assumed to switch, which is also realistic in applications implemented in NCL.

Given the input sequence of Table I, we propose another arrangement for the 3-of-5 NCL threshold gate, namely Arrangement 1 (*A1*). Figure 4 shows its CMOS schematic. The difference from *A0* is that the NMOS transistors of the logic stack were redistributed. Note the transistor parallel is now closer to node *O*. In this way, at each input switch, the charge stored in node *O* is slightly absorbed. This helps avoiding that bigger capacitances accumulate before coupling with node *O*, like in *A0*. For instance, assume that *A1* is submitted to the same initial setup as *A0* (instants of time 0-6). At this point, if any input switches to logical 1, less parasitic capacitances will be accumulating, since critical nodes of parallel transistors will have their capacitances coupled to node *O*. For the example scenario, in the next instant of time (7), when input *E* switches to logical 1, capacitance of node *n11* is coupled to *O*, absorbing part of its charge. In last instant of time, when input *A* switches, less capacitance accumulates and smaller glitches are expected.

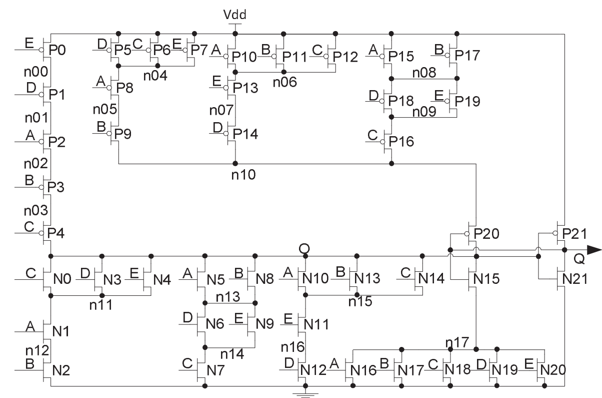


Figure 4 – A1 CMOS schematic for the 3-of-5 NCL gate.

This implementation alleviates glitches generated by charge sharing effects. A bigger capacitance in node *O* would provide further improvements. However, a bigger capacitance in this node can drastically impact the gate performance. In other words, a bigger load would take longer to be charged/ discharged or bigger transistors, which would display bigger power figures. In this context, we propose Arrangement 2 (*A2*), showed in Figure 5. In *A2* the capacitance of node *O* is only increased when the cell is not switching. This is done by placing parallel PMOS transistors of the feedback group closer to *P20*. Here, when the output is at logical 0 and is not switching, capacitance of node *O* is coupled with capacitance of node *n10*, which has an increased parasitic capacitance in *A2*, due to the larger number of transistors connected to it. Also, depending on the combination of inputs set to logical 0, bigger loads are coupled, as nodes *n04-n09* can be connected to node *n10*. When the cell is to switch its output to logical 1, the connection of nodes *n04-n10* to power is cut off, and their charge is drained through the direct connection to ground provided by the NMOS transistors of the logic stack.

For instance, say that *A2* is submitted to the initial setup of Table I. When input *E* switches to 1, at the time instant 7, there is a bigger capacitance coupled to node *O*, and the generated glitch is alleviated. When input *A* switches, at the next time instant, the same effect is observed. However, as soon as another input switches to 1, the connection of feedback group nodes to power source is cut off, lowering their interference in node *O* switching.

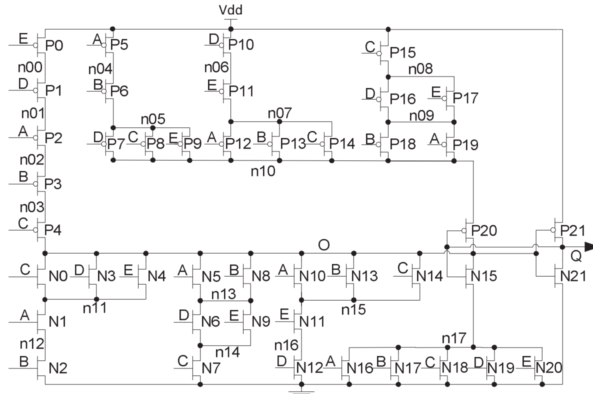


Figure 5 – A2 CMOS schematic for the 3-of-5 NCL gate.

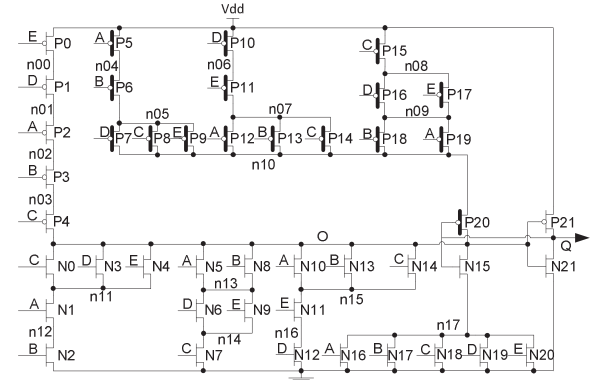


Figure 6 – A3 CMOS schematic for the 3-of-5 NCL gate. Thicker line gate transistors are low threshold.

Another optimization is to employ low threshold transistors in the feedback group. This can be done for multi-threshold technologies only. For deep submicron technology nodes, this is usually available. In this context, we propose Arrangement *A3*. The advantage is that feedback transistors provide faster responses to cut power connections and coupling capacitances. Figure 6 shows the CMOS schematic *A3*, which is very similar to *A2*. In the schematic, low threshold transistors have their gates drawn with thicker lines.

V. Experiments

To analyze the analog behavior of arrangements *A0* to *A3,* each arrangement was described in SPICE using the 65nm STMicroelectronics general purpose standard and low threshold (for *A3*) models. Each of the four arrangements was designed for six different driving strengths, X2, X4, X7, X9, X13 and X18, for a total of 4*6=24 different gates. The transistors of the output inverter group had the same size of those of an equivalent drive inverter of the core library. Transistors of the feedback group are minimum size, a classical approach for designing static standard cells for asynchronous circuits and the transistors of the logic stack had their size calculated according to the flow proposed in [12].

As explained before, SEUs in NCL gates caused by charge sharing are directly related to internal parasitic capacitances. In this way, an experiment was performed to define hazardous parasitic capacitance values. Note that no layout information is available in this phase. The schematic of all designed gates was simulated assuming no wire parasitics, only drain and source diffusion and gate capacitances, to isolate and evaluate the impact of the parasitics on nodes *n11-n16* that could be extracted after layout design. Simulations were performed inserting varying values of capacitances on these nodes and on node *O*, to define safe parasitic combinations limits. Capacitances were varied from 0 fF to 4 fF in 0.05 fF steps. In this way, a total of 80*80=6400 scenarios of parasitic capacitances combinations were simulated. All standard cells had 50 ps input slopes, equivalent to an average size inverter, and output loads equivalent to four inverters of the same driving strength, a common metric known as fanout-of-4 (FO4). The input sequence scenario is that in Table I. During simulations, the peak of the glitch, lowest voltage value, in node *O* was measured after time instant 8 (remember glitches here are from high to low voltage, i.e. negative peaks). Results used typical process models operating at typical conditions (1 V and 25º C). The choice simulator was Cadence Spectre.

Figure 7 summarizes the results obtained for the X2 drive version of each arrangement. In the charts, PU Cap is the parasitic capacitance in node *O* and PD Cap is the parasitic capacitance in each node of the pull down network of the logic stack (*n11-n16*). As Figure 7(a) shows, for *A0*, considering no parasitic capacitance in node *O* (PU Cap=0), PD Cap values bigger than 0.6 fF, generate SEUs. The abrupt fall towards 0 V in the chart represents the critical points, where if bigger PD Caps are present, the glitch generated in node *O* is sufficiently big to cause it to be latched in the memory scheme, switching the output value to logical 1 and the value of node *O* to logical 0.

In other words, for the bottom of the chart (0 V), the glitch generated by charge sharing effects was sufficiently large to be latched and generate an SEU. PD Cap values that are lower than those in the critical point also generate glitches. However such glitches are not high enough to cause an SEU. For instance, for a PU and PD Cap of 0, the peak of the glitch measured in node *O* is of 0.67 V, a glitch of 33% of VDD. However, maintaining the PU Cap of 0, for a PD Cap of exactly 0.6 fF, just before the critical point, the glitch peak is 0.36 V, which is much more critical (64% of VDD) but not enough to be latched. Note that PD Caps of 0 are impossible in practice, especially because node *O* is used for interconnecting many transistor branches. The problem is that, even if PU Cap is raised to 3 fF, or even 4 fF, values of PD Cap close to 1 fF can generate SEUs, which is 3 or 4 times lower than PU Cap.

The values measured for *A1*, showed in Figure 7(b), move the step that represents the critical point to larger values of PD Cap. In this way, more PD Cap parasitics are required to cause SEUs. In fact, in average, for the same PU Cap values, PD Cap parasitics in *A1* must be 45% bigger than those in *A0.* Also, glitches for similar PD Caps, and lower than those of the critical point, are roughly 22% smaller in *A1* when compared to *A0.* As Figure 7(c) shows, for *A2*, these values are improved even more. For the same PU Cap values, PD Cap parasitics in *A2* must be in average 92% bigger than those in *A0* to cause SEUs and glitches for similar PD Caps are roughly 36% smaller. In *A3* (Figure 7(d)) there still more optimization. Comparing to *A0*, this arrangement requires in average PD Caps 98% higher to cause SEUs and glitches for same PD Caps are in average 42% smaller. In fact, as the charts of Figure 7 show, the optimizations proposed move the fall (critical points) steadily to the right, representing arrangements increasingly more robust to charge sharing effects. Another analysis of the same simulation results appears in Figure 8(a). In the chart, critical points (the highest value of PD Cap for each PU Cap before causing an SEU) were isolated for each arrangement. In this way, *A2* and *A3* are almost twice as robust as *A0.*

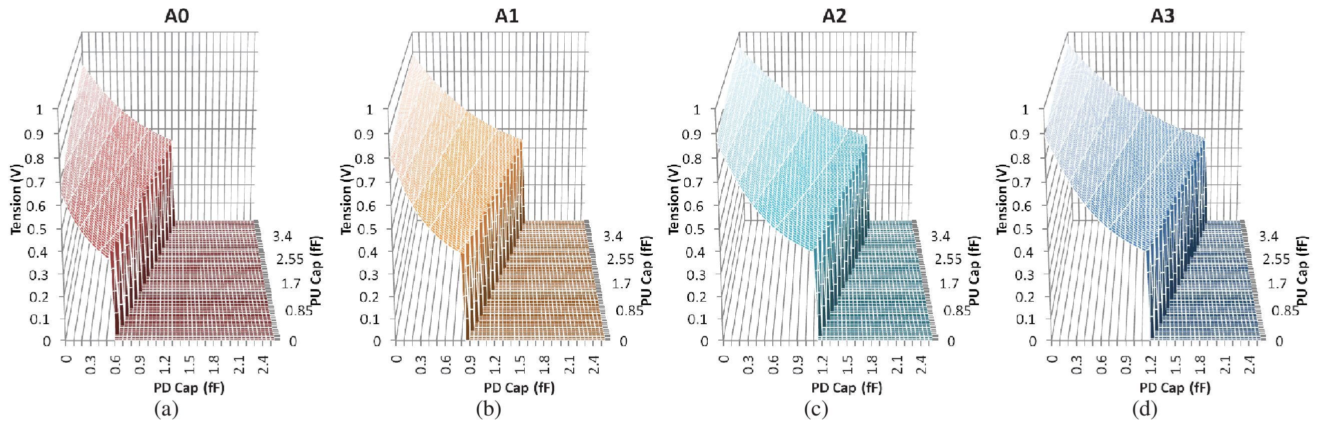
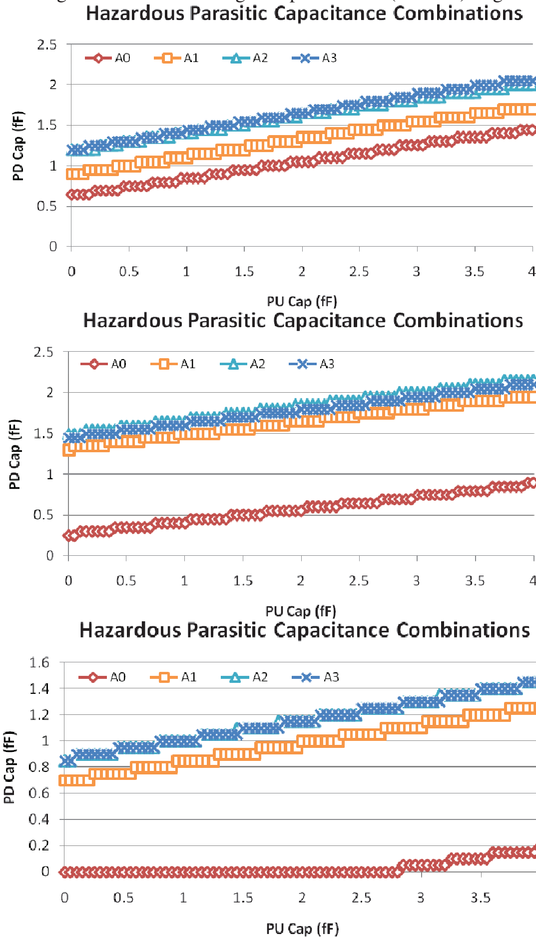


Figure 7 – Maximum negative peak values (in Volts) of glitches caused in node *O* due to charge sharing effects for (a) *A0* , (b) *A1,* (c) *A2, and* (d) *A3*.



(a)

(b)

(c)

Figure 8 – Critical PU and PD Cap for the (a) X2 and (b) X18 drive cells, for a typical process and typical operating conditions, and (c) for the X18 drive for a slow PMOS and fast NMOS process and typical operating conditions.

After analyzing the results obtained for all driving strengths, we noticed that bigger drive cells are more susceptible to charge sharing problems, depending on the arrangement. The difference on the results for different drives is just quantitative, not qualitative. Thus, only the worst case, X18 is analyzed here. Figure 8(b) shows critical PU and PD Cap combinations for this drive for the same scenario above. It is clear that the improvements are much bigger. For instance, in worst case, PD Caps of 0.25 fF are required to generate SEUs for *A0*, while for *A1*, *A2* and *A3*, this value is of 1.3 fF, 1.5 fF and 1.45 fF, respectively. This means arrangements 5.2, 6 and 5.8 times more robust than *A0*, respectively.

The reason why bigger drives are more susceptible to charge sharing effects is the elevated diffusion capacitances of NMOS transistors of the logic stack that contribute to internal parasitics, making lower interconnection parasitics required for causing SEUs. Also these transistors are bigger and discharge node *O* at faster rates. For larger drives, *A2* presents better results than *A3* for all cases. In this way, the multi threshold approach is not better in general to cope with charge sharing in NCL gates. The presented results are all based on typical fabrication processes. However, as variations get critical in current technology nodes, this can be an overoptimistic approach. Accordingly, we simulated all the circuits for the same scenario for a worst case fabrication process variation: slow PMOS transistors and fast NMOS transistors. Thus, bigger glitches are generated in node *O* and less glitches are filtered by the output inverter.

Figure 8(c) presents the results obtained for the most critical case, namely the X18 drive cells. As observable in the Figure, for *A0* no interconnection parasitics are needed to cause SEUs. However, the proposed optimizations push critical PD Cap values to over 0.65 fF. In this way, it is expected that 3-of-5 NCL gates designed according to *A0* for an X18 driving strength generate SEUs when under the scenario shown in Table I. In fact, we detected that this problem starts at driving strengths larger than X7. The degradation in robustness of NCL gates as their driving strength get bigger observed for *A0* is undesirable. In fact, the quality of standard-cell-based logic synthesis relies on the availability of gates with different functionalities and driving capabilities. Under a functional point of view, the more gates available to implement logic functions, the more optimizations are possible. However optimizations in power, area and speed require different driving strengths for each gate during physical synthesis. Thus, it is essential that NCL gates maintain their robustness for increased driving strengths, in order to provide quality standard-cell-based design.

Another perspective on the obtained results shows the observed degradation. Figure 9(a) shows the critical PU and PD Cap combinations for each driving strength. The chart is a steep ramp where results get worse as the driving strength increases. There is a small deviation, in drive X13, which presents better robustness than X9. However this effect is due to transistor sizing. After employing the flow of [12], transistors of the logic stack in

these drives were set to similar dimensions, while transistors of their output inverter are different; in X13 they are bigger. A bigger output inverter leads to a bigger capacitance in *O* for X13 while maintaining the same logic stack nodes capacitance, generating a slightly more robust cell. The steep ramp problem is not observed in the proposed optimizations. As charts of Figure 9(b), (c) and (d) show, *A1*, *A2* and *A3*, present very similar robustness no matter the driving strength. For these arrangements, the ramp observed in *A0* is turned into flatter surfaces, which indicates that these arrangements are suited for increasing the robustness of NCL-based logic, while maintaining the advantages of using a standard-cell-based approach for circuit design.

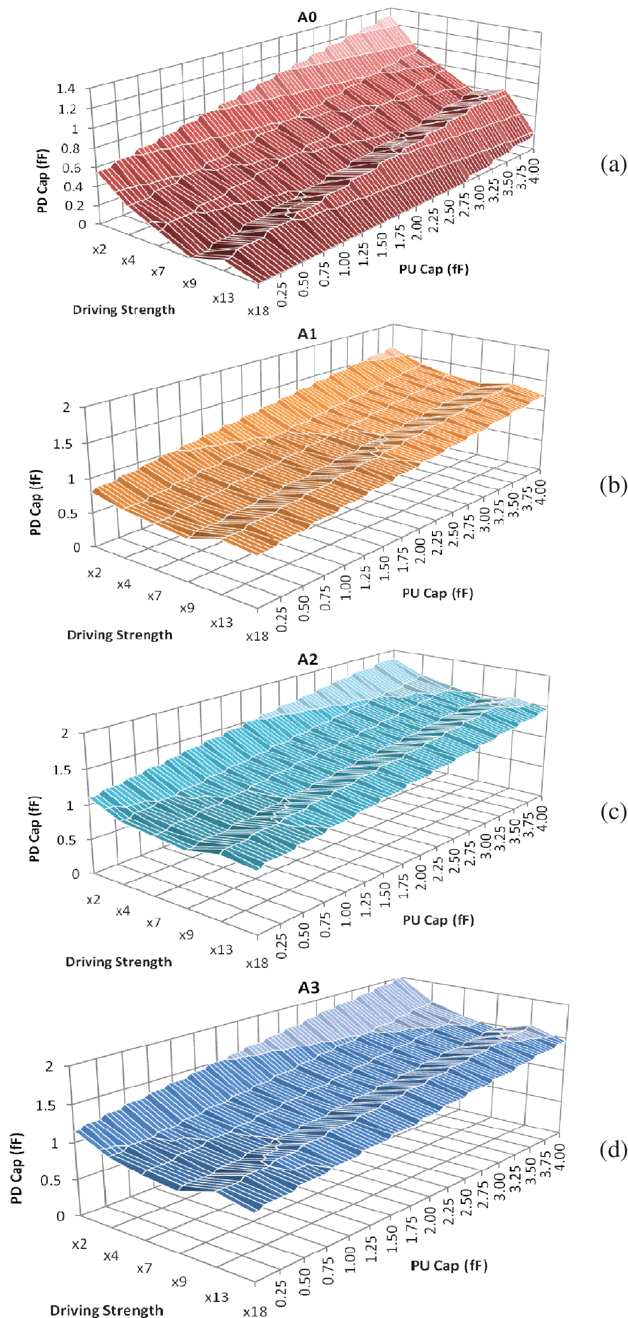


Figure 9 – Robustness of arrangements *A0* (a), *A1* (b), *A2* (c) and *A3* (d).

VI. Conclusions

This work presented an analysis of the analog behavior for NCL gates in the presence of hazards caused by charge sharing effects. Four different transistors arrangements were proposed and better robustness in absolute terms was verified for arrangements *A1*, *A2* and *A3*. In this way, and given that the only difference between the arrangements is the transistor topology (there is no area overhead associated), we strongly recommend the use of *A2* or *A3*. As future works, a tradeoff in terms of operating speed and power will be performed for the proposed arrangements. Also, it is necessary to perform an examination of hazards caused by charge sharing effects in inverted gates, which we believe is a much more severe problem. This is because in these gates the output is right after the logic stack (node *O*). In this way, generated glitches are not attenuated by the regeneration effect provided by the output inverter, but directly propagated to the next gate.

References

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

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

[3] K. M. Fant and S. A. Brandt. NULL convention logic: a complete and consistent logic for asynchronous digital circuit synthesis. In ASAP'96, pp. 261-273.

[4] J. M. Rabaey, A Chandrakasan and B. Nikolic. Digital Integrated Circuits a Design Perspective. Pearson Education, 2003, 761p.

[5] Z. Liang, S. C. Smith and D. Jia. Bit-Wise MTNCL: An ultra-low power bit-wise pipelined asynchronous circuit design methodology. In: MWSCAS'10, pp. 217-220.

[6] G. Xuguang, L. Yu and Y. Yintang. Performance Analysis of Low Power Null Convention Logic Units with Power Cutoff. In APWCS'10, pp. 55-58.

[7] W. Jun and C. Minsu. Latency & area measurement and optimization of asynchronous nanowire crossbar system. In I2MTC'10, pp. 1596-1601.

[8] Y. Yang, Y. Yang, Z. Zhu and D. Zhou. A high-speed asynchronous array multiplier based on multi-threshold semi-static NULL convention logic pipeline. In ASICON'11, pp. 633-636.

[9] F. K. Lodhi, O. Hasan, S. R. Hasan and F. Awwad. Modified null convention logic pipeline to detect soft errors in both null and data phases. In MWCAS'12, pp. 402-405.

[10] S. Andrawes and P. Beckett. Ternary circuits for Null Convention Logic. In ICCES'11, pp. 3-8.

[11] F. A. Parsan and S. C. Smith. CMOS implementation comparison of NCL gates. In: MWSCAS'12, pp. 394-397.

[12] M. Moreira, C. Oliveira, R. Porto and N. Calazans. Design of NCL Gates with the ASCEnD Flow. In: LASCAS'13, 6p.

[13] C. Chorng-Yeong and M. A. Horowitz. Charge-Sharing Models for Switch-Level Simulation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 6(6), 1987, pp. 1053- 1061.

[14] T. Karnik and P. Hazucha. Characterization of soft errors caused by single event upsets in CMOS processes. IEEE Transactions on Dependable and Secure Computing, 2004, 1(2), pp. 128- 143.

[15] R. P. Bastos, G. Sicard, F. Kastensmidt, M. Renaudin and R. Reis. Evaluating transient-fault effects on traditional C-element's implementations. In: IOLTS'10, pp. 35-40.

[16] P. Song and R. Manohar. Efficient failure detection in pipelined asynchronous circuits. In DFT'05, pp. 484- 493.

[17] C. LaFrieda and R. Manohar. Fault detection and isolation techniques for quasi delay-insensitive circuits. In: DSN'04, pp. 41- 50.

[18] K. Weidong, Z. Peiyi, J. S. Yuan and R. F. Demara. Design of Asynchronous Circuits for High Soft Error Tolerance in Deep Submicrometer CMOS Circuits. IEEE Trans. on VLSI'10, 18(3).

[19] S. Mohammadi, S. Furber and J. Garside. Designing robust asynchronous circuit components. IEE Proceedings - Circuits, Devices and Systems, 2003, 150(3), pp. 161- 166.