# Cost Efficient Flip-Flop Designs With Multiple-Node Upset-Tolerance and Algorithm-Based Verifications

Aibin Yan, Yuting He, Zhengfeng Huang, Wenjie Yan, Jie Cui, Xiaolei Wang, Tianming Ni, Patrick Girard, Xiaoqing Wen

# Cost Efficient Flip-Flop Designs with Multiple-Node Upset-Tolerance and Algorithm-based Verifications

Aibin Yan, Yuting He, Zhengfeng Huang, Wenjie Yan, Jie Cui, Xiaolei Wang, Tianming Ni, Patrick Girard, *Fellow*, *IEEE*, and Xiaoqing Wen, *Fellow*, *IEEE*

***Abstract*—This paper presents radiation-hardened flip-flop (FF) designs capable of tolerating soft errors, e.g., single-node upsets (SNUs), double-node upsets (DNUs) and multiple-node upsets (MNUs). First, a 2-input FF and a 3-input FF are proposed as the baseline FFs that not only respectively tolerate SNUs and DNUs but also exhibit cost efficiency in terms of delay, power, and area. Through adding two stages of C-elements, a 4-input FF and a 5-input FF are proposed as the baseline FFs as well. Utilizing the structural characteristics of these FFs, an N-1 input FF and an N input FF are proposed as the extended FFs capable of tolerating more node upsets. Moreover, a highly efficient algorithm for verifying MNU-tolerance of these FFs is proposed. Algorithm and HSPICE-tool-based verification results both demonstrate the MNU-tolerance for the proposed FFs with more inputs.**

***Index Terms*—Flip-Flop, verification algorithm, C-element, radiation hardness, multiple-node-upset**

## I. Introduction

With the advancement of *complementary metal oxide semiconductor* (*CMOS*) technologies, the manufacturing process of *integrated circuits* (*ICs*) has reached the deep nano-scale level, and the performance of modern systems has been considerably improved. However, ICs are increasingly becoming more vulnerable to radiative particles causing soft errors, seriously affecting the reliability of ICs [1]. Common soft errors include *single-node upsets* (*SNUs*) and *double-node upsets* (*DNUs*). However, because of the rapid decrease of node spacing between adjacent nodes in advanced CMOS technologies, the impact of a single radiative particle can lead to *multiple-node upsets* (*MNUs*), including *triple-node upsets* (*TNUs*) and more-node upsets. Therefore, it is crucial to propose and implement advanced *flip-flops* (*FFs*) protected against MNUs to construct highly robust ICs and systems used in safety-critical applications.

In recent decades, many radiation-hardened circuits, e.g., latches [1-3], FFs [4-11], and *static random access memories* (*SRAMs*) [12-13], have been proposed, to improve robustness. This paper focuses on the design innovation of FFs, aiming to mitigate MNUs. Note that, a preliminary version of this paper has been proposed [5], in which the slave latch includes a keeper to keep the output stable. However, for high-performance applications, the keeper is redundant leading to extra overhead. In existing designs, *C-elements* (*CEs*) are widely used. Figure 1 shows the *transmission gate* (*TG*) and the adopted implementation of a 2-input CE. The clock-gated CE can be controlled by *system clock* (*CLK*) and *negative CLK* (*NCK*) as well. A CE works as an inverter outputting a reversed value if its input values are identical. If its inputs change and have different values, its output can temporarily have the previous correct value due to intrinsic parasitic capacitance.

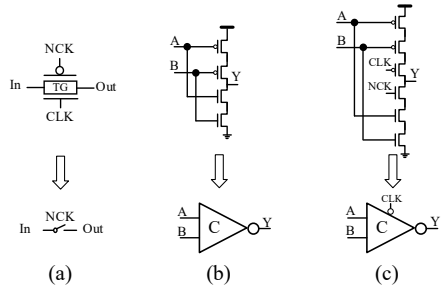



Fig. 1. Schematic of transmission gate (TG) and C-element (CE). (a) TG. (b) 2-input CE. (c) Clock-gating based 2-input CE.

Note that existing FFs cannot tolerate more-node upsets and HSPICE-tool-based exhaustive simulations are impractical for error injections with too many combinations. In this paper, cost effective and robust FFs are firstly proposed for safety-critical applications. The proposed 2-input and 3-input baseline FFs comprise two-stage of CEs providing SNU and DNU tolerance, respectively. The proposed FFs are extended to tolerate more node upsets. Note that the N-input FF in this paper means that the first-stage of CEs totally has N input nodes (i.e., N1 to Nn). Moreover, a highly efficient algorithm is proposed in this paper to automatically verify the MNU-tolerance of FFs. Simulation results demonstrate the efficiency of the proposed solutions.

The rest of this paper is organized as follows. Section II introduces the proposed FFs as well as their algorithm-based MNU-tolerance verification solution. Section III shows the results of comprehensive comparisons between the proposed FFs and the existing designs. Section IV summarizes the paper.

## II. Proposed Solutions

### A. Proposed Baseline Flip-Flop Designs

Figure 2 presents the proposed baseline FF designs. It can be seen that the 2-input and 3-inputs FFs in Fig. 2-(a) and (b) mainly consist of two-stage of CEs. Fig. 2(c) shows the layout of the 2-input FF (the layouts of the other FFs are not provided

This work was supported by NSFC under 62274052, 62174001, 62027815.

Aibin Yan, Yuting He, and Jie Cui are with School of Computers, Anhui University, Hefei 230601, China. (E-mail: abyan@mail.ustc.edu.cn, baiwhitebai@163.com, cuijie@mail.ustc.edu.cn).

Zhengfeng Huang, Wenjie Yan and Xiaolei Wang are with School of Microelectronics, Hefei University of Technology, Hefei 230601, China. E-mail: huangzhengfeng@139.com, jlnu@163.com, wangxiaolei@hfut.edu.cn

Tianming Ni is with School of Integrated Circuits, Anhui Polytechnic University, Wuhu 241000, China. (E-mail: timyni126@126.com). *He is the corresponding author.*

Patrick Girard is with LIRMM, University of Montpellier / CNRS, Montpellier 34095, France (E-mail: girard@lirmm.fr)

Xiaoqing Wen is with Department of Computer Science and Networks, Kyushu Institute of Technology, Fukuoka, Japan (wen@cse.kyutech.ac.jp).

due to page limitation). Fig. 2 (d) and (e) show the 4-input and 5-input FFs extended from the 2-input and 3-input FFs. The proposed FFs consists of a master latch (see the left side) and a slave latch (see the right side).

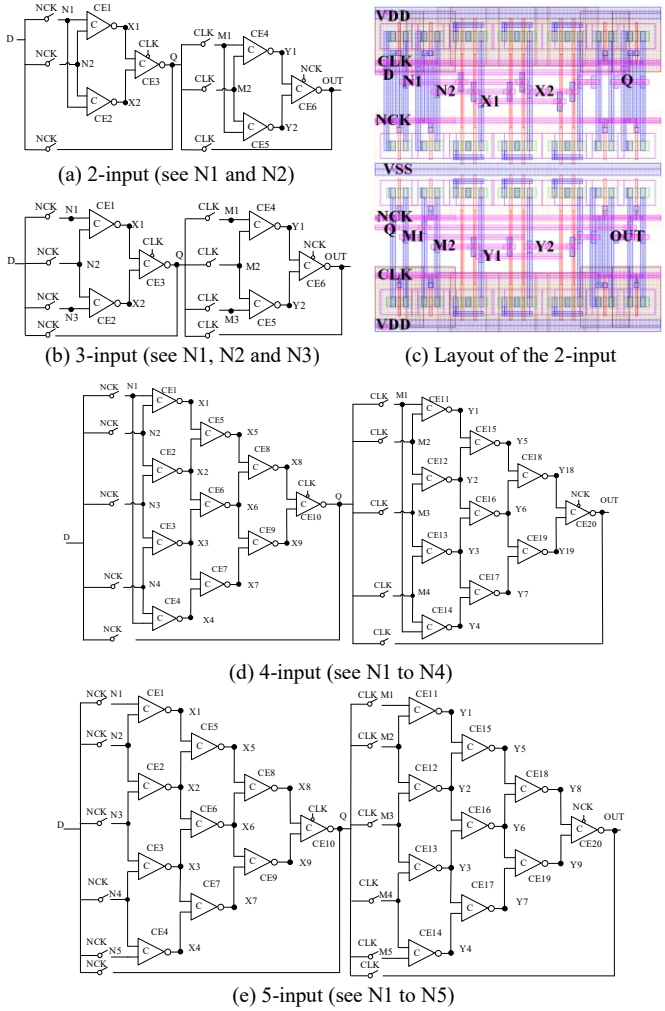



Fig. 2. Proposed baseline Flip-Flop designs.

In Fig. 2-(a), the first stage includes two parallel CEs (CE1 and CE2), the second stage includes one CE (CE3), and the outputs (X1 and X2) are used as the inputs of the CE in the second stage to tolerate SNUs. The 3-input FF in Fig. 2-(b) is constructed by splitting N1 of Fig. 2-(a) into two new inputs (N1 and N3) to tolerate DNUs. By adding two stages of CEs to Fig. 2-(a), a 4-input FF as in Fig. 2-(d) is constructed to tolerate TNUs. Figure 2-(e) shows the 5-input FF with quadruple node upset tolerance. Note that D is the input and OUT is the output.

Let us now take the 3-input FF as an example to discuss its working principle.

(1) When CLK = 1, the master latch works in transparent mode. D value can pass to the master latch through three TGs so that N1, N2, N3, X1 and X2 can be determined. However, Q value can be only determined through the left-bottom TG instead of clock-gated CE3 so as to reduce current competition on Q to reduce delay and power. Moreover, the TGs in the slave latch that works in hold mode are off, preventing them from receiving values from the master latch.

(2) When CLK = 0, the master latch works in hold mode and the slave latch works in transparent mode. D value cannot pass to N1, N2, N3 and Q through TGs and the Q value can output through CE3. The slave latch now can receive the value from the master latch and output the value to OUT through a bottom TG in the slave latch in a fast way.

Let us now discuss the fault-tolerance of the proposed FFs. Since the master latch is similar to the slave latch, only the fault-tolerance of the master latch is considered here. Let us first analyze the SNU-tolerance of the proposed 2-input FF. Since the FF is symmetrically constructed, only single nodes N1, X1, and Q need to be considered. When only N1 is affected by an SNU, since CE1 and CE2 can intercept this error, their outputs (X1 and X2) can still have their original correct values, and thus the output (Q) of CE3 can still have the original correct value. In another scenario, when X1 or Q is impacted by an SNU, since the input values of CE1 and CE2 remain unaffected, all output nodes of CEs in the FF, i.e., X1, X2, and Q, will refresh to output the original correct value. To summarize, the proposed 2-input FF can provide the correct value when suffering from an SNU. It should be noted that the proposed 2-input FF cannot tolerate DNUs. There is a counter-example where the FF will output an incorrect value if node pair <N1, N2> is impacted by a DNU. Considering the incapability of the 2-input FF to tolerate DNUs, the 3-input FF is introduced.

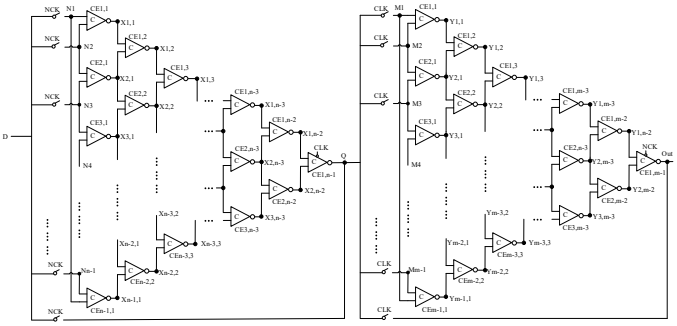
Let us analyze the DNU-tolerance principle of the proposed 3-input FF. When both affected nodes belong to a single latch (e.g., the master latch) in the worst case, there are only three representative cases: (D1) Any two internal nodes of the FF are affected; (D2) one input node of the first-stage CE and another internal node of the FF are affected; (D3) two input nodes of the first-stage of CE of the FF are affected.

In the case of (D1), since the input nodes of the first-stage of CEs of the FF are not affected, thus having their original correct output values, internal nodes X1, X2, and Q will be refreshed to output their previous correct values. In the case of (D2), since the master latch is symmetrically constructed, the indicative key node-pairs are only <N2, X1> and <N2, Q>. When <N2, X1> is hit by a DNU, since one input (N2) and the output (X1) of CE1 are simultaneously impacted, CE1 will retain the flipped value at X1. However, since only one input (N2) of CE2 is affected, CE2 still has the original value at X2. Clearly, only one input (X1) of CE3 has an error. Thus, the master latch can still have the correct value since the error at X1 can be intercepted by CE3. When <N2, Q> is hit by a DNU, since only one input (N2) of CE1 and CE2 is affected, CE1 and CE2 can still have the original correct values at X1 and X2. Thus, the master latch can still have the original correct output value because the inputs (X1 and X2) of CE3 are still correct. In the case of (D3), since the master latch is symmetrically constructed, there is only one indicative node-pair <N1, N2>. When <N1, N2> is hit by a DNU, since the inputs of CE1 are impacted by the DNU, the output (X1) of CE1 will be flipped. Then, this case becomes similar to the case where <N2, X1> is hit by a DNU in case D2. Therefore, the master latch can still have the correct output value since the soft errors can be intercepted by CE3. To summarize, the proposed 3-input FF can provide the correct value when suffering from a DNU. In the same manner, the TNU-tolerance of the 4-input FF and the quadruple-node upset-tolerance of the 5-input FF can be got.

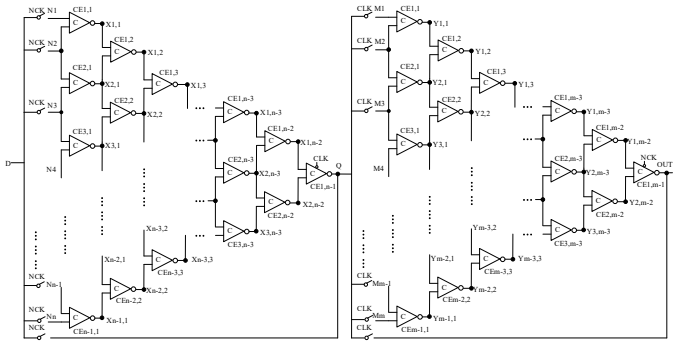
### *B. Proposed Extended Flip-Flop Designs*

Figure 3 shows the extended N-1 and N inputs FFs. Note that the N-1 inputs FF in Fig. 3-(a) is an extension of the 2-input FF in Fig. 2-(a), and that the N inputs FF in Fig. 3-(b) is an extension of the 3-input FF in Fig. 2-(b).

For the N-1 inputs FF in Fig. 3-(a), if it simultaneously suffers from N-2 errors, it can be easily found that the following issues can never happen: (M1) all inputs of the first-stage of CEs have flipped values, (M2) N-1 CEs have flipped values on both their input and output, (M3) all inputs of the last-stage of CE (CE1, n-1) have flipped values. Consequently, the FF can still provide the previous correct output value. To summarize, the N-1 inputs FF can tolerate N-2 errors.



(a) N-1 input (see N1 to Nn-1)



(b) N input (see N1 to Nn)

Fig. 3. Proposed extended Flip-Flop designs.

In the same manner, as for the N inputs FF in Fig. 3-(b), if the FF simultaneously suffers from N-1 errors, after an extensive investigation, it can be found that the above-mentioned issues (M1), (M2), and (M3) can never happen. Hence, the FF can still provide the original correct value. To summarize, the N inputs FF can tolerate N-1 errors. Note that the first-stage of CEs of the proposed extended FFs can be split to have more inputs to intercept soft errors.

### *C. Proposed Algorithm-based Verification Methodology*

To verify the MNU-tolerance of each proposed FF, we use a method that maps its structure into a forest. Figure 4 illustrates the transformation of the proposed 3-input FF into a forest model. The forest comprises multiple trees, with the number of trees corresponding to the number of CEs within the FF. In each tree, the hierarchical relationship between nodes signifies the connectivity between the inputs and outputs of the CEs.

Taking CE1 as an example, X1 serves as its output node, while N1 and N2 act as its input nodes. When converting CE1 into a corresponding binary tree, X1 becomes the root node of the tree, with N1 and N2 as its child nodes. Similarly, for an N-input CE, the root node corresponds to the output node, while the child nodes represent the input nodes. The primary distinction between converting an N-input CE and a 2-input CE into their corresponding trees lies in the number of child nodes. To verify the MNU-tolerance of the proposed FFs, we propose a novel algorithm for MNU-tolerance verification (see Alg. 1). The inputs of this algorithm consist of three parts.

**Algorithm 1**: MNU-Tolerance Verification for a Flip-Flop.

**Input:** A Flip-Flop with a forest structure, Output node of Flip-Flop, Flipped Node Count *Flip_Node_Count*

**Output:** (see the print function)

1: *Total_list* ← Create all possible combinations of *Flip_Node_Count* number of nodes from the *Node_List*
2: **for** each T*NU_list* in *Total_list* **do**
3: *Real_list* ← *TNU_list*
4: **do**
5: *is_add_node_to_real_list* ← false
6: **for** each node in the *Node_List* but not in *Real_list* do
7: **if** all child nodes of the current node are in *Real_list* then
8: add the current node to *Real_list*
9: *is_add_node_to_real_list*←true
10: break
11: **end if**
12: **end for**
13: **while** (*is_add_node_to_real_list*)
14: *Affected_list*←removes parent node whose child nodes not in *Real_list*
15: **if** *root* is in *Affected_list* **then**
16: print ( “Error kept, i.e., the Flip-Flop cannot tolerate the errors.”)
17: break
18: **end if**
19:**end for**

(1) The forest representing the FF structure. Once the FF is transformed into a forest model, it can be input into the algorithm as the base model.

(2) A count of flipped nodes. This count indicates how many single nodes are simultaneously affected to simulate an MNU.

(3) The output node of the FF structure to be verified. Since each node of a different FF has a unique flag, it is necessary to specify the flag of the output node of the FF when performing the MNU-tolerance verification.

Through this method, we can accurately and efficiently verify the MNU-tolerance of the FF structure. To simulate all MNUs comprehensively, the algorithm first obtains the complete list of FF nodes from the input forest structure. Based on the provided count of flipped nodes, the algorithm calculates all possible combinations of flipped node groups in the *Total_list*. For each combination, the algorithm executes a series of steps to update the list, determining what extra nodes are flipped, resulting in the final *real_list*. Note that, during these steps, every node in the FF may be directly or indirectly affected by an MNU. Subsequently, the algorithm traverses the tree structure hierarchically, deciding to retain or remove a parent node based on the states of its child nodes. If a parent node exists in the *real_list*, yet all its child nodes do not exist in the *real_list*, the parent node is considered recoverable to the correct state and can be removed from the *real_list.* This process is recursively executed throughout the entire tree structure until all parent nodes have been checked, ensuring the integrity of the validation. Finally, the affected node combination *Affected_list* is obtained. *Affected_list* contains all unrecoverable nodes and it verifies the MNU-tolerance by

determining whether the root node is in *Affected_list*.

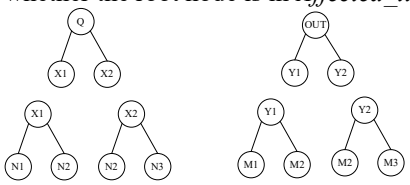



Fig. 4. Modeled forest for the proposed 3-input Flip-Flop.

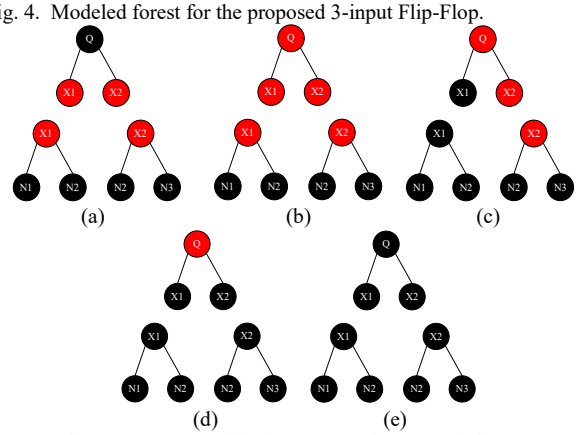



Fig. 5. The process to verify the MNU-tolerance of the proposed 3-input Flip-Flop. (a) Initially *real_list* = <X1, X2>, (b) add the affected node Q, and thus the updated *real_list* = <X1, X2, Q>, (c) remove the recoverable node X1 and thus the updated *real_list* = <X2, Q>, (d) remove the recoverable node X2 and the thus updated *real_list* = <Q>, and (e) the final *Affected_list* is null.

Let us discuss the verification process in detail. Consider the scenario where the master latch of the proposed 3-input FF is affected by a DNU, with the initial *real_list* containing <X1, X2> (see Fig. 5-(a)). As the *real_list* is updated, we notice that Q is the output node of CE3, with input nodes X1 and X2 in the *real_list*. Since Q serves as the root node of the binary tree, and both X1 and X2 as its child nodes are all affected, we conclude that Q is also affected and thus include it in the *real_list* (see Fig. 5-(b)). After this update, the *real_list* becomes <X1, X2, Q>. We iterate this process until we obtain the final updated *real_list*. When completing the hierarchical traversing of the forest structure and reaching the root node X1, we observe that its child nodes N1 and N2 do not exist in *real_list*. This observation prompts us to remove node X1 (see Fig. 5-(c)). Similarly, node X2 should also be removed from *real_list* (see Fig. 5-(d). Until the root node Q is traversed, its child nodes X1 and X2 are not in *real_list*, and thus node Q should be removed from *real_list* (see Fig. 5-(e)). This recursive process continues until all nodes have been inspected, resulting in the *Affected_list* after eliminating recoverable nodes. Finally, we verify if the root node Q persists in the *Affected_list*. Upon verification, node Q is absent in the *Affected_list*, indicating that the FF can tolerate the DNU for nodes <X1, X2>. We can subsequently examine the other node combinations to simulate all DNUs. Through successive validations, it becomes evident that node Q consistently remains absent in the *Affected_list*, implying the FF's ability to tolerate DNUs.

A counter-example is discussed here to demonstrate the TNU-not-tolerance for the 3-input FF. Let us assume the FF is impacted by a TNU, with the initial *real_list* comprising nodes <N1, N2, N3>. The verification process of the algorithm is shown in Fig. 6. The *real_list* undergoes continuous updates until it stabilizes, resulting in *real_list* = <N1, N2, N3, X1, X2, Q>. When traversing each tree's root node, the roots that can be restored to the correct states are eliminated from the *real_list*. When traversing to the root node Q, it is found that its child nodes X1 and X2 both remain in the *real_list*, indicating that node Q cannot be restored to the correct state. The node Q still exists in *Affected_list* (see Fig. 6-(d)). Thus, it is concluded that the proposed 3-input FF cannot tolerate the TNU.

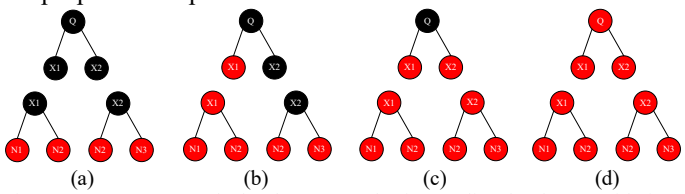



Fig. 6. A counter-example for the proposed 3-input Flip-Flop but considering three node-upsets. (a) Initially *real_list* = <N1, N2, N3>, (b) add the affected node X1, and thus the updated *real_list* = <N1, N2, N3, X1>, (c) add the affected node X2, and thus the updated *real_list* = <N1, N2, N3, X1, X2>, (d) add the affected node Q, and thus the updated *real_list* = <N1, N2, N3, X1, X2, Q> and thus the final *Affected_list* = <N1, N2, N3, X1, X2, Q>.

In summary, through the algorithm-based validations for the MNU-tolerance of the proposed 3-input FF, it is found that the FF can tolerate all SNUs and DNUs. Nevertheless, for triple and more node-upsets, the FF fails to tolerate them since there is at least a counter-example that cannot provide tolerance. Note that the running of the algorithm was performed on a laptop equipped with an Intel Core i7-12700 processor, 8GB DDR4 memory, and an Intel(R) UHD Graphics 770. Windows 11 OS and Python 3.9 were used. To ensure the consistency and repeatability of the results, all experiments were repeated three times under the same environmental parameters. Through the algorithmic evaluation, we found that the runtime of the 3-input FF verification is 1.99 ms only. This significantly reduces the verification time required when compared to the traditional verification method using EDA tools that also require manual intervention. Note that, all of our proposed FFs in this paper can be quickly verified using this algorithm, and the verification running-time is very short as well.

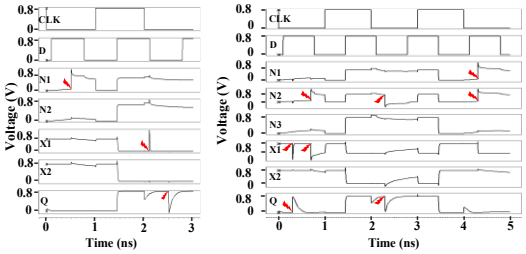



Fig. 7. Simulation results for the proposed baseline FFs. (a) 2-input FF with SNU injections (the left), and (b) 3-input FF with DNU injections (the right).

### *D. Simulations*

The proposed FFs were designed/implemented using the same conditions, i.e., a 22 nm advance CMOS library, a 0.8V Vdd, and W/L of PMOS transistors with 90nm/22nm and W/L of NMOS transistors with 45nm/22nm. Extensive simulations using HSPICE from Synopsys were performed. Note that we used a controllable double exponential current source model to perform all the SNU injections. The injected amount of charge was up to 25fC in the worst case which is large enough to flip the correct value of the node and demonstrate the efficiency of the error injections. The time constants of the rise and fall of the current pulse were set to 0.1ps and 3.0ps [1], respectively. This is true for all error injections in this paper.

It can be seen from Fig. 7-(a) that the proposed FFs can tolerate all the injected SNUs and/or DNUs with red-lightening

marks (when Q is impacted, it only resulted in a small pulse) and the algorithm-based verification results are completely consistent with the HSPICE-based simulation results. Note that we used two simultaneous SNUs to mimic a DNU.

## III. Evaluation and Comparative Results

To make a fair comparison, all the compared FFs in this paper are implemented under the same conditions mentioned in the previous section, i.e., a 22 nm advanced CMOS library, a 0.8V Vdd and the room temperature.

TABLE I
Reliability and Overhead Comparisons among Hardened Flip-Flops.

| Flip-Flop | Ref. | SNU Tol. | DNU Tol. | TNU Tol. | QNU Tol. | Area ($\mu m^2$) | Delay (ps) | Power ($\mu W$) | $10^{-2}\times$ ADPP |
|---|---|---|---|---|---|---|---|---|---|
| Quatro-FF | [6] | YES | NO | NO | NO | 6.14 | 38.99 | 4.95 | 11.85 |
| TMR-FF | - | YES | NO | NO | NO | 9.66 | 45.38 | 2.97 | 13.02 |
| DNUR-FF | [7] | YES | NO | NO | NO | 11.29 | 42.70 | 2.26 | 10.89 |
| DICE-FF | [8] | YES | NO | NO | NO | 5.64 | 17.13 | 1.79 | 1.73 |
| DRRH-FF | [9] | YES | NO | NO | NO | 5.94 | 43.16 | 1.58 | 4.05 |
| SNUR-NVFF | [10] | YES | NO | NO | NO | 5.94 | 18.30 | 1.03 | 1.12 |
| HPST-FF | [11] | YES | NO | NO | NO | 8.32 | 23.00 | 1.00 | 1.91 |
| DUT-FF | [4] | YES | YES | NO | NO | 14.60 | 14.00 | 2.13 | 4.35 |
| HRLPFF | [5] | YES | YES | NO | NO | 8.02 | 29.24 | 0.71 | 1.66 |
| 2-input | Proposed | YES | NO | NO | NO | 5.94 | 9.15 | 0.39 | 0.21 |
| 3-input | Proposed | YES | YES | NO | NO | 6.53 | 8.99 | 0.40 | 0.23 |
| 4-input | Proposed | YES | YES | YES | NO | 15.44 | 15.03 | 1.34 | 3.11 |
| 5-input | Proposed | YES | YES | YES | YES | 16.04 | 14.49 | 1.36 | 3.16 |

Table I shows the reliability and overhead comparisons of FFs. Note that "Tol." denotes "Tolerant". Also note that, some works, e.g., those in [2-3, 14-16], are not compared, since they not mainly focus on FF designs. According to the comparison data in Table I, the proposed baseline FFs have excellent fault tolerance against node-upsets; however, most existing FFs are limited to tolerating SNUs and/or DNUs. As mentioned above, the N input FF exhibits tolerance against N-1 node upsets which demonstrates the high reliability and high scalability of our proposed FFs with adjustable count of inputs.

Table I also describes the cost comparison among FFs. "Area" is the silicon area extracted as in [1], "Power" is the average power dissipation (dynamic and static), "Delay" is D to OUT transmission delay, i.e., the average of rise and fall delays from D to OUT, and "ADPP" denotes the area-delay-power product calculated by multiplying area, delay, and power.

***In terms of area***, the FFs in [8-10] and the proposed 2-input FF consume small silicon area due to the small number of transistors used. However, to achieve MNU-tolerance, the proposed 4-input FF and 5-input FF necessitate additional transistors, resulting in extra silicon area. ***In terms of delay***, the proposed FFs have small delay, while the other FFs exhibit large delay. This discrepancy arises from the presence of redundant transistors on the D to OUT path in the other FFs, thereby increasing the delay. ***In terms of power***, the proposed 2-input FF exhibits the lowest power consumption, attributed to its structure with no current competition. In contrast, Quatro-FF has the highest power dissipation due to its utilization of Quatro cells, resulting in significant current competition among its internal nodes. ***In terms of ADPP***, the proposed 2-input FF and 3-input FF have small ADPP due to their small delay, area, and power. Conversely, the proposed 4-input FF and 5-input FF exhibit a moderate ADPP compared to some of the other FFs, primarily due to their extra silicon area required to accommodate MNU-tolerance. The proposed FFs shows high superiority over the other FFs. Compared to the DNU-tolerant FFs, the proposed 3-input FF achieves an average reduction of 36.03% in silicon area, 52.52% in delay, 62.44% in power, and 90.43% in ADPP. In summary, the proposed 2-input FF and 3-input FF almost have the smallest overhead compared to the same type of FFs. The proposed more input FFs have moderate delay and power while ensuring the highest reliability; although extra area is required, the modest sacrifice accompanies a significant improvement in performance.

## IV. Conclusion and Further Works

This paper has proposed a series of flip-flops tolerating node upsets with cost effectiveness especially in terms of delay and power. This paper has also proposed an algorithm-based node-upset tolerance verification method. Simulation results have demonstrated the efficiency of the proposed solutions. In our further work, the proposed algorithm-based methodology will be further investigated and expanded to make it applicable to the verification for large circuits with more node upsets.

## References

[1] A. Yan, Z. Li, J. Cui, et al, "LDAVPM: A Latch Design and Algorithm-based Verification Protected against Multiple-Node-Upsets in Harsh Radiation Environments," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 6, pp. 2069-2073, 2023.

[2] A. Amirany, et al, "High-Performance and Soft Error Immune Spintronic Retention Latch for Highly Reliable Processors," *Iranian Conference on Electrical Engineering*, pp. 1-5, 2020.

[3] A. Amirany, et al, "Low Power, and Highly Reliable Single Event Upset Immune Latch for Nanoscale CMOS Technologies," *Iranian Conference on Electrical Engineering,* pp. 103-107, 2018.

[4] A. Yan, et al., "Two Double-Node-Upset-Hardened Flip-Flop Designs for High-Performance Applications," *IEEE Trans. on Emerging Topics in Computing*, vol. 11, no. 4, pp. 1070-1081, 2023.

[5] A. Yan et al., "A Highly Robust and Low-Power Flip-Flop Cell with Complete Double-Node-Upset Tolerance for Aerospace Applications," *IEEE Design & Test*, vol. 40, no. 4, pp. 34-41, 2023.

[6] Y. Li, H. Wang, R. Liu, et al., "A Quatro-Based 65 nm Flip-Flop Circuit for Soft-Error Resilience," *IEEE Trans. on Nuclear Science*, vol. 64, no. 6, pp. 1554-1561, 2017.

[7] F. Alghareb and R. DeMara, "Design and Evaluation of DNU-Tolerant Registers for Resilient Architectural State Storage," *ACM Great Lakes Symposium on VLSI*, pp. 1-4, 2019.

[8] K. Kobayashi, et al., "A Low-Power and Area-Efficient Radiation-Hard Redundant Flip-Flop, DICE ACFF, in a 65 nm Thin-BOX FD-SOI," *IEEE Trans. on Nuclear Science*, vol. 61, pp. 1881-1888, 2014.

[9] G. Jaya, S. Chen, and S. Liter, "A Dual Redundancy Radiation-Hardened Flip-Flop Based on C-element in 65nm Process," *IEEE International Symposium on Integrated Circuits*, pp. 1-4, 2016.

[10] F. S. Alghareb, R. Zand and R. F. Demara, "Non-Volatile Spintronic Flip-Flop Design for Energy-Efficient SEU and DNU Resilience," *IEEE Trans. on Magnetics*, vol. 55, no. 3, pp. 1-11, 2019.

[11] Z. Huang, G. Liang and S. Hellebrand, "A High Performance SEU Tolerant Latch," *J. of Electronic Testing,* vol. 31, pp. 349-359, 2015.

[12] E. Abbasian, et al, "A Reliable Low Standby Power 10T SRAM Cell With Expanded Static Noise Margins," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 69, no. 4, pp. 1606-1616, 2022.

[13] A. Yan, et al, "Quadruple and Sextuple Cross-Coupled SRAM Cell Designs with Optimized Overhead for Reliable Applications," *IEEE Trans. on Device and Materials Reliability,* vol. 22, pp. 282-295, 2022.

[14] A. Dweik and Y. Iraqi, "Error Probability Analysis and Applications of Amplitude-Coherent Detection in Flat Rayleigh Fading Channels," *IEEE Trans. Communications*, vol. 64, pp. 2235-2244, 2016.

[15] S. Campitelli, et al, "F-DICE: A multiple node upset tolerant flip-flop for highly radioactive environments," *IEEE Internat. Symp. Defect & Fault Tolerance in VLSI & Nanotechnology Systems*, pp. 107-111, 2013.

[16] V. Bakhtiary, et al, "An SEU-hardened ternary SRAM design based on efficient ternary C-elements using CNTFET technology," Microelectronics Reliability, vol. 140, 2023.