Journal of Electronic Testing (2024) 40:45–60
https://doi.org/10.1007/s10836-024-06105-x

A High-Performance Quadruple-Node-Upset-Tolerant Latch Design and an Algorithm for Tolerance Verification of Hardened Latches

Hui Xu[1] · Xuewei Qin[1] · Ruijun Ma[1] · Chaoming Liu[1] · Shuo Zhu[1] · Jun Wang[1] · Huaguo Liang[2]

Received: 5 September 2023 / Accepted: 19 January 2024 / Published online: 2 March 2024

Abstract
With the development of semiconductor technology, the shrinking of feature size in integrated circuits has made them more sensitive to multiple-node-upsets (MNUs). Researchers have proposed various circuit-hardened methods, such as hardened latches, to address this issue. Currently, the reliability verification of latches relies on complex EDA tools, such as HSPICE, Cadence Virtuoso, and other tools for error injection. Therefore, this article proposes a high-performance quadruple-node-upset (QNU) tolerant latch design, called the HQNUT latch, based on 32 nm CMOS technology. Additionally, an algorithm-based latch verification process is proposed to enhance the efficiency and reliability of latch verification. This approach enables a fast and accurate assessment of the latch's fault-tolerant capability. Due to clock gating technology and high-speed path technology, HQNUT's power consumption and delay are reduced. Simulation results show that the proposed algorithm can certify the soft-error-tolerability of hardened Latches. Compared with existing QNU-tolerable hardened latches, the proposed latch reduced power consumption, area, delay, and power-delay product (PDP) by about 36.9%, 5.6%, 19.8%, and 46.4%, respectively.

Keywords Latch design · Algorithm-based verification · Fault tolerance · Multiple-node-upset

1 Introduction

Since semiconductor technology entered the nano-meter era, the size of transistors has significantly reduced, resulting in a notable decrease in the critical charge of circuit nodes. Integrated Circuits (ICs) are increasingly prone to soft errors. These soft errors primarily results from single-event effects (SEEs) [10] occurring within ICs situated in radiation environments. However, these errors do not induce lasting damage to the circuitry; they temporally modify a fraction of the logical values stored within the circuit. Hence, they are termed "soft errors" [14].

Radiation environments often contain an abundance of high-energy particles, including protons and α particles. While these particles strike the sensitive area of a circuit in such an environment, electron–hole pairs are produced along the trajectory. [3]. The drain of transistors collects these charges due to the combined effects of the electric field and the PN junction characteristics, and when the charge is over the critical charge, the circuit's logic state may change, leading to a single-node-upset (SNU). The size of transistors has significantly reduced nowadays, which increases the probability of radiation particles'

Responsible Editor: F. Vargas

✉ Huaguo Liang
huagulg@hfut.edu.cn

Hui Xu
xuhui@aust.edu.cn

Xuewei Qin
qinxuewei1@foxmail.com

Ruijun Ma
ruijun_ma@foxmail.com

Chaoming Liu
lchaoming@foxmail.com

Shuo Zhu
2021201167@aust.edu.cn

Jun Wang
junwangwy@163.com

[1] School of Computer Science and Engineering, Anhui University of Science and Technology, Huainan 232001, China

[2] School of Microelectronics, Hefei University of Technology, Hefei 230601, China

impact-induced charges being collected by multiple sensitive nodes, affecting two, three, or even four close nodes, resulting in double-node-upset (DNU), triple-node-upset (TNU), or quadruple-node-upset (QNU) [1]. The latch is the most commonly used sequential structure in integrated circuits. Hardening the latch to tolerate these soft errors has become a pressing issue.

According to reference [20], precisely calculating the probability of QNU occurrence is a highly challenging task because QNU is affected by six or more parameters. Therefore, to ensure the latch's reliability, we introduce QNU-tolerant design requirements. Due to charge sharing, in harsh radiation environments, QNU can be induced by the impact of one high-energy particle. If advanced circuits are manufactured with ultra-small technology such as 7 nm, the spacing of transistors and nodes will be even smaller, greatly increasing the probability of QNU occurrence [21].

Currently, researchers verify the multi-node tolerance of the latch by using EDA tools with fault injection testing. This method is effective for structures with fewer nodes. However, the workload increases significantly when the structure has more nodes and multiple nodes can be flipped. For example, for a latch with 16 nodes, verifying its QNU tolerance would require C_{16}^{4}=1820 experiments. Therefore, most researchers validate the latch's tolerance capability by testing representative node combinations. However, this approach can easily lead to omissions, and transistor has higher driving strength ($W_{NMOS} = \lambda$ and $W_{PMOS} = 2\lambda$), therefore if both PMOS and NMOS transistors are ON, the output is "0". If both PMOS and researchers cannot guarantee that all possible scenarios resulting in latch errors are considered. In summary, verifying the multi-node tolerance capability of a latch with a large number of nodes is a difficult and error-prone task. In the past, a verification algorithm [21] was proposed, but it could only perform self-recovery verification and was only applicable to latches that used Muller C-Elements (MCEs) and inverters.

To address the issues above, this article proposes a highly reliable latch design that can maintain correct output even in the presence of QNU. This design ensures that electronic devices can work normally in radiation and high-frequency environments. Additionally, this article has proposed a new algorithm to assess the fault-tolerance performance of latches. Through testing, this algorithm verifies that our proposed latch can produce correct output results in the presence of any QNU. This algorithm provides a robust evaluation of the latch's fault tolerance, ensuring its reliability under various error conditions. Moreover, compared to previous algorithms[21], this algorithm can perform both tolerance and self-recovery verification. It can be applied to latches that use input-split inverters, dual interlocked storage cells (DICE), or other structures that truth tables can characterize, and it is not limited to MCE-based latches.

The rest of the paper is organized as follows. Section 2 shows the previous radiation-hardened component and latch designs. Section 3 outlines the proposed latch design and presents the introduced latch tolerance verification algorithm. Section 4 presents fault tolerance validation, experimental evaluation, and performance comparison with other latch designs having similar functionality. Finally, Section 5 summarizes the entire paper.

2 Previous Work

2.1 Existing Component

Figure 1 illustrates commonly used reinforcement structures in latch designs. Figure 1(a) shows a dual-input inverter, which functions similarly to an inverter but has two inputs. The gate of the PMOS transistor is connected to In1, while the gate of the NMOS transistor is connected to In2. When In1 and In2 have the same logical value, it works like a standard inverter, producing an output complementary to In1 and In2 [4]. The NMOS.

NMOS transistors are OFF, the output retains its previous logic state. This design provides SNU tolerance ability. Figure 1(b) and (c) show the schematic of the CEs [11]. They will output the logical value opposite to the input when all inputs are the same. If one of the inputs changes, they will keep the same outputs. So they have complete SNU tolerance ability. Figure 1(d) shows the inverter. Figure 1(e) shows the schematic of the Modified two-input CE [7]. When In1 is 1, and In2 is 0, the pull-up network is OFF while the pull-down network is ON, its output value is 0. When In1 is 0 and In2 is 1, the pull-up network is ON, and the pull-down network is OFF, its output value is 1. In other cases, the output doesn't change. This structure has complete SNU tolerance ability and will be used in the latch design proposed in this paper.

2.2 Existing Solutions

(a) SHLR latch

Figure 2(a) shows an SNU self-recoverable latch named SHLR [8]. It contains three input-split inverters and two two-input CEs. When an error occurs in any of N1, N2, or N3, the CEs intercepts the error to ensure the correctness of both Q and Qa. If an error affects Q or Qa, the three input-split inverters filter out the error, ensuring the correctness of the other nodes. This achieves SNU self-recovery. However, the input-split inverters have higher power consumption, and SHLR cannot tolerate DNU.

(b) NTHLTCH latch

Fig. 1 Schematics, Truth Tables and element type of **a** input-split inverter, **b** two-input CE, **c** three-input CE, **d** inverter., **e** Modified two-input CE

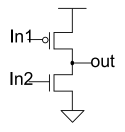
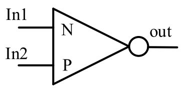
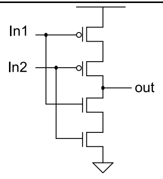
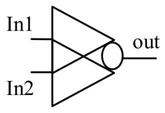
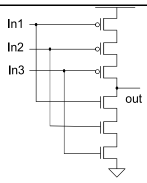
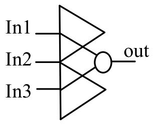
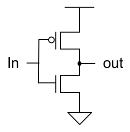
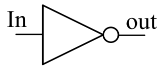
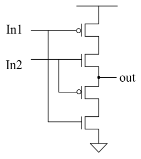
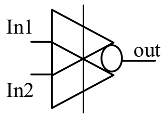
Transistor-level Schematic	Truth Table	Gate-level Schematic	Component Type
 (a)	In1 In2 out 1 1 0 0 0 1 1 0 Z 0 1 0		0
 (b)	In1 In2 out 1 1 0 0 0 1 other Z		1
 (c)	In1 In2 In3 out 1 1 1 0 0 0 0 1 other Z		2
 (d)	In out 1 0 0 1		3
 (e)	In1 In2 out 1 0 0 0 1 1 other Z		4

Figure 2(b) shows the structure of the NTHLTCH [9] latch. It consists of a 3×3 array of nine dual-input CEs and three inverters. Each CE is capable of blocking an error in one of its inputs. When an error occurs in a column of the array, it will be filtered out at the next stage and subsequently restored by other correct nodes. It can tolerate up to two node errors but cannot tolerate TNU. The area overhead is too large.

(c) DNURL latch

The schematic of the DNURL latch [17] is shown in Fig. 2(c). It contains three SNU self-recovering units. Each single-node self-recovering unit includes two CEs and three two-input inverters. Its three CEs are interconnected through feedback to maintain the correctness of nodes' logic values. When one of the nodes experiences an error, it ensures that the values of other nodes are not affected, and the correct nodes can recover the erroneous node. Connecting three such modules, it can tolerate errors in any two nodes in one module or in any two nodes across different modules. In other words, it achieves DNU tolerance but cannot handle TNU.

(d) LCTNUT latch

Figure 2(d) shows the schematic of the LCTNUT latch [19]. It consists of two modules: the storage module (SM) on the left and the error interception module on the right. The storage module comprises eight input-split inverters that form a loop among themselves to store internal data. The error interception module filters out errors that may occur in the storage module. This configuration ensures that the values of the output nodes do not change, thereby achieving TNU tolerance. Nonetheless, as a result of the extensive use of input-split inverters, there is a higher power consumption caused by current internal competition.

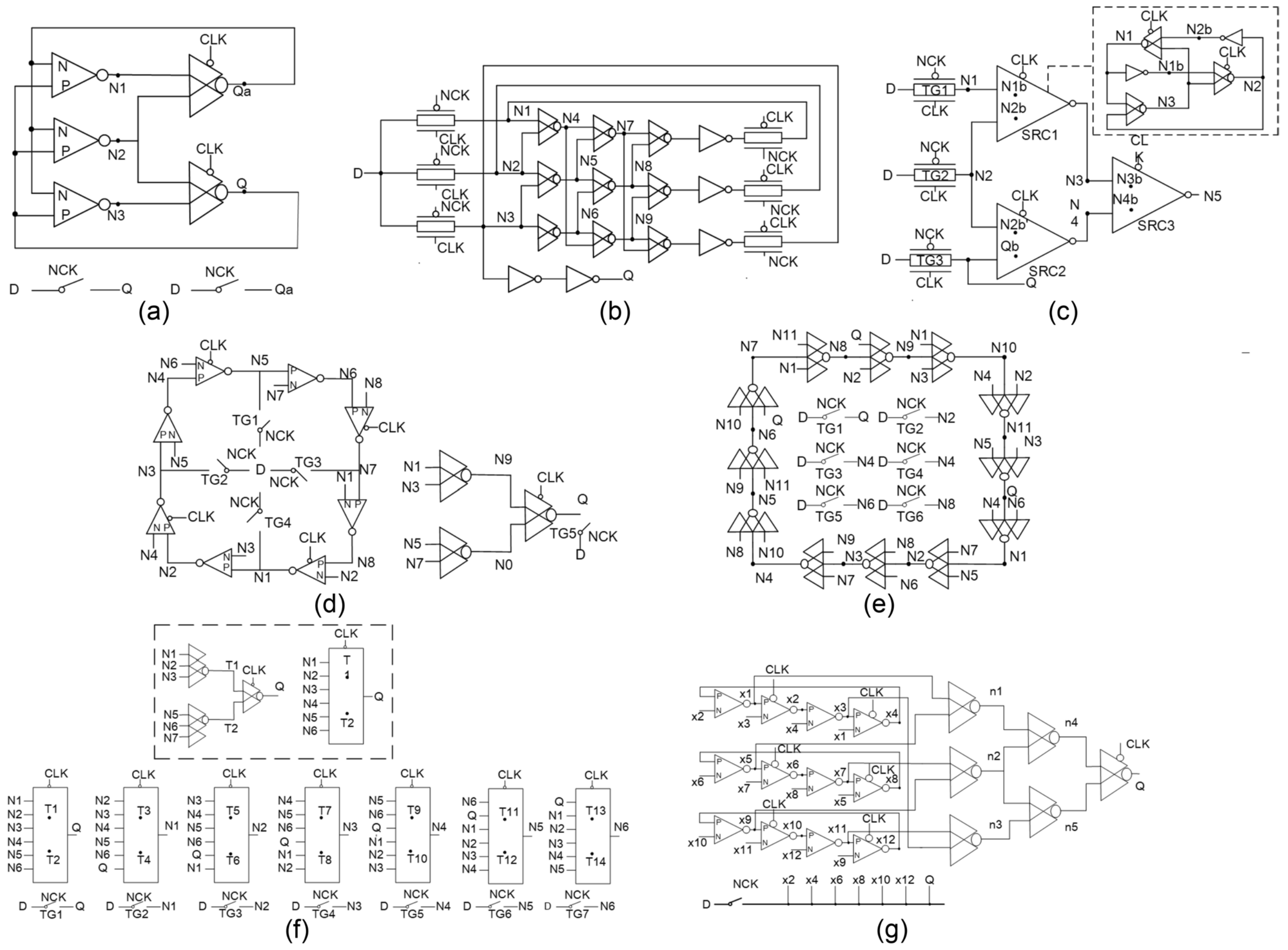


Fig. 2 Schematics of existing hardened latches. **a** SHLR [8]. **b** NTHLTCH [9]. **c** DNURL [17]. **d** LCTNUT [19]. **e** LCTNURL [12]. **f** TNURL [18]. **g** QNUTL-CG [20]

(e) LCTNURL latch

Figure 2(e) displays the structure of LCTNURL [12], which consists of a circular structure formed by connecting 12 three-input CEs in a chain to store data. This configuration enables tolerance to any TNU. Since only CEs are used internally, it results in lower power consumption and delay. However, the tradeoff is that it incurs a larger area overhead due to the increased number of components.

(f) TNURL latch

Figure 2(f) shows the structure of the TNURL [18] latch. Its main structure consists of seven error interception modules, each comprising one two-input CE and two three-input CEs. These seven modules are interconnected to store internal data, and the TNURL latch can restore from errors occurring in three or fewer nodes. It shares similar advantages and disadvantages with the LCTNURL latch, as it uses only MCE and exhibits lower power consumption and delay. However, it incurs a larger area overhead. Furthermore, it can only tolerate TNU. When encountering QNU or more severe disruptions, the latch may be unable to recover from the resulting data errors.

(g) QNUTL-CG latch

Figure 2(g) demonstrates the QNUTL-CG latch [20]. It consists of two modules: a storage module and an error interception module. The storage module consists of three DICE cells. The error filtering module is a three-stage filtering structure comprising six two-input CEs. When errors occur in the DICE cells, the error filtering module intercepts the errors and ensures the correctness of the output nodes. Thus, the QNUTL-CG latch achieves complete tolerance to QNU. However, due to the DICE cells' internal structure, there is a significant current crowding phenomenon, leading to increased power consumption and delay.

3 Proposed Solutions

3.1 Proposed MNU Tolerant Latch

Figure 3 shows the proposed latch structure. In the figure, D is the input, Q is the output, CLK is the system clock signal, and CLKB is the negative signal. TG1-TG7 are seven transmission gates, and n1-n15 and Q are internal data nodes. We proposed an Self-Recovery (SF) module composed of two modified C-elements and one regular C-element, capable of achieving Self-Recovery Under SNU. SF1-SF3 are used to store data. C1-C6 are two-input CEs, where n1, n2, n4, n5, n7, and n8 serve as inputs to C1, C2, and C3. Meanwhile, n3, n6, and n9 are used to maintain the correctness of the internal data within the SFs module. In addition, a clock-gating (CG) inverter after C6 is used in the circuit to reduce overhead.

When CLK = 1 and NCK = 0, the HQNUT works in transparent mode, and D transmits data to nodes so that n1, n2, n4, n5, n7, n8, n15 and Q can be pre-charged by D. Obviously, n1 and n2 can determine n3 through SF1. Similarly, n6 and n9 can be determined through SF2 and SF3, respectively. Therefore, the values of n1, n2, n4, n5, n7, and n8 are all the same. Ensure to reduce power consumption and transmission delay, the clock-gating (CG) technique be used on the inverter to decrease current competition on node Q. Therefore, the HQNUT can normally work, and Q are driven by D.

When CLK = 0 and NCK = 1, the HQNUT works in hold mode, and all TGs are OFF. Therefore, Q is driven only by n15 rather than by D. At this point, the three nodes inside each SF transfer values to each other's inputs, forming a stable cyclic structure, storing the internal value, and outputting it through Q.

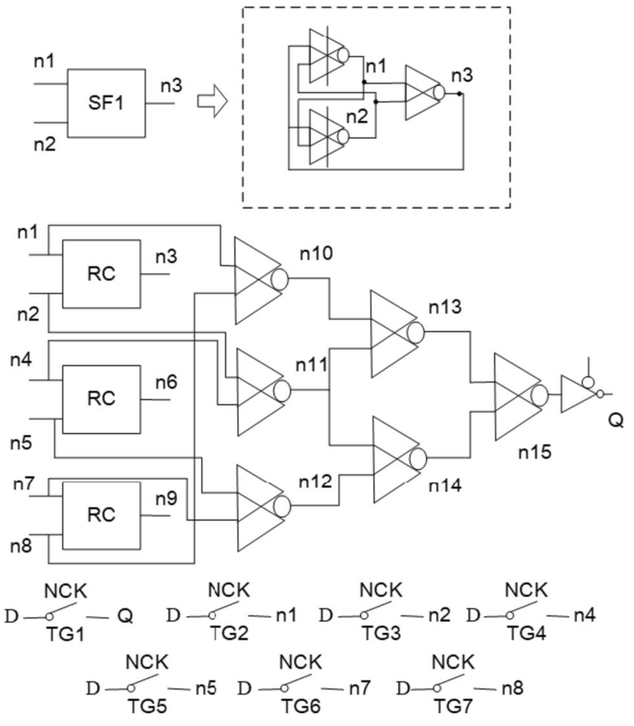


Fig. 3 Proposed HQNUT latch design

3.2 Proposed Algorithm-based Verification Methodology

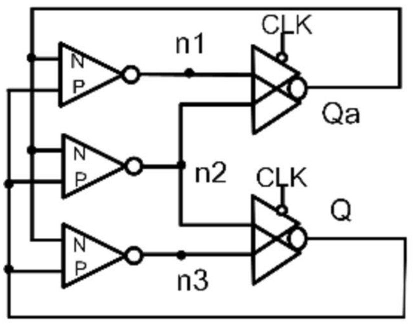
This article introduces a novel validation algorithm to verify the MNU tolerance of the HQNUT (see Alg. 1). This algorithmhas four inputs: the matrix representation of the latch structure; the initial states and component types; the list of all nodes in the latch; and the count of flipped nodes. The false_list in the algorithm is used to store all error nodes.

Algorithm 1: MNU Tolerant Verification of Latches

```
Input: Latch-structure Adjacency_Matrix, All-node List Node_Lists, the two-dimensional array of the initial state and component types of the latch structure State_List, Flipped Node Count m, The number of output Node Q
Output: (see the print function)
1:  begin
2:      select all possible combinations of m nodes from the Node_Lists and add them to the Test_lists
3:      foreach list in Test lists do
4:          add the nodes that occur errors to false_list
5:          do
6:              foreach node in Node_List but not in false_list do
7:                  Calculate the value of node based on the truth table and the values of input nodes.
8:                  if The calculated value is different from the value in State_List then
9:                      Modify the value of the node in State_List
10:                     Add node in false_list
11:                 end if
12:             end foreach
13:         while the number of nodes in false_list remains unchanged
14:         do
15:             foreach node in false_list do
16:                 Calculate the value of node based on the truth table and the values of input nodes.
17:                 if The calculated value is different from the value in State_List then
18:                     Modify the value of the node in State_List
19:                     Pop node in false_list
20:                 end if
21:             end foreach
22:         while the number of nodes in false_list remains unchanged
23:         if the state of the output node of the latch is different from its initial state then
24:             print"This latch cannot tolerant"
25:             break;
26:         end if
27:         Resetting State_List to prepare for the next list
28:     end foreach
29:     print"the latch can tolerant"
30: end
```

To begin with, we will model the latch and convert it into an adjacency matrix representing the latch interconnections and a two-dimensional array representing the states and component types. Because the algorithm requires the calculation of outputs based on different components, we need to define the working principles of various components in advance and determine the output result based on the truth table and input values.

Let us consider the SHLR [8] latch as an example latch. Figure 4(a) shows the SHLR latch, which contains two two-input CEs and three input-split inverters. Figure 4(b) represents the modeling matrix for the SHLR latch. It has 5 rows and 5 columns, corresponding to the 5 components in the latch. The values in the matrix represent the input–output relationships of the components. "1" or "2" indicates a connection, while "0" indicates a disconnection. For example, in the fourth column of the matrix, there are two "1" values in the first and second rows, with the other values being "0". This indicates that the component with Q as the output has input n1 and input n2. It is important to note that the order of inputs for certain components can affect the output. Therefore, we must distinguish between inputs 1 and 2 or other inputs using different numbers in the matrix to represent the input order. For instance, in the first column, the values in the fourth and fifth rows are "1" and "2", respectively, with the other values being "0". This represents that the component with n1 as the output has inputs Q and Qa, where Q corresponds to input 1 and Qa to input 2. In Fig. 4(c), the two-dimensional data represents the initial state of the latch and the component type. For example, the two values in the first column are "0" and "0", indicating that the initial logic value of the n1 node is 0, and the component corresponding to the output n1 can be identified as an input-split inverter based on the Component Types of Fig. 1.



(a)

$$\begin{array}{c|ccccc} & n1 & n2 & n3 & Q & Qa \\ \hline n1 & 0 & 0 & 0 & 1 & 0 \\ n2 & 0 & 0 & 0 & 1 & 1 \\ n3 & 0 & 0 & 0 & 0 & 1 \\ Q & 1 & 1 & 1 & 0 & 0 \\ Qa & 2 & 2 & 2 & 0 & 0 \end{array}$$

(b)

n1	n2	n3	Q	Qa	
0	0	0	1	1	initial state
0	0	0	1	1	types of components

(c)

Fig. 4 **a** SHLR latch. **b** Modeled matrix of SHLR latch. (*Adjacency_Matrix*) **c** the two-dimensional array of the initial states and the components types of SHLR latch. (*State_Array*)

PMOS transistors may induce a 0- > 1 error in a node, while NMOS transistors may cause a 1- > 0 error [13]. When high-energy particles strike sensitive latch nodes, electrons and holes appear in pairs along their trajectory. Under the influence of an electric field, PMOS transistors push holes toward the drain, and NMOS transistors push electrons toward the drain [3]. Depending on the amount of charge collected, the drain voltages of PMOS and NMOS transistors may vary, resulting in different node error modes. A single impact of high-energy particles may cause one node to undergo a 0- > 1 error while causing another to undergo a 1- > 0 error. Therefore, it is necessary to consider all scenarios when conducting fault-tolerance tests.

To simulate all possible MNU scenarios, we can select all possible node combinations from all the nodes. Different latches have different combinations for MNU detection. Taking SHLR as an example, it has 5 nodes. There are C_5^1=5 possible scenarios for SNU detection (n1, n2, n3, Q, Qa), requiring 5 tests. There are C_5^2=10 possible scenarios for DNU detection (< n1, n2 > , < n1, n3 > , < n1, n4 > ,……, < n3, Qa > , < Q, Qa >), requiring 10 tests. For HQNUT, which has 16 nodes, if we perform QNU error detection, there will be C_{16}^4=1820 error combinations (< n1, n2, n3, n4 > , < n1, n2, n3, n5 > , < n1, n2, n3, n5 > ,……, < n13, n14, n15, Q >), requiring 1820 tests.. We create a list called *Test_lists* to store these node combinations. We select a node combination from the list each time and execute the algorithm to simulate errors occurring in the latch. We then check the changes in the output nodes. If any combination results in an output node error, the latch cannot achieve QNU tolerance.

Figure 5 presents the flowchart of the algorithm, and two stages divide into each node detection. In the first stage, we simulate the error propagation process. We create a list called *false_list* to store all the error nodes. Then, we execute the code from lines 10 to 17 to detect all the nodes not in the *false_list*.

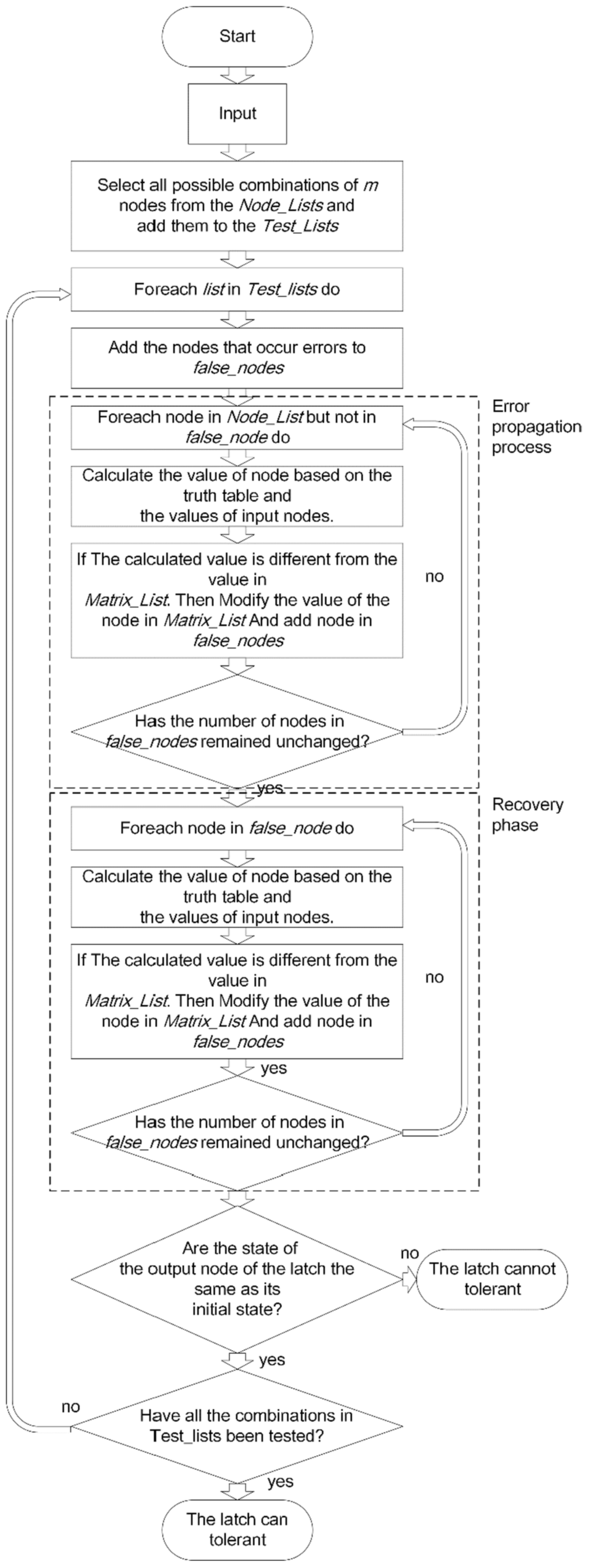

Start
Input
Select all possible combinations of *m* nodes from the *Node_Lists* and add them to the *Test_Lists*
Foreach *list* in *Test_lists* do
Add the nodes that occur errors to *false_nodes*
Foreach node in *Node_List* but not in *false_node* do
Calculate the value of node based on the truth table and the values of input nodes.
If The calculated value is different from the value in *Matrix_List*. Then Modify the value of the node in *Matrix_List* And add node in *false_nodes*
Has the number of nodes in *false_nodes* remained unchanged?
no
yes
Error propagation process
Foreach node in *false_node* do
Calculate the value of node based on the truth table and the values of input nodes.
If The calculated value is different from the value in *Matrix_List*. Then Modify the value of the node in *Matrix_List* And add node in *false_nodes*
yes
Has the number of nodes in *false_nodes* remained unchanged?
no
Recovery phase
Are the state of the output node of the latch the same as its initial state?
no
The latch cannot tolerant
yes
Have all the combinations in Test_lists been tested?
no
yes
The latch can tolerant

Fig. 5 The flowchart of the proposed algorithm

We calculate the value of this node based on the node's corresponding component type and the values of its input nodes. If a flip occurs, we add this node to the *false_list* until the *false_list* is no longer updated, which indicates that the error propagation has ended. Then, we enter the second phase, which is the recovery phase. We check all the nodes in the *false_list* and calculate their values based on the corresponding component type and input node values. If a node experiences a flip again (i.e., it is recovered), we remove it from the *false_list*. We repeat this process until the *false_list* no longer updates, indicating that all the recoverable nodes have been restored. In this way, we obtain the final list of node states. We check the state of the output node. When it changes, it indicates that the latch does not have the MNU tolerate ability, and we terminate the algorithm. When there is no change, we check the next error node combination to simulate other MNU scenarios. When there are no changes in the output final node state for all the node states, the latch can tolerate all possible MNU cases. Let us discuss the algorithm using an example with our proposed latch. Figure 6(a) shows the proposed HQNUT latch's modeling matrix. And in Fig. 6(b), the two-dimensional array represents the initial state of the latch and the types of components. Suppose the node list < n1, n8, n11, n12 > suffers from a QNU. At this point, the *false_list* contains four nodes: n1, n8, n11, and n12, the *false_list* is < n1, n8, n11, n12 > . The first values in columns 1, 8, 11, and 12 of the *State_Array* are flipped.

As shown in Fig. 7(b), first, we enter the error propagation stage, where we need to check all nodes not in the *false_list*. Taking node n10 as an example, according to the *Adjacency_Matrix*, we know that node n10 corresponds to the inputs n1 and n8. Referring to the *State_Array*, n10 as the output element type is 1, and the value of n1 and n8 is 1. According to the truth table of the component corresponding to the type 1 in Fig. 1, we can determine that the final output of n10 is 0, which is different from its previous value. Therefore, we flip the value of n10 and add it to the *false_list*. We also update the value of n10 in the State_Array to 0. Next, we move on to the next node not in the false_list, and repeat the above steps until the false_list no longer changes. The final states of State_Array are shown in Fig. 7(c). And the false_list is < n1, n8, n11, n12, n13, n14, n15, Q > .

Next, we move on to the recovery stage and only check the nodes in the *false_list*. Taking n1 as an example, according to the *Adjacency_Matrix*, we know that n1 corresponds to the input nodes n2 and n3. Referring to the State_Array, n1 as the output element type is 4, and the values of n2 and n3 are 1 and 0, respectively. Based on the truth table of the component relativing to type 4 in Fig. 1, we determine that the final output of n1 is 0,which differs from the value in State_Array. Therefore, n1 has been recovered and can be removed from the false_list. Additionally, the value of n1 in

$$\begin{array}{c|cccccccccccccccc}
 & N1 & N2 & N3 & N4 & N5 & N6 & N7 & N8 & N9 & N10 & N11 & N12 & N13 & N14 & N15 & Q \\
\hline
N1 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
N2 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
N3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
N4 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
N5 & 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
N6 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
N7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
N8 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
N9 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
N10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
N11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
N12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
N13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
N14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
N15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
Q & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

(a)

N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	Q
0	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0
4	4	1	4	4	1	4	4	1	1	1	1	1	1	1	3

(b)

Fig. 6 **a** Modeled matrix for the proposed HQNUT latch.(*Adjacency_Matrix*) **b** the two-dimensional array of the initial state and the types of components for the proposed HQNUT latch.(*State_Array*)

State_Array is updated to 1. We then check the next node in the false_list and repeat the above steps until the false_list no longer changes. The final state of State_Array is shown in Fig. 7(d). The false_list is null. We check the value of Q in State_Array and find that it is the same as the initial value. This indicates that the latch can tolerate the QNU of < n1, n8, n11, n12 > . Then, we check the next set of QNU and find that none of the QNU node combinations can affect the value of Q. Therefore, the latch can tolerate any QNU.

Here, a counterexample is presented. Let's consider the case where < n1, n2, n8, n11, n12 > suffers from errors. After the error propagation phase, the state of *State_Array* is shown in Fig. 8(c). After the recovery phase, the result is shown in Fig. 8(d). The final value of the Q node is 1, which is different from the initial value. Therefore, this latch cannot achieve quintuple-node-upset tolerance.

3.3 Simulations

This section will discuss the fault-tolerant principles of the HQNUT latch. If a latch can tolerate QNUs, it must tolerate SNUs, DNUs, and TNUs. So this paper mainly analyzes QNUs. Because of the symmetry in the latch's structure, we just need to analyze the following five cases:

Case Q1: In three SF modules, a QNU impacts no node. representative node list includes < n10, n11, n14, Q > . In this case, nodes n10, n11, n12, n13, n14, n15 and Q will be affected.However, none of the nodes in any SF modules is affected. All nodes can recover to their previous correct value through CEs. Therefore, the HQNUT has complete QNU tolerance for Case Q1.

Case Q2: In three SF modules, a QNU impacts at most one node. Because of the symmetry in the latch's structure, the representative node list only includes < n1, n10, n11, n12 > . In this case, nodes n1, n10, n11, n12, n13, n14, n15 and Q will be affected.Due to the characteristics of the SF modules, n1 will recover to its previous correct value by SF1. The recovery process following this will be the same as in Case 1, where all nodes will be recovered individually. Therefore, the HQNUT has complete QNU tolerance for Case Q2.

Case Q3: In three SF modules, a QNU impacts at most two nodes. Because of the symmetry in the latch's structure, the representative node list just contains < n1, n8, n11, n12 > , < n1, n2, n11, n12 > . When < n1, n8, n11, n12 > suffers from a QNU, because n1 and n8 is the input of C1, so n10 is impacted. n1, n10, n11, n12, n13, n14, n15 and Q will be flipped.Then n1 will be recovered

N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	Q
0	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0

(a)

1	0	1	0	0	1	0	1	1	1	0	0	0	0	1	0

(b)

1	0	1	0	0	1	0	1	1	1	0	0	1	1	0	1

(c)

0	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0

(d)

Fig. 7 The states of the nodes **a** The initial. **b** After suffered QNU. **c** After the first stage. **d** After the second stage

N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	Q
0	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0

(a)

1	1	1	0	0	1	0	1	1	1	0	0	0	0	1	0

(b)

1	1	0	0	0	1	0	1	1	1	0	0	1	1	0	1

(c)

1	1	0	0	0	1	0	0	1	0	0	1	1	1	0	1

(d)

Fig. 8 The states of the nodes. **a** The initial. **b** After suffered 5NU. **c** After the first stage. **d** After the second stage

through SF1, and n8 will be recovered through SF3. The recovery process following this will be the same as in Case 1, where all nodes will be recovered individually. The value of Q can be restored correctly. Therefore, < n1, n8, n11, n12 > can tolerate the QNU. Likewise, the other node lists can not change the value of Q. Therefore, the HQNUT has complete QNU tolerance for Case Q3.

Case Q4: In three SF modules, a QNU impacts at most three nodes. Because of the symmetry in the latch's structure, the representative node list just contains < n1, n5, n8, n11 > , < n1, n2, n4, Q > , < n1, n2, n3, n12 > . We are only considering the worst case. When < n1, n5, n8, n11 > is impacted by a QNU, n1, n7, n8, n9, n10, n11 and n13 will be flipped.The error propagation ends at this point. Because n13 and n14 are the input of C6, and n14 does not change, the output of C6 will not change. The value of Q does not change. Therefore, < n1, n5, n8, n11 > can tolerate the QNU. Likewise, the other node lists can not change the value of Q. Therefore, the HQNUT has complete QNU tolerance for Case Q4.

Case Q5: In three SF modules, a QNU impacts at most four nodes. Because of the symmetry in the latch's structure, the representative node list just contains < n1, n2, n4, n8 > , < n1, n2, n4, n5 > , < n1, n2, n3, n4 > . We are only considering the worst case. When < n1, n2, n4, n8 > is impacted by a QNU, n1, n2, n4, n8, n10, n11 and n13 will be flipped, n1, n2 can not be restored because an SF module cannot recover DNUs. Just one input of CE6 are impacted. So, the output Q of CE6 does not change its value because the characteristics of the CE. < n1, n2, n4, n8 > can not be impacted by the QNU. Similarly, the other node lists also have QNU tolerance. Therefore, the HQNUT has complete QNU tolerance for Case Q5.

The 32 nm CMOS technology [15] and the HSPICE simulation tool were used in the simulations. The supply voltage, temperature, clock period and the duty cycle of the simulation is 0.9 V, 27 ℃, 4000 ps and 50%. The ratio W/L of the PMOS transistors is 64/32 nm, and the ratio W/L of the NMOS transistors is 32/32 nm. We used the double exponential current source [2] in Eq. (1) in fault injection and set the rise time (τ_1) to 0.1 ps and the fall time (τ_2) to 3 ps.

$$I_{inj}(t) = \frac{Q_{inj}}{\tau_1 - \tau_2}\left(e^{\frac{-t}{\tau_1}} - e^{\frac{-t}{\tau_2}}\right) \tag{1}$$

Q_{inj} is the injection charge,t is the time of the fault injected, τ_1 is the aggregation time constant, and τ_2 is the ion trajectory establishment time constant [6].

Figure 9 illustrates the simulation outcomes demonstrating the tolerance of the HQNUT to SNU, DNU, and TNU. To ensure that D changes during the hold mode, D will change 50 ps after the CLK falls. For 0–6 ns, it is simulated toward the SNUs situations for nodes n1, n6, n10, n13, n15, and Q. They can be recovered to the correct state. For 6–14 ns, possible DNU cases are simulated, injecting fault into nodes pairs: < n1, n10 > , < n3, n10 > , < n3, n13 > , < n3, n15 > , < n3, Q > , < n13, n14 > , < n10, n13 > , < n1, n2 > . It's clear that no node pairs exert an influence on the output. For 14-30 ns, possible TNUs cases are simulated. Node sets < n1, n2, n3 > , < n1, n2, n4 > , < n2, n4,

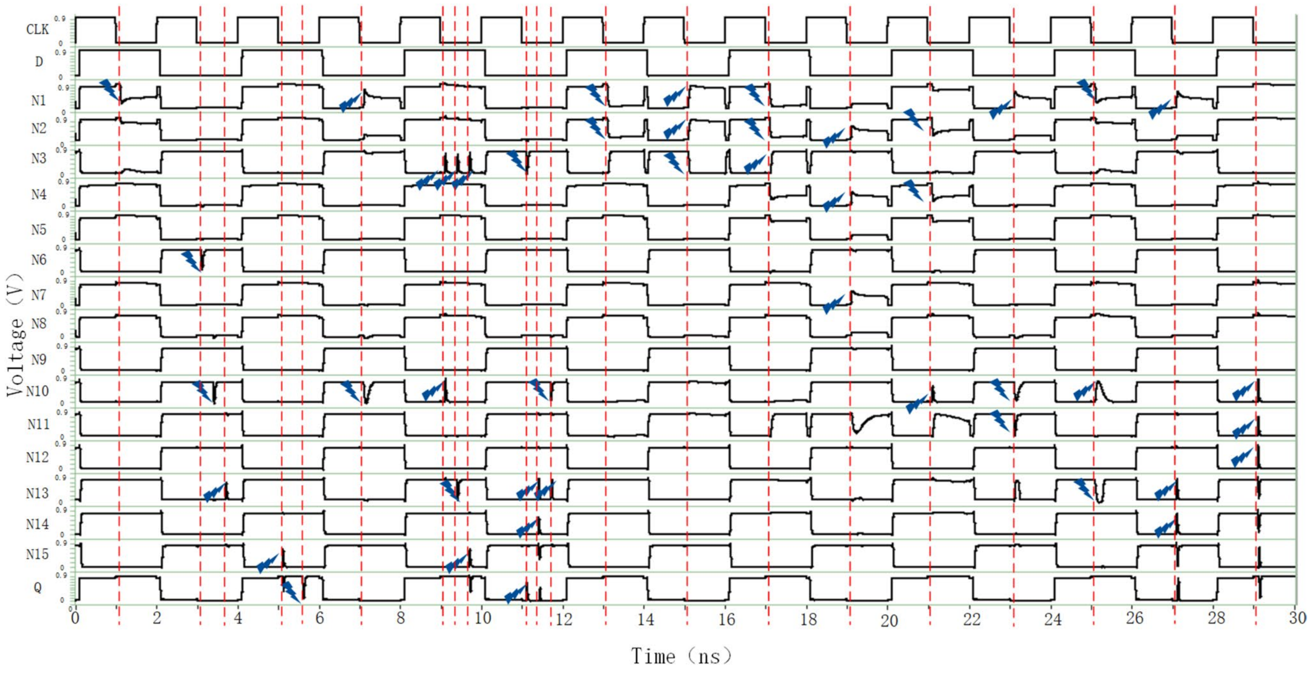


Fig. 9 Simulation waveforms for SNU,DNU and TNU cases

n7>,<n2, n4, n10>,<n1, n10, n11>,<n1, n10, n13>,<n1, n13, n14> and <n10, n11, n10> are provided as illustrative examples. The HQNUT can tolerate all node sets.

Figure 10 presents the simulation results of injecting QNU faults introduced in Section 3.3. Simulation results show that particle strikes can not influence the output nodes. All node sets cannot change the value of Q. Consequently, the HQNUT is completely QNU hardened. The simulation experiments show that the HQNUT has complete SNU, DNU, TNU, and QNU fault tolerance capability.

4 Evaluation and Comparative Results

In order to evaluate the performance of our proposed HQNUT latch, we conduct a relative analysis against several existing latch designs. This evaluation encompasses parameters such as power consumption, delay, and other relevant factors, all measured under identical conditions (the 32 nm PTM technology, the supply voltage is 0.9 V, the temperature is 27℃, W/L = 64/32 nm for the PMOS transistors, W/L = 32/32 nm for the NMOS transistors). At the same time, we utilized the algorithm from Section 3 to conduct complete testing on all these latches. We calculated how much of the errors these registers can intercept when faced with MNU, aiming to assess their fault-tolerant capabilities.

Table 1 compares different latches' SNU, DNU, TNU, and QNU fault tolerance capabilities. The table shows that SHLR can tolerate SNU but does not provide complete DNU fault tolerance. NTHLTCH and DNURL provide. complete SNU and DNU tolerance capabilities but only tolerate 41.67% of TNU. LCTNUT, LCTNURL, and TNURL can tolerate SNU, DNU, and TNU, but they only tolerate 32.1%, 74.09% and 92.1% of QNU respectively. Only D-latch, QNUHL, QNUTL-CG, and the proposed HQNUT have SNU, DNU, TNU, and QNU robustness, they offer the highest reliability against MNU.

Table 1 compares different latches' reliability, power consumption, delay, and area. "power" refers to the mean value of both dynamic and static power consumption, and "delay" is the transmission delay of D to Q. "Number of USTs" represents the area and "PDP" indicates the power-delay-product. Table 1 shows that compared to latch designs that cannot tolerate QNU, HQNUT requires a higher number of transistors. However, it significantly reduces power consumption and delays while improving reliability. Compared to other QNU latches, HQNUT uses fewer transistors and achieves low overhead without compromising reliability. Additionally, HQNUT exhibits a smaller PDP than other QNU-tolerant latches The HQNUT is cost-efficient.For a clearer comparison, Table 2 presents the relative saving of other QNU latches in terms of power (ΔPower), delay (ΔDelay), the number of transistors (ΔArea) and PDP (ΔPDP) compared to the HQNUT. Δ is the factor used for showing the percentage change. The relative overhead of the latch is calculated as follows:

$$\Delta = \frac{Compared\ Latch - Proposed\ Latch}{Proposed\ Latch} \tag{2}$$

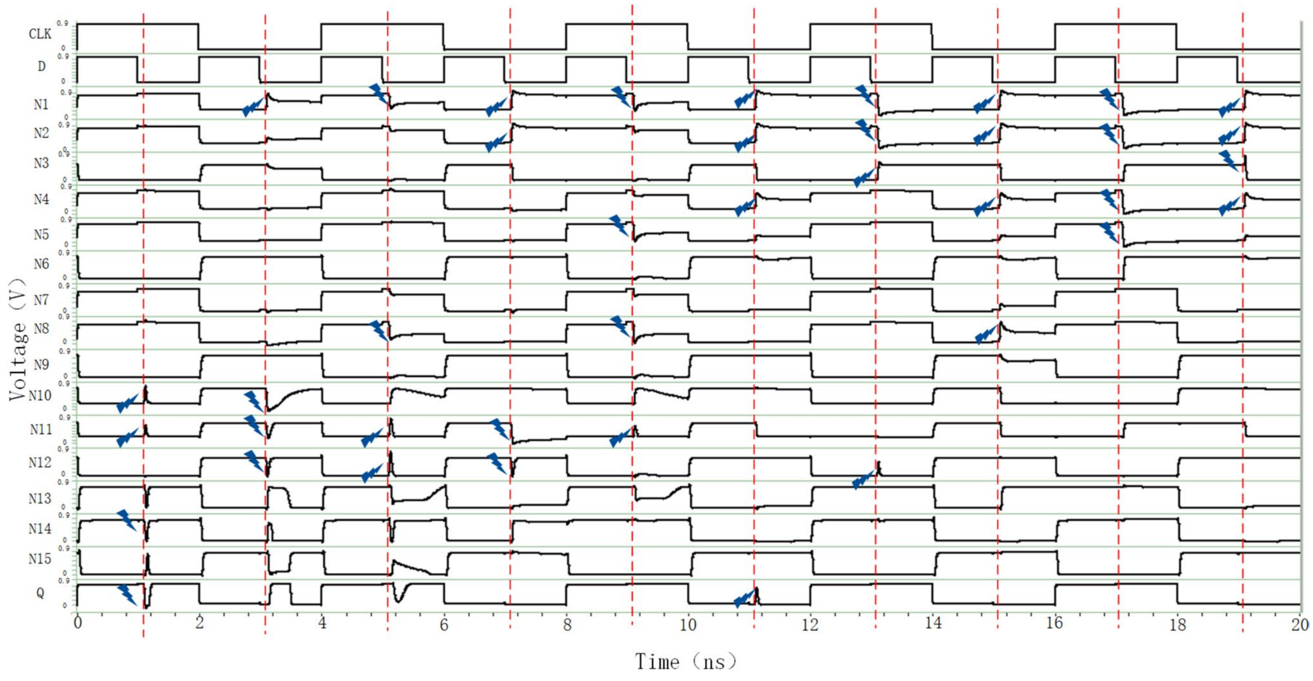


Fig. 10 Simulation waveforms for QNU cases

Table 1 Reliability and overhead comparisons among the hardened latches

Latch	SNU Tol	DNU Tol	TNU Tol	QNU Tol	Power (μw)	Delay(ps)	Number of transistors	PDP (10^{-18} J)
SHLR [8]	100%	0%	0%	0%	0.443	12.22	18	5.41
NTHLTCH [9]	100%	100%	67.86%	41.67%	0.883	15.59	58	13.76
DNURL [17]	100%	100%	67.86%	41.67%	0.9544	6.626	66	6.32
LCTNUT [19]	100%	100%	95.15%	32.1%	0.824	4.534	48	3.73
LCTNURL [12]	100%	100%	100%	74.09%	1.841	6.15	84	11.32
TNURL [18]	100%	100%	100%	92.1%	1.002	7.79	128	7.8
D-latch [5]	100%	100%	100%	100%	1.211	11.22	100	13.59
QNUHL [22]	100%	100%	100%	100%	0.819	4.54	76	3.72
QNUTL-CG [20]	100%	100%	100%	100%	0.831	4.54	76	3.77
Proposed	100%	100%	100%	100%	0.582	4.54	78	2.64

As shown in Table 2, compared to other existing QNU tolerant latches, the HQNUT latch exhibits a reduction of 19.8% in transmission delay, 36.9% in power consumption, 5.6% in transistor count, and 46.4% in PDP. In comparison, the HTNUT latch has the lowest overhead. Therefore, the proposed latch ensures reliability and achieves performance improvements superior to other latch designs. It reduces overhead without compromising reliability, providing significant cost-effectiveness.

During the development of integrated circuits, PVT can impact the performance of the latch. Thus, to secure that the latch reliable operation under PVT fluctuation [6], we employed the HSPICE simulation tool to conduct the analysis and evaluation of the latch. The sample standard deviation in Eq. (3) was employed for a more intuitive and accurate comparison of the power consumption and delay variations among different latches. A smaller sample standard deviation indicates a smaller range of fluctuations, indicating better stability of the latch.

$$s = \sqrt{\frac{\sum_{i=1}^{n} \left(x_i - \overline{x}\right)}{n-1}} \tag{3}$$

S,x_i, $\overline{x}$ and n represent the sample standard deviation, observation in the sample, sample mean and quantum of observations in the sample.

In our simulation, the supply voltage varied from 0.6 V to 1.2 V, and the temperature ranged from – 25 ℃ to 125 ℃. The process corners have five types: Fast NMOS Fast PMOS (FF), Fast NMOS Slow PMOS (FNSP), Typical NMOS Typical PMOS (TT), Slow NMOS Fast PMOS (SNFP) and Slow NMOS Slow PMOS (SS).

Table 2 Relative saving comparison of latches with the same functionality as HQNUT

Latch	ΔPower(%)	ΔDelay(%)	ΔArea(%)	ΔPDP(%)
D-latch	-51.9	-59.5	-22	-80.6
QNUHL	-28.9	0	2.6	-28.8
QNUTL-CG	-29.9	0	2.6	-29.9
Average	-36.9	-19.8	-5.6	-46.4

Figure 11 shows the impact of power supply voltage variations on the power consumption; Fig. 11(a) performances a line chart; it can be observed that the power consumption of the latch increases with the power supply voltage, and HQNUT has lower power consumption, LCTNURL has high. Figure 11(b) shows the stability histogram, calculated by Eq. (3); the smaller the value, the more stable the latch is. From Fig. 11(b), the HQNUT latch is the most stable; the LCTNURL is the least stable. The main reason is that LCTNURL does not use clock gating, resulting in current internal competition.

Figure 12 illustrates the change of delay under the different power supply voltages. From Fig. 12(a), the delay of the latch decreases with the power supply voltage. This phenomenon occurs because higher voltage leads to greater conduction current in the devices, resulting in increased power consumption and decreased delay [6, 16]. The delay of HQNUT is low. From Fig. 12(b), it can be observed that LCTNURL is the most stable, while HQNUT is also relatively stable, only slightly inferior to LCTNURL, and the NTHLTCH is the least stable.

Figures 13 and 14 show the power consumption and delay at different temperatures. We can see that HQNUT has lower power consumption from Fig. 13(a). Except for LCTNURL, the power consumption of other latches doesn't vary significantly with temperature. Among them, HQNUT exhibits relatively lower power consumption, as shown in Fig. 13(b).

Most of the latches show an increase in delay with rising temperatures from Fig. 14(a), with HQNUT's curve still positioned relatively lower. The changes in SHLR and NTHLTCH are significant, and the HQNUT latch is the most stable from Fig. 14(b).

Figure 15 shows the power consumption in process corner variation. It can be observed that latches have the highest

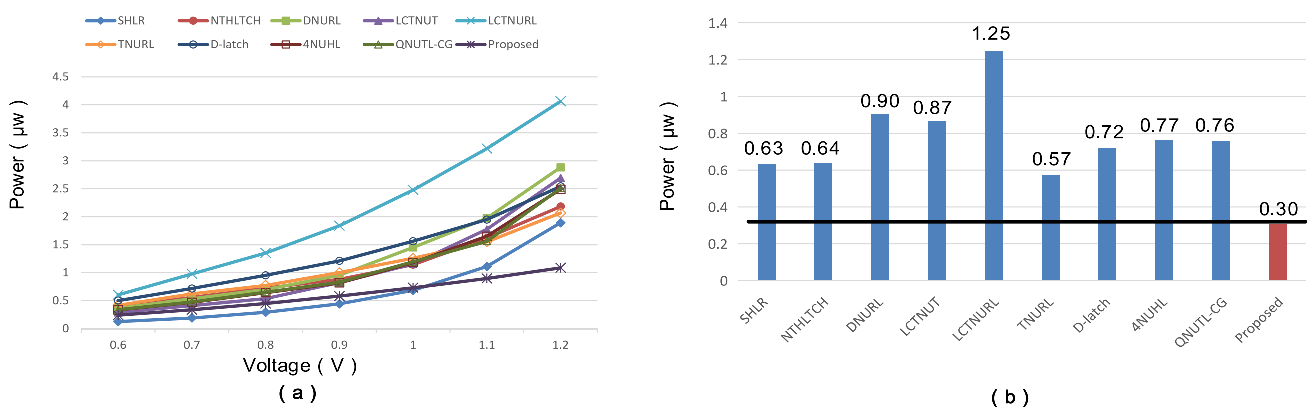


Fig. 11 **a** The impact of a different voltage on the power consumption. **b** Sample standard deviation for latches

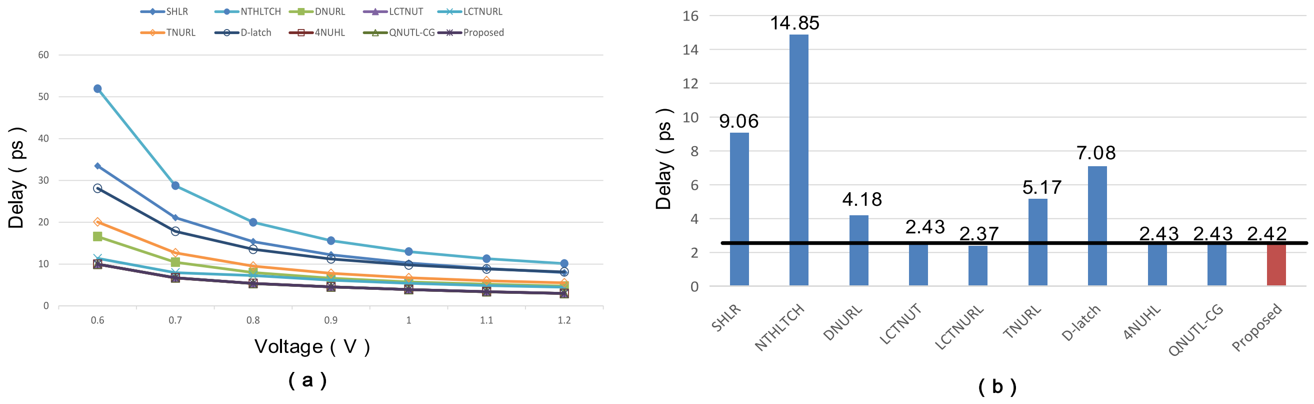


Fig. 12 **a** The impact of a different voltage on the delay. **b** Sample standard deviation for latches

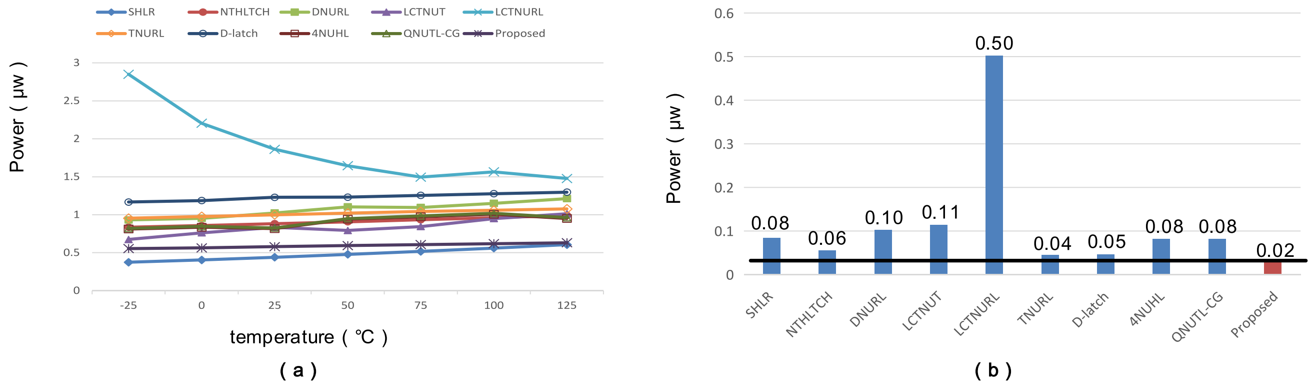


Fig. 13 **a** The impact of a different temperature on the power consumption. **b** Sample standard deviation for latches

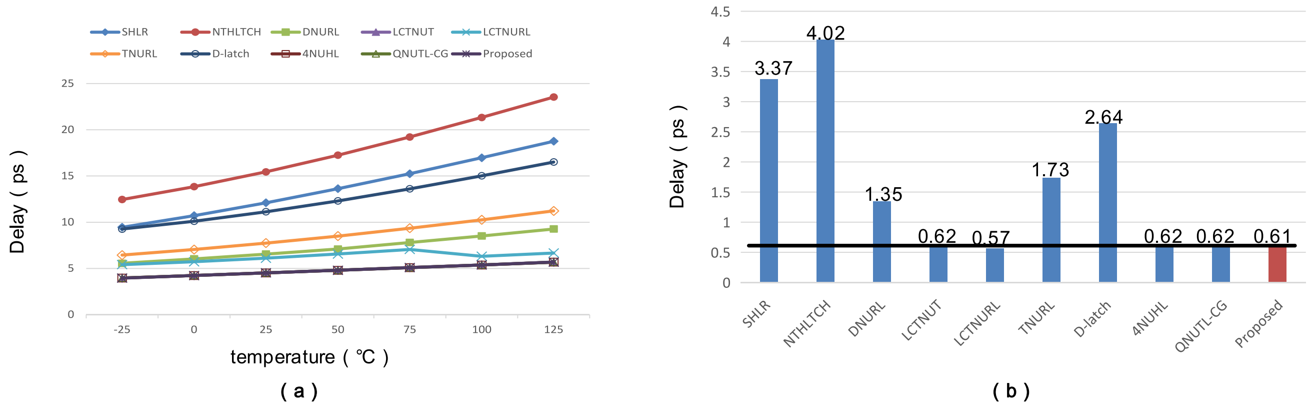


Fig. 14 **a** The impact of a different temperature on the delay. **b** Sample standard deviation for latches

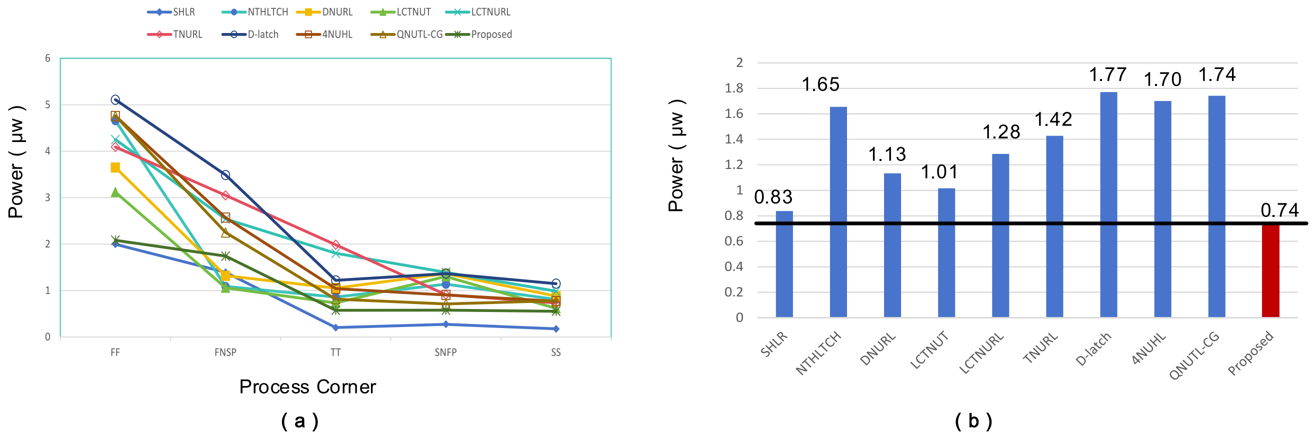


Fig. 15 **a** The impact of the process corner variation on the power consumption. **b** Sample standard deviation for latches

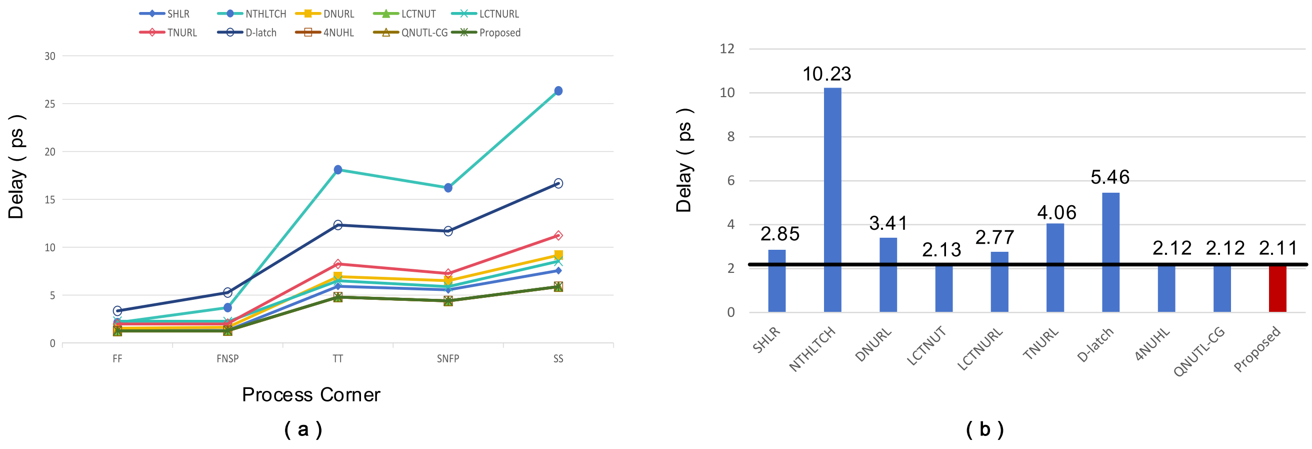


Fig. 16 **a** The impact of the process corner variation on the delay. **b** Sample standard deviation for latches

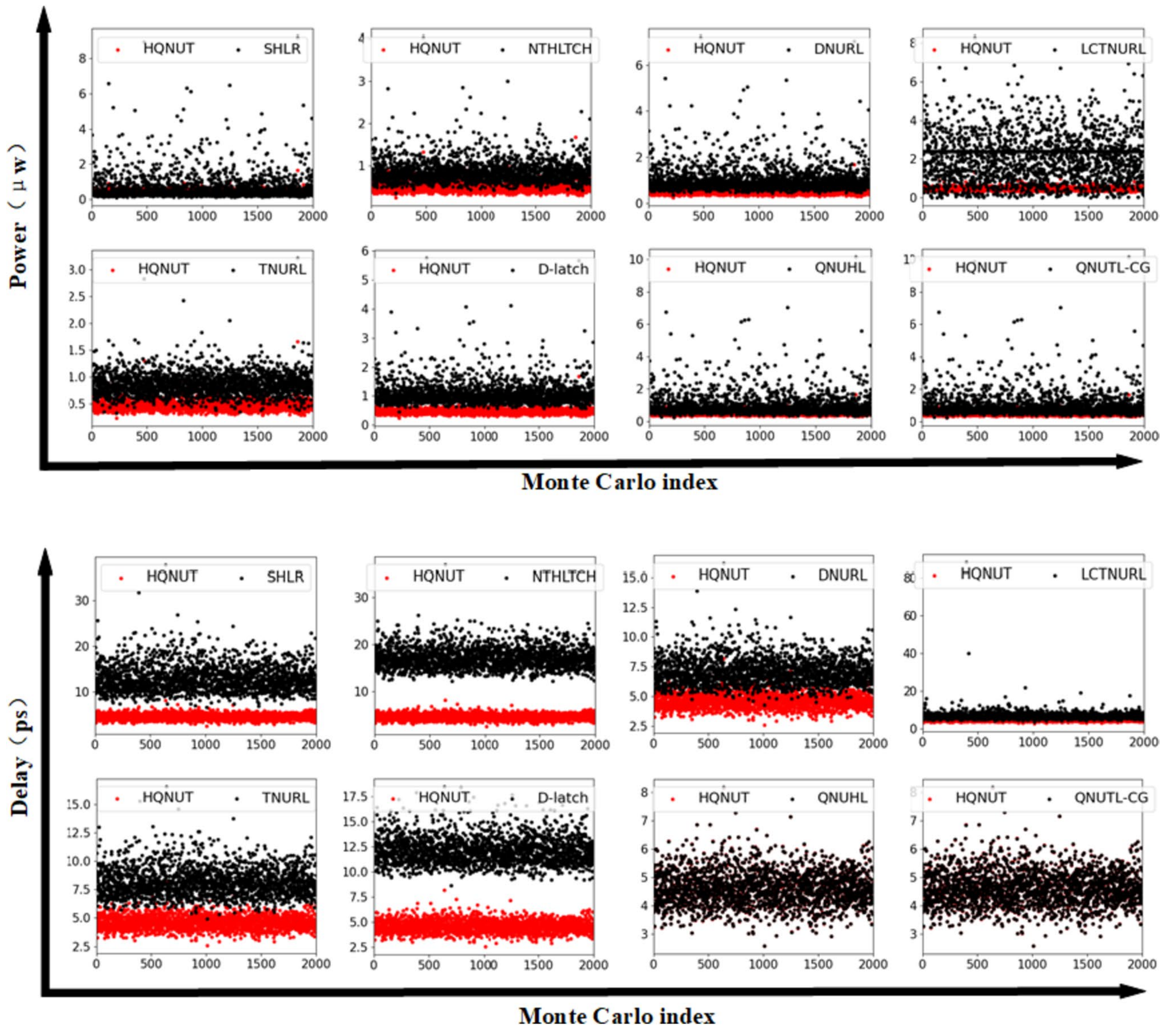


Fig. 17 Monte Carlo simulation

power consumption at FF and the lowest power consumption at SS, as seen in Fig. 15(a). HQNUT has lower temperature changes under different process corners from Fig. 15(b).

Figure 16 shows the delay in process corner variation. Relatively, latches have the lowest delay at FF and have the highest delay at ss from Fig. 16(a). From Fig. 16(b), the changes in NTHLTCH are quite significant. HQNUT latch is the most stable.

We conducted 2000 Monte Carlo simulations to assess the impact of PVT variations on radiation-hardened latches. The simulation contains the temperature, voltage and threshold voltage of transistors, Fig. 17 shows the simulation result. This simulations were executeed by sweeping the voltage and threshold voltage of transistors using $\pm 20\%$ Gaussian distribution with variation at the $\pm 3\sigma$ level. And the temperature uses absolute Gaussian distribution with variation at the $\pm 3\sigma$ level, and the fluctuation range is -25 ℃-125 ℃. From Fig. 17, it is evident that compared to other latches with the same functionality, HQNUT has lower power consumption than D-latch. Its power consumption is only slightly different from QNUHL and QNUTL-CG. And power data for HQNUT is more concentrated and has a lower dispersion degree, indicating that its power consumption is less sensitive to PVT variations. Additionally, HQNUT has a lower delay compared to the other three QNU-tolerant latches, with more concentrated delay data as well. So the delay of HQNUT is also insensitive to PVT variations.

In summary, our proposed latch demonstrates low sensitivity to PVT variation compared to other latch designs.

5 Conclusion

With the advancement of microelectronics technology, circuits are becoming increasingly susceptible to errors produced by high-energy radiation. Most existing latches lack resistance to QNU or have high overhead. Additionally, the current verification of latch reliability relies on various EDA and fault injection tools. This paper proposes an algorithm for verifying latch tolerance ability and a high-performance, low-power HQNUT latch for these issues. Experimental validation demonstrates that the algorithm can quickly and accurately verify whether a latch can tolerate various MNUs. Additionally, HSPICE simulation results demonstrate the QNU tolerance capability of the HQNUT latch. The HQNUT latch exhibits low power consumption, low delay, smaller area, and lower PDP than existing radiation-hardened latches. Suitable for use in satellites or other aerospace equipment. In the future, further research is needed to design more efficient and reliable latch structures.

Funding This work was supported in part by the Special Fund for Research on National Major Research Instruments of China under grant 62027815.

Data Availability The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflicts of Interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Black J, Dodd P, Warren K (2013) Physics of Multiple-Node Charge Collection and Impacts on Single-Event Characterization and Soft Error Rate Prediction. IEEE Trans. Nucl. Sci 60(3):1836–1851. https://doi.org/10.1109/TNS.2013.2260357
2. Black D, Robinson W, Wilcox I, Limbrick D, Black J (2015) Modeling of Single Event Transients With Dual Double-Exponential Current Sources: Implications for Logic Cell Characterization. IEEE Trans Nucl Sci 62(4):1540–1549. https://doi.org/10.1109/TNS.2015.2449073
3. Ebara M, Yamada K, Kojima K, Furuta J, Kobayashi K (2019) Process Dependence of Soft Errors Induced by Alpha Particles, Heavy Ions, and High Energy Neutrons on Flip Flops in FDSOI. IEEE J Electron Devices Soc 7:817–824. https://doi.org/10.1109/JEDS.2019.2907299
4. Eftaxiopoulos N, Axelos N, Pekmestzi K (2017) DIRT latch: A novel low cost double node upset tolerant latch. Microelectron Reliab 68:57–68. https://doi.org/10.1016/j.microrel.2016.11.006
5. Hatefinasab S, Ohata A, Salinas A, Castillo E, Rodriguez N (2022) Highly Reliable Quadruple-Node Upset-Tolerant D-Latch. IEEE Access 10:31836–31850. https://doi.org/10.1109/ACCESS.2022.3160448
6. Huang Z, Liang H, Hellebrand S (2015) A high performance SEU tolerant latch. J J Electron Test 31:349–359. https://doi.org/10.1007/s10836-015-5533-5
7. Kumar C, Anand B (2019) A Highly Reliable and Energy-Efficient Triple-Node-Upset-Tolerant Latch Design. IEEE Trans. Nucl. Sci 66(10):2196–2206. https://doi.org/10.1109/TNS.2019.2939380
8. Kumar S and Mukherjee A (2021) A Self-Healing, High Performance and Low-Cost Radiation Hardened Latch Design. 2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Athens, Greece, pp 1–6. https://doi.org/10.1109/DFT52944.2021.9568359
9. Li Y, Wang H, Yao S, Yan X, Gao Z, Xu J (2015) Double node upsets hardened latch circuits. J Electron Test 31(5):537–548. https://doi.org/10.1007/s10836-015-5551-3
10. Lin D, Wen C (2020) DAD-FF: Hardening Designs by Delay-Adjustable D-Flip-Flop for Soft-Error-Rate Reduction. IEEE Trans Very Large Scale Integr VLSI Syst 28(4):1030–1042. https://doi.org/10.1109/TVLSI.2019.2962080
11. Mitra S, Zhang M, Seifert N, Mak T, Kim K (2007) Built-In Soft Error Resilience for Robust System Design. 2007 IEEE International Conference on Integrated Circuit Design and Technology, Austin, TX, USA, pp1–6. https://doi.org/10.1109/ICICDT.2007.4299587
12. Nan H, Choi K (2012) High Performance, Low Cost, and Robust Soft Error Tolerant Latch Designs for Nanoscale CMOS Technology. IEEE Trans Circuits Syst I Regul Pap 59(7):1445–1457. https://doi.org/10.1109/TCSI.2011.2177135
13. Saki T, Masao Y, Shi Y (2019) Transition Detector-Based Radiation-Hardened Latch for Both Single- and Multiple-Node Upsets. IEEE Trans Circuits Syst II 67(6):1114–1118. https://doi.org/10.1109/TCSII.2019.2926498
14. Watkins A, Tragoudas S (2020) Radiation Hardened Latch Designs for Double and Triple Node Upsets. IEEE Trans Emerg Top Comput 8(3):616–626. https://doi.org/10.1109/TETC.2017.2776285
15. Xu H, Sun C, Zhou L, Liang H, Huang Z (2021) Design of a Highly Robust Triple-Node-Upset Self-Recoverable Latch. IEEE Access 9:113622–113630. https://doi.org/10.1109/ACCESS.2021.3104335
16. Yan A, Liang H, Huang Z, Jiang C, Ouyang Y, Li X (2016) An SEU resilient, SET filterable and cost effective latch in presence of PVT variations. J Microelectron Reliab 63:239–250. https://doi.org/10.1016/j.microrel.2016.06.004
17. Yan A, Huang Z, Yi M, Xu X, Ouyang Y, Liang H (2017) Double-Node-Upset-Resilient Latch Design for Nanoscale CMOS Technology. IEEE Trans Very Large Scale Integr VLSI Syst 25(6):1978–1982. https://doi.org/10.1109/TVLSI.2017.2655079
18. Yan A, Feng X, Hu Y, Lai C, Cui J, Chen Z, Miyase K, Wen X (2020) Design of a Triple-Node-Upset Self-Recoverable Latch for Aerospace Applications in Harsh Radiation Environments. IEEE Trans Aerosp Electron Syst 56(2):1163–1171. https://doi.org/10.1109/TAES.2019.2925448
19. Yan A, Lai C, Zhang Y, Cui J, Huang Z, Song J, Guo J, Wen X (2021) Novel Low Cost, Double-and-Triple-Node-Upset-Tolerant Latch Designs for Nano-scale CMOS. IEEE Trans Emerg Top Comput 9(1):520–533. https://doi.org/10.1109/TETC.2018.2871861
20. Yan A, Xu Z, Feng X, Cui J, Chen Z, Ni Z, Ni T, Huang Z, Girard P, Wen X (2022) Novel Quadruple-Node-Upset-Tolerant Latch Designs With Optimized Overhead for Reliable Computing in Harsh Radiation Environments. IEEE Trans Emerg Top Comput 10(1):404–413. https://doi.org/10.1109/TETC.2020.3025584
21. Yan A, Li Z, Cui J, Huang Z, Ni T, Girard P, Wen X (2023) LDAVPM: A Latch Design and Algorithm-Based Verification Protected Against Multiple-Node-Upsets in Harsh Radiation Environments. IEEE T COMPUT AID D 42(6):2069–2073. https://doi.org/10.1109/10.1109/TCAD.2022.3213212

22. Yan A, Ding L, Shan C, Cai H, Chen X, Wei Z, Huang Z, Wen X (2021) TPDICE and Sim Based 4-Node-Upset Completely Hardened Latch Design for Highly Robust Computing in Harsh Radiation. 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Korea, pp 1–5. https://doi.org/10.1109/ISCAS51556.2021

Hui Xu received the Ph.D. degree in computer application technology from Hefei University of Technology, Hefei, in 2013. He is currently a Professor with Anhui University of Science and Technology. His research interests include aging tolerating and radiation hardening by design for nano-scale CMOS ICs, such as latches, flip-flops, and memory cells.

Xuewei Qin received the B.S. degree from Hefei University, Hefei, in 2020. He is currently pursuing the M.S. degree in computer science major with Anhui University of Science and Technology. His current research interests include reliability design and fault tolerance on integrated circuits.

Ruijun Ma received an M.E. degree in Electric Engineering from Anhui University of Science and Technology, China, in 2017. He received a Ph.D. degree in the Department of Creative Informatics at Kyushu Institute of Technology, Japan, in 2022. His research interests include soft-error tolerance and design-for-test.

Chaoming Liu received the B.S. degree from College of Continuing Education of Nanjing University of Aeronautics and Astronautics, in 2020. He is currently pursuing the M.S. degree in software engineering major with Anhui University of Science and Technology. His current research interests include reliability design and fault tolerance on integrated circuits.

Shuo Zhu is currently pursuing the M.S. degree in computer science and technology major with Anhui University of Science and Technology. His current research interests include reliability design and fault tolerance on integrated circuits.

Jun Wang is currently pursuing the M.S. degree in software engineering major with Anhui University of Science and Technology. His current research interests include reliability design and fault tolerance on integrated circuits.

Huaguo Liang received the Ph.D. degree in computer science from the University of Stuttgart, Germany, in 2003. He is currently a Professor and PhD supervisor with the School of Microelectronics, Hefei University of technology, Hefei, China. He has taken charge of many projects (e.g. DFG, National Natural Science Foundation, Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Min istry). His main research fields include SoC and VLSI design & test, etc.