5-2014

Radiation-Hardened Delay-Insensitive Asynchronous Circuits for Multi-Bit SEU Mitigation and Data-Retaining SEL Protection

John Davis Brady
University of Arkansas, Fayetteville

Radiation-Hardened Delay-Insensitive Asynchronous Circuits for
Multi-Bit SEU Mitigation and Data-Retaining SEL Protection

Radiation-Hardened Delay-Insensitive Asynchronous Circuits for Multi-Bit SEU Mitigation and Data-Retaining SEL Protection

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering

by

John Brady
University of Arkansas
Bachelor of Science in Computer Engineering, 2012

May 2014
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Dr. Jia Di
Thesis Director

Dr. J. Patrick Parkerson
Committee Member

Dr. Dale Thompson
Committee Member

Abstract

Radiation can have highly damaging effects on circuitry, especially for space applications, if designed without radiation-hardening mechanisms. Delay-insensitive asynchronous circuits inherently have promising potentials in mitigating the effects of radiation due to their delay insensitivity. This thesis proposes the use of two delay-insensitive asynchronous logic architectures to mitigate the effects of up to two single-event upsets (SEU) and a single-event latch-up (SEL). The multi-bit SEU mitigation with SEL protection architecture improves the original design by providing more integrity against data corruption and lock-ups caused by multi-bit SEUs, and it is expanded to simultaneously provide protection against SEL. The multi-bit SEU mitigation with data-retaining SEL protection architecture extends the original architecture by guaranteeing no data loss during the power cycling for mitigating SEL. The results show that the proposed architectures function correctly, at the transistor level, in mitigating up to two SEUs and an SEL without data loss.

Acknowledgments

I thank Dr. Jia Di for allowing me to work in this area. I would not have accomplished so much without his support and guidance. Additionally, I thank my fellow lab members for sharing their technical experience and helping to solve the day-to-day issues. I would also like to thank Dr. J. Patrick Parkerson and Dr. Dale Thompson for taking the time to be involved with my work.

Table of Contents

1. Introduction

Radiation from outer space poses a serious threat to integrated circuits because of its ability to cause data corruption and even permanent damage. More specifically, a Single-Event Upset (SEU) can cause a change of state for an internal node by discharging upon contact with the circuit. While this type of event is non-destructive, the resulting change of state can lead to a permanent effect in the data should it be latched and propagate through the rest of the circuit. Without a golden reference or SEU-detection circuit, it may not be possible to know the validity of the resulting data.

A Singe-Event Latch-Up (SEL) is an event in which a charged particle causes a short circuit, for example, between power and the substrate. This short can have a destructively high current that, if not dealt with, can cause permanent damage [1]. Since a power cycle is required to correct a SEL, current data propagating through the circuit is lost and must be recalculated. The result of the possibility of both an SEU and an SEL is not only potential data loss, corruption, or permanent damage, but also unreliability.

Because of the effects of radiation, special techniques and architectures have to be implemented in order to ensure the security and reliability of a circuit. Delay-insensitive asynchronous circuits offer an advantage in this area because of their innate ability to accommodate the delay resulting from mitigating radiation effects. NULL Convention Logic (NCL), an asynchronous logic design paradigm, provides a solid foundation for detecting an SEU, due to its dual-rail logic nature, and accommodates the interrupt and delay resulting from an SEL.

The Multi-Bit SEU Mitigation with SEL Protection Architectures presented in this thesis expands on an original methodology to prevent the effects of an SEU. Original components are modified to add increased protection to particularly vulnerable areas. This architecture also adds a protection component designed to detect the event of a SEL, correct the results of this event, and restore the circuit to normal operation. Since this is a pipelined architecture, an individual stage, where the SEL occurs, can be power-cycled and restored to correct operation without interrupting other stages of the circuit.

The second architecture, Multi-Bit SEU Mitigation with Data-Retaining SEL Protection, expands the first architecture to guarantee no data loss during the event of an SEL. This addition takes advantage of the pipeline-nature in creating a DATA redundancy solution. The ability to guarantee that no data will be lost during an SEL requires no additional delays beyond power-cycling and resetting the specific stage, as required for the first architecture.

The abilities of both architectures are tested using the 130-nm IBM 8RF bulk CMOS process. A multiplier-based test circuit is used to simulate the architectures' correct working order, as well as their defense against a two-bit SEU and an SEL. The results are promising and show the architectures' role in the future of radiation mitigation.

This thesis is organized in the following manner: Chapter 2 introduces NCL and explains the original methodology that is the foundation for this thesis; Chapter 3 explains each component and the requirements of the Multi-Bit SEU Mitigation with SEL Protection Architecture; Chapter 4 covers the Multi-Bit SEU Mitigation with Data-Retaining SEL Protection Architecture; Chapter 5 shows the normal operating conditions of the architectures along with simulating the effects of a two-bit SEU and an SEL; and Chapter 6 provides conclusions and future work for this thesis.

2. Background

2.1 NCL

NULL Convention Logic (NCL) is a delay-insensitive asynchronous circuit design methodology that uses multi-rail logic, in this case dual-rail logic, to create delay-insensitivity [2]. Each dual-rail logic signal has three possible states: DATA0, DATA1, and NULL. DATA0 ($\text{Signal}^0 = 1$ and $\text{Signal}^1 = 0$) corresponds to Boolean logic 0 and DATA1 ($\text{Signal}^0 = 0$ and $\text{Signal}^1 = 1$) to Boolean logic 1. The NULL state ($\text{Signal}^0 = 0$ and $\text{Signal}^1 = 0$) is used to signify the signal does not have a DATA value. The two rails of each signal are mutually exclusive, meaning the last state ($\text{Signal}^0 = 1$ and $\text{Signal}^1 = 1$) is an invalid state.

NCL circuits use a composition of 27 fundamental gates, creating a complete set of functions for up to four single-rail inputs. Most often a separate set of logic consisting of these 27 gates is used to find the value of each rail. Each gate is a threshold gate following the naming convention, THmn, where at a minimum *m* of the *n* inputs are required to be asserted for the output of the gate to be asserted. This symbol is shown in Figure 1.

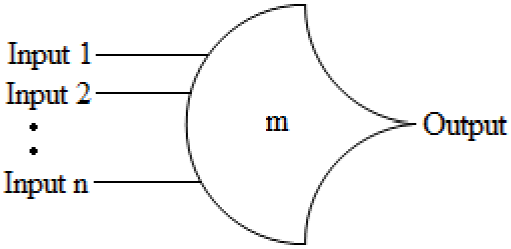

Figure 1: THmn gate symbol [2].

NCL gates use hysteresis which requires the output to remain asserted until all inputs have been deasserted. As a result, all inputs must be deasserted before each new set of data to ensure the output holds the correct result of the current data instead of retaining the result of the previous set of data.

The threshold gate structure above is expanded to add weight to individual inputs such that an input can be worth more than one, following the naming convention ThmnW$w_1w_2..w_R$ [4]. The subscript of each weight refers to the input of the gate where *R* is less than or equal to *n*, the number of inputs. Each weight is assigned as an integer to the input denoted in subscript, for example TH34W2. For this gate, the first input has a weight of two, which means the *m* requirement can be met if the first input and only one of the other three inputs are asserted. A depiction of this threshold gate's structure is shown in Figure 2.

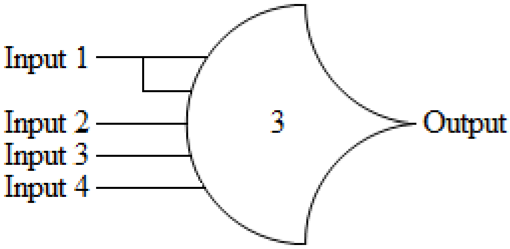

Figure 2: TH34W2 symbol [4].

In addition, many gates have reset functionality. If the output is deasserted when the gate is instructed to reset, it is said that the gate resets to '0'. This type of gate is specified by adding an *n* on the end of the gate's name, for example TH22n. Likewise, if the output of the gate is asserted when reset, it is said that the gate resets to '1' and is specified by a *d* following the gate name, for example TH22d. Resettable gates are used in all registers to enable the circuit to be reset to a known state.

NCL circuits contain at least two delay-insensitive (DI) register sets at the beginning and end of the circuits to create stages, as shown in Figure 3.

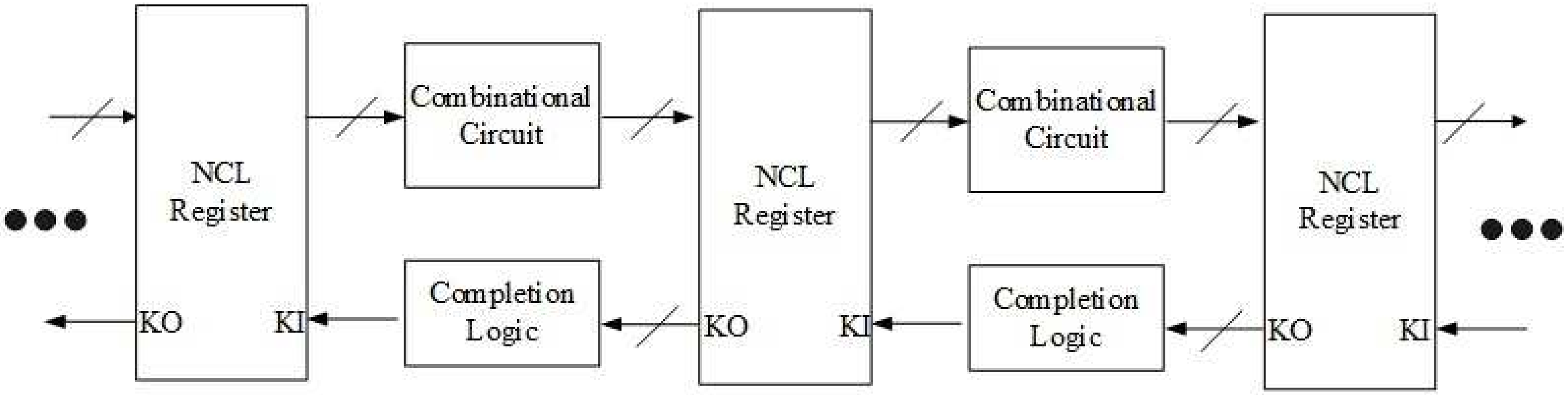

Figure 3: NCL architecture [2].

Register sets communicate through the use of *KO* signals. When a register set detects, through the use of a completion logic block, that the outputs of each register is all DATA or all NULL, the register set sends a *KO* (0) requesting for NULL (*rfn*) or a *KO* (1) requesting for DATA (*rfd*), respectively.

Each register uses resettable gates that take as input the dual-rail logic signal and *KI* signal. By employing the use of the *KI* signal in each register, it is ensured that the register is unable to change its output to DATA or NULL until the following register set makes a request for DATA or NULL.

Because of the structure of the registers, every two consecutive DATA waves propagated through the circuit are separated by a NULL wave. This prevents DATA from being overwritten by a preceding or succeeding DATA wave since a stage is flushed before a new DATA wave propagates through.

2.2 Original Architecture

The architecture proposed by this thesis is based off of an architecture (Figure 4) that modifies the standard NCL architecture and uses double-modular redundancy to prevent SEUs from corrupting data as it propagates through each stage of the pipeline [5]. This original design modifies the standard NCL architecture in the following ways:

- Add a second copy of the circuit
- Replace the TH22n gates in the registers with TH33n
- Add a set of TH22 gates at the output of each register set (two register sets per stage since there are now two copies of the circuit)
- Remove the completion logic from the register sets and add it to the end of the TH22

sets

- Add a second completion signal.

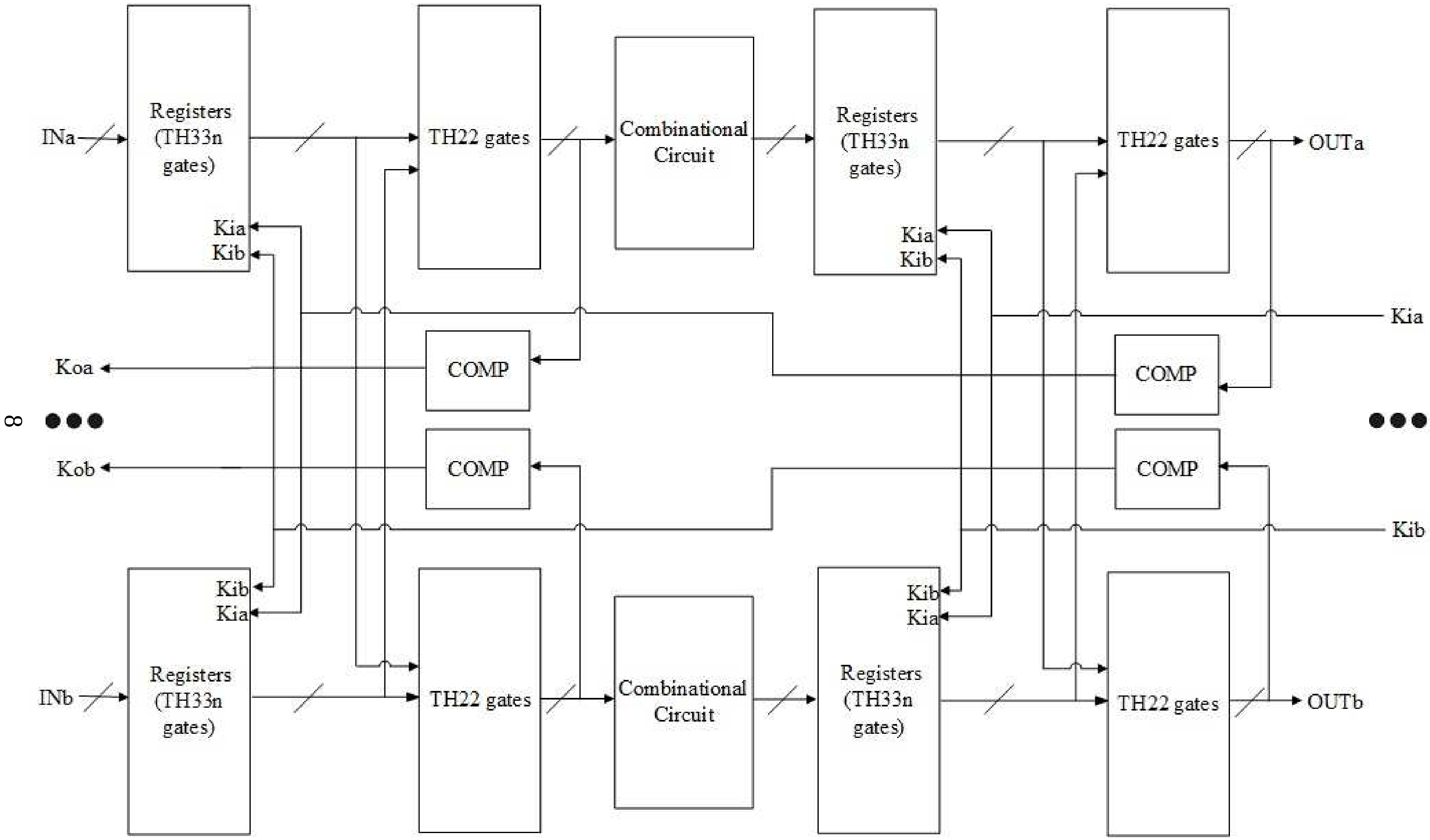

Figure 4: Original Architecture [5].

The second copy of the circuit is used to determine that the outputs of both register sets are equivalent at each stage, meaning the output is correct. This is achieved by adding a set of TH22 gates at the output of the registers. In each set of TH22 gates, there is one TH22 gate per bit being compared. Since dual-rail signals use two bits to represent a logic value, two TH22 gates are required to compare two dual-rail signals. One of the TH22 gates is used to compare the rail^{0} component of the signal, while the other gate compares the values of each signal's rail^{1} component. In order for the output of the TH22 gates to change, both pairs of rail^{1} and rail^{0} have to be the same. Otherwise, the output will remain in its previous state. If an SEU occurs in either copy of the circuit, the corrupted rail will be compared with its counterpart from the opposite circuit. Since the two values are not the same, the output of the TH22 gate will not change. Once the SEU subsides, the corrupted rail will change to the correct value, which will be compared to its counterpart in the opposite circuit. This will cause the output of the effected TH22 gate to be in the correct state.

Once the output of each TH22 set is correct, the data is propagated through the completion logic that was moved to the output of the TH22 sets. The completion logic determines when all dual-rail signals are DATA, whether it is DATA0 or DATA1, or NULL. Since the input to the completion logic is based on data that has been compared by the TH22 sets to ensure correct values, the output of the completion logic will be correct. There are two sets of completion logic. One is attached to the output of the first circuit's TH22 set and the other is attached to the output of the second circuit's TH22 set. This ensures that if an SEU occurs in between the TH22 sets and their corresponding completion logic block, it does not corrupt the completion signals which communicate to the previous register set when the stage is ready for NULL or ready for DATA.

As stated above, this architecture uses two completion signals to communicate when a stage is ready for NULL or ready for DATA. Both signals have to have the same value before a register set will send a NULL wave or DATA wave. With this design, an SEU can occur on one of the *KO* lines, corrupting the propagation of DATA and NULL waves. Since there are two *KO* signals instead of one, each register in a register set needs to devote two inputs to *KO* signals and one input to one rail of a data signal. This results in using TH33n gates in the register sets instead of TH22n. With two *KO* signals being used, the register ensures that both *KO* signals are correct and will know when one of the signals is effected by an SEU, due to its incorrect value.

2.3 Single-Bit SEU Mitigation

Since the circuit is now dual-modular redundant, a type of voting system can be used to determine if there is an error caused by an SEU. Regardless of which rail of a signal is affected by the SEU, the signal's counterpart in the opposite module will serve to show the error because the two rails do not match. The TH22 gates that take both register sets' outputs perform the role of the voting system. If the outputs of each register set do not match, then data cannot proceed past the TH22 gates.

It is important to note that the specific affected signal does not proceed, since the signals are compared one-by-one. Since there is one signal that has not matched its counterpart in the opposite circuit, the COMP component does not detect a full DATA wave. This requires the stage to wait until the signal that was affected to regain its correct value before a full DATA wave is detected, and a NULL wave can be requested. This process prevents corrupted data from continuing. Each stage will hold until all signals match their counterparts, meaning the values propagating to the next stage are correct.

As soon as the SEU subsides, the data becomes correct, which is detected by TH22 gates. As soon as the DATA or NULL matches, meaning it is correct, it can proceed past the TH22 gates into the completion blocks and combinational logic of the next stage.

The completion blocks create two separate KO signals to indicate that the output of each TH22 set is completely DATA or NULL. Should an SEU affect one of the *KO* signals, both register sets will not incorrectly allow their outputs to change because it requires both *KO* signals to be 1 or 0 before the output can change to DATA or NULL.

3. Multi-Bit SEU Mitigation with SEL Protection Architecture

This architecture has two main purposes:

1. Prevent up to two simultaneous SEUs from corrupting data or causing deadlock
2. Detect an SEL and restore normal operation

The diagram for this architecture is presented in Figure 5, including all of the added components.

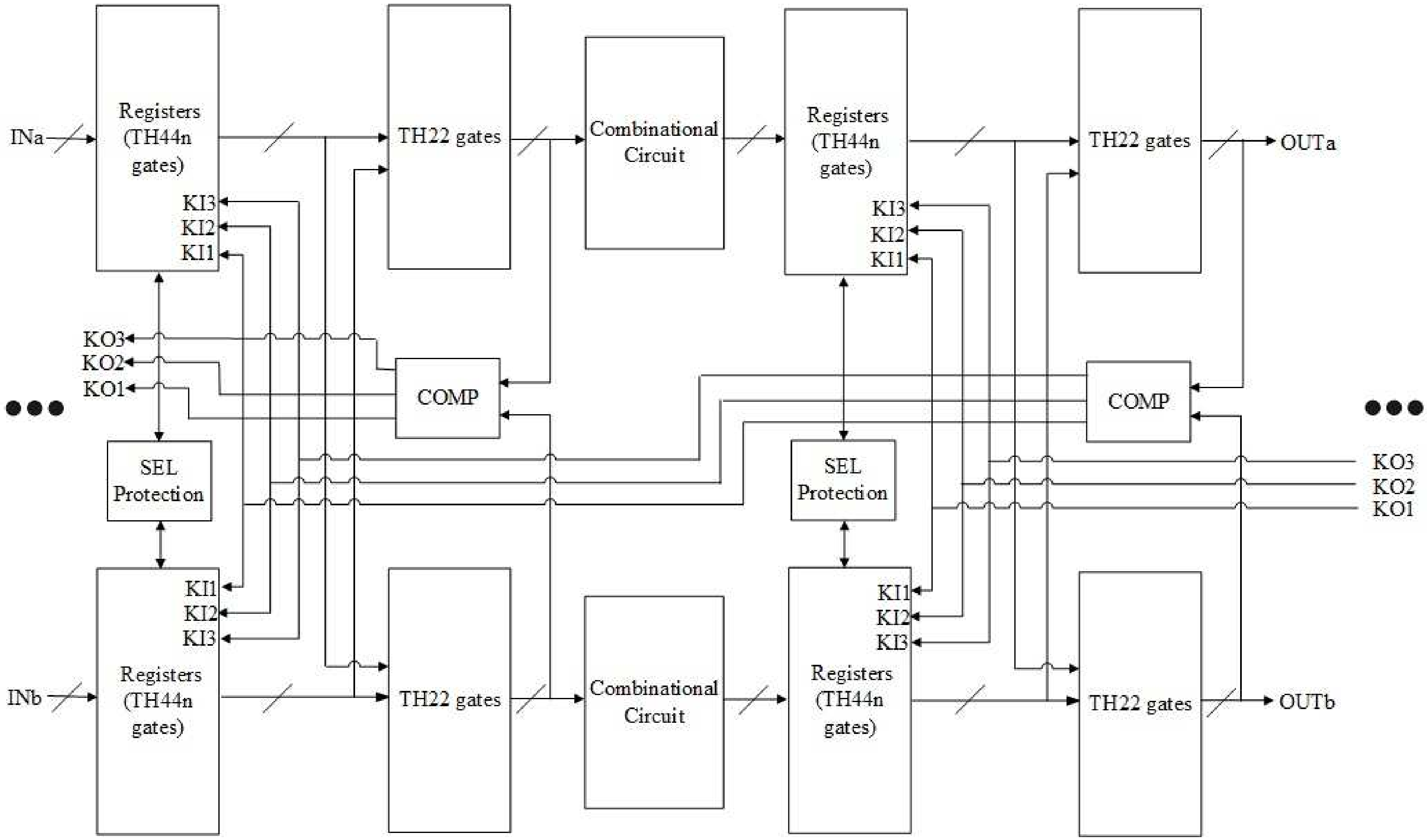

Figure 5: Multi-bit SEU mitigation with SEL protection architecture diagram.

Before describing the process by which the architecture mitigates SEUs and SELs, each component's addition or modification is explained for its place in improving the original architecture, beginning with SEU-mitigation components.

The following components enable the architecture to prevent errors from up to two simultaneous Single-Event Upsets: TH22 gates, COMP, and registers using TH44n gates.

3.1 TH22 Gates Component

Instead of sending the data from the registers directly to the combinational logic and completion block, the data from the output of each register set is sent to two sets of TH22 gates. These sets of gates serve to ensure the validity of the data coming from each register set by comparing the data to ensure faultlessness. This is detected by the use of two TH22 gates per set of dual-rail logic data being compared. For a set of dual-rail logic data being compared, for example *input1* and *input2*, both *input1*^{0} and *input2*^{0} are input to the first TH22 gate while *input1*^{1} and *input2*^{1} are input to the second TH22 gate, as shown in Figure 6.

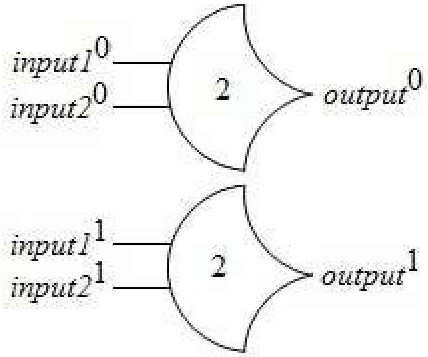

Figure 6: TH22 component for comparing two dual-rail logic signals.

This structure prevents the output of both TH22 gates from changing until the rail^0 and rail^1 components of *input1* and *input2* are the same.

Once the data being compared is the same for all inputs, the output of the TH22 gates changes to reflect the same value as the data output from both register sets. This data, which has been confirmed to be faultless, is now sent to the COMP component and combinational logic block for the next stage.

3.2 COMP Component

For this architecture, a more robust wave signaling protocol is implemented by using three *KO* signals instead of two. Without this change, this methodology is very susceptible to lock-up from two SEUs occurring on the *KO* signals, which will be further discussed later chapter 5.

Three *KO* signals are now used, instead of two, to ensure a stage's NULL wave or DATA wave is complete and a new DATA wave or NULL wave is requested, respectively, which requires a modified COMP component (Figure 7). These three separate completion signals are created from the *INa* and *INb*, which come from the top and bottom TH22 blocks respectively. This ensures that the KO signals are correct. Without this check, it would have to be assumed that an SEL did not occur between the output of the TH22 components and the output of the COMP component, which are the three *KO* signals.

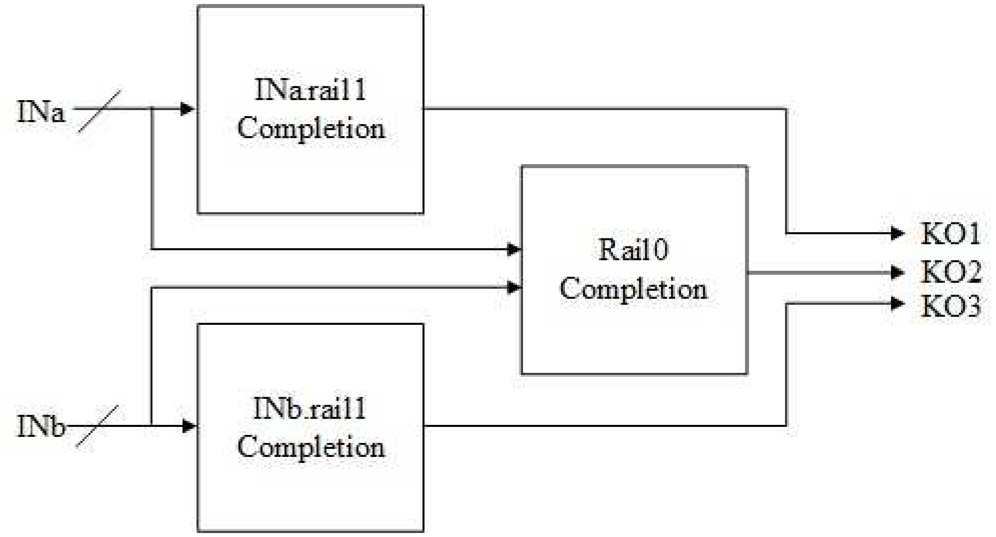

Figure 7: COMP component diagram.

The first and second *KO* signals, *KO1* and *KO3*, are created using the data from the top and bottom TH22 components, respectively. These two completion blocks, INa.rail^1 Completion and INb.rail^1 Completion, use the completion component and TH12b gates to output the correct *KO* value. This structure ensures that all of the individual dual-rail logic data have the value NULL or all have the value DATA before *KO1* or *KO3* can change.

For each dual-rail logic signal input into the COMP component, there is a TH12b gate. The outputs from the TH12b gates are the individual KO signals for each dual-rail logic data. In order to find the overall *KO* signal for the INa.Rail^1 component or INb.Rail^1 component, all of the individual *KO* signals have to be compared by using a completion component, which receives the TH12b gates' outputs. The completion blocks consist of TH44, TH33, and TH22 gates. These gates are used to AND together the output signals coming from the TH12b gates. By using this component, the output of the INa.Rail^1 and INb.Rail^1 components cannot change from a value of 0 to 1 until all individual *KO* signals have a value of 1. Likewise, the output of the INa.Rail^1 and INb.Rail^1 components cannot change from a value of 1 to 0 until all individual *KO* signals have a value of 0.

By checking this requirement, the KO signals coming from INa.Rail^1 and INb.Rail^1 components represent each dual-rail logic data output from the top and bottom TH22 components. *KO1* and *KO3* now correctly reflect that a NULL or DATA wave is complete and sends the correct request to the previous stage for the next DATA or NULL wave.

The last completion component, Rail^0, creates a *KO* signal using both sets of dual-rail data coming from the top TH22 set and the bottom TH22 set. The data for each set is sent through TH12 gates where there is a TH12 gate per each dual-rail logic data. As in the INa.Rail^1 and INb.Rail^1 components, the outputs of every TH12 gate are sent to the completion blocks. This signal is then inverted to create the correct *KO* signal, *rfd* or *rfn*.

Together, these three components determine that the data coming out of the bottom and top TH22 sets are both correct and complete, whether they are complete NULL waves or complete DATA waves, and request for the next DATA or NULL wave.

3.3 TH44n Registers

For a standard NCL register, two TH22n gates are used along with a TH12b to create a single register for each dual-rail logic data. This structure allows for the use of one *KI* signal. Since this architecture uses three *KI* signals instead of one, a different register component is required.

In order to accommodate for the three KI signals received from the succeeding register stage, two TH44n gates are used in each register. This enables each register to check each of the three KI signals before changing its output to 1, if all four inputs are 1, or 0, if all four inputs become 0, for both output^0 and output^1. When the output of each TH44n gate is zero, neither output can change to a value of one until all three KI signals have a value of one. Likewise, if all three KI signals do not have a value of zero, the values of both rails of data cannot change to zero.

TH44n gates are used instead of TH44 gates so that the output^0 and output^1 of each register is equal to zero after a reset occurs. This enables each register's output to be reset, which transitions the whole circuit to a known state. Resetting to null is an important requirement that is discussed further in Section 4.1.

The TH12b gate remains in the register, just as it is used in the standard NCL register. Its inputs depend only on output^0 and output^1 of each register. Since the TH44n gates serve the same purpose as the TH22n gates in the standard NCL register, the TH12b gate is unaffected by the changes made for SEU mitigation.

This design prevents the corruption of a DATA or NULL wave passing through the combinational logic in each stage when one or two SEU affect the *KO* signals. Even though there is a possibility of two of the *KO* signals having incorrect values due to two SEUs, the third *KO* signal prevents an incorrect DATA or NULL wave from being requested before the previous wave is finished.

The final two components, SEL protection and DATA redundancy, enable the architecture to detect an SEL and restore the circuit to normal operation without data loss.

3.4 SEL Protection Component

The SEL protection component is used to detect an SEL and restore the circuit to normal operation. Each stage of the circuit has one SEL protection component to prevent one stage's SEL mitigation process from interrupting or disturbing another stage. Figure 8 shows the major internal components in the SEL protection component.

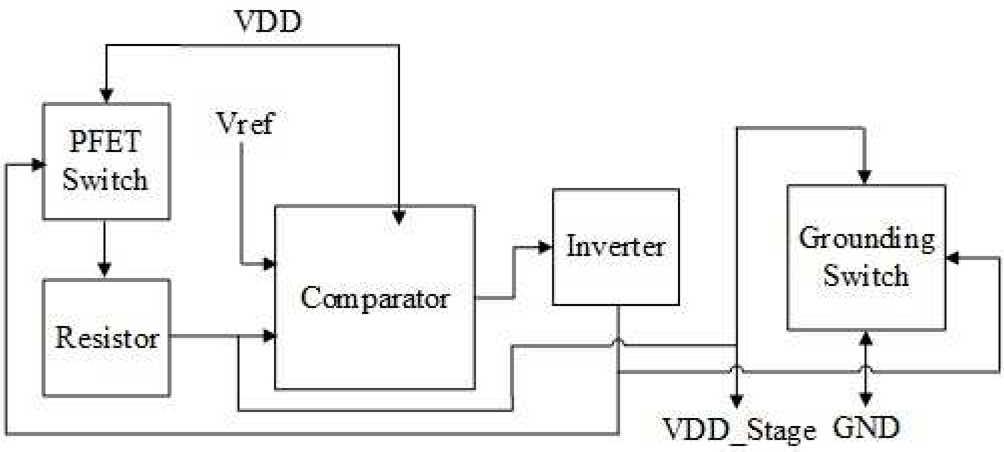

Figure 8: SEL Protection component.

VDD is a separate VDD for each stage, and it is used as the input to the PFET Switch. When an SEL is not detected, the PFET switch is on and *VDD* is connected to the input of the resistor. The purpose of the resistor is to vary the voltage of *VDD* based on the amount of current the stage is using. When the stage uses more current, there is a higher voltage drop across the resistor. The value of the *VDD_Stage* at the output of the resistor is then compared to *Vref*. *Vref* is calculated through circuit simulation to be the voltage of *VDD* when the maximum amount of current is being used by the stage. By using this value, it is known that *VDD_stage* will only be a lower voltage than *Vref* if the stage is using an abnormally high amount of current, which causes a larger voltage drop across the resistor.

Vref is calculated by simulating the current drawn by each stage, since each stage will draw different amounts of current depending on the combinational logic blocks and number of registers used. A small buffer to the *Vref*s calculated for each stage to allow for error and changes in operating conditions.

If the two inputs, *Vref* and *VDD_stage*, are compared and *VDD_stage* is less than *Vref*, it is known that an unusually high amount of current is being drawn by the stage, and thus an SEL has a occurred. When this occurs, two of the components are affected, PFET Switch and

Grounding Switch.

When an SEL occurs, the PFET switch is turned off (input value of 1). This cuts off the stage from power. At the same time, the Grounding Switch, which contains an NFET, is turned on (input value of 1). Since power is disconnected from *VDD_stage*, *VDD_stage* will now discharge to ground through the Grounding Switch. Once the stage is grounded, the SEL protection block resumes normal operation, allowing *VDD* to pass through the PFET Switch. The SEL protection component continues to compare the voltage of *VDD_stage* and *Vref*, waiting for the next SEL occurrence, while the stage operates normally.

While the SEL protection component works correctly in preventing damage to the circuit and restoring normal operation when an SEL occurs, there is still a possibility of DATA loss if an SEL occurs in a stage holding DATA.

4. Multi-Bit SEU Mitigation with Data-Retaining SEL Protection Architecture

The purpose of this architecture is to modify the Multi-Bit SEU Mitigation with SEL protection architecture by adding data-retention to the SEL protection component, denoted as DATA redundancy. This architecture is shown in Figure 9.

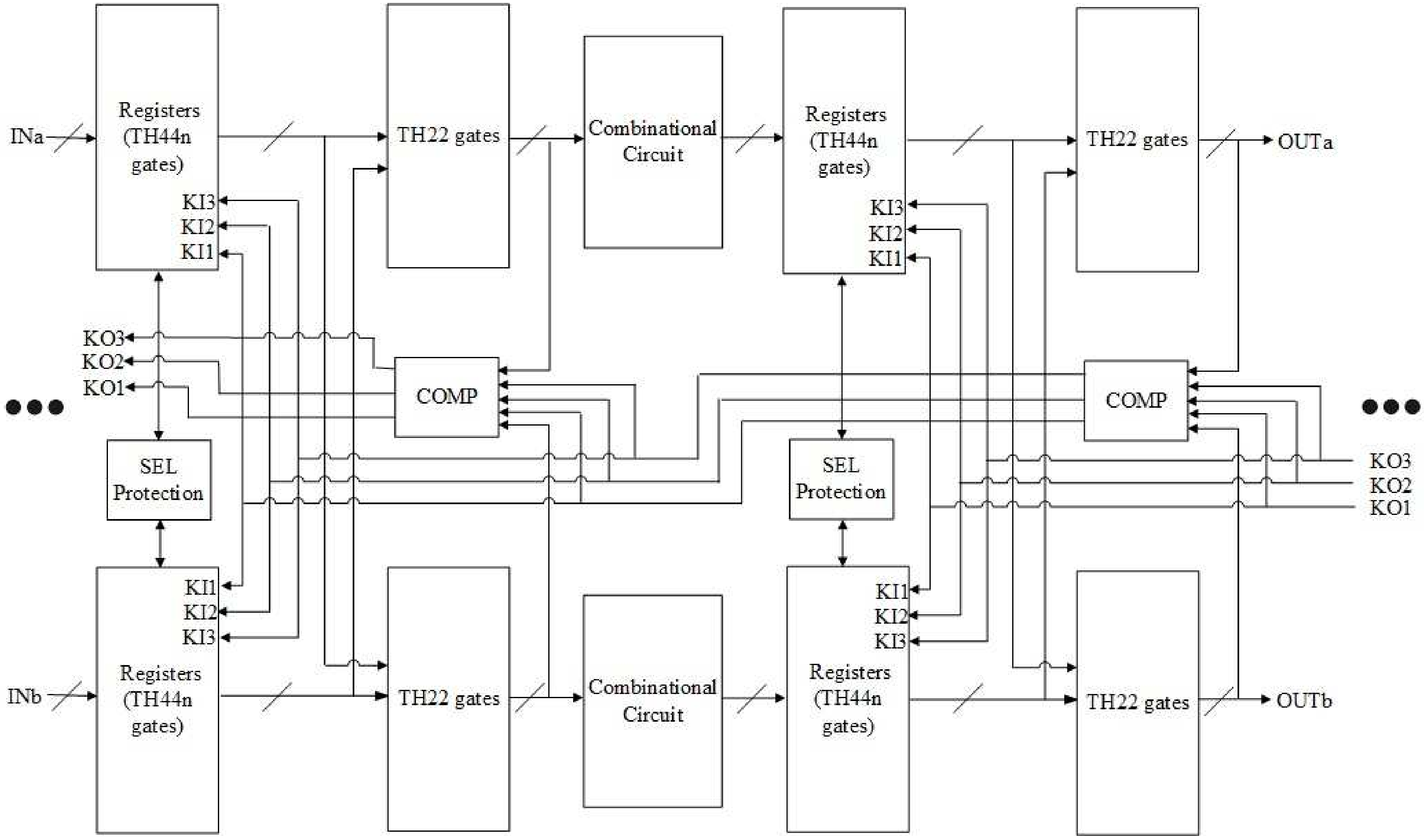

Figure 9: Multi-Bit SEU Mitigation with Data-Retaining SEL Protection Architecture.

4.1 DATA Redundancy

The DATA Redundancy component prevents the circuit from losing DATA when an SEL is detected and recovered from by the SEL protection component. It works by allowing each DATA wave to be immediately followed by another DATA wave containing the same DATA, meaning these two waves are not separated by a NULL wave. Since the DATA is the same in two consecutive waves, if one of the two stages is reset when the circuit is being restored from an SEL, data loss will not occur because another copy of the data exists in either the preceding or succeeding wave. The placement of the DATA Redundancy component is shown in Figure 10.

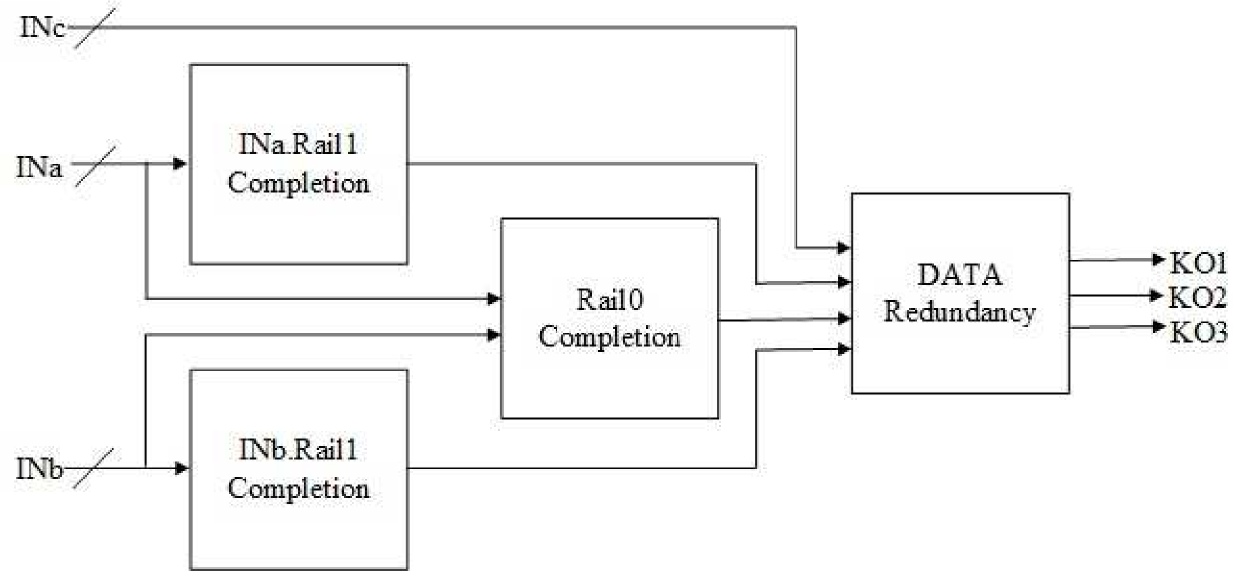

Figure 10: DATA Redundancy diagram.

As seen in Figure 9, the DATA Redundancy component is added to the COMP Component. It takes as input the *KO* signals from the three completion blocks (INa, Inb, and Rail0) as well as *INc*. *INc* contains the three *KO* signals from the succeeding stage. This structure causes a stage's *KO* signals to be impacted by the succeeding two stages.

Inside each DATA redundancy component are three TH22 gates. Because of this, there are six individual inputs (two inputs per TH22 gate). These inputs come from the *KO* signals of the current and next stage (three *KO* signals per stage). The output of each TH22 gate is one *KO* signal that is sent to the preceding stage.

With this design, a stage's operation (sending a NULL or DATA wave to the next stage) is dependent on the following two stages. The two succeeding stages have to both request for DATA or both request for NULL. When this situation occurs, the first of the three stages sends its DATA or NULL wave as requested. This situation occurs for each set of three stages in a circuit.

One requirement for this design to work is that each register used in every stage can be reset to NULL, as opposed to reset to DATA. This is why TH44n gates are used instead of TH44 gates or TH44d gates. Take the following situation to understand this requirement: there is a DATA wave in stage four succeeding another DATA wave (stage three) followed by two NULL waves. Since stage four and stage three both contain data waves, it is required that there are two NULL waves in front in stages five and six, as well. This example is illustrated below in Figure 11.

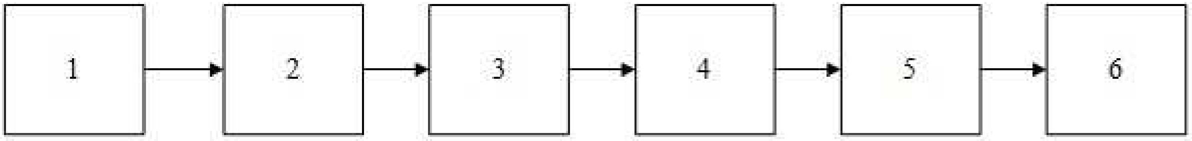

Figure 11: Illustration of DATA Redundancy.

Figure 11 above shows six stages. Each stage can either be processing a NULL wave or DATA wave, but only in groups of two. Each NULL wave is immediately followed by a NULL wave, and each DATA wave is immediately followed by a DATA wave with the same DATA. Figure 12 is a table showing what happens during the reset of a specific stage when an SEL occurs.

State	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
1	NULL	NULL	DATA	DATA	NULL	NULL
2	NULL	NULL	DATA	NULL	NULL	NULL
3	NULL	NULL	DATA	DATA	NULL	NULL

Figure 12: Effects of an SEL.

The first state shows the stages in a circuit when SEL are not occurring. Each set of two NULL cycles is followed by a set of two DATA cycles and each set of DATA waves is followed by a set of two NULL waves.

Now assume an SEL occurs in stage 4 during state 1. This is detected by the SEL protection component and causes the stage to be reset. Since the stage is reset before the DATA wave is finished and received by the succeeding stage, the DATA is lost and the current wave is now a NULL wave. Since each DATA wave is followed by a DATA wave of the same value, stage 3 still has the DATA that was lost when stage 4 was reset. Similarly, if stage 3 is reset due to an SEL, the stage loses its DATA and is now processing a NULL wave. Since stage 4 contains the same data wave that stage 3 had before an SEL occurred, the DATA is not lost.

State 2, in Figure 12, shows the resulting waves for each stage after an SEL occurs in stage 4. Because of the DATA Redundancy component, the circuit can still continue normally. Since stages 4 and 5 have completed their NULL waves, the three *KO* signals from stage 4 request DATA from stage 3. The same situation occurs for stages 5 and 6. Since both of their NULL waves have completed, stage 5 requests data from stage 4. Stage 4 currently has a NULL wave, so stage 5 must wait until stage 4's next DATA wave completes.

State 3 shows the current waves after stage 4 requests DATA from stage 3. Since stage 3 has the same data wave as the original data wave in stage 4 during state 1, stage 4 now has the same data wave that it lost during the SEL occurring in state 1. The circuit has now recovered from the SEL occurrence and can continue in normal operation.

One of the important components that allows this design to work is using TH44n gates in all of the registers for each stage, instead of TH44 gates or TH44d gates. Since the TH44n gates reset to NULL, any register that uses them resets to NULL. When stage 4 resets during state 1, stage 4 has to be NULL in order to recover from the process. This is important because both stage 4 and stage 5 have to be requesting DATA before the DATA in stage 3 can move forward in the pipeline.

5. Scenarios and Simulations

Both architectures are tested at the transistor level using a five-by-five multiplier in the IBM 130-nm 8RF process. In order to test both architectures, injection points are set up such that while unaffected, both circuits behave normally. When being tested, these injection points can be changed to incorrect values, as would be caused by an SEU, or cause a reset, as would be caused by an SEL.

The first simulation shows what the results should be under normal operating conditions. Since the two architectures differ only in how they respond to radiation, the results will be the same regardless of which architecture is used.

This test begins with resetting the multiplier to a known state. Since TH44n gates are used in all registers, the output of each register set becomes NULL after a circuit-wide reset. The NULL wave output by each register set is sent to the TH22 gates component where the values of the top and bottom register outputs are compared. Since the outputs match, as they are both NULL waves, the data is sent through to the COMP component. At this point, there are still no errors, and the data from outputs of the top and bottom TH22 gates is correct. The COMP component detects complete NULL waves from the TH22 gates and, as a result, requests a DATA wave to each previous stage.

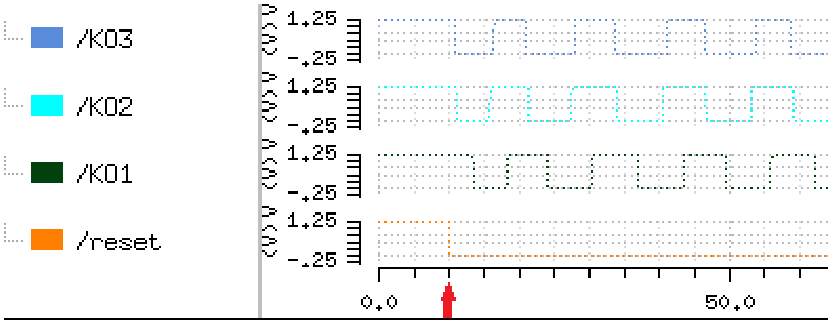

Figure 13: *Reset* and *KO* signal waveforms.

Figure 13 shows the three *KO* signals' output from the first stage of the circuit, as well as the main *reset* signal. It is important to note that this signal is not the same as the reset signal used in the SEL protection component. The SEL protection component can only reset the individual stage where it checks for SEL occurrences. These *KO* signals in Figure 13 tell the outside world when the circuit is ready for a new DATA wave or NULL wave, and as soon as the circuit is reset (the end of the reset is marked by the red arrow), all three *KO* signals request a new DATA wave from the controller of the multiplier.

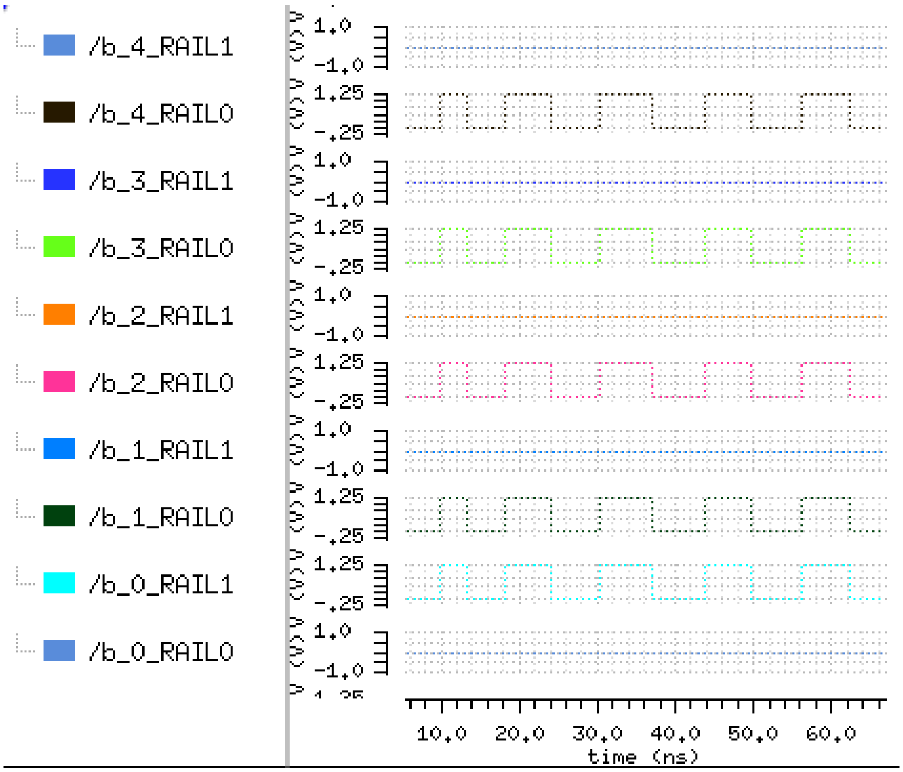

Figure 14: Waveform of input signal *b*.

Figure 14 shows the values of the second input signal, *b*. This input is one of the two five-bit dual-rail logic operands. Signal *b* alternates between DATA and NULL, as requested by the three *KO* signals in Figure 13. To simplify the demonstration, *b* holds a value of '1' during each DATA wave, so the results of the simulation will be equal to the value of the other operand, signal *a*.

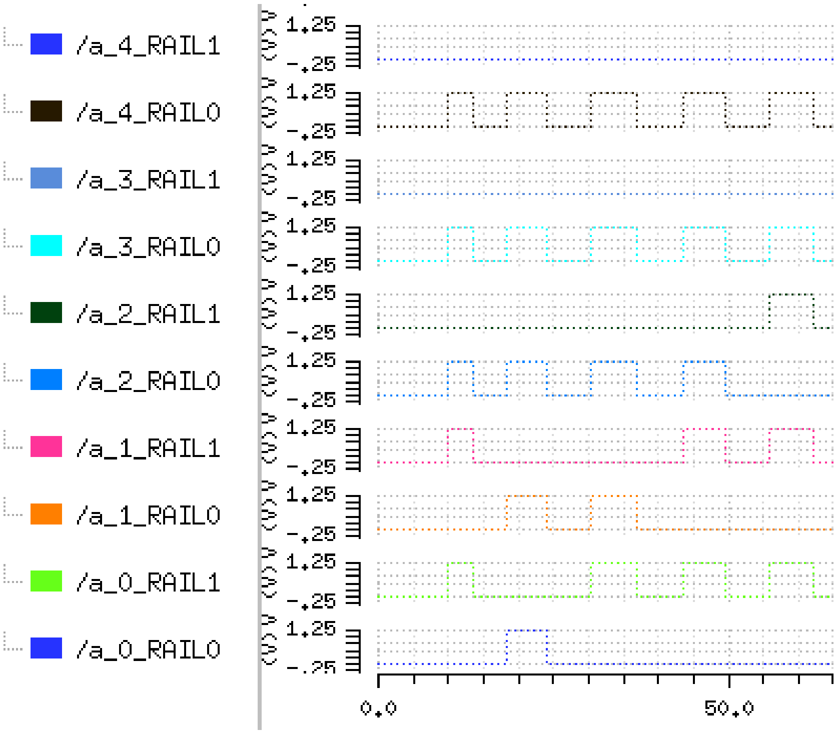

Figure 15: Waveform of input signal *a*.

The second five-bit dual-rail logic operand of the multiplier, *a*, is shown in Figure 15. As seen in the waveform for *b*, *a* alternates between DATA and NULL values as requested by the first stage of the multiplier. This process continues as data propagates through each stage of the multiplier, and is eventually output as a result. The input values of *a* vary widely to indicate clearly in the results that each DATA wave completes correctly.

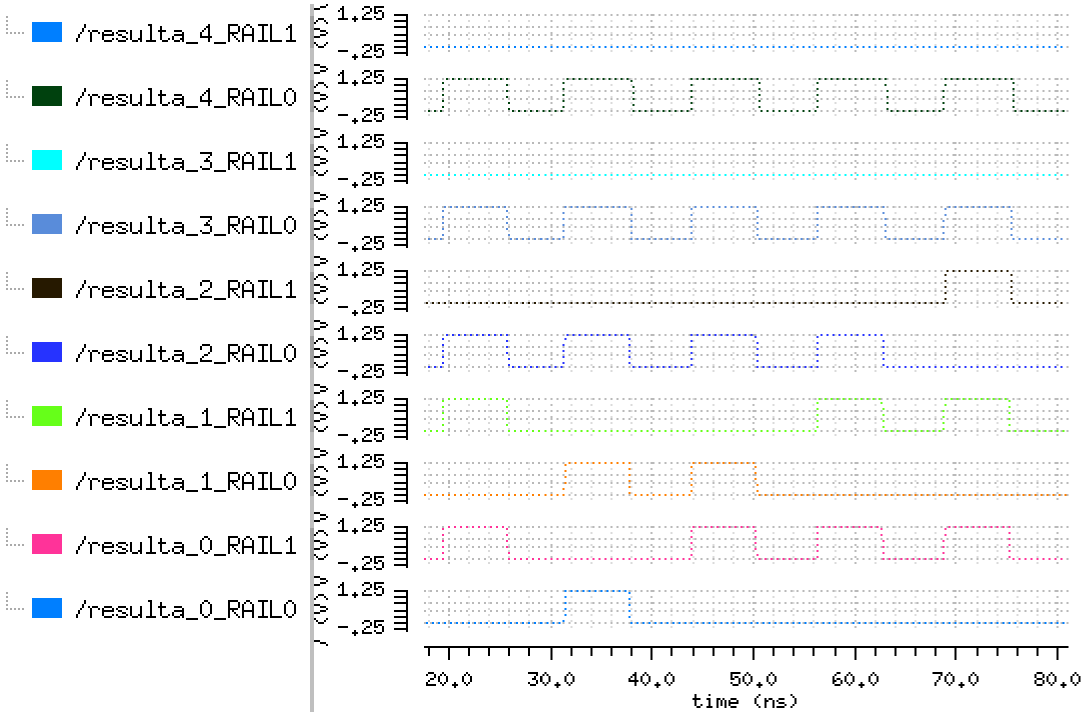

Figure 16: Waveform of output signal, *result*.

Result, as shown in Figure 16, is a ten-bit dual-rail logic signal. Since the high-order five significant bits of the signal remain DATA0 for each DATA wave, only the low-order five bits are shown. To complete this demonstration, Figure 17 is used to show the integer values of each input and output signal at the times shown in each waveform for *a*, *b*, and *resulta*.

Input Time (ns)	*a*	*b*	*resulta*	*Resulta* time (ns)
10	3	1	3	19
18	0	1	0	31
30	1	1	1	44
44	3	1	3	56
56	7	1	7	68

Figure 17: Table of the inputs, *a* and *b*, and the result of operation, *resulta*.

The values in the table are correct, and show the correct operation of the multiplier test circuit. Each waveform can be referenced from the values in the table to confirm the operations and the correct outputs. The next scenarios cover the mitigation of two-bit SEUs and an SEL while preventing data loss.

5.1 Two-bit SEU Scenario

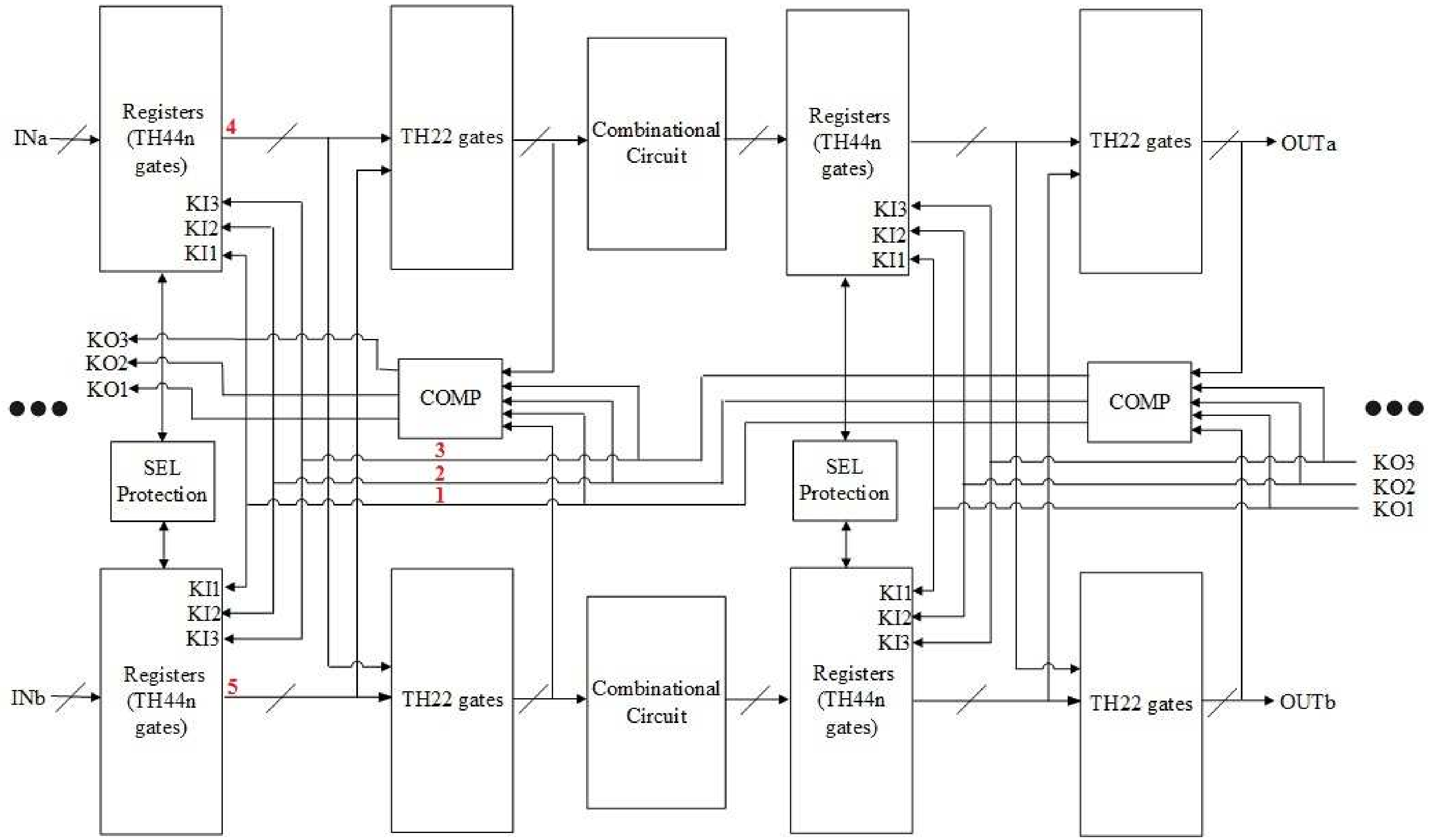

Figure 18: Multi-Bit SEU Mitigation with Data-Retaining SEL Protection Architecture.

Signal (Marker)	Initial State	Second State	Final State
KI1 (1)	rfd	rfn	rfd
KI2 (2)	rfd	rfn	rfd
KI3 (3)	rfd	rfd	rfd
Top register set output (4)	DATA	DATA	DATA
Bottom register set output (5)	DATA	DATA	DATA

Figure 19: States of signals during two-bit SEU scenario.

Figure 19 shows the states of various signals of the architecture (Figure 18) during the progression and mitigation of two SEUs on the *KO* signals, which are sent from the right-most stage (stage two) to the left-most stage (stage one). A marker is shown in parenthesis next to each signal in the table. These markers correspond to the red numbers shown in Figure 18. In the table, the initial state shows normal operation in that all *KO* signals from stage two, which are the *KI* signals for stage one, are requesting data (rfd). At this point, data is propagating from the output of the register sets in stage one through the TH22 sets and onto the combinational logic.

The second state in Figure 19 shows what immediately happens when two SEUs occur that affect *KI1* and *KI2*. Choosing these two signals is arbitrary. The effects would be the same if this demonstration used any two of the three *KI* signals of any given stage.

As seen in Figure 19 during the second state, *KI1* and *KI2* are affected by the SEUs and temporarily have values of zero (rfn). During this time, *KI3* retains the correct value of one (rfd). If *KI1* and *KI2* together controlled stage one, stage one would be temporarily allowed to send a NULL wave.

Since there is an additional *KI* signal, *KI3*, in the new architectures, *KI1* and *KI2* are unable to corrupt the data and possibly cause a lock-up. In the final state of Figure 19, *KI1* and

KI2 go back to their original and correct values of 1 once the two SEUs subside, and the circuit is able to continue in normal operation.

For a moment, assume there is not a third *KI* signal, as in the original architecture. With the two *KI* signals (*KI1* and *KI2*) effected by two SEUs and controlling the first stage, the NULL wave would corrupt the DATA that is currently propagating from the output of stage one, which is the DATA propagating through the TH22 components and combinational logic blocks, at this point. A lock-up would occur after *KI1* and *KI2* temporarily transitioned to rfn and back to their correct values, rfd. During the time that *KI1* and *KI2* are incorrect, a partial-NULL wave is propagated throughout the TH22 sets and combinational logic eventually reaching the inputs of the register sets for stage two. At this point, part of the correct DATA is erased, due to the partial-NULL wave, leaving some registers with inputs of DATA and others with inputs of NULL. The problem now is that stage two is still waiting for DATA, but the DATA is corrupted. Until it receives a complete DATA wave, it cannot send a request for NULL. It will continue to wait for data, but the data will never come resulting in a lock-up.

5.2 Two-bit SEU Simulation

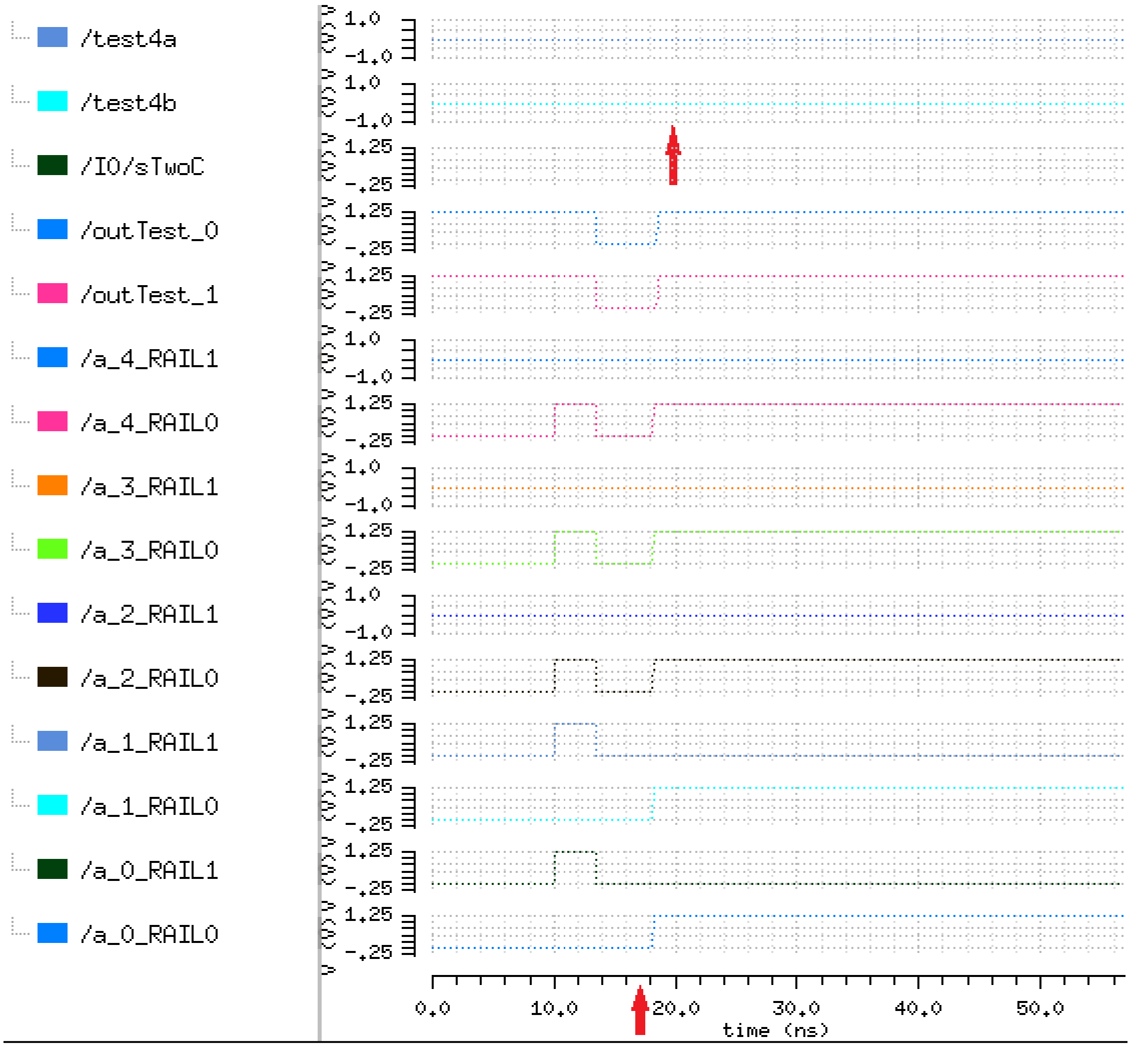

Figure 20: Waveforms of SEU-affected *KO* signals and input signal *a*.

This simulation tests the architecture's ability to mitigate two SEUs that affect the KO signals between two stages.

The problem with one stage having two *KO* signals is the possibility of those two *KO* signals being affected by two SEUs at the same time. If two SEUs changed the value of both *KO* signals, the inputs coming from the previous stage could propagate through the next stage before they should. This situation could result in incorrect data being processed or even cause the whole circuit to lock-up.

To mitigate this issue, three *KO* signals are used between each stage. Figure 20 shows five different signals. Signals *sTwoC*, *outTest_0*, and *outTest_1* are the *KO* signals coming out of stage two of the circuit. The signals *test4a*, *test4b*, and s*TwoC* are the actual KI signals for stage one. This means that *test4a* and *test4b* the injection signals, as explained at the beginning of this section.

During the circuit's normal operation, *test4a* and *test4b* have the same value as *outTest_0* and *outTest_1*, respectively. Since they are injection inputs, *test4a* and *test4b* can also have values other than that of *outTest_0* and *outTest1*. The structure of the *test4a* and *test4b* inputs allow the simulation of two SEUs on the *KO* signals between stage two and stage one of the test circuit.

As seen in Figure 20, the circuit stops its operation when *test4a* and *test4b*'s values do not match the value of *sTwoC* (top arrow of Figure 20). This means that the *KI* signals for stage one have different values (two *KI*s have a value of 0 and the last *KI* has a value of 1). Even though the signal *sTwoC* is requesting DATA from stage one, stage one will not send DATA until both *test4a* and *test4b* have a value of 1, matching the request of *sTwoC*.

Figure 20 elaborates on this situation by showing the circuit's operation is stopped by the two SEUs, which are affecting *test4a* and *test4b*. Until these two signals return to normal operation, where they are equal to *sTwoC* given by stage two, the overall operation of the circuit is halted.

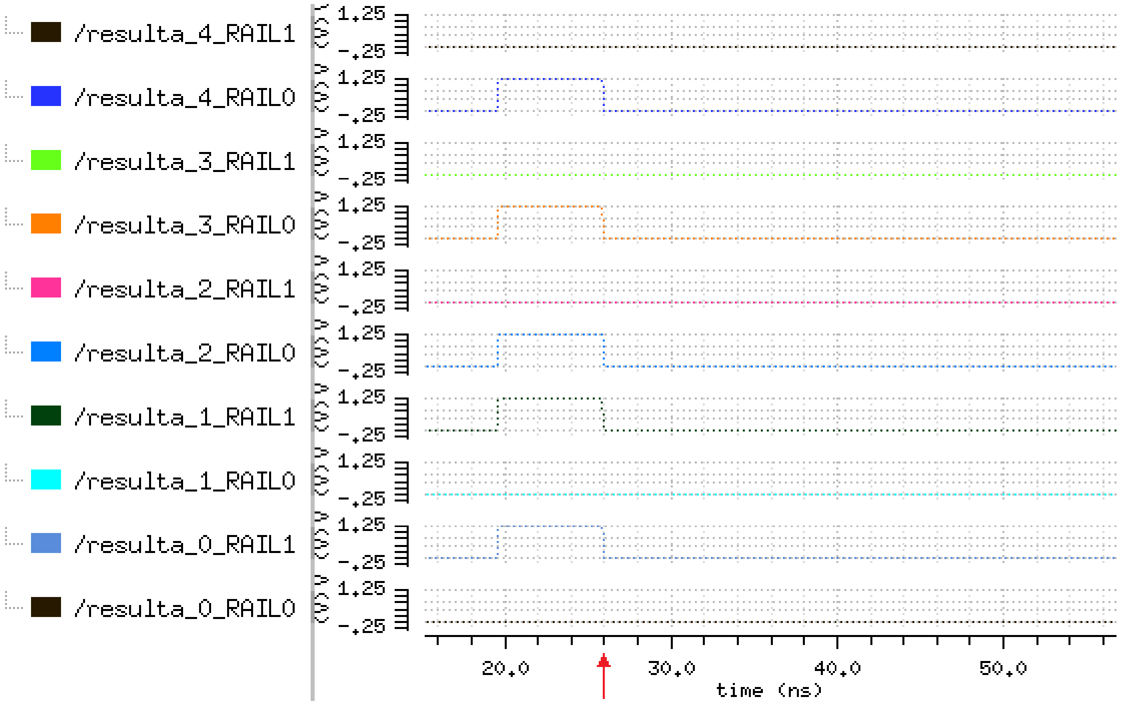

Figure 21: Result signals.

In Figure 20, the circuit only propagates one set of input DATA, the multiplication of a set of inputs when *INa* is equal to '3' and *INb* is equal to '1' (*Inb* retains a value of '1' for all simulations). Because of this, the *resulta* signal only shows one output, the multiplication of '3' and '1'. Figure 21 shows the output is correct because only *resulta_0* and *resulta_1* have a value of DATA1 (shown by the red arrow), which is the integer value 3 for the *resulta* signal as a whole. The second set of inputs is never propagated through due to the simulated effects of two SEUs on two of the *KO* signals between stage two and stage one.

5.3 SEL with Data Recovery Scenario

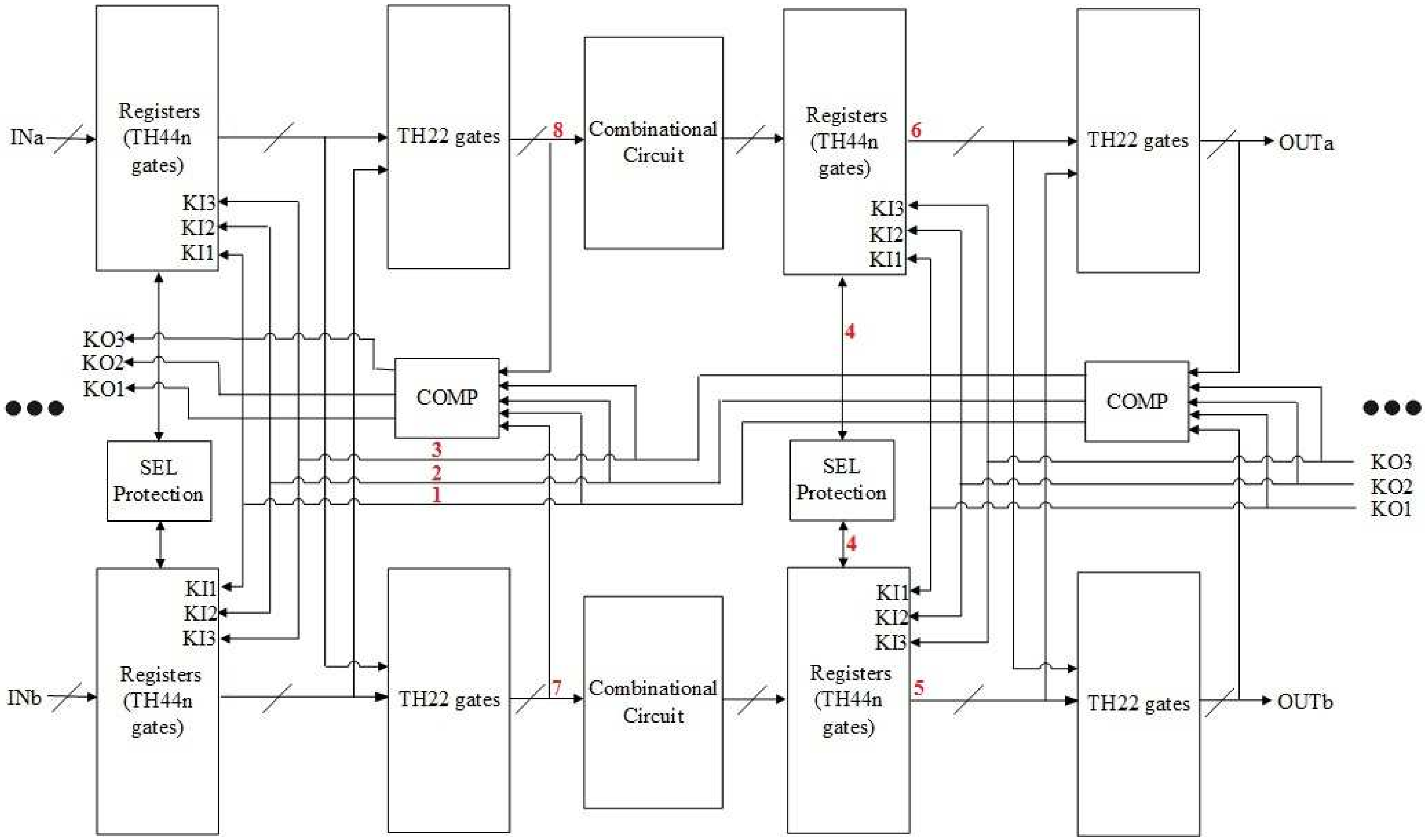

Figure 22: Multi-Bit SEU Mitigation with Data-Retaining SEL Protection Architecture.

Signal (Marker)	Initial State	Second State	Third State	Final State
KO1 (1)	rfd	rfd	rfd	rfn
KO2 (2)	rfd	rfd	rfd	rfn
KO3 (3)	rfd	rfd	rfd	rfn
SELreset (4)	0	1	0	0
Bottom register set output (5)	NULL	NULL	NULL	DATA
Top register set output (6)	NULL	NULL	NULL	DATA
Stage data (7)	DATA	NULL	DATA	DATA
Stage data (8)	DATA	NULL	DATA	DATA

Figure 23: Shows the progression of a latch-up recovery.

Figure 23 shows the values of signals and their markers (corresponding to the red numbers in Figure 22) as an SEL occurs in the circuit and later resumes normal operation.

During this scenario, DATA is propagating through the stage (markers 7 and 8) as the SEL occurs, which is the initial state. The output of the registers (markers 5 and 6) are both NULL and, as a result, are requesting DATA from the previous stage using *KO1*, *KO2*, and *KO3*.

The second state shows the signals as the stage is reset (marker 4) consequentially erasing the DATA in the stage (markers 7 and 8). The DATA does not reach the output of the bottom and top (markers 5 and 6) register sets, so the values of the bottom and top register sets remain NULL. Since the outputs are still NULL, the stage still requests DATA from the previous stage.

In the third state, the *SELreset* is disabled and the circuit can resume normal operation. The DATA from the preceding stage is output and set as the value of *stage data* (markers 7 and 8). Since each DATA wave is immediately followed by a DATA wave of the same value, the erased DATA during the SEL is replaced with the same DATA from another DATA wave.

The final state shows the recovered DATA now progressing to the next stage, meaning the output of the top and bottom (markers 5 and 6) register sets are now the recovered DATA. The *KO* signals now reflect the new DATA wave as they request NULL and the circuit is propagating inputs normally.

5.4 SEL with Data Recovery Simulation

This test shows the effect of resetting stage two of the five-by-five multiplier during an SEL while the circuit is processing data.

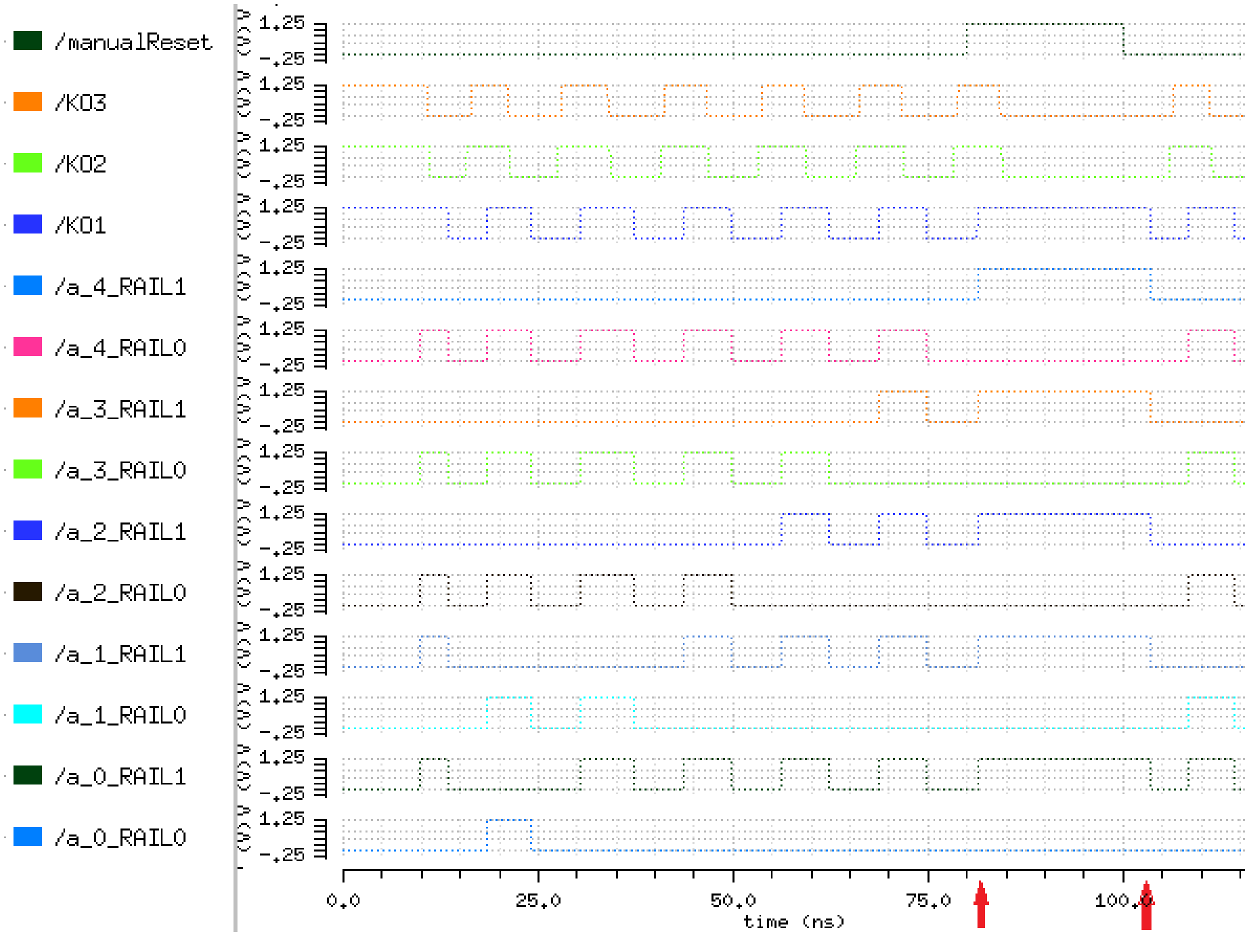

Figure 24: Input *a* and *KO* values.

Figure 24 shows the three *KO* signals coming out of the first stage of the circuit. These three signal tell the controller of the circuit when to send a NULL wave or DATA wave. Shortly after 80 ns (when the reset signal is turned on for stage two marked by the red arrows in the figure), an SEL is detected and all three *KO* signals are halted, meaning the circuit has ceased operation at some stage during operation. It is important to note that even though this stage has stopped operating, data that has previously passed through this stage can still propagate through the rest of the pipeline to the output.

Figure 24 also shows the progression of input values for *a*, which is a five-bit dual-rail logic signal. The other input value for the circuit, *b*, has an integer value of one for the entire duration of this simulation. As can be seen in Figure 24, the value of *a* changes from DATA to NULL. The integer values of during each DATA wave of *a* between each NULL wave include values 0 at 20 ns, 1 at 35 ns, 3 at 45 ns, 7 at 60 ns, 15 at 70 ns, and 31 at 80 ns. The *manualReset* signal shown in the Figure 24 is the reset signal used by the SEL protection component specifically for stage two of the multiplier. This stage is arbitrarily chosen for showing the effects of this architecture. The same process and result could be demonstrated on any stage of the multiplier.

As seen in Figure 24, *manualReset* is enabled at 80 ns. Once this *manualReset* is enabled, the circuit stops taking in new DATA and NULL waves for *a* and *b*, and any DATA or NULL waves before stage three's register set cannot propagate through the rest of the circuit. Because of this, the value of *a* at 80 ns is held at the integer value 31 until the reset is turned off. This is further demonstrated by the results, as seen in Figure 20 below.

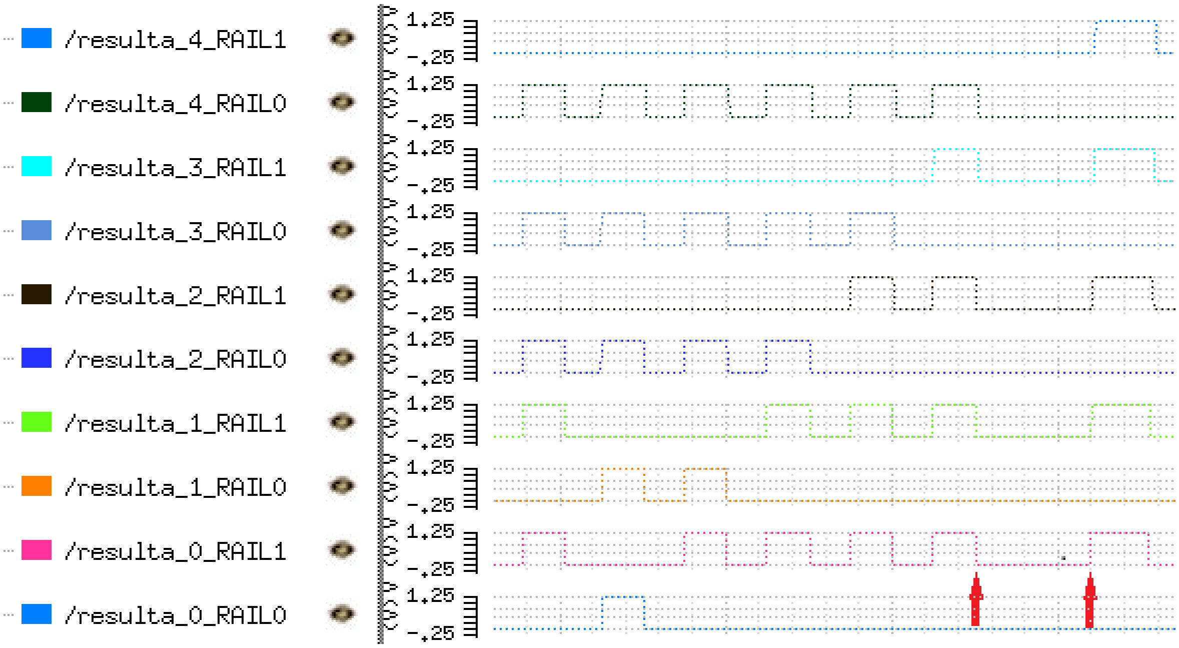

Figure 25: The results of the first five calculations.

In Figure 25, it is shown that the results are unaffected despite resetting an individual stage of the multiplier. The value of *resulta* after the second red arrow in Figure 25 is 31, which is correct since 31 (*a*) is multiplied by 1 (*b*). The process of detecting and recovering from the SEL only delays the outputs without losing any of the DATA.

Since the reset caused by the SEL only affects stage three of the circuit, any DATA being processed by stage four and on will continue to propagate giving a result value after the last stage. This is shown in Figure 25 because each input for *a* (which is multiplied by 1) is seen as a result before the delay in *resulta* is seen.

6. Conclusion and Future Works

Radiation is a huge concern for electronics, especially for space applications, due to its ability to change the state of signals or cause permanent damage to the chip. The multi-bit SEU mitigation with SEL protection architecture modifies the original architecture to create a more robust, SEU-resistant architecture, while including SEL recovery, by adding a third completion block with a corresponding third *KO* signal per stage, using TH44n gates in the registers instead of TH33n gates, and adding an SEL protection component. The multi-bit SEU-mitigation with Data-Retaining SEL protection architecture takes the first architecture a step further to guarantee no data loss when an SEL occurs. Significant weaknesses of the original architecture, such as the latch-up problem with the *KO* signals, are improved upon to further the radiation-hardness of the architectures.

Using an IBM 130-nm process, a test five-by-five multiplier was created to simulate the new architectures. It is shown that with the newly created components, the architecture works correctly in outputting the correct data. Next, a scenario was presented in which the architecture was able to correctly mitigate a two-bit SEU when simulated at the transistor-level. The two-bit SEU only affected the circuit by causing a short delay until the SEU subsided. Finally, a scenario was shown in which the Multi-bit SEU Mitigation and Data-Retaining SEL Protection Architecture was affected by an SEL. The second architecture correctly resolves the effects of the SEL without losing any data. The results of these simulations show the effectiveness of both architectures, which are promising for building space electronics for radiation hardness.

While the techniques used in this thesis to mitigate the effects of radiation work correctly, additional work could be made in improving the defensive abilities of both architectures. If these architectures were tested with radiation, additional weaknesses could be identified. With this information, a stronger architecture could be engineered to further the radiation hardness.

References

[1] R.D. Schrimpf, and D.M. Fleetwood (eds), Radiation Effects and Soft Errors in Integrated Circuits and Electronic Devices. World Scientific Publisher, 2004.

[2] Karl M. Fant, and Scott A. Brantd, “NULL Convention Logic: a complete and consistent logic for asynchronous digital circuit synthesis.” in ASAP 96, Chicago, IL, Aug. 1996.

[3] Scott C. Smith, and Jia Di, Designing Asynchronous Circuits using NULL Convention Logic (NCL). Morgan Claypool Publishers, 2009.

[4] Gerald E. Sobelman and Karl M. Fant, "CMOS Circuit Design of Threshold Gates with Hysteresis," IEEE International Symposium on Circuits and Systems (II), 1998.

[5] J. Di, "A Framework on Mitigating Single Event Upset using Delay-Insensitive Asynchronous Circuit," IEEE Region 5 Technical Conference, Apr. 2007.