CSUN® | COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

Fall 2024

California State University, Northridge

Department of Electrical & Computer Engineering

Final Report: Triple Module Redundancy

I hereby attest that this lab report is entirely my own work. I have not copied either code or text from anyone, nor have I allowed or will I allow anyone to copy my work.

Name (printed) Zachary Martin

Name (signed) Date 12/14/2024

ECE 526

Written By: Zach Martin

December 14, 2024

I. INTRODUCTION

A. *Background*

In a theoretical environment without confounding variables, a sufficiently advanced circuit can perform complex and powerful tasks perfectly, realizing virtually any imagined outcome. Unfortunately, circuits don't exist in theoretical environments, and there are many confounding variables that can affect circuit behavior. One such variable is ionizing radiation. Cosmic rays and nuclear materials produce ionizing particles that carry enough energy to strip electrons from atoms, inducing erratic behavior in transistors upon collision [1]. When an ionizing particle causes a change of state in a circuit or component, it is known as a single event upset (SEU).

Fault-tolerance is the characteristic of a system to resist faults, or errors, such as SEUs [2]. There are two primary methods of fault-tolerant design: error correction and N-modular redundancy. Error correction can be implemented as software or hardware and serves to detect and resolve errors in-place, often through the use of parity bits [3]. N-modular redundancy involves duplicating components that all perform the same task and taking a majority vote of the output, which is then passed to succeeding stages of the system [4]. The number of redundant components is often odd in order to avoid a tied vote. Triple modular redundancy (TMR) is a case of N-modular redundancy that uses three identical components and a three-input voter to resolve single points of failure.

B. *Design*

A TMR system consists of the triplicate modules and at least one voter. It is possible to implement TMR with three voters, themselves fault-tolerant to an extent, but often provides minimal accuracy improvement in most cases and will not be considered in this report. A high level block diagram of TMR can be seen in figure 1. The voter is the only aspect of TMR that requires its own design, as the triplicate modules are necessarily identical. Two implementations of a majority voter, one using AND and OR gates, and the other using only NAND gates, can be seen in figure 2.

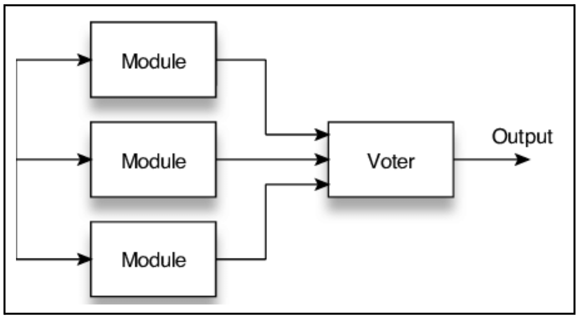


Fig. 1. TMR block diagram [5].

TABLE I

VOTER TRUTH TABLE

Module 1 Output	Module 2 Output	Module 3 Output	Voter Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

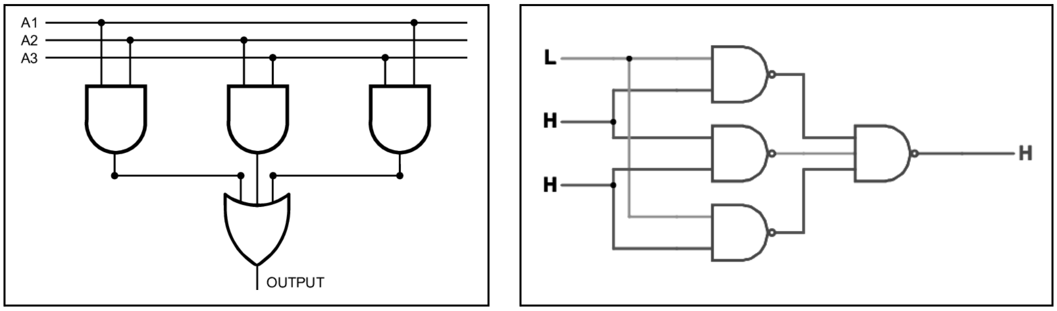


Fig. 2. Three-input voter using (a) AND and OR gates. (b) NAND gates. [6] [7]

The error rate of TMR with one voter (considering a constant SEU probability over time) can be modeled as the probability of all three modules failing plus the probability of any two modules failing [8]:

$$E_{TMR} = 3P_{SEU}^{2} - 2P_{SEU}^{3}$$

This formula will be used to compare hypothetical failure rates against simulation data rates in the conclusion section of this report.

C. *Applications*

TMR has existed since long before the invention of the electronic circuit. Mariners traveling through the oceans have known since the invention of the chronometer that one should “never go to sea with two chronometers; take one or three.” [9] If two chronometers were to contradict, it would be impossible to evaluate the current time, as there is no way to know which timepiece is correct. The HMS Beagle carried as many as 22 chronometers during Charles Darwin’s expeditions from 1831 to 1836, forming 22-module redundancy [10]. A similar adage exists as Segal's law: "A man with a watch knows what time it is. A man with two watches is never sure." [11]

In computer systems, TMR generally is most applicable when a module is expected to experience a high volume of SEUs and when its function is critically important. For this reason, nuclear power plants have implemented TMR in their emergency diesel generators [12] and Instrumentation and Control (I&C) systems, such as that of the Tricon programmable logic controllers (PLC) [13].

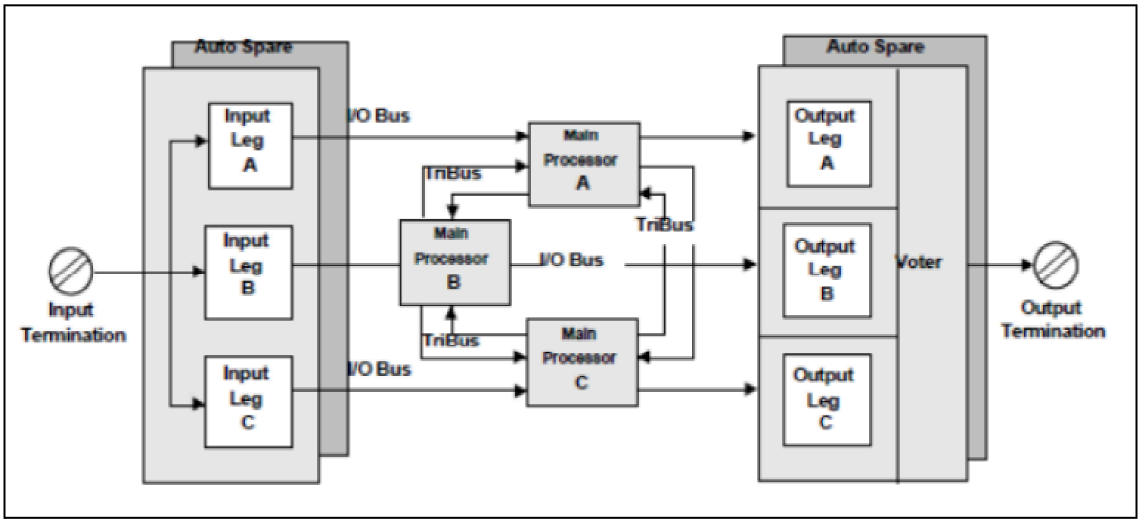


Fig. 3. TMR architecture in the Tricon PLC I&C system. [13]

TMR is also used commonly in space exploration vehicles and satellite design. On December 21, 1968, the first human spaceflight reached the Moon. This flight was the first crewed launch aboard the Saturn V rocket, which carried in its ring-shaped instrument unit the Launch Vehicle Digital Computer (LVDC) [14], [15]. The LVDC was a 72 pound computer that facilitated the autopilot and trans-lunar injection burn required to travel nearly a quarter million miles through space, orbit ten times around the Moon, and return the crew to Earth. IBM created the computer and implemented TMR to ensure the safety of the humans aboard Apollo 8. All seven stages of the onboard logic system were included in triplicate, with an average of 10 voters per stage [16]. Additionally, 9 gyroscopic sensors were used in triplicate to measure the pitch, roll, and yaw of the rocket [17].

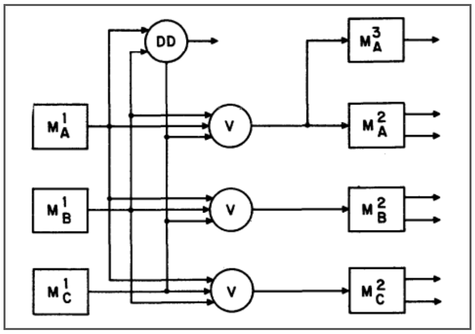


Fig. 4. TMR design used in the LVDC [16].

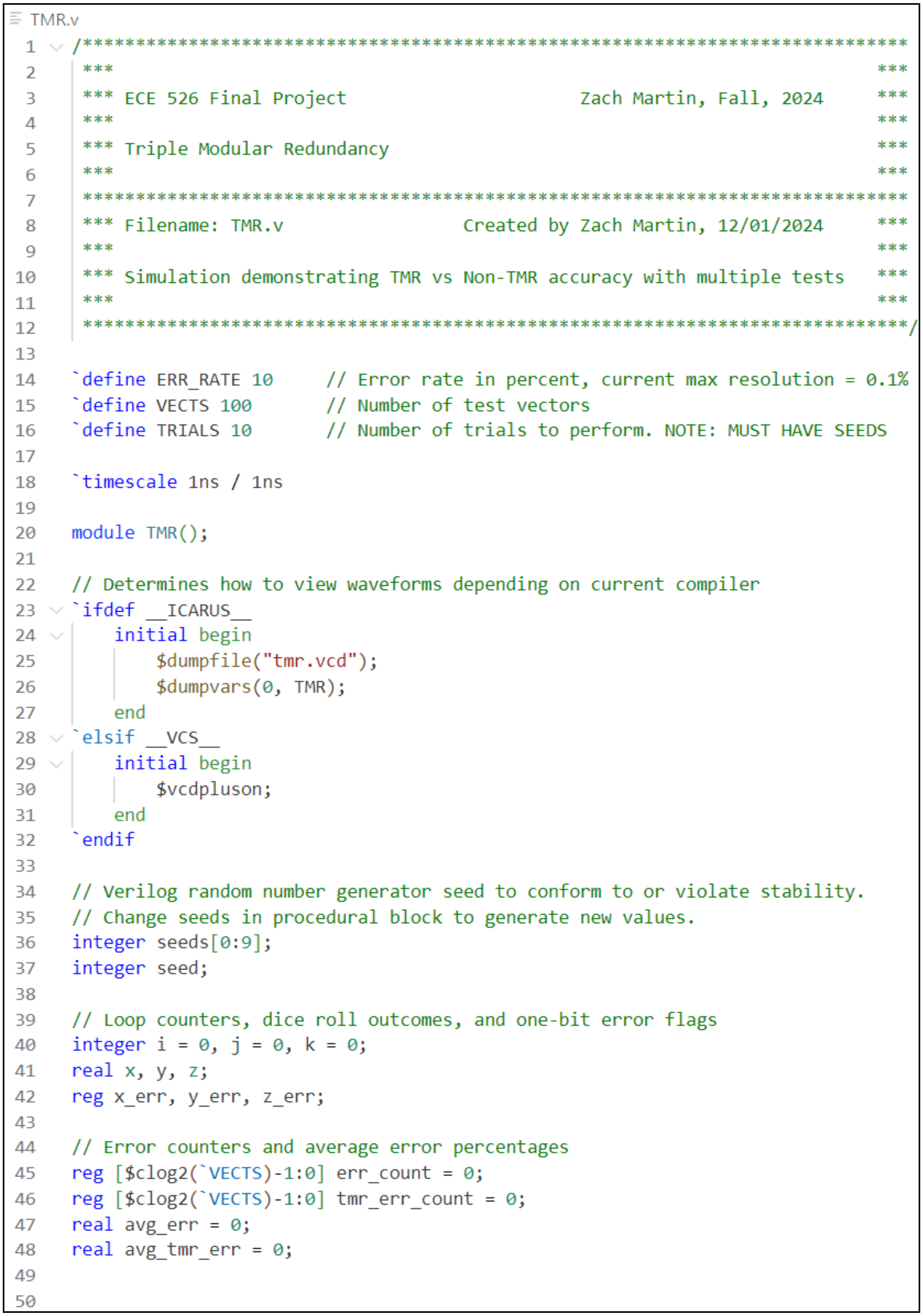
D. *Limitations*

There is no implementation of TMR that escapes its greatest penalty: running a system in triplicate requires three systems. Three systems occupy three times the space, use three times the power, and cost three times as much as a single system at best. Additionally, due to the fact that multiple-fault states will generate an error despite the existence of a single unaffected system in the TMR configuration, the accuracy rate decreases below the accuracy rate of a single system without TMR if the probability of SEUs increases beyond 50%. That is to say, TMR produces the greatest accuracy improvements as the SEU probability approaches zero, and offers diminishing returns as it approaches 50% [17]. Other fault-tolerance techniques offer a more uniform improvement across all SEU probabilities, but are often slower and more complicated to design [18].

II. PROCEDURE

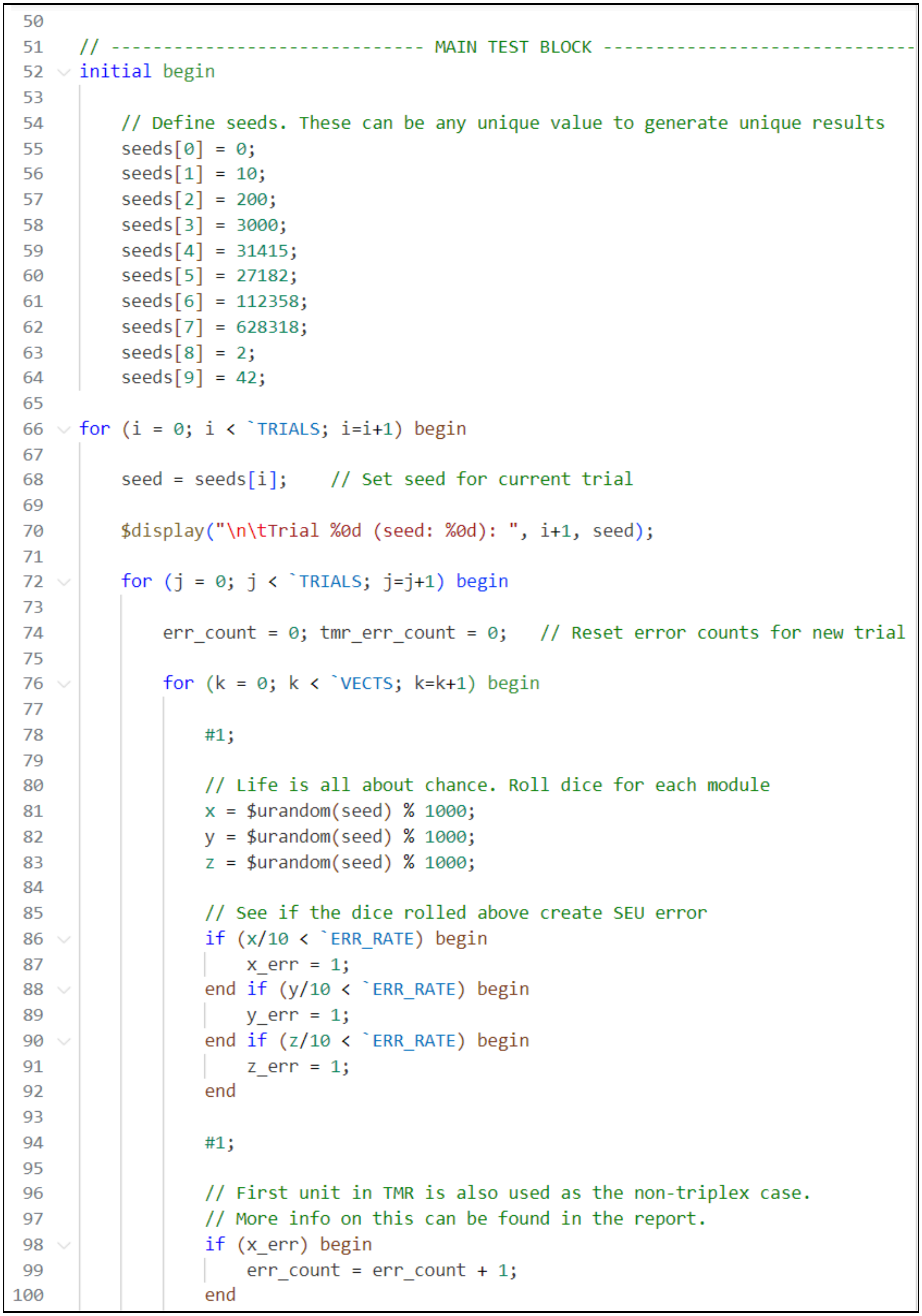
A Verilog module was written to simulate ionizing particles inducing SEU faults in three identical circuits. Preprocessor directives were used to assign the SEU probability (ERR_RATE), number of vectors per trial (VECTS), and number of trials per batch (TRIALS). The program generated a pseudorandom value between 0 and 1000 for each of the three circuits (x, y, z) with the $urandom system function using a unique seed for each batch. If a circuit's value exceeded the SEU probability threshold (and had therefore experienced an SEU), the circuit was marked as having erroneous output by setting a flag register (x_err, y_err, z_err) to 1. In order to account for inherent randomness in the experiment, the first of the three circuits (x) was used as both the reference and in the triplex configuration. Configuring the test to re-use the first circuit ensures that the results reflect exposure to identical ionizing particles in a given period of time, both with and without TMR, allowing for direct comparison.

If the first circuit's error flag was high, its error counter (err_count) incremented by 1. If the sum of the three error flags was greater than 1 (representing a multiple-fault state), the TMR system's error counter (tmr_err_count) incremented by 1. All flags were reset and the process repeated until all vectors had been simulated for all trials, after which the seed was changed and a new batch of trials began. The average error rate without TMR (avg_err) and the average error rate with TMR (avg_tmr_err) was calculated across all trials in a batch and displayed to the command line, along with the number of errors in the first trial and the accuracy improvement factor when using TMR.



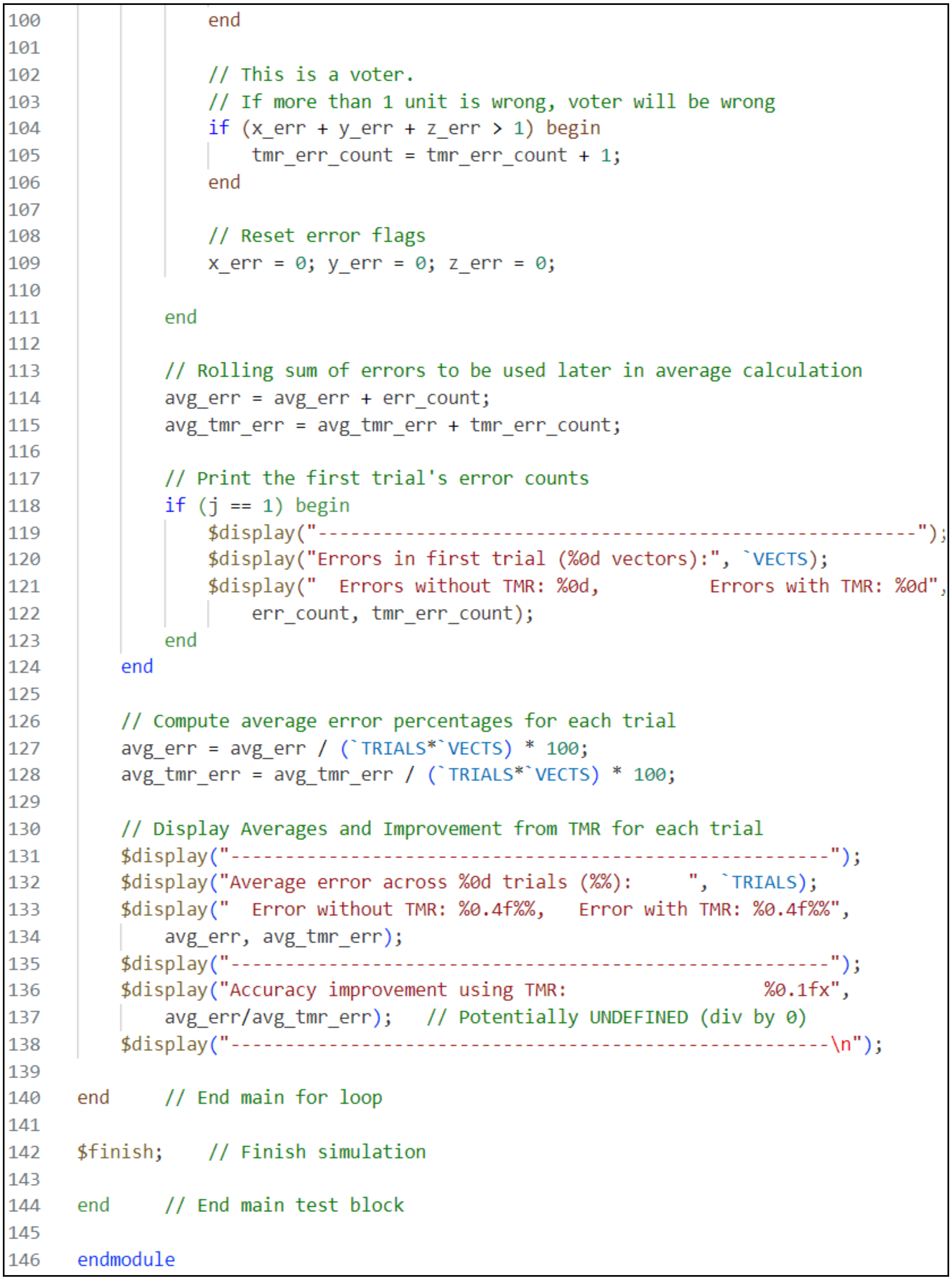
```
/*******************************************************************************
 ***                                                                         ***
 *** ECE 526 Final Project                       Zach Martin, Fall, 2024     ***
 ***                                                                         ***
 *** Triple Modular Redundancy                                               ***
 ***                                                                         ***
 *******************************************************************************
 *** Filename: TMR.v                   Created by Zach Martin, 12/01/2024    ***
 ***                                                                         ***
 *** Simulation demonstrating TMR vs Non-TMR accuracy with multiple tests    ***
 ***                                                                         ***
 *******************************************************************************/

`define ERR_RATE 10     // Error rate in percent, current max resolution = 0.1%
`define VECTS 100       // Number of test vectors
`define TRIALS 10       // Number of trials to perform. NOTE: MUST HAVE SEEDS

`timescale 1ns / 1ns

module TMR();

// Determines how to view waveforms depending on current compiler
`ifdef __ICARUS__
    initial begin
        $dumpfile("tmr.vcd");
        $dumpvars(0, TMR);
    end
`elsif __VCS__
    initial begin
        $vcdpluson;
    end
`endif

// Verilog random number generator seed to conform to or violate stability.
// Change seeds in procedural block to generate new values.
integer seeds[0:9];
integer seed;

// Loop counters, dice roll outcomes, and one-bit error flags
integer i = 0, j = 0, k = 0;
real x, y, z;
reg x_err, y_err, z_err;

// Error counters and average error percentages
reg [$clog2(`VECTS)-1:0] err_count = 0;
reg [$clog2(`VECTS)-1:0] tmr_err_count = 0;
real avg_err = 0;
real avg_tmr_err = 0;


```

Fig. 5. TMR.v Verilog module.



```

// ------------------------------ MAIN TEST BLOCK ------------------------------
initial begin

    // Define seeds. These can be any unique value to generate unique results
    seeds[0] = 0;
    seeds[1] = 10;
    seeds[2] = 200;
    seeds[3] = 3000;
    seeds[4] = 31415;
    seeds[5] = 27182;
    seeds[6] = 112358;
    seeds[7] = 628318;
    seeds[8] = 2;
    seeds[9] = 42;

for (i = 0; i < `TRIALS; i=i+1) begin

    seed = seeds[i];    // Set seed for current trial

    $display("\n\tTrial %0d (seed: %0d): ", i+1, seed);

    for (j = 0; j < `TRIALS; j=j+1) begin

        err_count = 0; tmr_err_count = 0;   // Reset error counts for new trial

        for (k = 0; k < `VECTS; k=k+1) begin

            #1;

            // Life is all about chance. Roll dice for each module
            x = $urandom(seed) % 1000;
            y = $urandom(seed) % 1000;
            z = $urandom(seed) % 1000;

            // See if the dice rolled above create SEU error
            if (x/10 < `ERR_RATE) begin
                x_err = 1;
            end if (y/10 < `ERR_RATE) begin
                y_err = 1;
            end if (z/10 < `ERR_RATE) begin
                z_err = 1;
            end

            #1;

            // First unit in TMR is also used as the non-triplex case.
            // More info on this can be found in the report.
            if (x_err) begin
                err_count = err_count + 1;
            end
```

Fig. 5. TMR.v Verilog module (continued).



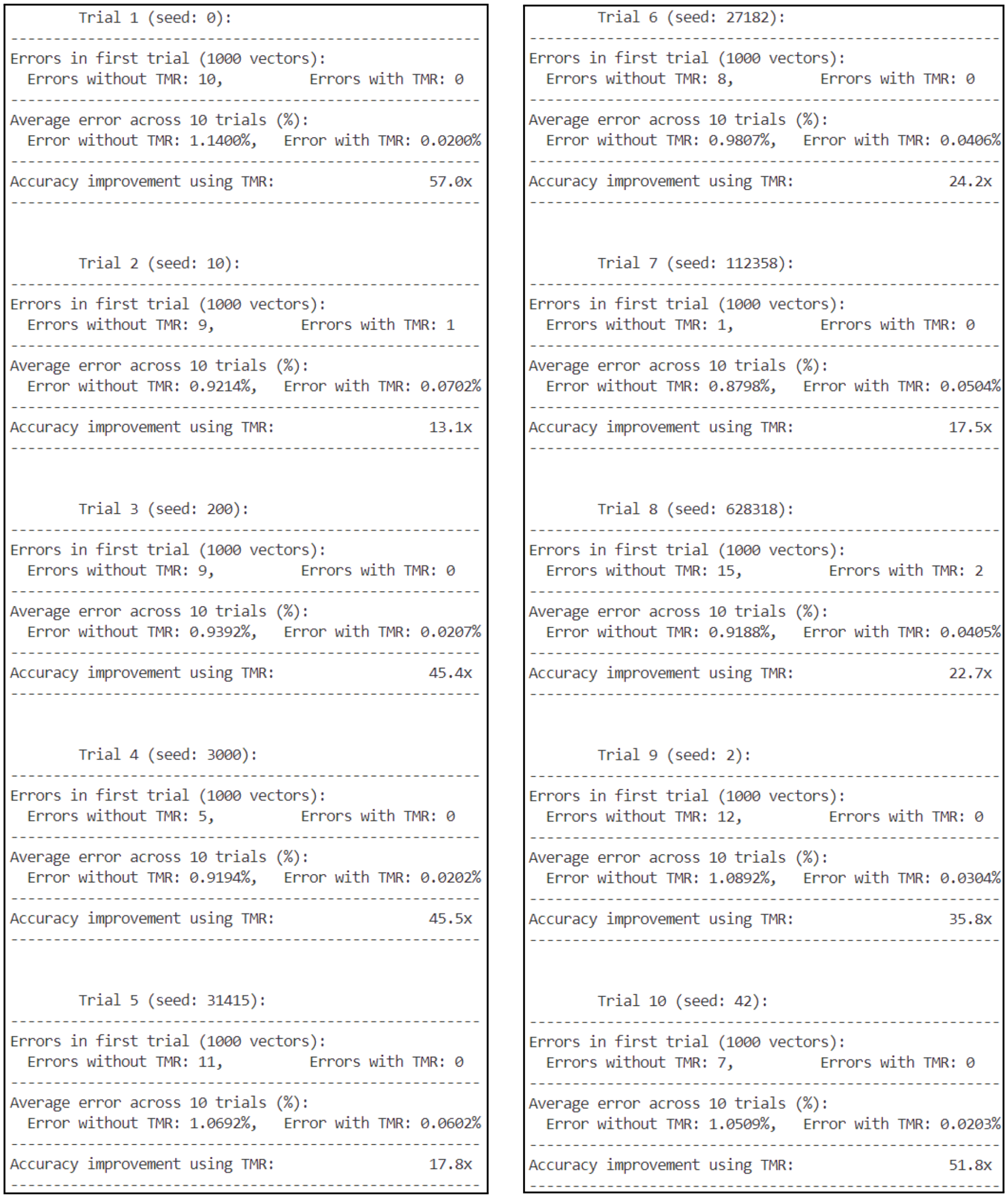
```
            end

            // This is a voter.
            // If more than 1 unit is wrong, voter will be wrong
            if (x_err + y_err + z_err > 1) begin
                tmr_err_count = tmr_err_count + 1;
            end

            // Reset error flags
            x_err = 0; y_err = 0; z_err = 0;

        end

        // Rolling sum of errors to be used later in average calculation
        avg_err = avg_err + err_count;
        avg_tmr_err = avg_tmr_err + tmr_err_count;

        // Print the first trial's error counts
        if (j == 1) begin
            $display("-------------------------------------------------------");
            $display("Errors in first trial (%0d vectors):", `VECTS);
            $display("  Errors without TMR: %0d,          Errors with TMR: %0d",
                err_count, tmr_err_count);
        end
    end

    // Compute average error percentages for each trial
    avg_err = avg_err / (`TRIALS*`VECTS) * 100;
    avg_tmr_err = avg_tmr_err / (`TRIALS*`VECTS) * 100;

    // Display Averages and Improvement from TMR for each trial
    $display("-------------------------------------------------------");
    $display("Average error across %0d trials (%%):     ", `TRIALS);
    $display("  Error without TMR: %0.4f%%,   Error with TMR: %0.4f%%",
        avg_err, avg_tmr_err);
    $display("-------------------------------------------------------");
    $display("Accuracy improvement using TMR:                  %0.1fx",
        avg_err/avg_tmr_err);   // Potentially UNDEFINED (div by 0)
    $display("-------------------------------------------------------\n");

end     // End main for loop

$finish;    // Finish simulation

end     // End main test block

endmodule
```

Fig. 5. TMR.v Verilog module (continued).

III. RESULTS

The program was executed 9 times with 9 different SEU probabilities using the Icarus Verilog compiler. The generated data was collected and processed in Google Sheets to calculate the average error with and without TMR as well as the accuracy improvement factor for each SEU probability, and to compare the experimental improvement to the calculated improvement given by the TMR error formula found in the introduction section of this report.

As expected, the improvements offered by TMR were found to be inversely proportional to the SEU probability. As SEU probability approached 50%, the accuracy improvements from TMR reduced to 0% (equivalent to a factor of 1). Past 50% SEU probability, the triplex system produced more errors than the reference system. As SEU probability approached 100%, accuracy degradation increased to 0% since both systems produced nearly 100% erroneous outputs. The experimental accuracy improvement factor conformed closely (error ≤ 0.05%) with the calculated accuracy improvement factors in all cases except for the SEU probability of 1%, which produced a percent error between the two values of approximately 10%. This is a result of minor variation in the experimental data; the calculation involves division of two very small values and is therefore sensitive to variation. This can be considered a consequence of too few test vectors creating a relatively low resolution of accuracy analogous to a large quantization error.



```
        Trial 1 (seed: 0):
-------------------------------------------------------
Errors in first trial (1000 vectors):
  Errors without TMR: 10,           Errors with TMR: 0
-------------------------------------------------------
Average error across 10 trials (%):
  Error without TMR: 1.1400%,   Error with TMR: 0.0200%
-------------------------------------------------------
Accuracy improvement using TMR:                   57.0x
-------------------------------------------------------


        Trial 2 (seed: 10):
-------------------------------------------------------
Errors in first trial (1000 vectors):
  Errors without TMR: 9,           Errors with TMR: 1
-------------------------------------------------------
Average error across 10 trials (%):
  Error without TMR: 0.9214%,   Error with TMR: 0.0702%
-------------------------------------------------------
Accuracy improvement using TMR:                   13.1x
-------------------------------------------------------


        Trial 3 (seed: 200):
-------------------------------------------------------
Errors in first trial (1000 vectors):
  Errors without TMR: 9,           Errors with TMR: 0
-------------------------------------------------------
Average error across 10 trials (%):
  Error without TMR: 0.9392%,   Error with TMR: 0.0207%
-------------------------------------------------------
Accuracy improvement using TMR:                   45.4x
-------------------------------------------------------


        Trial 4 (seed: 3000):
-------------------------------------------------------
Errors in first trial (1000 vectors):
  Errors without TMR: 5,           Errors with TMR: 0
-------------------------------------------------------
Average error across 10 trials (%):
  Error without TMR: 0.9194%,   Error with TMR: 0.0202%
-------------------------------------------------------
Accuracy improvement using TMR:                   45.5x
-------------------------------------------------------


        Trial 5 (seed: 31415):
-------------------------------------------------------
Errors in first trial (1000 vectors):
  Errors without TMR: 11,           Errors with TMR: 0
-------------------------------------------------------
Average error across 10 trials (%):
  Error without TMR: 1.0692%,   Error with TMR: 0.0602%
-------------------------------------------------------
Accuracy improvement using TMR:                   17.8x
-------------------------------------------------------
```

```
        Trial 6 (seed: 27182):
-------------------------------------------------------
Errors in first trial (1000 vectors):
  Errors without TMR: 8,           Errors with TMR: 0
-------------------------------------------------------
Average error across 10 trials (%):
  Error without TMR: 0.9807%,   Error with TMR: 0.0406%
-------------------------------------------------------
Accuracy improvement using TMR:                   24.2x
-------------------------------------------------------


        Trial 7 (seed: 112358):
-------------------------------------------------------
Errors in first trial (1000 vectors):
  Errors without TMR: 1,           Errors with TMR: 0
-------------------------------------------------------
Average error across 10 trials (%):
  Error without TMR: 0.8798%,   Error with TMR: 0.0504%
-------------------------------------------------------
Accuracy improvement using TMR:                   17.5x
-------------------------------------------------------


        Trial 8 (seed: 628318):
-------------------------------------------------------
Errors in first trial (1000 vectors):
  Errors without TMR: 15,           Errors with TMR: 2
-------------------------------------------------------
Average error across 10 trials (%):
  Error without TMR: 0.9188%,   Error with TMR: 0.0405%
-------------------------------------------------------
Accuracy improvement using TMR:                   22.7x
-------------------------------------------------------


        Trial 9 (seed: 2):
-------------------------------------------------------
Errors in first trial (1000 vectors):
  Errors without TMR: 12,           Errors with TMR: 0
-------------------------------------------------------
Average error across 10 trials (%):
  Error without TMR: 1.0892%,   Error with TMR: 0.0304%
-------------------------------------------------------
Accuracy improvement using TMR:                   35.8x
-------------------------------------------------------


        Trial 10 (seed: 42):
-------------------------------------------------------
Errors in first trial (1000 vectors):
  Errors without TMR: 7,           Errors with TMR: 0
-------------------------------------------------------
Average error across 10 trials (%):
  Error without TMR: 1.0509%,   Error with TMR: 0.0203%
-------------------------------------------------------
Accuracy improvement using TMR:                   51.8x
-------------------------------------------------------
```

Fig. 6. Command line output of TMR.v Verilog module using 1% SEU probability.

In order to enhance readability of waveform results, the program was modified to use a smaller number of test vectors. Figures 7 and 8 show simulation waveform data from 10 trials of 100 vectors each and SEU probability set to 10%. Note that err_count increases for any high x_err signal, but tmr_err_count only increases for two or more concurrently high error signals.

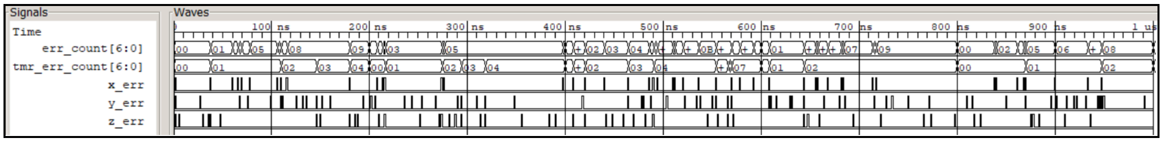


Fig. 7. Simulation waveform overview.

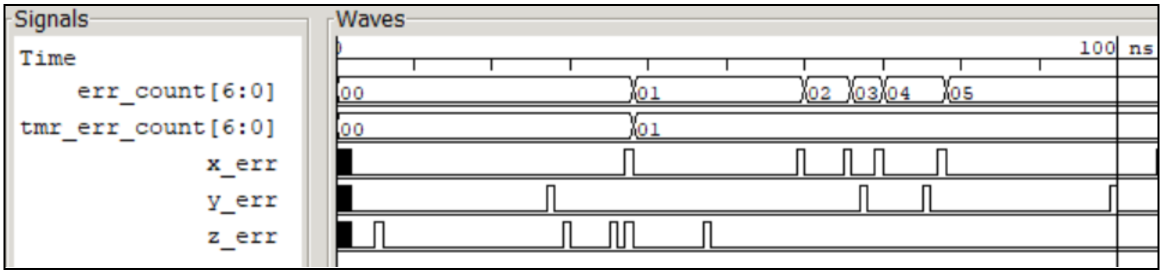


Fig. 8. Simulation waveforms showing the first of 10 trials (seed = 0).

TABLE II

SIMULATION RESULTS

Execution #	SEU Probability (%)	Average Error (%)	Average TMR Error (%)	Improvement (x)
1	1	0.9939	0.0271	36.676
2	10	9.9744	2.7915	3.573
3	25	24.9984	15.6314	1.599
4	40	40.0536	35.2596	1.136
5	50	50.0201	50.0319	1.000
6	60	60.0403	64.8380	0.926
7	75	75.0563	84.4115	0.889
8	90	90.0768	97.2490	0.926
9	99	99.0866	99.9970	0.991

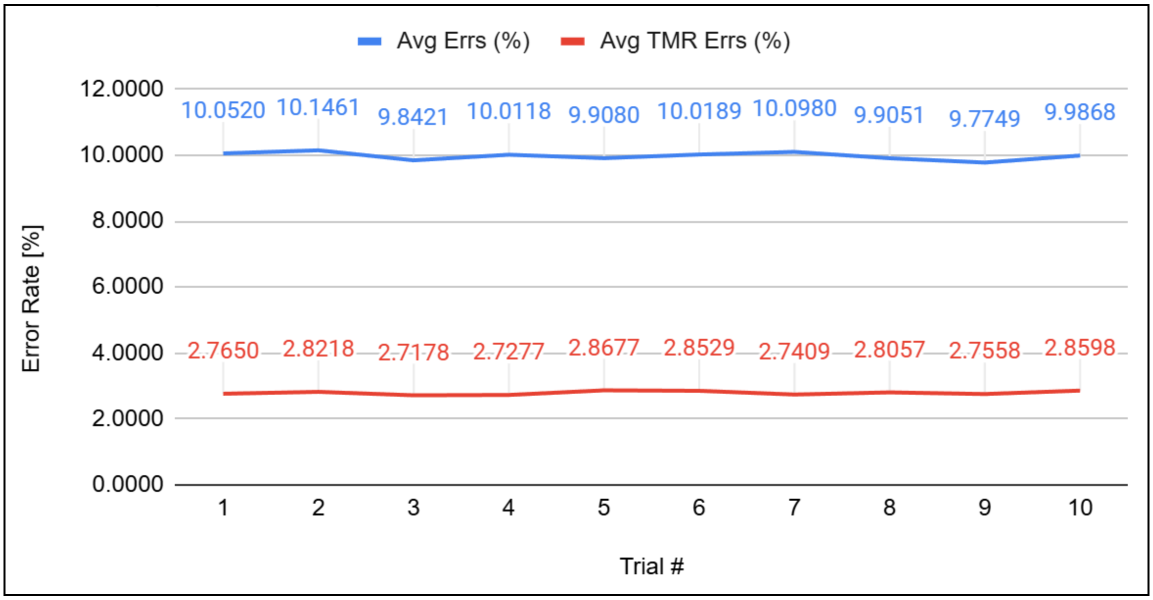


Fig. 9. Average error rate with and without TMR using 10% SEU probability.

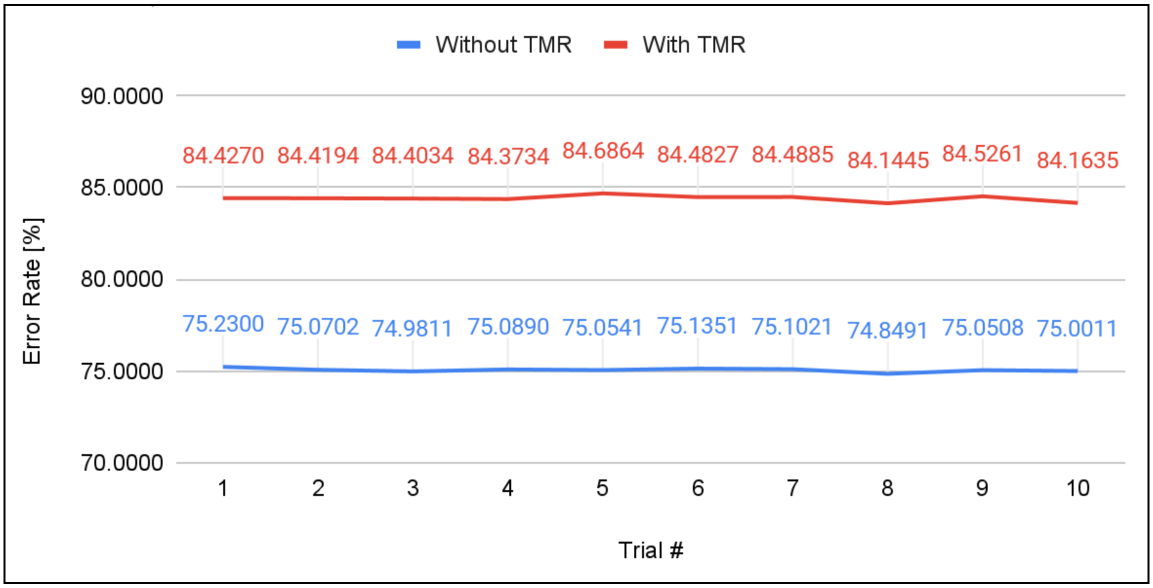


Fig. 10. Average error rate with and without TMR using 75% SEU probability.

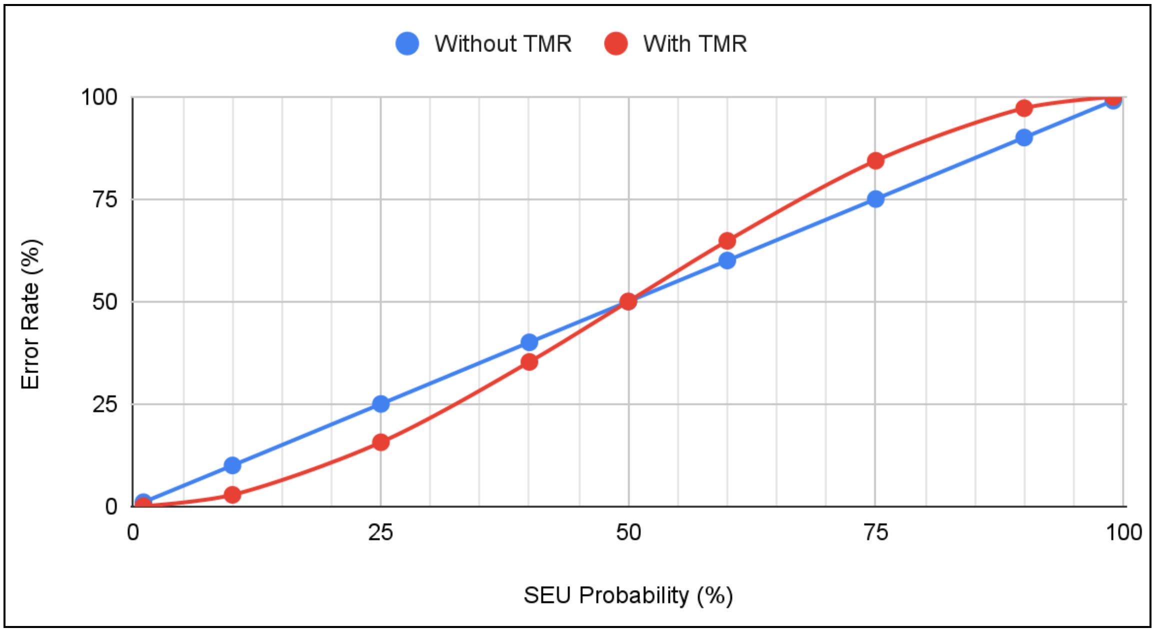


Fig. 11. Average error rate with and without TMR across all executions.

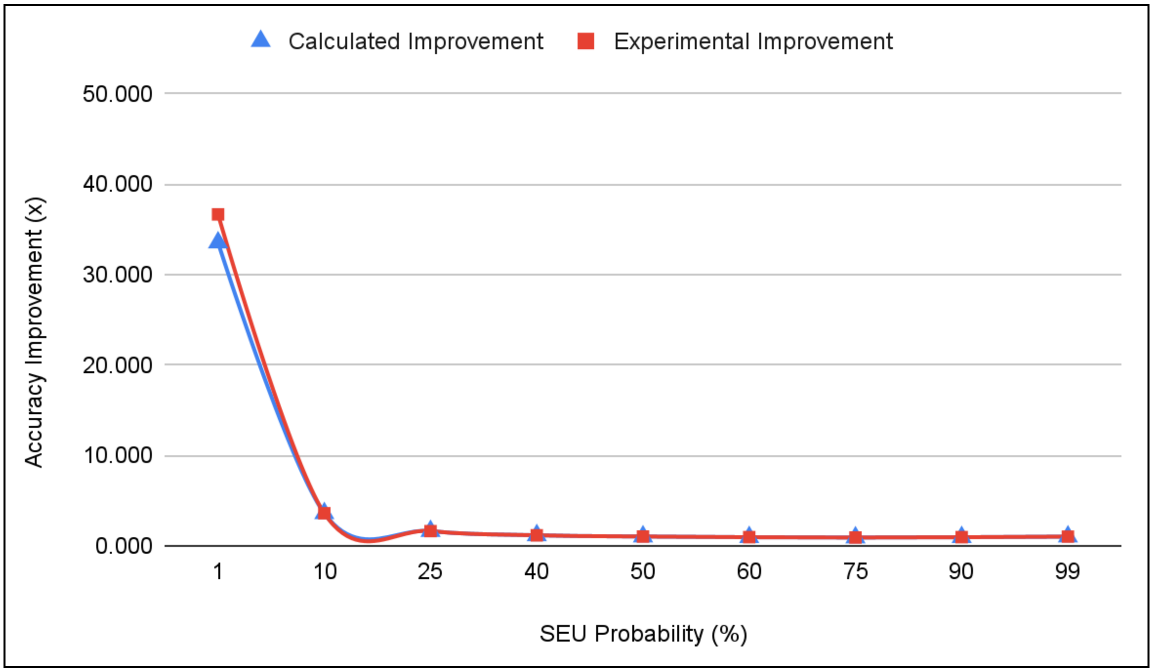


Fig. 12. Experimental and calculated accuracy improvement of TMR across all executions.

TABLE III

TMR IMPROVEMENT FACTORS

SEU Probability (%)	Calculated Error (%)	Calculated TMR Error (%)	Calculated Improvement (x)
1	1	0.0298	33.557
10	10	2.8000	3.571
25	25	15.6250	1.600
40	40	35.2000	1.136
50	50	50.0000	1.000
60	60	64.8000	0.926
75	75	84.3750	0.889
90	90	97.2000	0.926
99	99	99.9702	0.990

TABLE IV

EXPERIMENTAL VS. CALCULATED TMR IMPROVEMENT ERROR

SEU Probability (%)	Experimental TMR Improvement (x)	Calculated TMR Improvement (x)	Error (%)
1	36.676	33.557	9.29*
10	3.573	3.571	0.05
25	1.599	1.600	0.05
40	1.136	1.136	0.04
50	1.000	1.000	0.02
60	0.926	0.926	0.01
75	0.889	0.889	0.03
90	0.926	0.926	0.03
99	0.991	0.990	0.06

* Higher error is expected as SEU probability approaches zero due to variation in Verilog random number generation.

IV. Conclusion

This study investigated the performance of triple modular redundancy (TMR) as a fault-tolerant design strategy under various single event upset (SEU) probabilities. Through the simulation of three identical circuits exposed to ionizing radiation, both with and without TMR, the accuracy improvements offered by this technique were quantified and compared against theoretical predictions.

The results confirm that TMR provides substantial accuracy benefits when the probability of SEUs is low. As the SEU probability nears zero, TMR can improve reliability by an order of magnitude or more, effectively eliminating single-fault errors induced by radiation. The data indicate that the experimental accuracy improvements deviate minimally (less than 0.1% in most cases) from the theoretical values, reaffirming the validity of the model. The discrepancy seen at 1% SEU probability can be attributed to inherent statistical variation and the sensitivity of accuracy calculations at low error rates.

However, this study also highlights a critical limitation of TMR: its diminishing returns as SEU probability increases. Once the fault rate grows beyond 50%, TMR can, in fact, degrade overall reliability. Moreover, this approach inherently incurs higher costs, power consumption, and spatial requirements due to its triplication of components. These trade-offs must be weighed carefully, especially in high-radiation environments where TMR's gains are mitigated by its resource overhead.

TMR proves highly effective in environments with low to moderate SEU probabilities, offering a straightforward and robust method to enhance fault tolerance. When applied judiciously, such as in space exploration hardware, nuclear power plant instrumentation, and medical systems, TMR can significantly bolster reliability. Future research should explore hybrid approaches that combine TMR with other fault-tolerance methods to address its shortcomings in environments with high amounts of ionizing radiation and to optimize power, space, and cost considerations.

REFERENCES

[1] J. Gordon, "The Radiation Environment in Space," presented at the NM Workshop, University of Delaware, Newark, DE, 2015. [Online]. Available: https://www.bartol.udel.edu/~clem/NMworkshop2015/presentations/Gordon.pdf. [Accessed: Dec. 9, 2024].

[2] Imperva, "What is Fault Tolerance? | Creating a Fault Tolerant System," [Online]. Available: https://www.imperva.com/learn/availability/fault-tolerance/. [Accessed: Dec. 9, 2024].

[3] TechTarget, "What is Hamming code and how does it work?," [Online]. Available: https://www.techtarget.com/whatis/definition/Hamming-code. [Accessed: Dec. 9, 2024].

[4] D. P. Siewiorek and R. S. Swarz, "The Theory and Practice of Reliable System Design," in *Reliable Computer Systems: Design and Evaluation*, 2nd ed. Burlington, MA: Digital Press, 1988, pp. 395-422. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0164121288900143. [Accessed: Dec. 9, 2024].

[5] I. Tichy, "Triple Modular Redundancy Block Diagram," Wikimedia Commons, 2013. [Online]. Available: https://commons.wikimedia.org/wiki/File:Triple_Modular_Redundancy.JPG. [Accessed: Dec. 9, 2024].

[6] A. Author, "Majority Voter Logic Circuit," ResearchGate, 2015. [Online]. Available: https://www.researchgate.net/figure/Majority-voter-logic-circuit-designed-using-AND-gates-and-an-3-input-OR-gate_fig3_271207628. [Accessed: Dec. 9, 2024].

[7] E. Pet, "Majority Logic Circuit," Wikipedia, 2018. [Online]. Available: https://en.wikipedia.org/wiki/File:Majority_Logic.png. [Accessed: Dec. 9, 2024].

[8] R. E. Lyons and W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer Reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200-209, Apr. 1962. [Online]. Available: https://course.khoury.northeastern.edu/csg712/resources/Lyons-Vanderkulk-62.pdf. [Accessed: Dec. 10, 2024].

[9] F. P. Brooks Jr., *The Mythical Man-Month: Essays on Software Engineering*, Anniversary ed. Reading, MA: Addison-Wesley, 1995. [Online]. Available: https://web.eecs.umich.edu/~weimerw/2018-481/readings/mythical-man-month.pdf. [Accessed: Dec. 10, 2024].

[10] C. Darwin, *On the Origin of Species*, 2nd ed. London, UK: John Murray, 1860, p. 41. [Online]. Available: https://darwin-online.org.uk/content/frameset?viewtype=side&itemID=F10.2&pageseq=41. [Accessed: Dec. 10, 2024].

[11] K. Kostel, "Segal's Law of Navigation," National Deep Submergence Facility, Woods Hole Oceanographic Institution, Jul. 25, 2022. [Online]. Available: https://ndsf.whoi.edu/segals-law-of-navigation/. [Accessed: Dec. 10, 2024].

[12] U.S. Nuclear Regulatory Commission, "Digital Instrumentation and Control Systems in Nuclear Power Plants: Safety and Reliability Issues," NUREG/CR-6303, Sep. 1994. [Online]. Available: https://www.nrc.gov/docs/ml0929/ml092950511.pdf. [Accessed: Dec. 12, 2024].

[13] U.S. Nuclear Regulatory Commission, "Highly Integrated Control Rooms—Human Factors Issues," NUREG/CR-6399, Mar. 1997. [Online]. Available: https://www.nrc.gov/docs/ml1209/ML120900890.pdf. [Accessed: Dec. 12, 2024].

[14] NASA, "Apollo 8," Dec. 21, 1968. [Online]. Available: https://www.nasa.gov/mission/apollo-8/. [Accessed: Dec. 12, 2024].

[15] National Air and Space Museum, "Instrument Unit, Saturn," Smithsonian Institution, 1975. [Online]. Available: https://airandspace.si.edu/collection-objects/instrument-unit-saturn/nasm_A19750678000 . [Accessed: Dec. 12, 2024].

[16] J. Smith, "Saturn V Launch Vehicle Digital Computer And Data Adapter," in *Proceedings of the 10th Annual Conference on Example*, New York, NY, 2008, pp. 123-130. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/1464052.1464099. [Accessed: Dec. 12, 2024].

[17] NASA, "SA-503 Flight Manual," Dec. 1968. [Online]. Available: https://www.nasa.gov/wp-content/uploads/static/history/afj/ap08fj/pdf/sa503-flightmanual.pdf. [Accessed: Dec. 12, 2024].