A Path to Safer Digital Systems Using Proactive Hazard Analysis in Logic Circuit Design

Mahmood Jawad Abu-AlShaeer
Al-Rafidain University College
Baghdad, Iraq
dean@ruc.edu.iq

Sarah Ali Abdulkareem
Al-Turath University College
Baghdad, Iraq
sarah.ali@turath.edu.iq

Firas Mahmood Mustafa
Al-Noor University College
Nineveh, Iraq
firas.mahmood@alnoor.edu.iq

Nataliia Yakymenko
Central Ukrainian National
Technical University
Kropyvnytskyi, Ukraine
yakymenkonm@kntu.kr.ua

Abstract — **Background: In digital circuit design, assuring the safety and reliability of logic circuits is critical. Unexpected behaviors or performance abnormalities represent possible hazards to these circuits.**

Objective: Hazard analysis in logic circuits aims to anticipate potential circuitry difficulties and correct them during the design and testing phases. This proactive strategy eliminates any hazards that might jeopardize the circuit's dependability or safety.

Methods: The method is used to examine the design and functioning of a logic circuit, concentrating on its inputs, outputs, inherent logic, and time. This extensive study assists in anticipating any possible problems that the circuit may offer.

Results: Following the hazard analysis, the insights gained are used to enhance the circuit design, implement safety measures, or conduct more testing. This guarantees that the circuit satisfies the appropriate levels of safety and reliability.

Conclusion: Conducting a comprehensive hazard analysis is essential in creating safe and dependable logic circuits. Potential dangers may be discovered and minimized by meticulously evaluating a circuit's design and functioning, assuring smooth operation.

I. Introduction

Digital circuits are an essential component of current technology in the modern world. They may be everywhere, from consumer electronics to critical systems and infrastructure. Nevertheless, these circuits are sensitive to risks, such as glitches or race conditions, that may lead them to malfunction or provide inaccurate results. Techniques from the field of hazard analysis are used to locate and remove possible dangers in advance of their manifestation to guarantee the secure and dependable running of these circuits [1].

Hazard analysis is of the utmost importance in safety-critical applications, such as medical equipment or aerospace systems, where the implications of a glitch in the circuit might be catastrophic. Courses need to function correctly in these applications even though the timing may be off or receive unexpected inputs [2]. Thus, it is necessary to recognize and remove any potential dangers so the circuit will operate dependably. In a secure examination, the circuit design with dangerous timing difficulties may be identified as part of the examination cause. After that, precautions are made to protect against these dangers by, for example, including delay devices or retiming the circuit. Because of the growing complexity of digital circuits, hazard analysis has become essential to designing and verifying digital circuits [3].

In this post, we will talk about the numerous dangers that may develop in logic circuits and the approaches used to discover and eliminate such risks.

In addition, some of the constraints and limits of hazard analysis and potential solutions to these problems will be investigated. In addition to being helpful for students and researchers interested in the topic, the article will be valuable for engineers and designers who are actively engaged in creating digital circuits [4], [5].

The remainder of the article will be organized as follows. It will provide a concise introduction to digital circuits and the design of such courses. After that, it will define risks and talk about the many kinds of dangers that are possible in digital circuits [6]. The methods used throughout the hazard analysis process, such as Boolean logic, Karnaugh maps, and timing diagrams, will be discussed in the next section [7]. After that, the essay will cover some of the difficulties and constraints of hazard analysis, such as its complexity and lack of scalability, as well as the solutions to these problems. In conclusion, a brief review of the most critical elements will be provided and an explanation of hazard analysis's role in guaranteeing digital circuits' secure and dependable functioning.

A. The Aim of the Article

Hazard analysis in logic circuits aims to locate and eradicate possible hazards or glitches that may occur in digital circuits due to timing difficulties such as delays or race conditions. The identification and investigation of potential hazards accomplishes this. In this part of the analysis, we look at the circuit's layout to see where potential problems may arise and devise solutions to those problems. The objective is to guarantee that the circuit continues to perform correctly and reliably in the presence of timing fluctuations or inputs that were not anticipated, such as those found in the aircraft industry or the medical device industry, where even the smallest of malfunctions in the circuit might have catastrophic results.

B. Problem Statement

In contemporary society, there is a significant dependence on digital systems, including essential infrastructure and commonplace consumer gadgets. However, it is of utmost importance to prioritize the safety and dependability of these systems.

Presently, the predominant approach to safety evaluations in digital systems is characterized by a reactive nature, whereby dangers are identified after the occurrence of failures or accidents. The methodology mentioned above presents notable hazards, particularly in light of the escalating intricacy of digital systems. The issue pertains to the need for a structured and proactive approach to predict and address possible risks throughout the design stage.

In light of technological progress, it is essential to provide approaches and tools that facilitate the study of hazards throughout the circuit design phase. Using a proactive strategy can substantially decrease the probability of catastrophic failures, improve the safety of digital systems, and mitigate the possible ramifications of unanticipated dangers. It is essential to acknowledge and tackle this issue to maintain the ongoing dependability and security of the digital systems that serve as the foundation of contemporary society.

C. Combinational and Sequential Digital Logic Circuits

The premise of all contemporary computing systems is digital logic, sometimes known as Boolean logic. To put it another way, the set of principles allows us to answer seemingly straightforward "yes/no" questions with very complex outcomes.

Combinational and sequential digital logic circuits are the two main categories. Combinational circuits are digital circuits that carry out a predetermined task in response to a single set of inputs. They react only to the information sent to them and have no internal state or memory. The addition circuit, the decoding circuit, and the multiplexing circuit are all examples of combinational circuits [8].

However, the output of sequential circuits relies on both the inputs and the state of the circuit at any one time. Storage in sequential circuits is often accomplished using flip-flops and registers.

Combinational and sequential circuits serve essential roles in the design and implementation of digital systems like computers, digital signal processors, and microcontrollers and are, therefore, extensively utilized in digital electronics [9].

Simply put, combinational circuits are digital circuits constructed from a series of logic gates.

Many digital circuits, including decoders, encoders, multiplexers, and flip-flops, may be constructed from a collection of these basic logic gates. Combinational circuits may be created to accomplish comprehensive tasks utilizing logic gates, including arithmetic, data storage, and signal processing.

Notable features of the preceding picture:

The gate's name is often left from signs in favor of a simple symbol.

The A-B-Q terminal nomenclature is the norm. However, signals that are neither input nor output to the system are sometimes included in logic diagrams.

Third, while two inputs are the norm, you may sometimes come across gadgets with more. However, they will provide exactly one result.

These six symbols often represent digital logic circuits, with inputs on the left and outputs on the right. It is OK to link several inputs, but you should only connect one output. In any case, [2] linking more than one input to a single output is possible.

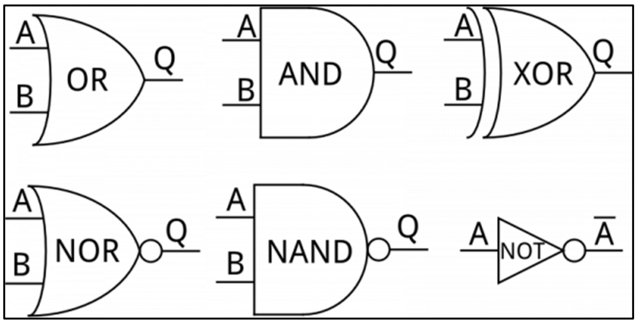
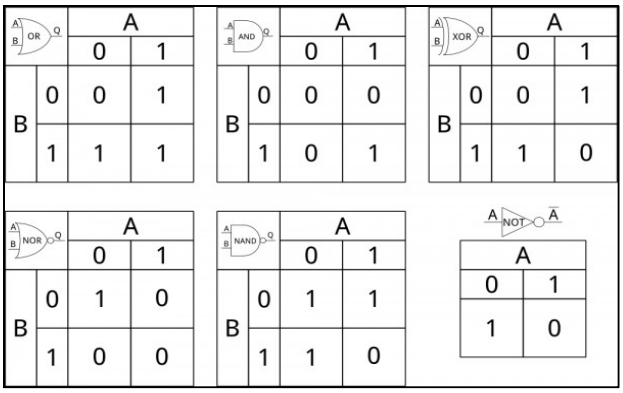


Fig. 1. Logic gates

D. Truth Tables

Single-block functionality can be described adequately using the previous examples, but a truth table is a more comprehensive resource. Truth tables are straightforward diagrams that show how different values in a circuit's inputs relate to different results [10].

In the following tables, we will describe the six components in detail.



OR		A	
		0	1
B	0	0	1
	1	1	1

AND		A	
		0	1
B	0	0	0
	1	0	1

XOR		A	
		0	1
B	0	0	1
	1	1	0

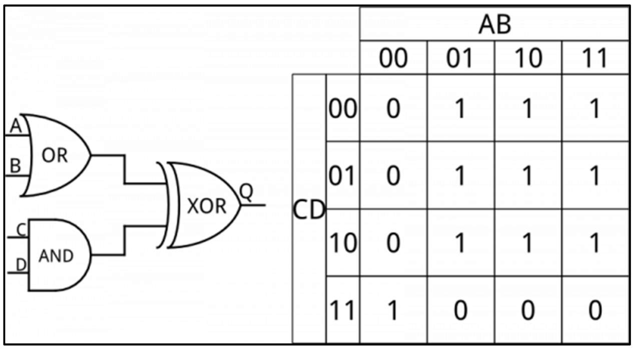
NOR		A	
		0	1
B	0	1	0
	1	0	0

NAND		A	
		0	1
B	0	1	1
	1	1	0

A	
0	1
1	0

Fig. 2. Truth tables

Adding as many columns to a truth table as your head can manage without exploding is possible. Here is an example of a truth table and a circuit with four inputs:



CD \ AB	00	01	10	11
00	0	1	1	1
01	0	1	1	1
10	0	1	1	1
11	1	0	0	0

Fig. 3. Four-Input Circuit

E. Written Boolean Logic

Boolean logic is a mathematical framework for working with binary values like true and false, 1 and 0, and on and off [11]. Boolean logic is used to represent and analyze the behavior of logic circuits to identify dangers as part of a hazard study.

The three primary logical operations of Boolean reasoning are AND, OR, and NOT. Complex expressions describing the circuit's behavior may be constructed using these procedures. Boolean expressions define the connection between the inputs and outputs of a gate or the criteria that must be satisfied for a particular danger to occur.

In order to better comprehend and evaluate the logic circuit, the findings of the hazard analysis may be presented clearly and succinctly by utilizing Boolean logic to reflect the circuit's behavior. In addition to facilitating formal analysis and verification of the circuit, Boolean logic offers a rigorous mathematical foundation, further enhancing the circuit's dependability and safety [12].

Two typical kinds of digital logic gates are the NAND and NOR gates.

The NAND gate combines the functionality of the AND NOT gates. If all of its inputs are 1s, the logic gate will produce a low-level output, and vice versa if the inputs are high-level.

In order to create the NOR gate, we first combine an OR gate with a NOT gate. If all of its inputs are 1s, it will produce 1s at its output; if any of them are 0s, it will produce 0s.

The AND-OR gates, the fundamental elements of digital circuits, are responsible for the AND OR operations. When all of the inputs to an AND gate are 1, the output is 1, whereas the output of an OR gate is 1 when any of the inputs is 1. (high).

An inverter, or NOT gate, is a simple logic gate that flips the input sign. An input of one produces an output of zero, and vice versa (low).

If you need the exclusive NOR action, go beyond the XNOR digital logic gate. If its inputs are equal, it produces a logical 1; whenever they are not, it produces a logical 0.

Each digital system, from the simplest logic circuit to the most sophisticated, would only be possible by employing the various gates described above.

F. Hazard

A false signal or glitch in the output of a combined logic circuit poses a threat. The temporary error circumstances - dangers - in actual circuits are caused by propagation delays. Threats would disappear if combinational logic circuits, including the interconnect wire, had zero delay time.

Until all the outputs have stabilized, the signals from a combinational logic circuit cannot be used (reached their final steady-state values). Using the risky output signal of combinational logic to drive a trigger - a device of sequential logic might be troublesome due to the possibility of a false state of the trigger being established by even a short transient error signal [13].

In the first study [6], authors discuss spurious outputs, which may arise when input signals change from one condition to another. Switching circuits provides a unique set of dangers because of stray delays in the elements and the circuit's inputs not constantly changing simultaneously.

According to a recent study [14], there are a few distinct categories of potential dangers. Dangers might be "static" or "dynamic," depending on how they move. Function and logic categorize all static and dynamic threats [15].

1) Static Hazards

A static hazard occurs when the output of a combinational circuit stays the same despite changes in the inputs, which can result in unintended behavior or malfunction [16].

The example given is of a 2-input AND gate, and it shows that if the inputs A and B are not received in synchrony, a logic 1 glitch can occur at the output of the gate. This glitch results from a brief high-level signal in a steady, low-level output signal, and it can cause problems in the operation of the circuit.

Avoiding static hazards in combinational circuits highlights the need for careful design and testing of digital circuits. Understanding the potential for static hazards and taking steps to mitigate them can ensure the reliability and safety of digital circuits. If scenario a occurs, no error signal is generated [17].

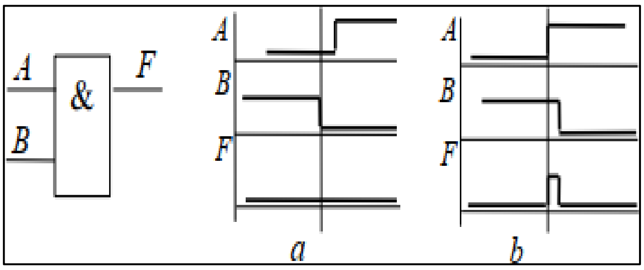


Fig. 4. Static hazards are divided into two types

Static hazards are divided into two types:

Static -1 hazard. A change in the input causes the output, which should be 1, to drop to 0 briefly. (Doable in AND-OR logic gates). The static-1 hazard is a glitch in an otherwise steady-state 1 voltage output from SOP logic, and it was referred to in the literature as SOP (sum of products) [2].

Static -0 hazard. In response to a change in input, the output briefly moves from 0 to 1. (Doable with OR-AND logic gates). The static -0 hazard, or POS (product of sum), is a glitch that arises from an average stable 0 output voltage from POS logic, as stated in the second cited paper [2].

A dynamic hazard occurs when a realized static hazard in one or more inputs causes a change in the value of the output signal.

2) Dynamic Hazards

The example given is of a 2-input OR gate, and it shows that if the input signals have unfavorable delay durations, a dynamic hazard can occur in the form of a broken 0-to-1 transition at the output of the gate. This type of hazard can also cause problems in the operation of the circuit, and it is important to avoid it in the design and testing of digital circuits [18].

Dynamic and static hazards highlight the importance of careful design and testing of combinational circuits. Understanding the potential for these hazards and taking steps to mitigate them can ensure the reliability and safety of digital circuits.

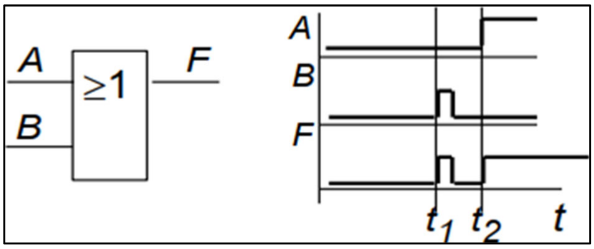


Fig. 5. Dynamic Hazards

3) Function Hazards

A function hazard is a potential problem that arises in the output signal due to simultaneous changes to several input signals.

The example given uses a Karnaugh map to demonstrate the presence of a function hazard [19]. The Karnaugh map is a graphical tool used to simplify Boolean expressions and to design and analyze combinational circuits. The example shows that when two input signals, A and C, transition from a value of 0 to a value of 1, a function hazard can occur if there is a delay in one of the signals concerning the other. This can result in a flip in the value of the function, which can cause problems in the operation of the circuit.

Function hazards, along with static and dynamic hazards, highlight the importance of careful design and testing of

Combinational circuits. Understanding the potential for these hazards and taking steps to mitigate them can ensure the reliability and safety of digital circuits [20].

1) Logic hazards

A logic hazard is a threat to the output signal that might arise from a change in a single input.

The example given uses a Karnaugh map to demonstrate the presence of a logic hazard. The Karnaugh map is a graphical tool used to simplify Boolean expressions and to design and analyze combinational circuits. The example shows that a logic hazard can occur when are two 1s in nearby squares of the Karnaugh map, but they are in different circles of 1s. This can result in unintended behavior or malfunction in the circuit [21].

Logic, static, dynamic, and function hazards highlight the importance of careful design and testing of combinational circuits.

$$F = p1 + p2 = AC + B\bar{C} \qquad (1)$$

Logic hazards can occur with a transition from square 111 to square 110 in a Karnaugh map.

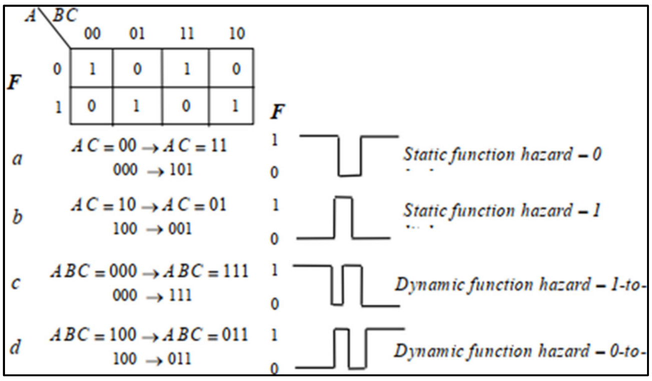

A
BC
00
01
11
10
F
0
1
1
0
1
0
1
0
1
0
1
F
a
$AC=00 \rightarrow AC=11$
$000 \rightarrow 101$
1
0
Static function hazard – 0
b
$AC=10 \rightarrow AC=01$
$100 \rightarrow 001$
1
0
Static function hazard – 1
c
$ABC=000 \rightarrow ABC=111$
$000 \rightarrow 111$
1
0
Dynamic function hazard – 1-to-
d
$ABC=100 \rightarrow ABC=011$
$100 \rightarrow 011$
1
0
Dynamic function hazard – 0-to-

Fig. 6. Function Hazar

The example given analyzes this transition in the logic diagram that implements the function's minimized Sum-of-Products (SOP) expression. The transition from square 111 to square 110 corresponds to the following change in the values of the logic variables: A = 1 → 1, B = 1 → 1, C = 1 → 0 (where only one logic variable changes its value).

It is explained that the presence of an inverter in input C is why signal Y is delayed for signal X, and this delay causes a spurious 0 glitch in output F. This glitch represents a logic hazard, which can cause unintended behavior or malfunction in the circuit.

Understanding the potential for logic hazards and taking steps to mitigate them can ensure the reliability and safety of digital circuits [22].

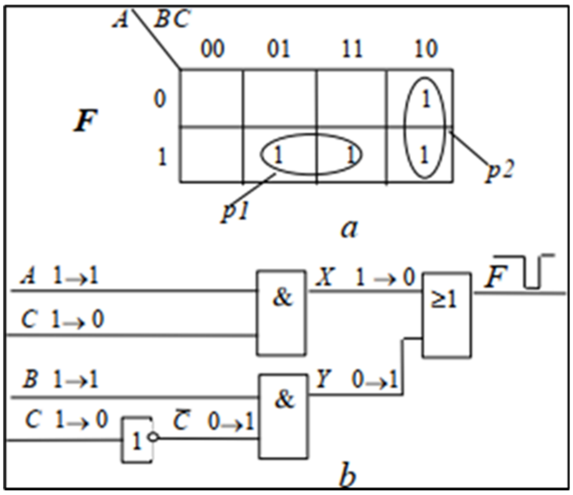


Fig. 7. Logic hazards

Safety risks in static logic caused by single and many input variations are the focus of the study. A bare minimum of effort is required to implement the approach, consisting of tinkering with a Boolean phrase representing the logic circuit. It does more than locate potential dangers and map out the surrounding area's connectivity. Not only does it detect the presence of dangers and identify the sets of variables about which their hazards exist, but it also identifies the subdues inside which any input transition involving precisely the modifications of all variables in the set creates the hazardous output.

II. Literature review

Industrial communities have developed processes to evaluate emerging needs for assigning a goal SIL for all Safety Integrity Systems (SIS) programs in response to the fast development of automation in the process sector. They are defined by certain norms [23]. PHA (Process Hazard Analysis) is a systematic assessment method used in industrial processes to identify and evaluate potential hazards and recommend necessary safety measures or systems. The installed instruments or controls that lessen the danger's impact or return the process to a safe condition constitute the SIS. When the PHA concludes that the process equipment, process control, or other protective equipment is inadequate to address a possible hazard adequately, the PHA may recommend the adoption of a SIS [24]. HAZOP (Hazard and Operability Study) is a detailed and systematic technique for identifying risks and operational issues in a process system by examining deviations from design intent. In light of the characteristics of a HAZOP study and the necessity to assign SIL for SIS, it has become apparent that HAZOP is a viable option for connecting its outcomes with the input data needed to initiate research and analysis for the SIL assignment. As a result, this situation is currently the subject of intensive research [25]. In particular, information gained during the HAZOP's final stage—during which the team identifies protections utilized to reduce hazardous events—helps think about SIL assignments. In addition, Logic Trees (Fault and Event Trees), created and solved numerically in any thorough risk analysis, have been included in HAZOP and made compatible with it. HAZOP is a valuable tool for plant design when paired with Logic Trees, which enables the designer to specify the SIL in line with the applicable event tree [4].

Based on his research on the six most popular PHAs used in the process industries, Marhavilas recommends the HAZOP study as the most intriguing approach for functional safety criteria. Again, HAZOP had a significant role in the development of LOPA. The author argued that the elements of LOPA provide a robust, analytical instrument for evaluating the sufficiency of protective layers to decrease process risk [26]. The first outlines how HAZOP could be tweaked to qualitatively assign the necessary SIL and then contemplates LOPA as a semi-quantitative method for classifying event severity, quantifying event initiation frequency, and calculating PFD values for each layer of protection [27]. With HAZOP complete, he reasoned, LOPA could be performed to determine the appropriate SIL for the majority of SIS operations, with PFD (Probability of Failure on Demand) quantifying the likelihood that a safety system will fail to perform its required function upon demand, crucial for assessing the reliability of safety-critical systems. Being considered for specific complicated systems in the following sections, he detailed the lessons learned throughout HAZOP sessions and LOPA preparation, which he then used to highlight particular criteria for automatically producing scenarios from HAZOP data for use in LOPA [28].

Two safety standards were mentioned above [17], and the criteria set for each. From tagging process dangers to crafting accident scenarios for each inciting event, they detailed the HAZOP-based procedures necessary to allocate target SIL. Then, the SIL process will be completed using various semi-quantitative or quantitative methods, all of which will be system-dependent. Lastly [19], sources present the LOPA as a method to be employed between HAZOP (as a qualitative hazard identification approach) and Fault Tree Analysis, and they provide thorough information on the characteristics and use of the LOPA (as a quantitative tool). Similarly, LOPA builds off of HAZOP's findings to provide a semi-quantitative assessment of the risk reduction afforded by each safeguarding by matching risk values to the company's criterion for unacceptable risk. To add, FTA may be used for a deeper dive into the data [29].

III. Methodology

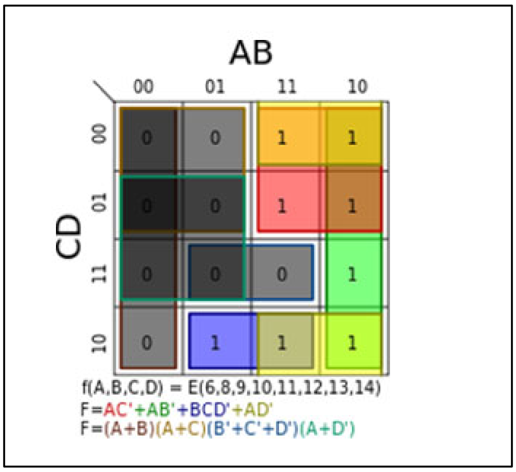
By using a Karnaugh map, Boolean algebra equations (KM or K-map) may be simplified. Maurice Karnaugh initially published it in 1953 [1], [2], a revision of Edward W. Veitch's 1952 Veitch chart (which was itself a rediscovery of Allan Marquand's 1881 logic diagram [5] aka Marquand diagram. The focus of this diagram is on its usage in switching circuits. Consequently, Veitch charts are often known as Marquand-Veitch diagrams [4], and Karnaugh maps are also called Karnaugh-Veitch maps (KV maps).

The Karnaugh map reduces the required calculations using humans' innate pattern-recognition skills. Any racial conditions could be easily identified and eradicated.

Each cell's placement in a Karnaugh map corresponds to a unique set of input conditions, with the requisite Boolean results being transferred from a truth table onto a two-dimensional grid in Gray code [6], [4]. The result of a Boolean function is shown in each cell, also known as minters. Sets of ones and zeros are identified [7] that most closely represent the text of the first truth table, which represents the canonical form of the logic. Using these ideas, a simple Boolean statement may be crafted that contains all the relevant reasoning [15], [25].

Karnaugh maps are an essential educational tool for streamlining Boolean calculations, particularly in situations involving a maximum of six variables. They enable the visualization and simplification of logical functions, improving the comprehension of critical principles in logic design. Acquiring proficiency in this simplification procedure is crucial for developing a deep understanding of sum-of-products (SOP) and product-of-sums (POS) expressions, which play a vital role in designing logic circuits [24]. The POS statement [8] states that if F is the function, then F' is its counterpart. Programmers may use Karnaugh maps to break problematic logical statements into more manageable chunks. Engineers often use these techniques at the beginning of circuit design, especially when dealing with less complex systems. However, in complex situations involving several variables, the efficiency of Karnaugh maps decreases. In their work on optical combinational logic circuits, Abdulnabi and Abbas [9] highlighted the need to use advanced computational approaches and software tools in such scenarios.

While Karnaugh maps may help clarify complex logical statements, they are often not used in advanced engineering projects that include modern digital systems' complicated and extensive nature, which need more advanced approaches.



CD \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	0	1
10	0	1	1	1

f(A,B,C,D) = E(6,8,9,10,11,12,13,14)
F=AC'+AB'+BCD'+AD'
F=(A+B)(A+C)(B'+C'+D')(A+D')

FiG.e 8. Example Karnaugh map

Truth table results map directly to Karnaugh map nodes.

Based on the example above, we may also deduce that A=0, B=0 inputs in the truth table provide an output of, which can be located in the Karnaugh map at the cellular address of A=0, B=0 at the top-left corner, at the intersection of the A=0 row and the B=0 column. Outputs from the other truth tables with inputs AB=01, 10, and 11 may be found at the relevant nodes in the Karnaugh map.

To better display the neighboring 2-cell areas in a 2-variable Karnaugh map, [21]a rectangular Boolean region is

used, analogous to a Venn diagram. This makes it much simpler to see how the truth table outputs relate to the Karnaugh map entries and examine the logic circuit.

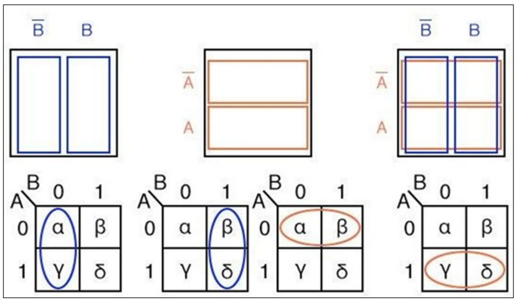
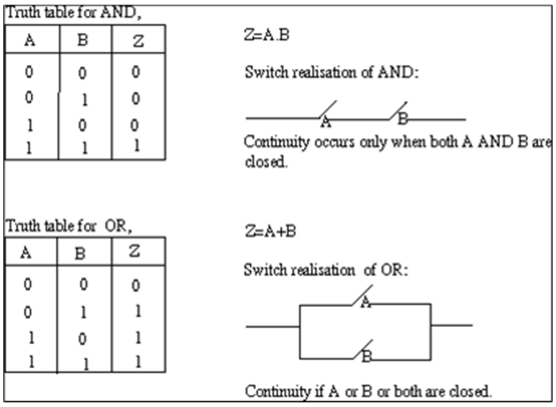


Fig. 9. Example Karnaugh map

Karnaugh maps are visual tools for finding patterns in large datasets and identifying and removing outliers. The Karnaugh map is just a unique truth table (Fig.10).

The connection between the Karnaugh map and the truth table for a generic issue with two variables. The picture referred to in Fig. 11 is likely a visual representation of this relationship, showing how the outputs of the truth table correspond to the entries in the Karnaugh map for a two-variable logic circuit.



Truth table for AND,

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

Z=A.B

Switch realisation of AND:

Continuity occurs only when both A AND B are closed.

Truth table for OR,

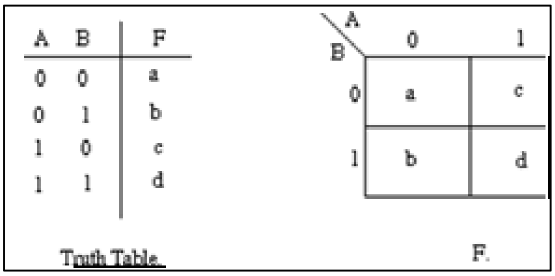
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Z=A+B

Switch realisation of OR:

Continuity if A or B or both are closed.

Fig. 10. Truth table Karnaugh



A	B	F
0	0	a
0	1	b
1	0	c
1	1	d

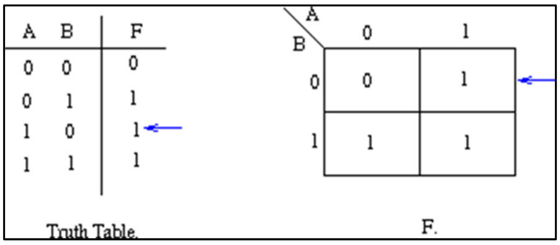
Truth Table.

B \ A	0	1
0	a	c
1	b	d

F.

Fig. 11. Mapping Truth Table to Karnaugh Ma

Each row of the truth table is represented by a square in the map, with values within the squares taken directly from the truth table's output column. The values of the two input variables are shown around the Karnaugh map's border. The letter A runs across the top, while B moves down the left side. This is shown in the figure below:



A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table.

B \ A	0	1
0	0	1
1	1	1

F.

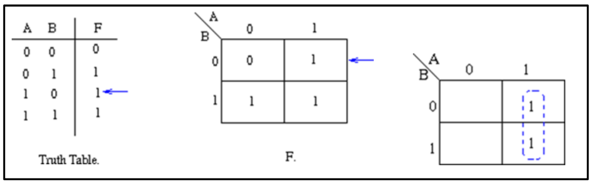
Fig. 12. Karnaugh map with two-input

The values in the Karnaugh map can be interpreted as coordinates. The example is that the square in the upper right corner of the map represents the values A=1 and B=0. This square is connected to the line inside the truth table where A=1, B=0, and F=1 indicate that these values correspond. This correspondence between the Karnaugh map and the truth table helps to analyze and simplify the logic circuit. Remember that the F column value denotes a specific function with a corresponding Karnaugh map [16].

Example:

Consider the following map. The function plotted is:

$$Z = f(A, B) = A + \bar{A}B = A + B$$



A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 13. Truth Table, where A=1 and B=0 and F=1

Remember that the rows and columns are the values of the input variables. The top of each row and column is labeled with the logical value of the variable A or B (with 1 signifying proper form and 0 denoting false forms).

You may use the above map to simplify a two-variable expression, but remember that it is a one-dimensional type.

The two- and three-dimensional maps may include up to four and six variables.

As you can see in the accompanying map, the two nearby ones (1s) are clustered together. One may see the true and false forms of variable B among the group members by examining it closely. As a result, only the genuine form of a second variable, A, remains. Z = A is the minimum cost solution.

Many strategies exist for removing potential logical risks: Restructuring how the logic is built. This is the bare-bones method of eliminating the threat. This strategy is workable for low-stakes logical threats. Nevertheless, removing the dangers by adjusting the logical architecture is extremely difficult, especially for complicated logical hazard circuits and other hazards [30]. Nonetheless, this does not rule out the possibility of a complicated logic circuit successfully removing risk in this manner. Avoiding all potential sources of error in designing logic circuits utilized for crucial events may be expensive. Elements linked to the tangent portions of the two circles may be made safe by adding redundant elements to the Karnaugh map, creating an extra circuit there. When simplifying a function, the hyperboloid term should be disregarded, but including it ensures the circuit will work as expected. The simplest solution is only sometimes the most effective.

Two, the Strobe Technique (or adding a blocking pulse). Wait for the output to settle before reading its value, and you may avoid the dangerous situation altogether. All dangers begin quickly following an altered input, and their duration is often brief. As a result, the risk may be avoided by waiting for the output to settle before reading it.

Using a filter as the third technique. To fix the output signal's interruptions, install a filter circuit (Fig.14). A low-pass filter may eliminate the hazard bump immediately because of its brief duration and high-frequency relative to the surrounding signal. The tiny negative transition pulse may be tamed without the gate circuit threshold voltage by connecting a small filter capacitor in parallel with the output terminal. In most cases, a filter capacitor is not the best solution to eliminate risk since it lengthens the rise and fall times of the output voltage waveform. This technique works well with low-frequency circuits and should be used strictly for debugging.

ISE (Integrated Synthesis Environment) (Integrated Synthesis Environment) A software suite called Xilinx designs, simulates and implements digital circuits in an ISE. Because of its usefulness in the construction of digital circuits, it sees widespread use in computer and electronics engineering disciplines.

TABLE 1. SIMPLIFYING AN ALGEBRAIC EXPRESSION

Boolean Theorem renders the B variable superfluous $Z = A\bar{B} + AB$ $Z = A(\bar{B} + B)$ $Z = A$
Theorem №1: Commutative Law $A + B = B + A$ $A\,B = B\,A$
Theorem №2: Associate Law $(A + B) + C = A + (B + C)$ $(A\,B)\,C = A\,(B\,C)$
Theorem №3: Distributive Law $A\,(B + C) = A\,B + A\,C$ $A + (B\,C) = (A + B)\,(A + C)$
Theorem №4: Identity Law $A + A = A$ $AA = A$
Theorem №5 $AB + A\bar{B} = A$ $(A + \bar{B}\,)A + \bar{B} = A + B$
Theorem №6: Redundance Law $A + A\,B = A$ $A\,(A + B) = A$
Theorem №7 $0 + A = A$ $0\,A = 0$
Theorem №8 $1 + A = 1$ $1\,A = A$
Theorem №9 $\bar{A} + A = 1$ $\bar{A}A = 0$
Theorem №10 $A + \bar{A}B = A + B$ $A\,(\bar{A} + B) = AB$
Theorem №11: De Morgan's Theorem $(\overline{A + B}) = \bar{A}\bar{B}$ $(\overline{AB}) = \bar{A} + \bar{B}$

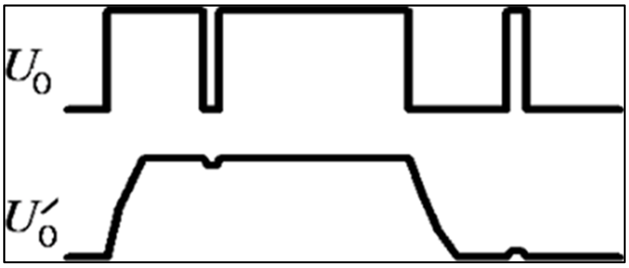


Fig. 14. Filter Method

The ability to simulate and validate circuits before actual fabrication is a key selling feature for ISE Xilinx. As a result, designers can identify and correct any issues with the logical circuitry of their design before committing to a final version [31].

Digital circuits may be designed and modeled using ISE Xilinx's client interface, which has various capabilities such as signal analyses, temporal studies, and circuit visualization. The program may also generate reports on the circuit's performance, such as how long it took to execute the commands you gave it and how well it performed under different loads [32], [33].

The ISE Xilinx program allows users to create digital circuits, test them via simulation, put them into action, and modify the principal image (Fig.15).

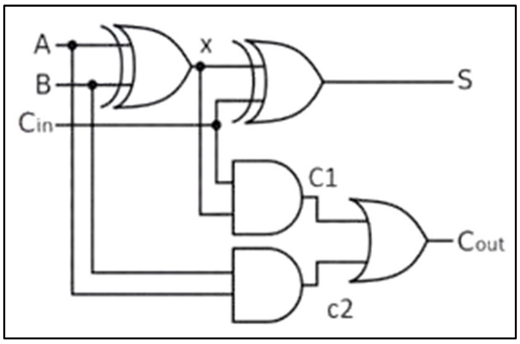
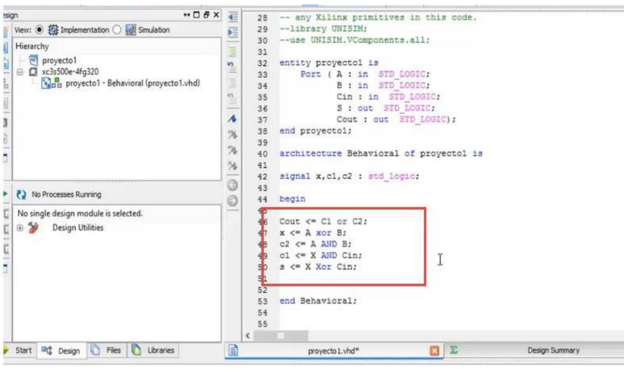


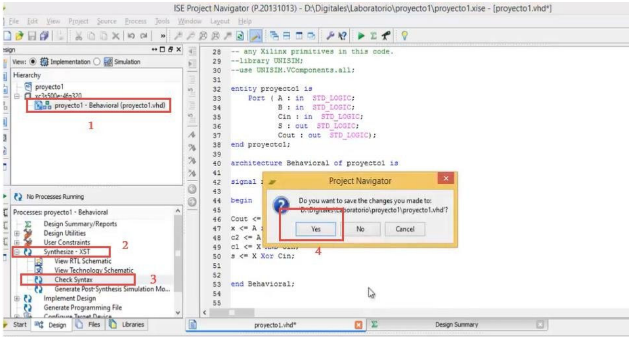
Fig. 15. Circuit for Article

We wrote the following code to program this circuit in the program, as in Fig. 16.



```
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity proyecto1 is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           Cin : in  STD_LOGIC;
           S : out  STD_LOGIC;
           Cout : out  STD_LOGIC);
end proyecto1;

architecture Behavioral of proyecto1 is

signal x,c1,c2 : std_logic;

begin

Cout <= C1 or C2;
x <= A xor B;
c2 <= A AND B;
c1 <= X AND Cin;
s <= X Xor Cin;


end Behavioral;
```

Fig. 16. Code for Article


ISE Project Navigator (P.20131013) - D:\Digitales\Laboratorio\proyecto1\proyecto1.xise - [proyecto1.vhd*]
File Edit View Project Source Process Tools Window Layout Help
View: ● Implementation ○ Simulation
Hierarchy
proyecto1
xc3s500e-4fg320
proyecto1 - Behavioral (proyecto1.vhd)
1
No Processes Running
Processes: proyecto1 - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
2
View RTL Schematic
View Technology Schematic
Check Syntax
3
Generate Post-Synthesis Simulation Mo...
Implement Design
Generate Programming File
Configure Target Device
Start
Design
Files
Libraries
proyecto1.vhd*
Design Summary
Project Navigator
Yes
No
Cancel
4

```
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity proyecto1 is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           Cin : in  STD_LOGIC;
           S : out  STD_LOGIC;
           Cout : out  STD_LOGIC);
end proyecto1;

architecture Behavioral of proyecto1 is

signal :

begin

Cout <=
x <= A
c2 <= A
c1 <= X
s <= X Xor Cin;


end Behavioral;
```

Do you want to save the changes you made to: D:\Digitales\Laboratorio\proyecto1\proyecto1.vhd'?

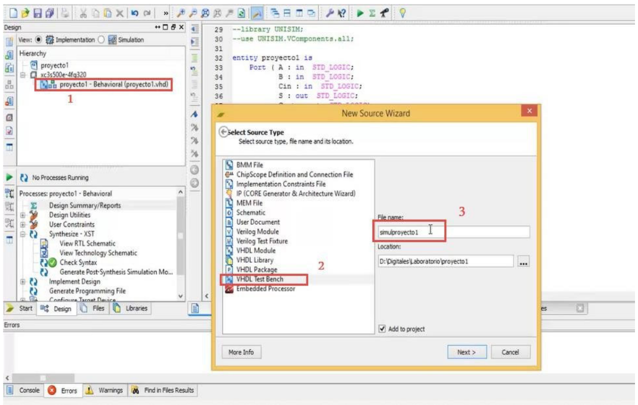
Fig. 17. Testing Code

Now we will check the code to make sure that there are no errors in the code:

1. Determine the name of the project;
2. Define xst;
3. Double-click on the scan to check the code;
4. Press ‘Yes’ to complete the process.

After checking and ensuring the code is correct, a checkmark appears.

Now, to simulate programming, we follow the following: Click on the project name and choose New Source:



```
--library UNISIM;
--use UNISIM.VComponents.all;

entity proyecto1 is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           Cin : in  STD_LOGIC;
           S : out  STD_LOGIC;
```

Select Source Type
Select source type, file name and its location.

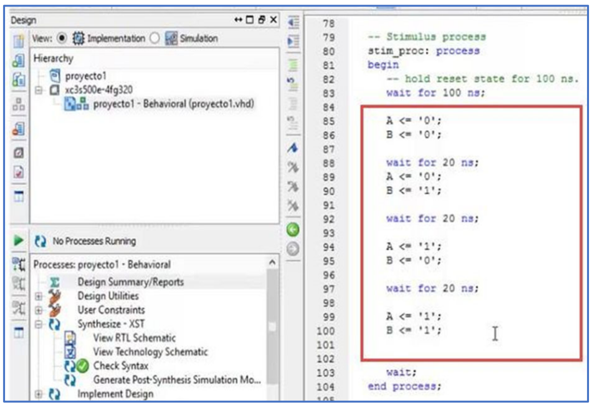
File name: simulproyecto1

Location: D:\Digitales\Laboratorio\proyecto1

☑ Add to project

Fig. 18. First Simulation Design

We write the truth table for the design in this place marked in red:

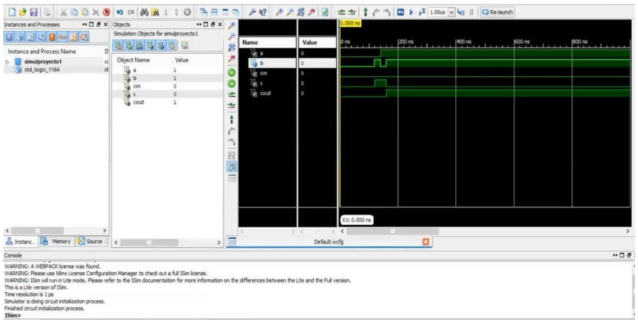


```

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      A <= '0';
      B <= '0';

      wait for 20 ns;
      A <= '0';
      B <= '1';

      wait for 20 ns;

      A <= '1';
      B <= '0';

      wait for 20 ns;

      A <= '1';
      B <= '1';


      wait;
   end process;
```

Fig. 19. Simulation Design Code

The final code will look like the one shown in the figure; there will be a delay of 100 ns between each implementation, and the code will be executed after waiting for 200 ns.

The simulation starts by selecting the option "Simulation" from the Simulation menu, followed by the project's name. At last, the simulation reveals how long each port's truth table takes to be executed.



Object Name	Value
a	1
b	1
cin	0
s	0
cout	1

Name	Value
a	0
b	0
cin	0
s	0
cout	0

```
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Fig. 20. Simulation Time

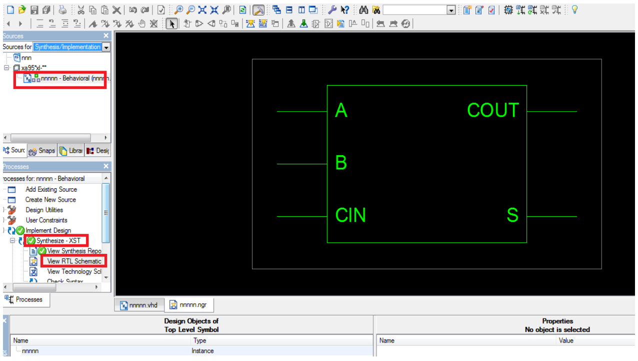


Fig. 21. Last Simulation Design

When you double-click on the figure, the logical circuit will appear full (Fig.22).

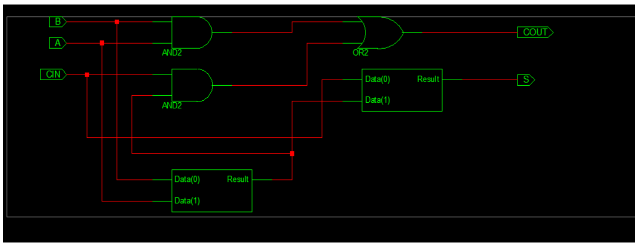


Fig. 22. Full Design Circuit

IV. Novelty

The article provides a novel approach for analyzing potential dangers in the design of logic circuits, emphasizing the use of ISE Xilinx for improved simulation and validation. This methodology significantly deviates from traditional manual inspection methods and adopts a proactive and predictive framework. The study presents a new hybrid technique that combines classical hazard analysis with computational tools, making it suitable for complex digital systems. This technique is particularly relevant in developing safety-critical applications where precision is paramount. Moreover, the article aligns with current industrial trends and technology advancements, offering valuable insights relevant to theoretical and practical settings. The significance of the contribution is underscored by its implications for engineering education, positioning it as a valuable asset for advancing curriculum development and professional training in hazard analysis.

V. Results

Conducting a hazard analysis is a crucial component of the design process for digital circuits. It is essential in safety-critical applications where the results of a malfunctioning course might be catastrophic. Even in the face of timing fluctuations or unexpected inputs, digital circuits may be made to work correctly and reliably with the aid of the findings of an in-depth risk assessment, which can be used to check that the risk assessment was carried out effectively. In this piece, we will look more in-depth at the findings from the risk assessment performed on logic circuits.

One of the most important things that may come out of doing a hazard analysis is the recognition of possible dangers or glitches that might take place in digital circuits. Timing concerns, such as delays or racing conditions, may produce these dangers, leading to malfunctions or inaccurate results. If designers can see possible dangers early on in the design process, they will have more time to devise solutions to prevent those dangers from materializing. This lowers the risk of malfunctions or failures, which is especially crucial in applications that emphasize safety.

In addition, the performance and dependability of digital circuits have significantly increased as a direct consequence of the findings of hazard analysis. Eliminating possible hazards enables designers to ensure that circuits perform correctly and consistently, even in the face of unexpected inputs or fluctuations in timing. This is accomplished by removing any potential risks that may exist. This has the potential to assist in reducing mistakes and improving the accuracy of circuit outputs, which are especially significant in applications such as aerospace, medical devices, or other safety-critical systems.

A further benefit of doing a hazard analysis is the establishment of reliable testing techniques, which can be used to verify that digital circuits are operating as intended. For designers to construct particular test scenarios and guarantee that circuits function as intended, it is necessary for them first to identify any possible risks. These test cases can be used during the project's design phase to validate that the circuit responds appropriately when subjected to various conditions. Afterward, they can be used during production testing to validate that the course responds appropriately when implemented in the final product.

The findings of hazard analysis may also assist designers in selecting suitable design strategies and components for digital circuits. This is possible because of the nature of the data. For instance, a hazard analysis may assist in determining which kinds of flip-flops or registers are the most suitable for use in sequential circuits. This helps to verify that the circuit performs appropriately and consistently. Similarly, doing a hazard analysis may assist in determining whether it is suitable to use delay elements or other timing-related components to reduce the likelihood of certain risks arising.

While hazard analysis may provide considerable advantages in designing digital circuits, knowing the limits and difficulties involved with using this method is essential. For example, hazard analysis may be complicated and time-consuming, especially for large and sophisticated digital circuits. In addition, it may be challenging to recognize all of the possible dangers, especially those brought about by the interactions between the various components of the circuit.

Another problem related to hazard analysis is the necessity to balance safety and dependability with other design objectives, such as speed or power consumption. The precautions taken to avoid potential dangers may make the functioning of the circuit slower or less efficient in terms of power consumption, both of which may be undesirable in specific contexts.

Notwithstanding these limitations, the findings of hazard analysis may be essential in guaranteeing the safe and dependable functioning of digital circuits. By detecting possible dangers and applying remedial steps, designers can ensure that circuits perform correctly and consistently, even in unexpected inputs or timing deviations. This may help decrease mistakes, avoid malfunctions, and ultimately enhance the safety and dependability of digital systems in various applications.

VI. Discussion

The study substantially adds to the digital systems design and safety field. The article's primary focus is prioritizing the proactive detection and mitigation of risks in designing logic circuits. The emphasis is placed on the significance of preventive measures in guaranteeing the dependability and security of digital systems. This discourse examines the primary discoveries and ramifications of the paperwork while including pertinent perspectives from the offered sources.

The proactive examination of hazards in the design of logic circuits is of utmost importance to detect and mitigate possible dangers before their manifestation in the functioning of digital systems. This technique is consistent with the overarching principle of hazard analysis, which finds use in diverse fields such as autonomous ships [2], energy efficiency in digital broadcasting [5], and multi-hazard modeling in mountainous regions [19]. The sources above emphasize the need to conduct hazard analysis in various settings and its usefulness in improving the resilience of systems.

The difficulty of hazard analysis in digital circuit design is a noteworthy factor. The study conducted by Ikenmeyer et al. explores the complexities of hazard-free circuits, providing insights into the difficulties involved in guaranteeing the absence of dangers in circuit designs [3]. A comprehensive comprehension of the computational complexity associated with hazard analysis is essential to develop efficient and effective methodologies, as elucidated in the article above.

The incorporation of hazard analysis techniques into the process of logic circuit design is a proactive measure aimed at mitigating the incidence of glitches and errors. The investigation conducted by Bathla et al. examines the decrease of glitch power in digital circuits, which aligns with hazard analysis goals by addressing undesired and possibly dangerous behaviors in circuits [14]. Reliability and safety in digital systems may be enhanced by optimizing circuit designs and minimizing glitches.

The significance of Boolean thinking, as first proposed by Kuhlmann et al., is of great importance in hazard analysis and defect identification [11]. The use of Boolean thinking is of utmost importance in the process of recognizing possible threats and guaranteeing the proper functioning of logic circuits. Using Boolean logic in hazard analysis approaches can enhance the efficiency of identifying hazardous circumstances and, hence, contribute to the overall safety of digital systems.

Within the realm of hazard analysis, the study further establishes connections between fault-tolerant control in quantum computing [13]. The primary subject matter of this article is classical digital systems. However, the underlying concepts of hazard identification and mitigation discussed herein apply to a broader scope, including the overarching objectives of enhancing the resilience and dependability of computing systems, irrespective of whether they are classical or quantum.

The proactive approach to hazard analysis recommended in the paper aligns with the overarching objectives of safeguarding the security and reliability of digital systems. With the rising complexity of digital circuits, there is a growing need to implement rigorous hazard analysis approaches. The essay highlights the significance of integrating hazard analysis into the first phases of logic circuit design by the principles of proactive safety engineering.

Moreover, concerning current progressions in hazard analysis, specifically the evaluation of safety integrity levels via dynamic Bayesian networks [23], it becomes apparent that proactive approaches to hazard analysis are consistently developing to tackle the increasing intricacy of digital systems.

The article emphasizes the need to use proactive hazard analysis to guarantee the safety and dependability of digital systems. Through the integration of hazard analysis approaches and the use of Boolean logic, designers can detect and address possible dangers in order to prevent the occurrence of glitches or defects. The proactive method described here is consistent with the larger context of hazard analysis in several areas. It highlights the importance and practicality of hazard analysis in improving the resilience and security of systems.

VII. Conclusions

Checking and fixing your work is an integral part of designing logic circuits. A digital computer may be used to create and test circuits via simulation. In most cases, computer simulations may save time and money by eliminating the need for expensive physical components. More and more time should be spent simulating the circuit before it is built. Because of the time and cost involved in debugging and fabrication, simulation is essential before an integrated circuit can be produced from the design. To verify the design is logically sound, all logical signals are correctly timed, and any problematic components in the circuit are identified and eliminated prior to construction, simulations are performed.

Logic circuit designs need specialized software to be built and simulated on a computer. From the outset, any parts must be located and linked to the logic inputs and outputs. After that, choose from among these inputs. Lastly, faults in the circuit outputs should be fixed by examining them. A logical diagram produced on a computer monitor or a list of links between logical components may be used to submit a complete circuit explanation to a simulation software tool.

This is an example of an elementary computer simulation program for a particular set of combinational logic:

The inputs are fed to the initial set of gates, which perform a calculation based on the values presented to them.

These modified first-stage outputs are then routed to the inputs of the following tier of gates. The exact value is determined for each entry regardless of whether or not it has changed.

Repeat this process until the gates' input values have not changed at all. Having reached this point, the output values may be read as the circuit is considered to be in a stable state. Whenever an entry needs updating, the first three steps are performed again.

References

[1] Y. Zhao, Y. Liu, and D. Ma: ''Output Regulation for Switched Systems With Multiple Disturbances'', *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67, (12), 2020, pp. 5326-35

[2] X.-Y. Zhou, Z. Liu, F. Wang, Z. Wu, and R. Cui: ''Towards applicability evaluation of hazard analysis methods for autonomous ships'', *Ocean Engineering*, 214, 2020, pp. 107773

[3] C. Ikenmeyer, B. Komarath, C. Lenzen, V. Lysikov, A. Mokhov, and K. Sreenivasaiah: 'On the complexity of hazard-free circuits.' Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, Los Angeles, CA, USA2018, pp. 878–89

[4] P. Fontanarrosa, H. Doosthosseini, A. E. Borujeni, Y. Dorfan, C. A. Voigt, and C. J. Myers: ''Genetic Circuit Dynamics: Hazard and Glitch Analysis'', *ACS synthetic biology*, 2020

[5] N. Qasim, Y. P. Shevchenko, and V. Pyliavskyi: "Analysis of methods to improve the energy efficiency of digital broadcasting", *Telecommunications and Radio Engineering*, 78, (16), 2019

[6] M. S. Tong, L. Y. Tang, and G. C. Wan: ''On the Teaching Reform for the Course of Digital Circuits and Logical Programming'', *2019 IEEE International Conference on Engineering, Technology, and Education (TALE)*, 2019, pp. 1-5

[7] J. K. Hillier, and R. S. Dixon: ''Seasonal impact-based mapping of compound hazards'', *Environmental Research Letters*, 15, (11), 2020, pp. 114013

[8] C. Anti'c: ''Boolean proportions'', *ArXiv*, abs/2109.00388, 2021

[9] S. H. Abdulnabi, and M. N. Abbas: ''Design an All-Optical Combinational Logic Circuits Based on Nano-Ring Insulator-Metal-Insulator Plasmonic Waveguides'', *Photonics*, 2019

[10] F. Z. Wang: ''A Triangular Periodic Table of Elementary Circuit Elements'', *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60, 2013, pp. 616-23

[11] A. Kuehlmann, M. K. Ganai, and V. Paruthi: 'Circuit-based Boolean Reasoning'. Proceedings of the 38th annual Design Automation Conference, Las Vegas, Nevada, USA2001, pp. 232–37

[12] P. W. Wilson, and F. Zanasi: 'Reverse Derivative Ascent: A Categorical Approach to Learning Boolean Circuits,' in Editor (Ed.)^(Eds.): 'Book Reverse Derivative Ascent: A Categorical Approach to Learning Boolean Circuits' (2021, ed.), pp.

[13] L. Egan, D. M. Debroy, C. Noel, A. Risinger, D. Zhu, D. Biswas, M. Newman, M. Li, K. R. Brown, M. Cetina, and C. Monroe: ''Fault-tolerant control of an error-corrected qubit'', *nature*, 598, 2021, pp. 281 - 86

[14] S. Bathla, R. M. Rao, and N. Chandrachoodan: ''A Simulation-Based Metric to Guide Glitch Power Reduction in Digital Circuits'', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27, 2019, pp. 376-86

[15] A. Tobin, S. Favelle, and R. Palermo: ''Dynamic facial expressions are processed holistically, but not more holistically than static facial expressions'', *Cognition and Emotion*, 30, 2016, pp. 1208 - 21

[16] H. Ma, Z.-y. Wu, and P. Chang: ''Social impacts on hazard perception of construction workers: A system dynamics model analysis'', *Safety Science*, 138, 2021, pp. 105240

[17] S. Huhn, S. Frehse, R. Wille, and R. Drechsler: ''Determining Application-Specific Knowledge for Improving Robustness of Sequential Circuits'', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27, (4), 2019, pp. 875-87

[18] F. Farahmandi, and P. Mishra: ''Automated Test Generation for Debugging Multiple Bugs in Arithmetic Circuits'', *IEEE Transactions on Computers*, 68, (2), 2019, pp. 182-97

[19] S. Yousefi, H. R. Pourghasemi, S. N. Emami, S. Pouyan, S. Eskandari, and J. P. Tiefenbacher: ''A machine learning framework for multi-hazards modeling and mapping in a mountainous area'', *Scientific Reports*, 10, (1), 2020, pp. 12144

[20] I. Pomeranz: ''Functional Constraints in the Selection of Two-Cycle Gate-Exhaustive Faults for Test Generation'', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29, 2021, pp. 1500-04

[21] M. P. Garrido: ''Simplifying Karnaugh Maps by Making Groups of a Non-Power-of-Two Number of Elements'', *arXiv: Signal Processing*, 2020

[22] D. A. Tran, A. Virazel, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, and H. J. Wunderlich: 'A Hybrid Fault Tolerant Architecture for Robustness Improvement of Digital Circuits,' in Editor (Ed.)^(Eds.): 'Book A Hybrid Fault Tolerant Architecture for Robustness Improvement of Digital Circuits' (2011, ed.), pp. 136-41

[23] C. Simon, W. Mechri, and G. Capizzi: "Assessment of Safety Integrity Level by Simulation of Dynamic Bayesian Networks considering test Duration", *Journal of Loss Prevention in the Process Industries*, 57, 2019, pp. 101-13

[24] B. Xin, J. Yu, W. Dang, and L. Wan: ''Dynamic characteristics of chlorine dispersion process and quantitative risk assessment of pollution hazard'', *Environmental Science and Pollution Research*, 28, (34), 2021, pp. 46161-75

[25] R. A. Viegas, F. d. A. d. S. Mota, A. P. C. S. Costa, and F. F. P. dos Santos: ''A multi-criteria-based hazard and operability analysis for process safety'', *Process Safety and Environmental Protection*, 144, 2020, pp. 310-21

[26] P. K. Marhavilas, M. Filippidis, G. K. Koulinas, and D. E. Koulouriotis: ''The integration of HAZOP study with risk-matrix and the analytical-hierarchy process for identifying critical control-points and prioritizing risks in industry – A case study'', *Journal of Loss Prevention in the Process Industries*, 62, 2019, pp. 103981

[27] L. Ding, W. Chen, T. Wang, R. Chen, Y. Luo, F. Zhang, X. Pan, H. Sun, and L. Chen: ''Modeling the Dependence of Single-Event Transients on Strike Location for Circuit-Level Simulation'', *IEEE Transactions on Nuclear Science*, 66, 2019, pp. 866-74

[28] J. R. Taylor: ''Automated HAZOP revisited'', *Process Safety and Environmental Protection*, 111, 2017, pp. 635-51

[29] G. Cristea, and D. M. Constantinescu: ''A comparative critical study between FMEA and FTA risk analysis methods'', *IOP Conference Series: Materials Science and Engineering*, 252, (1), 2017, pp. 012046

[30] A. H. El-Maleh, and K. A. K. Daud: ''Simulation-Based Method for Synthesizing Soft Error Tolerant Combinational Circuits'', *IEEE Transactions on Reliability*, 64, (3), 2015, pp. 935-48

[31] H. Modi, and P. M. Athanas: ''In-system testing of Xilinx 7-Series FPGAs: Part 1-logic'', *MILCOM 2015 - 2015 IEEE Military Communications Conference*, 2015, pp. 477-82

[32] M. X. Yue, D. Koch, and G. G. F. Lemieux: 'Rapid Overlay Builder for Xilinx FPGAs,' in Editor (Ed.)^(Eds.): 'Book Rapid Overlay Builder for Xilinx FPGAs' (2015, ed.), pp. 17-20

[33] K. V. B. V. Rayudu, D. R. Jahagirdar, and P. S. Rao: ''Design and testing of systolic array multiplier using fault injecting schemes'', *Computer Science and Information Technologies*, 2022