Design An Expert System To Detect The Errors In Logic Circuits

Eman Yousif Nasir*

Received on:29/3/2010
Accepted on:30/6/2010

Abstract

An expert system is a computer program designed to simulate the problem-solving behavior of a human who is an expert in a narrow domain or discipline. An expert system is normally composed of a knowledge base (information, heuristics, etc.) and inference engine (analyzes the knowledge base), and the end user interface (accepting inputs, generating outputs). One of the attractive features of expert systems is the program's ability to review a consultation and provide the user with an explanation for how its conclusion was derived. In this research, the expert system is used to detect the errors in logic circuits. The program is written in visual basic programming language depending on the truth table of logic gates so that the user doesn't need to save these truth tables in his mind. This program is applied in combinational logic circuit, which doesn't contain feedback. The program is answered for the questions of user "Why" and "How" to provide the description and details about gates and their works.

تصميم نظام خبير لاكتشاف الأعطال في الدوائر المنطقية

الخلاصة

النظام الخبير هو برنامج حاسوب يتم تصميمه لمحاكاة تصرف الانسان الخبير لحل مشكلة في مجال معين أو في فرع من فروع المعرفة . يتألف النظام الخبير من قاعدة معرفة (معلومات، موجهات (مساعدات على الكشف)، الخ) ووسائل استنتاج (لتحليل قاعدة المعرفة) والتعامل مع المستفيد (استقبال مدخلات وتوليد نتائج) . واحدة من الصفات الجذابة للنظام الخبير هو قدرته على استعراض وتقديم الشرح للمستخدم حول كيفية الوصول الى هذا الاستنتاج. تم استخدام النظام الخبير في هذا البحث للكشف عن الأعطال في الدوائر المنطقية وذلك بكتابة برنامج باستخدام لغة فيجوال بيسك بالاعتماد على جداول التحقق للبوابات المنطقية وهذا ما يغني المستخدم عن حفظ جداول التحقق للبوابات المنطقية المختلفة وتم تطبيقه على الدوائر المركبة فقط والتي لا تحتوي على التغذية الخلفية. كما تمت الاجابة على اسئلة المستخدم للنظام الخبير "لماذا" و "كيف" لمعرفة وصف وتفاصيل عمل البوابات.

1. Introduction

Expert systems (ES) emerged as a branch of artificial intelligence (AI), from the effort of AI researchers to develop computer programs that could reason as humans.[1]

Artificial Intelligence (AI) is a branch of computer science, which focuses on the development of computer systems to simulate the processes of problem solving and duplicate human brain

*Laser and Optoelectronics Engineering Department, University of Technology/Baghdad

https://doi.org/10.30684/etj.28.16.13
2412-0758/University of Technology-Iraq, Baghdad, Iraq

functions. Expert systems (ES) are a category of programs based on the theory and methods of artificial intelligence. [2]

Expert systems are built on knowledge gathered from human experts, analogous to a database but containing rules that may be applied to solving a specific problem. An interface allows the user to specify symptoms and to clarify a problem by responding to questions posed by the system. [3]

It derives its answers by running the knowledge base through an inference engine, which is software that interacts with the user and processes the results from the rules and data in the knowledge base.

Examples of uses are medical diagnosis, equipment repair, investment analysis, financial, estate and insurance planning, vehicle routing, contract bidding, production control and training [3]

Expert systems tend to be more effective than other computer based applications, because they:

- May combine the knowledge of many experts in a specific field,
- Can store an unlimited amount of information, and works much faster, than a human.
- Are available 24 hours a day, and can be used at a distance over a network.
- Are able to explain their information requests and suggestions.
- Can process client's uncertain responses and, by combining several pieces of uncertain information, may still be able to make strong recommendations.
- Can accumulate the knowledge of high level employees for any company, which is especially useful when the company needs to fire them due to worsened market conditions. [2]

In this research, the expert system is used to detect the errors in combinational logic circuits depending on the truth table of logic gates.

2. Expert System Structure

The expert system structure helps in refining the information from raw data to the one supporting a particular application like fault diagnosis or condition monitoring or preventive maintenance or any other area where extensive data is available and assimilation of data or extracting useful information or knowledge from the data is difficult. It is a rule based expert system shell developed using design details.
As shown in figure 1. The development of the expert system shell Involves:
1. Knowledge Base Design:
The knowledge base of the expert system shell stores the extensive knowledge gathered from experts, historical data and books regarding the application in the form of rules. The knowledge can be either factual or heuristic. The knowledge is stored in the form of production rules, which contain if-else rules. The knowledge

base is divided into three major sections namely: variable section, rules section and asks section. The variables section is used to declare all the variables used in the knowledge base, the rules section is used to define all the rules used to represent the knowledge and the asks section includes all the questions and options for the user to select and continue with the consultation for getting the required assistance from the expert system.

2. Database Design:

The information gathered is either factual, heuristic or query based. Of the data gathered, the data which can be put in the form of a question and answer are put in the database of the application. The database table of the expert system shell has three fields namely: the id, question and answer. The database can be accessed by entering the keyword related to the information seeked. This leads to displaying of all the questions containing the keyword and selection of any one of the questions would provide for the displaying of the required information.

3. Development of an inference engine

It provides the system control. It applies the expert domain knowledge to what is known about the present situation to determine new information about the domain. The inference engine is the mechanism that connects the user input in the form of answers to the questions to the rules of knowledge base and further continues the session to come to conclusions.

This process leads to the solution of the problem. The inference engine also enables the expert system's interface to data sources and to the user. The inference engine also supports to identify the rules of the knowledge base used to get decision support from the system and also forms the decision tree.

4. Development of a graphical user interface

The graphical user interface of the expert system comprises of:

a. Input Interface:

This allows the expertise to be uploaded as the knowledge base files in text format and also allows updating the database tables

b. User Interface:

This allows for creation of new users and also allows the existing user to consult the expert system in a user friendly manner for decision support.

c. Working Memory:

The working memory contains the information that the system has received about a problem at hand. In addition, any information the expert system derives about the problem is also stored in the working memory. [2, 4]

3. Explanation system

The major distinction between expert systems and traditional systems is illustrated by the following answer given by the system when the user answers a question with another question.

It is very difficult to implement a general explanation system (answering questions like "Why" and "How") in a traditional computer program. An expert system can generate an explanation by retracing the steps of its reasoning. The response of the expert system to the question WHY is an exposure of the underlying knowledge structure. It is a rule; a set of antecedent conditions which, if true, allow the assertion of a consequent. The rule references values, and tests them against various constraints or asserts constraints onto them. This, in fact, is a significant part of the knowledge structure. There are values, which may be associated with some organizing entity. There are also rules, which associate the currently known values of some attributes with assertions that can be made about other attributes. It is the orderly processing of these rules that dictates the dialog itself. [3]

4. Digital Logic Circuits

Electronic circuits which process information encoded as one of a limited set of voltage or current levels. Logic circuits are the basic building blocks used to realize consumer and industrial products that incorporate digital electronics. Such products include digital computers, video games, voice synthesizers, pocket calculators, and robot controls.

Everything in the digital world is based on the binary number system. Numerically, this involves only two symbols: 0 and 1. Logically, we can use these symbols or we can equate them with others according to the needs of the moment. Thus, when dealing with digital logic, we can specify that:

0 = false = no

1 = true = yes

Using this two-valued logic system, every statement or condition must be either "true" or "false;" it cannot be partly true and partly false. While this approach may seem limited, it actually works quite nicely, and can be expanded to express very complex relationships and interactions among any number of individual conditions.

One essential reason for basing logical operations on the binary number system is that it is easy to design simple, stable electronic circuits that can switch back and forth between two clearly-defined states, with no ambiguity attached. It is also readily possible to design and build circuits that will remain indefinitely in one state unless and until they are deliberately switched to the other state. This makes it possible to construct a machine (the computer) which can remember sequences of events and adjust its behavior accordingly. [5]

Digital logic may be divided into two classes: *combinational logic*, in which the logical outputs are determined by the logical function being performed and the logical input states at that particular moment; and *sequential logic*, in which the outputs also depend on the prior states of those outputs. Both classes of logic are used extensively in all digital computers.[5,6]

4.1 Combinational Logic Circuit

The outputs of Combinational Logic circuits are only determined by their current input state as they have no feedback, and any changes to the signals being applied to their inputs will immediately have an effect at the output. In other words, in a Combination Logic circuit, if the input condition changes state so too does the output as combinational circuits have No Memory.

Combination Logic circuits are made up from basic logic AND, OR or NOT gates that are "combined" or connected together to produce more complicated switching circuits.

As combination logic circuits are made up from individual logic gates they can also be considered as "decision making circuits" and combinational logic is about combining logic gates together to process two or more signals in order to produce at least one output signal according to the logical function of each logic gate. [7]

Logic gates

Logic gates are electronic circuits that operate on one or more input signals to produce an output signal. Electronic signals such as voltages or currents exist throughout a digital system in either of two recognizable values. Voltage-operated circuits respond to two separate voltage levels that represent a binary variable equal to logic 1 or logic 0. For example, a particular digital system may define logic 0 as a signal equal to 0 volt and logic 1 as a signal equal to 4 volts. [8]

The basic logic AND**,** OR are referred to as *binary operators,* because they require two operands. NOT is a *unary operator*, meaning that it requires only one operand. The NOT operator returns the complement of the input: 1 becomes 0, and 0 becomes 1. When a variable is passed through a NOT operator, it is said to be *inverted*.

While this is a simple example, it is representative of the fact that any logical relationship can be expressed algebraically with products and sums by combining the basic logic functions AND, OR, and NOT.

Several other logic functions are regarded as elemental, even though they can be broken down into AND, OR, and NOT functions. These are not–AND (NAND), not–OR (NOR), exclusive–OR (XOR), and exclusive–NOR (XNOR). XOR is an interesting function, because it implements a sum that is distinct from OR by taking into account that 1 + 1 does not equal 1. XOR plays a key role in arithmetic for this reason. Figure 2 shows how the basic logic gates are drawn on a circuit diagram. Naming the inputs of each gate A and B and the output Y is for reference only; any name can be chosen for convenience. A small bubble is drawn at a gate's output to indicate a logical inversion. [6]

The input and output information of any Logic Gate or circuit can be plotted into a table to give a visual representation of the switching function of the system and this is commonly called a Truth Table. Logic gate truth table shows each possible input to the gate or circuit and the resultant output depending upon the combination of the input(s). [7]

Truth tables are often used to illustrate logical relationships as shown in Table 1. A truth table provides a direct mapping between the possible inputs and outputs. A basic AND operation has two inputs with four possible combinations, because each input can be either: 1 or 0 — true or false. Mathematical rules apply to Boolean algebra, resulting in a nonzero product only when both inputs are 1.

A common means of representing the output of a generic logical function is with the variable Y. Therefore, the AND function of two variables, A and B, can be written as Y = A & B or Y = A*B. As with normal mathematical notation, products can also be written by placing terms right next to each other, such as Y = AB. Notation for the inverted functions, NAND, NOR, and XNOR, is achieved by inverting the base function. Two equally valid ways of representing NAND are Y = A & B and Y =!(AB). Similarly, an XNOR might be written as Y = A⊕B. In circuit terminology, the logical operators are called *gates.* [6]

Multiple-input gates

Adding more input terminals to a logic gate increases the number of input state possibilities.

The number of possible input states is equal to two to the power of the number of inputs:

Number of possible input states = 2^n

Where,

n = Number of inputs

This increase in the number of possible input states obviously allows for more complex gate behavior. [9]

The three input AND gate responds with logic 1 output if all three inputs are logic 1. The output produces logic 0 if any input is logic 0. The four-input OR gate responds with logic 1 if any input is logic 1; its output becomes logic 0 only when all inputs are logic 0. All binary operators can be chained together to implement a wide function of any number of inputs. [6]

4.2 Sequential Logic Circuit

Sequential logic differs from combinational logic in that the output of the logic device is dependent not only on the present inputs to the device, but also on past inputs; *i.e.*, the output of a sequential logic device depends on its present internal state and the present inputs. This implies that a sequential logic device has some kind of *memory* of at least part of it's ``history'' (*i.e.*, its previous inputs).

A simple memory device can be constructed from combinational devices with which we are already familiar. By a memory device, we mean a device which can remember if a signal of logic level 0 or 1 has been connected to one of its inputs, and can make this fact available at an output. A very simple, but still useful, memory device can be constructed from a simple OR gate, as shown in Figure 3 Note that the output of the memory is used as one of the inputs; this is called *feedback* and is characteristic of programmable memory devices. (Without feedback, a ``permanent'' electronic memory device would not be possible.) The use of feedback in a device can introduce problems which are not found in strictly combinational circuits. In particular, it is possible to inadvertently construct devices for which the output is not determined by the inputs, and for which it is not possible to predict the output.[10]

5. A new method for calculating the effect of faults in logic circuits: The bit string method

For the efficient simulation of classical faults in logic circuits, the two basic methods used are the parallel fault calculation method and the deductive fault calculation method. The 'bit string' method is a hybrid of the parallel and deductive methods. Instead of a Tval (true logic value) and fault word (for parallel) or fault list (for deductive), each gate has a Tval, a bit string of length just long enough to hold all the bits that are different from the Tval, and the starting displacement of the string (fault number of the first bit in the string). Algorithms for the efficient calculation of the output bit string of a gate given its input bit strings were developed. A bit string simulator was compared to similar implementations of a deductive simulator and a 2048 fault parallel simulator. The comparison was made by simulating representative circuits such as shift registers, sequences, counters, memory units, arithmetic units, controllers, and a processor. The results indicate that the bit string simulator is faster than the 2048 fault parallel simulator, sometimes by a large amount. [11]

System and method for providing error recovery to an asynchronous logic circuit is presented. The asynchronous logic circuit with error recovery may use temporal redundancy to compare the results of an asynchronous computation and initiate error recovery if necessary. Outputs of the asynchronous logic circuit are compared using a plurality of asynchronous register voters. If an asynchronous register voter detects an inconsistent result, the asynchronous register voter clears itself. A majority of common data outputs from the plurality of asynchronous register voters is provided as an output that is representative of the output of the asynchronous logic circuit. [12]

6. Applied System

In this research, the expert system is implemented to detect the errors in logic combinational logic circuit. The program is written in visual basic programming language. When the user execute this program, he can read the steps of "about operation" option from the help menu as shown in figure 4 and follows these steps which lead him to design new logic circuit or open file contained logic circuit designed previously, then choose the option of expert system to detect the circuit and select "Why" or "How" options later.

To design new logic circuit, the user drag and drops the logic gates (NOT; AND: (2..4 inputs); OR: (2..4 inputs); NAND: (2..4 inputs); NOR: (2..4 inputs); XOR: (2..4 inputs); XNOR: (2..4 inputs)) from the toolbar to the page. The gate appears on the page with its number. For example as shown in figure 5, gate number: 1 is: 2 inputs "AND" gate, gate number: 2 is: 2 inputs "AND" gate, and gate number: 3 is: 2 inputs "OR" gate. Then user connects between the gates on the page by dragging from one gate and dropping to another one. Lines are drawing to represent this connection. The connection is either from input to input or input to output or output to input and closed loop or feedback is not allowed here because the system is applied for combinational logic circuit only, also the connection from output to output is not allowed. Messages are appeared when error case is occurred. Knowing the connection starts from the input or output of the gate is identified from the position of the mouse on the picture of gate. As shown in figure 6, the 1st input of gate number 1 is connected to the 1st input of gate number 2 and the outputs of gate number 1 and gate number 2 are connected to the inputs of gate number 3.

As shown in figure 7, the user can selects "Complete circuit" option from "Design" menu, and text boxes are appeared to define the inputs and outputs of gates and he can't make new connections later. As shown in figure 8, the 1st input of gate number 2 doesn't have text box because it is the same input of the 1st input of gate number 1which is connected to it. Also, there are no text boxes for the inputs of gate number 3 because these inputs are the same of outputs of gates number 1 and number 2 which are connected to them.

The user selects "Complete setting" option from "Design" menu as shown in figure 9, then "Expert System" menu is appeared and he can choose "Detect" option from this menu and he can't change the inputs and outputs of gates unless he choose "New setting" option from "Design" menu which shown in figure 10 and "Expert System" menu becomes invisible and he can sets new inputs and outputs for gates and select "Complete setting" option again from "Design" menu and the "Expert System" menu becomes visible again. In this example, the user set number 1 for the 1st input of gate number 1 and gate number 2, and set number 1 for the 2nd input of gate number 1 and 0 for the 2nd input of gate number 2, the output of gate number 1 is 0, the output of gate number 2 is 0, and the output of gate number 3 is 0.

As shown in figure 11, the user can select "Detect" option from "Expert System" menu, the system checks the inputs and output for each gate according to the truth table of the gate and the message is appeared later to define the error gates which are their outputs are invalid. For this example and as shown in figure 12, there is an error on gate number 1.

Then the user can select "Why" or "How" options from "Expert System" menu to know the reason for any gate which valid or not. "Why" option explain the type of gate and the values of inputs and output and if its output is invalid define the correct output and said there is an error or there is no error when output is valid. While "How" option explain in more details about the gate because it is not explain the value of inputs only but it is describe if this input is the output of another gate and explain this gate and so on until reach the direct inputs which defined by the user.

As shown in figure 13, the user select "Why" option from "Expert System" menu, then as shown in figure 14 he selects the gate number for "Why" option, and as shown in figure 15 for gate number 1 the "Why" option explain that the type of this gate is "AND", the value of 1st input is 1 and the value of 2nd input is 1 and the output is 0 but the correct output is 1 then the error is occurred in this gate.

As shown in figure 16, the user select "How" option from "Expert

System" menu, then as shown in figure 17 he selects the gate number for "How" option, and as shown in figure 18 the explanation here is the same as option "Why" because gate number 1 has direct inputs only (its inputs are not the outputs of another gates).

While figure 19 shown the answer of "Why" question of gate number 3 which is "OR" and its 1st input is 0 and the 2nd input is 0, and the output is 0 which is valid then this gate doesn't have error.

Figure 20 shown the answer of "How" question of gate number 3 which is more detailed than the answer of "Why" question" and explain that gate number 3 which is "OR" gate has two inputs: the 1st input is 0 which is the output of gate number 1 and the 2nd input is 0 which is the output of gate number 2, then the explanation of gate number 3 is followed by explanation of gate number 1 and gate number 2 because the inputs of gate number 3 depending on them.

The user can selects any option of "File" menu as shown in figure 21 to "Save" or "Save As" this logic circuit in file or choose "New" option to make new design or choose "Open" option to load logic circuit which is saved previously in file, or choose "Exit" option to end this program.

7. Conclusions and Future Work

Expert system use human knowledge to solve problems that normally would require human intelligence. These expert systems represent the expertise knowledge as data or rules within the computer. These rules and data can be called upon when needed to solve problems

Expert systems are most valuable to organizations that have a high-level of know-how experience and expertise that cannot be easily transferred to other members. They are designed to carry the intelligence and information found in the intellect of experts and provide this knowledge to other members of the organization for problem-solving purposes.

This research implements the theory of expert system in combinational logic circuit to simulate the human beings experience in this failed. We can use this program to teach the pupils how to design the logic circuit and the truth tables of different logic gates and error detection.

For future work, we can update this program to be used in sequential logic circuit.

7. References

[1] Y. Duana,*, J.S. Edwardsb, "Web-based expert systems: benefits and challenges", M.X. Xuc UK,Available online 8 December 2004

[2] Artificial Intelligence and expert systems, Clever Ace Copyright © 2002-2009.

[3] Answers.com, The world's leading Q & A site, Copyright © 2009 Answers Corporation.

[4] Authorized licensed use limited to: IEEE Xplore. Downloaded on February 2, 2009 at 07:54 from IEEE Xplore. Restrictions apply.

[5] All pages on www.play-hookey.com copyright © 1996, 2000-2009 by Ken Bigelow Copyright © 2009 The Japan Society of Applied Physics Contact Information.

[6] Mark Balch, "Complete Digital Design", The McGraw-Hill Companies, Copyright © 2003.
[7] Page Designed and Written by Wayne Storr. Last updated December 2009 ,
[8] M. Morries Mano, "Digital Design", 3rd edition, California State University, Los Angeles. 2003
[9] © C Neil Bauers, Truth Tables and Logic, 2003
[10] Department of Computer Science Course: CS 3724, Memorial University of Newfoundland.
[11] Michael, R. E. Ph.D. Thesis West Virginia Univ., Morgantown.
[12] Rouse, Gordon, F.; (Us), Thomas, Steven, H.; (Us). Elgersma, Michael, R.; (US). Error Recovery in Asynchronous Combinational Logic Circuits, 23.02.2006.

Table (1) AND, OR, NAND, NOR, XOR, XNOR Truth Table

A	B	A AND B	A OR B	A NAND B	A NOR B	A XOR B	A XNOR B
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

Table (2) Symbolic Representations

Standard Boolean Operators	Boolean Operation Operators
AND	*, &
OR	+, |, #
XOR	⊕, ^
NOT	!, ~, $\overline{A}$

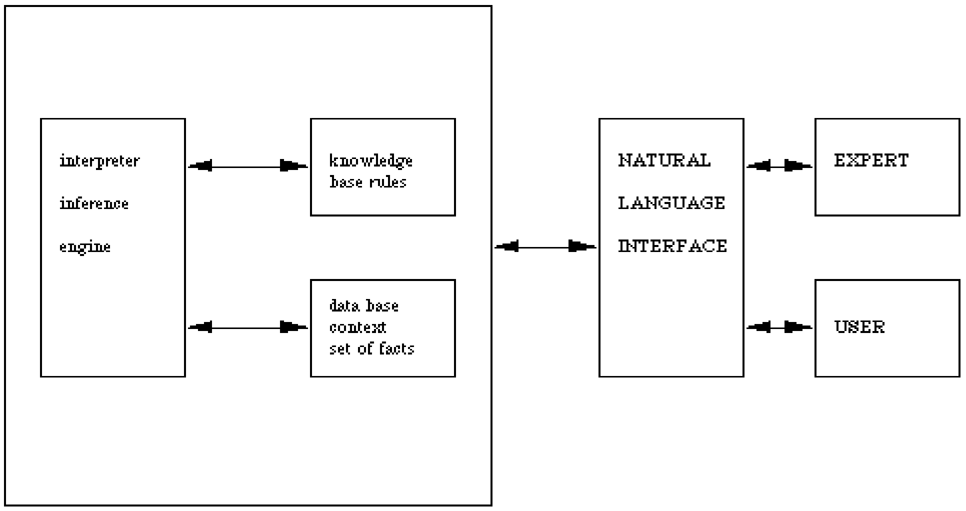


Figure (1) Expert System Structure

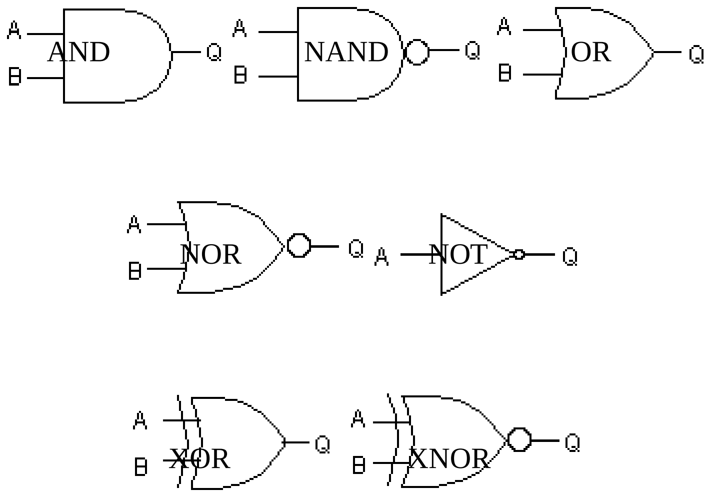


Figure (2) Logic gates

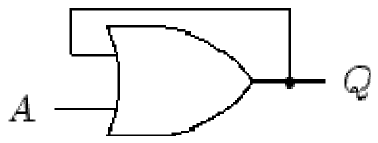
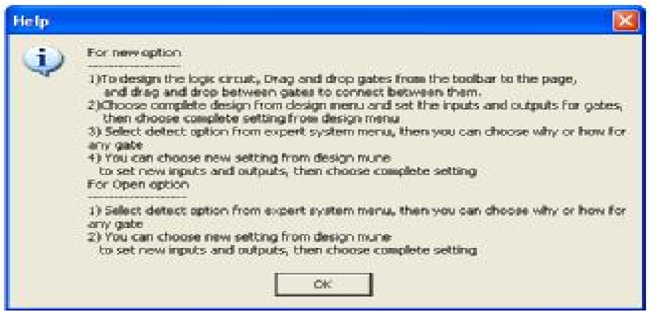


Figure (3) Example of sequential circuit

NoName Logic Circuit File
File Design Help
- - - Expert System to Detect The Errors in Logic Circuit - - -



For new option
----
1)To design the logic circuit, Drag and drop gates from the toolbar to the page,
and drag and drop between gates to connect between them.
2)Choose complete design from design menu and set the inputs and outputs for gates,
then choose complete setting from design menu
3) Select detect option from expert system menu, then you can choose why or how for any gate
4) You can choose new setting from design mune
to set new inputs and outputs, then choose complete setting
For Open option
----
1) Select detect option from expert system menu, then you can choose why or how for any gate
2) You can choose new setting from design mune
to set new inputs and outputs, then choose complete setting

Figure (4) Help – About Operation

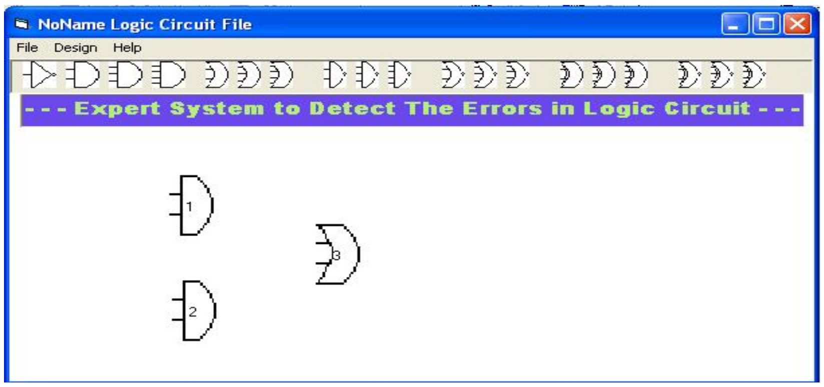

NoName Logic Circuit File
File Design Help
- - - Expert System to Detect The Errors in Logic Circuit - - -
1
2
3

Figure (5) Drag and Drop Gates

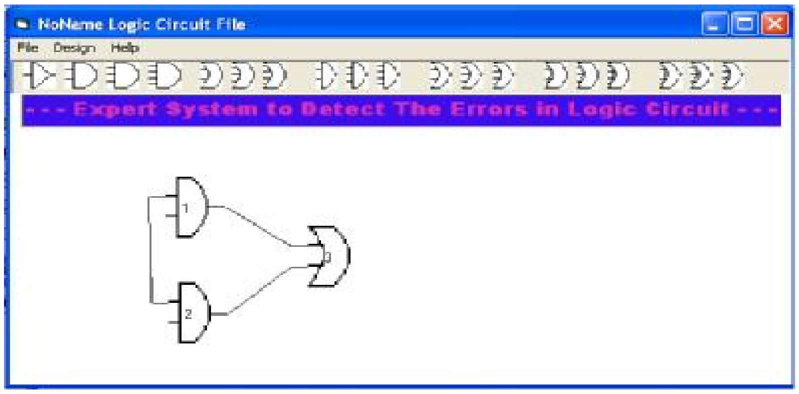

NoName Logic Circuit File
File
Design
Help
- - - Expert System to Detect The Errors in Logic Circuit - - -
1
2

Figure (6) The Connections between gates

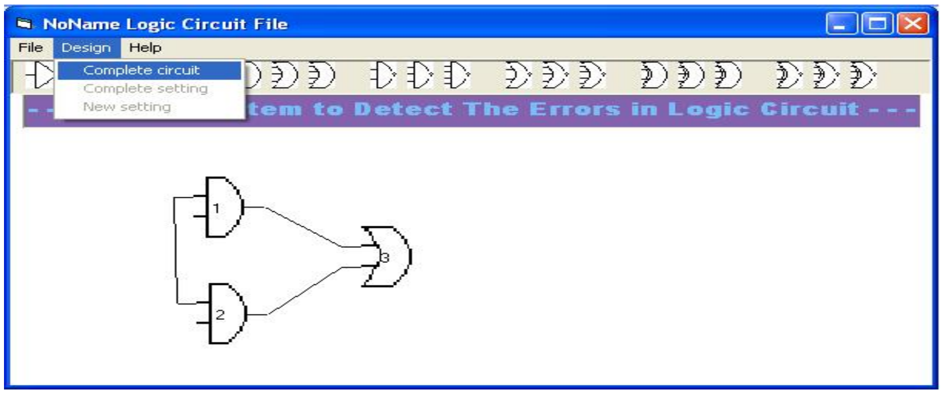

NoName Logic Circuit File
File
Design
Help
Complete circuit
Complete setting
New setting
tem to Detect The Errors in Logic Circuit - - -
1
2
3

Figure (7) Complete Design Option

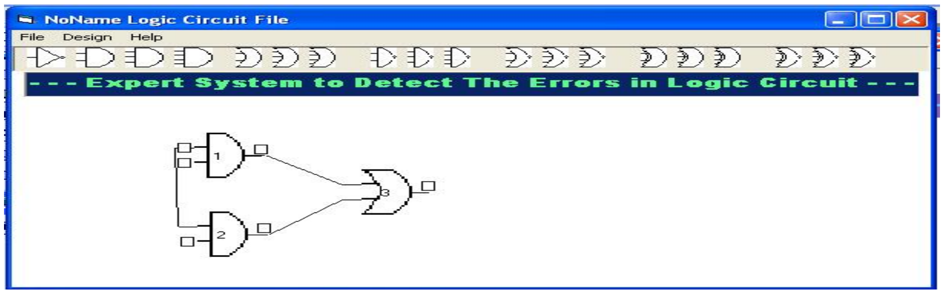

NoName Logic Circuit File
File
Design
Help
- - - Expert System to Detect The Errors in Logic Circuit - - -
1
2
3

Figure (8) Setting Inputs and Outputs of Gates

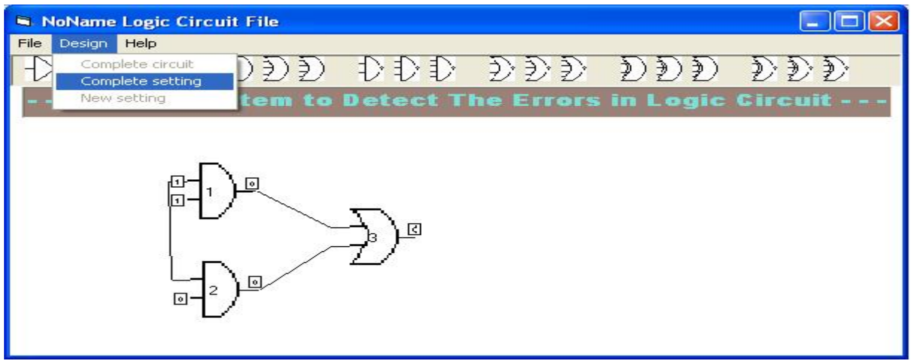

NoName Logic Circuit File
File
Design
Help
Complete circuit
Complete setting
New setting
tem to Detect The Errors in Logic Circuit - - -
1
1
0
1
2
3
0
0

Figure (9) Complete Setting Option

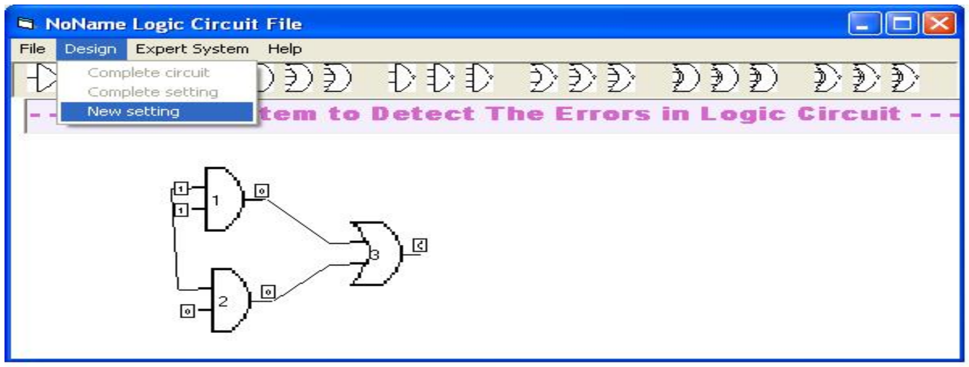


Figure (10) New Setting Option

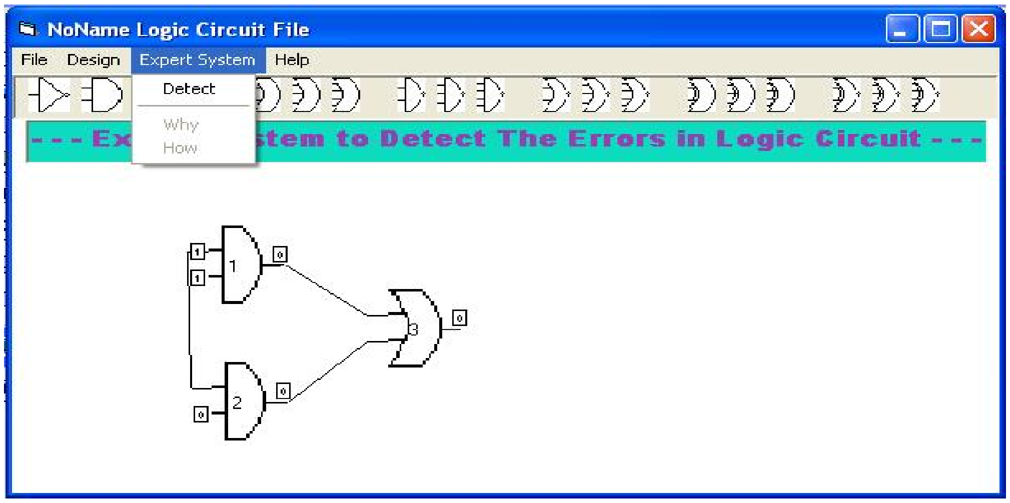
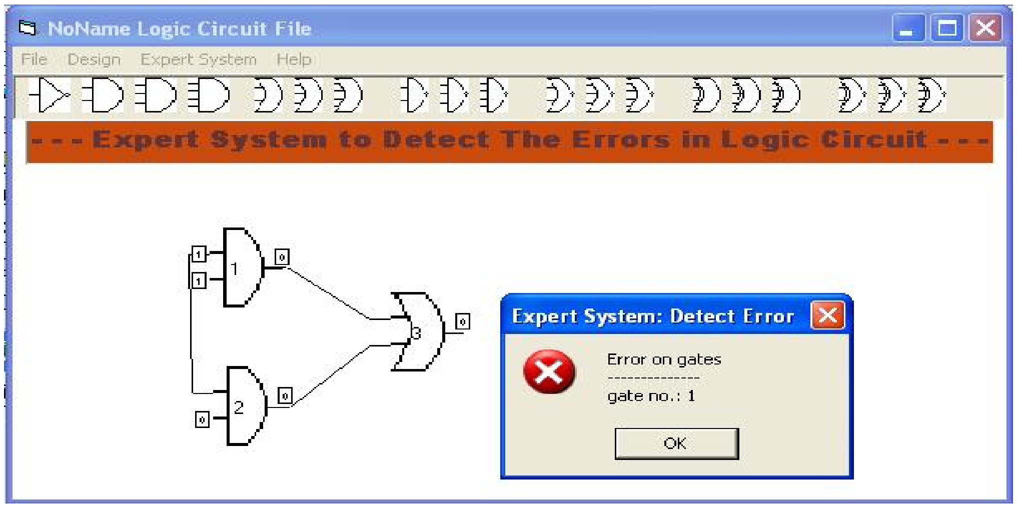

NoName Logic Circuit File
File
Design
Expert System
Help
Detect
Why
How
- - - Expert System to Detect The Errors in Logic Circuit - - -

Figure (11) Detect Option


NoName Logic Circuit File
File
Design
Expert System
Help
- - - Expert System to Detect The Errors in Logic Circuit - - -
Expert System: Detect Error
OK

Error on gates
----
gate no.: 1

Figure (12) Message of Detect Option

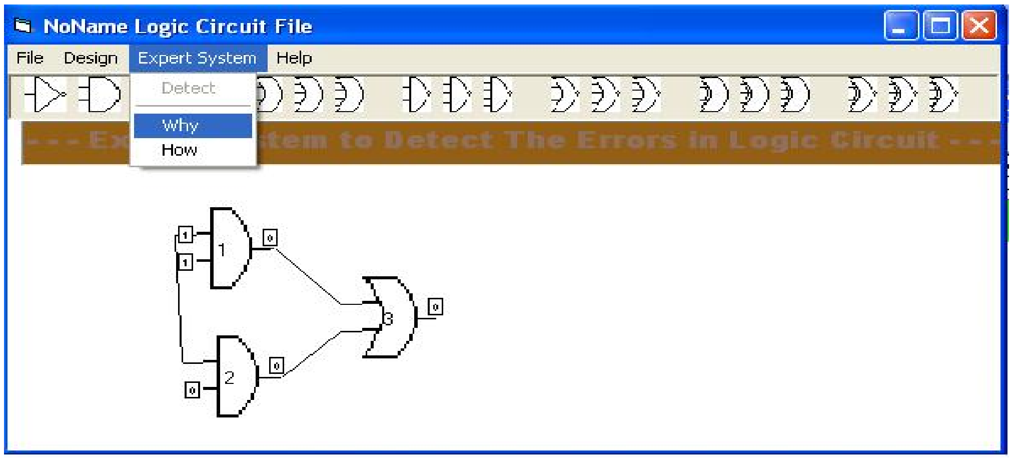
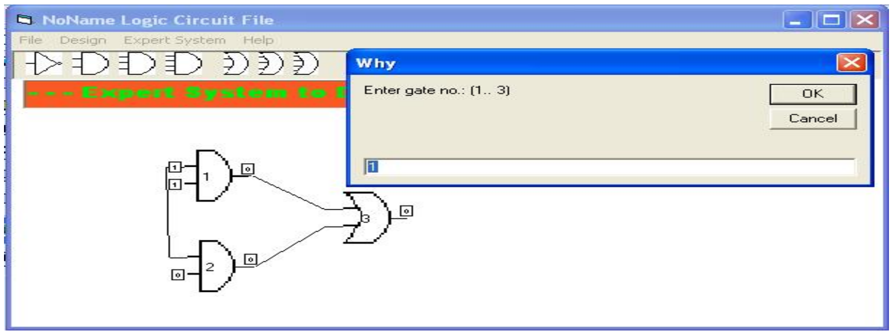

NoName Logic Circuit File
File
Design
Expert System
Help
Detect
Why
How
--- Expert System to Detect The Errors in Logic Circuit ---

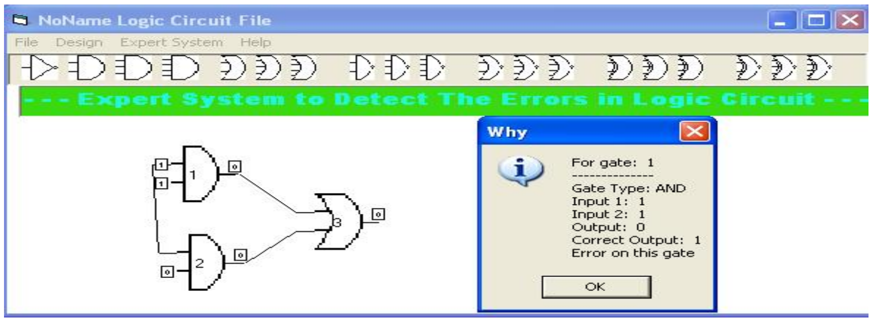
Figure (13) Why Option



Enter gate no.: (1.. 3)

1

Figure (14) Select Gate Number of Why Option


NoName Logic Circuit File
File
Design
Expert System
Help
--- Expert System to Detect The Errors in Logic Circuit ---
Why
OK

For gate: 1

Gate Type: AND
Input 1: 1
Input 2: 1
Output: 0
Correct Output: 1
Error on this gate

Figure (15) Message of Why Question for Gate Number 1

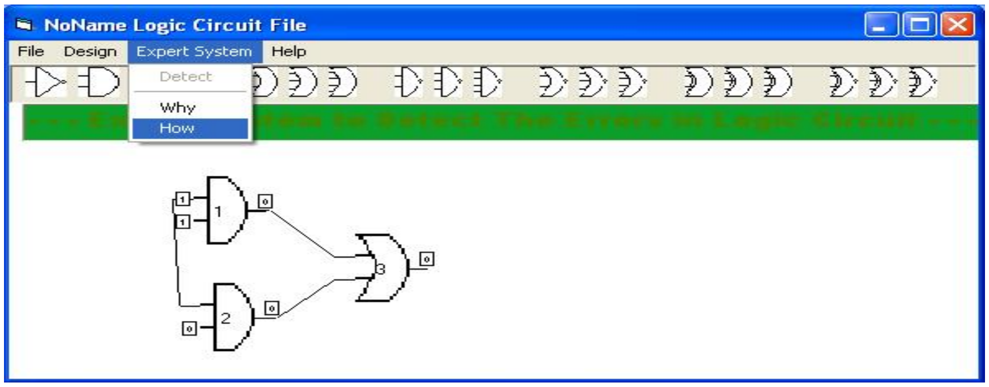
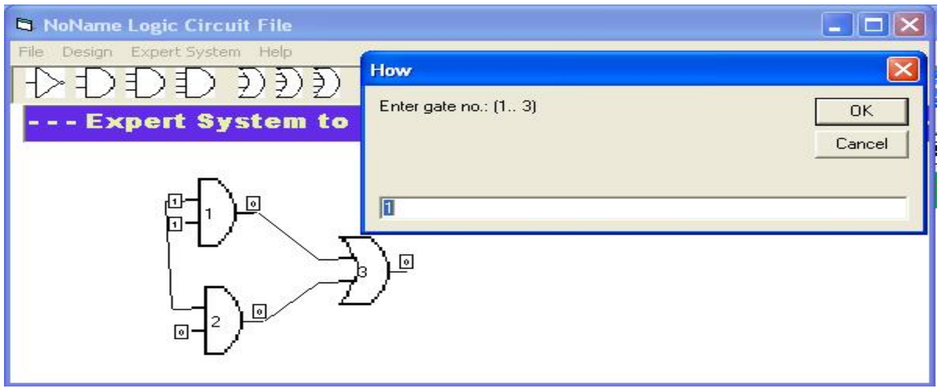


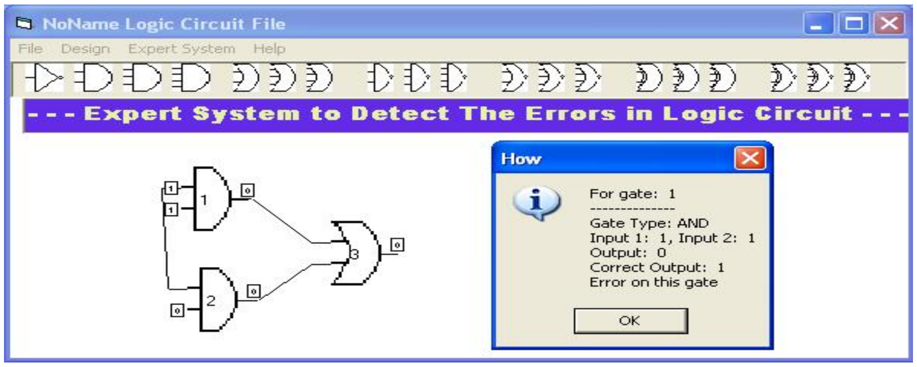
Figure (16) How Option



Enter gate no.: (1.. 3)

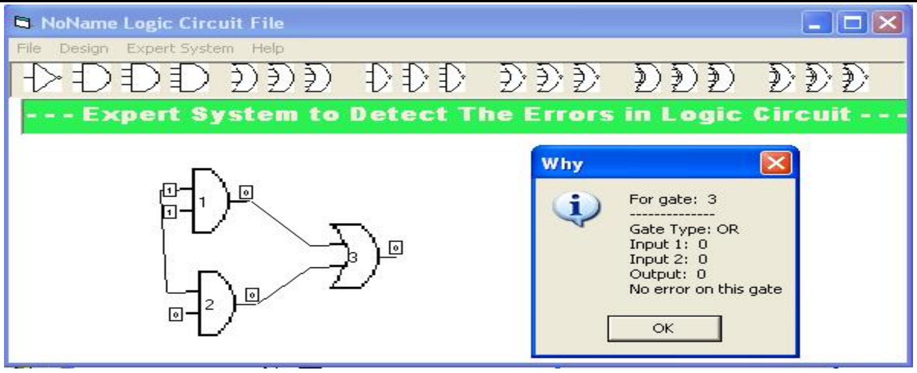
1

Figure (17) Select Gate Number of How Option


NoName Logic Circuit File
File Design Expert System Help
- - - Expert System to Detect The Errors in Logic Circuit - - -
How
OK

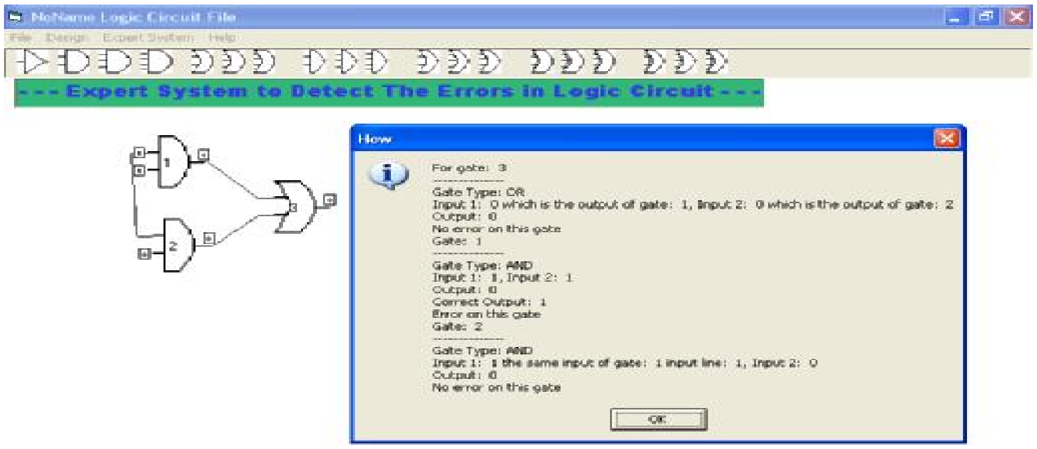
```
For gate:  1
--------------
Gate Type: AND
Input 1:  1, Input 2:  1
Output:  0
Correct Output:  1
Error on this gate
```

Figure (18) Message of How Question for Gate Number 1


NoName Logic Circuit File
File Design Expert System Help
- - - Expert System to Detect The Errors in Logic Circuit - - -
Why
OK

```
For gate:  3
--------------
Gate Type: OR
Input 1:  0
Input 2:  0
Output:  0
No error on this gate
```

Figure (19) Message of Why Question for Gate Number 3


NoName Logic Circuit File
File Design Expert System Help
- - - Expert System to Detect The Errors in Logic Circuit - - -
How
OK

```
For gate:  3
--------------
Gate Type: OR
Input 1:  0 which is the output of gate:  1, Input 2:  0 which is the output of gate:  2
Output:  0
No error on this gate
Gate:  1
--------------
Gate Type: AND
Input 1:  1, Input 2:  1
Output:  0
Correct Output:  1
Error on this gate
Gate:  2
--------------
Gate Type: AND
Input 1:  1 the same input of gate:  1 input line:  1, Input 2:  0
Output:  0
No error on this gate
```

Figure (20) Message of How Question for Gate Number 3

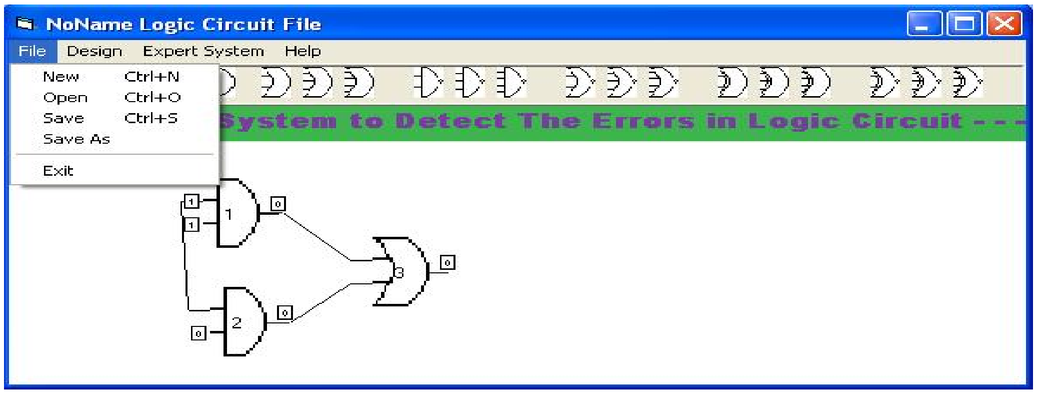

NoName Logic Circuit File
File Design Expert System Help
New Ctrl+N
Open Ctrl+O
Save Ctrl+S
Save As
Exit
System to Detect The Errors in Logic Circuit - - -

Figure (21) Options of File Menu