MODELING OF SOFTWARE BASED COMBINATIONAL LOGIC TRAINER

Nnaji, G. A., Iloh, J. I. and Nwabueze, C. A.
Department of Electrical/Electronic Engineering, Chukwuemeka Odumegwu Ojukwu University, Uli, Anambra State.

ABSTRACT

Intelligent systems are nature-inspired, mathematically sound and computationally intensive problem solving tools and methodologies that have become extremely important for advancing the current trends in information technology. Artificial intelligent systems currently utilize computers to emulate various faculties of human intelligence and biological metaphors. They use a combination of symbolic and sub-symbolic systems capable of evolving human cognitive skills and intelligence, not just systems capable of doing things humans do not do well. In this paper, an intelligent system program (software) developed with Microsoft Visual Basic 6.0 evaluates student's background knowledge on combinational logic design, simulates a customized training module based on the student's previous knowledge of the course module and tests the student's grasp of the course through feedback. The result will be useful to students for better understanding of logic design, for lecturers and users of combinational logic for improved system design.

1.0 INTRODUCTION

Computers have been employed to achieve a variety of educational goals since the early 1960s. Some of these goals include automated testing, routine drill and practice tasks that had been mechanized with earlier technologies. Other computer assisted instructional programs engage the students in challenging and entertaining reasoning tasks and capitalize on multimedia capabilities to present information. Computer based instruction has successfully penetrated all educational and training markets, homes, schools, universities, business and government, but remains far from the required educational experience. In the early 1970s a few researchers defined new and ambitious goals for computer-based instruction. These adopted the human tutor as their educational model and sought to apply artificial intelligence techniques to realize models in "intelligent" computer-based instruction (Cohen et al, 2002; Larkin and Chabay, 2005).

The goal of intelligent tutoring systems (ITSs) would be to engage the students in sustained reasoning activity and to interact with the student based on a deep understanding of the students' behavior. If such systems realize even half the impact of human tutors, the payoff for society promised to be substantial. In the first intelligent training programs, students were expected to communicate through a sequence of natural language questions and answers. The tutor could both ask and answer questions and keep track of the ongoing dialogue structure. The tutor was constructed around a semantic network model of domain knowledge. Such network models of conceptual knowledge were revolutionizing the understanding of question answering and inferential reasoning in cognitive science and remain the modal model of conceptual knowledge today (Anderson, 2003; Collins and Loftus, 2005).

The interest in Intelligent Tutoring Systems (ITS) emerged from using various instructional materials – textbooks and published papers for teaching introductory Software Engineering to non-engineering students. The most recent incarnation of these efforts, Integrated Introduction to Computing, drew upon the literature and experience of a number of specialists to create a course which was innovative, engaging, and effective (Weinshank et al, 2005; Steven and Collins, 2007). Comparing student performance and instructor ratings for the course with its predecessor which indicates that various goals were met: students in the course performed better on standardized tests and on a range of programming problems than did students in the previous course. Although the average performance and student ratings of the course improved over its predecessor, the dispersion of these measures is as great as or greater than it had been in the past. This implies that there are problems reaching students who are well below the mean, losing a great number of the very intelligent students who most need these materials to be successful in an information-rich world. In addition, an increasing number of students bring richer backgrounds to the course than did previous students. These students do not find the course challenging and consequently believe that Computer/Electronic Engineering is uninteresting. Hence, a large number of students who might have had potential to become successful Electronic/Software Engineering majors are lost (Mayor, 2008; Boolos and Jeffery, 2009). With these factors in mind, a better architecture of the course with more modular, finer-grained curriculum could have the potential to address many of these concerns. There is therefore need for instructional technology to leverage faculty resources where instructional technology is the theory and practice of design, development, utilization, management and evaluation of processes and resources for learning are enhanced.

2.0 COMBINATIONAL LOGIC

Combinational logic (sometimes incorrectly referred to as combinatorial logic) is a type of digital logic which is implemented by Boolean circuits, where the output is a pure function of the present input only. This is in contrast to sequential logic, in which the output depends not only on the present input but also on the history of the input. In other words, sequential logic has memory while combinational logic does not (Anderson and Corbett, 2003).

Practical computer circuits normally contain a mixture of combinational and sequential logic. For example, the part of an Arithmetic Logic Unit, or ALU, that does mathematical calculations is constructed using combinational logic. Combinational logic is used for building circuits where certain outputs/inputs are desired. The construction of combinational logic is generally done using one of two methods: a sum of products, or a product of sums.

2.1 Combinational Logic Expression

- Combinational logic function value is a combination of function arguments.
- A logic gate implements a particular logic function.
- Both *specification* (logic equations) and *implementation* (logic gate networks) are written in Boolean logic

Boolean Algebra Terminology

Function:

$$f = a'b + ab' \qquad (1)$$

where a is a variable; a and a' are literals and ab' is a term.

Logic Gates

Logic gates are circuits that implement the Boolean logic operations. Gates form the basic building blocks for all digital circuits and systems. Symbols are used to describe the input and output signals of gates. Boolean algebra helps to precisely express input/output relationships of logic gates. A truth table shows a gate's output condition for all possible input conditions. A timing diagram shows same information but in a graphical form. Thus, it can be said that the tools for understanding digital circuits are *symbols, truth tables, Boolean expressions,* etc.

Boolean Expressions and Equations

A Boolean expression is an important tool for understanding, analyzing and designing digital systems and circuits. A Boolean expression gives the relationship between signals, for example, A + B, A.B, A.B.C, X + Y, etc. The Boolean equation gives the input/output relationship for any logic gate or digital system. For example, Y = A + B, X = A.B.

Truth Table

A Truth table shows a gate's output condition for all possible input conditions. If a logic gate has one input signal, only two input conditions are possible. If there are two input signals, four input conditions are possible. In general, for n inputs, 2" input conditions are possible. A sample truth table for a system with two inputs A & B and an output X is shown in Table.

Table 1: A Sample Truth Table.

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Standard Logic Gates

Gates are circuits that perform logic operations. Examples include:
AND, OR, NOT, NAND, NOR, EXCLUSIVE-OR and EXCLUSIVE-NOR gates.

NAND Gate: The term NAND is a contraction of NOT-AND and implies an AND function with complemented (inverted) output. In this gate, output is *low* only when all inputs are *high*. A standard logic symbol for a 2-input NAND gate and the equivalent AND gate followed by an

INVERTER are shown in Figure 1. The standard symbol for the NAND gate is the same as the AND gate symbol except for small circle on its output. The small circle denotes the inverse operation. Thus the NAND gate operates like an AND gate followed by an INVERTER, so that the circuits shown in (a) and *(b)* are equivalent.
The operation of this gate can be described as below

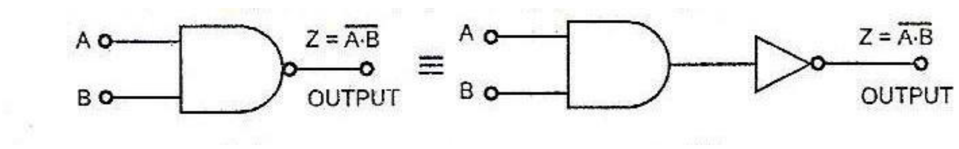


Figure1: The NAND Gate

$$Z' = A.B.C....N \qquad (2)$$

Table 2: Truth Table for 2-input NAND Gate.

Inputs		**Output Z**
A	**B**	**Z**
0	0	1
0	1	1
1	0	1
1	1	0

The logical operation *Truth Table for NAND Gate* represented by eqn. (2) is shown in Table 2. The NAND gate output is the exact inverse of the AND gate for all possible input conditions. The NAND gate output goes *low* only when all the inputs are *high*, while the AND gate output goes *high* only when all the inputs are *high.*

The logic equation for NAND gate is given as:

$$Z = A.B \qquad (3)$$

and is read as 'Z' equals NOT (A AND B).

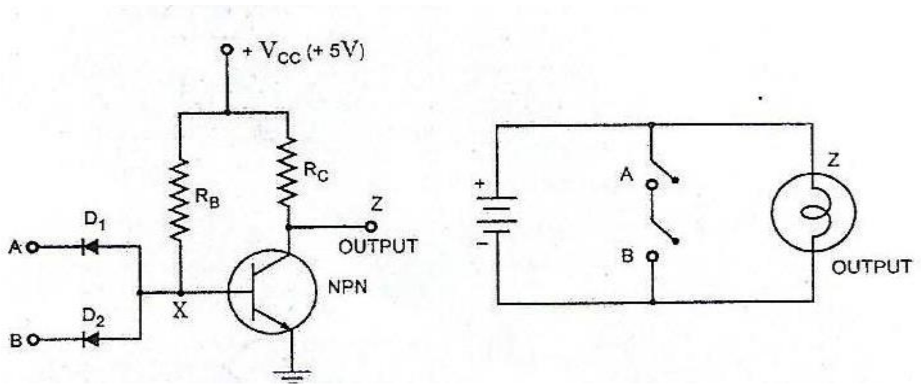


Figure 2: (a) Circuit for NAND Gate (b) Electrical Analog of NAND Gate

The electrical circuit and electrical analog of NAND gate are shown in this Figure 2 (a) and (b) respectively. Here a diode AND circuit is connected to a transistor NOT circuit that gives the NAND circuit. It can be seen from figure 2(a) that the point X would be driven to ground when either of the diodes D_1 or D_2 or both D_1 and D_2 conduct. Under such conditions, the transistor is cutoff and therefore, the output goes *high*. Output is *low* only when both input voltages are *high* (at + V) so that X is at + Vcc and transistor operates in saturation. From Figure 2(b), it is evident that the lamp does not glow (i.e. the output is zero) only when both of the switches A and B are closed (i.e. both the inputs are high). The lamp glows (i.e. the output is high) when either or both of the switches are open (either or both of the inputs are low). These are depictions used in combinational circuits necessary for operational training.

3.0 SYSTEM DESIGN

3.1 Design Flowchart

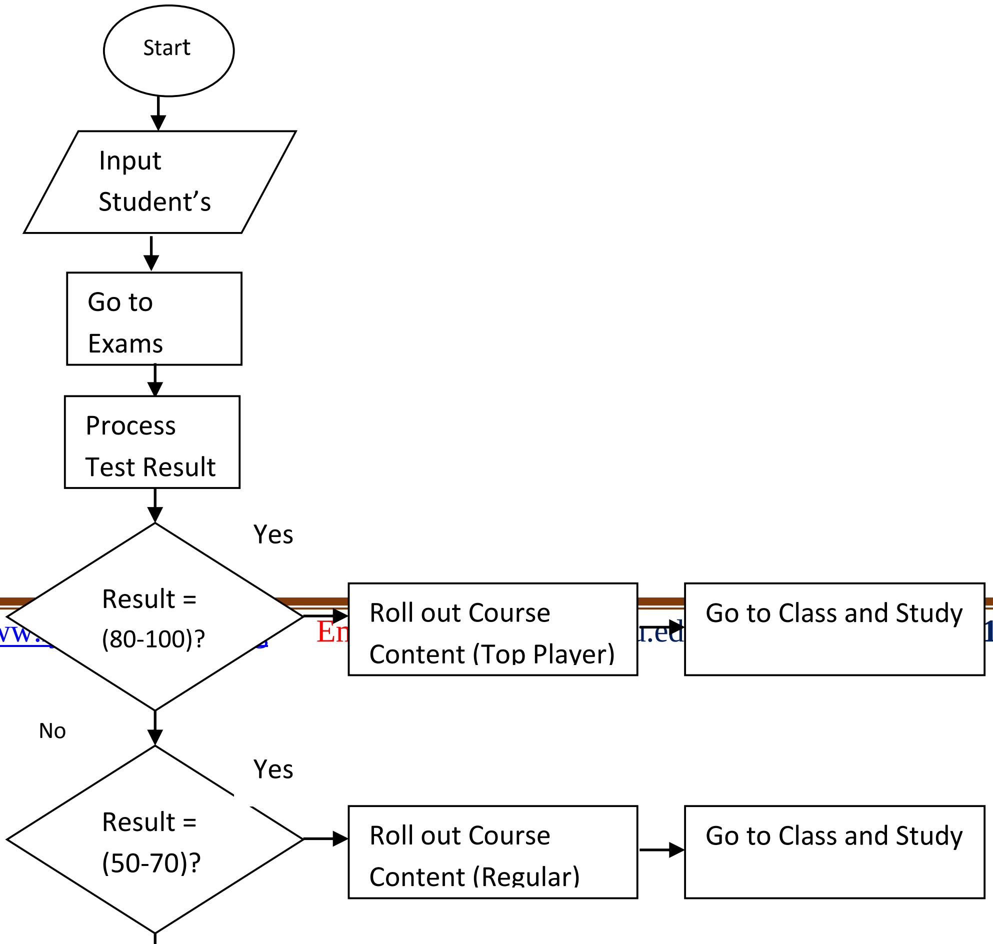


Figure 3: System Design Flowchart

3.2 Program Design Pseudocode:

```
Start
Input Student's Name
Load Test Questions
Student Takes Test Questions
Process the student's performance in the test
If result score is between 80-100
Load/Display Course content for (Category Top Player.)
Conduct the training process for the student.
If result score is between 50-70
Load/Display Course content for (Category Regular.)
Conduct the training process for the student.
If result score is between 30-40
Load/Display Course content for (Category Starter.)
Conduct the training process for the student.
If result score is between 10-20
```

Load/Display Course content for (Category Green Horn.)
Conduct the training process for the student.
Evaluate performance of student after training.
End.

3.3 Database Design

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity. The term database design can be applied to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS) (Mayer, 2008).

The process of developing a database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must determine the relationships between the different data elements and superimpose a logical structure upon the data on the basis of these relationships (see figure 3).

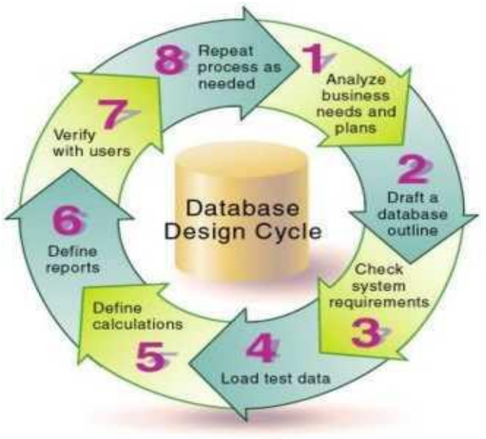

Database Design Cycle
1 Analyze business needs and plans
2 Draft a database outline
3 Check system requirements
4 Load test data
5 Define calculations
6 Define reports
7 Verify with users
8 Repeat process as needed

Figure 4: Database Design Process.

3.4 Project Block Diagram

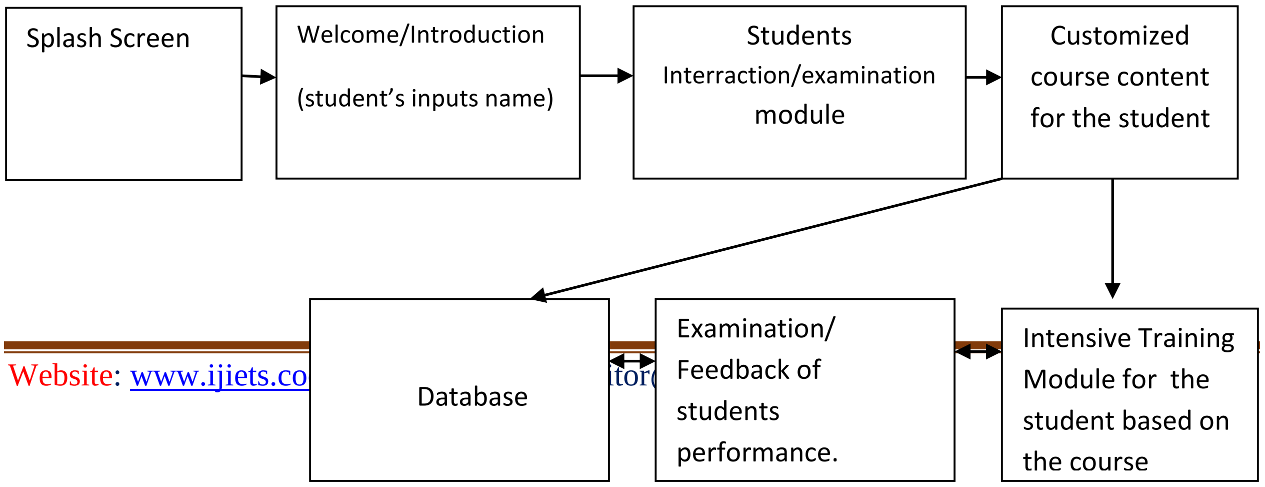

Splash Screen
Welcome/Introduction (student's inputs name)
Students Interraction/examination module
Customized course content for the student
Database
Examination/ Feedback of students performance.
Intensive Training Module for the student based on the course

Figure 5: Project Block Diagram.

The project block diagram is shown in figure 5.

4.0 IMPLEMENTATION RESULTS AND ANALYSIS OF THE SOFTWARE

The software was tested on the four (4) different categories of persons described by the software according to grades (80-100, 50-70, 30-40 and 0-20 which are categorized as green horns, starters, regular and top players respectively) for one hundred (100) year two students in Electrical Engineering Department, Imo State University (IMSU), Federal Polytechnic Nekede, Owerri and Imo State Polytechnic, Umuagwo, Imo State. This evaluation was done at a period close to their examination. This period was chosen to be sure that the students must have gone through the course very well. Table 3 shows the number of student base on the categories of scores for IMSU, while figure 6 shows a pie chart with percentage of student scores according to their categories. After inputting the names of the individual, it directs the individual to a test environment which shows the capability of the individual to suggest the course or subject to start from. This was done to aid the individual based on their knowledge of the course.

Table 3: Summary of scores and completion time of individual students in IMSU

STUDENT NO	TIME FOR COMPLETING TEST (S)	SCORE	SUGGESTED MODULES
1	1.1	30	Starter
2	7.81	50	Regular
3	4.53	70	Regular
4	2.97	40	Regular
5	3.58	50	Regular
6	4.82	80	Top player
7	4.31	80	Top player
8	6.99	50	Regular
9	6.75	40	Regular
10	0.07	10	Green horn
11	0.79	10	Green horn
12	4.61	40	Regular
13	7.78	80	Top player
14	8.17	70	Top player
15	6.31	90	Top player

16	3.65	20	Starter
17	8.87	60	Regular
18	2.51	40	Starter
19	0.66	30	Starter
20	7.27	30	Starter
21	7.67	80	Top player
22	8.98	90	Top player
23	7.67	100	Top player
24	9.47	70	Top player
25	5.36	20	Starter
26	9.6	30	Starter
27	9.78	40	Starter
28	5.22	10	Green horn
29	8.45	40	Regular
30	8.98	50	Regular
31	9.31	50	Regular
32	4.58	90	Top player
33	7.59	90	Top player
34	9.39	10	Green horn
35	8.11	20	Green horn
36	9.3	10	Green horn
37	4.47	30	Starter
38	8.34	90	Top player
39	9.88	60	Regular
40	3.7	60	Regular
41	1.71	60	Regular
42	8.23	10	Green horn
43	5.87	90	Top player
44	9.62	70	Regular
45	4.89	50	Regular
46	2.54	30	Starter
47	6.78	40	Starter
48	9.14	40	Regular
49	6.17	50	Regular
50	3.22	20	Green horn
51	4.91	30	Starter
52	4.07	60	Regular
53	0.84	10	Green horn

54	5.92	30	Starter
55	8.79	70	Top player
56	5.53	40	Regular
57	2.99	30	Starter
58	8.97	70	Top player
59	7.94	80	Top player
60	7.76	50	Regular
61	5.99	10	Green horn
62	5.57	30	Starter
63	3	70	Top player
64	5.48	70	Regular
65	9.92	20	Green horn
66	4.83	40	Starter
67	6.85	10	Green horn
68	4.8	0	Green horn
69	7.47	20	Green horn
70	2.11	30	Starter
71	2.48	40	Starter
72	0.98	0	Green horn
73	7.09	60	Regular
74	8.56	80	Top player
75	7.89	100	Top player
76	7.43	90	Top player
77	1.05	60	Regular
78	5.21	90	Top player
79	8.47	10	Green horn
80	8.44	80	Top player
81	3.78	80	Top player
82	2.72	60	Regular
83	1.25	50	Regular
84	6.91	40	Starter
85	5.55	10	Green horn
86	0.08	0	Green horn
87	4.16	30	Starter
88	4.39	10	Green horn
89	7.84	10	Green horn
90	8.3	70	Top player
91	5.9	90	Top player

92	1.42	80	Top player
93	5.93	100	Top player
94	7.26	70	Top player
95	4.28	90	Top player
96	2.11	100	Top player
97	2.65	20	Green horn
98	9.93	30	Starter
99	4.81	10	Green horn
100	8.28	60	Regular

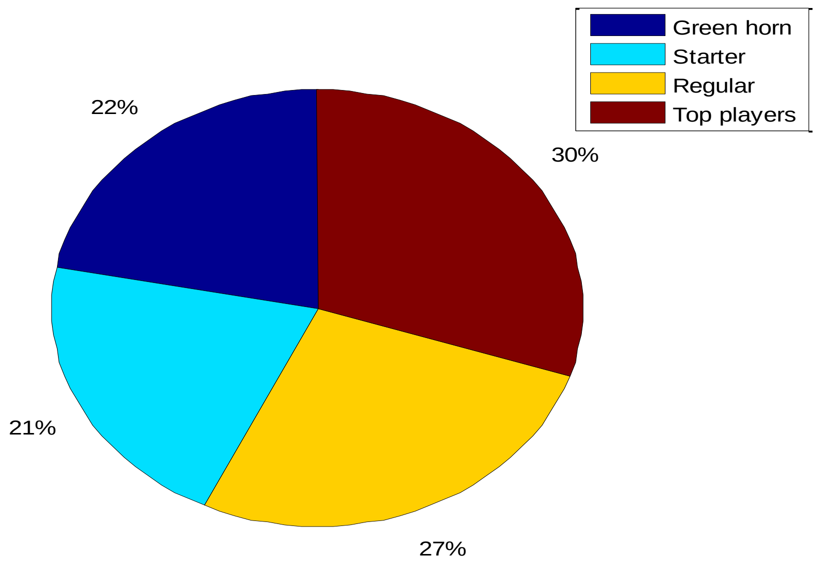
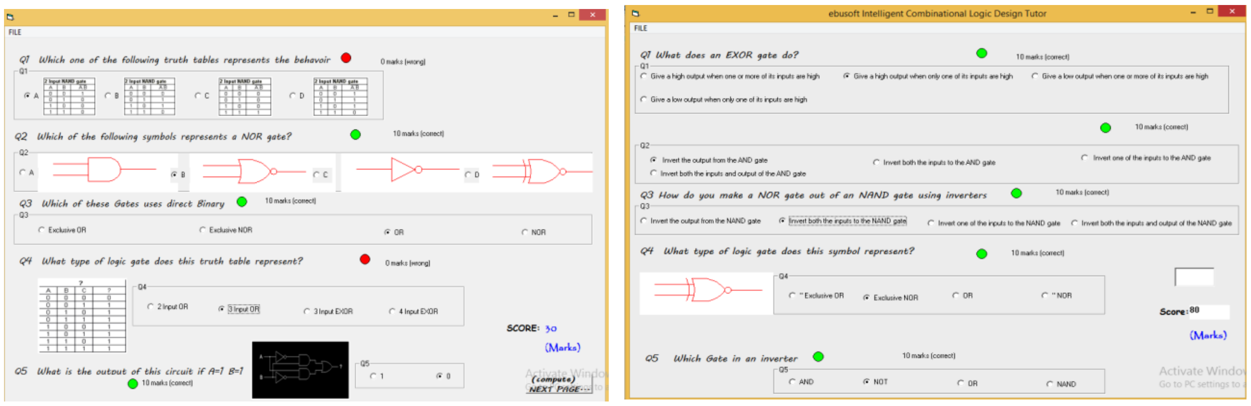


Figure 6: Percentage Scores of categories of students in IMSU

4.1 Software implementation result and analysis for individuals within 80-100

The first group considered has score of 80 - 100 as shown in figures 7 and 8 which gives account of the software performance for an excellent student.



FILE

Q1 Which one of the following truth tables represents the behavoir 🟥 0 marks (wrong)

Q1: [✓ A] B C D

2 Input NAND gate (A)

A	B	AB
0	0	1
0	1	0
1	0	0
1	1	0

2 Input NAND gate (B)

A	B	AB
0	0	0
0	1	1
1	0	1
1	1	0

2 Input NAND gate (C)

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

2 Input NAND gate (D)

A	B	AB
0	0	1
0	1	1
1	0	1
1	1	0

Q2 Which of the following symbols represents a NOR gate? 🟩 10 marks (correct)

Q2: A [✓ B] C D

Q3 Which of these Gates uses direct Binary 🟩 10 marks (correct)

Q3: Exclusive OR Exclusive NOR [✓ OR] NOR

Q4 What type of logic gate does this truth table represent? 🟥 0 marks (wrong)

A	B	C	?
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Q4: 2 Input OR [✓ 3 Input OR] 3 Input EXOR 4 Input EXOR

SCORE: 30 (Marks)

Q5 What is the output of this circuit if A=1 B=1 🟩 10 marks (correct)

Q5: 1 [✓ 0]

(compute) NEXT PAGE---

ebusoft Intelligent Combinational Logic Design Tutor

FILE

Q1 What does an EXOR gate do? 🟩 10 marks (correct)

Q1: Give a high output when one or more of its inputs are high [✓ Give a high output when only one of its inputs are high] Give a low output when one or more of its inputs are high Give a low output when only one of its inputs are high

🟩 10 marks (correct)

Q2: [✓ Invert the output from the AND gate] Invert both the inputs to the AND gate Invert one of the inputs to the AND gate Invert both the inputs and output of the AND gate

Q3 How do you make a NOR gate out of an NAND gate using inverters 🟩 10 marks (correct)

Q3: Invert the output from the NAND gate [✓ Invert both the inputs to the NAND gate] Invert one of the inputs to the NAND gate Invert both the inputs and output of the NAND gate

Q4 What type of logic gate does this symbol represent? 🟩 10 marks (correct)

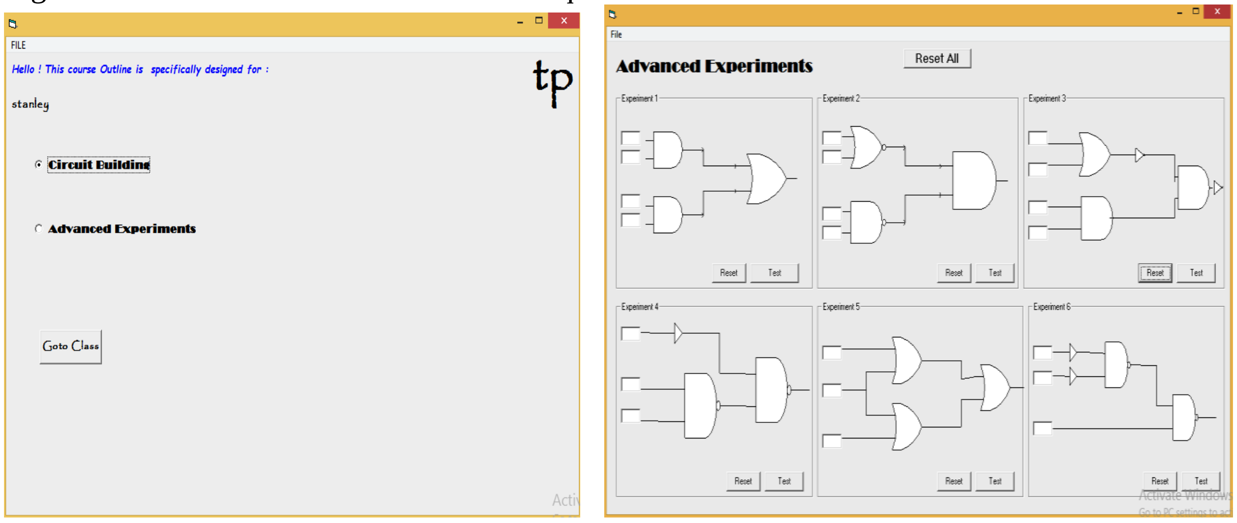
Q4: "Exclusive OR [✓ Exclusive NOR] OR "NOR

Score: 80 (Marks)

Q5 Which Gate in an inverter 🟩 10 marks (correct)

Q5: AND [✓ NOT] OR NAND

Figure 7: First and Second Parts of the test question and the score



FILE

Hello ! This course Outline is specifically designed for :

stanley

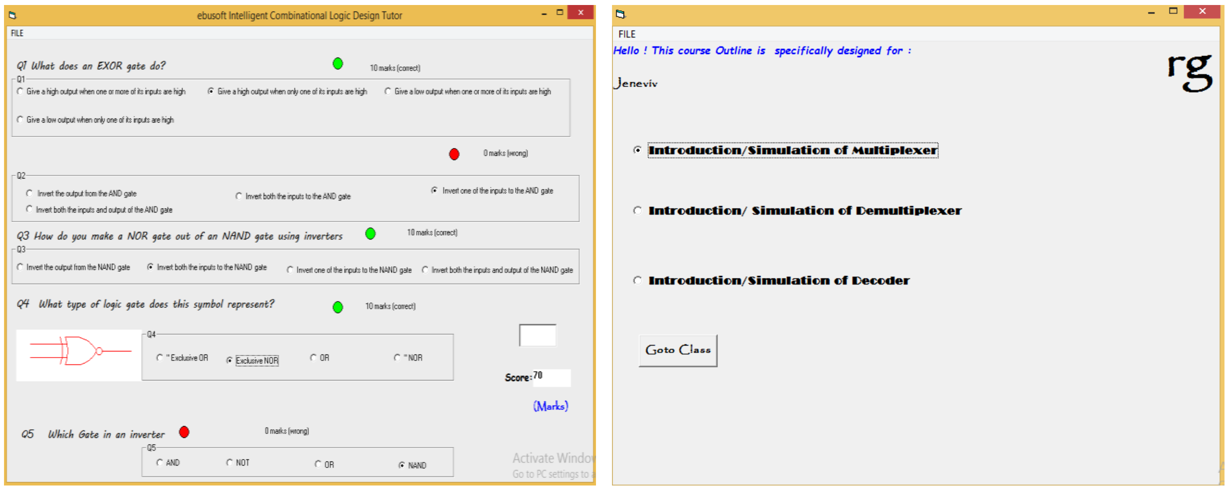
[✓ Circuit Building] Advanced Experiments

Figure 8: The outline designed for scores between 80-100 and Advanced Experiment Scores.

Due to their excellent performance, it is assumed that they should be able to know and understand the aspects which have been treated for those below 80. Figure 7 gave account of the general test question which will be given to individuals, of which Figure 8 gave the total score for the test question and the course outline for the tutor for individuals within 80-100 was shown with introduction of the course with an advanced experiment.

4.2 Software Implementation Result and Analysis for Individuals within 50-70

The next individual or student considered is one whose scores are from 50 - 70 as shown in figure 9.


ebusoft Intelligent Combinational Logic Design Tutor

FILE

Q1 What does an EXOR gate do? ● 10 marks (correct)

Q1: ( ) Give a high output when one or more of its inputs are high [✓ Give a high output when only one of its inputs are high] ( ) Give a low output when one or more of its inputs are high ( ) Give a low output when only one of its inputs are high

● 0 marks (wrong)

Q2: ( ) Invert the output from the AND gate ( ) Invert both the inputs to the AND gate [✓ Invert one of the inputs to the AND gate] ( ) Invert both the inputs and output of the AND gate

Q3 How do you make a NOR gate out of an NAND gate using inverters ● 10 marks (correct)

Q3: ( ) Invert the output from the NAND gate [✓ Invert both the inputs to the NAND gate] ( ) Invert one of the inputs to the NAND gate ( ) Invert both the inputs and output of the NAND gate

Q4 What type of logic gate does this symbol represent? ● 10 marks (correct)

Q4: ( ) "Exclusive OR [✓ Exclusive NOR] ( ) OR ( ) "NOR

Score: 70

(Marks)

Q5 Which Gate in an inverter ● 0 marks (wrong)

Q5: ( ) AND ( ) NOT ( ) OR [✓ NAND]

Activate Windows
Go to PC settings to

FILE

Hello ! This course Outline is specifically designed for :

Jeneviv

rg

[✓ Introduction/Simulation of Multiplexer]

( ) Introduction/ Simulation of Demultiplexer

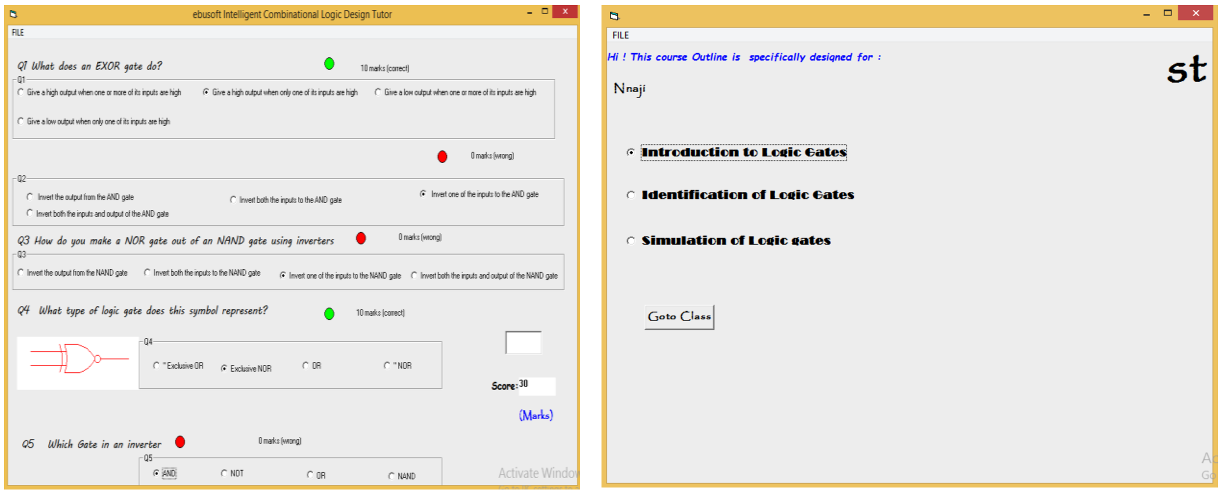
( ) Introduction/Simulation of Decoder

Figure 9: Second part of the test question and the total score and outline for scores within 50-70.

Base on grade, individuals who scored within 50 -70 were advised to first study the introduction to the simulation of multiplexer, de-multiplexer and decoder as shown in figure 9 which shows the total score of the individuals with tutor environment starting with the introduction and simulation of multiplexers.

4.3 Software Implementation Result and Analysis For Individuals within 30-40

The next individual or student considered is one whose score is 30 as shown in figure 10.


ebusoft Intelligent Combinational Logic Design Tutor

FILE

Q1 What does an EXOR gate do? ● 10 marks (correct)

Q1: ( ) Give a high output when one or more of its inputs are high [✓ Give a high output when only one of its inputs are high] ( ) Give a low output when one or more of its inputs are high ( ) Give a low output when only one of its inputs are high

● 0 marks (wrong)

Q2: ( ) Invert the output from the AND gate ( ) Invert both the inputs to the AND gate [✓ Invert one of the inputs to the AND gate] ( ) Invert both the inputs and output of the AND gate

Q3 How do you make a NOR gate out of an NAND gate using inverters ● 0 marks (wrong)

Q3: ( ) Invert the output from the NAND gate ( ) Invert both the inputs to the NAND gate [✓ Invert one of the inputs to the NAND gate] ( ) Invert both the inputs and output of the NAND gate

Q4 What type of logic gate does this symbol represent? ● 10 marks (correct)

Q4: ( ) "Exclusive OR [✓ Exclusive NOR] ( ) OR ( ) "NOR

Score: 30

(Marks)

Q5 Which Gate in an inverter ● 0 marks (wrong)

Q5: [✓ AND] ( ) NOT ( ) OR ( ) NAND

Activate Windows

FILE

Hi ! This course Outline is specifically designed for :

Nnaji

st

[✓ Introduction to Logic Gates]

( ) Identification of Logic Gates

( ) Simulation of Logic gates

Figure 10: Third part of the test question and design for total score of 30 – 40.

Base on grade, individuals who scored within 30 - 40 were advised to first study logic gates as shown in the course outline with the tutor environment starting with the Introduction of logic gates.

The software implementation done at IMSU shows that only 30% of the individuals understood the course very well, 27% are at the average, 21% have limited knowledge of the course and 22% have no knowledge of the course as shown in Tables 4. This shows that greater numbers of the students are at risk to write the exam and will incur low class performance which shows that the software will be of greater advantage to them. Figures 11 and 12 shows the spread of individual student scores and completion time for each student.

Table 4: General result for software tutor test conducted in IMSU

Category	**Number of students**	**Percentage (%)**
Green horns	22	22
Starters	21	21
Regular	27	27
Top players	30	30
Total	**100**	**100**

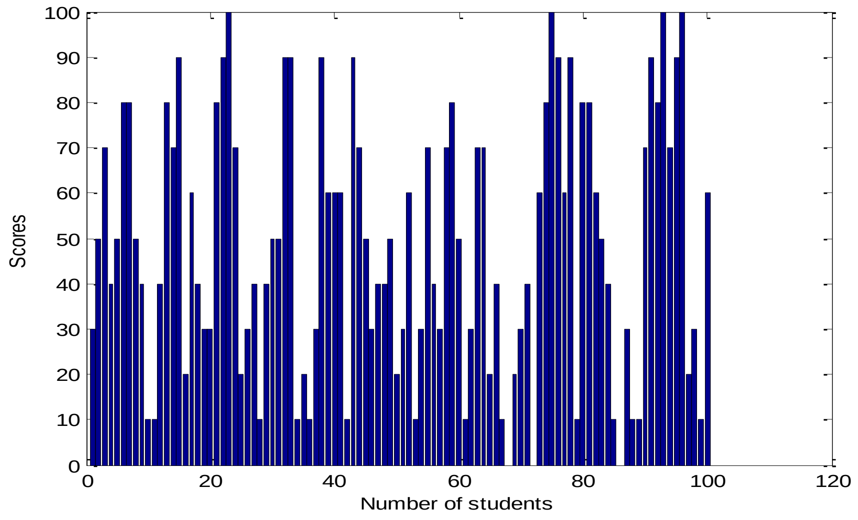


Figure 11: Scores of individual students in IMSU

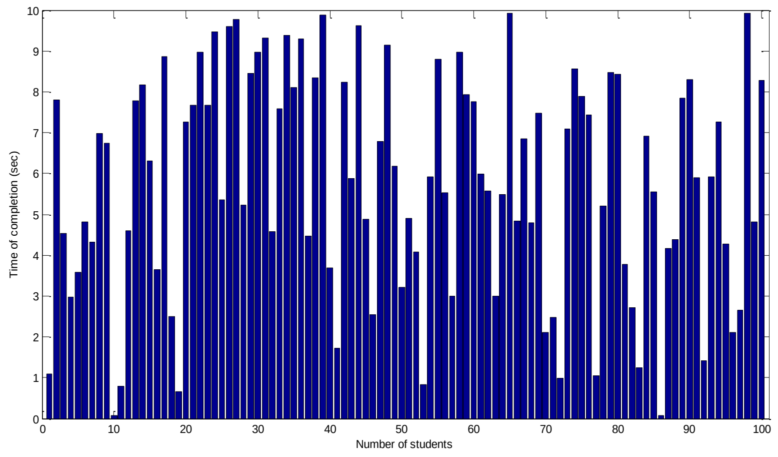


Figure 12: Completion time of individual students in IMSU

CONCLUSION

Personal human tutors provide a highly efficient learning environment and have been estimated to increase mean achievement outcomes by as much as two standard deviations. The goal of intelligent tutoring systems (ITSs) would be to engage the students in sustained reasoning activity and to interact with the students based on a deep understanding of the students' behavior. If such systems realize even half the impact of human tutors, the payoff for society promises to be substantial. By developing a system (software based system) to aid the teaching of combinational logic system to students, future researchers will want to develop systems on computer aided training in combinational logic systems.

REFERENCES

Anderson, J. R. and Corbett, K. (2003), *Artificial Learning Systems Using Robots*, Artificial Intelligence and Systems, V. H. Winston and Sons Publication, pp. 12 - 20.

Anderson, J. R. (2003), *The Architecture of Cognition*, Cambridge, Massachusetts: Harvard University Press, pp. 3 – 4.

Boolos, S. P. and Jeffrey, K. I. (2009), *Introduction to Final Reading*, New York Times Press, NY, pp. 2 – 6.

Cohen, P. A., Kulik, J. A. and Kulik, C. C. (2002), *Educational Outcomes of Tutoring: A Meta Analysis of Findings*, American Educational Research Journal, Vol. 19, pp. 237 - 248.

Collins, A. M. and Loftus, E. F. (2005), *A Spreading-Activation Theory of Semantic Processing*, Psychological Review, Vol. 82, pp. 407- 428.

Larkin, T. and Chabay, R. (2005), *Computer Assisted Learning Program Guidelines,* V. H. Winston and Sons Publication, Washington, D. C., pp. 12 – 18.

Mayer, R. E. (2008), *Teaching and Learning Computer Programming: Multiple Research Perspectives*, Lawrence Erlbaum Associates Publication, Hillsdale, New Jersey, pp. 7-14.

Steven, T. and Collins, F. (2007), *Artificial Learning a Practical Guide*, V. H. Winston and Sons

Publication, Washington, D. C., pp. 1 – 6.

Weinshank, D. J., Urban-Lurain, M., Danieli, T. and McCuaig, G. (2005), *Integrated Introduction to Computing*, Kendall/Hunt Publishing Company Dubuque, Iowa, pp. 2 – 9.