

An Artificial Intelligence Teaching Assistant for Microelectronic Circuits Problems

by

Salam Ibrahim Nachawi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2021

© Salam Ibrahim Nashawi 2021

Author Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

For several decades, personal computers have been used as a tool to enhance training and education. Some of the early computer systems used in this field, referred to as Computer Based Training (CBT) and Computer Aided Instructions (CAI), are static in nature. As such, the problem of these systems is that they do not adapt to learners' needs and abilities. Intelligent Tutoring Systems (ITS) were proposed to offer more human-like instruction to students by taking both the student and the knowledge domain into consideration. This thesis examines the components of such systems, discusses the challenges faced by designers, provides examples of previously implemented systems, and proposes a new system to tutor microelectronic circuits. This system concentrates on MOSFET circuits and their current-voltage characteristics. It also forms the base for future research to implement a more comprehensive microelectronics ITS.

Acknowledgements

I would like to express my sincere gratitude and appreciation to my supervisors, Professor Vincent Gaudet and Professor Mohamed I. Elmasry for their continuous support and encouragement during my research and academic work. This work would not have been successful without their advice and guidance.

I would also like to thank Professor William Bishop for reviewing my thesis and providing invaluable feedback and notable comments. In addition, I would like to thank Professor James Barby for his valuable discussions and comments during the thesis seminar.

I would like to thank my professors at Kuwait University for their efforts, support, and encouragement during my undergraduate and graduate studies, especially, Professor Imtiaz Ahmad and Doctor Sa'ed Abed for their continuous advice and guidance.

My deepest gratitude and appreciation goes to my mother, Maha Taha Nashawi. No words are enough to even describe how special and precious you are in my life. I am forever grateful and indebted for your endless encouragement and guidance throughout my life. I would also like to thank my father, Professor Ibrahim Sami Nashawi. Thanks for all the sacrifices you have made and for always believing in my capabilities. Moreover, a special thank to my sisters, Houda I. Nashawi and Rima I. Nashawi, for their continuous support, encouragement, and guidance. I could not wish for a more loving, caring, and thoughtful family.

Finally, I would like to take this opportunity to thank everyone who ever taught me in my life. I would have never reached this point in education without them.

Dedication

First and foremost, I am ever grateful to Almighty Allah for all his blessings and mercy that He has bestowed upon me throughout my life. I thank Him for giving me the courage, the mental capabilities, and the physical strength to carry on this research project and complete this thesis. It is under His grace that I live and learn.

I dedicate this thesis to my beloved parents Ibrahim Sami Nashawi and Maha Taha Nashawi and my adorable sisters Houda I Nashawi and Rima I. Nashawi.

Table of Contents

List of Figures	viii
List of Tables	x
1 Introduction	1
2 Literature Review on Intelligent Tutoring Systems	3
2.1 History of Intelligent Tutoring Systems	3
2.2 Components of an ITS	6
2.3 Equation Handling and Manipulation	9
2.4 Feedback Generation	20
2.5 ITS for Circuits	24
2.6 Summary of Existing ITSs	26
2.6.1 Andes	31
2.6.2 Mastering Physics	31
2.6.3 Socratic	31
2.6.4 Darisni	32
2.6.5 Khan Academy	32
2.6.6 Go Learning Bus University	32
2.6.7 Basic Physics Formulas	33
2.6.8 iPhysics	33
2.6.9 Memrise	33
2.6.10 Duolingo	34
3 Proposed Microelectronic Circuits ITS	35
3.1 Skeletal Algorithm	35

4	Experimental Results	42
4.1	First Example Question – Problem 4.48	42
4.2	Second Example Problem	58
4.3	Third Example Problem	63
5	Conclusions and Future Work	65
	References	67

List of Figures

2.1	Traditional Components of an ITS and their Interactions	7
2.2	Screenshot taken from Andes Interface showing a Newtonian Physics Problem (left), Variables (top right), and Equations Entered by the Student (bottom right)	10
2.3	A partial Solution Graph for the Inclined Plain Problem. It shows the Expert Model Reasoning to Find the Required Value (Final Velocity of the Car) and How to Apply Physics principles to Calculate It	12
2.4	Relationship between Physics Concepts Related to Force	14
2.5	A Portion of a Class Diagram Showing Some Vector and Scalar Principles	16
2.6	Parse Tree of Equation (1)	17
2.7	Parse Tree of a Correct Answer	18
2.8	Parse Tree of a Wrong Answer	19
3.1	General Code of the Student Module	36
3.2	Flowchart of the Teacher's Module	38
3.3	Flowchart of the Student's Module	41
4.1	Problem 4.48 page 498 [41]	43
4.2	LTspice Schematic of Problem 1	44
4.3	Screenshot of the Variables Imported from LTspice	45

4.4	MOSFET Specification Entry by the Teacher	46
4.5	A Sample of Variables that were not entered as a MOSFET Specification	47
4.6	List of Defined Variables Shown to the Teacher	48
4.7	Defining the Variables that the Student Must Solve for	49
4.8	Given and Required Variables of Problem 1 as Shown to the Student ...	50
4.9	Defining I_{D1} and I_{D2}	51
4.10	Output Produced by ITS when there is more than one Unidentified Variable in the Student's Equation	52
4.11	Error Generated when Using Triode Equation Instead of Saturation	53
4.12	Feedback for Errors in the I-V Characteristic Equations	54
4.13	Student Entering Correct Equations and Required Variables for Part 1 ..	55
4.14	The Error Message the Student Gets when Entering an Equation with too many Unknown Variables	56
4.15	Applying Kirchhoff's Current Law to Find V_{3b}	57
4.16	Finding I_{D3} and I_{D4}	57
4.17	Finding the Final Values for Problem 1	58
4.18	Second Example Problem	59
4.19	Circuit Schematic for Problem 2 as Entered into LTspice	60
4.20	Teacher Module Screen for Problem 2	61
4.21	Student Module Identifying Errors in a PMOS Equation	62
4.22	Circuit Schematic for Problem 3 as Entered into LTspice	63
4.23	Statement of the Third Example Problem	63
4.24	Feedback for Problem 3	64

List of Tables

2.1	Description and Comparison between Few of the Currently Available ITS ...	27
-----	---	----

Chapter 1

Introduction

An intelligent tutoring system (ITS) is a computing system that aims to facilitate and automate teaching by providing students with real-time feedback that is precise and adaptive. An ITS tries to mimic a human tutor by providing one-to-one instruction to each student according to their knowledge level and educational needs [1]. It also adds flexibility to the material to be taught and allows students to control the pace at which new knowledge is presented [2]. Many such systems have been proposed for use in different knowledge domains to help students and human tutors. Such domains include and are not limited to: flight simulation, general physics, and mathematics. To my knowledge, the field of microelectronic circuits is lacking an adequate automated system for teaching the subject efficiently. This research aims to fill that need by the design and development of such a system.

As the field of microelectronic circuits depends heavily on mathematical equations, previously proposed ITSs that tackle the physics and mathematics domains will first be studied. Also, circuit simulators such as SPICE - often already used as a training tool - will

be studied and integrated into the system to help instructors create and generate new problems.

The overall organization of the thesis is as follows: Chapter 2 presents a literature review of the previous related works. This chapter is subdivided in six main sections. Section 2.1 discusses some major ITS in history. Section 2.2 presents an in-depth study of ITSs and their components. Section 2.3 focuses on ITSs that target physics and mathematics and how they handle equations. Section 2.4 discusses feedback and hint generation in previous ITSs. Section 2.5 presents some ITSs that target circuits instruction. Finally, Section 2.6 presents a summary of some existing ITSs. Chapter 3 proposes the design of the new microelectronic circuits system. Section 3.1 designs an algorithm to solve a simple problem. Chapter 4 presents the experimental results. Chapter 5 discusses conclusions and future work.

Chapter 2

Literature Review on Intelligent Tutoring Systems

2.1 History of Intelligent Tutoring Systems

One of the earliest examples of an ITS, reported by Carbonell in 1970 [3], was called SCHOLAR. SCHOLAR was designed to help instruct students learn the geography of South America. What made it "intelligent" was its ability to handle student questions that were not expected by the programmer. Also, it was able to give instructional knowledge at various levels according to context. SCHOLAR's input and output language was English, and it was capable of initiating sessions by asking questions to the student, or have the student initiate the session by inputting a question. Carbonell called his model "Information Structure-Oriented Computer-Aided Instruction" (ISO-CAI) [3]. In 1990, Nwana [4] claims that SCHOLAR was in fact the earliest reported ITS.

Brown, *et al.* [5] extended SCHOLAR's ability to initiate dialog or have a dialog initiated as a response to the student's input, to increase the ways a student can interact with the

system. The platform they created was called SOPHIE (**SOPH**isticated **I**nstructional **E**nvironment). As opposed to the SCHOLAR pedagogical model, which was basically focused on giving information to the students, SOPHIE's pedagogical model would receive and assess student input and give advice accordingly. SOPHIE's scope of learning was electronic troubleshooting, and it allowed the students to experiment with faulty components in a safe environment, and it gave them experience in detecting electronic faults.

GUIDON [6] was designed as an ITS that teaches problem solving and diagnostic skills. It helps instruct the students to diagnose and treat bacterial infections. It initiates the learning process by giving students general information about a patient and allowing them to ask questions to further understand the symptoms to make a sound diagnosis. GUIDON evaluates students by referring to the MYCIN [7] medical expert knowledge system.

The ETOILE system [8] was developed in 1997 in an attempt to simulate a human tutor by implementing multiple teaching methods. It has five different teaching agents, each agent adapting a teaching method supported by educational psychology. Baylor's MIMIC system [9] is another system that has different teaching agents, this time three. These three agents teach instructional design from three different perspectives. They interact with each other as well as with the student to provide an optimized learning experience.

ALEKS is an ITS developed in 1996 by Canfield [10] to teach algebra and mathematics, in a factual manner. It has since expanded to teach math, chemistry, statistics, and

accounting [11]. Students can create an account for a simple fee; then, they start an assessment test in the course they want to master. The aim of that test is to determine what the developers call the “knowledge state” of the student. This state represents what topics the student is familiar with, and their degree of mastery in each topic so that the program can determine what material to present next. As the student progresses through the course, periodic assessments would update the knowledge state according to the student’s acquired knowledge. Other systems that employ the knowledge state approach include SIETTE [12] and RATH [13].

Some systems use adaptive navigation support to students to go through course material [14]. These systems give the student the ability to choose what to learn next while maintaining the general sequence of the material by hiding some material links and viewing others. Some of the systems that implement this technique include ELM-ART [15,16], InterBook [17], and De Bra's course [18].

The ViPS system [19], reported in 2013, allows students to create and simulate virtual pulley systems. This ITS helps to correct student misconceptions about the concepts of force and gravity by virtual experimentation. This guidance happens during the student attempts to solve the problem given by ViPS by creating a system of pulleys by the means of providing generic feedback and adding more case-specific hints if the student continues to

struggle. The ViPS system addresses conceptual misunderstanding problems; however, it does not teach numerical analysis skills.

Another ITS that focuses on the conceptual understanding of physics is WHY2-ATLAS [20]. Unlike ViPS that helps the students learn by experimenting, WHY2-ATLAS instructs students by having a dialog with them supported by a natural language processing engine. WHY2-ATLAS can analyze both long and short answers that students provide for a certain question and gives feedback on the answers' correctness and completeness. This was seen to extend ITS capabilities since most previously implemented systems could only process short, well-formatted answers.

2.2 Components of an ITS

An ITS has five main components [21]: (1) a student model, (2) a pedagogical module, (3) domain knowledge, (4) a communication model, and (5) an expert model. Figure 2.1 shows a diagram of these components and their interactions.

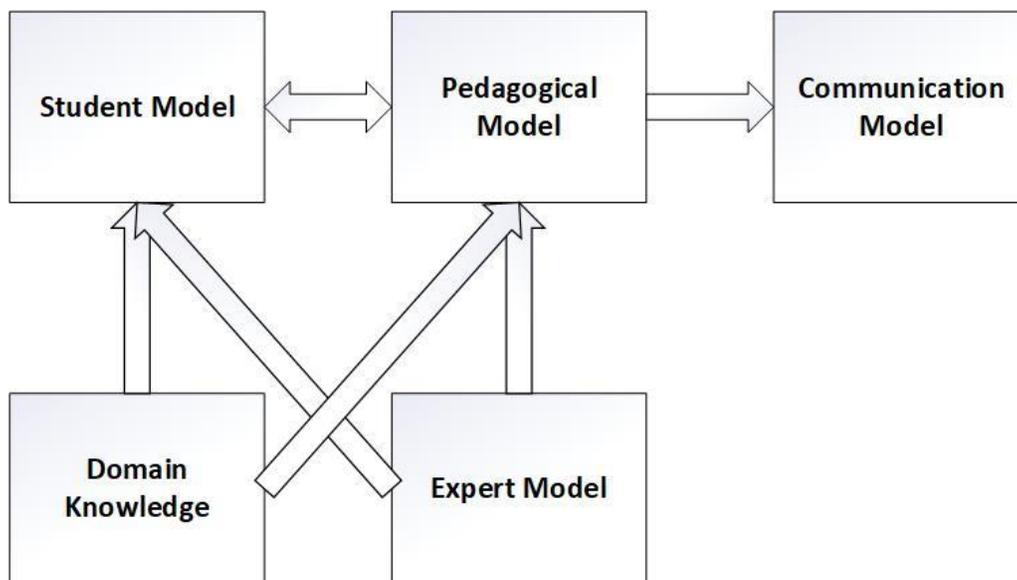


Fig. 2.1. Traditional Components of an ITS and their Interactions

Student model: The student model holds learner-specific information, including the current state of learner knowledge. This model helps the ITS to adapt to each individual learner’s needs. There are two common techniques to represent the student information: (1) overlay models and (2) Bayesian networks [21]. The overlay model represents student knowledge as a subset of the expert knowledge, and ITSs gradually tutor the student so that the knowledge of the two becomes exactly the same. One disadvantage of this model is that it does not show if the student has previous misconceptions that are not part of the knowledge base; therefore, an extension that allows for the explicit representation of “buggy” knowledge was suggested by Holt *et al.* [22]. A Bayesian network represents pieces of

knowledge in nodes and associates these nodes with the probability that the student knows this information based on their interaction with the system [21]. The student model can include more general information about the student than just their knowledge of the domain. An example is the students' preference to look at examples before they attempt to solve questions.

Pedagogical model: The pedagogical model determines the materials or topics to be presented to the learner, according to the information communicated by the student model [21]. One of the main challenges in designing this model is selecting the meta-strategy used to teach the domain. The best solution is to implement multiple teaching meta-strategies and let the student model choose the strategy that best suits the individual student. There are three low-level issues that need to be addressed by the pedagogical model designer after the meta-strategy is selected. The first one is selecting the topic the student needs to currently focus on. The second is that once a topic is selected, the system should generate and present the student with a problem to solve. This problem should have an appropriate difficulty level according to the student ability. Finally, when a student struggles with a problem, the system should generate and give appropriate feedback. The amount of help a student gets should again be according to their skills and ability.

Domain knowledge: The domain knowledge contains the information to be taught by the ITS. One of the main challenges is how to represent the knowledge so that it could be easily accessed by the other models.

Communication model: The communication domain is the ITS interface with the student. It presents the knowledge, the problems and the feedback, and takes questions and answers from the learner.

Expert model: The expert model contains the information being taught by the ITS and can simulate the ability of an expert to solve problems in the domain.

2.3 Equation Handling and Manipulation

Andes [23] is an ITS specialized in Newtonian physics. It targets college students that have previous algebraic experience. In the second version of Andes known as Andes 2 [23], a solution graph is generated for each problem. The solution graph is the main data structure used in Andes, where all possible solutions of a problem are generated and stored. Each solution graph has a solution point, where every variable of the problem is solved with its numeric value had the problem been solved. A color-by-number approach, where the variable is substituted with its correct value, is used to evaluate the correctness of the equation. If the left-hand side of the equation is equal to its right-hand side, the equation is said to be balanced; then, the equation is correct and is colored green; otherwise, it is wrong

and is colored red. This method has a flaw where adding extra variables that their value is zero or multiplied by zero is not considered as an error; however, it is claimed that this situation rarely happens. Figure 2.2 shows a screenshot from Andes interface [23].

The screenshot shows the ANDES Physics Workbench interface. The main window title is "ANDES Physics Workbench - [dt5a.fbd]". The menu bar includes "File", "Edit", "Diagram", "Variable", "View", and "Help". The toolbar contains various icons for file operations and physics tools.

The problem text reads: "A 2000-kg car in neutral at the top of a 20.0 deg inclined driveway 20.0 m long slips its parking brake and rolls down. If we ignore friction and drag, what would the magnitude of the velocity of the car be when it hits the garage door?"

The answer field is empty, with the label "Answer:" above it.

The free-body diagram shows a green circle representing the car. A vertical green arrow labeled F_w points downwards. A green arrow labeled d points upwards and to the left. A green arrow labeled c points downwards and to the right. A coordinate system is shown with a vertical green arrow labeled $+Y$ and a horizontal green arrow labeled $+X$.

The diagram of the inclined plane shows a red car at the top of a 20.0 m long driveway inclined at 20.0 degrees. A vertical line represents the garage door at the bottom of the driveway.

The Variables table is as follows:

Name	Definition
T0	car starts rolling
T1	car hits garage door
x	axis
mc	mass of car
d	magnitude of the Displacement of
Fw	magnitude of the Weight Force on

The equations entered by the student are:

- $mc = 2000 \text{ kg}$
- $d = 20.0 \text{ m}$
- $Fw_y = mc * g$
-

Fig. 2.2. Screenshot taken from Andes Interface showing a Newtonian physics problem (left), variables (top right) and equations entered by the student (bottom right) [23]

The solution graph is generated by a problem solver that uses three groups: known quantities, generated equations, and sought quantities [23]. Initially, the set of known quantities includes the quantities given in the question, and the set of sought quantities includes the quantities the student is asked to calculate. To generate the solution, Andes iteratively chooses a problem-solving method (PSM) to apply to one of the sought quantities. A PSM uses a hierarchical algorithm with multiple steps that works towards applying a major physics principle and the minor principles associated with it. The major physics principle that the PSM will choose is the one that contains the selected sought quantity and then updates the known quantities and the sought quantities set. The problem is solved when the sought quantities set is empty. The following is part of the solution graph for the problem in Andes illustrated in the screenshot in Figure 2.2.

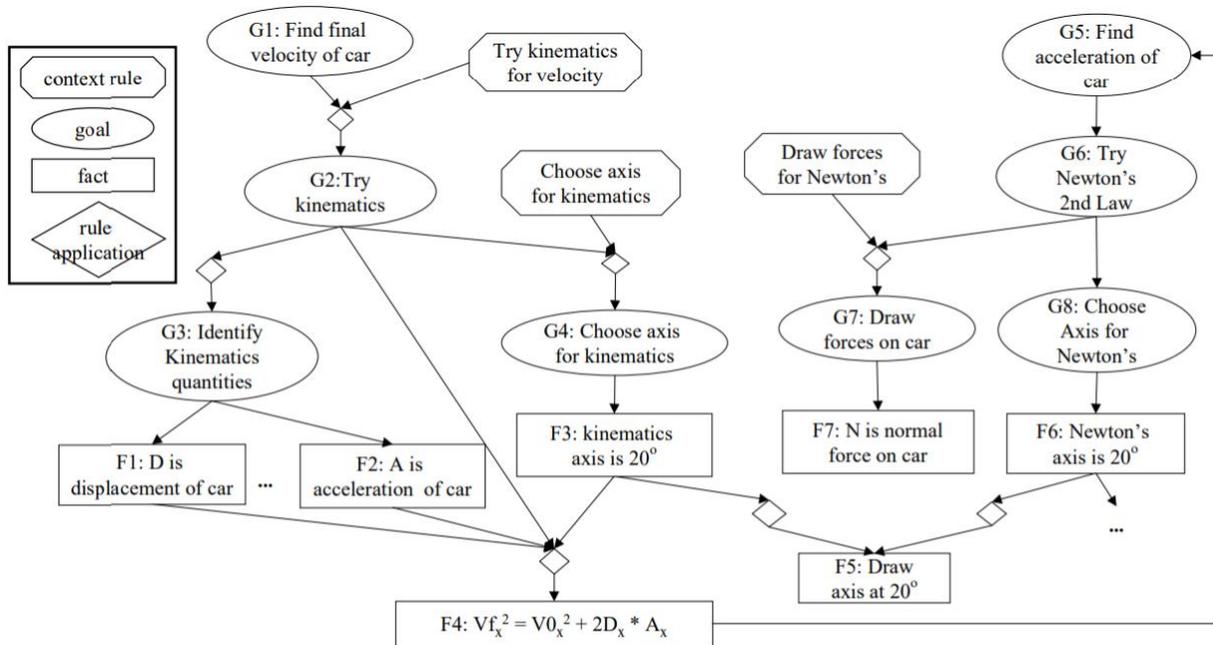


Fig. 2.3. A Partial Solution Graph for the Inclined Plane Problem. It shows the expert model reasoning to find the required value (final velocity of the car) and how to apply physics principles to calculate it [24].

In 1999, Liew *et al.* [25] presented an intelligent tutoring system called the PHSYICS-TUTOR. This system accepts equations as answers from students and provides an algorithm to reason about such answers. Liew *et al.* [25] claimed that such an algorithm must have the ability to manipulate algebraic equations and represent the basic concepts of physics. Before representing their reasoning mechanism, they discussed the then-available methods of accepting students' answers and the weaknesses of each method as follows.

- a) Multiple choice question: This type of question is the easiest to grade and can have pre-programmed immediate feedback. The problem with this method is that the tutor does not know how the students reached their answers.
- b) Natural language text: This type of question uses keywords and phrases to assess the answer. Liew *et al.* [25] claimed that natural language text is not appropriate for answers to questions concerning physics, as there are many ways to represent the same equation.
- c) Numeric answers: Numeric answers can be used for questions when all variables have given values except for one. The problem with this type of question is that when a student gives a wrong answer, it is hard to determine the source of the error as there are no intermediate steps to generate the final value.

Liew *et al.* [25] also claimed that although the Andes system accepts equations as input from students, it does not apply the knowledge of physics and its principles to reason about the students' answers. Andes uses equation comparison to evaluate and give feedback. The problem with Andes' approach is the huge number of equivalent equations for each answer, which makes pre-programming all solutions an infeasible task. This, in turn, limits Andes' ability to provide suitable feedback for each equation entered by the student. Liew *et al.* [25] found that the basic principles in the domain of introductory physics can be represented

easily due to their fairly manageable number. Also, algebraic equations are only conceptually correct if they are properly derived from these principles. PHYSICS-TUTOR has a knowledge base that is composed of these basic principles, their dimensions, and the operations that can be carried out on them and whether the result of the operation is a new instance of physics. This knowledge is represented as a relationship diagram. Figure 2.4 shows a relationship diagram of physics concepts related to force.

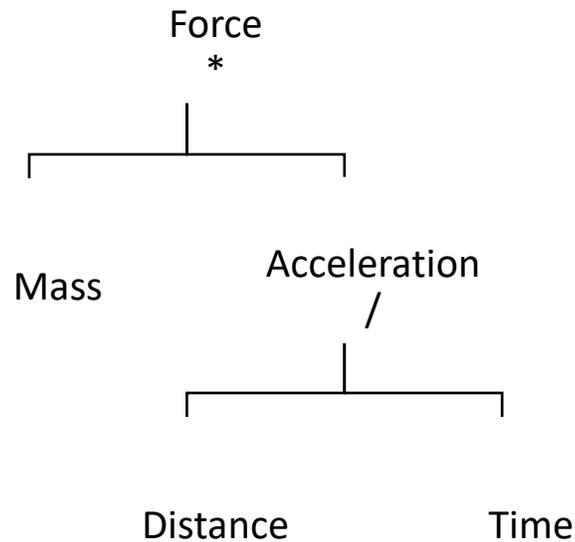


Fig. 2.4. Relationship between Physics Concepts Related to Force

The diagram in Figure 2.4 shows that the force can be calculated by multiplying mass and acceleration, and that the result of dividing the distance by time is the velocity. A diagram can also imply that certain operations cannot be applied to certain principles; for example, mass cannot be added to force. The domain knowledge also has a *class* diagram, where each physics concept is associated with its structure. This allows the system to know which operations are allowed on which concept [25]. For example, force is a vector; hence, it can be decomposed to its principal components. The decomposition operation cannot be applied to mass because it is a scalar quantity. A portion of a class diagram is shown in Figure 2.5.

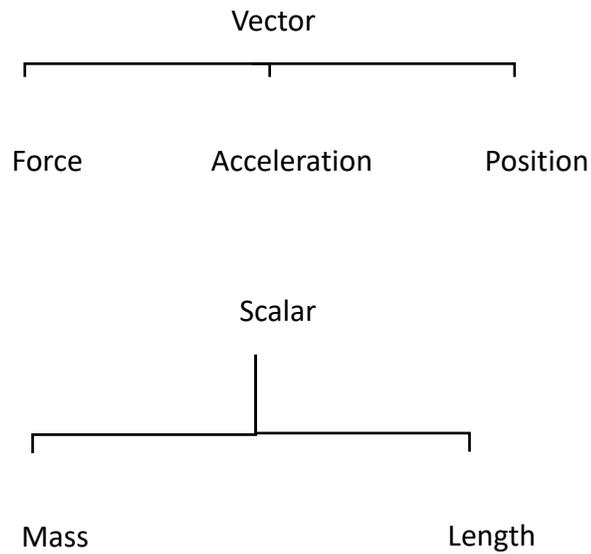


Fig. 2.5. A Portion of a Class Diagram Showing Some Vector and Scalar Principles [25]

For each question, the instructor supplies PHYSICS-TUTOR with an answer. This answer is parsed to form a tree, called *parse* tree. For example, the equation

$$m_A * g \sin(\theta) = m_B * g \tag{2.1}$$

is parsed into the parse tree of Figure 2.6 [25].

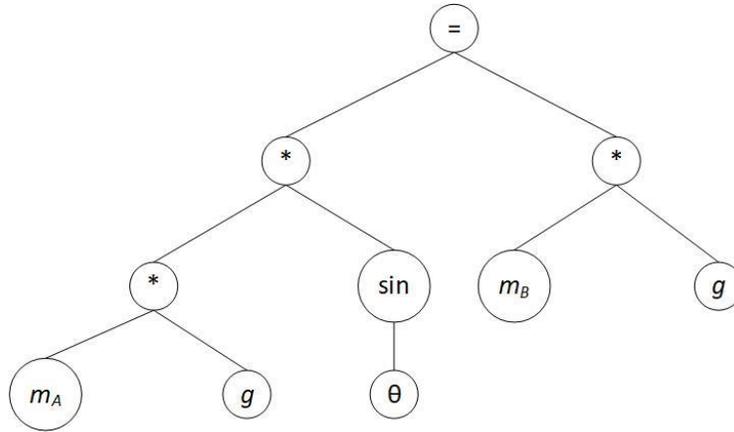


Fig. 2.6. Parse Tree of Equation (1)

When the student enters an equation as their answer, PHYSICS-TUTOR first runs a dimensionality check on the parse tree by using the domain knowledge to assign dimensions for the variables in the student's answer, e.g., kg, m/s, etc. [25]. This ensures that the answer is dimensionally consistent. If the answer is not dimensionally consistent the student is asked to enter a new one. An example of a correct answer that the student may enter is:

$$m_A * \sin(\theta) - m_B = 0 \quad (2.2)$$

The answer in the previous equation has been rearranged and simplified by removing a common factor. This answer is parsed as shown in Figure 2.7 [25].

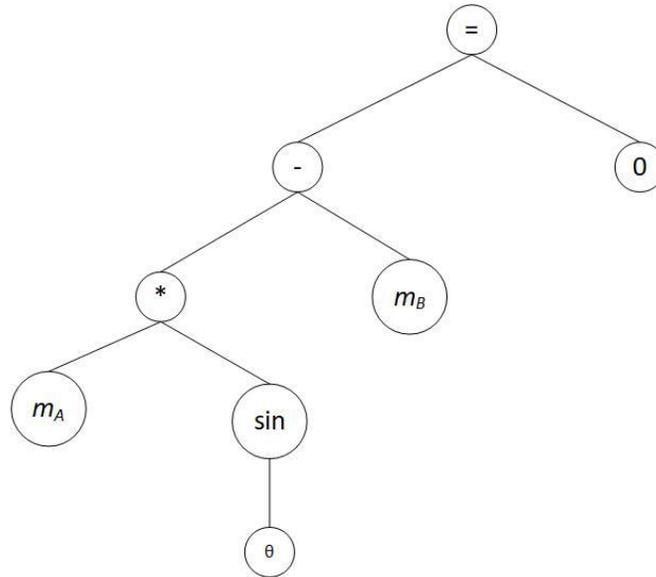


Fig. 2.7. Parse Tree of a Correct Answer

To match the student's answer with the instructor's, the system does the following [25]:

1. The system scans the tree from the bottom up to make sure that all terms match a physics concept in the domain knowledge. If it does not find a match, it constructs a list of possible matching concepts. The concept that matches both term in the example is *force = mass * acceleration*.
2. Then, the system finds a modification to the terms so that they form a physical quantity. In this example, it adds the acceleration, g , to both terms.
3. At the end, the system applies algebraic manipulation to the tree until it matches the solution tree.

Let us assume that the student has provided the following wrong answer [25]:

$$m_B * g + m_A * g \sin(\theta) = 0 \quad (2.3)$$

The parse tree of this answer is shown in Figure 2.8 [25].

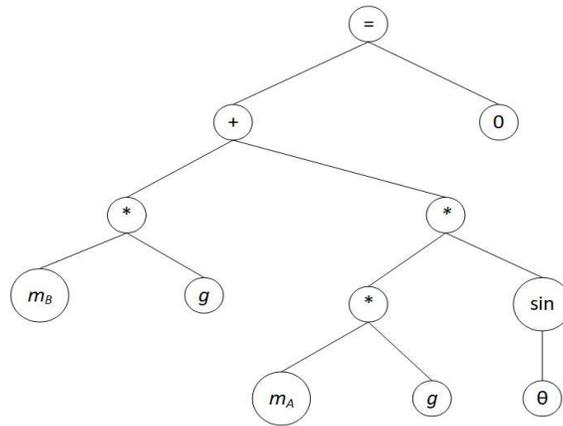


Fig. 2.8. Parse Tree of a Wrong Answer

This answer is not dimensionally consistent; therefore, the system will not be able to match it to the correct answer. The closest correct match it could find is when the (+) operator is replaced with a (-). The system has data knowledge of the different errors that a student can make, and it has different handlers to handle different types of errors.

The PHYSICS-TUTOR system has some limitations regarding hit-generation, handling time and periodic quantities, and differentiation and integration [25].

2.4 Feedback Generation

In 1998, Gertner [26] developed a feedback system to help students correct their mistakes in quantitative physics equations in a timely manner. This feedback system was developed to be integrated with the Andes Newtonian physics ITS. The system individually evaluates each equation entered by the student to check whether it is part of the correct solution. One of the main problems of the Andes ITS is that students often complain that the system occasionally flags some of the correct equations as incorrect. Gertner [26] explained that this problem can happen for one of three reasons. The first reason is that the equations were correctly flagged incorrect; however, the student's final answer was correct. This, phenomenon along with lack of appropriate feedback, made the student confuse their equations to be correct. The second reason is students using interpretations different from those that are used by the ITS. The third reason is that the equations are correct; but, they are not part of the partial solution pre-generated and stored by the ITS. Gertner [26] addressed the first and second reasons, whereas the third reason was partially addressed by adding alternative solutions to Andes and giving the students feedback if their solutions were not the optimum one. To give feedback on an incorrect equation, Andes compares it with all equations stored in the system that are part of the solution for that problem. If there is a match, the equation is correct; else if there is a close equation, the system gives a hint on the

difference. If no match is found, the system uses hints to guide the students to the correct solution.

The equation comparison is done by representing the student's equation and Andes' generated equations in canonical form. The equation is converted to its equivalent canonical form by first rearranging it in such a way that all the terms are moved to one side of the equation and other side is equal to zero. Then, the resulting polynomial is converted to a vector. The first element of the vector is a variable and the coefficient at the $n+2$ position of the vector is the coefficient of the variable raised to power n . For example, $x^2 + 2$ is represented by the vector $[x \ 2 \ 0 \ 1]$ where x is the variable; 2 is the coefficient of x^0 ; 0 is the coefficient of x ; and 1 is the coefficient of x^2 . Similarly, the vector $[x \ 2 \ [y \ 0 \ 1] \ 1]$ is the canonical form of the polynomial $2 + x^1[0y^0 + y^1] + x^2$ which is the polynomial $x^2 + xy + 2$.

The closest equation to the student's answer is found by calculating a match score between the student's equation and every equation in the hash table generated by Andes. The match score between two vectors ranges between 0 and 100. This score is calculated by computing individual match scores between the elements of the vectors as follows. If both elements are variables, a function that calculates the similarity score between the variables is used. If both elements are numbers, the score is 100 if they are the same; 80 if they are negative of each other; 70 if $n_1 = 1/n_2$, $n_1 = \cos(n_2)$, or $n_1 = \sin(n_2)$; and 30 otherwise.

If the two elements are of different types; then, the score is zero. After that, the individual scores are multiplied by a factor based on the element's relative positions in the two vectors. At the end, the scores are summed up and divided by the average length of the two vectors to give the final match score. After finding the closest equation, Andes calculates match scores for each term in the equation. The term with the lowest match score is used to generate a hint from a preset table. That was the case in Andes 1.

Andes 2 has what the authors call the "what is wrong help hint" button. This button generates hints by using error handlers [23]. When the student presses the "what is wrong" button, a group of error handlers each specialized with one type of error (sign, trigonometry, etc.) edits the student's equation according to the error it is designed to handle. After each edit, the equation is re-evaluated. If it becomes correct then the error handler returns the solution and a priority; otherwise, it returns nil. The error handler that generates a solution with the highest priority is used to generate three hints: a pointing hint, a teaching hint, and a bottom-line hint. Andes gives hints to only one mistake at a time even though it can detect more than one mistake because it lacks the natural language processing power required to generate comprehensible hints for multiple errors at once.

The solution graph mentioned in Section 2.2 is also used to generate what in Andes is called the "what's next" hint [23]. These hints are used when the student is stuck and does not know what to do next. When a "what's next" hint is provoked, Andes checks the solution

graph to find a main principle that the student has not applied yet and points them towards applying it.

Liew and Smith [27] proposed a framework to provide feedback for a system of physics equations. They determined that five factors make the mapping process between the student equations and the ideal answer more difficult. These factors are: (1) variable renaming, (2) simple aliasing of variables (variables that are equivalent to each other), (3) mapping coefficients for a pair of variables, (4) elimination a class of variables (implicitly representing some variables as an expression of others), and (5) the choice of coordinate axes.

In Liew and Smith's [27] system the student should submit all equations before feedback is generated. As it is the case for Andes 1, the first step of analysing the student equation is transforming it to canonical form. After that, the whole set of equations is algebraically transformed to eliminate extra variables; for example, the variable $v_1 = k * v_2$ is eliminated by substituting it with the expression $k * v_2$. Also, intermediate variables that are only represented in the student answer are eliminated by the same method. When the student set of equations has the same cardinality as the answer set of equations, a one-to-one mapping between the variables is carried out by first looking for variable names that are a perfect match, and then uses heuristics and domain knowledge to map the remaining variables. Once the variables are mapped, the algorithm maps each equation in the student set to its

corresponding equation in the answer set. Only equations with the same dimensions are compared for finding a possible match. An equation is considered a good or bad match based on heuristics such as the presence or absence of a certain term in both equations.

After the equations in the student's set are mapped to their corresponding equations in the answer set, they are compared for differences for feedback generation [27]. Examples of such differences include a '+' sign instead of a '-' and vice versa, missing or extra terms, incorrect coefficients, missing or extra trigonometric functions, and other differences that do not result in a change of the equation's dimensions. Some differences such as a switch of sign can happen due to the use of different coordinate axes. This is eliminated by inspecting if the sign switch is consistent in all of the equations in the student set. Feedback is generated based on the differences found in the comparison stage.

2.5 ITS for Circuits

Electronic circuits present their own challenge when designing an ITS. This is due to their vast domain and the large variety of skills the student should master. These skills include conceptual understanding, circuit analysis, troubleshooting and simulation, and design. The latter gives the ITS designer the greatest challenge as design problems are usually open ended and have multiple valid answers.

SOPHIE [5], discussed in Section 2.1, was one of the first attempts to develop an ITS for electronic circuits. Its target was to help students master simulation and circuit troubleshooting skills by allowing them to experiment with a simulated platform and give them feedback to their designs.

The Circuit Exercise [28] was proposed in 1992 to assist students in basic circuit analysis. It generates a circuit problem randomly, solves it and gives the students feedback as they solve it. The drawback of this system is the limited number of circuit components it can analyse. The components that it can work with are resistors, capacitors, and coils.

The Interactive Multimedia Intelligent Tutoring System (IMITS) [29] is another ITS that targets the basic undergraduate circuit instruction including AC, DC, and transient analysis. It challenges students with real life problems that would face an electrical engineer. It allows students to access the material which includes interactive tutorials and gives them access to a simulation laboratory to test their designs. The students can choose the order in which they would solve the exercise questions and the system monitors their knowledge evolution and suggests what material to cover next.

A more recent electronics ITS is the ElectronixTutor [30]. The goal of ElectronixTutor is tutoring apprentice electrical technicians in the navy. It incorporates many learning agents that support conversation with a student and multiple choice questions with feedback as well as some classic textual material and videos.

As it has been discussed in the previous sections, the available ITS systems either target basic electrical circuits, or are focused on simulation and have little to no problem-solving exercises. This led us to strongly believe that an ITS that facilitates the instruction of microelectronic circuits with non-linear components such as MOSFET diodes and bipolar transistor is urgently needed. This research provides a steppingstone for the development of such an ITS. The following chapter will discuss how this problem is approached.

2.6 Summary of Existing ITSs

Nowadays, there are many types of ITSs in the market. Table 2.1 provides a brief description and comparison between some of them.

Table 2.1. Description and Comparison between a Few of the Currently Available ITS

Name	Research/ Commercial	Year	Tested	Claims	Language	Input	Subjects
1- Andes	Research	1997	Yes	As good as pencil and paper with the need of manual grading	English	Equations	Physics
2- Mastering Physics	Commercial	1998	Yes	<ul style="list-style-type: none"> - Help students develop problem-solving skills for success in physics - Build conceptual and quantitative understanding - Encourage students to learn and apply physics concepts 	English	No search engine. MCQ	Physics
3- Socratic	Commercial	2013	Yes	<ul style="list-style-type: none"> - Works for all subjects - Built for learning - Powered by Google AI - Loved by educators 	English	<ul style="list-style-type: none"> - Handwritten problems - Search engine 	Biology, Chemistry, Physics, Earth Science, Environmental Science
4- Darisni	Commercial				Arabic	MCQ	Math, Science, English, Arabic
5- Khan Academy	Non-profit	2008	Yes	<ul style="list-style-type: none"> - Personalized learning - Trusted content - Tools to empower teachers 	English	MCQ, numeric	Math, Science, Economics and Finance, Arts and Humanities. Computing
6- Go Learning Bus University	Commercial	2015	Has 93 rating of average of 4.1/5 on the App Store	<ul style="list-style-type: none"> - Combined over 300 applications in one - AI driven coach - 60 minutes a week can help becoming great in more than 300 topics 	English	MCQ	School, University, Professional Training, Languages
7- Basic Physics Formulas	Commercial	2015	4 ratings of 5/5	Apply technology to teach physics better and faster, and to provide tools for learning that are objective, so the students can achieve their goals (good grades), but in a captivating way, in the sense that knowledge is truly incorporated and will not be easily forgotten.	English	N/A	Physics

Table 2.1 (Continued - 1)

	Levels	Type of Questions	Feedback	Interactive	AI	Online	Teacher/ Student	Specific Curriculum	Input type
1- Andes	University	Problem Solving	Immediate	- Allows for three hints generated according to entered equation - Allows for next step hint	Yes	On-Offline	Both	No	Text
2- Mastering Physics	High school and University	MCQ	Immediate	- Hints pre-programmed by choice - has interactive simulators and learning games	No	Yes	Both	Yes	Text
3- Socratic	N/A	Answers questions and shows examples	N/A	No	NN probably used for character recognition	Yes	Student	No	Oral, text, picture
4- Darisni	All school levels	MCQ	After Exam is done and submitted	Can live connect to a teacher	No	Yes	Student	Yes	Text
5- Khan Academy	All levels	MCQ, problem solving	Immediate	Allows for step by step hints	Yes, student model employs AI to know which parts of the knowledge base have been perfected	Yes	Both	No	Text
6- Go Learning Bus University	All Levels	MCQ	Immediate	No	AI couch for paid users	Yes	Student	No	Text
7- Basic Physics Formulas	High school and University	N/A	N/A	Shows detailed solution at request	No	No	Student	No	Text

Table 2.1 (Continued - 2)

Name	Research/ Commercial	Year	Tested	Claims	Language	Input	Subjects
8- iPhysics	Commercial		5 ratings of 4.6/5 on the App Store	<ul style="list-style-type: none"> - Learn quickly from over 70 topics, - Easily repeat formulas, theorems, definitions and properties - Quickly find the topics that matters to you 	English	MCQ	Physics
9- Memrise	Commercial		139.9k ratings of 4.8/5 on the App Store	<ul style="list-style-type: none"> - Learn with locals - Practice games - Improve your pronunciation - Learn anywhere 	English	MCQ, Words formed from list of letters	Spanish, French, Japanese, German, Korean, Italian, Russian, Chinese, Portuguese, Arabic, Norwegian, Dutch, Swedish, Polish, Turkish or Danish.
10- Duolingo	Commercial	2012	87.778K ratings of 4.6/5 on the App Store	<ul style="list-style-type: none"> - Feels like a game and makes sure you stay motivated - Take 10 minutes a day, and you'll surprise yourself with how well you can speak a new language 3 months from now - Practice with fun lessons that will leave you eager to learn more, and develop reading, writing, speaking, listening and conversation skills along the way 	English, Arabic, Chinese, Czech, Dutch, French, German, Greek, Hindi, Hungarian, Indonesian, Italian, Japanese, Korean, Polish, Portuguese, Romanian, Russian, Spanish, Thai, Turkish, Ukrainian, Vietnamese	Choice from list	Languages

Table 2.1 (Continued - 3)

Name	Levels	Type of Questions	Feedback	Interactive	AI	Online	Teacher/ Student	Specific Curriculum	Input Type
8- iPhysics	Highschool, introductory university	MCQ	Immediate	No	Calculate a percentage of topic mastery	Yes	Student	No	Text
9- Memrise	Beginner to expert	MCQ, translate word	Immediate	Yes	Yes, the words appearance in questions depends on how much a student struggle with them	Yes, offline paid	Student	No	Text
10- Duolingo	Beginner to expert	MCQ, translate word, match the pair	Immediate	Yes	Not apparent	Yes	Student	No	Text

2.6.1 Andes [31]

Andes is a problem-solving platform that forces the student to draw vectors and define all variables to be able to solve the question. At each step of drawing, variable definition, or equation entry, Andes gives immediate feedback by color. If the feedback is that the student has made a mistake, the student can ask for up to three gradual hints, and the third one gives the answer. If the student is stuck and does not know what to do next, they can also ask for up to three hints.

2.6.2 Mastering Physics [32]

Mastering Physics is an online platform that includes explanatory videos, interactive simulators, and a huge question bank. It allows teachers to set up homework and quizzes for their students.

2.6.3 Socratic [33]

Socratic is a research engine that gathers education material from all over the World Wide Web (WWW) and videos from YouTube. Students can ask their questions orally, by text, or by scanning a written problem. Then, Socratic shows the student one or more websites that has a problem-solving engine for that particular problem.

2.6.4 Darisni [34]

Darisni is an online platform for school students in all grades. It has explanatory videos and MCQ exams. The students do not know whether they answered correctly to a certain question until they submit the whole exam. The students can also schedule time with an actual teacher for live lessons.

2.6.5 Khan Academy [35]

Khan Academy is an online learning platform for various topics at different levels. It divides each course into chapters and has a quiz at the end of each one. These quizzes assess the students' understanding of the knowledge base with the help of AI.

2.6.6 Go Learning Bus University [36]

This application is the main application of over 300 learning applications that teach various courses at various levels. It teaches the material by text and graphs and has MCQ quizzes with immediate feedback after each chapter. It also offers an AI coach for subscribed users.

2.6.7 Basic Physics Formulas [37]

Basic Physics Formula is an application that summarises and explains basic physics by text and graphs. The formulas are organized by topics like mechanics, fluids, optics and so on. Each topic is divided into subtopics where a brief explanation and examples are presented.

2.6.8 iPhysics [38]

iPhysics is an application that summarizes physics topics and offers links to Wikipedia for further explanation. It offers three levels of MCQ quizzes at the end of each topic.

2.6.9 Memrise [39]

Memrise is a language learning application that teaches the students by sound and text. It has MCQ and word/phrase translation questions that ask the students to form words from a list of letters, to help students' revision. It can detect the words that a particular student struggles with and include them more often in revision questions. It also has an explore mode where the students can use the smartphone camera to learn the name of the objects in their surroundings.

2.6.10 Duolingo [40]

Duolingo is a language learning platform that is offered in many languages. The languages available to learn depend on the student native language. It offers sound and text and has MCQ, match the pair, and word/phrase translation questions, form words from a list of letters.

This chapter has presented a review of current ITS with emphasis on the ones that target circuits and physics. The next chapter will propose our ITS and discuss its features.

Chapter 3

Proposed Microelectronic Circuits ITS

This chapter will discuss the proposed microelectronic circuits ITS. First, the techniques used will be briefly described followed by a description of the two modules that compose the system: (1) the teacher's module and (2) the student module.

3.1 Skeletal Algorithm

It is suggested to use a similar approach for the proposed circuits ITS; however, instead of using a problem solver to find the value of the sought quantities, an open-source Spice circuit simulator can be used to calculate the values of the various variables that are included in the problem. The instructor will have to draw the circuit for any new problem to be added to the ITS and identify the known and sought variables. Once this is done, the circuit simulator will find the value of the sought variable and send it to a framework that generates “for loops” to accept input from the student, similar to the initial code shown in Figure 3.1. below

```

###Taking and processing input part a
while ((Flag_V1a&Flag_V2a) !=1):
    eq=input("Enter an equation or final value for part a\n")
    answer=eq.split("=") #split the equation by the "=" sign
    lhs=answer[0] #left hand side
    rhs=answer[1] #right hand side
    #checking answer
    lhs_parsed = parser.expr(lhs).compile()
    rhs_parsed = parser.expr(rhs).compile()
    lhs_value= eval(lhs_parsed)
    rhs_value=eval(rhs_parsed)
    rhs_value=round(rhs_value, 6)
    lhs_value=round(lhs_value, 6)
    if (lhs_value==rhs_value):
        answer=1
        print ("\nYour entry is correct\n")
    else :
        answer=0
        print ("\nYour entry is incorrect\n")
        %run error handlers

print("\nWell done!! Part a is done")

```

Fig. 3.1. General Code of the Student Module

The next step of the development process is to add a module for the teacher where he/she can run LTspice and generate an operating point simulation RAW file. That raw file is then read by the teacher's module and the simulated variables are extracted and displayed; then, the teacher will be asked to complete the following steps:

1. Enter the MOSFET specification for each device in the simulated file.
2. Enter any variables needed to solve the problem.
3. Specify the given variables from the list of the variables that have been entered.

4. Specify the number of parts that the question has and enter the variables the student is required to find to complete the solution.

Figure 3.2 displays the flowchart of the teacher's module.

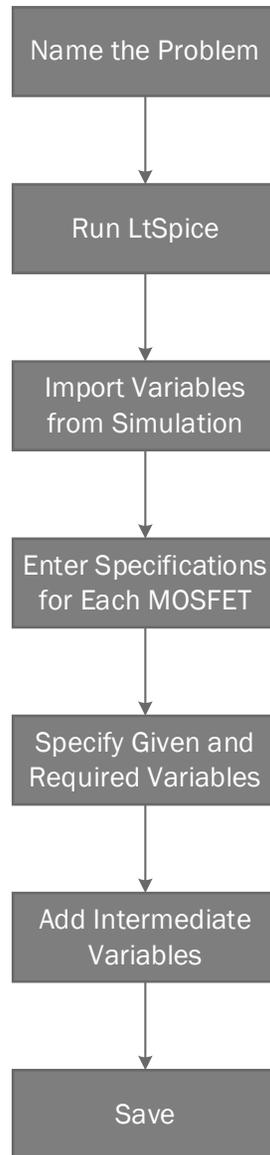


Fig. 3.2. Flowchart of The Teacher's Module

After completion of the previously mentioned steps, the variables are saved in the problems' library and are ready to be imported by the student module to be solved.

When the student module is run, it asks the student to enter the name of the problem to be solved and the program will load its variables. Then, the given variables and their values will be displayed, and the student is asked to find the required variables. The console then asks the student to enter an equation or final value for each part of the problem until all required variables for that part have been entered correctly. Once the student enters an equation, the student gets an immediate textual feedback telling hem if heir entry is correct or not. This is done by substituting all variables in the student's entry with their correct values and checking whether the equation is balanced. If the student enters a new variable that has not been listed as given in the problem, the student receives an undefined variable message, unless the entry is recognized by the module as a variable definition.

When the student enters an incorrect equation, the feedback generation routine is activated. As the focus of this thesis is MOSFET circuits, the error handlers only look for errors in the MOSFET I-V characteristic equations. Common errors that are included in the proposed ITS are:

1. Using an equation for the wrong region of operation.
2. Mistaking K'_n and K_n (or K'_p and K_p) and hence not multiplying by W/L .
3. Confusing the source and the drain terminals.

4. Using V_{GS} instead of the overdrive voltage V_{ov} .
5. Not applying the body effect, when applicable.

The algorithm detects these mistakes by substituting the erroneous variable with the correct one and re-evaluating the equation. If the equation becomes correct, the ITS displays a hint appropriate to the student's mistake. The algorithm is able to detect one error per entry and is able to detect errors in equations whether the variables are represented numerically or symbolically. Figure 3.3 displays the flowchart of the student's module.

Thus far, I have proposed a new ITS for microelectronic circuits and discussed its components. The next chapter (Chapter 4) will present the experimental results and screenshots of actual runs of the ITS system.

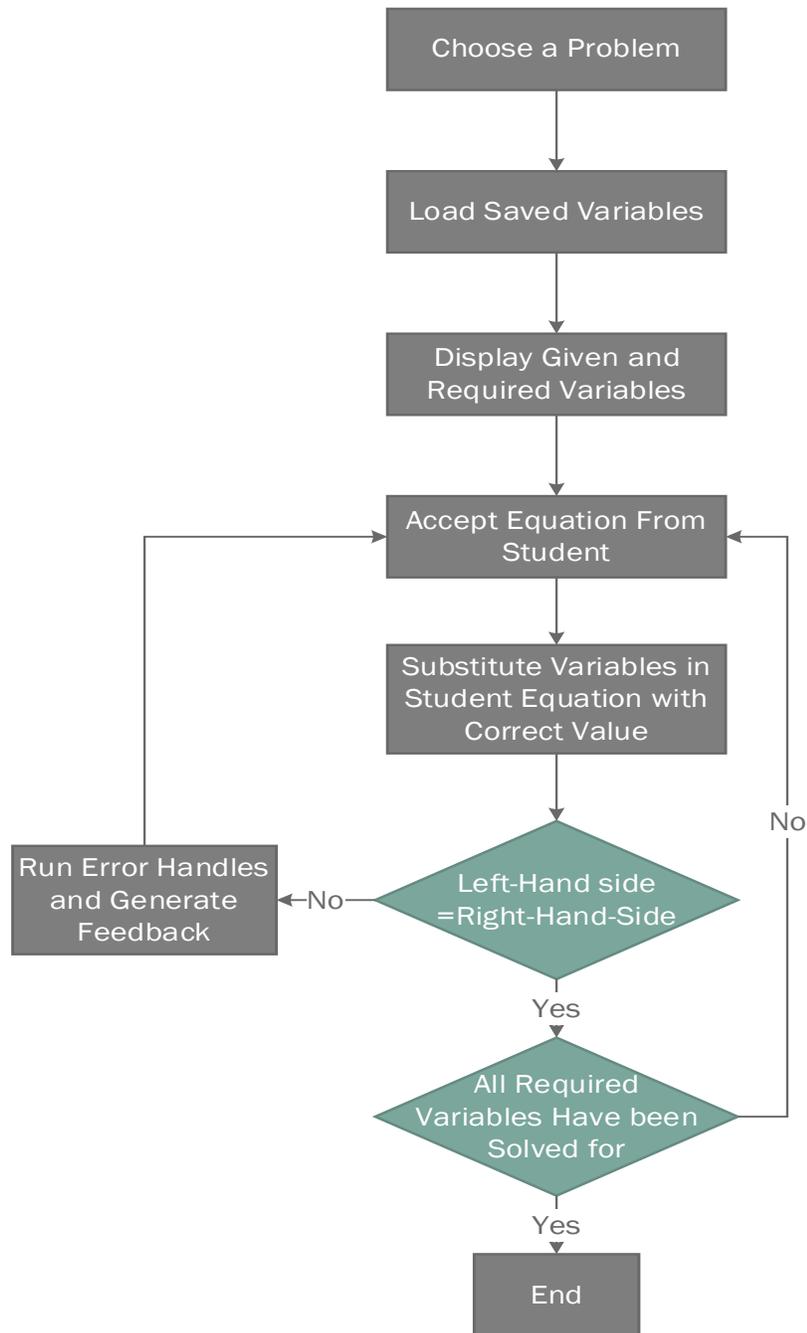


Fig. 3.3. Flowchart of the Student's Module

Chapter 4

Experimental Results

In this chapter, I will present the test methodology for the microelectronic circuits ITS presented in Chapter 3. The program has been tested on three different problems/examples taken from the textbook “Microelectronic Circuits”, commonly known as Sedra/Smith. The used problems are taken from the International sixth edition [41].

4.1 First Example Question – Problem 4.48

The first question is Problem 4.48 on page 498 [41]. It was chosen to test the ITS program's ability to deal with a typical NMOS circuit containing multiple MOSFETS working in the saturation region. It also tests the program's ability to track the required variables of problems containing more than one part. The text of the question is illustrated in Figure 4.1 as follows:

In the circuit of Fig. P4.48 [41], transistors Q_1 and Q_2 have $V_t = 1\text{V}$ and the process transconductance parameter $K'_n = 100\mu\text{A}/\text{V}^2$. Assuming $\lambda = 0$, find V_1 and V_2 for each of the following cases:

a) $(W/L)_1 = (W/L)_2 = 20$

b) $(W/L)_1 = 1.5(W/L)_2 = 20$

Fig. 4.1. Problem 4.48 page 498 [41]

The following pages present a detailed work through for problem 4.48, and the following subsections introduce two more problems and present some screenshot of their solutions.

When the teachers first run the teacher's module, they are asked to enter a name for the problem. The previous problem was named "Problem 1", as it was the first problem used to test the program. After naming the problem, LTspice is run automatically. For ease of simulation, the two MOSFETS in the two parts of the problem are duplicated to form four MOSFETS named M_1 , M_2 , M_3 , and M_4 , two for each part. Similarly, V_1 , V_2 , and V_3 are duplicated and named V_1 , V_2 , and V_3 for part a, and V_4 , V_5 , and V_6 for part b. Note that this naming scheme is for simulation purposes only and can be changed by the teachers when they specify the problem's variables. Figure 4.2 shows a screenshot of the LTspice circuit schematic for the example problem.

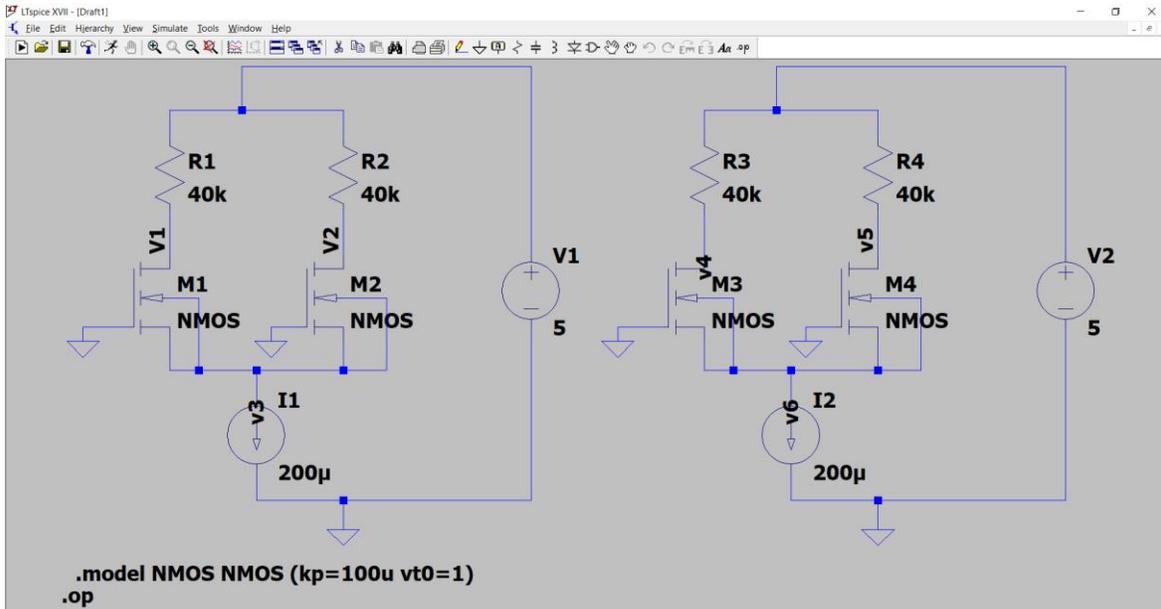


Fig. 4.2. LTspice Schematic of Problem 1

After the teacher finishes simulating the circuit and closes LTspice, they are asked to enter the name and path of the LTspice raw file that has been generated by running the simulation. Once this is done, the raw file is read by the help of the `ltspsy3.py` module [42] and the number of MOSFETs in the simulation is determined to facilitate entering their specifications. Then, the program displays the values of the variables generated by the simulation and the number of MOSFETs it has detected. It will also ask the teacher to confirm this number. Figure 4.3 shows a screenshot of the console interface after quitting LTspice.

```

IPython console
Console 1/A
Enter the name of the problem you want to add:
problem 1
LtSpice will run Please draw and simulate the problem you would like the student to solve

Enter the name and path of the raw file
problem 1.raw
variables obtained from ltspice are:

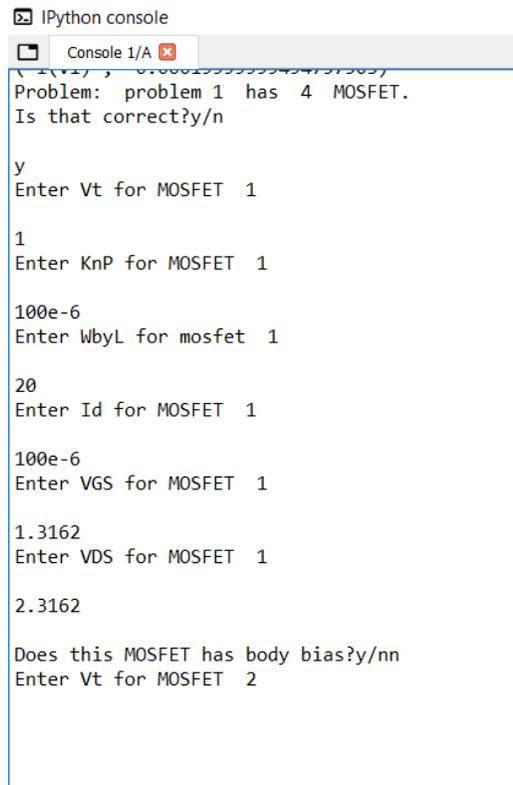
('V(n001)', 5.0)
('V(v1)', 1.0)
('V(v2)', 1.0)
('V(v3)', -1.3162277936935425)
('V(n002)', 5.0)
('V(v4)', 0.19995209574699402)
('V(v5)', 1.8000478744506836)
('V(v6)', -1.346411943435669)
('Id(M4)', 7.999880472198129e-05)
('Ig(M4)', 0.0)
('Ib(M4)', -3.1564596856409333e-12)
('Is(M4)', -7.999879744602367e-05)
('Id(M3)', 0.00012000119750155136)
('Ig(M3)', 0.0)
('Ib(M3)', -1.5563639786664307e-12)
('Is(M3)', -0.00012000119750155136)
('Id(M2)', 9.999999747378752e-05)
('Ig(M2)', 0.0)
('Ib(M2)', -2.326227752091903e-12)
('Is(M2)', -9.999999747378752e-05)
('Id(M1)', 9.999999747378752e-05)
('Ig(M1)', 0.0)
('Ib(M1)', -2.326227752091903e-12)
('Is(M1)', -9.999999747378752e-05)
('I(I2)', 0.00019999999494757503)
('I(I1)', 0.00019999999494757503)
('I(R4)', 7.999880472198129e-05)
('I(R3)', 0.00012000119750155136)
('I(R2)', 9.999999747378752e-05)
('I(R1)', 9.999999747378752e-05)
('I(V2)', -0.00019999999494757503)
('I(V1)', -0.00019999999494757503)
Problem: problem 1 has 4 MOSFET.
Is that correct?y/n

```

Fig. 4.3. Screenshot of the Variables Imported from LTspice

I() in Figure 4.3 stands for a current flowing through the circuit element between the parentheses, for examples, and Id(M) stands for a current flowing in the drain terminal for MOSFET (M).

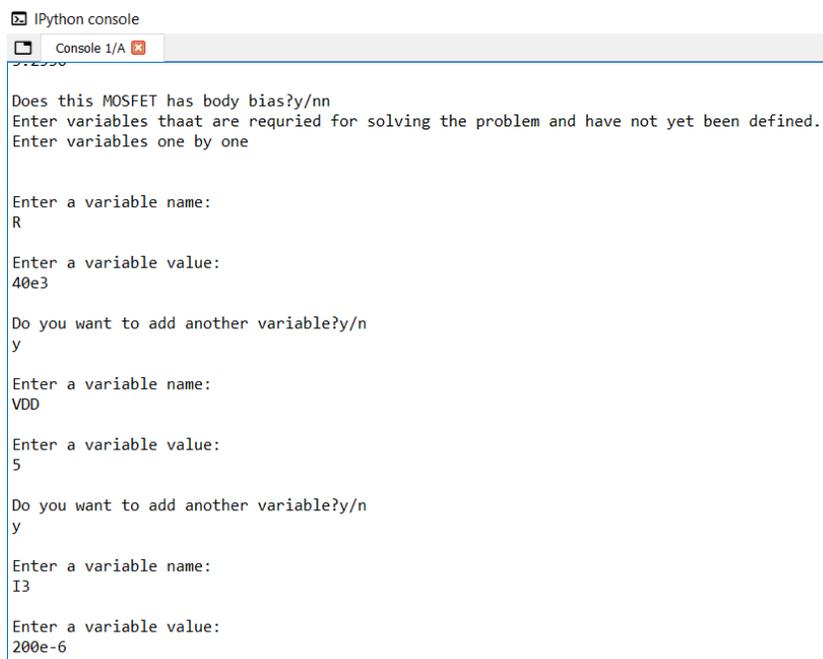
After the teacher confirms or modifies the number of MOSFETs in the problem, they are asked to enter the specifications for each MOSFET one by one, as shown in Figure 4.4. These specifications are: V_{t0} , K'_n , W/L , I_D , V_{GS} , V_{DS} , and body bias. If the MOSFET does have a body bias, further specification is sought to calculate V_t from V_{t0} .



```
IPython console
Console 1/A
Problem: problem 1 has 4 MOSFET.
Is that correct?y/n
y
Enter Vt for MOSFET 1
1
Enter KnP for MOSFET 1
100e-6
Enter WbyL for mosfet 1
20
Enter Id for MOSFET 1
100e-6
Enter VGS for MOSFET 1
1.3162
Enter VDS for MOSFET 1
2.3162
Does this MOSFET has body bias?y/nn
Enter Vt for MOSFET 2
```

Fig. 4.4. MOSFET Specification Entry by the Teacher

After defining the MOSFET specifications, the teacher is asked to enter any variable that is needed to solve the problem but has not yet been defined. An example is presented in Figure 4.5.



```
IPython console
Console 1/A x
Does this MOSFET has body bias?y/n
Enter variables that are required for solving the problem and have not yet been defined.
Enter variables one by one

Enter a variable name:
R

Enter a variable value:
40e3

Do you want to add another variable?y/n
y

Enter a variable name:
VDD

Enter a variable value:
5

Do you want to add another variable?y/n
y

Enter a variable name:
I3

Enter a variable value:
200e-6
```

Fig. 4.5. A Sample of Variables that were not entered as a MOSFET Specification

When the teacher indicates that there are no more variables to be defined, the console displays all the variables that have been defined so far and asks the teacher to specify which of them are “given” to the student at the beginning of the problem, as indicated in Figure 4.6.

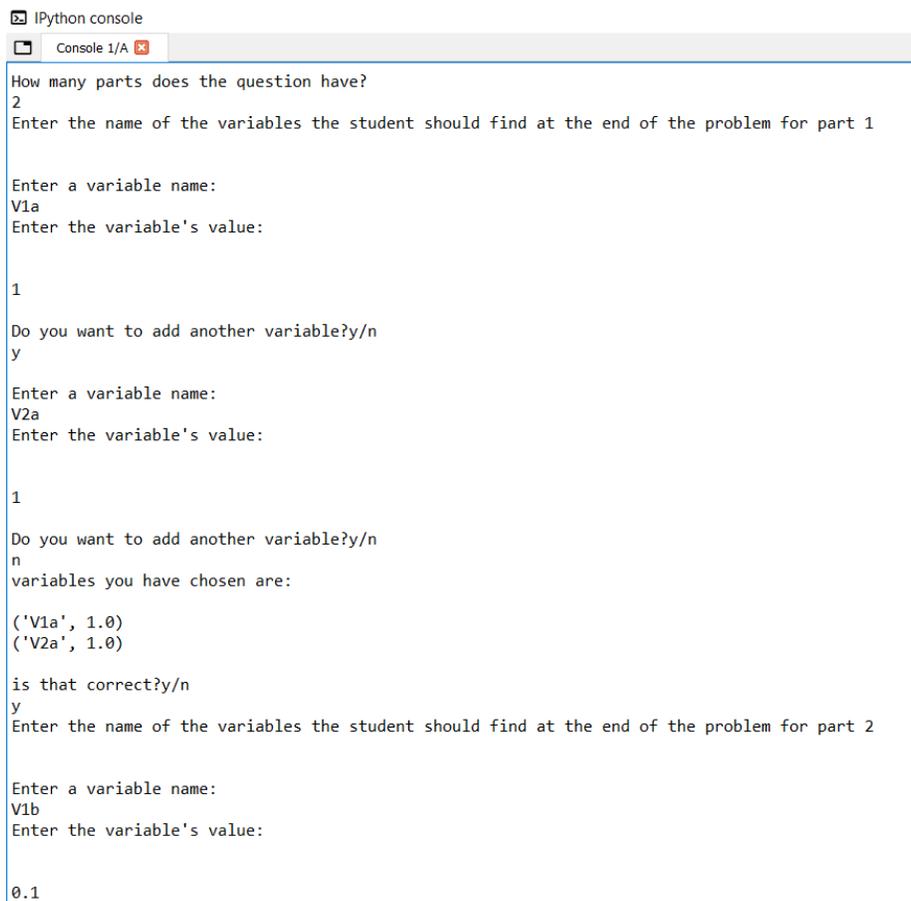
```
IPython console
Console 1/A
is that correct?y/n
y
The list of all variables you have entered:
('R', 40000.0)
('VDD', 5.0)
('I3', 0.0002)
('V3a', -1.3162)
('V3b', -1.3536)
('Id1', 0.0001)
('Id2', 0.0001)
('Id3', 0.0001225)
('Id4', 7.75e-05)
('VGS1', 1.3162)
('VGS2', 1.3162)
('VGS3', 1.3536)
('VGS4', 1.3536)
('VDS1', 2.3162)
('VDS2', 2.3162)
('VDS3', 1.4636)
('VDS4', 3.2536)
('KnP1', 0.0001)
('KnP2', 0.0001)
('KnP3', 0.0001)
('KnP4', 0.0001)
('WbyL1', 20.0)
('WbyL2', 20.0)
('WbyL3', 20.0)
('WbyL4', 13.33)
('Vov1', 0.31620000000000004)
('Vov2', 0.31620000000000004)
('Vov3', 0.35359999999999999)
('Vov4', 0.35359999999999999)
('Vt1', 1.0)
('Vt2', 1.0)
('Vt3', 1.0)
('Vt4', 1.0)
From the list of entered variables please identify the given variables
Enter the given variables name

R

Do you want to add another variable?y/n
y
Enter the given variables name
```

Fig. 4.6. List of Defined Variables Shown to the Teacher

Once that is done, the teacher is asked to identify how many parts the problem has; then, the teacher must define the variables that the student is required to calculate as illustrated in Figure 4.7 below. At the end, all these variables are saved as dictionaries in the problem databank, and the teacher's work is concluded.



```
IPython console
Console 1/A x
How many parts does the question have?
2
Enter the name of the variables the student should find at the end of the problem for part 1

Enter a variable name:
V1a
Enter the variable's value:

1

Do you want to add another variable?y/n
y

Enter a variable name:
V2a
Enter the variable's value:

1

Do you want to add another variable?y/n
n
variables you have chosen are:

('V1a', 1.0)
('V2a', 1.0)

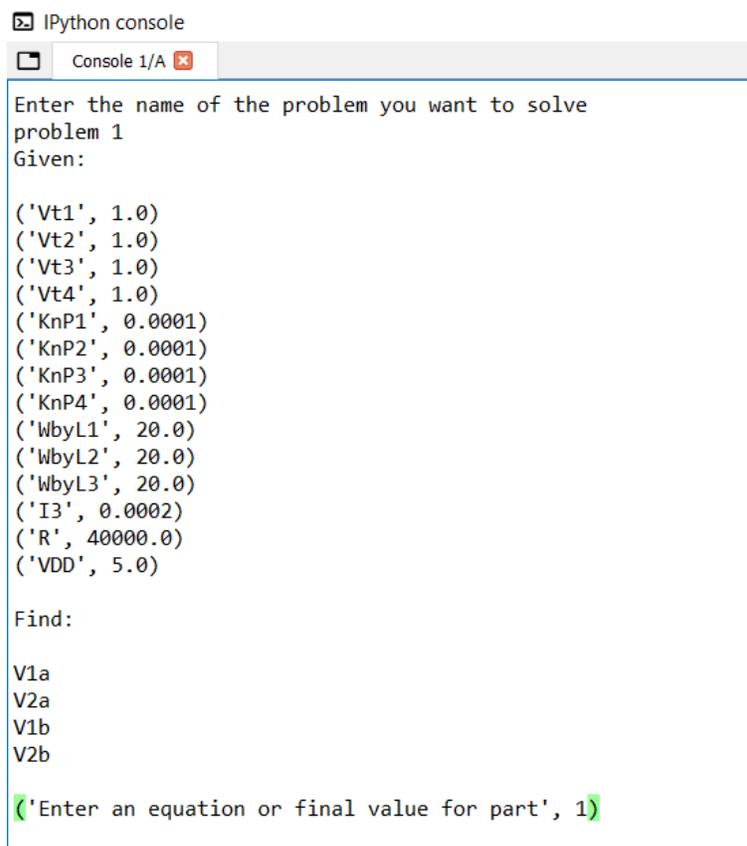
is that correct?y/n
y
Enter the name of the variables the student should find at the end of the problem for part 2

Enter a variable name:
V1b
Enter the variable's value:

0.1
```

Fig. 4.7. Defining the Variables that the Student Must Calculate

When the student runs the student module, they are asked to enter the name of the problem they want to solve. After that, the given and required variables are displayed, as shown in Figure 4.8.



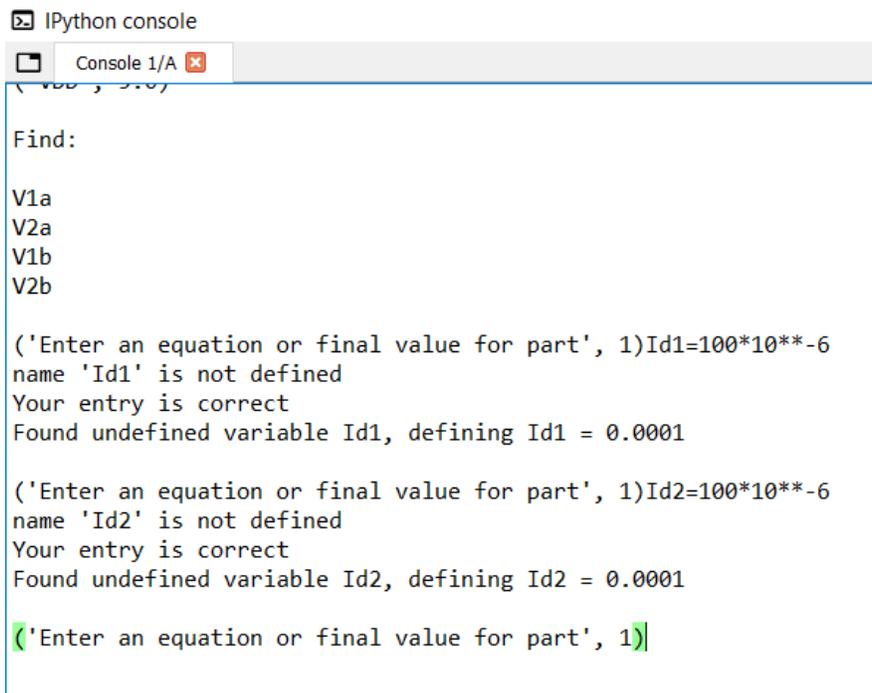
```
IPython console
Console 1/A
Enter the name of the problem you want to solve
problem 1
Given:
('Vt1', 1.0)
('Vt2', 1.0)
('Vt3', 1.0)
('Vt4', 1.0)
('KnP1', 0.0001)
('KnP2', 0.0001)
('KnP3', 0.0001)
('KnP4', 0.0001)
('WbyL1', 20.0)
('WbyL2', 20.0)
('WbyL3', 20.0)
('I3', 0.0002)
('R', 40000.0)
('VDD', 5.0)

Find:
V1a
V2a
V1b
V2b

Enter an equation or final value for part', 1)
```

Fig. 4.8. Given and Required Variables of Problem 1 as Shown to the Student

An experienced student will realize that $I_{D1} = I_{D2} = 100 \mu\text{A}$ for part 1, since both MOSFETs are identical so he will enter the equations $I_{D1} = 100 \mu\text{A}$ and $I_{D2} = 100 \mu\text{A}$. Because both I_{D1} and I_{D2} have not been given nor defined before, the console will send a confirmation message that these two variables have now been defined and their values can be used in other equations, as illustrated in Figure 4.9.



```
IPython console
Console 1/A
Find:
V1a
V2a
V1b
V2b

('Enter an equation or final value for part', 1)Id1=100*10**-6
name 'Id1' is not defined
Your entry is correct
Found undefined variable Id1, defining Id1 = 0.0001

('Enter an equation or final value for part', 1)Id2=100*10**-6
name 'Id2' is not defined
Your entry is correct
Found undefined variable Id2, defining Id2 = 0.0001

('Enter an equation or final value for part', 1)
```

Fig. 4.9. Defining I_{D1} and I_{D2}

Notice that Python 3 syntax of expressing power as `**` is used to be read correctly by the parser.

At this stage, the student inserts the value of I_{D1} and I_{D2} in the MOSFET I-V characteristic equation to solve for V_{GS1} and V_{GS2} . The I-V characteristic equations for a MOSFET that is operating in the triode region is given as:

$$I_D = K'_n * (W/L) * (V_{GS} - V_t - (0.5 * V_{DS})) * V_{DS} \quad (4.1)$$

Although both M_1 and M_2 are operating in the saturation region, the student may assume that they are operating in triode. The program will detect that error and provide the appropriate hint to the student as illustrated Figure 4.10.

```

IPython console
Console 1/A
name 'Id2' is not defined
Your entry is correct
Found undefined variable Id2, defining Id2 = 0.0001

('Enter an equation or final value for part', 1)Id1=KnP1*WbyL1*(VGS1-Vt1-(0.5*VDS1))*VDS1
name 'VGS1' is not defined
Too many undefined variables

('Enter an equation or final value for part', 1)

```

Fig. 4.10. Output produced by the ITS when there is more than one Unidentified Variable in the Student's Equation

The program detected that there are two unidentified variables, which are V_{GS1} and V_{DS1} ; hence, it cannot solve the equation. If the student defines one of these values, let's say V_{DS1} ,

then, the student enters the same equation of the triode region of operation again, the result will be as shown in Figure 4.11.

```

IPython console
Console 1/A ✖
Too many undefined variables
('Enter an equation or final value for part', 1)VDS1=2.3162
name 'VDS1' is not defined
Your entry is correct
Found undefined variable VDS1, defining VDS1 = 2.3162
('Enter an equation or final value for part', 1)Id1=KnP1*WbyL1*(VGS1-Vt1-(0.5*VDS1))*VDS1
Your answer is incorrect
Are you sure MOSFET 1 is in the triode region
('Enter an equation or final value for part', 1)

```

Fig. 4.11. Error Generated when Using Triode Equation Instead of Saturation

Now, assume the student has identified that the MOSFET is operating in the saturation region for which I-V characteristics equation is:

$$I_D = 0.5 * K'_n (W/L) * (V_{GS} - V_t)^2 \quad (4.2)$$

A few of the errors the student might commit while entering the correct equation include assuming K'_n is K_n , hence not multiplying by W/L , or using V_{GS} instead of the overdrive voltage. Both of these cases and their appropriate feedback are shown in Figure 4.12.

```
IPython console
Console 1/A x
('Enter an equation or final value for part', 1)Id1=0.5*KnP1*(VGS1-Vt1)**2
Your answer is incorrect

Multiply KnP by WbyL

('Enter an equation or final value for part', 1)Id1=0.5*KnP1*WbyL1*(VGS1)**2
name 'VGS1' is not defined
Your answer is incorrect

Use Vov instead of Vgs.
Vov=Vgs-Vt

('Enter an equation or final value for part', 1)
```

Fig. 4.12. Feedback for Errors in the I-V Characteristic Equations

When the student enters the correct equation, V_{GS1} will be defined; then, it can be used to find the value of V_{3a} ; also, V_1 and V_2 can be calculated from the following equation:

$$V_1 = V_2 = V_{DD} - (R * I_{D1}) \quad (4.3)$$

Figure 4.13 shows the student entering these equations and completing part 1 of the question.

```
IPython console
Console 1/A x

Use Vov instead of Vgs.
Vov=Vgs-Vt

('Enter an equation or final value for part', 1)Id1=0.5*KnP1*WbyL1*(VGS1-Vt1)**2
name 'VGS1' is not defined
Your entry is correct
Found undefined variable VGS1, defining VGS1 = 1.3162

('Enter an equation or final value for part', 1)V3a=-VGS1
name 'V3a' is not defined
Your entry is correct
Found undefined variable V3a, defining V3a = -1.3162

('Enter an equation or final value for part', 1)V1a=VDD-(R*Id1)
name 'V1a' is not defined
Your entry is correct
Found undefined variable V1a, defining V1a = 1.0

('Enter an equation or final value for part', 1)V2a=5-(40*10**3*Id2)
name 'V2a' is not defined
Your entry is correct
Found undefined variable V2a, defining V2a = 1.0

Well done!! Part 1 is done

('Enter an equation or final value for part', 2)
```

Fig. 4.13. Student Entering Correct Equations and Required Variables for Part 1

As illustrated in Figure 4.13, the program can evaluate both numerical and symbolic variables. Furthermore, it can detect when all the required variables for a certain part have been found.

In this and the subsequent sections, we will discuss part 2 of the problem. When students move to part 2 and try to apply the current equation for one of the MOSFETS in the saturation region as they did in part 1, they get a notification that their equation is correct; however, it has too many unknowns to be solved as shown in Figure 4.14.

```
IPython console
Console 1/A
Your entry is correct
Found undefined variable V2a, defining V2a = 1.0

Well done!! Part 1 is done

('Enter an equation or final value for part', 2)Id3=0.5*KnP3*WbyL3*(VGS3-Vt3)**2
name 'Id3' is not defined
Your entry is correct
Too many undefined variables

('Enter an equation or final value for part', 2)
```

Fig. 4.14. The Error Message the Student Receives when Entering an Equation with too many Unknown Variables

The student should be able to apply Kirchhoff's Current Law at node V3 and deduce that:

$$I_{D1} + I_{D2} = 200 \mu\text{A} \quad (4.4)$$

Then, the student can apply the MOSFET current equation on both I_{D3} and I_{D4} , and substitute both V_{GS3} and V_{GS4} with $-V_{3b}$. After inserting the resulting equation, the program will solve for V_{3b} as illustrated Figure 4.15.

```

IPython console
Console 1/A
('Enter an equation or final value for part', 2)(0.5*100*10**-6*WbyL3*(-V3b-1)**2)+(0.5*KnP4*13.33*(-V3b-1)**2)=200*10**-6
name 'V3b' is not defined
Your entry is correct
Found undefined variable V3b, defining V3b = -1.3536
('Enter an equation or final value for part', 2)

```

Fig. 4.15. Applying Kirchhoff's Current Law to Find V_{3b}

After that, the student can find both I_{D3} and I_{D4} for the MOSFET current equation as in Figure 4.16.

```

IPython console
Console 1/A
('Enter an equation or final value for part', 2)(0.5*100*10**-6*WbyL3*(-V3b-1)**2)+(0.5*KnP4*13.33*(-V3b-1)**2)=200*10**-6
name 'V3b' is not defined
Your entry is correct
Found undefined variable V3b, defining V3b = -1.3536

('Enter an equation or final value for part', 2)Id3=0.5*KnP3*WbyL3*(-V3b-1)**2
Your entry is correct
Found undefined variable V3b, defining V3b = -1.3536

('Enter an equation or final value for part', 2)Id4=0.5*KnP4*WbyL4*(-V3b-1)**2
name 'Id4' is not defined
Your entry is correct
Found undefined variable Id4, defining Id4 = 7.75e-05
('Enter an equation or final value for part', 2)

```

Fig. 4.16. Finding I_{D3} and I_{D4}

At the end, the student will find V_{1b} and V_{2b} by applying Ohm's Law, as demonstrated in Figure 4.17.

```
IPython console
Console 1/A
Found undefined variable Id4, defining Id4 = 7.75e-05

('Enter an equation or final value for part', 2)V1b=5-(R*Id3)
name 'V1b' is not defined
Your entry is correct
Found undefined variable V1b, defining V1b = 0.1

('Enter an equation or final value for part', 2)V2b=5-(R*Id4)
name 'V2b' is not defined
Your entry is correct
Found undefined variable V2b, defining V2b = 1.9

Well done!! Part 2 is done
Well done. You have solved the problem!

In [2]: |
```

Fig. 4.17. Finding the Final Values for Problem 1

As shown Figure 4.17, the program recognizes that all required variables have been correctly found and the session is terminated.

4.2 Second Example Problem

A second example problem is chosen to test how well the program can handle PMOS. The problem is example 4.7 on page 380 of the Sedra/Smith sixth International edition [41]. The problem statement is illustrated in Figure 4.18.

Design the circuit of Figure 4.25 so that the transistor operates in saturation with $I_D = 0.5\text{mA}$ and $V_D = +3$. Let the enhancement-type PMOS transistor have $V_t = -1\text{V}$ and $K'_p(W/L) = 1\text{mA/V}^2$ assume $\lambda = 0$. What is the largest value that R_D can have while maintaining saturation region operation?

Fig. 4.18. Second Example Problem

To test if the program can detect confusion between K_p and K'_p for a PMOS transistor, it is assumed that $K'_n = 100 \mu\text{A/V}^2$ and $W/L = 10$.

First, we run the teacher module to setup the problem. In the ITS, the problem's name is "problem 2". Figure 4.19 shows the circuit to be simulated by LTspice.

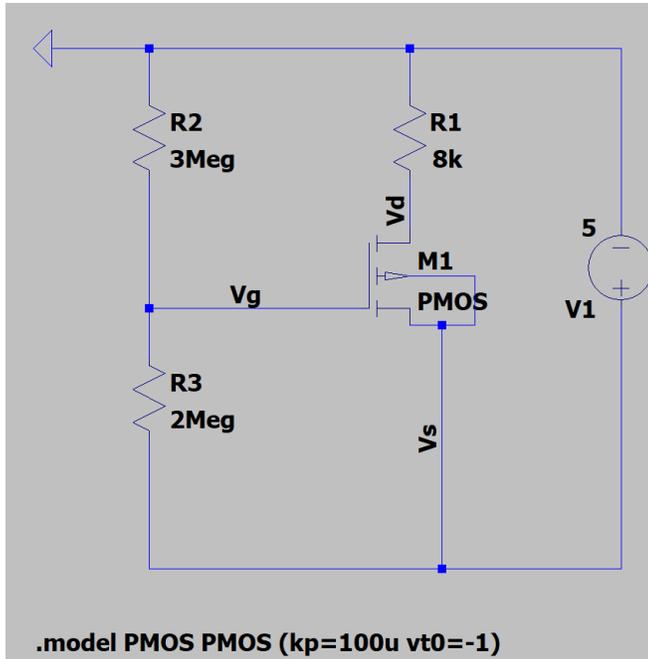


Fig. 4.19. Circuit Schematic for Problem 2 as Entered into LTspice

Figure 4.20 shows that the program has successfully identified that the circuit has one MOSFET.

```
IPython console
Console 1/A
In [1]: run_line_cell('0.7/mes132/code/spice/teacher_module.py', wait='0.7/mes132/code/spice/')

Enter the name of the problem you want to add:
problem 2
LtSpice will run Please draw and simulate the problem you would like the student to solve

Enter the name and path of the raw file
problem 2.raw
variables obtained from ltspice are:

('V(vd)', 4.000000016079998)
('V(vg)', 3.0)
('V(vs)', 5.0)
('Id(M1)', -0.0005000000237487257)
('Ig(M1)', -0.0)
('Ib(M1)', 1.0100000263219e-12)
('Is(M1)', 0.0005000000237487257)
('I(R3)', -9.999999974752427e-07)
('I(R2)', -9.999999974752427e-07)
('I(R1)', -0.0005000000237487257)
('I(V1)', -0.0005009999731555581)
Problem: problem 2 has 1 MOSFET.
Is that correct?y/n
.
```

Fig. 4.20. Teacher Module Screen for Problem 2

Now, let us test how well the program identifies errors in the MOSFET current equation for a PMOS. This is demonstrated in Figure 4.21.

```

Python console
Console 1/A
Enter the name of the problem you want to solve
problem 2
Given:
('VDD', 5.0)
('Vd', 3.0)
('Id1', 0.0005)
('KnP1', 0.0001)
('WbyL1', 10.0)
('Vt1', -1.0)

Find:
Rd

('Enter an equation or final value for part', 1)Id1=0.5*KnP1*(VGS1-Vt1)**2
name 'VGS1' is not defined
Your answer is incorrect

Multiply KnP by WbyL

('Enter an equation or final value for part', 1)Id1=0.5*KnP1*WbyL1*(VGS1)**2
name 'VGS1' is not defined
Your answer is incorrect

Use Vov instead of Vgs.
Vov=Vgs-Vt

('Enter an equation or final value for part', 1)VGS1=-2
name 'VGS1' is not defined
Your entry is correct
Found undefined variable VGS1, defining VGS1 = -2.0

('Enter an equation or final value for part', 1)Id1=KnP1*WbyL1*(VGS1-Vt1-(0.5*VDS1))*VDS1
name 'VDS1' is not defined
Your answer is incorrect

Are you sure MOSFET 1 is in the triode region

('Enter an equation or final value for part', 1)

```

Fig. 4.21. Student Module Identifying Errors in a PMOS Equation

As can be seen in Figure 4.21, the program successfully identified the errors in the MOSFET current equation for a PMOS.

4.3 Third Example Problem

The third problem is a NMOS-based circuit that is designed by the writer to test the program for identifying the body bias and triode region of operation. The circuit schematic is shown in Figure 4.22.

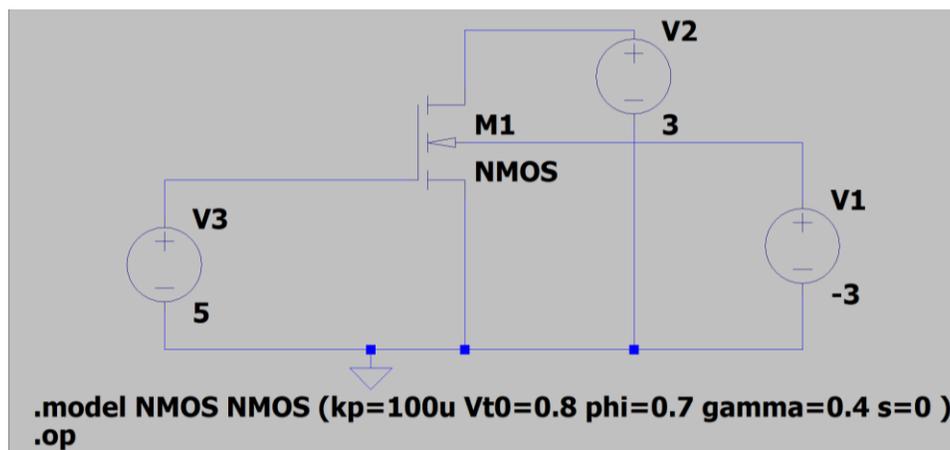


Fig. 4.22. Circuit Schematic for Problem 3 as Entered into LTspice

The problem statement is given in Figure 4.23.

Find I_d for the circuit shown in Figure 4.22 given that $K'_n = 100 \mu\text{A}/\text{V}^2$, $W/L = 10$, $V_{t0} = 0.8\text{V}$, $\gamma = 0.4$, and $2\phi_f = 0.7$

Fig. 4.23. Statement of the Third Example Problem

Figure 4.24 shows the program giving the appropriate hint to a student who used the value of V_{t0} instead of V_t . Furthermore, the program correctly identifies that the student used the equation for the saturation region instead of triode.

```

IPython console
Console 1/A
Enter the name of the problem you want to solve
problem 3
Given:

('VGS1', 5.0)
('VDS1', 3.0)
('WbyL1', 10.0)
('KnP1', 0.0001)
('Vt01', 0.8)
('VBS1', 3.0)
('gama1', 0.4)
('phiF1', 0.7)

Find:

Id1

('Enter an equation or final value for part', 1)Id1=KnP1*WbyL1*(VGS1-0.8-(0.5*VDS1))*VDS1
name 'Id1' is not defined
Your answer is incorrect

Make sure to to add the body effect for MOSFET 1

('Enter an equation or final value for part', 1)Id1=0.5*KnP1*WbyL1*(VGS1-1.2346)**2
name 'Id1' is not defined
Your answer is incorrect

Are you sure MOSFET 1 is in the saturation region

('Enter an equation or final value for part', 1)Id1=KnP1*WbyL1*(VGS1-1.2348-(0.5*VDS1))*VDS1
name 'Id1' is not defined
Your entry is correct
Found undefined variable Id1, defining Id1 = 0.0068

Well done!! Part 1 is done
Well done. You have solved the problem!

In [9]: |

```

Fig. 4.24. Feedback for Problem 3

In this chapter, three different MOSFET problems were tested. These problems were chosen to test how the program handles different types of MOSFETs running in different operation regions.

Chapter 5

Conclusions and Future Work

In this thesis, a new ITS was designed and implemented for microelectronics circuits featuring MOSFET circuit elements. The proposed ITS program allows the teachers to add and simulate simple MOSFET circuits and specify the variables for students to solve. The students, on the other hand, have the ability to choose a problem from the databank to solve and enter equations that work towards solving it. The students receive immediate feedback on the correctness of their equations and some hints if their entry is wrong; moreover, they have the chance of committing one error concerning the equation of MOSFET current-voltage characteristic equation.

Three example problems have been presented to illustrate the applicability, capability, and effectiveness of the proposed ITS program in solving problems and giving appropriate feedback to the users.

As for future work, it is proposed to implement a graphical user interface (GUI) that I strongly believe that it will be an excellent addition to the presented ITS. As illustrated in

Chapter 4, the existing program only supports a textual interface. Adding a GUI to the ITS will make it easier to handle, user friendly, and more appealing to the students.

Secondly, I would like to build our own circuit analyser and problem solver. This will add many new benefits to the program and enhances its capability. An example of the new features would be to use the equations generated by the problem solver as a base for comparison with the students' equations. This comparison with the use of AI will help the program to detect more than one error per equation, as well as giving it the ability to give more sophisticated and accurate hints and feedback. Furthermore, the addition of a circuit analyser will make it easier to expand the program so that it allows the addition of problems with circuit elements other than MOSFETs, such as bipolar junction transistors and diodes. Last, but not least, the circuit analyser will facilitate the addition of open-ended design problems, as it gives the program the ability to fully analyse and simulate the students' circuits.

Finally, I plan to develop a student module that uses Bayesian networks to track the learning progress of each student individually and give feedback tailored specifically to the knowledge and understanding of the student.

The results of this thesis have been submitted in a paper to the International Conference on Information, Intelligence, Systems and Applications (IISA 2021).

References

- [1] P. Phobun and J. Vicheanpanya, "Adaptive intelligent tutoring systems for e-learning systems," *Procedia - Social and Behavioral Sciences*, vol. 2, no. 2, pp 4064-4069, (Dec.) 2010.
- [2] J. Dabolins and J. Grundspenkis, "The role of feedback in intelligent tutoring system," *Lietiskas datorsistemas*, vol. 14, pp. 88-93, 2013.
- [3] J. R. Carbonell, "AI in CAI: an artificial intelligence approach to computer-assisted instruction," *IEEE Transactions on Man-Machine Systems*, vol. 11, pp. 190-202, (Dec.) 1970.
- [4] H. S. Nwana. (1990), "Intelligent tutoring systems: an overview," *Artificial Intelligence Review*, vol. 4, pp. 251-277, (Dec.) 1990.
- [5] J. S. Brown, R. R. Burton, and A. G. Bell, "SOPHIE: a step towards a reactive learning environment," *International Journal of Man-Machine Studies*, vol. 7, pp. 675-696, 1975.
- [6] W. J. Clancey, *Knowledge-Based Tutoring*, MIT Press, London, 1987.

- [7] E. H. Shortliffe, *Computer Based Medical Consultations: MYCIN*. Elsevier, New York, 1976.
- [8] J. C. Lester and B. A. Stone, “Increasing believability in animated pedagogical agents,” In *Proceedings of the First International Conference on Autonomous Agents*, pp. 16-21, (Feb.) 1997.
- [9] A. L. Baylor, “Multiple intelligent mentors instructing collaboratively (MIMIC): developing a theoretical framework,” *Proceedings of International Cognitive Technology Conference*, San Francisco, CA., 1999.
- [10] W. Canfield, “ALEKS: a web-based intelligent tutoring system,” *Mathematics and Computer Education*, vol. 35, no. 2, pp. 152-158, (June) 2001.
- [11] <https://www.aleks.com/>
- [12] R. Conejo, E. Guzman, E. Millán, M. Trella, J. L. Pérez-de-la Cruz, and A. Rios, “SIETTE: a web-based tool for adaptive teaching,” *International Journal of Artificial Intelligence in Education*, vol. 14, pp. 29–61, 2004.
- [13] C. Hockemeyer, T. Held, and D. Albert, “RATH—a relational adaptive tutoring hypertext WWW–environment based on knowledge space theory,” C. Alvegard, editor. In *Proceedings of fourth International Conference on Computer Aided*

- Learning and Instruction in Science and Engineering, Proceedings of the CALISCE' 98*, Göteborg, Sweden, pp. 417–423, 1998.
- [14] P. Brusilovsky, “Adaptive and intelligent web-based educational systems,” *International Journal of Artificial Intelligence in Education*, vol. 13, pp.156-169, 2003.
- [15] G. Weber and P. Brusilovsky, “ELM-ART: An adaptive versatile system for web-based instruction,” *International Journal of Artificial Intelligence in Education*, vol. 12, no. 4, pp. 351-384, 2001.
- [16] P. Brusilovsky, E. Schwarz, and G. Weber, “ELM-ART: An intelligent tutoring system on World Wide Web,” In C. Frasson, G. Gauthier, and A. Lesgold Editors, *Third International Conference on Intelligent Tutoring Systems, ITS-96*, vol. 1086, pp. 261-269, Berlin: Springer Verlag, 1996.
- [17] P. Brusilovsky, E. Schwarz, and G. Weber, “A tool for developing hypermedia-based ITS on WWW,” *Proceedings of workshop “architectures and methods for designing cost-effective and reusable ITSs,” Third International Conference on Intelligent Tutoring Systems, ITS-96*, Montreal, Canada, June 12-14, 1996.
- [18] P. M. E. De Bra, “Teaching hypertext and hypermedia through the Web,” *Journal of Universal Computer Science*, vol. 2, no. 12, pp. 797-804, 1996.

- [19] L. S. Myneni, N. H. Narayanan, S. Rebello, A. Rouinfar, and S. Pamtambekar, "An interactive and intelligent learning system for physics education," *IEEE Transactions on Learning Technologies*, vol. 6, no. 3, pp. 228-239, (Jul. - Sept.) 2013.
- [20] P. Jordan, M. Makhatchev, U. Pappusamy, K. Van Lehn, and P. Albacete, "A natural language tutorial dialogue system for physics," In *Proceedings of FLAIRS 2006*. Menlo Park, CA: AAAI Press, 2006.
- [21] J. Beck, M. Stern, and E. Haugsjaa, "Applications of AI in education," *Crossroads*, vol. 3, no. 1, pp. 11-15, 1996.
- [22] P. Holt, S. Dubs, M. Jones, and J. Greer, "The State of student modelling," In *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, J. Greer and G. McCalla, editors, Springer-Verlag, New York, pp. 3-39, 1994.
- [23] K. Van Lehn, C. Lynch, K. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill, "The Andes physics tutoring system: Lessons learned," *International Journal of Artificial Intelligence in Education*, vol. 15, no. 3, pp. 147-204, 2005.
- [24] A. S. Gertner, C. Conati, and K. VanLehn, "Procedural help in Andes: generating hints using a Bayesian network student model," In *Proceedings of National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, pp. 106-111, 1998.

- [25] C. Liew, J. A. Shapiro., and D. Smith, “Reasoning about algebraic answers in physics,” In *Proceedings Twelfth International Florida AI Research Society Conference*, pp. 167-171, 1999.
- [26] A. Gertner, “Providing feedback to equation entries in an intelligent tutoring system for physics,” B. Goettl, H. Halff, C. Redfield and V. Shue, editors, In *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems, ITS’98*, Springer-Verlag, Berlin, pp. 254-263, 1998.
- [27] C. W. Liew, and D. E. Smith, “Reasoning about systems of physics equations,” In Cerri, Gouarderes and Paraguacu, editors, *Intelligent Tutoring Systems*, 2002.
- [28] A. Yoshikawa, M. Shintani, and Y. Ohba, “Intelligent tutoring system for electric circuit exercising,” *IEEE Transactions on Education*, vol. 35, no. 3, pp. 222-225, (Aug.) 1992.
- [29] B. P. Butz, M. Duarte, and S. M. Miller, “An intelligent tutoring system for circuit analysis,” *IEEE Transactions on Education*, vol. 49, no. 2, pp. 216-223, (May) 2006.
- [30] A. C. Graesser, X. Hu, B. D. Nye, *et al.* “ElectronixTutor: an intelligent tutoring system with multiple learning resources for electronics,” *International Journal of STEM Education*, vol. 5, no. 15, (April) 2018.
- [31] The Andes Physics Tutor.<http://www.andestutor.org/> Accessed March 30, 2021.

- [32] Person Mastering Physics,
<https://www.pearsonmylabandmastering.com/northamerica/masteringphysics/>
Accessed April 22,2021
- [33] App Store Preview, Socratic by Google <https://apps.apple.com/ca/app/socratic-by-google/id1014164514> Accessed April 22, 2021
- [34] App Store Preview, Darisni, <https://apps.apple.com/kw/app/darisni-%D8%AF%D8%B1%D8%B3%D9%86%D9%8A/id1103683682> Accessed April 22, 2021
- [35] Khan Academy, <https://www.khanacademy.org/> Accessed April 22, 2021
- [36] App Store Preview, GoLearningBus University
<https://apps.apple.com/us/app/golearningbus-university/id1037335675> Accessed
April 22, 2021
- [37] App Store Preview, Basic Physics-Formulas, <https://apps.apple.com/us/app/basic-physics-formulas/id880245410> Accessed April 22, 2021
- [38] App Store Preview, iPhysics <https://apps.apple.com/ca/app/iphysics/id592372510>
Accessed April 22, 2021
- [39] Memrise, <https://www.memrise.com/apps/> Accessed April 22, 2021

- [40] Duolingo, <https://www.duolingo.com/> Accessed April 22, 2021
- [41] A. S. Sedra and K. C. Smith, *Microelectronics Circuits: Theory and Application*, International Sixth Edition, Oxford University Press, USA, 2011.
- [42] T. Lehmann, <http://www2.ee.unsw.edu.au/~tlehmann/ltspy3.py> Accessed April 22, 2021