A Tool for Facilitating the Automated Assessment of Engineering/Science Courses

Journal:	*The International Journal of Electrical Engineering & Education*
Manuscript ID	MIJEEE-2019-0443.R2
Manuscript Type:	Original Article
Date Submitted by the Author:	n/a
Complete List of Authors:	Beg, Azam; United Arab Emirates University, Computer & Network Alhemeiri, Mouza ; UAE University Beg, Ajmal; Cortex Business Solutions
Keywords:	Massive open online course (MOOC), automated assessment, engineering education, electrical engineering, logic schematics

A Tool for Facilitating the Automated Assessment of Engineering/Science Courses

Journal Title
XX(X):2–??

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Azam Beg[1], Mouza Alhemeiri[1], and Ajmal Beg[2]

Abstract

In recent years, massive open online courses (MOOCs) have become quite popular. Such courses are either completely free or cost nominally. Generally, the MOOCs face the challenge of not being recognized as 'regular' courses (i.e., the ones taken at the traditional learning institutions). One of the main reasons for this lack of acceptability is the assessment in an unsupervised environment, which is prone to the problems of test-taker's online lookups or interaction with others for finding the answers to the test questions. A few ways of alleviating this problem include: limiting the time for answering the questions, the avoidance of repetitive questions, and the creation of a large number of questions.

This paper presents a tool named QAgen that enables the automatic creation of a large number of questions and answers related to different topics in computer/electrical engineering (ECE), computer science, physics, etc. Specifically, the tool is related to the courses on digital logic design, computer architecture, etc. The generated questions are in a format that is suitable both for learning management system (LMS) based and/or non-LMS-based assessment in conventional courses or MOOCs. The proposed tool is based on open-source software, thus eliminating the need for any commercial software packages. The underlying principles of QAgen are applicable to other engineering/science courses as well, if the assessment methods require the creation of some connected-object diagrams, tables and equations.

For assessing the usefulness of QAgen, practice question sets were created for three different courses. The student surveys for these courses indicated that the questions helped students prepare for actual tests/examinations. Especially favored by the students was the availability of correct answers at the end of each practice test/examination.

Keywords
Massive open online course (MOOC), engineering education, distance learning, e-learning, automated assessment, test-bank, logic design, circuit schematics.

Introduction

In the past few years, massive open online courses (MOOCs) have gained significant popularity. Such courses are either completely free or cost very little. Due to the very nature of the MOOCs, a learner is not constrained by time or physical space. A reasonably-sized MOOC is expected to be taken by the students with a variety of strengths and weaknesses, learning styles, and cultural and educational backgrounds. For such students, the *adaptive learning* methodology is particularly useful as it offers them efficient and customized paths to learning, especially in the MOOC settings [Hao and Huiyan (2018), Lu et al. (2018)].

The learning outcomes of a MOOC can be assessed using a set of formative and/or summative tests [Ip et al. (2018), Cross et al. (2019)]. Currently, hundreds of MOOCs are available through different platforms, but they still face the challenge of not being recognized as 'regular' courses (i.e., the ones taken at traditional learning institutions) [Banks (2016)]. One of the main reasons for this lack of acceptability is the course assessment in an unsupervised environment being prone to the problem of an examinee seeking help from off-line and/or online sources during an examination/test [De Rosa and Pistolese (2019)]. Although, at present, a few companies (for example, ProctorU, Examity, ProctorFree, etc.) offer online, video-based proctoring solutions, their wide-spread use may remain limited due to the scalability and the cost issues [Soltani et al. (2019)]. Other possible solutions include: constricting the test times, not repeating test questions, using a large test question database, etc [Posner (2020)]. For many courses related to science or engineering, even a moderately-sized set of questions can be turned into a large question database by varying the test question data within pre-specified ranges. Such an *expanded* database can be instrumental in facilitating the MOOC assessment.

Presently, many electrical/computer engineering (ECE) courses are offered in the MOOC format [Cou (2020), Uda (2020)]. The automated assessment of such courses can be enhanced significantly by using large databases comprising *unique* questions-and-answers – doing so is the aim of our current work.

The rest of this paper is organized as follows: In the first section (Literature Review), we review the work related to the test creation and test administration. The section also includes the work related to the drawing of logic/circuit diagrams (a key component

[1]United Arab Emirates University, Al-Ain, United Arab Emirates
[2]Cortex Business Solutions, Calgary, Canada

Corresponding author:
Azam Beg
Email: abeg@uaeu.ac.ae

Prepared using sagej.cls *[Version: 2017/01/17 v1.20]*

of the assessment instruments of many ECE courses). The next section provides the implementation details of QAgen, the proposed enabling tool for automated assessment. The QAgen algorithms (primarily for schematic creation) are also explained in this section. Additionally, a few examples of the QAgen usage and application are provided. The last section concludes the paper.

Literature Review

Test Creation and Test Administration

The test creation and the test administration are essential parts of any academic assessment process. The tests can be generated statically by using *integer linear programming* (using multiple assessment variables) [Nguyen and Fong (2013)] or by using the *semantic*-based method [Miranda et al. (2013)]. Whereas the dynamic test creation can be done using the *bi-proportional matrix scaling* technique [Paul and Pawar (2013)]. The integration of different tools for generating the question sets for high-school subjects was proposed in [Bednarik and Kovacs (2012)]. A multi-year project on Internet-based learning and evaluation tool for the skill improvement of the freshman and sophomore years of a university was presented in [Smaill (2005)]. In order to automatically create test sheets for the candidates taking the Examination System of Electrical Energy Measurement, a *fishnet* algorithm was proposed in [Yuan-Bin and Jie (2012)]. The use of a concept inventory for digital logic courses was proposed in [Herman and Handzik (2010)] and [Herman and Loui (2011)], but without the details of the generation of question-and-answer sets. A genetic algorithm was used for test question creation in [Liu et al. (2010)], however, the answer generation was not addressed. The *adaptive learning* technique can create efficient and customized learning paths for the MOOC students. The adaptive learning directly benefits from artificial intelligence techniques, such as data mining, machine learning, and predictive analytics [Lu et al. (2018)]. In order to facilitate the LMS-based courses, *learning object repositories* can be used [Nascimento et al. (2013)]. The costs of design, delivery and assessment for such courses can be reduced by using *software agents* [Bassi et al. (2014)]. A learning analytic tool [Al-Ashmoery and Messoussi (2015)] can also be deployed for course instructors to study the interaction patterns of online students. Ahmed et al. proposed a scheme for grading text answers to digital design course comparing a known set of keywords and phrases in the correct answers with the students' using cosine similarity scores [Ahmed et al. (2018)]. In order to stimulate creativity and motivation among students, Oliveira et al. combined online peer assessment with scoring rubrics to allow students to assess their peers' work [Oliveira et al. (2018)].

Drawing Circuit Schematics

At present, a variety of university-level ECE courses are offered as MOOCs [Cou (2020), Uda (2020)]. The assessment instruments in many of these courses require the drawing of circuit/logic diagrams/schematics [Chen (2011), Rahman and Ogunfunmi (2010)]. For

use in the assessment instruments, the auto-generation of the diagrams in *bulk* is one of the daunting challenges that is yet to be tackled.

Over the years, the drawing of the logic diagrams has been addressed [Brennan (1975), Arya et al. (1985), Aleman and Couleur (1990), Chiueh (1991), Naveen and Raghunathan (1993), Lageweg (1998), Kim et al. (2000)], however, none of the presented techniques can be utilized in the MOOC-assessment process, mainly due to the problem of scalability. The circuit diagram tool presented in [Beg and Beg (2016), Beg and Beg (2018)] is limited to the analog domain. The logic-circuit-related open-source tools include: ChipVault [Chi (2020)], Icarus [Ica (2020)], QFlow [QFl (2020)], Verilator [Ver (2020)], and Yosys [Yos (2020)]. ChipVault and Icarus synthesize logic but do not create logic diagrams. QFlow and Verilator are other tools for logic synthesis *sans* logic diagrams. Yosys is also primarily used for logic synthesis; the tool creates block diagrams instead of logic diagrams.

Two other non-commercial (open-source) tools for the *manual* drawing of circuit diagrams are SmartDraw [Sma (2020)] and KTechLab [KTe (2020)]. Even though a few commercial products (for example, Cadence [Cad (2020)], Concept Engineering [Con (2020)], Silvaco [Sil (2020)] and Zuken [Zuk (2020)]) generate logic diagrams, these are not suitable for academic assessment because the symbols in the diagrams are labeled with details (labels, sizes, etc.) which are only required for industrial circuit design. The other major issue is the heavy licensing/administration cost of these commercial products.

QAgen – A Tool for Automatically Generating Questions and Answers

None of the literature known to the authors presents a technique for automatically generating large sets of questions and the related diagrams; this points to the need for a method for creating not just the test questions but also the associated diagrams, especially for the purpose of MOOC assessment.

This paper proposes an automated tool that helps improve the automated assessment of engineering/science courses. The tool enables the creation of a large number of questions and answers related to many topics in computer/electrical engineering, computer science, etc. The examples of the covered engineering courses include: digital logic design, hardware description language (HDL) based design, computer architecture, digital circuit testing, etc.

The proposed question-and-answer generation tool named ‘QAgen’, is based on Octave, an open-source scripting language (compatible with Matlab) for scientific applications. Being a commercial product, Matlab software and toolboxes carry hefty licensing cost for non-students. As an example, QAgen automatically creates both the questions and their answers related to these topics in a logic design course: analysis of combinational logic circuits, analysis of sequential logic circuits, design of combinational logic circuits, and design of sequential logic circuits. The specific topics in such a course include: number systems, digital building blocks (gates, decoders, multiplexers, etc), Boolean functions and their simplification, etc. Thousands of *unique*

question-answer sets can be produced within minutes by running QAgen on an ordinary computer. Depending on the topics being assessed, the question creation may require one or more of these steps: automatic generation of tables of varying sizes (truth tables and Karnaugh maps), creation of properly formatted Boolean equations with multiple variables, and the *drawing* of logic diagrams representing arbitrary Boolean functions. Among these steps, the automated drawing of diagrams is significantly harder than the creation of question text, tables or Boolean equations.

QAgen is fully configurable and allows an examiner to customize the questions and answers by setting these parameters: (a) type of circuit elements – gates, decoders, multiplexers, etc; (b) number of circuit inputs/variables; (c) number of circuit outputs; (d) number of gates, decoders, multiplexers, etc; (e) circuit depth (for gate-only circuits); (f) types of gates; and (h) question format – multiple-choice, fill-in-the-blank, etc.

QAgen generates all of the question-related 'outputs', i.e., the pieces of code for: (a) creating the complete question text including Boolean equation, truth table, etc., (b) drawing any related circuit schematic, and (c) finding the answer for grading purpose.

The LaTex format of the QAgen outputs is suitable for both the traditional paper-based testing and the LMS-based testing for MOOCs. For use with LMSes, the open-source tools (for example, LaTeXML) can be used to easily convert the LaTex into the web-friendly HTML or XML formats; such text-based formats allow bulk uploading of large number of questions-and-answers into the well-known LMSes, such as Moodle and Blackboard.

QAgen can be readily used for test-generation for multiple courses offered by the Departments of Computer Engineering and Electrical Engineering at our university, for instance: Digital Design & Computer Organization, Computer Architecture, and Digital System Design.

QAgen Algorithms

This section presents a set of algorithms for generating random Boolean functions and related circuit descriptions (i.e., Verilog models), in a speedy fashion. A few examples and use-cases are also included. The algorithms can be coded in any web-friendly language, such as, Java, Perl, PHP, Python, etc. The algorithms serve these core functions [Beg et al. (2017)] (see Appendix A):

1. Creating a random Boolean equation
2. Representing the Boolean equation in reverse polish notation (RPN)
3. Creating Verilog code from the RPN
4. Drawing a logic diagram representing the Verilog code

The creation of *random* Boolean equations is constrained by a set of user-specified parameters. The user can specify the complexity of a logic function/circuit in terms of number of input variables, number of logic gates, and number of layers/levels. The *allowed* logic functions can also be specified, i.e., INV, AND, OR, NAND, NOR, XOR,

and XNOR. Being able to specify the complexity of logic functions helps tailor the questions to the level of difficulty of a course or a topic. (Refer to the pseudo-code for Algorithm-1 in Appendix A).

A Boolean equation created by Algorithm-1 needs to be *pre-processed* for use by other components of QAgen. To do so, the widely-known RPN representation is used. The common RPN algorithms deal only with arithmetic functions and not with Boolean functions. In our case, an issue to contend with was the identification and processing of multi-input '*dual*' functions, i.e., NAND (applying NOT to the ANDed variables), NOR or XNOR. Therefore, we had to come up with a modified RPN algorithm to process logic operations, both *singular* and *dual*. (Refer to the pseudo-code for Algorithm-2 in Appendix A).

An RPN-formatted Boolean question facilitates the process of Verilog code creation. Firstly, the string array representing the RPN-equation is parsed to create an array that contains: type of logic gate, set of gate input signals, and gate output. The *internal* wires are also identified. Then, the Verilog-file creation starts by generating the required 'module' command and the set of 'input,' 'output,' and 'wire' declarations. After this step, all gate declarations are made in the proper sequence: type (AND, OR, etc.), label (G1, G2, etc.), output and input nets. Lastly, the 'endmodule' keyword is added before saving the ready-to-use Verilog file. (Refer to the pseudo-code for Algorithm-3 in Appendix A).

At the end, a logic diagram is generated from the Verilog code. To do so, firstly, the circuit inputs and outputs are parsed and then the gate descriptions (i.e., gate name, gate's input and output labels, etc) are parsed. The circuit input nodes are drawn first, followed by the gates driven only by the circuit inputs. The remaining gates are staggered (i.e., diagonally) to simplify the wiring. For aesthetic purposes, the output gates are *shifted up*. The wirings of the inputs are done first and then the *internal* wires are drawn. Lastly, the output wires are drawn. The input and output labels are placed next to the input and output terminals. (Refer to the pseudo-code for Algorithm-4 in Appendix A).

The creation of the textual descriptions of questions is much simpler than drawing the schematics. This simply involves displaying fixed/pre-set text strings for different question types.

The *correct answer* to a question is found by simulating its HDL code; an open-source simulator Icarus [Ica (2020)] is used for this purpose.

QAgen Applications

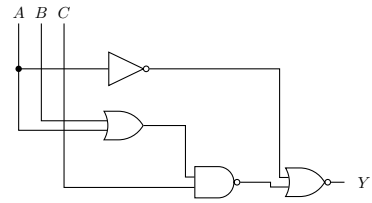
As mentioned earlier, QAgen has applications in many courses in a typical ECE curricula, as illustrated by the following examples.

Digital Design

Given below are three sample questions for our Digital Design & Computer Organization course and the Digital System Design course. Q 1.1 requires the display of question text and the drawing of a circuit schematic with a random mix of logic gates. The number of gates and their types are specified as input parameters to QAgen. Q 1.2 needs only the

question text and a randomly generated Boolean equation with a user-specified number of variables (three in this case, i.e., A, B, and C). Lastly, Q 1.3 requires the question text and a random piece of Verilog HDL code; the code is constrained by the user-defined parameters, i.e., number of circuit inputs and the count and the types of logic gates.

Q 1.1. Draw the truth table for the following logic circuit:



Q 1.2. Convert the following *dataflow* style function into *gate-level* HDL format:

$$Z = (A + B.C).(A \oplus C)$$

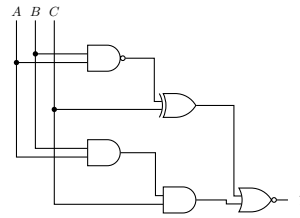
Q 1.3. Write an *exhaustive* testbench for the logic circuit described by the following Verilog HDL code:

```
module fcn1 (Y, A, B, C);
    input A, B, C;
    output Y;
    wire w1, w2, w3;
    nand G1 (w1, C, B);
    not  G2 (w2, A);
    not  G3 (w3, B);
    or   G4 (Z, w3, w2);
endmodule
```

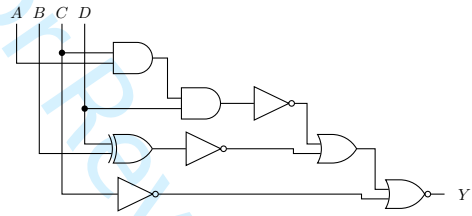
Computer Architecture

Given below are two sample questions related to the timing and pipelining of circuits in our Computer Architecture course. Q 2.1 involves the display of question text and the generation of the schematic of a random logic circuit, for which the contamination and the propagation delays need to be calculated. Here, the number of circuit inputs and numbers and types of gates are specified by the user. The numerical values (i.e., 'delays') vary randomly within a user-specified integer range (let's say between 2 ns and 10 ns). Q 2.2 is about the pipelining of a logic circuit. Although this question requires a logic circuit larger than the one in Q 2.1, the schematic drawing process remains the same.

Q 2.1. Refer to the following logic circuit in which all gates have the same delay of 10 ns. Calculate the circuit's contamination delay and the propagation delay.



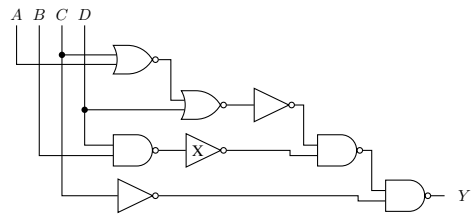
Q 2.2. Refer to the following logic circuit in which all gates have the same delay of 5 ns. Create a two-stage pipeline and determine the resulting circuit's maximum operating frequency.



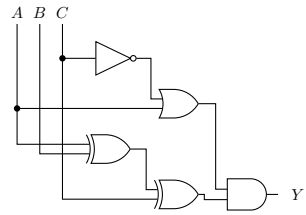
Hardware Testing & Fault Tolerance

Given below are two sample questions for our Hardware Testing and Fault Tolerance course. Both Q 3.1 and Q 3.2 include the text and randomly (but user-constrained) circuit schematics. In Q 3.1, a 'faulty' gate is randomly selected and marked with an 'X'. The number of inputs is kept small (up to four) to limit the time required to find the test vectors using a truth table method. The number of inputs in Q 3.2 is constrained to four to reduce the question complexity.

Q 3.1. Use the truth table method to find a set of test vectors for detecting a stuck-at-0 fault at the input of gate-X:



Q 3.2. Find the self-dual circuit for the following three-input circuit:



QAgen Evaluation

As a proof-of-concept, a large number of questions were generated for different traditional/classroom-based courses during the Spring 2019 semester. A few examples of the questions uploaded on BlackBoard (a well-known LMS) are shown in Appendix B. The examples include questions about logic equations, logic diagrams, Karnaugh maps, and truth tables for our Digital Design & Computer Organization course. At this stage, the questions were provided to the students only for practice, i.e., their scores were not included in their actual course grades. The students were asked to fill in surveys (similar to the ones in [Patel et al. (2002), Lawand and Pang (2014)]) about the questions for three courses. The results of the surveys are shown in Table I. The majority of students used the questions for practice before the actual (graded) test/examination. Most of the students felt that the questions helped them prepare for the tests/examinations. The availability of the answers to the questions was overwhelmingly favored by the students. A large number of students felt more confident in problem-solving after going through the questions. Finally, the students believed that their grades improved when they practiced with the question-answer sets. Our initial assessment of the QAgen tool motivates us to extend the tool's usage to MOOC(s).

Table 1. Student survey statements for three different courses (1 = Strongly disagree, 2 = Slightly disagree, 3 = Neutral, 4 = Slightly agree, 5 = Strongly agree)

Digital Design & Computer Organization

Question	1	2	3	4	5
I solve the practice questions before an actual test/exam.	3%	0%	9%	33%	55%
The practice questions help me prepare for an actual test/exam.	0%	0%	3%	15%	82%
Having answers to the practice questions is useful for my checking my understanding.	0%	0%	3%	3%	94%
It would be a good to work with practice questions for different topics in this course.	0%	0%	6%	24%	70%
I have more confidence in my skills after solving practice questions.	0%	0%	3%	24%	73%
My score in an actual assessment is higher when I solve practice questions.	0%	6%	9%	33%	52%

Computer Architecture

Question	1	2	3	4	5
I solve the practice questions before an actual test/exam.	6%	0%	18%	29%	47%
The practice questions help me prepare for an actual test/exam.	0%	12%	6%	29%	53%
Having answers to the practice questions is useful for my checking my understanding.	6%	0%	6%	13%	75%
It would be a good to work with practice questions for different topics in this course.	6%	0%	18%	6%	70%
I have more confidence in my skills after solving practice questions.	6%	6%	21%	11%	56%
My score in an actual test/exam is higher when I solve practice questions.	6%	6%	18%	23%	47%

Hardware Testing & Fault Tolerance

Question	1	2	3	4	5
I solve the practice questions before an actual test/exam.	0%	0%	44%	37%	19%
The practice questions help me prepare for an actual test/exam.	0%	0%	12%	19%	69%
Having answers to the practice questions is useful for my checking my understanding.	0%	0%	6%	13%	81%
It would be a good to work with practice questions for different topics in this course.	0%	0%	19%	25%	56%
I have more confidence in my skills after solving practice questions.	0%	0%	25%	13%	62%
My score in an actual test/exam is higher when I solve practice questions.	6%	0%	13%	31%	50%

Conclusions

The proposed QAgen tool is capable of automatically generating a large number of test questions and the corresponding answers in a format suitable for LMS and/or non-LMS-based traditional course or MOOC assessment. The tool is based on an open-source software thus eliminating the need for any commercial packages. The algorithms for automatically creating the circuit diagrams have been included in this work. The QAgen-generated questions have been used in recent offerings of a few courses; the student surveys from these courses show the students benefitted from the varied sets of practice problems, specifically, the students prepared better for their actual/graded tests/examinations.

The principles used herein can also be helpful for other traditional courses or MOOCs, provided they require the creation of diagrams with connected geometric objects, code-based problems, equations and their solutions, etc. Our future work will be focused on extending the capabilities of QAgen for a complete MOOC.

References

[Ahmed et al. (2018)] Ahmed B, Kagita M, Wijenayake C and Ravishankar J (2018) Implementation Guidelines for an Automated Grading Tool to Assess Short Answer Questions on Digital Circuit Design Course. In: *2018 IEEE Int. Conf. Teaching, Assessment, and Learning for Eng.* Wollongong, Australia, pp. 1142–1145.

[Al-Ashmoery and Messoussi (2015)] Al-Ashmoery Y and Messoussi R (2015) Learning analytics system for assessing students' performance quality and text mining in online communication. In: *2015 Intell. Syst. Comput. Vis.* Fez, Morocco, pp. 1–8.

[Aleman and Couleur (1990)] Aleman E and Couleur J (1990) Automated schematic capture with the USC51 embedded microcontroller. In: *Euro ASIC '90*. Paris, France, pp. 461–465.

[Arya et al. (1985)] Arya A, Swaminathan V, Misra A and Kumar A (1985) Automatic Generation of Digital System Schematic Diagrams. In: *22nd Des. Autom. Conf.* Las Vegas, NV, USA, pp. 388–395.

[Banks (2016)] C Banks and Edward M (2016) The Acceptability of MOOC Certificates in the Workplace. In: *Int. Conf. e-Learning.* Madeira, Portugal, pp. 419–423.

[Bassi et al. (2014)] Bassi R, Daradoumis T, Xhafa F, Caballe S and Sula A (2014) Software Agents in Large Scale Open E-learning: A Critical Component for the Future of Massive Online Courses (MOOCs). In: *2014 Int. Conf. Intell. Netw. Collab. Syst.* Salerno, Italy, pp. 184–188.

[Bednarik and Kovacs (2012)] Bednarik L and Kovacs L (2012) Implementation and assessment of the automatic question generation module. In: *2012 IEEE 3rd Int. Conf. Cogn. Infocommu.* Kosice, Slovakia, pp. 687–690.

[Beg et al. (2017)] Beg A, Alhemeiri M and Beg A (2017) Enhancing automated assessment for engineering MOOCs. In: *IASTED Int. Conf. Model. Simul. Identif. (MSI 2017)* Calgary, AB, Canada, pp. 1–5.

[Beg and Beg (2016)] Beg A and Beg A (2016) Auto-Generating Publication-Quality Circuit Schematics Using Open Technologies. In: *21st West. Can. Conf. Comput. Educ. (WCCCE '16)*. Kamloops, BC, Canada, pp. 51–56.

[Beg and Beg (2018)] Beg A and Beg A (2018) Using Open Technologies for Automatically Creating Question-and-Answer Sets for Engineering MOOCs. *Comput. Appl. Eng. Educ.* 26(3): 617–625.

[Brennan (1975)] Brennan RJ (1975) An Algorithm for Automatic Line Routing on Schematic Drawings. In: *12th Des. Autom. Conf.* Piscataway, NJ, USA, pp. 324–330.

[Cad (2020)] Cadence (2020). URL http://www.cadence.com/.

[Chen (2011)] Chen P (2011) Construction of test question database for electrical and electronic technology. In: *2011 Int. Conf. Electron. & Mechanical Eng. Inf. Technol.* Harbin, China, pp. 1347–1350.

[Chi (2020)] ChipVault (2020). URL http://chipvault.sourceforge.net/.

[Chiueh (1991)] Chiueh TC (1991) HERESY: A hybrid approach to automatic schematic generation (for VLSI). In: *Eur. Conf. Des. Autom.* Amsterdam, Netherlands, pp. 419–423.

[Con (2020)] Concept Engineering (2020). URL https://www.concept.de.

[Cou (2020)] Coursera (2020). URL https://www.coursera.org/.

[Cross et al. (2019)] Cross JS, Keerativoranan N, Carlon MKJ, Tan YH, Rakhimberdina Z and Mori H (2019) Improving MOOC quality using learning analytics and tools. In: *2018 13th Int. Conf. Comput. Sci. Educ.* Milwaukee, WI, USA, pp. 174-0179.

[De Rosa and Pistolese (2019)] De Rosa A and Pistolese M (2019) Evolution, Education and Massive Open Online Courses: A Multiverse Proposal. In: *2019 IEEE Learning With MOOCS (LWMOOCS)*. Milwaukee, WI, USA, pp. 191–195.

[Hao and Huiyan (2018)] Hao W and Huiyan L (2018) Research on Blended Teaching Reform and Innovation Strategy Based on MOOC Education. In: *2018 13th Int. Conf. Comput. Sci. Educ. (ICCSE 2018)* , Colombo, Sri Lanka, pp. 1–4.

[Herman and Handzik (2010)] Herman GL and Handzik J (2010) A preliminary pedagogical comparison study using the digital logic concept inventory. In: *2010 IEEE Front. Educ. Conf.* Washington, DC, USA, pp. F1G–1–F1G–6.

[Herman and Loui (2011)] Herman GL and Loui MC (2011) Administering a Digital Logic Concept Inventory at Multiple Institutions. In: *2011 ASEE Annu. Conf. Expo.* Vancouver, BC, Canada, pp. 1–12.

[Ica (2020)] Icarus (2020). URL https://sourceforge.net/projects/iverilog/.

[Ip et al. (2018)] Ip HSH, Li C, Leoni S, Chen Y, Ma K, Wong CH and Li Q (2018) Design and Evaluate Immersive Learning Experience for Massive Open Online Courses (MOOCs). *IEEE Trans. Learn. Technol.*: 1–10.

[Kim et al. (2000)] Kim NH, Kim KS, Choi KM and Kong JT (2000) RightTopologizer: an efficient schematic generator for multi-level optimization. In: *13th Annu. IEEE Int. ASIC/SOC Conf.* Arlington, VA, USA, pp. 387–391.

[KTe (2020)] KTechLab (2020). URL https://sourceforge.net/projects/ktechlab/.

[Lageweg (1998)] Lageweg C (1998) Designing an Automatic Schematic Generator for a Netlist Description. Technical report, Delft University of Technology, Delft, Netherlands.

[Lawand and Pang (2014)] Law E and Pang S (2014) Use of online assessments to monitor learning outcomes in higher level engineering courses. In: *4th Interdiscip. Eng. Des. Educ. Conf.* Santa Clara, CA, USA, pp. 18–23.

[Liu et al. (2010)] Liu D, Zheng L, Wang X and Zhuan S (2010) Research on Intelligent Test Paper Generation Based on Improved Genetic Algorithm. In: *2010 Second WRI Glob. Congr. Intell. Syst.*, volume 2. Guilin, China, pp. 363–365.

[Lu et al. (2018)] Lu M, Zhao H, Guo Y, Wang K and Huang Z (2018) A Review of the Recent Studies on MOOCs. In: *2018 13th Int. Conf. Comput. Sci. Educ. (ICCSE 2018)* , Colombo, Sri Lanka, pp. 1–5.

[Miranda et al. (2013)] Miranda S, Mangione GR, Orciuoli F, Gaeta M and Loia V (2013) Automatic generation of assessment objects and Remedial Works for MOOCs. In: *2013 12th Int. Conf. Inf. Technol. Based High. Educ. Train.* Antalya, Turkey, pp. 1–8.

[Nascimento et al. (2013)] Nascimento MGF, Brandao LO and Brandao AAF (2013) A model to support a learning object repository for web-based courses. In: *2013 IEEE Front. Educ. Conf.* Oklahoma City, OK, USA, pp. 548–552.

[Naveen and Raghunathan (1993)] Naveen B and Raghunathan K (1993) An automatic netlist-to-schematic generator. *IEEE Des. Test Comput.* 10(1): 36–41.

[Nguyen and Fong (2013)] Nguyen ML and Fong ACM (2013) Large-Scale Multiobjective Static Test Generation for Web-Based Testing with Integer Programming. *IEEE Trans. Learn. Technol.* 6(1): 46–59.

[Oliveira et al. (2018)] Oliveira T, Stringhini D and Corrêa D (2018) Online Peer Assessment and Scoring Rubric to Produce Better Digital Systems Designs in an Undergraduate Computer Engineering Curriculum. In: *Amer. Conf Learning Tech. (LACLO)*, São Paulo, Brazil, pp. 393–399.

[Patel et al. (2002)] Patel N, Covic G and Hussmann S (2002) Experience using a novel web-based tutorial and assessment tool for advanced electronics teaching. In: *Int. Conf. on Comput. in Educ.*, Auckland, New Zealand, pp. 643–647.

[Paul and Pawar (2013)] Paul D and Pawar J (2013) Dynamic Question Paper Template Generation Using Bi-proportional Scaling Method. In: *2013 IEEE Fifth Int. Conf. Technol. Educ.* Kharagpur, India, pp. 80–83.

[Posner (2020)] Posner Z (2020) What is Adaptive Learning Anyway? URL https://www.mheducation.com/ideas/what-is-adaptive-learning.html.

[QFl (2020)] QFlow (2020). URL http://opencircuitdesign.com/qflow/.

[Rahman and Ogunfunmi (2010)] Rahman M and Ogunfunmi T (2010) A set of questions for a concept inventory for a DC Circuits course. In: *2010 IEEE Int. Symp. Circuits Syst.* Paris, France, pp. 2808–2811.

[Sil (2020)] Silvaco (2020). URL https://www.silvaco.com.

[Smaill (2005)] Smaill C (2005) The Implementation and Evaluation of OASIS: A Web-Based Learning and Assessment Tool for Large Classes. *IEEE Trans. Educ.* 48(4): 658–663.

[Sma (2020)] SmartDraw (2020). URL https://www.smartdraw.com/circuit-diagram/.

[Soltani et al. (2019)] Soltani M, Zarzour H, Babahenini MC, Hammad M, Al-Smadi M and Jararweh Y (2019) An Emotional Feedback Based on Facial Action Coding System for MOOCs with Computer-Based Assessment. In: *2019 6th Int. Conf. Social Netw. Anal., Management and Security* Granada, Spain, pp. 286–290.

[Uda (2020)] Udacity (2020). URL https://www.cadence.com.

[Ver (2020)] Verilator (2020). URL https://www.veripool.org/.

[Yos (2020)] Yosys (2020). URL http://www.clifford.at/yosys/.

[Yuan-Bin and Jie (2012)] Yuan-Bin C and Jie D (2012) Design on Algorithm of Automatic Test Papers Generation for Examination System of Electric Energy Measurement. In: *2012 Int. Conf. Comput. Sci. Serv. Syst.* Nanjing, China, pp. 1397–1400.

[Zuk (2020)] Zuken (2020). URL https://www.zuken.com/.

APPENDIX A – QAGEN ALGORITHMS

```
Procedure: CreateBooleanEquation
Input:     integer nGateCount
Output:    string BoolEq

initialize g, boolEq;
while (g < nGateCount) {
  Pick a random logic operation
  Pick a random number of input variables
  Apply logic operation to randomly picked variables to create a logic gate
  Save the logic gate as array-element gate[g]
  Increment g
}
Append a random number of left brackets to boolEq
Append gate[0] to boolEq
Append a random logic symbol to boolEq
for (g=1; g<=nGateCount; g++) {
  Append gate[g] to boolEq
  Append a random number of right brackets to boolEq
  Append a random logic symbol to boolEq
  Append a random number of left brackets to boolEq
}
Append any remaining right brackets to boolEq
return boolEq
```

Pseudo-code 1. Creating Boolean Equations.

```
Procedure: ConvertBoolean2RPN
Input:     string BoolEq
Output:    string rpn

Store first token in BoolEq as output, outputName
for (i=2; i<length(BoolEq); i++) {
  Read token at i
  if token is a left parenthesis {
    Push it onto the stack
  }
  if token is a right parenthesis {
    Until the token at the top of the stack is a left parenthesis, pop stack contents to output string rpn
  }
  if token is an operator {
    while (operator precedence <= stack peek precedence) {
      Pop stack contents to rpn string
    }
  }
  if token is an operand {
    Append token to rpn string
  }
}
Pop remaining stack contents to rpn string
```

Pseudo-code 2. Transforming the Boolean functions into RPN notation.

```
Procedure: ConvertRPN2Verilog
Input:     string rpn
Output:    file code.v

% Parse rpn and store it in vArray[operation, output, operand1, operand2]
nflag     = 0; % consider nand/nor/xnor if high
gateCount = 0; % number of gates
prevOut   = '' % keep track of the previous operation

for (i=1; i<length(rpn); i++) {
  Read token at j
  if (token is an operand) {
    Push it onto the stack
    Append operand to inputList
  }
  else if (token is an operator) {
    Find operation name from operator
    if (operator is INV) {
      Store previous gate in prevOut, if any
      If(stack peek equals prevOut and nflag is high) {
        change vArray[gateCount,1] to the appropriate gate (NAND, NOR, XNOR)
      }
      else {
        Increment gateCount
        Store operation name in vArray[gateCount, 1]
        Create wire name and store it in wireList and vArray[gateCount, 2]
        Pop stack and store value in vArray[gateCount, 3]
        Push the created wire onto the stack
      }
    }
    else {
      Increment gateCount
      Store operation name in vArray[gateCount,1]
      Create wire name and store it in wireList and vArray[gateCount, 2]
      Pop stack for operand 1 and store it in vArray[gateCount, 3]
      Pop stack again for operand 2 and store it in vArray[gateCount, 4]
      Push the created wire onto the stack
    }
  }
}
% Generate verilog code and write to file
Print 'module' definition to a string
Append output name ('Y')
Append input list from vArray-inputList
Append wire list from vArray-wireList
for (r=1; r<=vArray rows; r++) {
  Print vArray contents in Verilog format
  newline
}
Append 'endmodule' to the string
Write string to a file 'code.v'
```

Pseudo-code 3. Creating the Verilog models from the RPN format.

```
Procedure: CreateLogicDiagram
Input:     file Code.v
Output:    file LogicDiagram.tex

% nInputs is number of input ports; nOutputs is number of output ports; and nWires is number of wires (internal nets)
Open Code.v file for reading
Identify code line(s) starting with 'input', and parse to store input labels in input[nInputs-1:0]
Identify code line(s) starting with 'output', and parse to store output labels in output[nOutputs-1:0]
Identify code line(s) starting with 'wire', and parse to store wire labels in wire[nWires-1:0]
nGate = 0;
while not('endmodule') {
  Parse the gate descriptions to extract input and output labels.
  Store gate input and output labels in gateInput[] and gateOutput[] arrays
  Identify which gate input is driven by which circuit-input or other gate-output
  nGate++
}
% --- Draw gates ---
% (x, y) is gate coordiate; x is horizontal position and y is vertical
Mark locations of all gates as 'not-assigned'
Set x = 0, y = 0
for (i=0; i<nGates; i++) {
  if gate[i] is driven only by circuit inputs, draw gate at (x, y) {
    mark gate[i] location as 'assigned'
    y--  % next gate below this one
  }
}
x = 1; y = 1;
for (i=0; i<nGates; i++) {
  if gate[i] location is 'unassigned' and it does not drive an output, draw gate at (x, y) {
    mark gate[i] location as 'assigned'
    x++; y--  % next gate is on the right and below
  }
}
for (i=0; i<nGates; i++) {
  if gate[i] location is 'unassigned' and gate drives an output {
    set gate[i] y as middle of its driving gate y
    draw gate at (x, y)
    mark gate[i] location as 'assigned'
  }
}
% --- Draw wires and labels ---
x = -1; y = 0
for (i=0; i<nInputs; i++) {
  for (j=0; j<nGateInputs; i++) {
    draw wire from (x, y) to all gate-inputs driven by input[i] with small horizontal lines at the gate input j and
       input[i]
    display input labels at (-0.5, y)
    y--
  }
}
for (i=0; i<nGates; i++) {
  for (j=0; j<nGateInputs; i++) {
    draw wire from gateInput[j] to its 'driver' gate (using wire[] with small horizontal lines at the gate input j
       and the driver output[i]
  }
}
for (i=0; i<nOutputs; i++) {
  draw small horizontal output wire
  display output label
}
Save logic diagram .tex file
```

Pseudo-code 4. Drawing logic diagrams from the Verilog code.

APPENDIX B – A SAMPLE BLACKBOARD-BASED QUIZ

QUESTION 1 2 points Save Answer

Which Boolean function does the following product-of-sums represent?

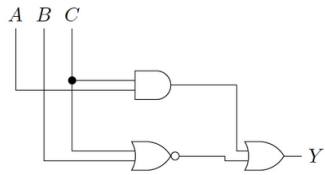
$$\prod M(1,2,3,6)$$

(a) $(A+B+C).(A+B+\bar{C}).(A+\bar{B}+C).(\bar{A}+B+C).(\bar{A}+B+\bar{C}).(\bar{A}+\bar{B}+\bar{C})$
(b) $(\bar{A}\bar{B}C)+(\bar{A}B\bar{C})+(\bar{A}BC)+(AB\bar{C})$
(c) $(A+B+\bar{C}).(A+\bar{B}+C).(A+\bar{B}+\bar{C}).(\bar{A}+\bar{B}+C)$
(d) $(A+B+C).(A+B+\bar{C}).(\bar{A}+B+C).(\bar{A}+B+\bar{C})$

○ (a) ○ (b) ○ (c) ○ (d)

QUESTION 2 2 points Save Answer

Create a truth table for the following logic diagram and represent it as a sum-of-products function:



QUESTION 3 2 points Save Answer

Use Karnaugh Map for minimization:

<-- CD -->

AB	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	1	1	0
10	1	0	0	1

(a) $\bar{A}\bar{C}+A\bar{B}C$
(b) $\bar{B}\bar{C}+\bar{B}CD+BC\bar{D}$
(c) $\bar{A}\bar{B}+\bar{B}\bar{D}+ABD$
(d) $\bar{A}BC+\bar{B}\bar{D}+ABD$

○ (a) ○ (b) ○ (c) ○ (d)

QUESTION 4 2 points Save Answer

Fill in the truth table for the following Boolean function:

$$\bar{A}BC+\bar{B}D+ABD$$

No.	Input A	B	C	D	Output Y
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	