Design of Control Systems as a Learning Activity

Ilya Levin
School of Education, Tel Aviv University

Abstract

Logical control design provides an excellent project-based learning activity in engineering education. It opens a way to teach the fundamentals of synthesis, analysis and decision-making using one and the same environment.

In the Computer Engineering curricula, there is a chain of courses dealing with the logical control design. These courses are "Switching Theory and Logic Design", "Advanced Switching Theory", "Digital Systems", "Computer Architecture", "VLSI Design", "Computer Aided Design". In the Electrical Engineering curricula, such courses as "Automation" and "Process Control" also include a logic control design oriented subject matter.

Actually, the logic control design weaves together the Computer and the Engineering Curriculums, forming one of the key issues in both of them.

The modern requirements to the logic control design course are:

- formal methods of design on each stage thereof, from initial specification of a controller up to its final implementation and verification,
- a computer-based environment for supporting the students' practical work.

This paper proposes: a) a universal formal notation, including concepts of ASM (Algorithmic State Machine) and FSM (Finite State Machine), as a methodological fundamental of the logical control design, b) an Interactive Learning Environment developed on the base of the formal notation.

The proposed approach enables transforming, uniting, minimizing, and decomposing both ASMs and FSMs. On one hand, transformation on a set of the formally defined ASMs and FSMs provides a rich variety of learning activities in the problem solving and the design. On the other hand, the use of the proposed Interactive Learning Environment enables students to design and explore complex control systems.

The proposed approach drastically increases a plurality of possible tasks and projects in a class and, consequently, opens an opportunity to enrich both the teaching and the learning processes.

1. Introduction

It is a widely known phenomenon, that students experience difficulties while establishing a comprehensive interconnection between theoretical knowledge of a complex subject and a practical knowledge of the same subject. In particular, there is a gap between a theoretical university courses of logical design on one hand, and engineering (practical) courses of circuits' design. This gap is natural and fundamental, since the theoretical courses are focussed on the methodological and

optimization problems, while the practical courses are oriented to development of an ability to find technical solutions.

It goes without saying that bridging such a gap would be a desired purpose for a teacher and a fruitful achievement for a student during the educational process. An educational approach to the above problem is proposed in this paper with reference to the above-captioned case. It has become possible owing to modern technological computer means.

Any controlled system can be described in a form of interaction of the two following parts: an operation part and a control part. The control part of the controlled system can be described by using a concept of Controlware, first proposed in work [2]. Controlware was defined as a special toolkit specially created for designing the control part of any controlled system.

The Controlware differs from traditional software in the following significant ways:

1. While software is universal means, Controlware is means oriented to programming of control units.
2. All traditional programs manipulate data flows while Controlware employs control flows.
3. Controlware exhibits a high degree of transparency; for instance, a student can immediately see the influence of the control part on behavior of the operational part.
4. While traditional software is usually oriented on a standard sequential computer architecture, Controlware is based on a parallel architecture, which is characteristic for control units.

In the frame of the novel educational approach, we propose to use the Controlware as a specialized toolkit for designing control systems. Blocks of this toolkit, when being presented and given to students for designing a control system in a class, enable construction of the system, while synchronously displaying it both as a state diagram of FSM and as a flow-chart of ASM. The simultaneous constructing and displaying of the control system in the two above-mentioned representations demonstrate unity of the theoretical and practical approaches to the logical design. The teacher is therefore able to give and the student is invited to acquire both the theoretical optimization knowledge and the practical design skills concerning the subject.

The main idea will be discussed in the following parts of the paper.

Section 1 explains two logical paradigms forming a cognitive basis of the proposed approach, namely – a programming and a design paradigm.

Section 2 presents two educational directions in teaching the logical control design, which are: teaching the design *per se*, and teaching the design automation.

Section 3 describes an educational software environment SMILE implementing the proposed approach.

1. Design of Control versus Programming of Control

In works [3, 4] two main paradigms were suggested as conveyors of very different cognitive approaches to control: the programming and the design paradigms.

By the programming paradigm, a person (e.g., student, user, technician, and designer) assumes the existence of a control performer (e.g., a microprocessor), in charge of running the control specifications. This kind of a logical control unit can be defined as a programmable controller. By this paradigm, the creating of control means will be creating of an appropriate program.

The key formal construct for the algorithmic paradigm is the Algorithmic State Machine (ASM) [1]. An ASM is a directed connected graph, which includes an initial vertex, a final vertex and a finite set of operator and conditional vertices. The final, operator and conditional vertices have only one input, and the initial vertex has no input. Initial and operator vertices have only one output, and conditional vertices have two outputs marked as "1" and "0". The final vertex has no outputs.

Defining characteristics for an ASM are also the following:

(a) Output vertices are connected by arcs to input vertices. Every output is connected only to one input, every input is connected from at least one output.

(b) Every vertex is part of at least one path leading from the initial vertex to the final vertex.

(c) A logical condition from set X of input variables of the control unit appears in each conditional vertex. A given logical condition may appear in different conditional vertices.

(d) A microinstruction or operator Y_i appears in every operator vertex.

By following the design paradigm, the person (e.g., student, user, technician, designer) focuses on the logical scheme inside the controller, implemented by means of logical elements of varied nature (e.g., logical gates, contacts, programmable logical devices). Contrary to the previous programming paradigm, where the controller was defined as a *programmable* controller, it is called a *designable* controller within the design paradigm.

A key idea within the design paradigm is that the control unit is presented as a finite state machine (FSM). The control unit can be characterized by its state and may perform different functions (i.e., changing to other states) depending upon its current state. A formal construct, which we consider the most appropriate for the formal-model definition, is the state diagram. The state diagram is a representation of the system's possible states and possible transitions between them. Nodes in the diagram indicate states, and arrows indicate transitions between the states coused by specific input values. Also, the FSM can be represented in the form of a state table (i.e. a tabular form of the state diagram). Columns of the table indicate consequently: a current state, an input value, a corresponding output, and a corresponding next state.

The concept of FSM is very close to the concept of an ASM. We will say that FSM implements a corresponding ASM. Any ASM can be transformed to the FSM form. To perform this transformation the following steps have to be taken. First of all, the ASM has to be marked, by marks reflecting states of FSM. The second step is the searching for paths between the marks within the ASM. Every such path has to include one operator vertex. Each path can be interpreted as a transition within the FSM. We will represent the FSM as a list of transitions, which are paths of the initial ASM.

An example, taken from [1], of the marked ASM with logical conditions $X = \{x_1, \ldots, x_4\}$, micro-operations $Y = \{y_1, \ldots, y_6\}$, microinstructions $F = \{Y_0, Y_1, \ldots, Y_6\}$ and marks (states of FSM) $\{a_1, \ldots, a_5\}$ is shown in Figure 1. The table of the corresponding FSM is presented in Table 1.

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$
a_1	a_2	x_1x_2	y_2,y_3
	a_4	$x_1x_2'x_3$	y_4
	a_1	$x_1x_2'x_3'$	-
	a_3	x_1'	y_2
a_2	a_4	1	y_1,y_4
a_3	a_1	x_4x_1	y_1,y_3
	a_3	x_4x_1'	-
	a_4	x_4'	y_1,y_4
a_4	a_5	x_2	y_5,y_6
	a_1	x_2'	-
a_5	a_1	1	y_1,y_3

Table 1. Table of FSM corresponding to ASM shown in Fig. 1.

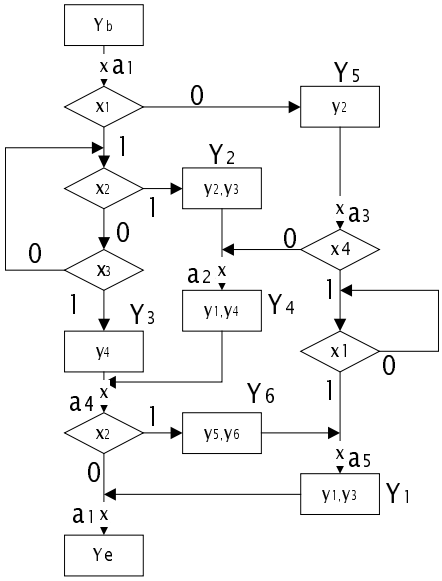

Figure 1. Example of the marked ASM

In Table 1:

a_m - current state,

a_s - next state,

$X(a_m, a_s)$ - transition function, i.e. a Boolean function which is equal to 1 iff FSM makes the transition from state a_m to state a_s (x_l' is negation of x_l).

$Y(a_m, a_s)$ - list of output signals which are equal 1 on the transition of the FSM from a_m to a_s.

In the light of the above, we may define control as a process by which the FSM's transition from one state to another occurs. That means that at any time the FSM is in any of a number of possible states; that there are conditions under which the current state changes; and that the control unit implies the choice.

2. Design Education versus Design Automation Education

Two directions can be distinguished in a logical design education: the *design* education and the *design automation* education. The logical design education can be defined as an educational area dealing with the teaching and the learning of logical system's design process. The design automation education relates to the teaching and the learning of methods for optimization of the logical systems design according to various criterions. Having different aims, these two directions require different teaching strategies, different learning activities, and different students' skills.

The main learning outcome in the design education is a real project of a logical device created by a student. The main learning activity in this direction is designing. The designing is performed by using Computer Aided Design (CAD) systems, hardware design languages (VHDL, Verilog, Altera, Xillinx etc.) or by using special design education environments, for example proposed in the present paper.

The main learning activity in the design automation education is solving problems of optimization. Ability to solve such problems is one of the basic skills of the CADs developer.

Learning activities in the design automation education can be based on using the proposed formal models of ASM and FSM. Examples of possible classroom activities are presented in the following list.

1. Combining several ASMs (FSMs) into one ASM (FSM) with minimum number of vertexes.
2. Decomposing of one ASM (FSM) into a network of constituent ASMs (FSMs).
3. Investigation of influence of certain changes made to ASM, on the FSM-representation of the control unit.
4. Implementation of the same control unit using the ASM and the FSM descriptions and analysis of the received results.

3. Interactive Learning Environment

The interactive learning environment was developed in the framework of SMILE-project (State Machine Interactive Learning Environment) in the School of Education of the Tel-Aviv University. SMILE-environment was developed for implementation of the above-mentioned ideas, i.e.:

- SMILE constitutes the Controlware toolkit;
- SMILE supports both the programming and design paradigms of teaching control concepts;
- SMILE serves two pedagogical needs: design teaching and design automation teaching;
- SMILE is capable of controlling real equipment.

SMILE comprises two Editors: the Algorithmic State Machine Editor (ASM Editor) and the Finite State Machine Editor (ASM Editor), having each its display window. The SMILE-environment screen shot is shown on Figure 2.

The environment enables the students to design a control unit both in the form of FSM by the FSM Editor, and in the form of ASM by the ASM editor. When the control unit is designed in one of these editors it can be visualized simultaneously in the windows of both editors. Any changes made in one of the editors cause relevant changes to appear not only in its window but also in the other editor window.

Both of the State Machine Editors ensure a number of very powerful features for describing virtually any type of a State Machine. For example, a student can represent a State Machine hierarchically, i.e. a State Machine having a large number of states can be described in multiple levels. For debugging a State Machine, two very important features are available in the Editors. The first is an animated simulation where any step in the control process is depicted by a change in color of the ASM (and/or FSM) symbolic states, which can be seen in the appropriate display windows. The second feature of the Editors is a possibility to study the control process watching the behavior of real equipment.

It should be noted that, when the Control Unit's description is debugged, the resulting State Machine file could be obtained in the accepted hardware description language forms (VHDL, Verilog). Such a file can be further used for a real VLSI design.

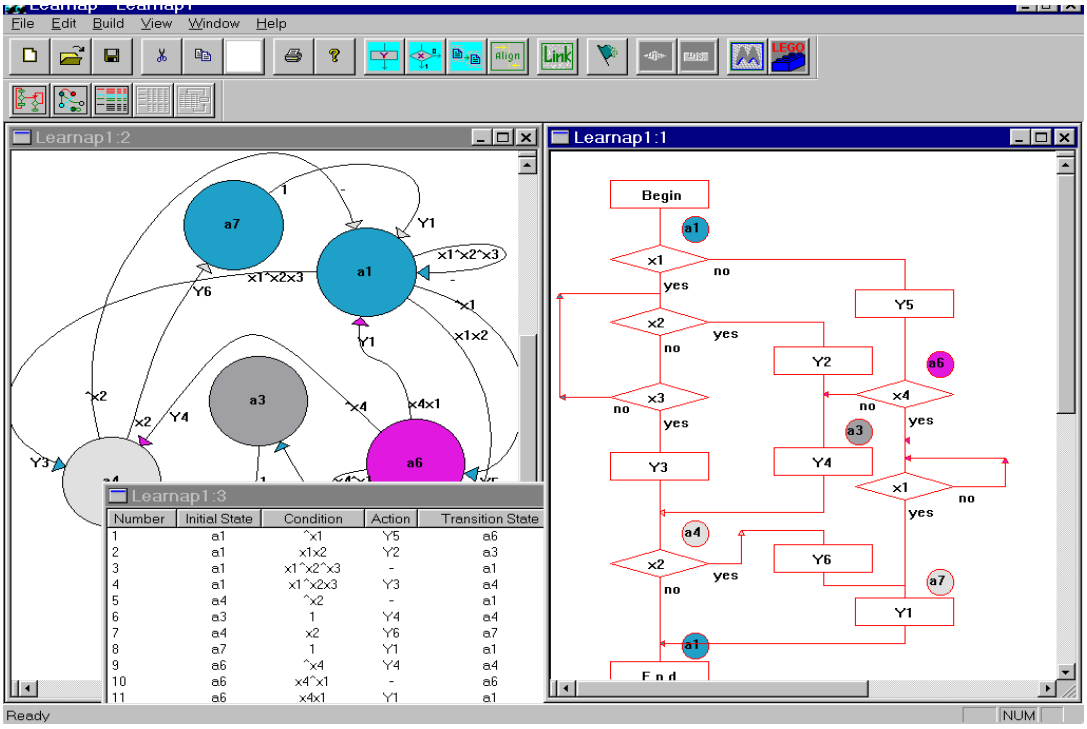

Figure 2. The SMILE-environment screen shot.

4. Conclusion

We have proposed a new educational approach for teaching logical design of control systems, which allows bridging a comprehension gap between theoretical and engineering courses related to the subject. The approach is implemented by the State Machine Interactive Learning Environment (SMILE) developed in the School of Education of the Tel-Aviv University. It is addressed to students of Electrical Engineering and Computer Engineering undergraduate education.

It is believed that after having been trained by SMILE the students will acquire both theoretical and practical basic skills required in the subject.

Bibliography

1. S. Baranov. Logic Synthesis for Control Automata. Kluwer Academic Publisher, Dordrecht/Boston/London. 1994.
2. I. Levin, V. Levit. "Controlware for Learning with Mobile Robots", Computer Science Education, Vol. 8, No2, pp. 1-11,1998.
3. I. Levin, D. Mioduser. "A Multiple-Constructs Framework for Teaching Control Concepts", IEEE Transactions of Education, Vol.39, Issue No.4, pp.488-496, 1996.
4. D. Mioduser, I. Levin. "Cognitive-conceptual Model for Integration Robotics and Control into the Curriculum", Computer Science Education, Vol. 7, No. 2, pp. 199-210, 1996.

ILYA LEVIN
Dr. Ilya Levin received the Ph.D. degree in Computer Science from Latvian Academy of Science. During 1985-1990 he was the Head of the Computer Science Department in the Institute of New Technologies of Leningrad. During 1993-1996 years he was the Chairman of the Computer Systems Department of Holon Center for Technological Education, Israel. Being presently a Senior Lecturer of the School of Education of Tel Aviv University, he is a supervisor of the Technology Education program. He is an author of more then 50 papers both in the Design Automation and in the Technology Education fields.