Journal of Engineering Sciences and Information Technology
Volume (4), Issue (1) : 30 Mar 2020
P: 107 - 121
ISSN: 2522-3321
مجلة العلوم الهندسية وتكنولوجيا المعلومات
المجلد (4)، العدد (1) : 30 مارس 2020م
ص: 107 - 121

A Simulator for Micro Programming and Hardware Simulation Integrated in a Computer Hardware Project

Yaser Chaaban
Faculty of Science and Arts || AlUla Branch || Taibah University || KSA

Abstract: Nowadays, micro-programming of machines becomes more common. It is technique used in several fields such as Computer Engineering. Here it is worth mentioning that micro-programming is employed throughout the design process. As well, designing the control unit of digital computers needs micro-programming, which is more complex than assembly languages. In this field, micro-programs can be written as sequences of micro-instructions. In this context, distinguished teaching of micro-programming of machines requires a suitable and carefully chosen Computer Simulation Tool (CST). This research designs a computer hardware project that introduces a special simulator achieving an easy-to-use micro programming environment and an user-friendly simulation tool. This tool presents a visualization environment in order to display the execution behavior of micro programs. It is a model/tool designed as a java program to ensure platform independence. This paper presents the Minimax simulator, which is used in the Minimax project. This project is a part of a hardware practical course. Therefore, it is a simulator for micro programming and hardware simulation. As a result, this simulator facilitates the process of micro-programming significantly enabling students to understand easily how a computer works. Here, two formal measures and metrics were presented to assess the implemented program, the execution time and the program length. Other results of this study showed how self-organized group work and project management can be accomplished.

Keywords: Simulator, micro-programming, machine language, Hardware Simulation, Computer Hardware Project.

محاكي للبرمجة المصغرة ومحاكاة الأجهزة المدمج في مشروع عتاد الكمبيوتر الصلب

ياسر شعبان
كلية العلوم والآداب || فرع العلا || جامعة طيبة || المملكة العربية السعودية

الملخص: لقد أصبحت البرمجة الدقيقة للآلات أكثر شيوعًا في الوقت الحاضر. إنها تقنية تُستخدم في العديد من المجالات مثل هندسة الكمبيوتر. تجدر الإشارة هنا إلى أن البرمجة المصغرة تستخدم في جميع مراحل عملية التصميم. بالإضافة إلى ذلك فإن تصميم وحدة التحكم في أجهزة الكمبيوتر الرقمية يحتاج إلى برمجة دقيقة، والتي هي أكثر تعقيدًا من لغات التجميع. في هذا المجال يمكن كتابة البرامج الصغيرة على شكل تسلسل من التعليمات المصغرة. في هذا السياق، يتطلب التدريس المتميز للبرمجة الدقيقة للآلات أداة محاكاة حاسوبية مناسبة ومختارة بعناية. يقوم هذا البحث بتصميم مشروع لأجهزة (عتاد صلب) الكمبيوتر الذي بدوره يقدم محاكيًا خاصًا ليحقق بيئة سهلة الاستخدام للبرمجة الدقيقة (المصغرة) وأداة محاكاة سهلة الاستعمال للمستخدمين. تقدم هذه الأداة بيئة تصورية (واجهة استخدام رسومية) لتوضيح سلوك تنفيذ البرامج الصغيرة. إنه نموذج/ أداة مصممة كبرنامج جافا لضمان استقلالية المنصة. تقدم هذه الورقة أداة محاكاة سمّيت (محاكي ميني ماكس) والتي تستخدم في مشروع سميّ (مشروع ميني ماكس). إنّ هذا المشروع هو جزء من كورس يهتم بالتطبيق العملي على العتاد الصلب للحاسب. لذلك هو يعتبر محاكيا للبرمجة المصغرة وأداة محاكاة لقطع العتاد الصلب بالحاسب. نتيجة لذلك، يسهّل هذا المحاكي عملية البرمجة المصغرة بشكل كبير مما يمكّن الطلاب من فهم كيفية عمل الكمبيوتر

بسهولة. هنا تم تقديم مقياسين شكليين وتم صياغة معيارين (معادلتين) لتقييم البرنامج المنفذ، هما مدة التنفيذ وطول البرنامج. وقد أظهرت نتائج أخرى لهذه الدراسة كيف يمكن إنجاز عمل جماعي ذاتي التنظيم و كذلك إدارة المشروع.

الكلمات المفتاحية: محاكي، البرمجة المصغرة، لغة الآلة، محاكاة العتاد الصلب، مشروع عتاد صلب الحاسوب.

Introduction

Micro programming is an important technique. It needs a micro programming language in order to teach several subjects in computer science. It helps understanding how compiler construction works. Computer Simulation Tools (CSTs) have become recently the most important research fields aiming to develop and simulate how machines or computer systems work [1]. CSTs use software packages (simulators) to emulate the behavior of machines and hardware systems (hardware simulation)[2]. These tools can be used by students to observe the execution of their programs. One of the most important tools is VHDL (VHSIC Hardware Description Language) [3]. It is a hardware description language used in the electronic design automation [4]. VHDL can define digital and mixed-signal systems. Another field of VHDL is parallel programming languages. In this context, students can test their design using VHDL that simulates machine execution [5].

Computer Simulation is an useful technique in order to understand the behavior of complex machines. Here, computer simulation models would be necessary to understand how the real system works and to design the individual components of the simulated system. In this context, teaching of micro programming can utilize computer simulation tools. In literatures, there are different computer simulation tools that can assist programmers (computer experts, teachers, students) in various diverse fields such as: software project management, computer hardware, computer architecture, telecommunications, networking, software engineering, etc [6] as described later in literature review and related work in this paper.

The motivation behind this research paper is to design a special user-friendly simulator so that micro programmers (also students) can have easy-to-use programming environment. Moreover, it manages a hardware project including a programming task where the given basic machine are simple. Therefore, this machine (hardware architecture) should be expanded using additional registers and/or new ALU commands, etc as described later.

Research problem and objectives

There are many computer simulation tools used to simulate machines or computer hardware. Here, the problem of research is how to provide micro programmers or students with a simple computer simulation tool in context of a hardware project to solve a complex programming task, where the given basic hardware architecture has to be expanded.

The specific objectives of this study are to designing a special project for students on one hand and a special simulator of micro programming for technical systems on the other. This represents an easy-to-use environment so that a strong understanding how a computer can be simulated by means of user-friendly tool (even at home). In this regard, also this study aims that self-organized group work and project management can be accomplished as explained later.

Literature Review and related work

This section presents a survey of the most closely related work (relevant state-of-the-art) that was published in the domain of micro programming or hardware simulation. Therefore, it is an overview of some architectures or approaches introduced concerning the micro programming and hardware simulation, defined especially for computer sciences education. This overview of existing related work aims to highlight the need of developing a novel concept to cover the gap recognized in the previous section (research problem).

The work in [7] presents a survey of microprogramming languages. It was a comprehensive survey presented from Balakrishnan and others, 1986, related to structured programming and efficient code generation. The survey aims to find a common way to define microprogramming language, vertical and horizontal microprogramming, local and global microcode compaction, and micro-architecture. As a result, this work arranged two comparison tables, so that the features of each language can be notified. These features are in the two categories: architecture specific and non-architecture specific.

Another work in [8] introduced micro-programming and Trickology. The author, Van der Poel, 1962, presented a basic principle that aims to take into account the growth of automatic programming in the context of micro-programming. This lends to problems in machine code programming and logical design. The programmer in this system owns direct access to the micro-programming of the machine. Here, some methods were showed to configure macro-instructions over a coding system. In this context, the ZEBRA code was chosen to refer to a particular machine code. The author presented the features of ZEBRA in [8]. The results have showed that, in order to devise the macro-instructions, a substantial ingenuity was needed and consequently the term "Trickology" was raised because of utilizing tricky programming.

Skrien has presented an interactive computer simulator in 1994. CPU Sim [9] is a computer simulator that was introduced for introductory computer organization-architecture. It runs on the Macintosh computer. The objective of this project was to develop a simulator that can be used as part of introductory courses, such as: machine organization and computer architecture. Here, it can be seen as an interactive low-level computer simulation program (software package). This software package will help the programmer (user) to define the simulated CPU, main memory, i/o channels, registers. In addition to that, assembly language instructions, machine instructions, and the micro-instruction set can be defined.

Consequently, as a result of this work, this simulator enables programmers (users) to write two kinds of programs, assembly language and machine programs by means of the built-in assembler and text editor.

A computer simulator that is used for systems programming courses was introduced in [10] from Newsome and Pancake in 1992. They aim to provide teachers with a graphical computer simulator so that a programming course becomes easier and not difficult anymore. "xSICSIH is an X-based graphical interface for the SICSIM computer simulation tool" [10]. It uses graphical elements of register-level components in order to remodel the SICSIM machine as "black box" to a "visual computer". Consequently, as a result of this project, it enables students to understand easily how a computer works. Additionally, xSICSIH uses many control features aiming to debug assembly language programs, such as: breakpoints, fast-execution, and Single step. More details of SICSIM can be found by Mihelič and Dobravec, 2015 in the work [11], where SICSIM was introduced as an interesting simulator used in a system-software course to simulate the educational SIC/XE computer.

An universal logic implementer introduced at Washington State University in [12] from Manwaring and others, 1991. The objective of their work was to develop a general purpose tool, which is used in laboratories of digital logic design. This tool is useful as a workstation for laboratory experiments, which belong to the computer architecture and digital logic courses at Washington State University. The results have showed that this tool has multi-faceted uses starting from simple gate-level implementations of logic circuits on one hand and the implementations of micro-programmed bit-slice processors on the other [12].

Similar to the simulation concept introduced in this research paper, it is relevant to mention an interesting work given in [6] focusing on utilizing computer simulation tools capabilities in computer sciences education. This work was presented by Alnoukari and others in 2013. They raised the question: "Are the simulation tools essential in Computer Sciences education or they are just a support method for helping the education process?" [6]. The following result was hereby obtained. Firstly, simulation should have a complementary role in existing educational techniques. Secondly, simulation tools play a role in helping programmers, teachers, students,.. etc, develop their practical skills in an engaging way. Thirdly, the authors think that using simulation will give more insight into aspects of computer science functions [6].

Although there are numerous works made towards building hardware simulation and Computer Simulation Tool (CST), defined especially for computer sciences education, a study of designing hardware project for students including a special simulator of micro programming of technical systems, does not exist yet (at least it is extremely rare published as a study).

To the best of our knowledge, this paper represents the first study towards designing student projects for learning and teaching micro programming using a hardware simulator.

Research Methodology and tools

The author relied on his research and efforts to introduce a special simulator for micro programming and hardware simulation as mini project. This project can be called a hardware project. Here, a general task has to be solved as a programming task in groups. That should be made on the basis of a given hardware machine (basic Minimax machine). This machine simulates how a computer works using a special simulator (Minimax simulator) designed for this problem domain. The simulator should possess a visualization environment so that the execution behavior of micro programs can be displayed. Furthermore, the basic structure of the given machine is not suitable (not enough) to solve the programming task and therefore should be expanded (additional registers, new ALU commands). The author presents the simulator as a model/tool designed as a java program. The java was used as programming language, because java will ensure platform independence. This simulator enables micro programmers to test every architectural extensions of the given hardware architecture.

In this context, it is imperative to carry out a thorough validity and reliability evaluation (check/analysis) of the methodology (system/solution). Therefore, two metrics (formal measures) are defined to assess the implemented program/architecture/algorithm/solution, so that its statistics will be trustworthy for research. These presented metrics are: the weighted runtime and the program length, which should also help to check the solution robustness. The author recommends to measure the metrics frequently based on given test-benches (test files as input for the simulator). Using such test-benches, worst-case and best-case can be estimated.

According to other objectives of this study, a self-organized project management methodology was selected for a group work as described later.

The next section introduces in detail the proposed system, the project computer engineering (mini project: Minimax machine).

Project Computer Engineering (Mini project: Minimax machine)

In this section, the Project Computer Engineering (mini project: Minimax machine) is presented. This project was created and carried out at department of Systems and Computer Architecture [13] at Leibniz University Hannover (LLH), Germany.

1- General task

In this mini-project, within the framework of the hardware project, the project participants will solve group-based programming tasks on the basis of the Minimax machine known from the lecture "Introduction to Computer Engineering". To solve the tasks, it is necessary here to suitably expand the given basic structure of the machine (new ALU commands, additional registers). Programming for problem solving will be done later on micro program as well as assembler level.

To solve the tasks, a Minimax simulator (as a Java program) is available, with which the project participants can also test their architectural extensions, micro- and assembler programming at home.

2- Minimax architecture

As part of the lecture "Fundamentals of Computer Engineering", basic knowledge of computer architecture is taught using the example of the Minimax machine (see Figure 1).

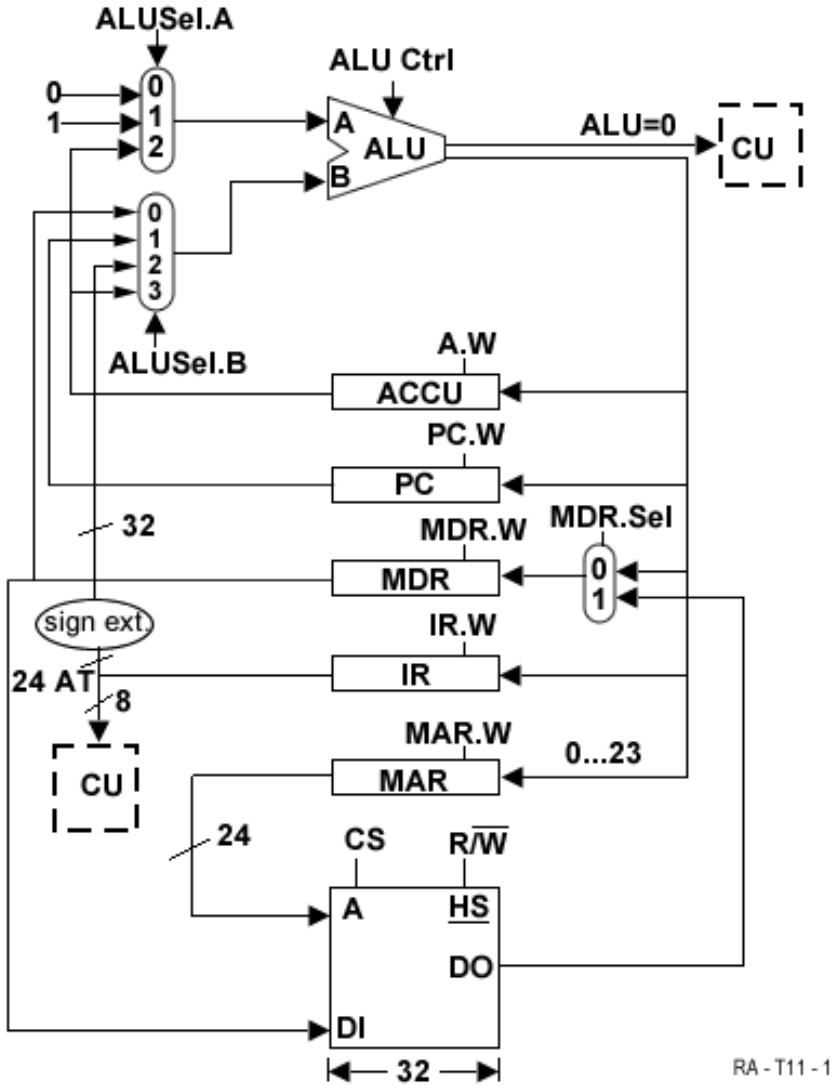

Figure (1) the basic Minimax machine [14]

This machine consists essentially of some registers, an ALU and a memory in which code and data are stored together. The command sequence(instruction cycle) is determined by a micro program. The architecture of the Minimax machine is largely fixed, but is extended for exam and practice tasks by additional registers or Sign Extension Units. Likewise, the ALU may be supplemented with additional functionality (e.g., DIV command). Figure 2 shows an extended Minimax machine.

By extending the basic structure of the Minimax machine, the specific assignment of the Minimax machine with control signals changes. With the Minimax simulator, it is possible to take these changes into account and to simulate the modified data path.

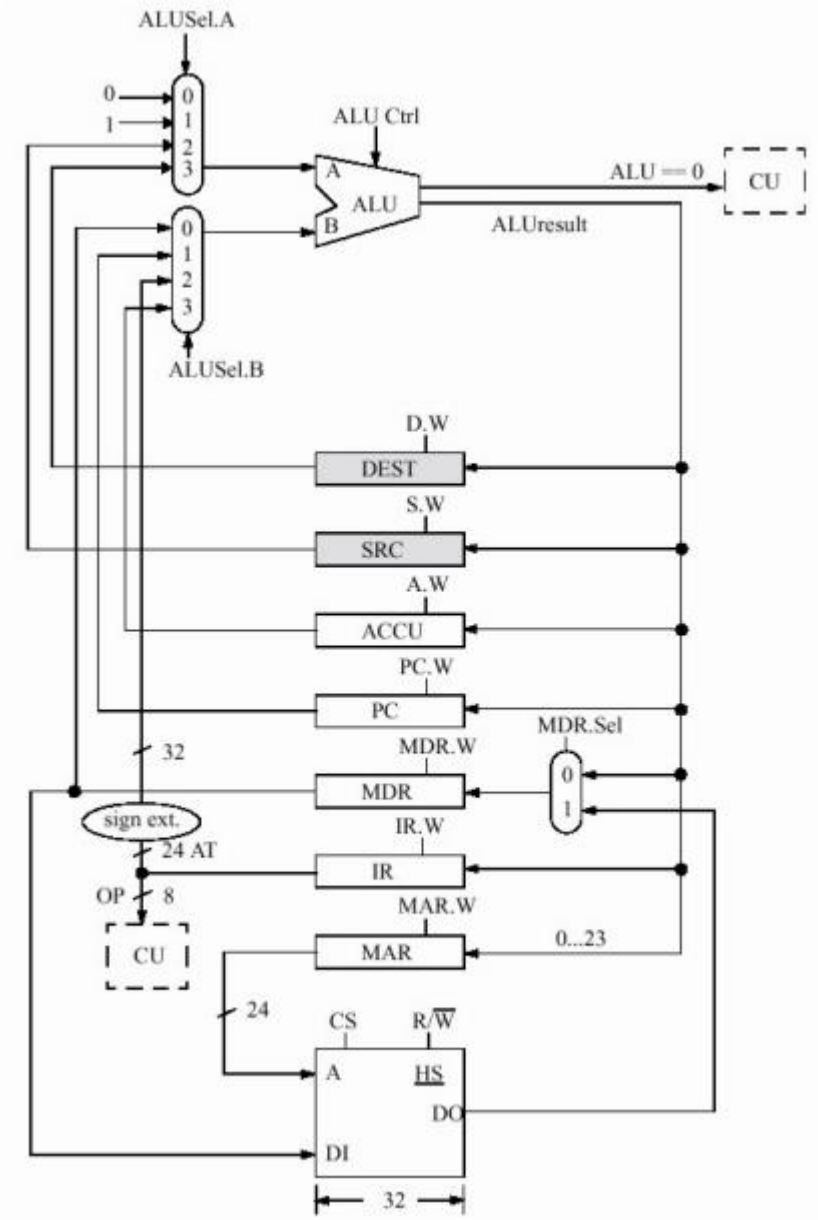

Figure (2) Extended Minimax machine [14]

For the concrete solution of the design task, the project group can decide which parts of the functionality should be moved into the micro program and how the machine program, based on it, looks like. Assessment standards are for the supervisor in addition to the error-free execution and the number of required clock cycles. The amount of resources used (additional registers and ALU commands) is also included in the evaluation.

3- The Minimax simulator

For the Minimax machine, there is a visualization environment for displaying the execution behavior of micro programs (see Fig. 3). The visualization environment reads a textual description of the specific Minimax-shaping (Minimax characteristic) and generates there from a clear graphical representation of the architecture to be simulated.

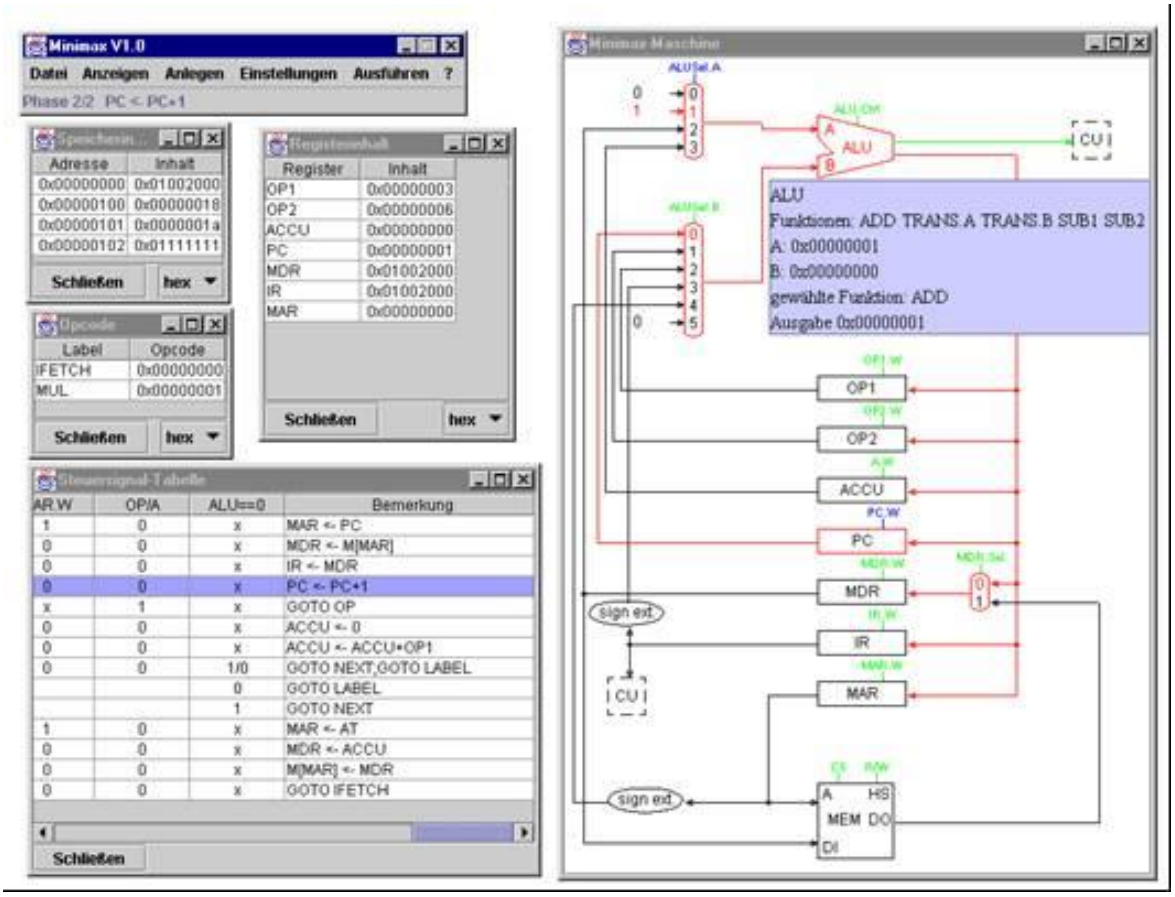

Figure (3) The Minimax simulator during execution of a machine program [15]

The user then has the opportunity to read micro programs and pre-configuration (default value(s) of registers and main memory addresses) to then simulate the execution of machine programs.

During this execution, which can be switched on step by step through pressing a button (Enter button, Return key), the occupied (allocated) resources are colored in the visualization. This allows a step-by-step simulation whereby the register and memory contents can be monitored at any time. It is also possible to set breakpoints in the simulator.

To ensure platform independence and use over the Internet, the implementation has been carried out as an application for Java 2 (Standard Edition, at least V 1.3.1).

4- Objectives of the mini-project

The mini-project deliberately differs from the "classic" laboratory experiments.. In each of these laboratory experiments, a completed subject area is dealt with on one day, whereby the activities (tasks) to be completed are strongly predefined. By contrast, the mini-project focuses on self-organized group work. The actual subject area of the Minimax machine is already familiar to all students from the course "Introduction to Computer Engineering". In the project, this knowledge is deepened and the degree of freedom is enhanced by the possibility to define an own architecture.

An essential part of the project work is the project management: how is the task distributed to the project participants, which steps are necessary for the solution, which documentation has to be created and how are the results finally best to be "sold" to the client (= project supervisor)? Soft skills are expected and trained by the students, which are difficult to be promoted in traditional courses, but are later valuable in their studies (for independent work) and at work.

5- Running of the project

The mini project is specified with a total cost of 32h, with 16h on preparation and 16h on actual laboratory work. The timing in the mini-project is largely determined by the project group, since not the entire 16h of laboratory work in the institute must be completed. In total, 4 people usually form a project group (in exceptional cases also 3 persons).

5.1 Introductory event (introductory launch event)/ distribution of the task

There will be an introductory event, which has the following sequence:

- Explanation of the task (the assignment).
- Appointment of a group leader who is responsible for communication with the supervisor.
- Announcement of the dates (Meeting or Deadline or....etc):
  - office hours (possibly by email).
  - Submission of the system specification (product brief or product specification) and next project meetings.
  - Submission of the solution (code stored in form of files) with documentation.
  - Final presentation (in front of group participants).

5.2 Workflow (work steps) during project work

5.2.1 Analysis phase (system specifications)

The project planning is to be carried out in the group. A specification for the task should be developed. The specifications must be submitted to the supervisor by a given date. Among others, it should answer the following questions:

- What is the exact task and what are the requirements (actual analysis of the given Minimax machine)?
- Which machine commands should be implemented for task solving and what should they perform in each case (Opcode, mnemonic, register assignment, short description)?
- Is a special algorithm used in the solution (why this one)?
- Which tools were used to solve the task (Software, literature)?

- Which results are to be delivered at the end of the project (files, documentation)?
- What are the individual work steps for managing the project and how are the tasks distributed to the individual project persons (schedule with hourly data per project participant)?

5.2.2 Concept and specification

The necessary expansions of the given basic machine must be documented. For each machine instruction implemented, a specification must be created for the associated micro program which clarifies the basic flow or sequence (for example, by a flow chart). The machine program for solving the overall task has also to be suitably documented.

5.2.3 Realization and test

By creating configuration files, the simulator must be adapted to this new architecture. The individual machine commands are to be implemented and tested by micro programs on this architecture. If the individual machine commands have been individually tested for correctness, the entire program must be implemented. This entire program should also be tested for error-freeness (accuracy) with suitable input values.

5.2.4 Assessment of the implemented solution

During the solution, the execution time in clock cycles on the Minimax machine has to be to determined. This should be based on the given test- benches and also a worst-case and best-case estimate has to be made.

The (weighted) runtime and program length of the solution must be calculated, which is composed of resources used, control memory size and execution speed.

5.2.5 Creating the documentation

The final report has to be prepared. In addition to the introduction (which again contains the task and the objective of the work) and the rough specification of the micro programs in the main part, this documentation includes a detailed description of the implemented micro programs. The documentation has to be designed in such a way that another developer can get familiar with the programs quickly and easily.

Likewise, the developed machine programs and tests are to be documented. Finally, the developed solution should be evaluated and (if possible) suggestions for improvement should be given

5.2.6 Final presentation

The project concludes with a 20-minute lecture in which the group presents the task being performed and its solution in front of the other groups and the project supervisor. Finally, the Minimax simulator should demonstrate the functionality of the developed solution.

5.2.7 Proof of successful participation in the mini-project

The mini-project will be graded as passed if all partial performances have been successfully completed:

- specification
- Functional (and tested and debugged) solution with all configuration and test files
- Project documentation
- final presentation

If partial performances flawed, additional tests may be carried out by the project supervisor.

Evaluation of the solution

- reg: number of supplemented Minimax registers
- se: number of supplemented sign extension units (0 or 1)
- const: number of added constants (one "0" and one "1" free)
- alu_add: Penalty sum for all added ALU commands
- alu_use: Penalty sum the used ALU commands in the program.

Runtime in Minimax-clocks

The weighted runtime of the solution in Minimax-clocks should be calculated using the next formula:

$$t\_evaluated = (t\_bench * (1 + 0.1*reg + 0.15*se + 0.015*alu\_add + 0.05*const))$$

where ***t_bench*** is the execution speed, which is the number of the Minimax-clocks that is needed to complete the running of the solution (program).

program length (length of the algorithm)

The program length (length of the algorithm) of the solution should be calculated using the next formula:

$$n\_evaluated = n\_algorithm + 5*reg + 10*se + 3*alu\_use + 5*const$$

where ***n_algorithm*** is the number of lines of the algorithm (solution).

additional ALU operations

The penalty of all added (alu_add) and used (alu_use) ALU commands (additional ALU operations) in the program can be taken from the next table:

Table (1) penalty of additional ALU operations

additional ALU operations	SUB1	INC/DEC	S.L, S.R	AND, OR, NOT	XOR	DIV	Custom
Penalty (1..20)	1	4	5	6	8	10	Up to 20

Results

Since this paper deals with a project model that manages a simulator of micro programming in the context of hardware simulations, the results will be presented in a special way showing how the project should be managed, designed and realized. Furthermore, it shows which evaluation metrics should be used to measure the accuracy of the introduced system/solution. Finally, some calculations and measurements of t_bench, t_evaluated and n_evaluate were reported based on various given test-benches.

As a result, this paper introduced a simulator used for micro programming and hardware simulation paying attention to designing it as a hardware project. Here, the goal is to develop an easy way to understand how a computer works. The total cost of running the project was 32h (16h on preparation and 16h on actual laboratory work). Here, the 16h of laboratory work should not be completed in the institute, because the project group will plan the timing for phases. Usually, a project group has 4 students. The hardware project focuses on self-organized group work. Furthermore, the project management constitutes an essential part of the project work. In this regard, there are several work steps during project work: analysis phase (system specifications), concept and specification, realization and test, assessment of the implemented solution, creating the documentation, final presentation, proof of successful participation in the hardware project.

Since students have the possibility to define their own system architecture /solution, the simulator accepts a new computer/machine architecture that is expanded in order to solve a complex programming task. That should be through creating configuration files that can be loaded into the simulator. Additionally, users can implement the machine commands and test them.

In order to evaluate the implemented solution, two metrics were presented. Firstly, the execution time metric (weighted runtime) of the solution in clock cycles (Minimax-clocks) on the Minimax machine. Secondly, the program length metric (length of the algorithm) of the solution. Both metrics should be measured frequently and reevaluated according to the given test-benches so that estimation of worst-case and best-case can be done. Two formulas were defined for these metrics in order to calculate t_evaluated

(runtime) and n_evaluated (program length). These formulas depend on resources used, control memory size and execution speed.

The calculation of execution times was based on the different given test-benches. Therefore, the measurement was repeated for every test-bench. Here, test-benches are test files that are given as input for the simulator. Some values of the measured t_bench (the execution speed in Minimax-clocks) were: 2317 clocks, 3270 clocks, 4215 clocks, 5115 clocks. Accordingly, the values of the calculated t_evaluated (the weighted runtime of the solution) were: 9515 clocks, 10240 clocks, 12280 clocks, 14570 clocks respectively. On the other hand, the values of the calculated n_evaluated (program length or length of algorithm) were: 273, 317, 457.

Discussion

In this paper, the assessment of the implemented program (algorithm, solution) was conducted using the given test-benches. It is noteworthy that the definition of the solution robustness varies according to the context, in which the system (a given task or problem) is used. Therefore, many meanings of solution robustness were introduced in this paper. Furthermore, various formal measures and metrics were defined to achieve the solution robustness. Here, the entire program should be tested for accuracy (error-freeness) using various suitable input values.

If a comparison is made between two different implementations of the algorithm, it is obvious that all values of t_bench, t_evaluated and n_evaluate will be different each time. So, a smarter solution/ implementation definitely requires less time and cost. Here, the number of supplemented Minimax registers, the number of supplemented sign extension units, the number of added constants, the number of added ALU commands, the number of used ALU commands in the program should be minimized as much as possible. Moreover, if additional ALU operations are required for the implementation, then the programmer should choose those operations that are equitable (least expensive).

Conclusion

The goal of Computer Simulation Tool (CST) is to develop a suitable tool helping in simulating machines and micro-programming. This is particularly interesting to help students learning hardware simulation and testing their program execution.

In this paper, a hardware simulator called "Minimax Simulator" was introduced. The focus was the study of designing student project including the design of a special simulator for micro programming tasks in the context of hardware simulation. This leads in turn to strongly understanding how a computer or program works.

The basic features of the Minimax simulator include easy-to-use environment, generality of applications, and robust executions.

In this regard, the basic machine (architecture, see Figure 1) can be extended. Consequently, the project participants are able to test their architectural extensions of the basic machine, micro- and assembler programming at home. Additionally, the Minimax simulator represents the functionality of the developed solution of the given task.

The development and the evaluation of this Minimax project were made using different metrics of solution/system performance. In order to evaluate the implemented micro-program, two formal measures and metrics were defined, the execution time and the program length. In this regard, different given test files (test-benches as input for the simulator) were used.

On the other hand, this research paper clarified how project management, which contains a self-organizing group work and a complex programming task, can be successfully implemented.

Review of related literature and studies concerning hardware simulation and micro programming, especially introduced for computer sciences education, drew attention to the requirement of a novel concept to fill the gap considered in the research problem of this paper. This research paper aimed to fill a part of this research gap by providing students (micro programmers) with the combination of an easy-to-use simulator and a complex programming task in the context of a hardware project. In this project, the given hardware architecture is suitable for small computational problems and consequently has to be expanded.

Finally, after the conclusion of this paper was drawn, a peek at future trends for follow-up research papers can be given.

Due to the vastness of this topic, it should be paid attention that only some aspects of the topic under study in this paper could be considered and therefore they were covered in the investigation.

In this paper, the idea of integrating the Minimax simulator in a hardware project was introduced. Therefore, the next step is to continue with the design and implementation of special hardware architectures aiming to completely realize our vision. Then, the implemented solution/system performance should be measured and reported based on standards. This leads in turn to determine whether the implemented architecture/system performance will be robust, weakly robust, or maybe even not robust.

One aspect that may be of interest for future work is the integration of concepts from different research areas into the Minimax project. These various concepts will form a practically applicable methodology.

References:

[1] Cvetkovic D. Modeling and computer simulation. Book, published in London, United Kingdom, IntechOpen 2019; ISBN: 9781789857955.

[2] Franciosa P, Abhishek D, Bolognese L, Charles M, Mistry A. Design Synthesis Methodology for Dimensional Management of Assembly Process with Compliant non-Ideal Parts, In: 2014 Joint Conference on Mechanical, Design Engineering & Advanced Manufacturing, 2014 June 18th–20th; Toulouse, France.

[3] Shahdad M, Lipsett R, Marschner E, Sheehan K, Cohen H. VHSIC Hardware Description Language. Computer 1985; vol. 18, no. 2, Pages: 94-103, IEEE. Doi: 10.1109/MC.1985.1662802.

[4] Heinkel U and others. The VHDL Reference: A Practical Guide to Computer-Aided Integrated Circuit Design Including VHDL-AMS. published by John Wiley & Sons, Ltd., 2000; ISBN 0-471-89972-0.

[5] Multiplier optimization in VHDL. [Internet] 2019. Available from: https://surf-vhdl.com/multiplier-optimization-in-vhdl/.

[6] Alnoukari M, Shafaamry M, Aytouni K. Simulation for Computer Sciences Education. Communications of the ACS Journal 2013; Vol. 6, N0.1.

[7] Balakrishnan M, Madan B, Bhatt P. A survey of microprogramming languages. Microprocessing and Microprogramming, ScienceDirect 1986; ISSN: 0165-6074, Vol: 17, Issue: 1, Pages: 19-27.

[8] Van der Poel W.L. Micro-programming and Trickology. In: Hoffmann W. (eds) Digitale Informationswandler/Digital Information Processors/Dispositifs traitant des informations numériques. Vieweg+Teubner Verlag, Wiesbaden, 1962. DOI: https://doi.org/10.1007/978-3-322-96260-7_7

[9] Skrien D. J. CPU Sim a computer simulator for use in an introductory computer organization-architecture class. Journal of Computing in Higher Education, September 1994; Volume 6, Issue 1, Pages: 3–13. https://doi.org/10.1007/BF03035480.

[10] Newsome M, Pancake C. A graphical computer simulator for systems programming courses. ACM SIGCSE Bulletin, 1992; 24(1), Pages:157–162.

[11] Mihelič J, Dobravec T. SICSIM: A simulator of the educational SIC/XE computer for a system-software course. Computer Applications in Engineering Education 2015; 23, pages: 137-146.

[12] Manwaring M. L, Baker W. I, Malbasa, V. D, Seamans D. A. Universal logic implementer: a general purpose tool for a digital logic design laboratory. IEEE Transactions on Education 1991; vol. 34, no. 2, Pages: 184-188. Doi: 10.1109/13.81599.

[13] Department of Systems and Computer Architecture , Leibniz University Hannover, Germany. [Internet] 2019. Available from: https://www.sra.uni-hannover.de/Lehre/WS19/L_MMM/.

[14] Department of Systems and Computer Architecture, Leibniz University Hannover, Germany. [Internet] 2019. Lecture: Fundamentals of Computer Architecture, unpublished manuscript.

[15] Department of Systems and Computer Architecture , Leibniz University Hannover, Germany. [Internet] 2019. Mini project: Minimax machine, unpublished manuscript.