
Software package for an educational computer system

Nenad Grbanovic, Jovan Djordjevic and Bosko Nikolic

Faculty of Electrical Engineering, University of Belgrade, Belgrade, Serbia and Montenegro

E-mail: jdjordjevic@etf.bg.ac.yu

Abstract Some aspects concerning the realisation of a software package for an educational computer system are considered. First, the rationale behind the development of the educational computer system and software is given. Then the basic characteristics of the educational computer system and typical user features of the software are described. The reasons for the choice of programming environment used in the realisation of the software package are discussed. Finally, the basic elements of the proposed software package are given.

Keywords computer architecture; digital circuits; education; visual simulator

Courses in digital circuits and computer architecture and organisation are the first two courses in the area of computer engineering taken by the students of all departments at the Faculty of Electrical Engineering, University of Belgrade. The first course covers switching functions, combinatorial and sequential switching circuits, methods used to carry out the analyses and syntheses of switching circuits and standard combinatorial and sequential modules such as multiplexers, decoders, arithmetic and logic units, registers, counters, etc. The second course deals with the computer system elements relevant to both the assembly language programmer and the computer system designer, referred to as computer architecture and computer organisation, respectively. The computer architecture elements considered include programmable registers, data types, instruction formats, addressing modes, instruction sets and interrupt mechanisms. The computer organisation elements studied are various techniques applied to design and connect computer system modules using combinatorial and sequential circuits.¹

Students attending the course in computer architecture and organisation are faced with the problem of implementing their theoretical knowledge of combinatorial and sequential circuits and concepts of computer architecture and organisation in the design of various computer system modules and their integration into a computer system. A common approach taken to tackle this problem has been to organise practical exercises in the laboratory using a suitable computer system simulator. A survey of the open literature indicates a variety of computer system simulators.² They have been analysed on the basis of three basic criteria: the simulator should be interactive, with a visual presentation; the computer system should incorporate the majority of topics lectured on in the course; and the simulator should provide presentation of the computer system down to the level of combinatorial and sequential circuits.

Some interesting simulators, such as the Newport simulator,³ were eliminated since they do not provide a visual presentation. The remaining simulators can be

grouped into simulators of systems having simple architectures and organisation, simulators of systems having vary complex organisation, and simulators of systems at the global level. The first group simulates computer systems that include only basic elements of computer architecture and use very simple techniques of computer organisation.⁴ ESCAPE is an interactive environment with the possibility to simulate two processors with microprogrammed and pipelined organisation and the same set of instructions. The second group deals mainly with the problem of the design and performance analysis of pipeline and multiprocessor systems.⁵ DLXview is an interactive pipeline simulator, which uses the DLX set of instructions and simulates three versions of the DLX pipeline: basic, scoreboard and Tomasulo. The third group makes it possible to configure systems having a wide range of complexity by linking already simulated modules.⁶ HASE includes a set of tools which allows one to configure computer systems by linking in-built modules and to simulate the systems so obtained. As a result of a critical analysis of existing simulators it was concluded that none of them met the above stated requirements. The first group of simulators did not cover all the topics lectured, the second group did not demonstrate basic but very sophisticated issues, and the third group of simulators did not show the system at the level of combinatorial and sequential circuits.

As a solution to the above problem an educational computer system has been devised, a manual for it written, a software package developed and laboratory exercises prepared.⁷ The educational computer system has been devised with the aim of incorporating the computer architecture and organisation concepts lectured on in the course. The manual has been written to explain the concepts and to illustrate implementation details. The software package has been developed to simulate the behaviour of the educational computer system, visually present its parts at the level of combinatorial and sequential circuits, and provide various user features. The laboratory exercises have been prepared to demonstrate, using the software package, the concepts implemented in the educational computer system. The features of both the educational computer system and the software package, as well as the organisation of laboratory exercises, are described elsewhere.⁷ Some aspects concerning realisation of the software package of the educational computer system are considered in the following section.

Educational computer system

The educational computer system is made up of the following modules: processor, memory, input/output units and system bus (Fig. 1).

The processor architecture is the load/store type. The programmable registers are 16 general-purpose registers and the program counter (PC), the stack pointer (SP), the program status word (PSW), etc. Data types are 16-bit signed and unsigned integers. The instruction format three address with instruction length 16 and 32 bits. The addressing modes provided for the Load and Store instructions are immediate, memory direct, register indirect, and register indirect with displacement, while all remaining instructions implicitly use the register direct addressing mode. The instruction set includes the transfer, arithmetic, logic, shift, rotate and control instruc-

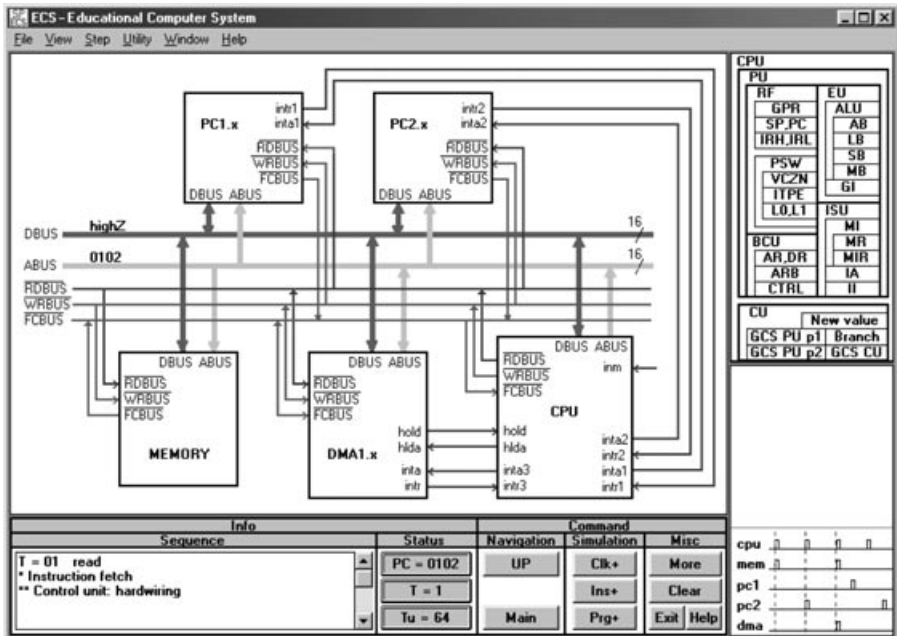


Fig. 1 The educational computer system.

tions. The interrupt mechanism is vectorised with internal and external interrupts. The processor organisation has the form processing unit (PU) and control unit (CU). The processing unit is made up of the register file unit (RF), execution unit (EU), interrupt service unit (ISU), and bus interface unit (BIU) interconnected via the 16-bit internal bus. The processing unit can work with one of four types of control unit. One uses a hardwired technique, while the remaining three use a microprogramming technique with mixed and vertical formats of microinstructions and nanoprogramming.

The memory has a capacity of 128 Kbytes and word length of 16 bits.

The input/output units include 12 peripheral devices and controllers organised in three groups denoted PC1.x, PC2.x and DMA1.x, where $x = 1, \dots, 4$. Groups PC1.x and PC2.x can have up to four peripheral devices with non-direct memory access controllers, and group DMA1.x four peripheral device controllers with direct memory access controllers. Input/output units within a group send their interrupt request signals to the processor via a common line and receive the interrupt acknowledge signal from the processor in a daisy-chained manner.

The system bus is asynchronous. It includes 16 address lines ABUS, 16 data lines DBUS and control signals RDBUS, WRBUS and FCBUS specifying the start of a read or write cycle and the completion of a cycle, respectively. The bus master can be the processor and any of the four DMA input/output units. Arbitration between them is carried out by the processor. DMA input/output units within a group send

their hold request signals to the processor via a common line and receive the hold acknowledge signal from the processor in a daisy-chained manner.

The processor, memory and input/output units in each of the three groups PC1.x, PC2.x and DMA1.x work asynchronously with five different clock periods denoted as C_{cpu} , C_{mem} , C_{pc1} , C_{pc2} and C_{dma} , respectively.

Software package features

The software package features facilitate simulation set up and simulation control.

Simulation set up

The simulation set up features make it possible to configure and initialise the software package.

The features provided to configure the software package offer three groups of facilities. The first one is to specify the number of input/output units in each of the three groups PC1.x, PC2.x and DMA1.x (Fig. 2). The second one is to define clock periods C_{cpu} , C_{mem} , C_{pc1} , C_{pc2} and C_{dma} of the computer system modules and the access times of peripheral devices and memory as multiples of time units t_u . The third one selects 18 signals for which the timing diagram can be drawn.

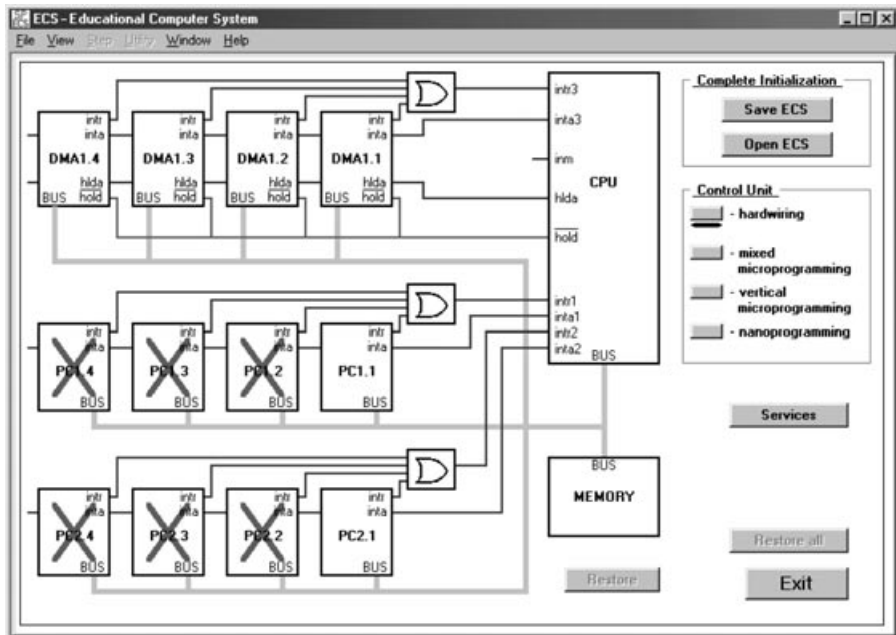


Fig. 2 Configuring the computer system.

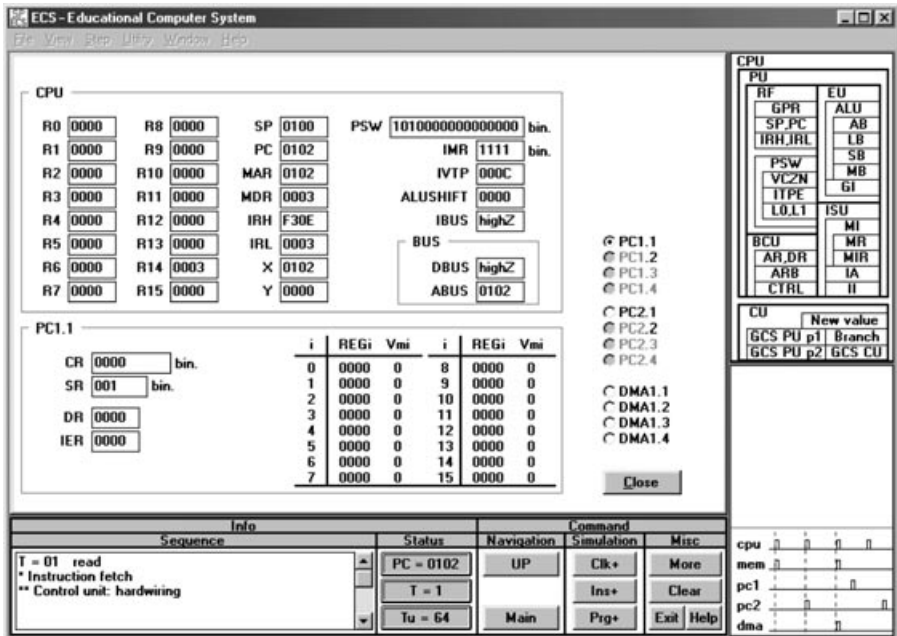


Fig. 3 Initialisation of the computer system.

The features provided to initialise the software package allow one to specify initial values for the processor, memory and peripheral devices and their controllers (Fig. 3). Initialisation of the processor involves loading the program counter, interrupt vector table pointer, interrupt mask register, stack pointer, general purpose registers, etc. Initialisation of the memory includes loading the appropriate memory locations with binary values of programs, data to be used during execution of the program, addresses of interrupt routines in the interrupt vector table, data to be sent to the output units, etc. Initialisation of the peripheral devices and their controllers includes loading both simulated input peripheral devices with data and also the registers of their controllers.

Simulation set up can be carried out either partially or completely. If partial simulation set up is chosen one can separately configure the computer system and separately initialise each computer system module. If complete simulation set up is chosen one can configure the computer system and initialise all computer system modules at once. Partial and complete simulation set up can be carried out either interactively or from files. Interactive simulation set up includes direct typing of appropriate values from the terminal. In addition, the interactive initialisation of memory locations can be carried out using the Editor and Translator of assembly language programs. Simulation set up from files involves the reading of files prepared earlier. The result of a simulation set up can be saved in files partially for each computer system module or completely for the complete computer system.

Simulation control

The simulation control features allow one to simulate the behaviour of the computer system when executing a program, visually present parts of the computer system and values of signals at various levels of detail and examine simulation results. Each screen, in general, is made up of four windows and the menu (Fig. 1).

The largest window in the upper left part of the screen, named the Block diagram window, shows parts of the computer system. Because only a limited number of elements can be displayed on screen, a hierarchical scheme of screens of computer system blocks is developed. Screens that do not belong to the highest level in the hierarchy of screens, show a composition of blocks and combinatorial and sequential circuits. Screens which present the highest level in the hierarchy of screens, show a composition of combinatorial and sequential circuits. The first screen, which the simulation begins with, gives the structure of the computer system at block level (Fig. 1). For a detailed structure of any of the blocks one needs to go one level lower in the screen hierarchy, by positioning the cursor at that block and clicking the mouse. Repetition of this action as many times as needed brings the screen to the most detailed (highest) level in the hierarchy (Fig. 4).

The upper right window, named the Hierarchy window, shows the hierarchical scheme of processor (CPU), which contains blocks of processing (PU) and control (CU) units. Each block contains further blocks down to blocks at the highest hierarchical level. The screen of any of these blocks can be directly selected for display in the Block diagram window by positioning the cursor at that block in the

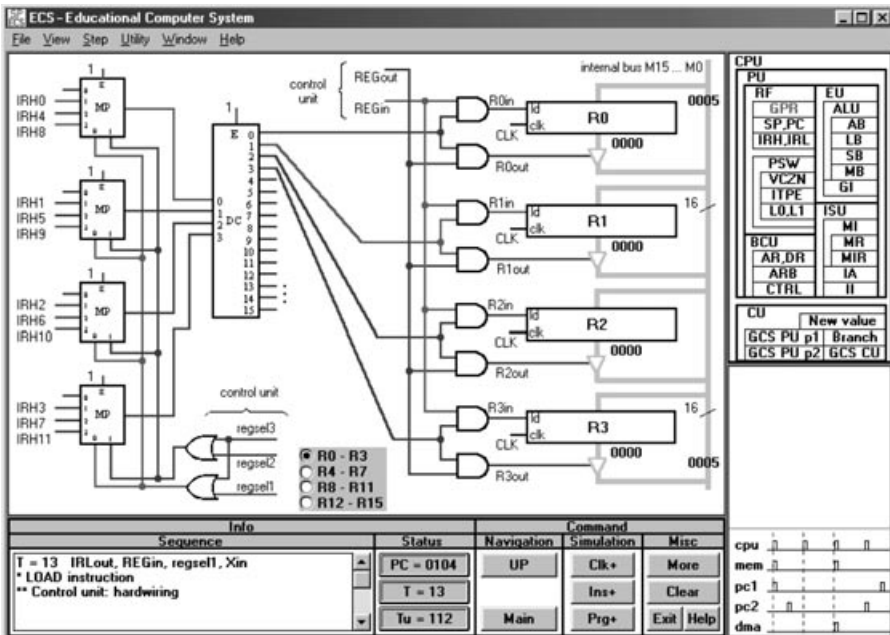


Fig. 4 The general purpose registers.

Hierarchy window and clicking the mouse. The name of the block shown in the Block diagram window is given in red (GPR) and the others in black (Fig. 4).

The lower left window, named the Info and Command window, is divided into the Info window at its left-hand side and the Command window at its right-hand side. The Info window contains the Sequence window and Status buttons. The Sequence window gives the value of either the step counter or the microprogram counter, then the control signals generated for that clock period and, finally, a brief explanation of the actions to take place during that clock period. The Status buttons PC, T and tu display the current values of the program counter, the step counter or the microprogram counter and the number of time units elapsed, respectively. The Command window contains three groups of command buttons: Navigation, Simulation and Miscellaneous. The Navigation command button UP allows one to move from the current screen to the screen one level up in the hierarchy, while button Main brings the screen back to the structure of the computer system (Fig. 1). The Simulation command buttons Clk+, Ins+ and Prg+ allow one to move on with the simulation a certain number of time units. Button Clk+ moves the simulation on by one clock period. Buttons Ins+ or Prg+ move it on until an instruction or a program have been executed, respectively. The Miscellaneous command buttons are More, Clear, Exit and Help. Button More opens a window that allows one to select one of three screens and to examine and set the values of memory locations and registers of the processor and peripheral devices and their controllers (Fig. 2), and to draw the timing diagrams of selected signals (Fig. 5). Button Clear returns the simulation to the

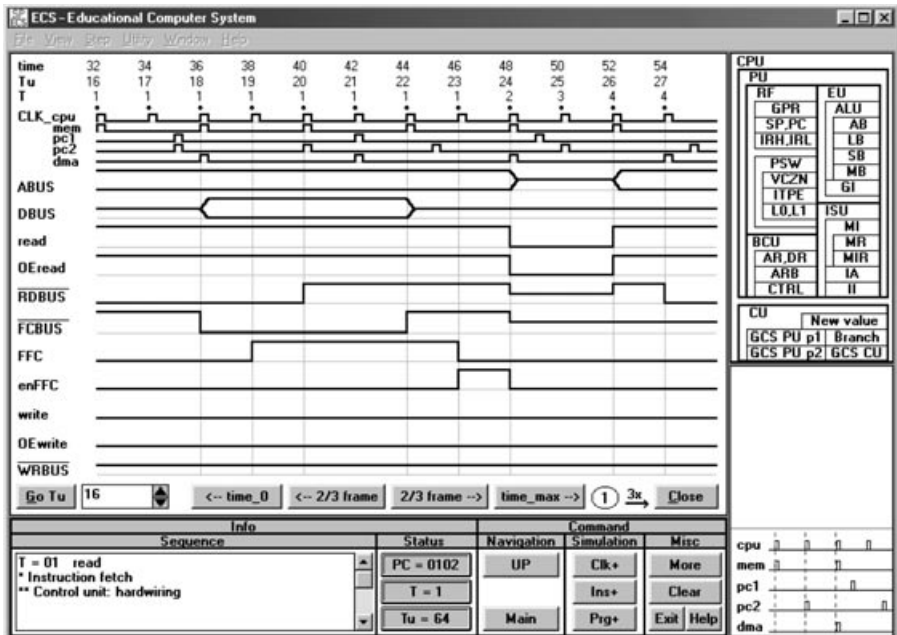


Fig. 5 Signal timing diagram.

beginning. Button Exit makes it possible to save the current state of simulation in a file and quit the simulation. Button Help activates the help system where details concerning the computer system and the software package are given.

The lower right window, named the Modules clocks window, allows one to view the timing diagrams of computer system module clocks at the moment when the simulation is stopped. The red vertical line points to clock(s) of those computer system module(s) which were operating when the stop occurred. The next two green lines point to the clocks of computer system module(s) where the next two simulation stops will occur.

The menu, which is permanently present on the screen, facilitates the same functions as those offered by the Command window buttons, and is provided for those who prefer to work with menus rather than buttons.

Programming environment

The choice of programming environment used for the development of the software package was made a few years ago when development of the software package started. It was the result of requirements imposed by the software package features and the possibilities of operating systems and programming languages available at that time. The basic software package requirements were to visually present parts of the computer system which are very complicated and contain many elements, simulate the behaviour of sequential and combinational circuits, and present the values of all signals, etc. For the operating system, the aim was to use one which would make it possible to run the software package on a large number of installations and provide the necessary services to the chosen visually orientated programming language. The programming language had to facilitate an efficient realisation of the software package features and provide satisfactory speed of execution and low demands on resources.

The choice of Windows as the operating system was a straightforward one, since it meets the above stated requirements.

The choice of Visual Basic as the programming language was the result of careful considerations.⁸ The requirement to visually present parts of the computer system which are very complicated and contain many elements reduced the choice to the visual programming languages. The candidate languages, such as Visual Basic, C, C++, Java and Delphi, have the possibility to draw complicated computer system structures with many elements and link them efficiently with the appropriate simulation code. The execution speeds of applications are similar, although Visual Basic is slightly better and Java slightly worse than the others. In addition, Visual Basic applications have the lowest demands on hardware and software resources. Finally, there was much greater experience in working with Visual Basic than with the other languages, which gave the decisive advantage to Visual Basic over other candidates.

Thus, the software package was developed with the *MS Visual Basic 3.00* programming language on the *MS Windows 3.1x* graphical environment and can run on computers with *MS Windows* operating systems.

Software package realisation

The software package is realised as a standard Windows application, which includes three basic parts: the Default State Set Up, the Simulation Set Up and the Simulation Control (Fig. 6). The Default State Set Up is executed at the start of the software package in order to bring it into the default state. The Simulation Set Up part is executed in cases when a user wants to configure and initialise the software package to a state different from the default state. The Simulation Control part is executed when a user utilises the navigation, simulation or miscellaneous features.

The software package is realised as an interactive event-driven environment. Therefore, execution of Simulation Set Up and Simulation Control begins as a reaction of the event listener to some event. In the software package events are created as the result of either the activation of the command buttons or the clicking the mouse at the appropriate block.

Default State Set Up

The Default State Set Up is executed when the software package is started and as a result its variables are initialised to default values.

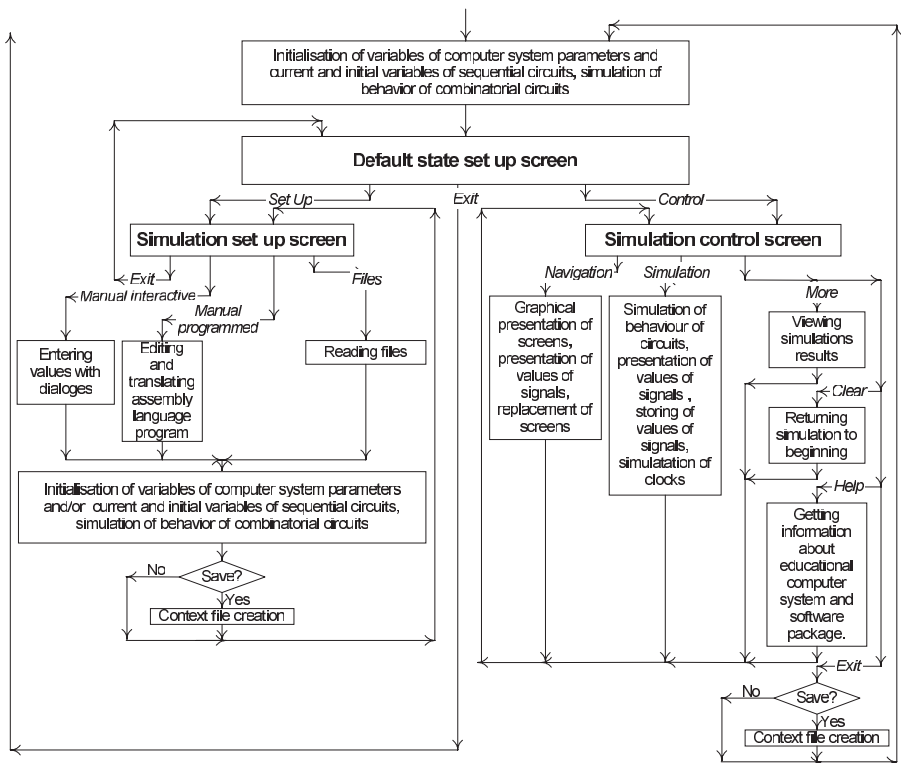


Fig. 6 Software package structure.

The software package has computer system parameter variables, current and initial sequential circuit variables and combinatorial circuit variables. The computer system parameter variables define the number of input/output units, the type of control unit, the clock rates of computer system modules and the access times of peripheral devices and memory. The current and initial sequential circuit variables and the combinatorial circuit variables describe the sequential and combinatorial circuits of the processor, the memory, the peripheral devices and their controllers. During Default State Set Up execution, first, the computer system parameter variables and the current and initial sequential circuit variables are initialised, and then the values of combinatorial circuit variables are calculated based on the current values of sequential circuit variables by executing procedures for simulation of combinatorial circuits.

It should be noted that the values of computer system parameter variables are not changed during a simulation, since they hold information about the computer system configuration. The current sequential circuit variables and combinatorial circuit variables are changed during the simulation in order to reflect changes that occur in the sequential and combinatorial circuits of the computer system modules. The initial sequential circuit variables are not changed during the simulation since their values are used, when the simulation returns to the beginning, to restore the values of current sequential circuit variables.

Default State Set Up execution finishes with the Default State Set Up Screen, which gives basic informations about the software package and contains the Set Up and Control command buttons. Activation of these buttons makes it possible to go to Simulation Set Up and Simulation Control, the execution of which begins with the appearance of the appropriate screen.

Simulation Set Up

Simulation Set Up is executed when a user wants to configure and initialise the software package. It allows one to change the default values of computer system parameter variables, change the default values of sequential circuit variables of the processor, the memory and the peripheral devices and their controllers. Simulation Set Up can be carried out either interactively or from files.

The interactive set up may be performed manually or may be programmed. Manual interactive set up is performed using a system of dialogues, which are realised as forms of sequences of *Visual Basic* Text Box objects. There are separate dialogues to configure the software package and to initialise each module of the computer system. Programmed interactive set up is performed by writing assembly language programs using the editor and translator. It can be used to initialise only the memory module. During the interactive set up the same values are written into both the current and initial variables of a particular sequential circuit.

Simulation Set Up from files is carried out by changing the default values of some or all variables with values read from either a context file of the complete computer system or context files of computer system modules. Context files of the complete computer system keep the values of parameters and the current and initial values of sequential circuits of all modules of the computer system. Context files of the com-

puter system modules keep separately the values of sequential circuits only for the processor, memory and peripheral devices and their controllers. Simulation Set Up from files can be complete or partial. In the case of complete set up the contents of the specified context file of the complete computer system are copied over the default values of parameter variables and all current and initial sequential circuit variables. In the case of partial set up the contents of the specified context file of a computer system module are copied into both current and initial sequential circuit variables of the appropriate part of the software package.

Context files of the complete computer system and of computer system modules can be created during Simulation Set Up by complete and partial saves in files, respectively. A context file of the complete computer system saves the values of parameter variables and initial and current sequential circuit variables of the software package. A context file of a computer system module saves the values of current sequential circuit variables.

Simulation control

Simulation Control is executed when a user performs the navigation, the simulation and the miscellaneous user features.

Navigation

Navigation makes it possible to display parts of the computer system in the Block diagram window. It is activated by the Navigation command buttons Up and Main in the Command window or by clicking the mouse at the appropriate block in the Block diagram or Hierarchy windows. Navigation is performed with the hierarchical scheme of screens, the realisation of which includes parts for the graphical presentation of screens, the presentation of signal values and the replacement of screens.

The graphical presentation of screens is realised in such a way that a separate screen form is created for each screen using various Visual Basic objects. Parts of the computer system presented as blocks are realised with Shape objects (sequential and combinatorial modules, such as registers, counters, multiplexers, decoders, etc.), flip-flops and AND, OR, XOR and NOT elements with Image objects, single lines and groups of lines with Line objects and texts denoting hexadecimal values for groups of lines and names of signals, elements, etc. with Label objects. In addition to that some other *Visual Basic* objects are used, but not as often as those mentioned. The total number of forms used for presenting screens is almost 200.

The presentation of signal values is carried out by form update procedures created separately for each screen form. Every time a screen is replaced, a new form is opened and its form update procedure called. The procedure updates the values of colours of Line objects and texts of Label objects of the currently active screen form. Depending on the signal values at the outputs of sequential and combinatorial circuits the procedure updates the colours of single lines and groups of lines and texts with hexadecimal values for groups of lines. A single line is presented with a thin line in blue, red or green depending on whether the signal on the line is inactive, active or in a high impedance state. Groups of lines are presented as thick

lines in grey and hexadecimal values, if they are not in the high impedance state, and in green and text highZ if they are in the high impedance state. The colour of a line is set by calling the RGB function with three parameters for the red, green and blue component of the resulting colour. It should be noted that the form update procedure of the currently active screen form is also called during the simulation run.

The replacement of screens is realised by using the above explained screen forms and form update procedures. When a request to replace the screen is registered the corresponding new screen form is opened and the appropriate form update procedure called. This causes updating of the presentation of signal values in the Block diagram window. Then, the identifier of the opened and currently active screen form is saved as a global variable for later use during the simulation run. Finally, the previously active form is closed, which completes the screen replacement.

Simulation

This equipment makes it possible to simulate the behaviour of a computer system over a period of time corresponding to one clock period, one instruction or a complete program, activated by the Simulation command buttons Clk+, Ins+ and Prg+ in the Command window. Procedures are included to simulate the behaviour of sequential and combinatorial circuits, present values of signals in the Block diagram window, store values of signals in the signals statistics file, and simulate the clocks of computer system modules.

The simulation of sequential and combinatorial circuits follows the behaviour of real digital systems. In such systems outputs of sequential circuits are led to inputs of combinatorial circuits and outputs of combinatorial circuits to inputs of sequential circuits. Sequential circuits change values at their outputs only when a clock pulse occurs, and combinatorial circuits only when any of their input signals change. In digital systems, sequential circuits are used to maintain appropriate values during a clock period and combinatorial circuits to transform their input signals and produce new output values. The simulation of sequential and combinatorial circuits is performed as follows. When a system reset occurs the values of signals at the outputs of sequential circuits are first initialised and then the values of signals at the outputs of combinatorial circuits calculated. When a clock pulse occurs, the signal values at the outputs of sequential circuits are first calculated, depending on the values of signals at the outputs of appropriate combinatorial circuits, and then the values of signals at the outputs of combinatorial circuits are calculated, depending on the values of signals at the outputs of appropriate sequential circuits. Since each sequential circuit changes its value depending on the values of signals at the outputs of the appropriate combinatorial circuit, the order in which sequential circuits are calculated is not important. However, combinatorial circuits are usually not realised in such a way that each sequential circuit has its own independent combinatorial circuit. In order to optimise the design, individual combinatorial circuits are parts of a few combinatorial circuits that give input signals for sequential circuits. Therefore, the order in which combinatorial circuits are calculated is important and depends on the topology of each combinatorial circuit.

The presentation of values of signals in the Block diagram window is carried out by the same procedures as those used in the hierarchical screens scheme.

Storing signal values in the signals statistics file is part of the miscellaneous feature of drawing signal timing diagrams. A record in the signals statistics file is created every computer system module clock period. It contains fields with the clock values of all computer system modules, signal values on the address and data lines of the system bus, values of 18 computer system signals, the value of either the step counter when hardwired control is used, or the microprogram counter when the microprogrammable control unit is used, and the current value of the time units counter.

Simulation of a computer system module clock is carried out by a procedure activated by one of the Simulation command buttons Clk+, Ins+ and Prg+ in the Command window. In this procedure one time unit is simulated. As a result the time units counter is incremented, computer system module clock counters are decremented and a check performed to determine whether any of them has reached value zero. If this is not the case the next time unit is simulated. This continues until at least one computer system module clock counter reaches the value zero. Then the actions of the appropriate computer system module clocks are simulated. As a result, for each of these modules, simulation of sequential and combinatorial circuits is carried out and appropriate variables updated. Then, the identifier of the currently active screen form is used to call the appropriate form update procedure and update the presentation of values of signals in the Block diagram window. Following this, the procedure to store the values of signals in the signals statistics file is called. Finally, the computer system module clock counters that have reached zero are initialised to their clock period values. Further activities depend on the command with which the procedure for simulation of computer system module clocks was called. If the Clk+ command was called the procedure is left. If Ins+ command was called the previously described activities are first repeated as many times as needed to simulate the number of processor clocks required for complete execution of an instruction and then the procedure is left. If Prg+ command was called, the previous steps are repeated as many times as is needed to simulate the number of processor clocks required for the execution of the instructions of a complete program, and then the procedure is left.

The computer system simulator is realised in such a way that simulation of the most complete possible configuration of the computer system is provided. During simulation set up for each configurable part of the computer system a flag is appropriately set indicating whether this part is included in the current configuration of the computer system or not. Therefore, the simulator checks each flag and executes the corresponding code only if the flag indicates that a particular part of the computer system is included in the current configuration.

Miscellaneous features

The miscellaneous features create a user friendly environment to work with the simulator. They are activated by Misc command buttons More, Clear, Exit and Help in the Command window. Their realisation includes procedures to view simulation

results, return the simulation to the beginning, save the current simulation state in a file and quit the simulation, and get information about the educational computer system and software package.

Viewing of simulation results is made up of two groups of procedures. Procedures in the first group are used to examine the values in memory locations and registers of the processor and peripheral devices and their controllers. These are the same procedures as those used for the interactive simulation set up. The procedure in the second group is used to draw signal timing diagrams using values from the signals statistics file. Timing diagrams are drawn using the Picture object in increments separately for each time unit. One record of the signals statistics file contains data for drawing a segment of the timing diagram corresponding to one time unit. The fields of a record are used for drawing signal timing diagrams and denoting on the time axis the values of either the step counter or the microprogram counter and the time elapsed, given in time units. In order to keep the screen presentation of signals in a readable form the following decisions have been made. Signals are drawn for up to 72 time units. The ratio of computer system module clocks periods given in time units can be only in the range 1 to 24. The signals drawn are scaled on the time axis in the range 1 to 3, giving 72, 48 or 24 time units on a screen. As a result the slowest clock of a computer system module can appear between 3 and 24 times on a screen.

Return of the simulation to the beginning first resets the time unit counter, initialises the computer system module clock counters and deletes the current and creates a new signals statistics file. Then, the current variables of sequential circuits are initialised with the values of initial variables of sequential circuits and procedures for simulation of combinatorial circuits executed. Finally, the identifier of the currently active form is used to call the appropriate form update procedure and update the presentation of values of signals in the Block diagram window.

Saving of current simulation state in a file is performed optionally before quitting the simulation. As the result a context file of the complete computer system containing parameter variables and current and initial sequential circuit variables of the simulator is created. The file can be used later to carry out complete simulation set up from file to resume the simulation.

Information about the educational computer system and the software package is provided by the Help system, which is realised in a standard way. A description of the educational computer system and software package features is first presented in the form of HTML pages. Then the HTML Help Workshop 1.2 tool is used to link the HTML pages to create the structure of the Help system. Finally, the complete contents are indexed.

Conclusions

This paper describes some aspects concerning the realisation of a software package for an educational computer system. The software package described is the latest version of a few years development effort. Previous versions have undergone a number of improvements based on reactions given by students who have worked

with the software package in the computer architecture and organisation laboratory. As the result the current version meets the initial objectives.

The software package has been developed with the Visual Basic programming language under the Windows operating system. The Visual Basic programming language was a satisfactory tool in solving problems such as graphical presentation of parts of the computer system, simulation of sequential and combinational circuits, presentation of signal values, etc. Even though a large number of forms have been used (around 200) with many objects on forms (around 15 000) and 20 modules with around 100 000 lines of code, maximum capacities have not been reached. The Windows operating system made it possible to run the software package on a large number of installations.

Further work on the software package will be focused in two directions. The first is to add new features that would make it possible to turn the simulation a few clock cycles back, create a simulation log file, maintain statistics about the frequency of appearances of instructions, addressing modes, etc., specify simulation break points, etc. The second is the development of a completely new software package using (most likely) programming language *Java* with the aim of facilitating distance learning via the Internet.

References

- 1 W. Stallings, *Computer Organization and Architecture* (Prentice Hall, New Jersey, 1996).
- 2 A. Clements, 'Computer architecture education', *IEEE Micro*, **20**(3) (2000), 10–12.
- 3 R. Hodson and J. Hereford, 'Interactive CPU simulator for computer organization instruction', IEEE Computer Society Technical Committee on Computer Architecture Newsletter, September 1997, pp. 16–24.
- 4 P. Verplaetse and J. V. Campenhout, 'ESCAPE: Environment for the simulation of computer architecture for the purpose of education', The Workshop on Computer Architecture Education (WCAE-98), 27 June, 1998, Barcelona, Spain.
- 5 Y. Zhang and G. B. Adams III, 'An interactive, visual simulator for the DLX pipeline', IEEE Computer Society, Technical Committee on Computer Architecture Newsletter, September 1997, pp. 9–12.
- 6 R. N. Ibbet, 'HASE DLX simulation model', *IEEE Micro*, Special Issue on Computer Architecture Education, **20**(3) (2000), 38–47.
- 7 J. Djordjevic, A. Milenkovic and N. Grbanovic, 'An integrated environment for teaching computer architecture', *IEEE Micro*, **20**(3) (2000), 66–74.
- 8 Language Reference, Microsoft Visual Basic, <http://msdn.microsoft.com/vbasic>