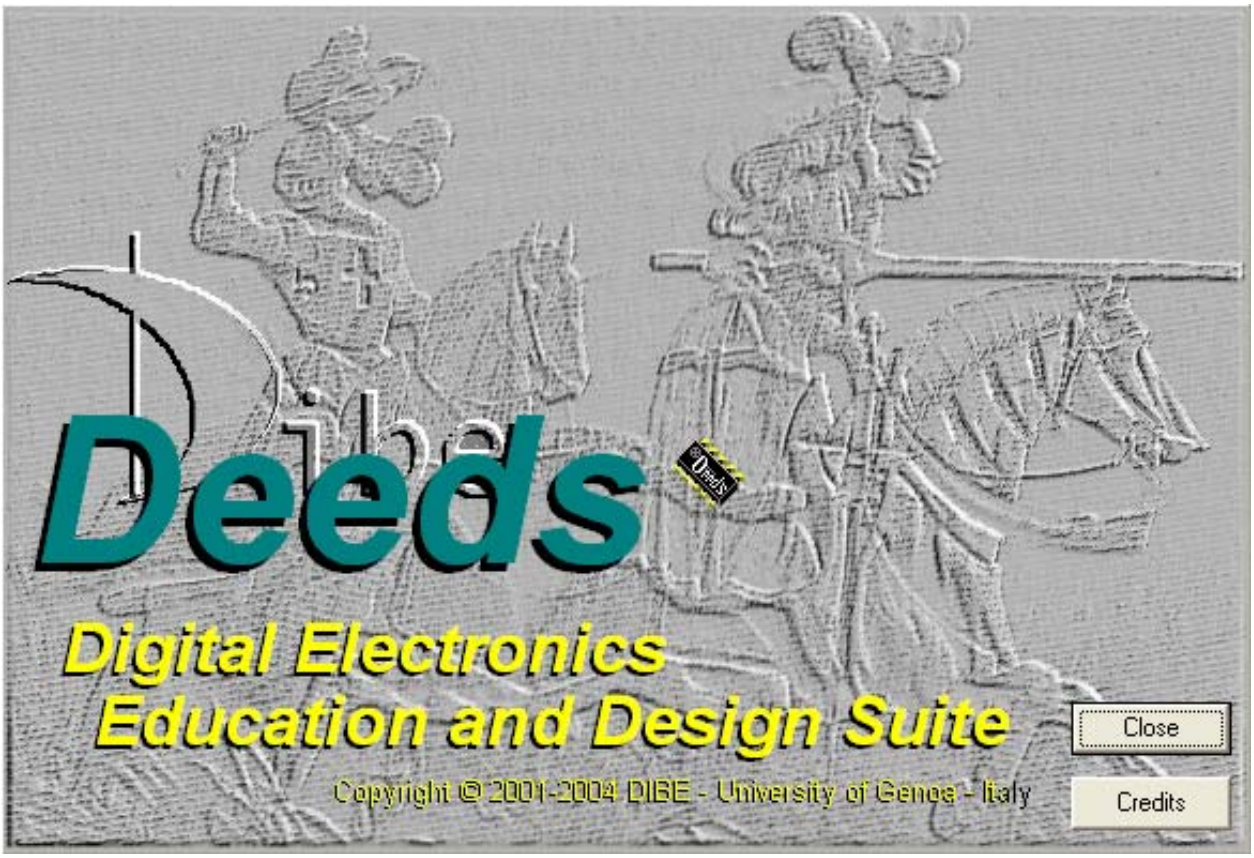
University of Genoa
Biophysical and Electronic Engineering Department
Electronic Systems and Networking Group

Deeds

Digital Electronics Education and Design Suite



User Manual
(Feb 2004)

*Edited by **Giuliano Donzellini** and **Domenico Ponta***

Exegi monumentum aere perennius
regalique situ pyramidum altius
quod non imber edax, non Aquilo impotens
possit diruere, aut innumerabilis
annorum series et fuga temporum.

Quinto Orazio Flacco

Deeds - Digital Electronics Education and Design Suite

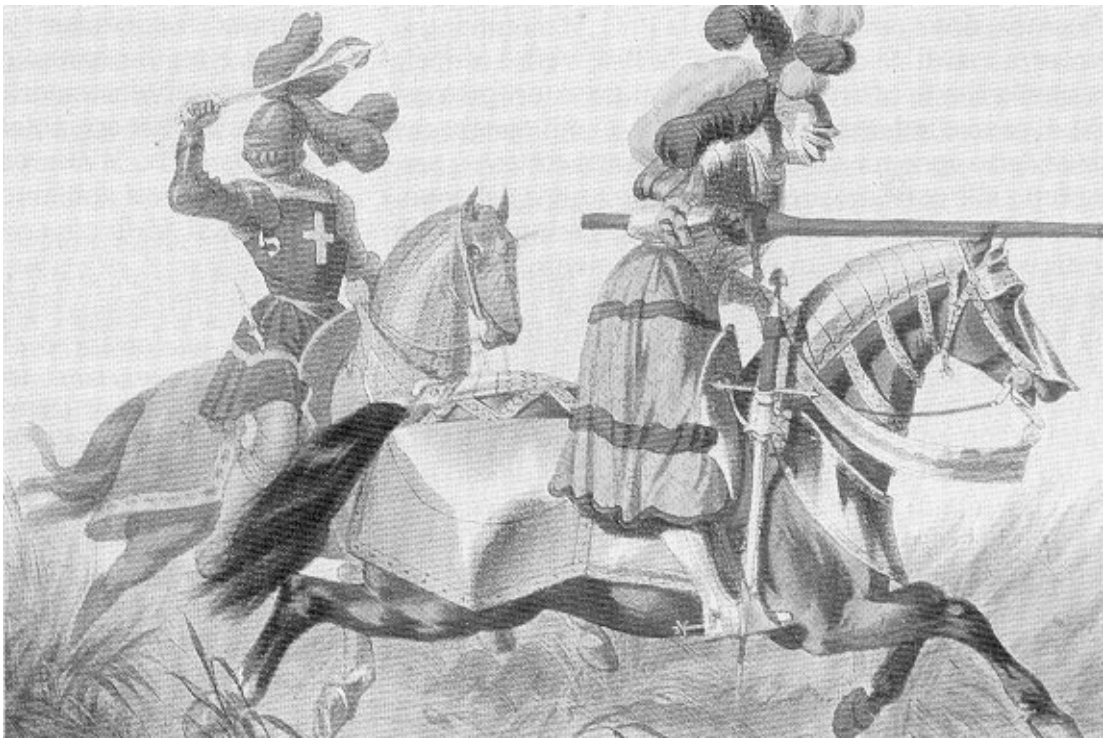
User Manual

Index

Index of Figures

Preface

Deeds is the acronym of ***Digital Electronics Education and Design Suite…*** but, as *"deeds"* mean, I'm not sure if they will be good or bad… just like *The Deeds of Gallant Knights* that the splash form recalls…



"The Deeds of Gallant Knights"
…from a picture of G. David, XVI Century - Paris, Musèe de l'Armèe

Deeds

Digital Electronics Education and Design Suite

Introduction

Deeds is conceived as a suite of simulators, tools and learning material for Digital Electronics. Deeds helps student acquiring theoretical foundations, analysis capabilities, ability to solve problems all over the subject topics, practical synthesis and design skills. Its approach is characterised by the "learning-by-doing" concept.

It covers the following areas of digital electronics:

- Combinational logic networks (from simple gates to decoders, encoders, multiplexers and demultiplexers);
- Sequential logic networks (from simple flip-flops to registers and counters);
- Finite state machine design;
- Micro-computer programming (at assembly level) and interfacing;

Major tools that Deeds includes are:

- An HTML main browser, to navigate in Internet, where students will find lessons, exercises and laboratory assignments;
- An HTML assistant browser, that assists students in their work;
- A schematic digital circuit editor (with component data-sheet support);
- An interactive circuit 'animator' (to experiment with components and simple networks directly on the schematics);
- An interactive logic simulator (with a timing diagram tracer to analyse events in the logic networks, and to interact step-by-step with the circuit);
- A finite state machine editor / simulator (the algorithm is described using an Algorithmic State Machine graphical editor);
- A microcomputer board emulator (the board include an 8 bit CPU, ROM, RAM, I/O ports);
- An assembler level / interactive debugger module.

Deeds tools can interact with each other:

- The HTML main and assistant browsers allows to launch all the other tools and interact with them;
- The browser can control editors and simulators, to realise a true interaction between text and experiments;
- The schematic editor allows to connect traditional logic circuits with subsystems defined by the user with the help of the finite state machine editors and the micro-computer emulator.
- It is possible to experiment with digital systems controlled by state machines.

The architecture of Deeds allows a "scalable" approach to the lessons, exercises and laboratory sessions. All the tools included allow either a simplified scenario to beginners and a more exhaustive and complete environment for skilled students.

Deeds as a learning environment for digital electronics

Deeds is conceived as a learning environment for digital electronics. With such term we mean a collection of tools and text material that help students acquiring:

- theoretical foundations of the subject;
- analysis capabilities;
- ability to solve problems all over the subject topics;
- practical synthesis and design skills.

Deeds is conceived as a common resource for all introductive courses in digital electronics. As such, it may contain different technical subjects, different pedagogical formats (lectures, exercises, lab assignments, etc.) delivered at different student levels. Deeds is therefore born as a set of tools (listed before) that teachers can complete and personalise to suit their pedagogical needs by contributing to the ”lecture space” with their own materials.
There is no need for a specific authoring tool, because the lecture space can be composed with any HTML editor, completed by a helper application that facilitates the linking of the editors and simulators’ commands to the lecture text.

How to use *Deeds* to teach theory

A “lecture” based on Deeds appears as HTML pages with text and figures. The page aspect and layout are totally up to the author. At this level, students see a “normal” on-line book or document. But many of the figures and visual objects are “active”, because they are connected to the editing and simulation tools of Deeds.
For example, let’s suppose that theory presents a certain digital circuit , visualising its schematics in a picture. When the user clicks on the picture, Deeds launches the corresponding simulator, and opens that schematic, together with another windows (the Helper) that contains step-by-step instructions on how to explore or test the circuit itself. Such procedure is equally useful to convey concepts on simple components or quite complex networks. In the first case, simulators allows to “animate” circuits, i.e. to explore them interactively. In the second one, their capabilities of tracing signals in the time and data domain allows a thorough test of the network.

How to use *Deeds* to solve exercises

The target of traditional exercises is to help understanding theory, applying it to simple cases and providing a feedback to the teacher through the delivery of the solutions. In our system exercises are presented as HTML pages, containing text and figures of the assignments. The role of Deeds is to allow students to check the correctness of the solutions obtained manually and to provide graphical tools for editing the web page containing their reports, until they are satisfied with their work and use Deeds to deliver the reports through the network.

The use of Deeds implies also a different approach to the structure of the exercises. In fact, with the simulator, students may be tempted to skip manual analysis. Exercises, therefore, must be targeted more to the real understanding of the issues than to the execution of repetitive tasks.

How to use *Deeds* to learn to design electronic systems

The development of a digital design project is the field where Deeds can fully be exploited. In fact, the interactive logic simulator, the finite state machine module and the microcomputer board emulator can work simultaneously in the simulation of a system where standard digital components can be controlled by a state machines as it is the case in contemporary digital design. Obviously, the modules can be used independently, to test separately the system's parts. The student can complete its work programming at assembly level a microcomputer board.
Students use Deeds to download the assignment from a web page. The assignment consists of a functional description and a set of specification of the system that students must design. The approach is meant to replicate the features of a professional environment, within the guidelines suggested by the educational purposes. Project development phases are guided by help and instructions supplied through the Assistant Browser. Such instructions can be given step-by-step or by simple guidelines: the use of the simulation tools can be more or less guided by the text of the assignment (to left creativity and fantasy to the user initiative). In Fig. 1, an example of laboratory student report, displayed in the main browser.

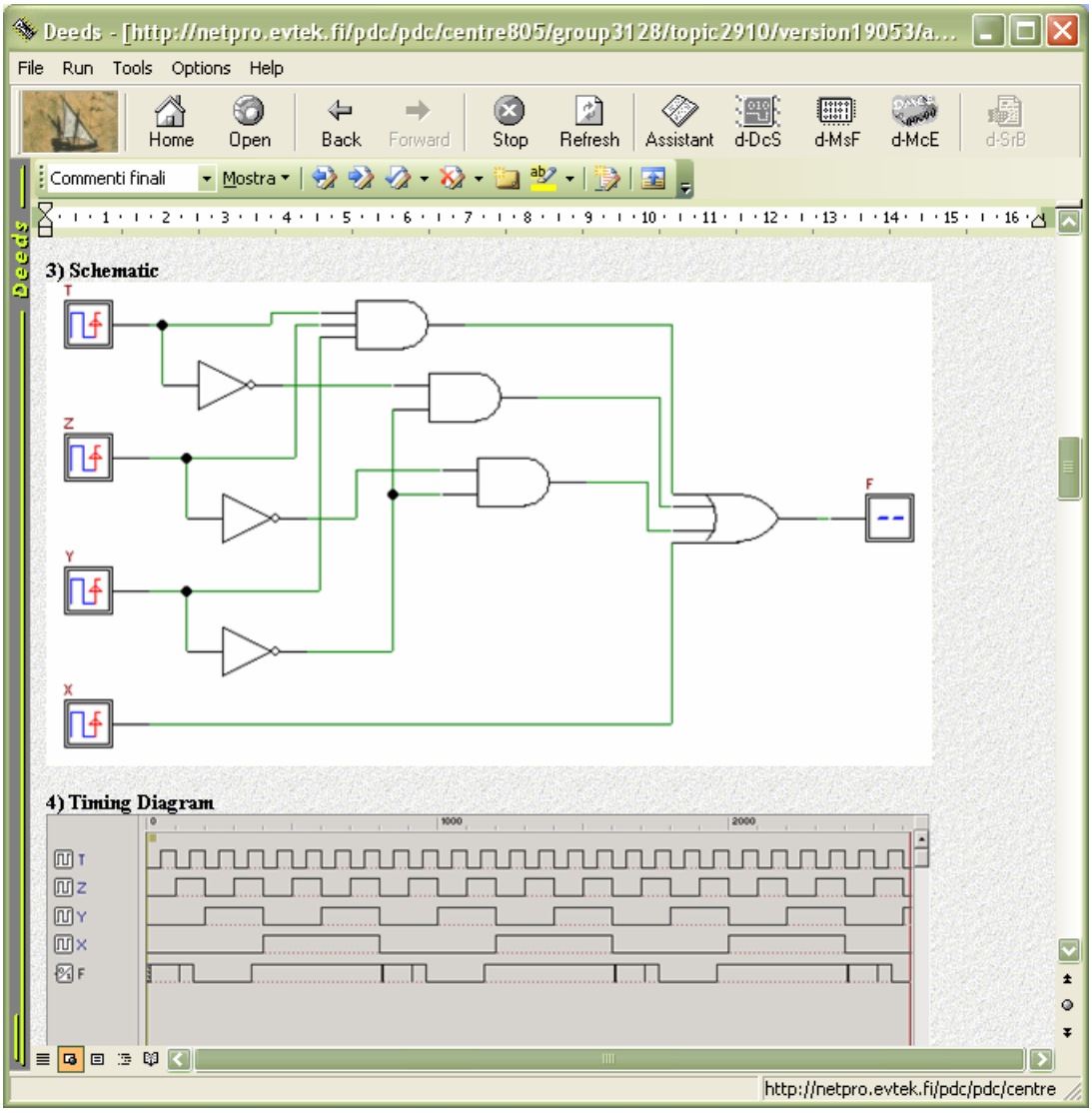


Fig. 1: An example of laboratory report displayed in the main browser of Deeds

The *Deeds* simulation tools

The simulation tools are three: a Digital Circuit Simulator (d-DcS), a Finite State Machine Simulator (d-FsM), and a Micro Computer Board Emulator (d-McE). All the simulation tools are characterized by a “learn-by-doing” approach. They are integrated together: design and simulation of complex networks integrating standard logic with state machines are possible. In Fig. 2 a few screen shots of the Deeds tools are shown.

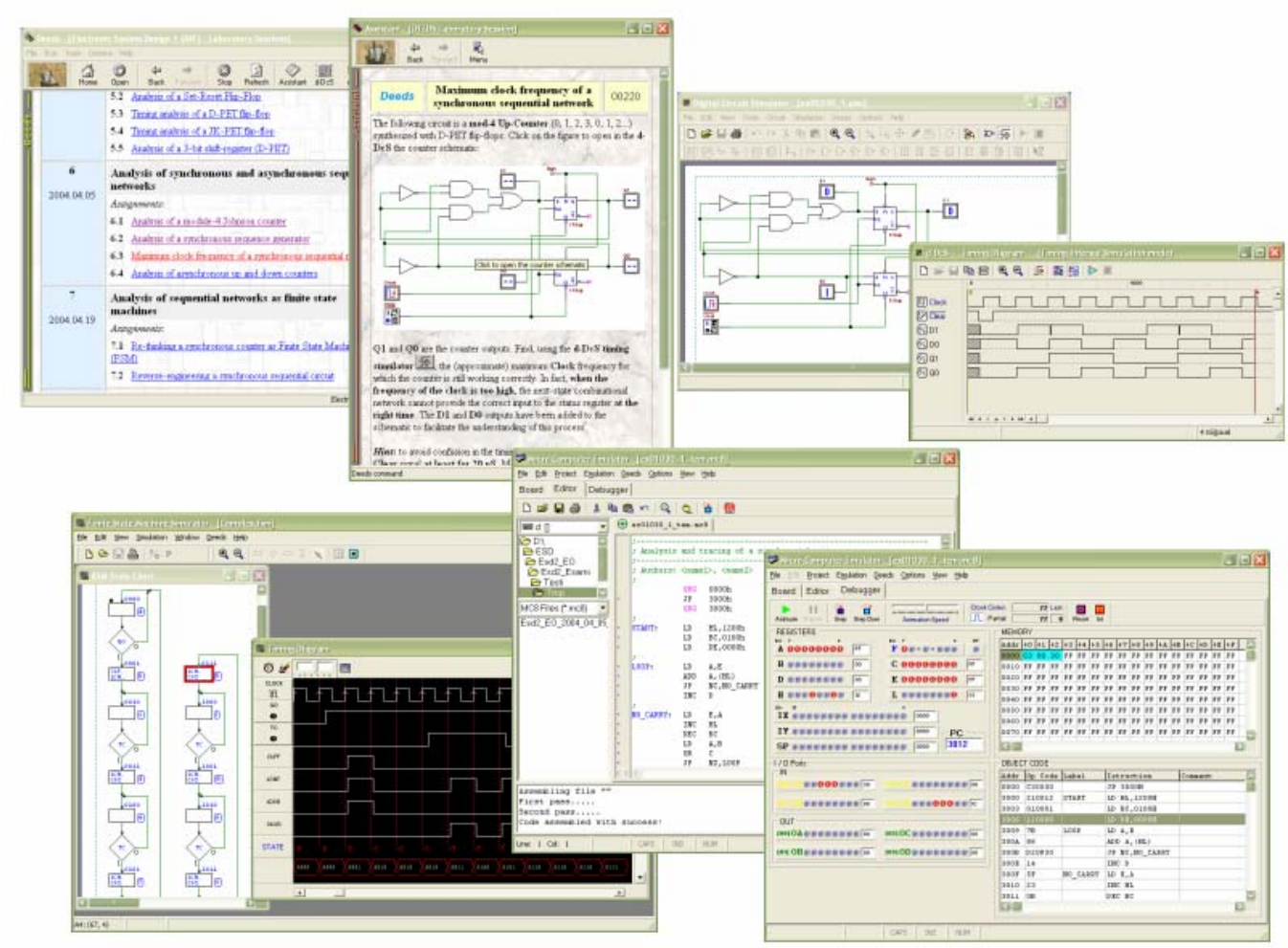


Fig. 2: The Deeds environment: the main and assistant browsers (on top left), and the three Simulation Tools: the Digital Circuit Simulator (on top right), the Finite State Machine Simulator (on bottom left) and the Micro Computer Emulator (on bottom right).

Deeds: The Main Browser

The simulators are integrated around two HTML browsers, enabling active Internet navigation to sites where students find pages with lessons, exercises and laboratory assignments. The main web browser of Deeds, when activated, shows a HTML page that allows to connect to the Deeds web site and to the 'on-line' learning materials developed at DIBE (University of Genoa).

The main browser (Fig. 3) has been developed around the standard Microsoft WebBrowser ® component, the same used by the Microsoft Internet Explorer ®, extended to support all the required functions by the Deeds environment. It is mainly used to connect to the sites containing the learning materials. The browser supports all the features that the user can expect to find, including JAVA Virtual Machine ®, JavaScript ®, VBScript ®, XML support.

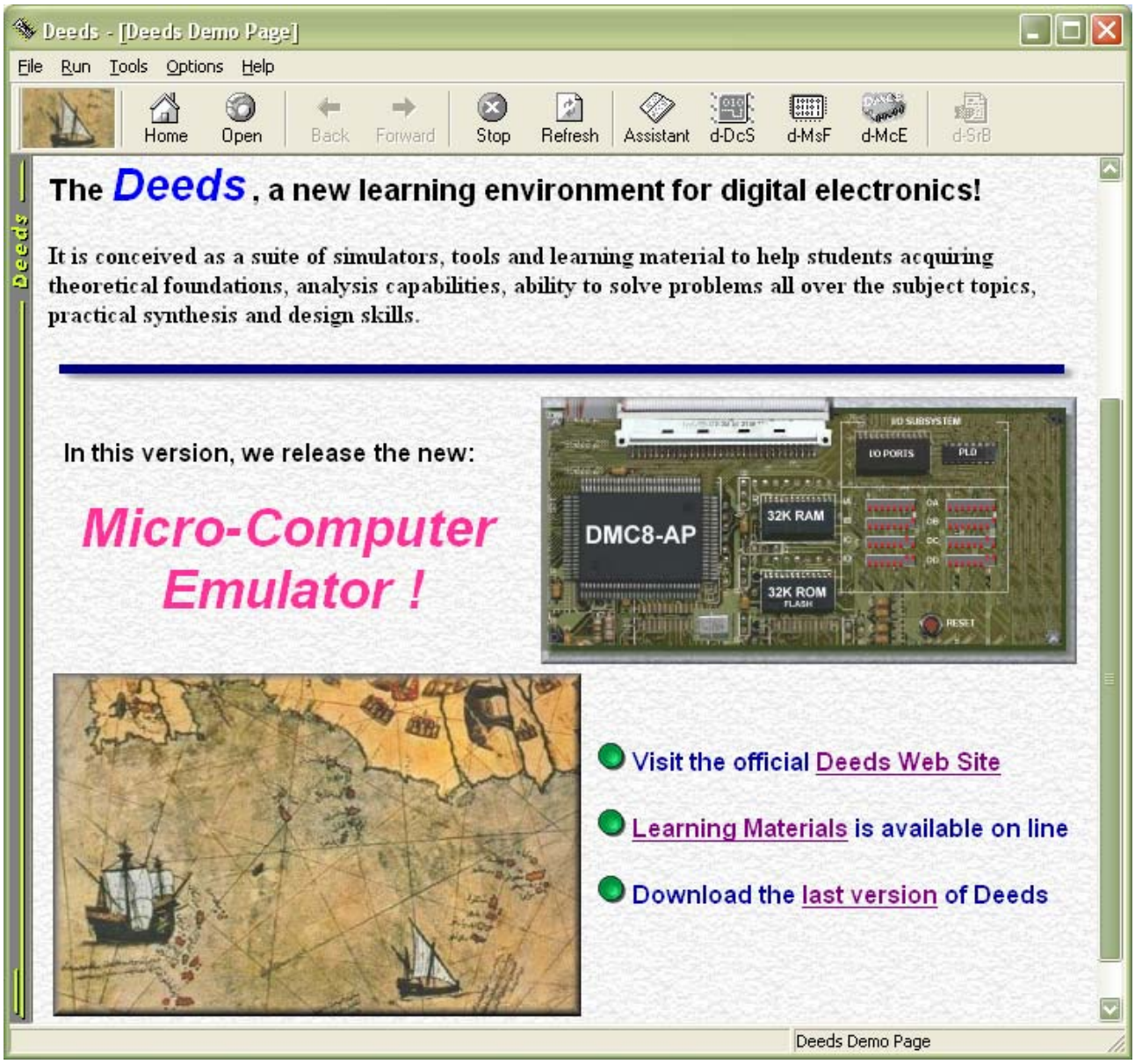


Fig. 3: The main browser of Deeds, showing the HTML page that allows to connect to the Deeds web site and to the 'on-line' learning materials.

When the user launches the Deeds environment, the main browser shows up. All the other tools can be activated by the menu and/or toolbar command. The main browser acts as 'main window' of the application suite.

With Deeds, the user can directly navigate to the own web site, where Learning Material are available. In Fig. 4 you see the 'screen shots' web page of the site.

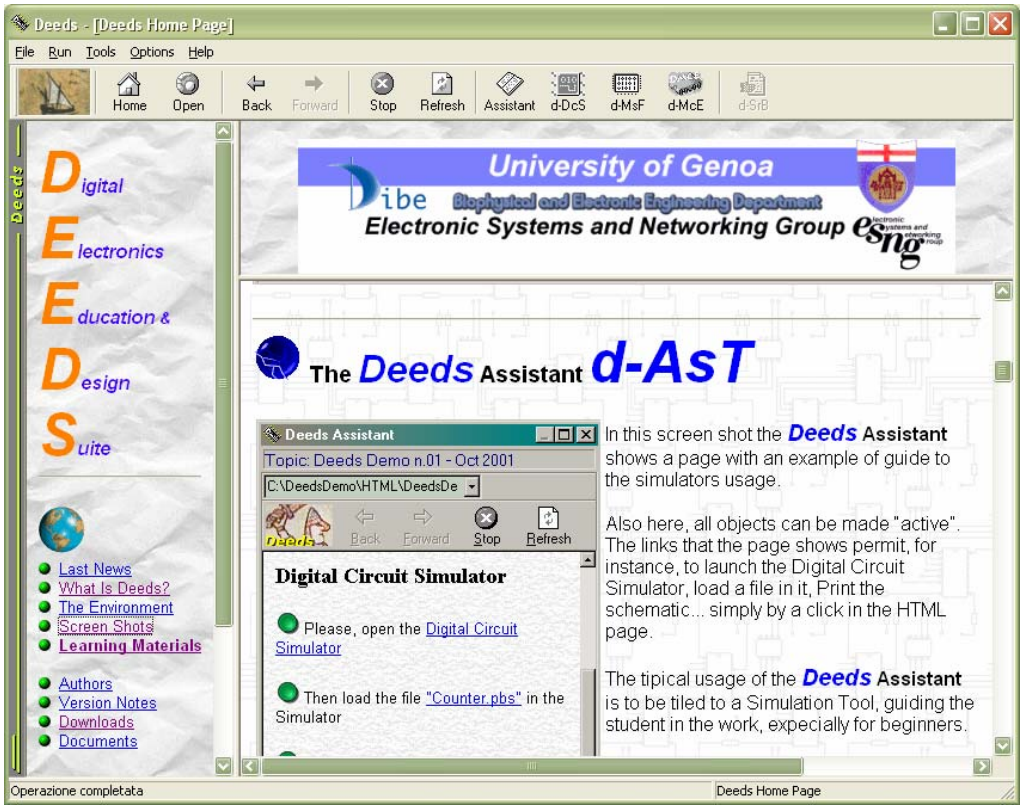


Fig. 4: The main browser, connected to the 'Sreen Shots' page of the Deeds web site.

The user can also download the last version of the Deeds suite, as soon as it become available (Fig. 5):

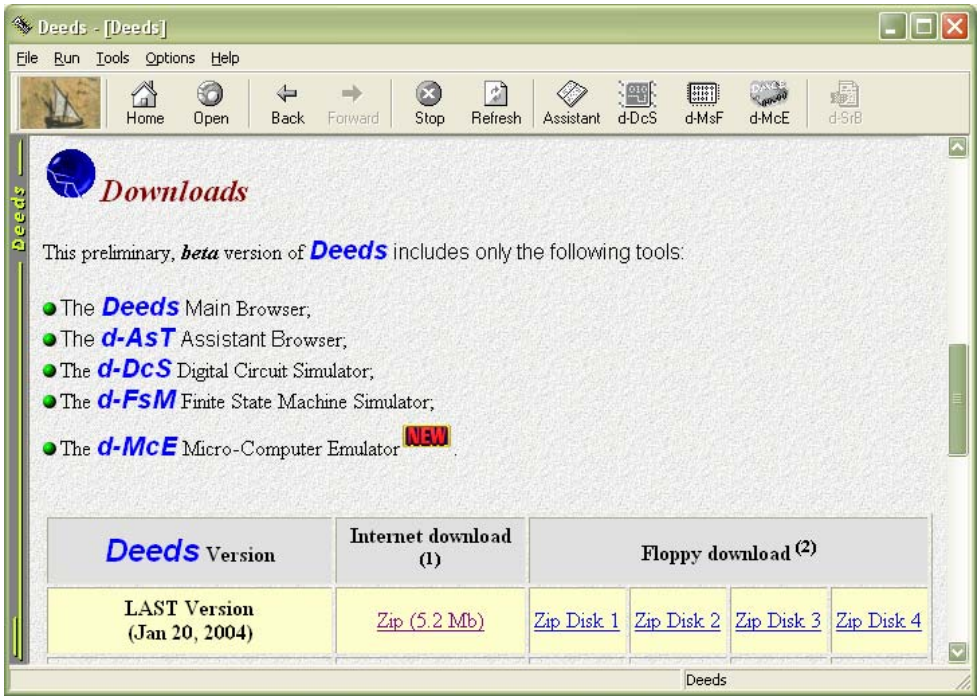


Fig. 5: The download page in the Deeds web site.

Deeds has been developed as common simulation tool to be shared among different institutions running courses on Digital Design, as a support of the activities of the NetPro project in the field of Electronic Engineering. NetPro, a European project of the Leonardo DaVinci program, develops project-based learning through Internet. It has created models, tools and services to facilitate communication and collaboration between distant students, and to manage access and control of project deliverables.
We test NetPro methodologies and tools by running pilot projects. An important characteristic of the pilot courses is that project groups can be distributed over different academic institutions and countries. A pilot course may have teams from more than one institution and more than one nation while teams themselves could be inter-institutional and international.
The immediate goal of the collaboration between pilot sites is to provide learning tasks that are meaningful for all students, independently of their local arrangements.
Joint working is possible if teams use the same language (all the components of our pilots, including student deliverables and communication, are in English) and if the classes involved study the same topic at the same time of year. All documents produced are available as web sites for on-line fruition or as downloadable files. In fig. 6 you see, opened in the main browser, the learning material index page, available in the Deeds web site.

Fig. 6: the learning material page (available in the Deeds web site), opened in the main browser.

Deeds: Main browser Menu

The main browser menu allows to navigate web site, to run simulators and tools, to switch between the opened tools, and to customize the user options.

File Menu

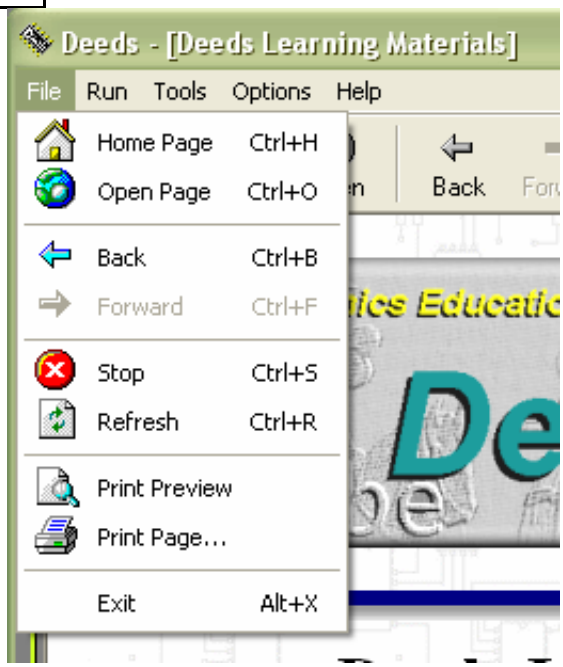


Fig. 7: The main browser "File" menu.

Home Page

Command to navigate to the main browser home page (it can be user-defined)..

Open Page

Open the Open Page dialog (Fig. 8). In this dialog window, the user can type directly a URL address, or browse the local network or disk. The chosen web page can be set as Home Page. A short history of previously opened pages is maintained.

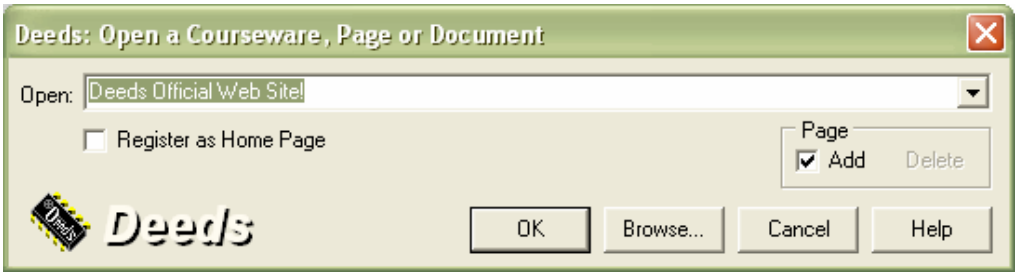


Fig. 8: The Open Page dialog window.

Back

Standard browsing command to return to the ‘previous’ opened page.

Forward

Standard browsing command to return to the ‘next’ opened page, after using the ‘Back’ command.

Stop

Standard browsing command to stop the download of the current page.

Refresh

Standard browsing command to reload the currently opened page.

Print Preview

Standard command to preview the current page before printing.

Print Page

Standard command to print the current page.

Exit

Standard command to close the Assistant.

Run Menu

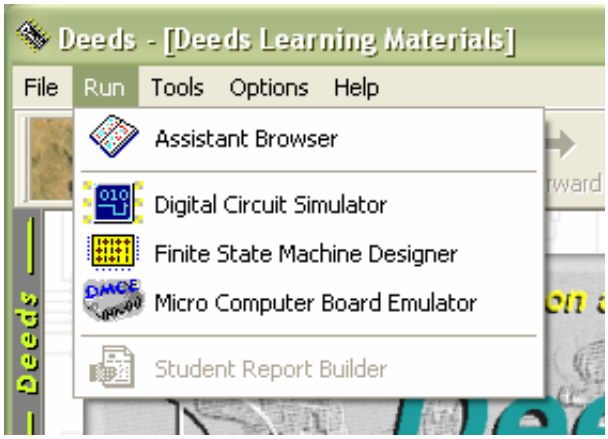


Fig. 9: The main browser "Run" menu.

Assistant Browser

Command to open manually an instance of the Assistant Browser.

Digital Circuit Simulator

Command to open manually an instance of the Digital Circuit Simulator (d-DcS).

Finite State Machine Designer

Command to open manually an instance of the Finite State Machine Designer/Simulator (d-FsM).

Micro Computer Board Emulator

Command to open manually an instance of the Micro Computer Board Emulator (d-McE).

Tools Menu

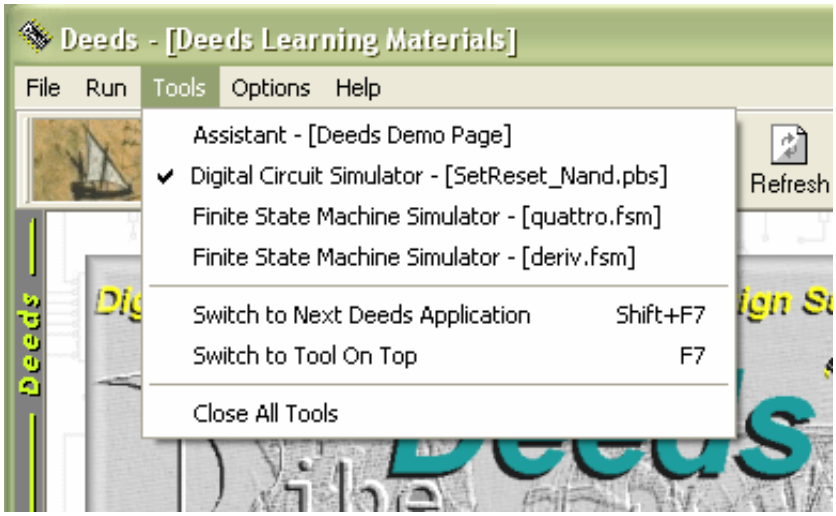


Fig. 10: The main browser "Tools" menu.

First items group

Commands to switch focus to the chosen tool. All the opened tools are indexed here, together with the name of the corresponding opened file, if any. When the user click on an item, the tool will go 'on top'.

Switch to Next Deeds Application

Command to switch to the next Deeds open tool or browser.

Switch to Tool on Top

Command to switch to the tool that was 'on top' before switching to the main browser.

Close All Tool

Command to close all the opened tool. If a file, opened in a tool, is not saved, the user will be prompted, and the close operation stopped.

Options Menu

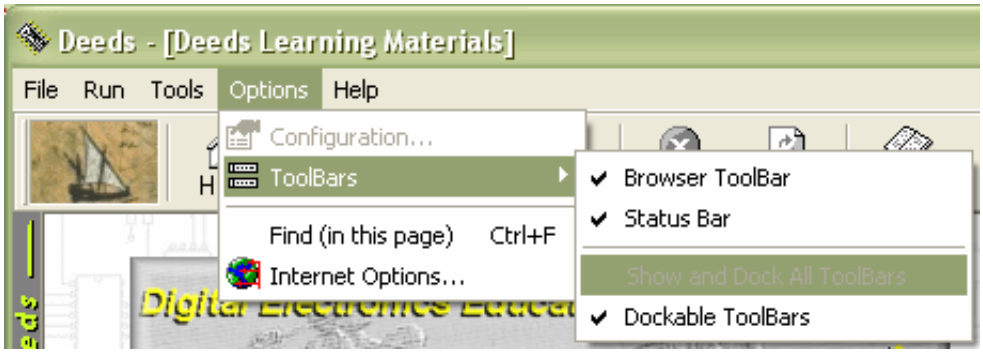


Fig. 11: The main browser "Options" menu.

Configuration

Command to change the application configuration (disabled in this version).

ToolBars

Commands to control ToolBars appearance.

Browser ToolBar

Command to hide or show the Browser ToolBar.

Status Bar

Command to hide or show the Status Bar.

Show and Dock All ToolBars

Command to show and dock in all the ToolBars.

Dockable ToolBars

Command to enable or disable the docking modality of the ToolBars.

Help Menu

Fig. 12: The main browser "Help" menu.

Index

Command to open the Deeds Help System.

License Agreement

Command to display the Licence Agreement.

Version Notes

Command to display the Version Notes file.

About

Command to display the Deeds 'splash' window dialog.

Deeds: The Assistant Browser (d-AsT)

The “Assistant” HTML browser has characteristics similar to those of the main browser, but it is specialized to assist students, side by side, in their work (fig. 13). This is the browser used to open lessons, exercises and laboratory assignments. As the main, also the Assistant browser has been conceived around the standard Microsoft WebBrowser ® component.
In Fig. 13 the Assistant browser is opened aside of the main one, showing a page with a problem assignment (from the ESD1 NetPro course). To open an assignment, the user will click on the desired topic, listed in the main browser: the Assistant will open automatically, showing itself aside.

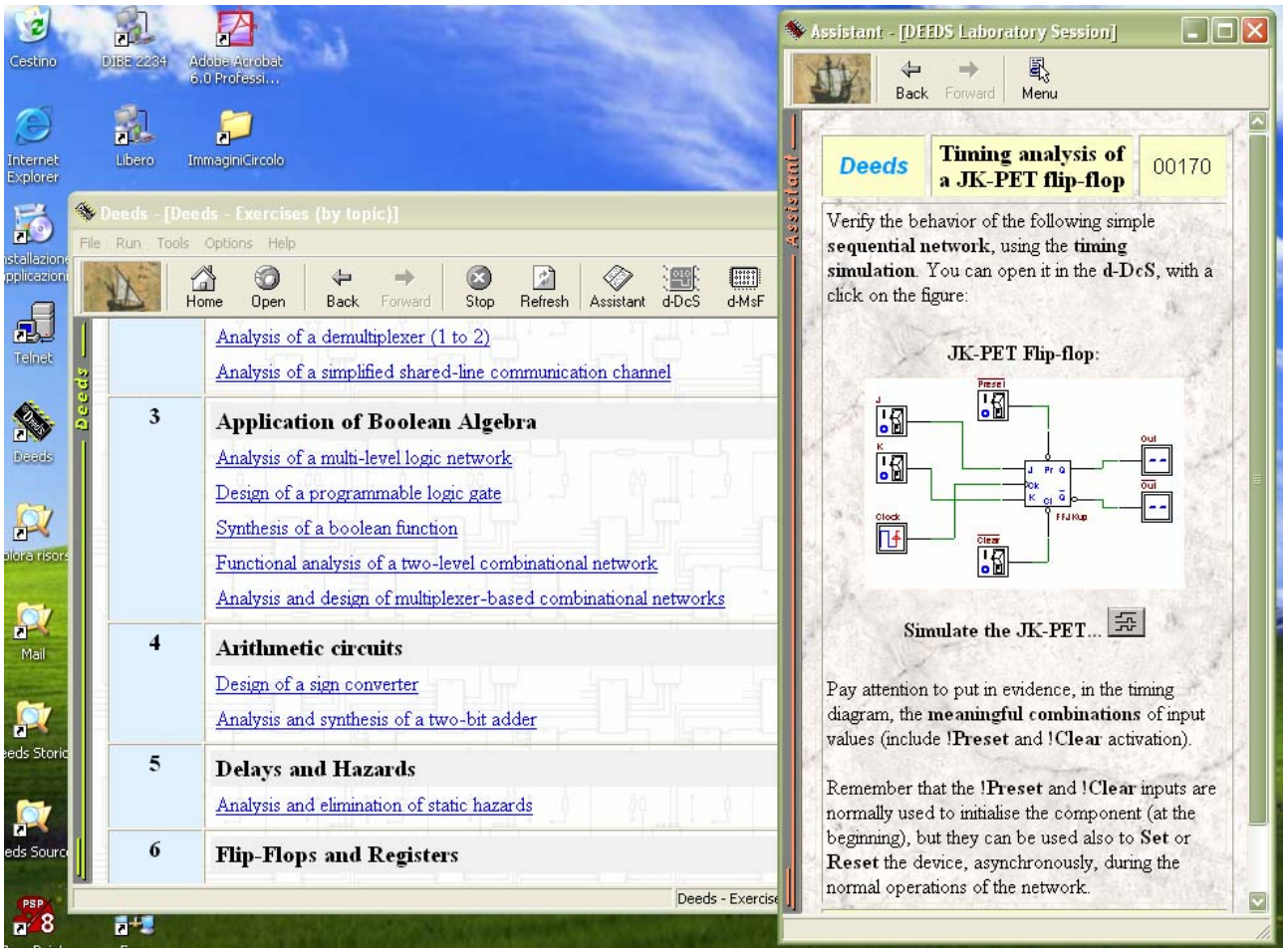


Fig. 13: The Assistant opened aside of the main browser, showing a page with a problem assignment.

All objects, that a web page visualises, can be made “active”. For instance, by clicking on the figure showing the schematics, the Digital Circuit Simulator could be started and the circuit loaded, ready to be tested (this important feature will be described in detail later).

Deeds: Assistant browser Menu

The Assistant menu has been reduced to the essential (Fig. 14), to simplify user operation. Its graphical shape has been chosen to minimize the window size, allowing the positioning of the Assistant aside of the simulation tool without occupy to much area of the screen.

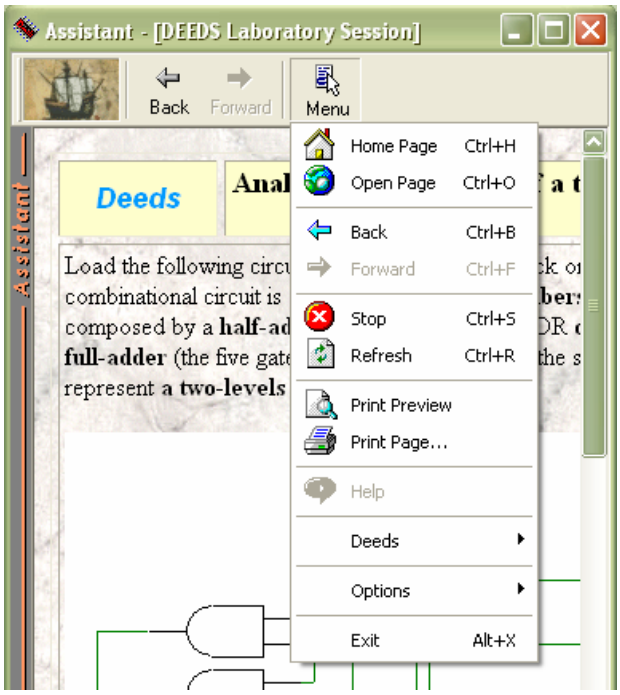


Fig. 14: The Assistant main menu, appended to the toolbar.

Home Page

Command to navigate to the Assistant local home page.

Open Page

Open the Open Page dialog (Fig. 15). In this dialog window, the user can type directly a URL address, or browse the local network or disk. The chosen web page can be set as Home Page. A short history of previously opened pages is maintained.

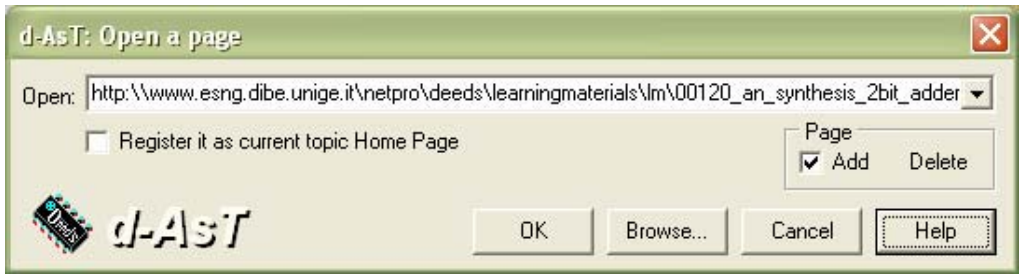


Fig. 15: The Open Page dialog window.

Back

Standard browsing command to return to the ‘previous’ opened page.

Forward

Standard browsing command to return to the ‘next’ opened page, after using the ‘Back’ command.

Stop

Standard browsing command to stop the download of the current page.

Refresh

Standard browsing command to reload the currently opened page.

Print Preview

Standard command to preview the current page before printing.

Print Page

Standard command to print the current page.

Deeds

Command group to navigate between the opened Deeds tools.

Options

Command group to change the Assistant configuration and options.

Exit

Standard command to close the Assistant.

Deeds: The Digital Circuit Simulator d-DcS

This image from the Tapestry of Bayeux, Bayeux Cathedral, France

Introduction

The Digital Circuit Simulator ***d-DcS*** appears to the user as a graphical schematic editor, with a library of simplified logic components, specialised toward pedagogical needs and not describing specific commercial products (Fig. 16).

As described before, the schematic editor allows to build simple digital networks composed of gates, flip-flops, pre-defined combinational and sequential circuits and custom-defined components (defined as Finite state machine).

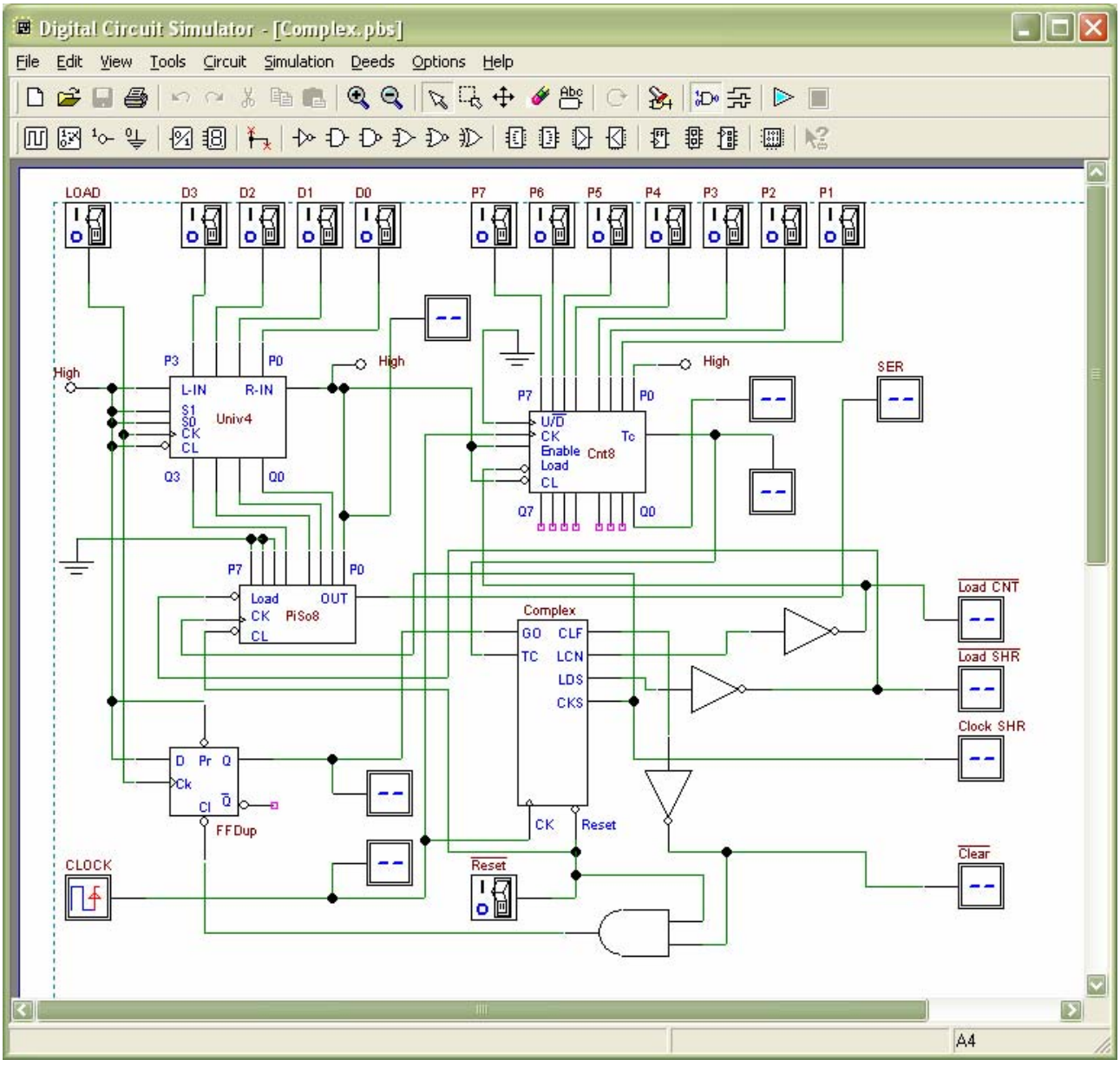


Fig. 16: The circuit editor of the Digital Circuit Simulator (d-DcS).

Simulation can be interactive or in timing-mode. In the first mode, the student can *"animate"* the digital system in the editor, controlling its inputs and observing the results. This is the simplest mode to examine a digital network, and this way of operation can be useful for the beginners.
In the timing mode, the behaviour of the circuit can be analysed by a timing diagram window, in which the user can define graphically an input signal sequence and observe the simulation results. This is the mode nearest to the professional simulators.

A simple example

In following screen shots (Fig. 17a,b,c), you can see the circuit during the drawing and then simulated by *animation*:

a) the student picks-up components from the bin on the Component Tool Bar, then
b) connects them using Wires. When finished,
c) the student activates the animation.

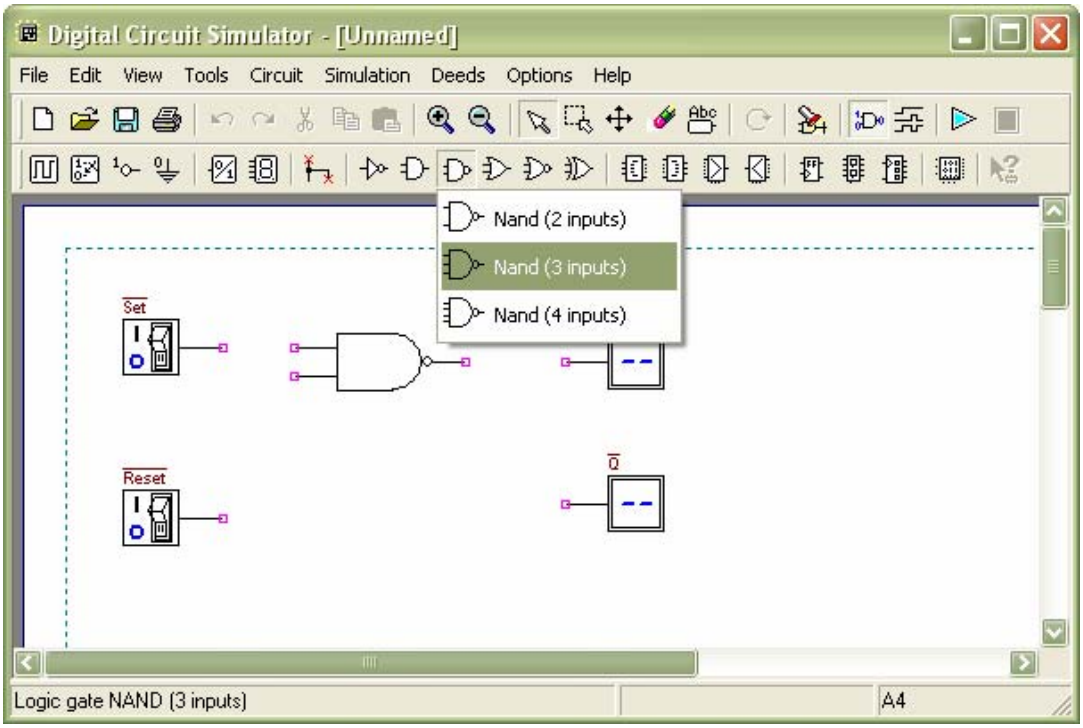


Fig. 17a: The drawing phase of the digital circuit editor: the insertion of components.

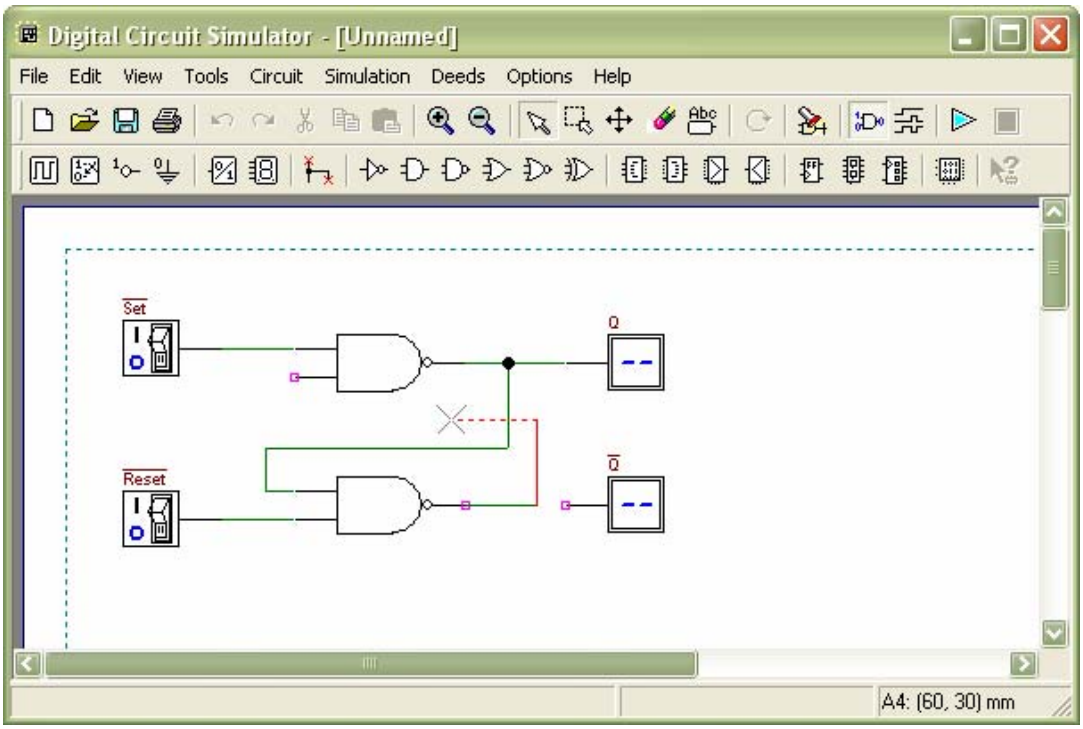


Fig. 17b: The next phase of the work: the connection of components, using wires  .

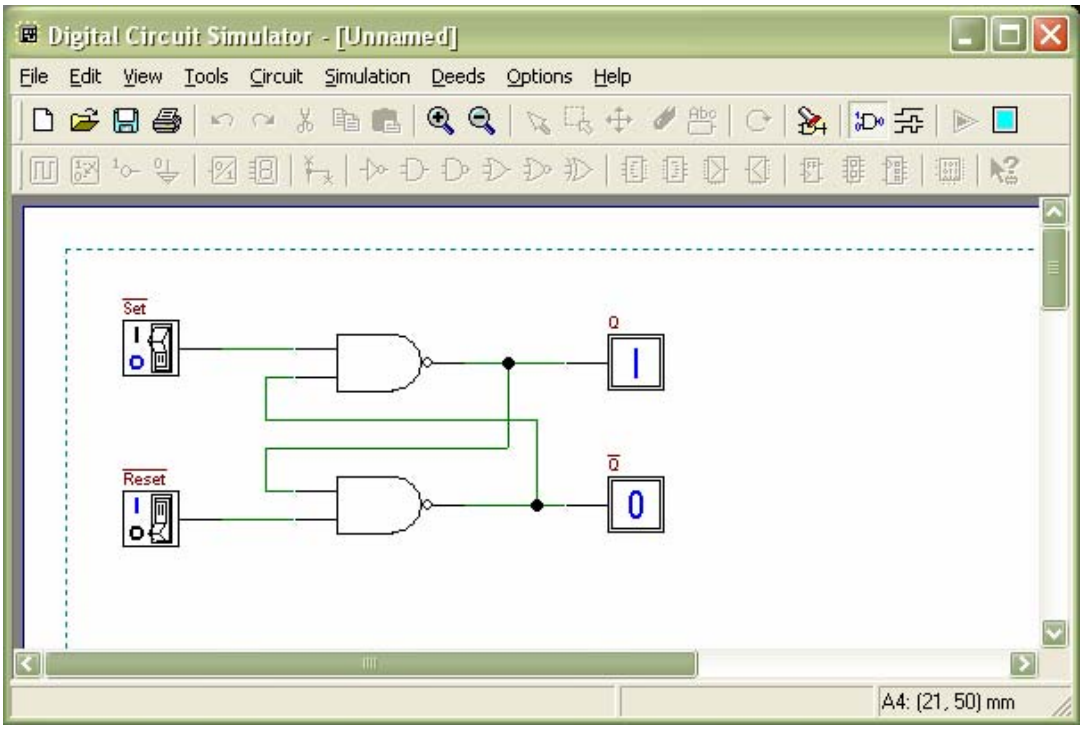


Fig. 17c: The animation at work: the user switches the Inputs and the circuit shows changes on the Outputs.

To enter the 'animation' mode, the user clicks on the triangular 'play' button on the toolbar.

During the animation, the editing command are disabled, and the circuit can't be changed; when the user clicks on the Input Switches (see Fig. 17c), the Outputs change according to the simulation results, showing '0' 0, '1' | or 'unknown' -- values.
To exit the 'animation' mode, it is necessary to click on the square 'stop' button.

Instead, if the timing simulation is to be performed, the user should click on the Timing Simulation button. This will show the Timing Diagram simulation window (Fig. 18), very similar to the ones that we find in professional tools for digital electronics.

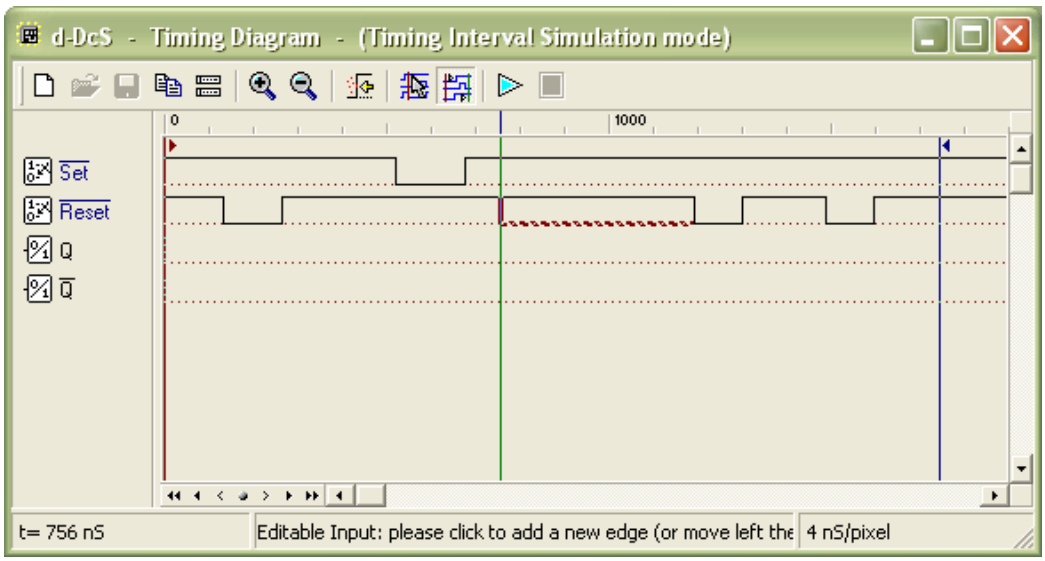


Fig. 18: The Timing Diagram simulation window.

In this window, first of all the user defines the timing of the input signals, drawing them on the diagram with the mouse. A vertical line cursor permits to define the ‘end time’ of the simulation. When the user clicks on the triangular ‘play’ button  on the toolbar, the simulation is executed, and its results are displayed in the same window (Fig. 19).

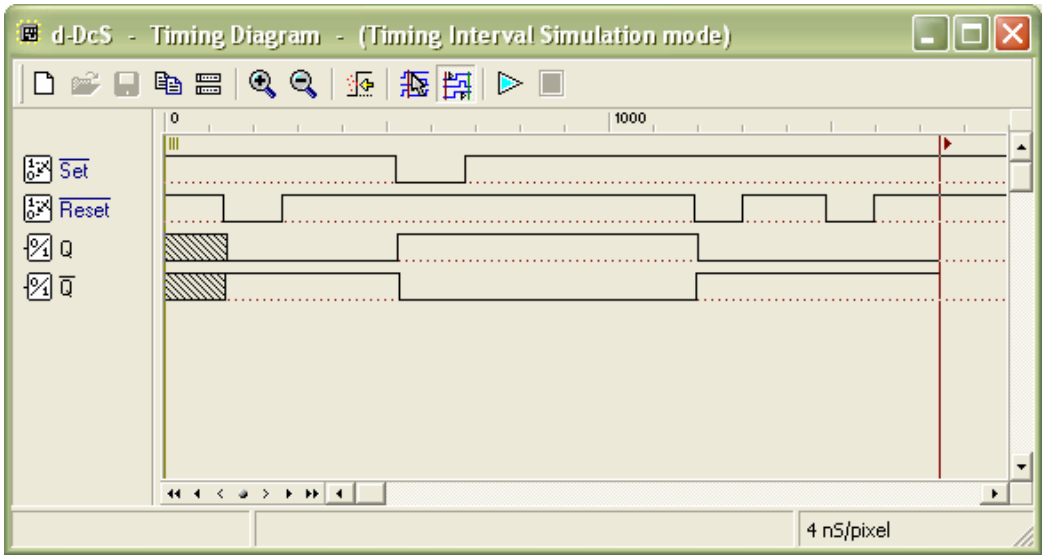


Fig. 19: The timing simulation results, displayed in the Timing Diagram window.

The student can verify the correct behaviour of the network under test, comparing simulation results with reasoning and theory concepts.

A simple example of interaction between Deeds browsers and d-DcS

In Fig. 20 a list of assignments is opened in the Deeds main browser. Suppose that the student has to attend the assignment # 2.1: *"Analysis of a demultiplexer (1 to 2)"*.

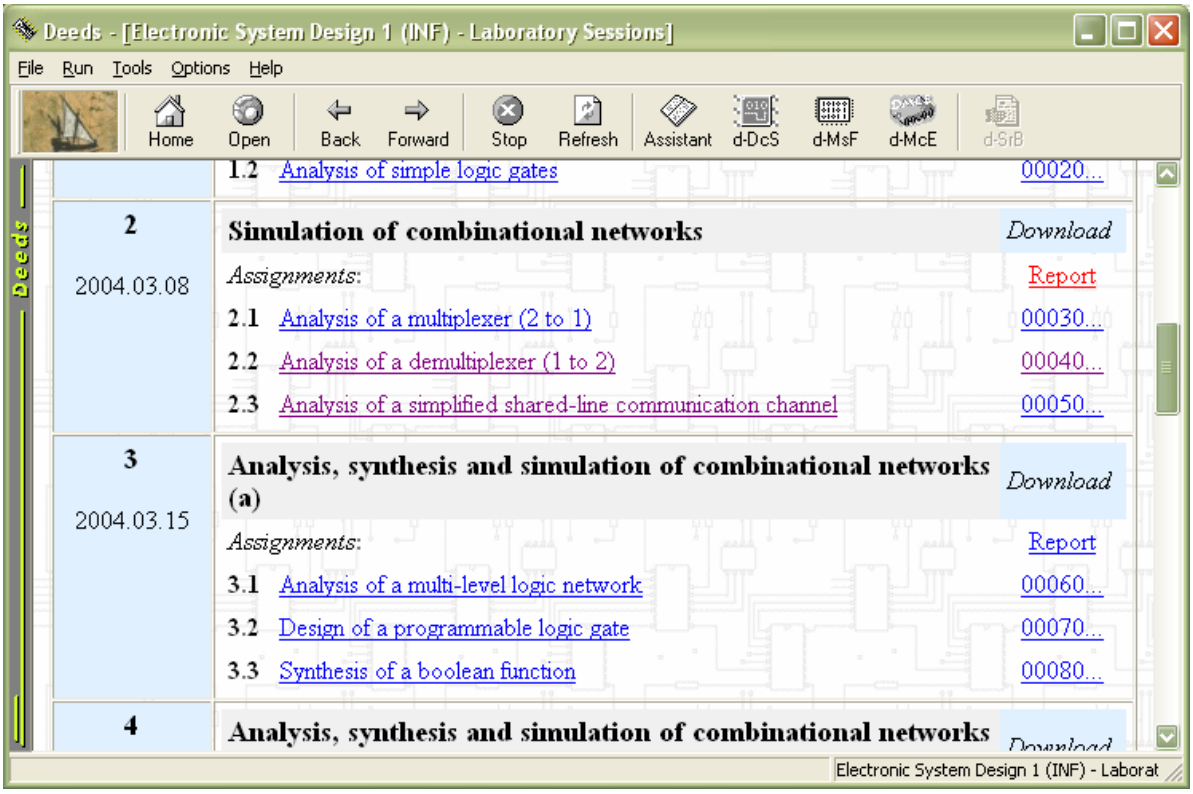


Fig. 20: A list of laboratory assignments, opened in the Deeds main browser.

Than, he or she clicks on the link, and the assignment will open in the Assistant (see Fig. 21).

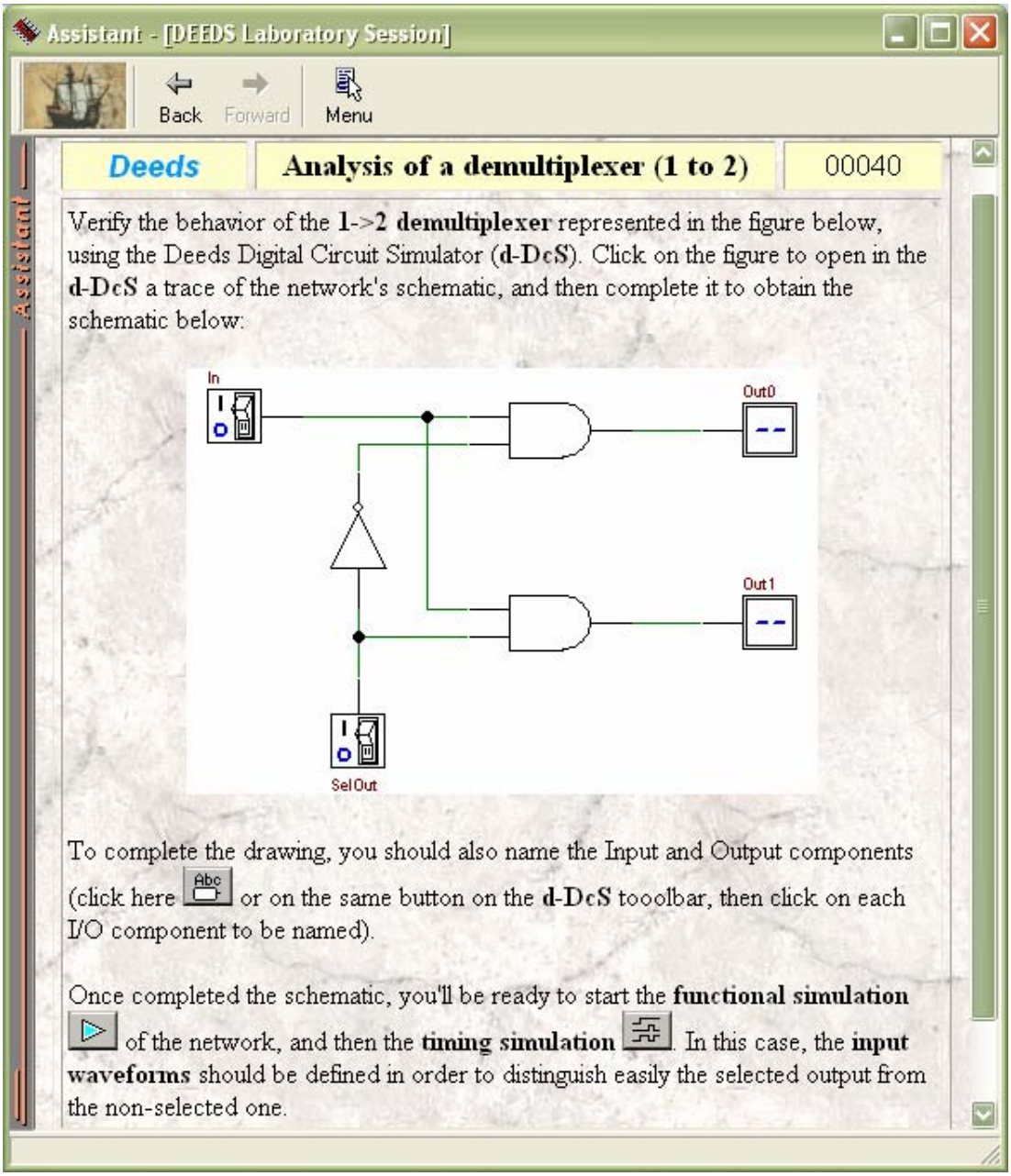


Fig. 21: The specific laboratory assignment, opened in the Assistant browser.

The assignment asks the user to verify the behavior of the 1->2 demultiplexer represented in the figure, using the Deeds Digital Circuit Simulator). The text suggests to click on the figure to open in the d-DcS a trace of the network's schematic, and then to complete it.

In this example, you see that it is necessary only a simple click on the figure to activate the simulator and to download from the web site a ‘template’ of the solution. This approach aims to simplify user operation, avoiding to spend time in no useful and distracting tasks.

The user will see the Digital Circuit Simulator, and the file downloaded in it, as in Fig. 22.

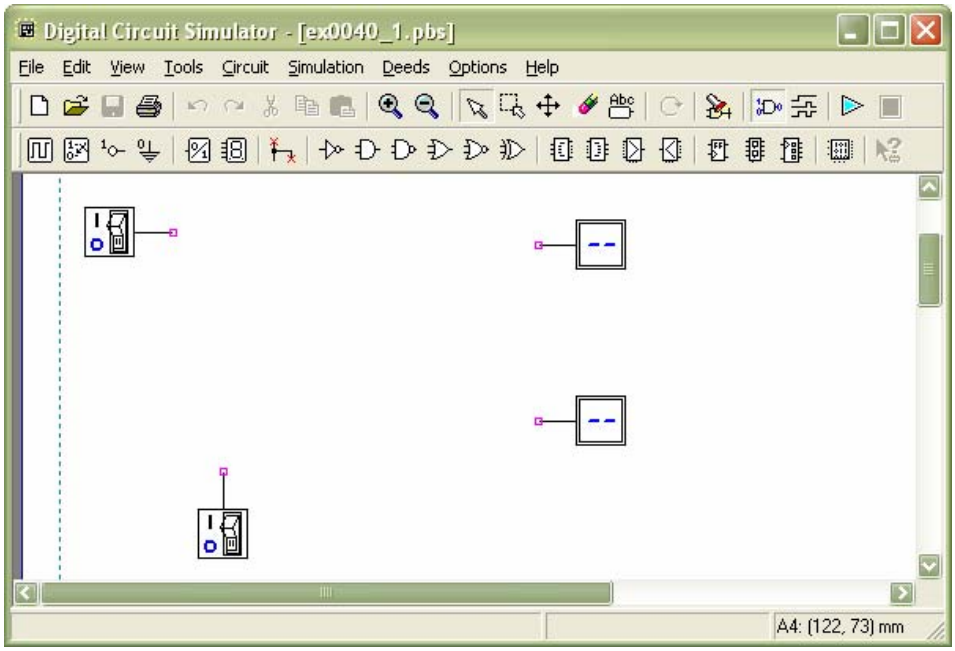


Fig. 22: The Digital Circuit Simulator, opened by a click on the web page. The circuit template has been automatically downloaded from the courseware site.

The assignment suggests now to complete the drawing, and also to activate a few useful simulator commands directly from the web page, with a simple click.

Once completed the schematic, also the simulation can be started, directly from the Deeds web page. In Fig. 23 you can see the results expected from the student work.

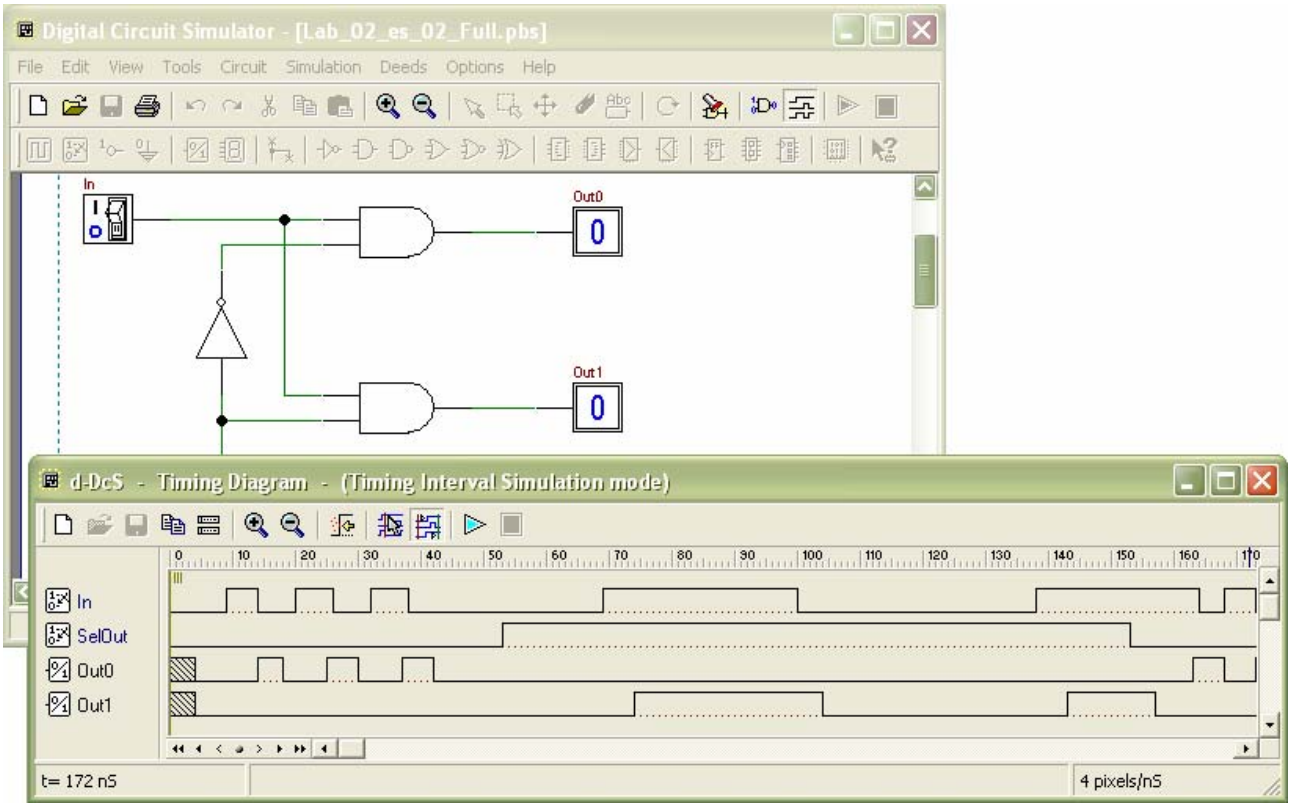


Fig. 23: The timing simulation of the circuit, once completed by the student.

Now is the time for the student to compile and deliver a good report. In the Deeds assignment page, a link is prepared to download and edit a report template file (Fig. 24).

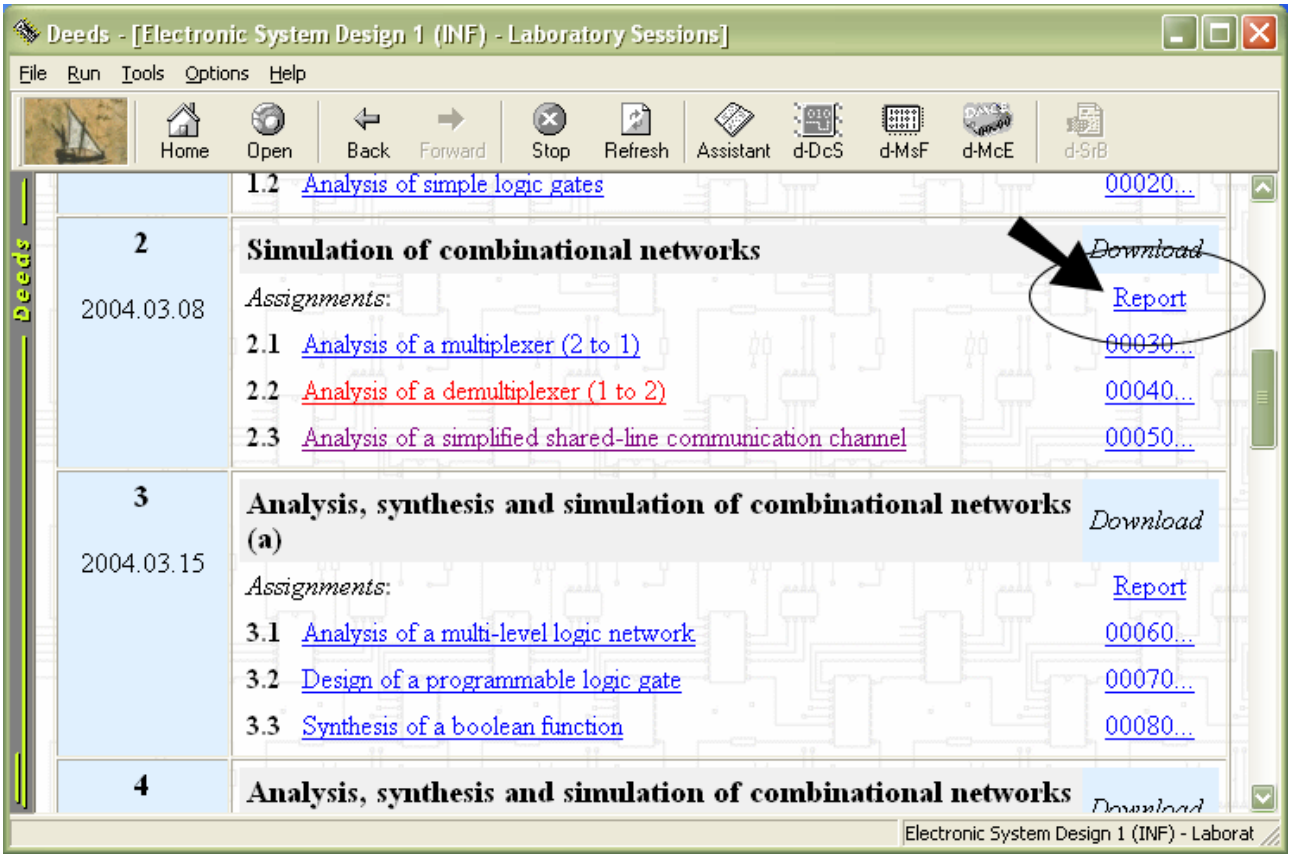


Fig. 24: The student can download the report template to speed up its compilation and delivering.

This has been previewed to uniform the report styles, making easier the teacher task, especially when the number of student is valuable. But the availability of a report template is very useful also to the student, because it saves a lot of time, speeding up the student work and leaving more time to concentrate on the arguments to learn.

This is the report template for this laboratory assignment (Fig. 25).

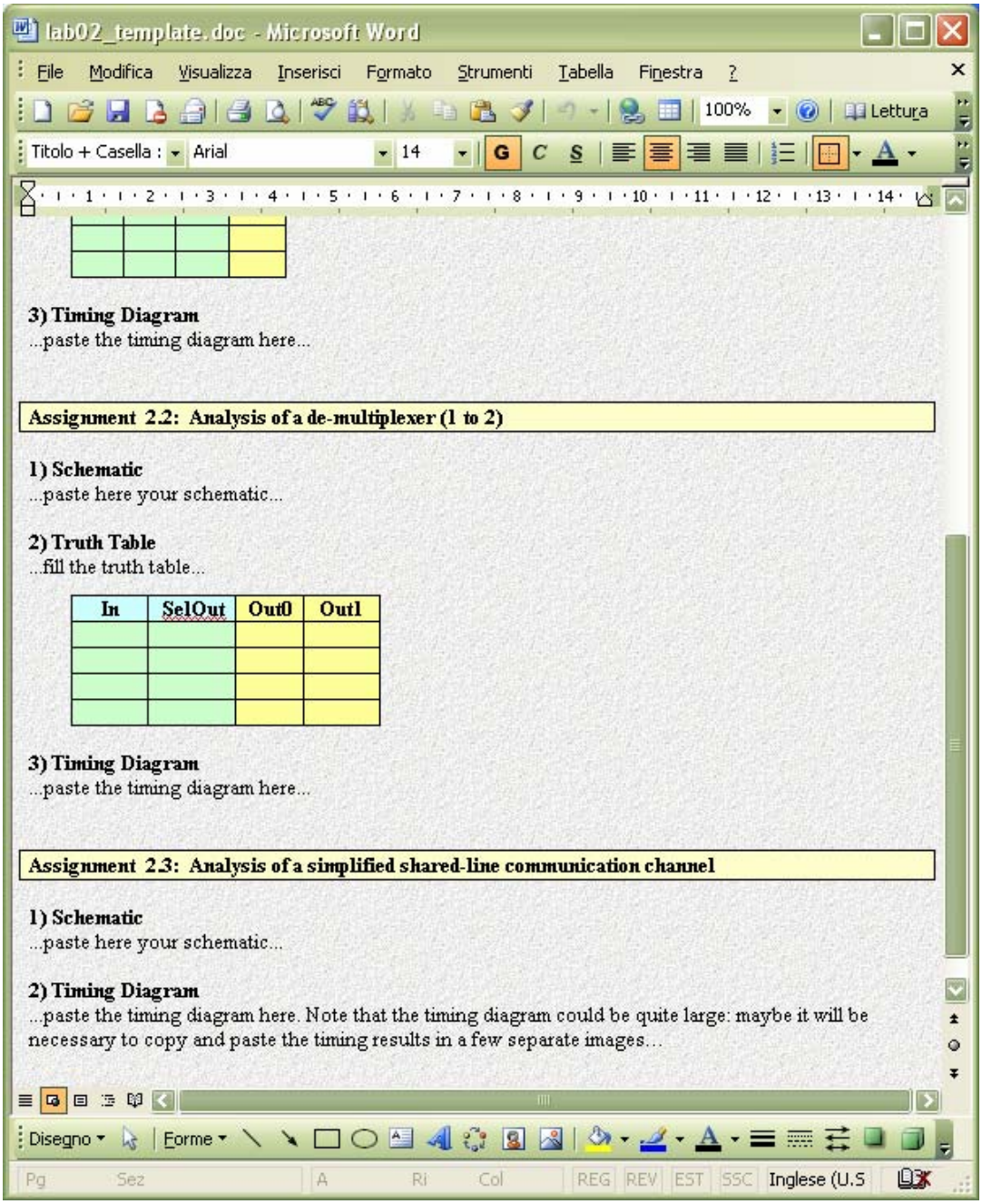


Fig. 25: The report template for this laboratory assignment.

d-DcS: Menu Commands

The menu of the Digital Circuit Simulator allows the user to access all the function of the application. The ToolBars replicate most of the commands already in the menu, to speed up user operations.

File Menu

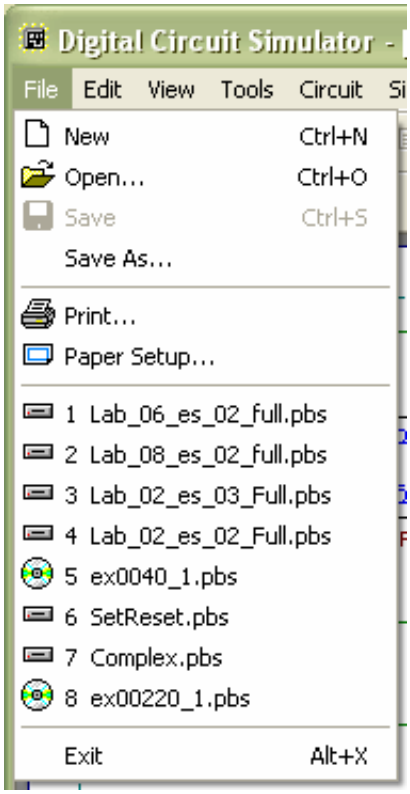


Fig. 26a: The d-DcS "File" menu.

New

Command to create a new circuit file.

Open

Command to open a circuit file. The file can be also downloaded directly from a web site.

Save

Command to save current circuit file.

Save as

Command to save current circuit file with a different name or in a different position.

Print

Command to print the circuit.

Paper Setup

Command to define current paper format and orientation. It displays the Paper Setup dialog window (Fig. 26b).

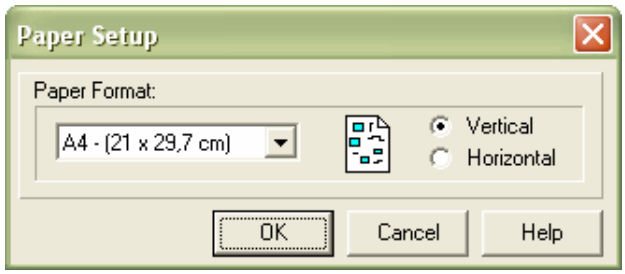


Fig. 26b: The Paper Setup dialog window.

Recent Files List

Commands to re-open the most recent files. Up to 8 recent files can be reopened with this list. The symbol that is displayed on the left of the file name means that:

	The file has been stored by the user on the local disk or network.
	The file has been downloaded from a web site, but it has not been saved (yet) on the local disk or network.
	The file has been loaded from a local courseware, where it is read only and it has not been saved (yet) on the local disk or network.

Exit

Standard command to close the application.

Edit Menu

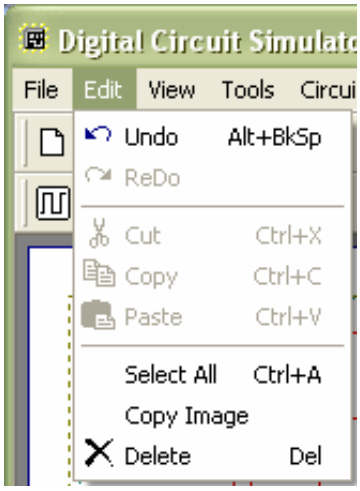


Fig. 27: The d-DcS "Edit" menu.

Undo

Command to undo the previous operation.

Redo

Command to redo the operation previously cancelled by the Undo command (command temporary inhibited).

Cut

Command to cut the selected part of the circuit, and copy it on the clipboard (command temporary inhibited).

Copy

Command to copy the selected part of the circuit on the clipboard (command temporary inhibited).

Paste

Command to paste the clipboard content in the circuit (command temporary inhibited).

Select All

Command to select all the object of the drawing.

Copy Image

Command to copy the selection as a bitmap image and put it on the Clipboard.

Delete

Command to delete all the selected components.

View Menu

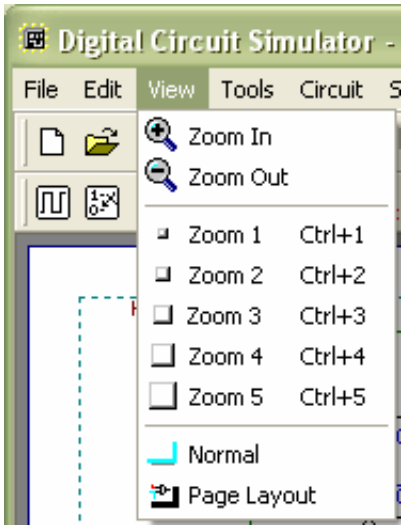


Fig. 28: The d-DcS "View" menu.

Zoom In

Command to "zoom in" the drawing.

Zoom Out

Command to "zoom out" the drawing.

Zoom 1,2,3,4,5

Command to "zoom" the view to different levels The "standard" level is the '3'.

Normal

Command to set the "normal view" of drawing space (i.e. as uniform continuous background, only with the indication of drawing margins).

Page Layout

Command to set the view of the drawing space as a paper foil (i.e. with visible foil borders and shadows, together with drawing margins).

Tools Menu

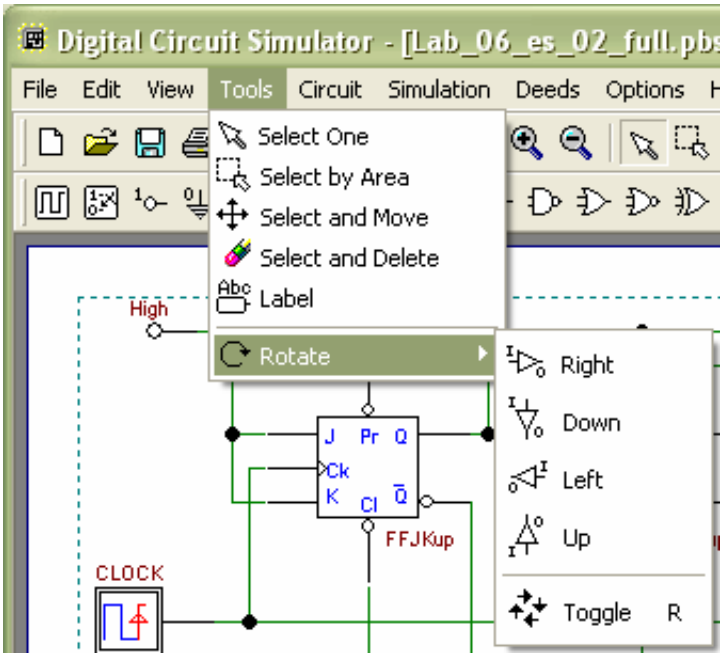


Fig. 29: The d-DcS "Tools" menu.

Select One

Command to selects one object (by point and click).

Select by Area

Command to select a group of objects in a rectangular area.

Select and Move

Command to select and move a single object (by point and click).

Select and Delete

Command to select and delete a single object (by point and click).

Label

Command to insert (or edit) the label of a selected object (it is possible to associate labels only to Input/Output blocks and to Finite State Machine components).

Rotate

Group of commands to rotate an object (during its insertion).

Right, Down Left, Up

Four commands to rotate an object (during its insertion) to the specified direction.

Toggle

Command to toggle the direction of an object (during its insertion).

Circuit Menu

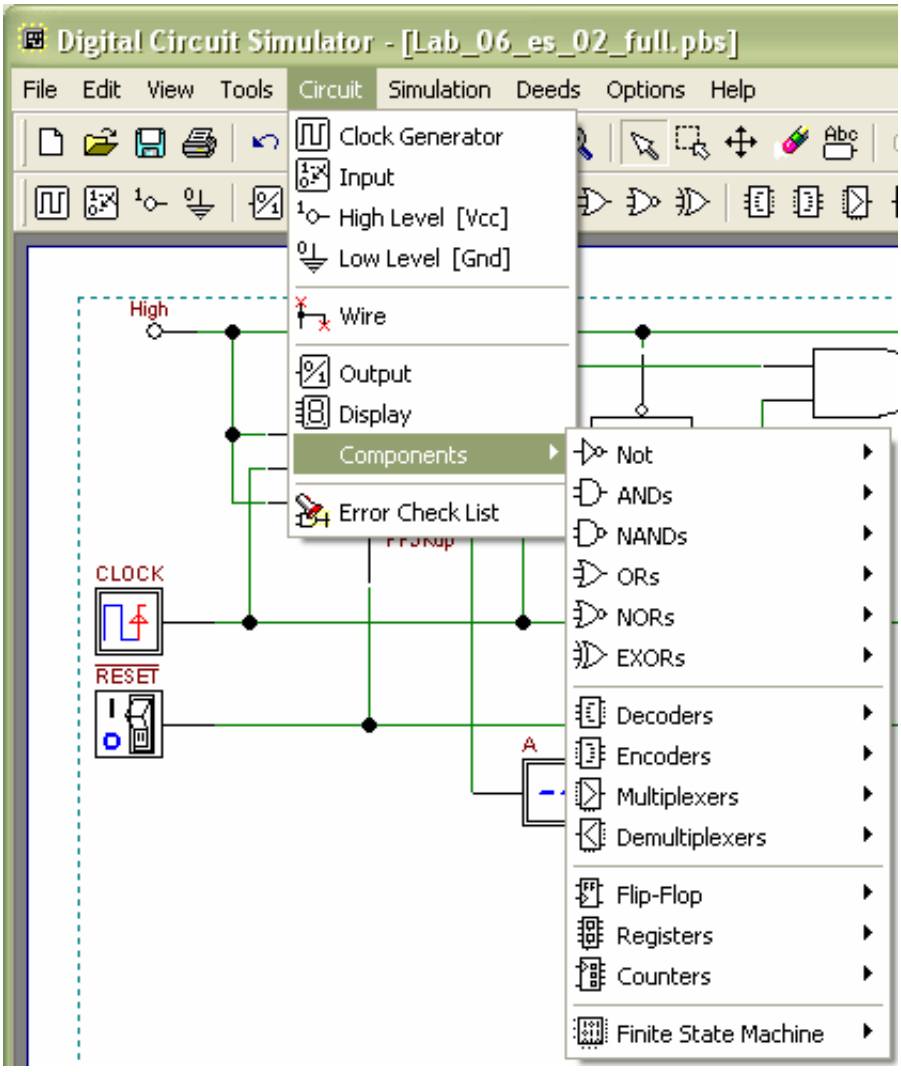


Fig. 30: The d-DcS "Circuit" menu.

Clock Generator

Command to insert in the circuit a Clock Generator component.

Input

Command to insert in the circuit a Input Switch component.

High Level

Command to insert in the circuit a High Level Input component (logic '1').

Low Level

Command to insert in the circuit a Low Level Input component (logic '0').

Wire

Command to insert in the circuit a wire segment. The wiring system supports automatic insertion of "wire nodes" when a wire is connected to another one.

Output

Command to insert in the circuit a binary Output Display component (it displays ‘0’, ‘1’ or ‘unknown’ symbols).

Display

Command to insert in the circuit an Hexadecimal Output Display component (it displays hex digits from ‘0’ to ‘F’, or a ‘unknown’ symbol).

Input

Command to insert in the circuit a Input Switch component.

Error Check List

Command to error check the wiring of the circuit. It shows or hides, at the bottom of the window, an “error check list” of wire connections.

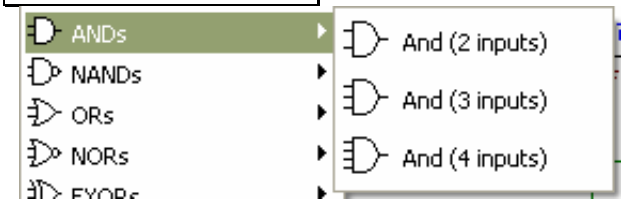
Components

Command to insert in the circuit a component, selected by the user in the sub menu. A description of all the sub menu’s is reported in the following.

Not

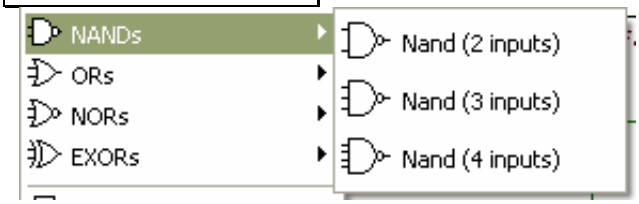
Command to insert a ‘NOT’ component.

ANDs



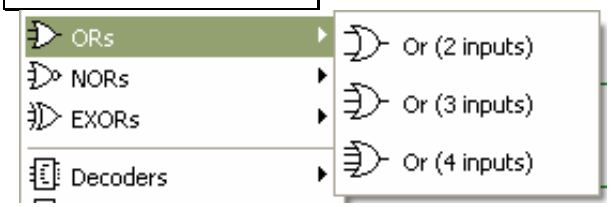
Commands to insert ‘AND’ components.

NANDs



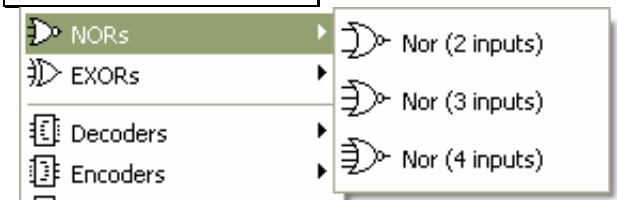
Commands to insert ‘NAND’ components.

ORs



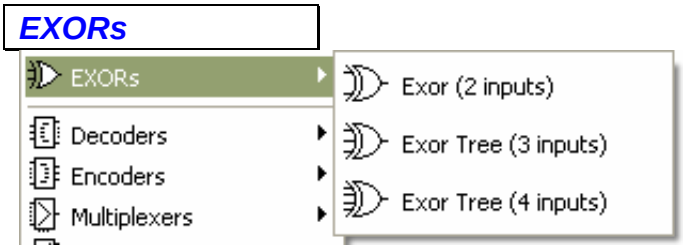
Commands to insert ‘OR’ components.

NORs



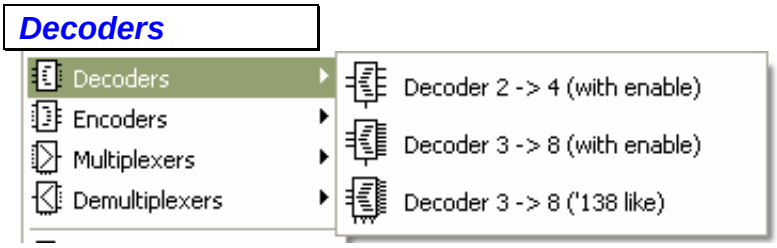
Commands to insert ‘NOR’ components.

EXORs



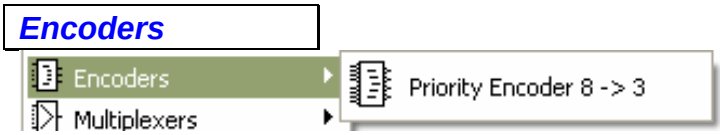
Commands to insert ‘EXOR’ and ‘EXOR tree’ components.

Decoders



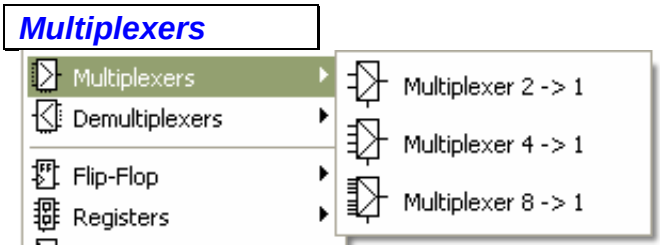
Commands to insert ‘Decoder’ components.

Encoders



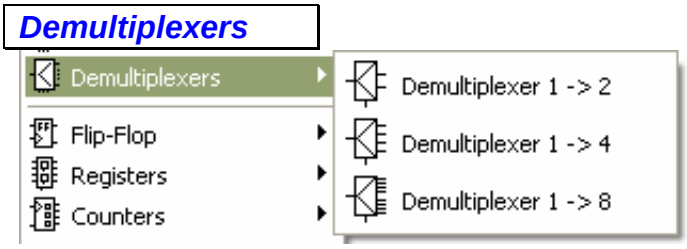
Commands to insert a ‘Priority Encoder’ component.

Multiplexers



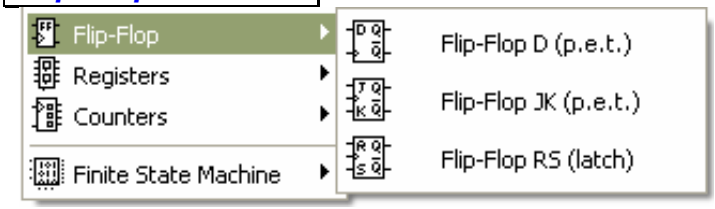
Commands to insert ‘Multiplexer’ components.

Demultiplexers



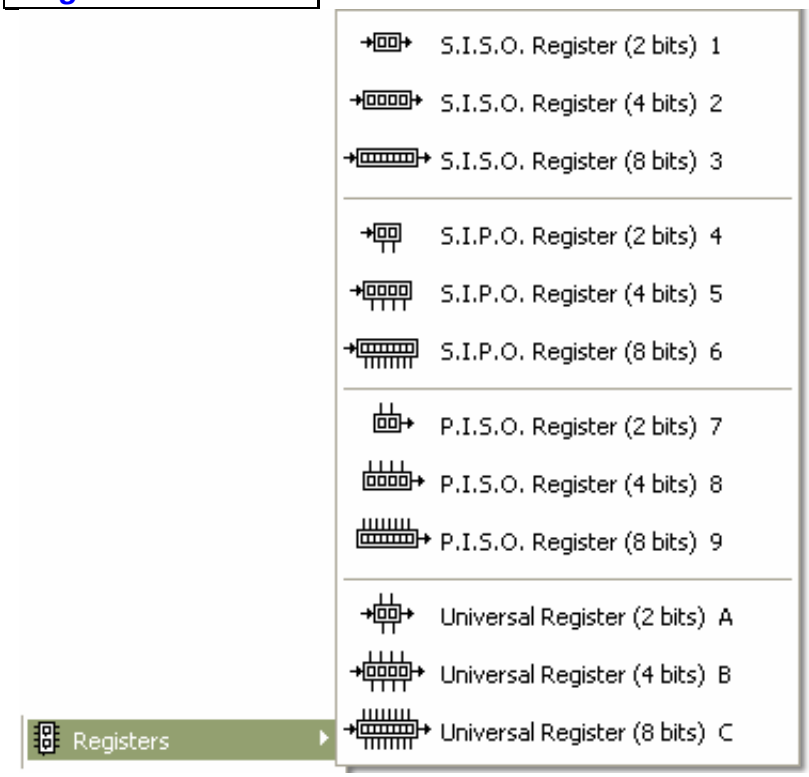
Commands to insert ‘Demultiplexer’ components.

Flip-Flop



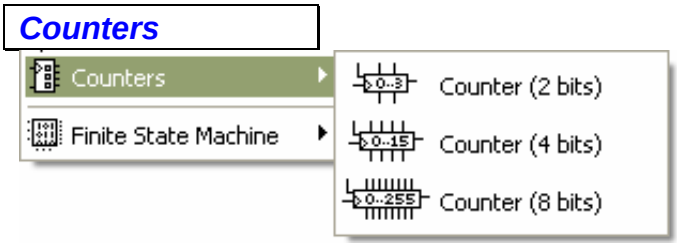
Commands to insert 'Flip-Flop' components.

Registers



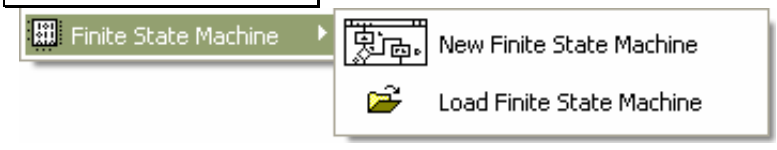
Commands to insert 'Register' components.

Counters



Commands to insert 'Counter' components.

Finite State Machine



Commands to insert 'Finite State Machine' components.
The 'New' command activate the Finite State Machine Simulator (d-FsM), allowing the user to create a new component 'from scratch'.
The 'Load' command allows the user to load a previously designed component.

Simulation Menu

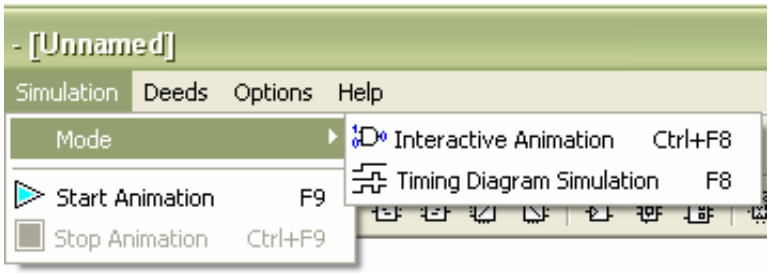


Fig. 31: The d-DcS "Simulation" menu.

Mode

Command group to set the simulation mode.

Interactive Animation

Command to set the Interactive Animation Mode for simulation. When activated, simulation don't start immediately. If the Timing Diagram window is opened, it will be closed. The editing commands are disabled, and the user is prompted to save the file in the schematic editor, if it is not.

Timing Diagram Simulation

Command to set the Timing Diagram Mode for simulation. When activated, simulation doesn't start immediately, but the Timing Diagram window is opened instead. The editing commands are disabled and the user is prompted to save the file in the schematic editor, if it is not.

Start Animation

Command to start simulation, when currently mode is 'Animation'.

Stop Animation

Command to stop simulation, when currently mode is 'Animation'.

Deeds Menu

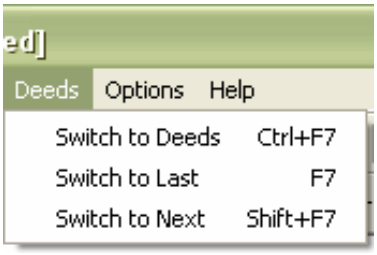


Fig. 32: The d-DcS "Deeds" menu.

Switch to Deeds

Command to switch focus to the Deeds main browser.

Switch to Last

Command to switch to the tool that was 'last on top' before switching to the currently opened instance of the d-DcS.

Switch to Next

Command to switch focus among all active Deeds applications, in order of activation.

Options Menu

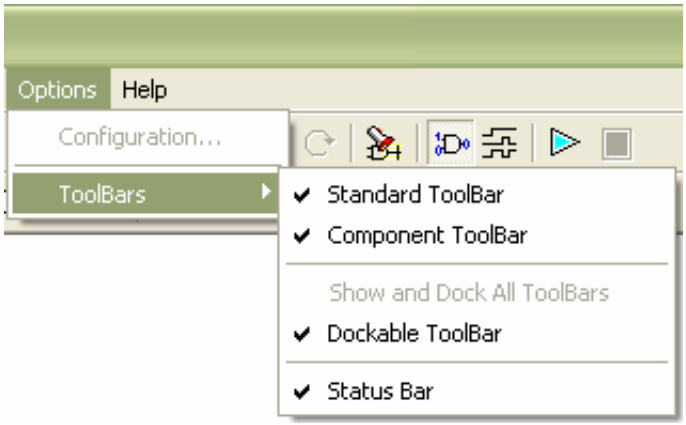


Fig. 33: The d-DcS "Options" menu.

Configuration

Command to change the application configuration (disabled in this version).

ToolBars

Commands to control ToolBars appearance.

Standard ToolBar

Command to hide or show the Standard ToolBar (the upper one).

Component ToolBar

Command to hide or show the Component ToolBar (the lower one).

Show and Dock All ToolBars

Command to show and dock in all the ToolBars.

Dockable ToolBars

Command to enable or disable the docking modality of the ToolBars.

Status Bar

Command to hide or show the Status Bar.

Help Menu

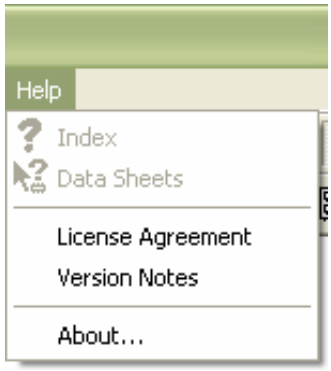


Fig. 34: The d-DcS "Help" menu.

Index

Command to open the d-DcS Help System (disabled in this version).

Data sheets

Command to open the Data Sheets help system (disabled in this version).

License Agreement

Command to display the Licence Agreement.

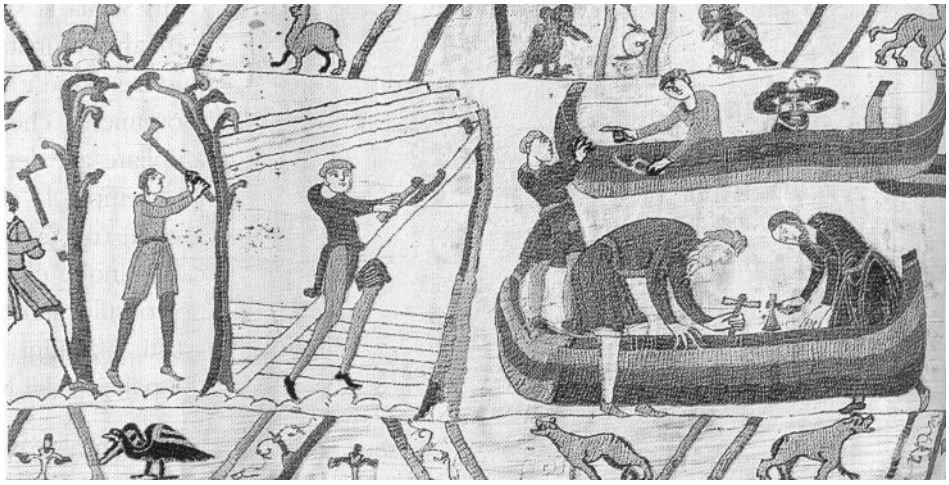
Version Notes

Command to display the Deeds "Version Notes" file.

About

Command to display the d-Dcs 'splash' window dialog.

Deeds: Finite State Machine Simulator d-FsM



This image from the Tapestry of Bayeux, Bayeux Cathedral, France

Introduction

The **Finite State Machine Simulator** ***d-FsM*** allows graphical editing and simulation of Finite State Machines components, using the ASM (Algorithmic State Machine) paradigm (fig. 35). The tool allows the local functional simulation of the finite state machines designed by the user, with runtime display of the relations between state and timing evolution (fig. 36).

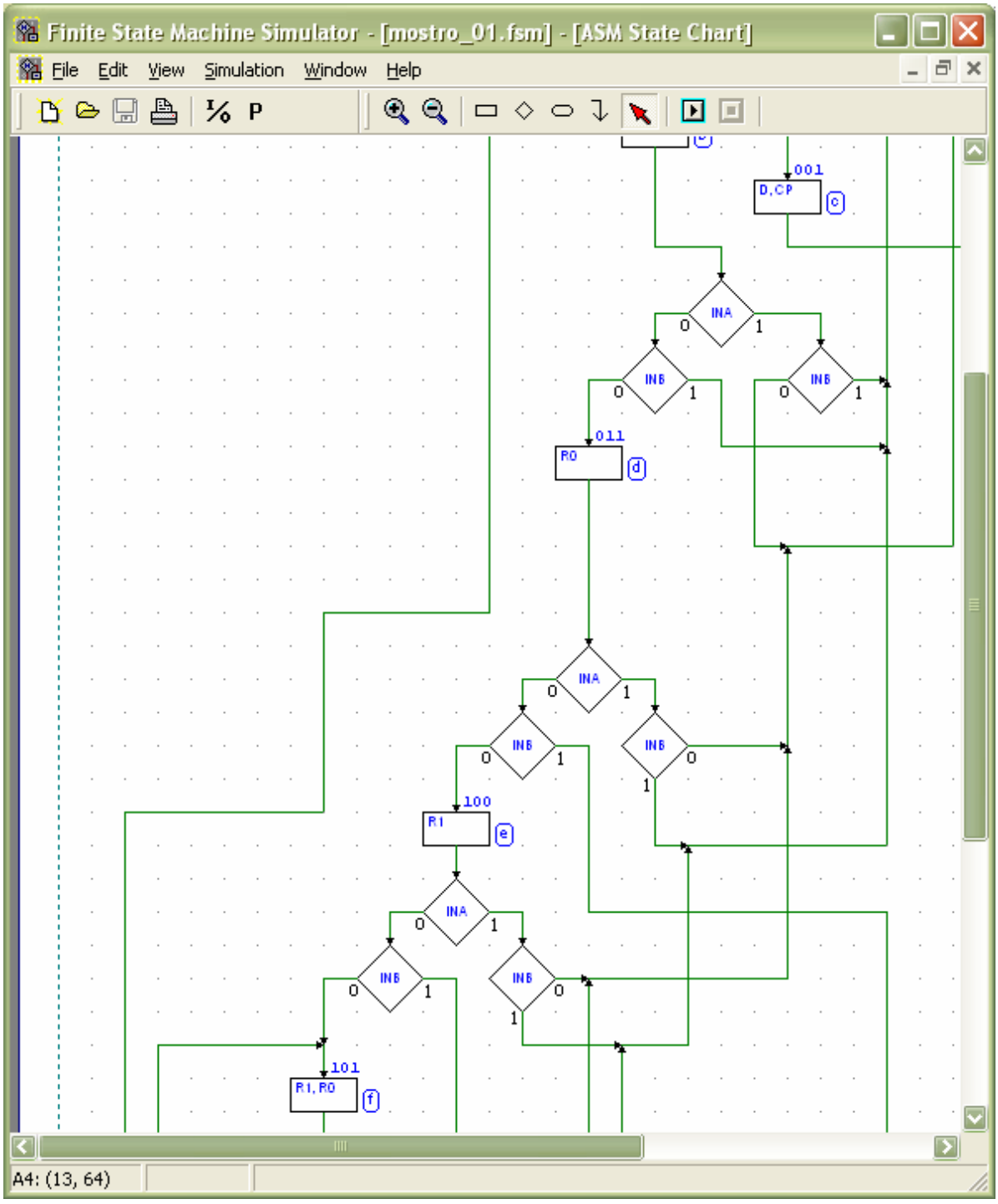


Fig. 35: The ASM editor of the Finite State Machine Simulator (d-FsM).

The components that the d-FsM produces can be directly used in the d-DcS and inserted into any digital circuit. Also, it can be exported in VHDL language.
A general purpose Finite State Machine software simulator helps the student to enhance his design skills and facilitates also the transition from the pedagogical to the professional field, by introducing CAD methodologies.

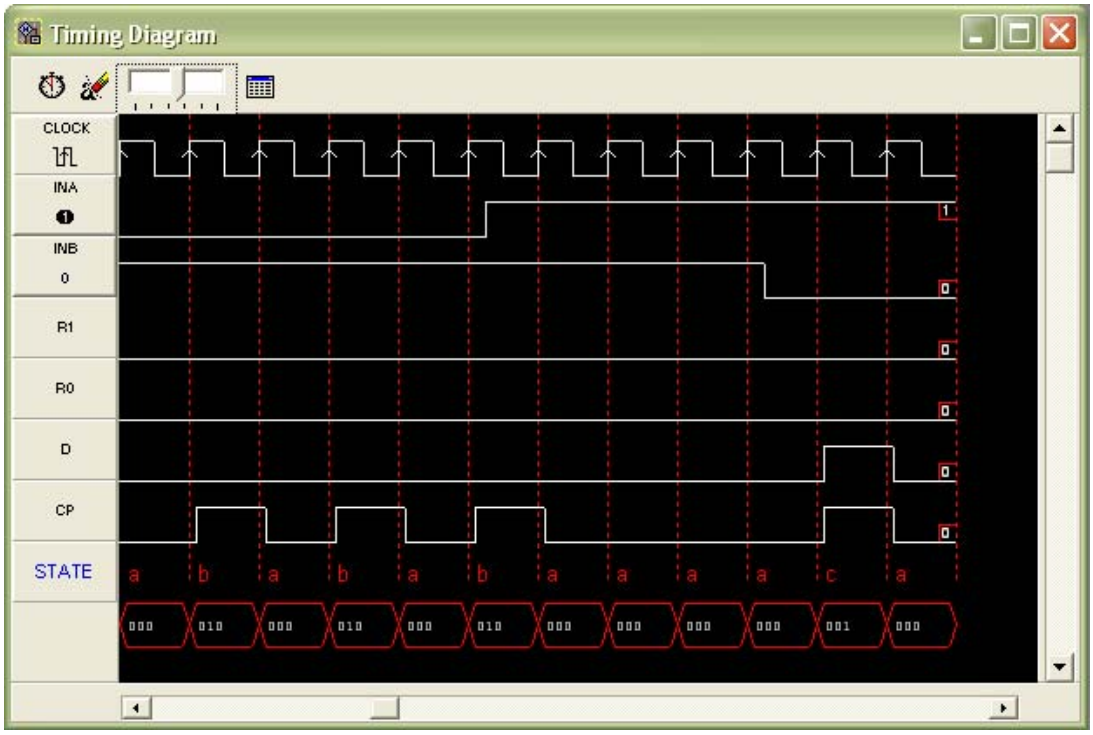


Fig. 36: The ASM editor of the Finite State Machine Simulator (d-FsM).

Finite State Machines

Finite State Machines (FSM) represent a model to design a class of digital sequential circuits. A sequential system is a block whose outputs are a function not only of the current inputs but also of the previous ones. In other words, the logic has a sort of “memory” which records previous input history so it can be responded to in the present.
Given this definition, sequential circuits would seem to require enormous amounts of memory to record all previous inputs. However, for any real logic design task, the fact that previous input combinations result in only a finite number of distinct output classes reduces this memory requirement to manageable levels. This class of design is called a Finite State Machine, or just a state machine.
Modern digital circuit design is essentially based on Finite State Machines. Design, synthesis and documentation of a state machine require a formal approach. Currently, several design methods are employed, based either on graphic, tabular or textual representations of the algorithm underlying the state machine.

FSM description languages: ASM charts

The most common graphical methods currently in use to describe a FSM are Moore and Mealy State Diagrams. In our simulator we use the ASM (Algorithmic State Machine) method, instead.

A typical ASM chart (or diagram) resembles flowchart notation (Fig. 37), even if they are not the same thing. It describes state flow, the output functions and the next-state functions of a state machine. ASM charts have the same function as Moore and Mealy State Diagrams: they describe the behaviour of finite state machines so that it is clearly understandable for the designer and, at the same time, ASM charts support a direct translation into a hardware realization of the control algorithm.

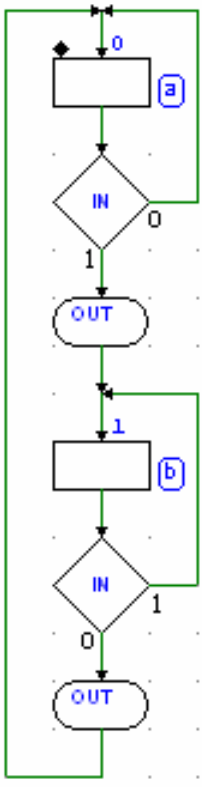


Fig. 37: A simple Algorithmic State Machine (ASM) diagram.

An ASM chart is composed of three basic elements, the **State** (rectangular box), the **Decision Block** (diamond) and the **Conditional Output Box**.

A set composed of one state box, decision blocks and conditional output blocks is named **ASM Block**. An ASM Block has one entry point, but may have any number of exit paths, each of them connecting to another state box.

The FSM moves from state to state at each clock cycle; each state may have a state output; conditional blocks allow choosing a direction as a function of the value of the inputs; conditional outputs depend not only on states but also on input values.

State Block

On an ASM chart, a state is represented by a state box, which is a rectangle with the name of the state encircled and placed at the side of the rectangle (Fig. 38a). You can specify that an output signal is *unconditionally* active in a particular state by writing the output signal's name inside the corresponding state box. Output signals written inside state boxes are known as state outputs or Moore outputs.

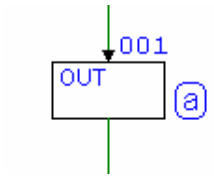


Fig. 38a: State Block

Decision Block

While unconditional transitions can be represented with a straight, not labelled arrow traced between two state boxes, conditional transitions deserve a more specific symbol. This is called decision diamond. Depending on the value of the expression written inside the diamond, the machine will follow one of the two labelled transition arrows going out of the diamond. A diamond has always two outgoing arrows, one labelled "1" (or TRUE) and the other labelled "0" (or FALSE) that corresponds to the values of the *boolean expression* inside.

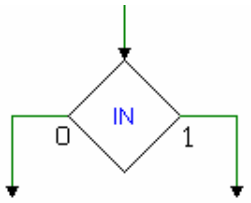


Fig. 38b: Decision Block

Conditional Output Block

Sometimes you may need to activate an output signal in a particular state only if a certain condition on inputs is satisfied (such output signals are known as conditional outputs or Mealy outputs). In that case you need to use the conditional output block.
Just put the ellipse on a transition arrow coming out of a decision diamond, and write inside the ellipse the name of the output signal you want to activate when the expression inside the diamond is true. Please notice that the conditional block does not represent a state; instead it activates an output that it is active in the state it descends from.

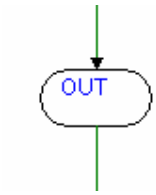


Fig. 38c: Conditional Output Block

ASM Charts & State Diagrams

It is easy to convert a State Diagram in an ASM Chart, and vice versa. In Fig. 39a we report a basic example of State Diagram:

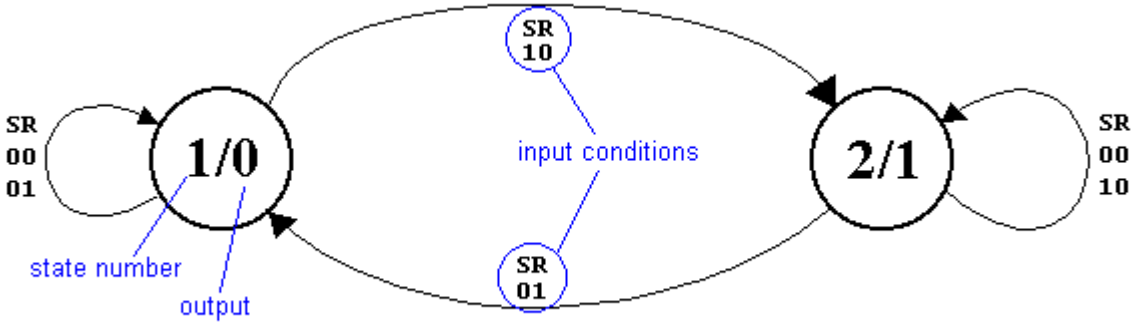


Fig. 39a: The State Diagram representation of a SR flip-flop.

The following ASM Chart (Fig. 39b) can be used to model exactly the same behaviour:

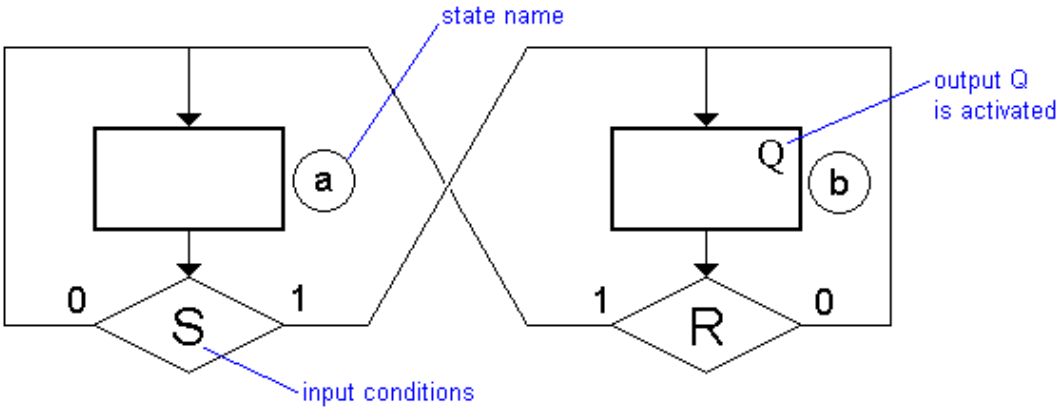


Fig. 39b: The ASM Chart representation of a SR flip-flop.

The first thing you can see is that in both models you have an object to represent the states of the machine. The states are numbered (1, 2) in the State Diagram and labelled with letters (a, b) in the ASM Chart, but the 1:1 relationship between them is obvious:

- state 1 = state a = flip-flop output "0"
- state 2 = state b = flip-flop output "1"

Another noticeable thing is that the two models are morphologically very similar. In both models you can observe that every state has two outgoing transitions, one being a loop on the state itself, and the other going to the other state. This similarity is always true if you make a conversion between ASM Charts and State Diagrams, just remember that in ASM Charts conditional transitions come out of decision diamonds which are not states (but they "belong" to the state they descend from).

The method used to represent conditional transitions on ASM Charts is more algorithm-oriented, as it uses flow-chart syntax, which is less redundant than State Diagram syntax. In this case, for example, it helps the reader understand that the transition that follows state a depends only on the value of the S input. Similar considerations can be done about the transition that follows state b: only the value of the R input is relevant in that case.

The following pictures are examples of ASM Chart <-> State Diagram conversion (Fig. 40a and 40b).

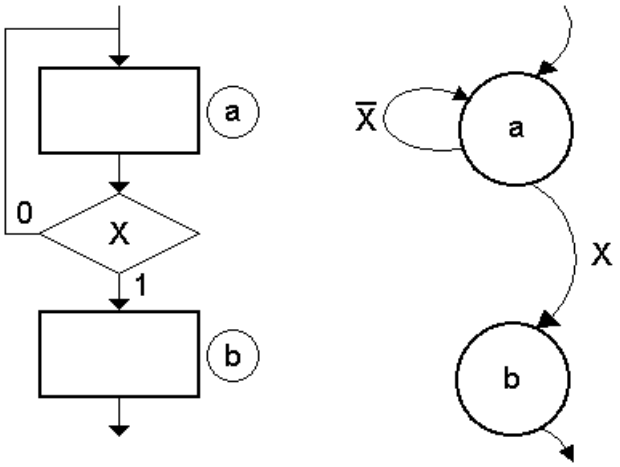


Fig. 40a: ASM chart and State diagram representing the same algorithm: the FSM waits in the state 'a' until the x input goes to one.

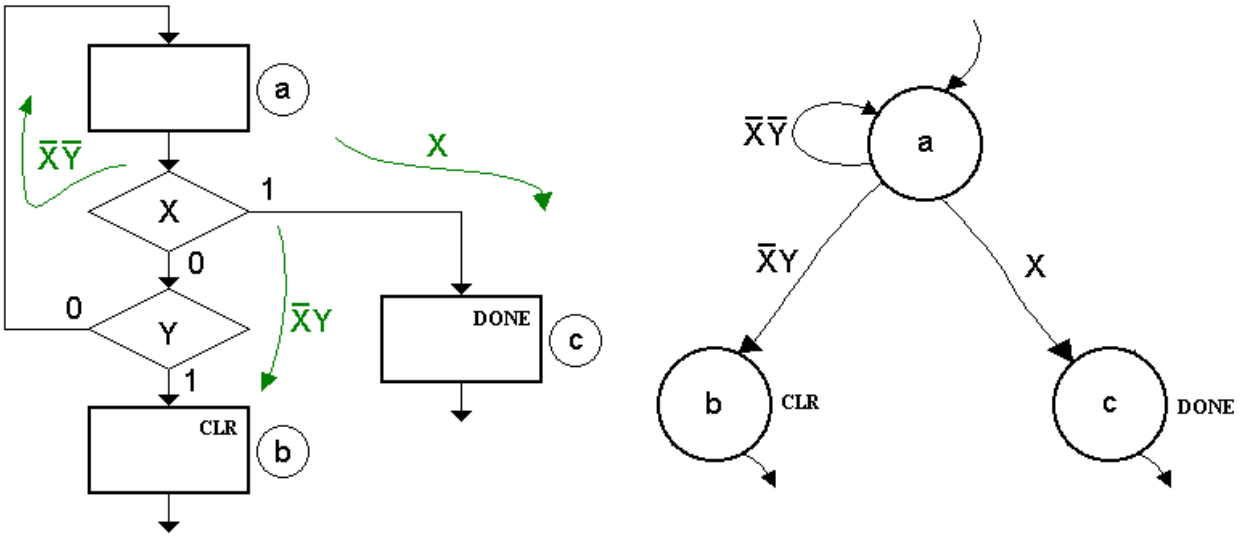


Fig. 40b: Another example of ASM chart and State diagram representing the same algorithm.

FSM description languages: state transition table

The state transition table (Fig. 41) is the most compact description of a FSM and lends itself very well to be used as interface with computer software and as a basis for the logical synthesis of the hardware. Of course, the table is not a valid FSM design tool because it does not provide any help in conceiving the FSM algorithm. Its main usefulness rests therefore in its use as a synthetic representation that may be common to both the languages described above.

ASM Table

	IN	State	OUT	Next State
1	0	0	0	0
2	1	0	1	1
3	0	1	1	0
4	1	1	0	1

Fig. 41: The state transition table of the example above, as generated by the d-FsM.

FSM description languages: hardware description language

The use of circuit description languages (HDL, VHDL, Verilog) to represent finite state machine has gained a strong diffusion and probably in many cases has replaced the graphical languages. The description of the state machine takes in this case the format of a high level software program.
The Finite State Machine Simulator exports the FSM components in VHDL format (Very High speed integrated circuits Hardware Description Language). In Fig. 42 you can see the VHDL equivalent of the ASM diagram in Fig. 37, as generated by the Finite State Machine Simulator.

The list starts with the “Entity” i.e. the definition of the FSM as a block with inputs and outputs. Then an object (Architecture) of the entity is instantiated. An entity may be described in three different ways: structural, data flow, functional. The structural description decomposes the entity in terms of basic digital components and their connections. The data flow description represents the FSM in terms of signals and operations on them. The last description, the functional one, is the more powerful because it allows to see the hardware circuit as a software program with input and output variables.

The FSM is therefore described as a process activated, in our case, by the clock or reset signals. Each state is coded as an internal variable. An instruction “case” within each state defines the outputs to activate and the next state.

```
------------------------------------------------------------
-- DEEDS (Digital Electronics Education and Design Suite)
-- VHDL Code generated
--      by Finite State Machine Simulator (d-FsM)
-- Copyright © 2001-2004 DIBE, University of Genoa, Italy
--      Web Site:  http://esng.dibe.unige.it/netpro/Deeds
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Deriv_UC IS
  PORT( ----------------------->Clock & Reset:
        Ck:    IN std_logic;
        Reset: IN std_logic;
        ---------------------->Inputs:
        IIN:   IN std_logic;
        ---------------------->Outputs:
        OOUT:  OUT std_logic );
END Deriv_UC;

ARCHITECTURE behave OF Deriv_UC IS       -- (Behavioral Description)
  TYPE states is ( state_a,
                   state_b );
  SIGNAL State,
         Next_State: states;
BEGIN
  -- Next State Combinational Logic ---------------------------------
  FSM: process( State, IIN )
  begin
    CASE State IS
      when state_a =>
                if (IIN = '1') then
                  Next_State <= state_b;
                else
                  Next_State <= state_a;
                end if;
      when state_b =>
                if (IIN = '1') then
                  Next_State <= state_b;
                else
                  Next_State <= state_a;
                end if;
    END case;
  end process;

  -- State Register -------------------------------------------------
  REG: process( Ck, Reset )
  begin
    if (Reset = '0') then
              State <= state_a;
    elsif rising_edge(Ck) then
              State <= Next_State;
       end if;
  end process;

  -- Outputs Combinational Logic ------------------------------------
  OUTPUTS: process( State, IIN )
  begin
    -- Set output defaults:
    OOUT <= '0';

    -- Set output as function of current state and input:
    CASE State IS
      when state_a =>
                if (IIN = '1') then
                  OOUT <= '1';
                end if;
      when state_b =>
                if (IIN = '0') then
                  OOUT <= '1';
                end if;
    END case;
  end process;
END behave;
```

Fig. 42: The VHDL equivalent of the ASM diagram in Fig. 37, as generated by the d-FsM.

Learning FSM: methods and problems

The choice of a FSM description language is very important under the pedagogical point of view. When first introducing the state machine, we believe it is essential that the learner masters its fundamental concepts and develop an intuitive understanding of its behaviour. At this level, therefore we believe it is convenient to represent the machine algorithms with graphical methods, in our case ASM charts.
When the student has gained familiarity with the design method and is ready to develop non-standard digital structures described by a set of specifications, switching to an hardware description language will develop his abstraction skills and introduce him to professional design.
The VHDL export feature has been developed to make easier the transition from ASM description method to the HDL-based world.

Reusing FSM component: they can be imported in d-DcS

As said before, the component the d-FsM produces can be directly used in the d-DcS and inserted into any digital circuit. In Fig. 43 you see a screen shot where the simple component (seen before) is imported in the d-DcS, and the network is simulated.

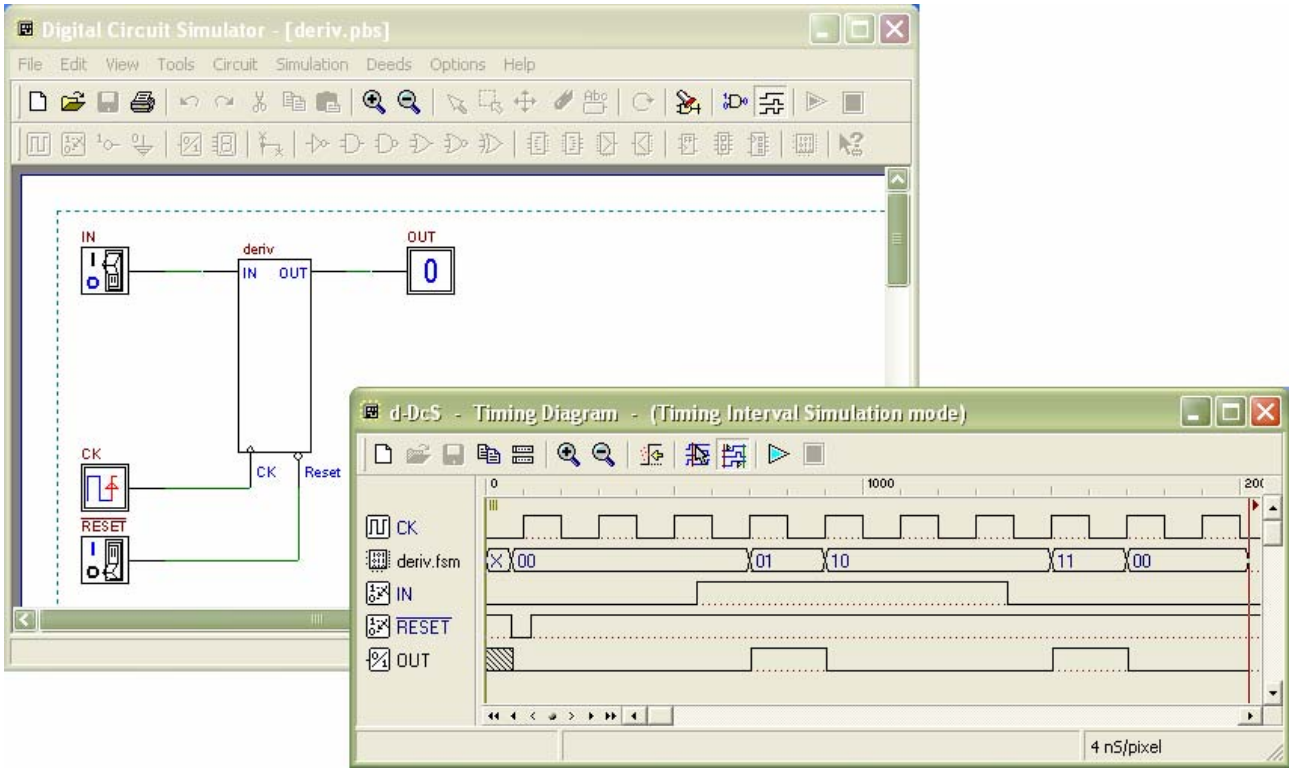


Fig. 43: In this example, a component, designed with the d-FsM, has been imported in the d-DcS.

In the d-DcS the FSM interpreter works together the simulator kernel to produce functional results. FSM to a maximum number of 64 states can be designed and simulated, and a practical limitation to 8 inputs and 8 outputs has been introduced, mostly for graphical reasons. Such limitations are largely compatible with the learning aims of the simulator. The FSM interpreter is able to simulate synchronous FSM with conditioned outputs.
In the d-DcS the student can drive the inputs and observe the outputs of the FSM block as well as the internal state of the FSM (in Fig. 43, the row named with the name of the component: 'deriv.fsm').
The user can connect standard digital components to the FSM block and therefore simulate digital systems characterised by a functional division between architecture and controller, the last one being implemented by a finite state machine.
If a student wishes to compare the results with the ones obtained by traditional synthesis, he can proceed manually using the table of transitions or the ASM chart in order to obtain a traditional structure with a state register made by flip-flops and a combinational network based on logic gates.

A simple example

In following screen shots of the d-FsM (Fig. 44a,b,c), you can see the drawing of an ASM diagram, followed by a preliminary verification in the internal timing simulator.

d) the student picks-up state blocks from the bin on the Tool Bar (Fig. 44a), then
e) adds conditional blocks (Fig. 44b) and, by last,
f) connect logically them using lines (Fig. 44c).

At every step, when needed, the student sets properties of each introduced block.

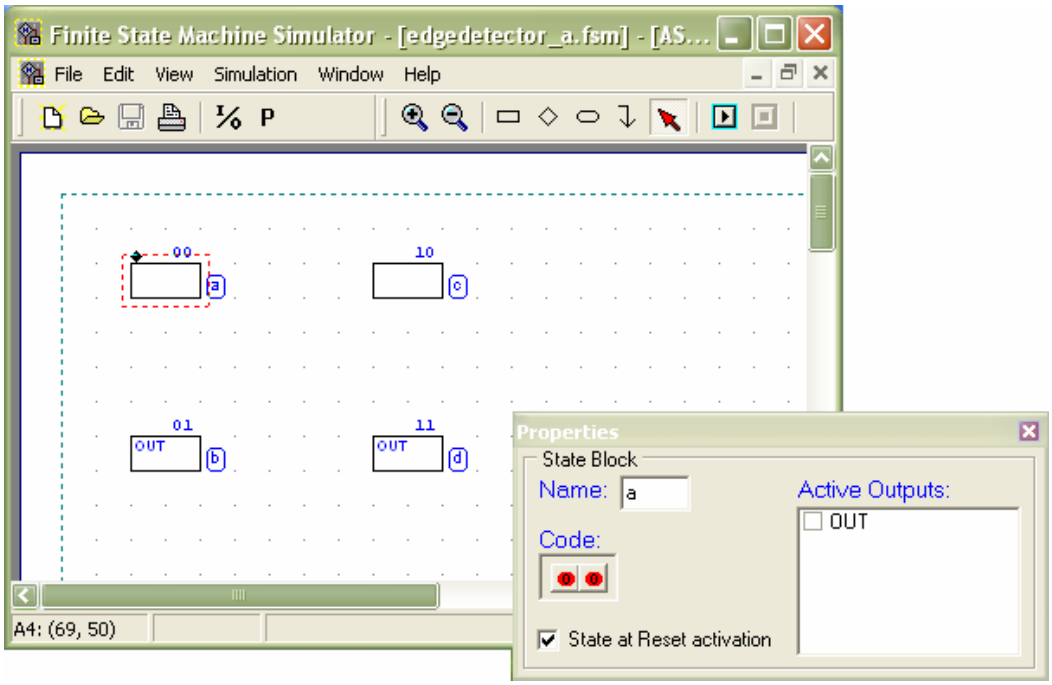


Fig. 44a: The student inserts state blocks, setting their properties.

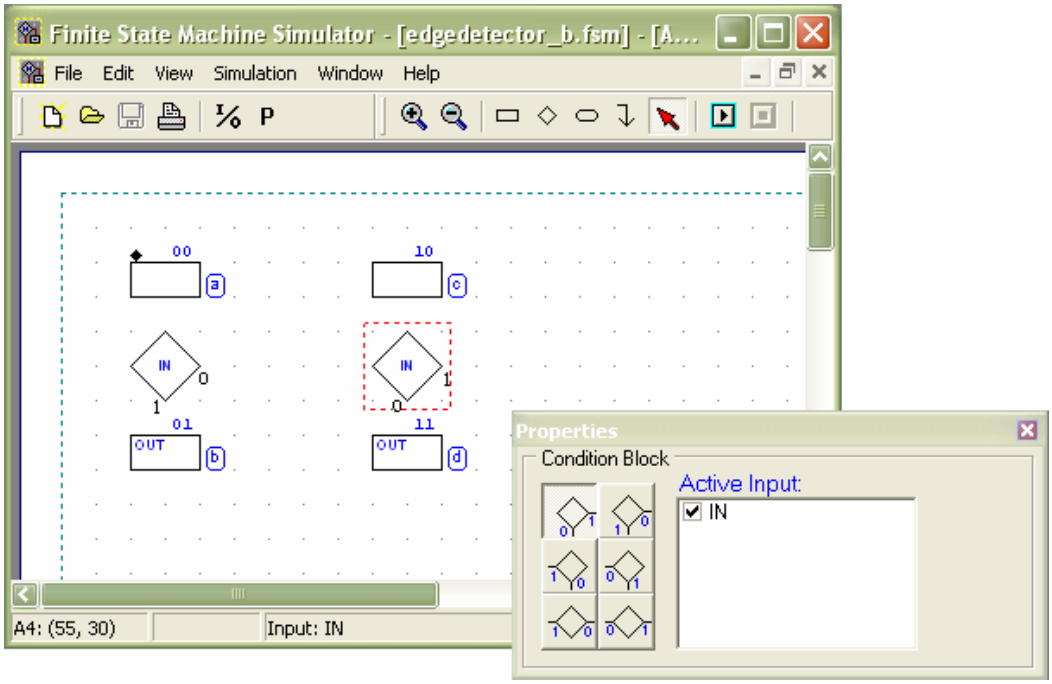


Fig. 44b: The student inserts conditional blocks, setting their properties.

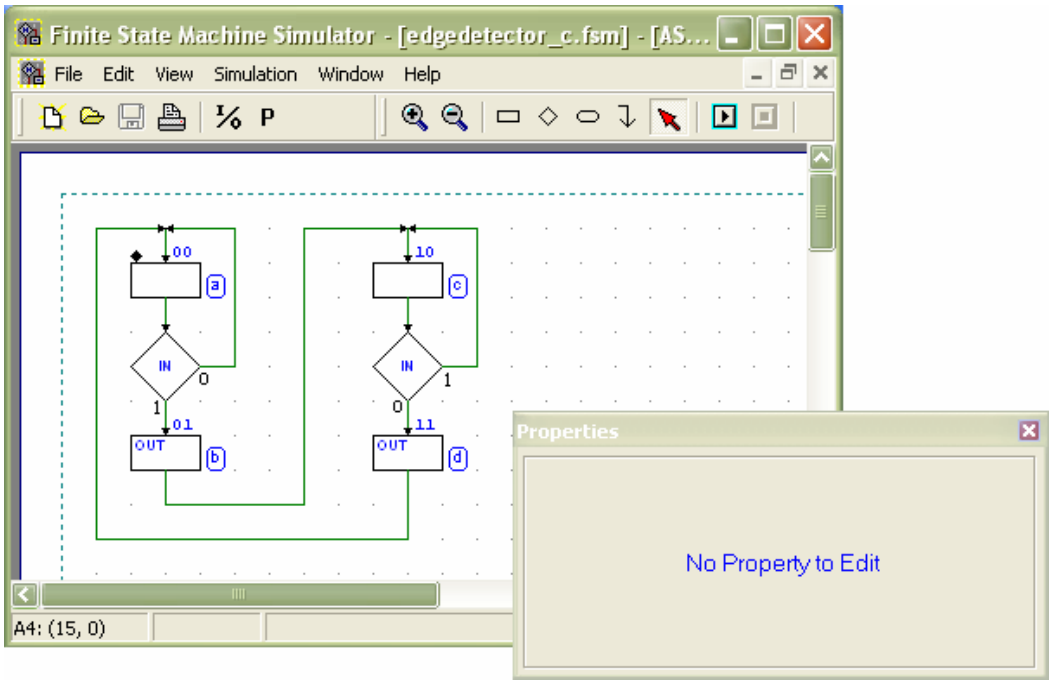


Fig. 44c: The student inserts logical path (the green lines, no property needs to be set). Note that the line arrows are automatically added.

The diagram describes an edge detector. Each time the input 'IN' presents a transition ('0' to '1', or '1' to '0'), an output pulse, of the duration of one clock cycle, is generated.

To verify its behaviour, it is possible simulate it with the d-FsM. The timing simulation of the d-FsM is only functional: it don't take in count component delays, for instance, because it simulates directly the algorithm, without synthesize the network in term of gates and flip-flops. Fig. 45 shows the results of the simulation. As expected, the output line OUT goes 'high' for one clock cycle each time the input line IN presents a level transition.

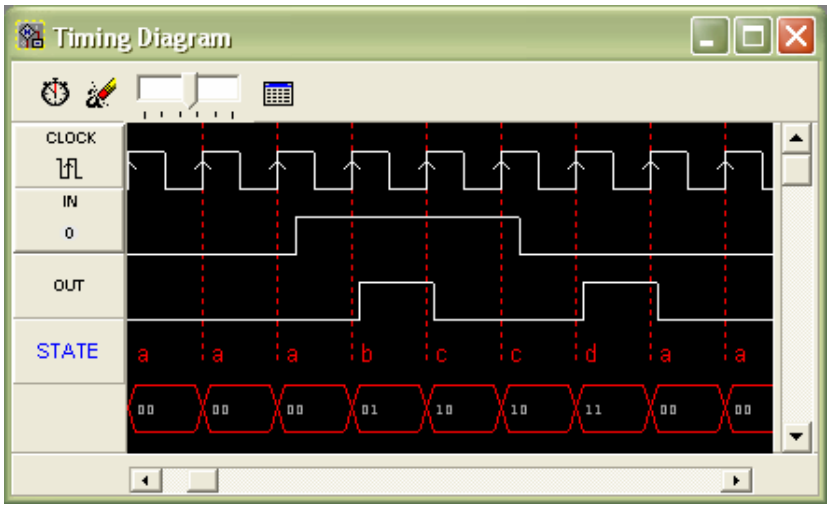
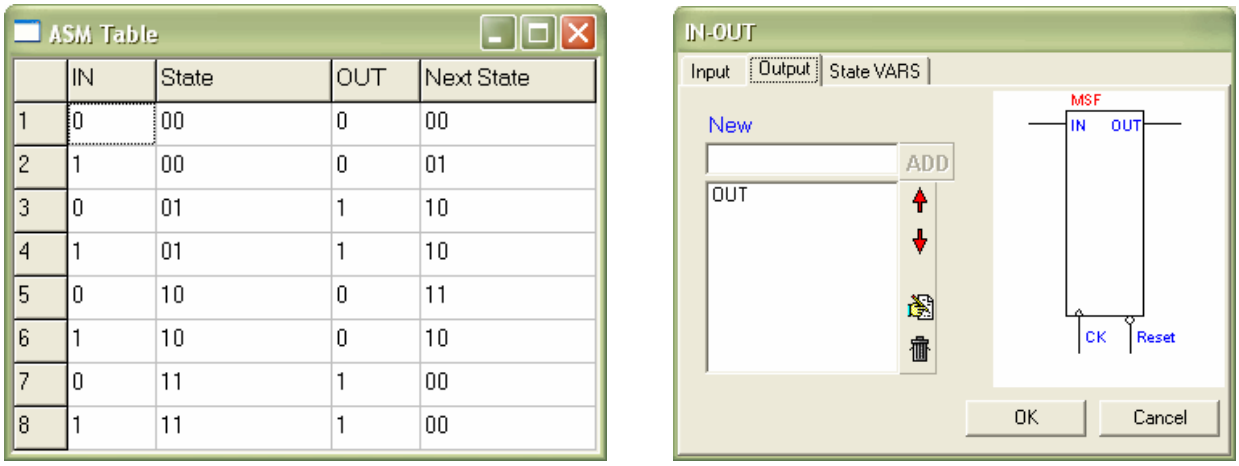


Fig. 45: The simulation results for the edge detector described above.

In Fig. 46 is reported the ASM transition table describing the designed FSM, as well as the preview of the automatically generated symbol of the new component.



	IN	State	OUT	Next State
1	0	00	0	00
2	1	00	0	01
3	0	01	1	10
4	1	01	1	10
5	0	10	0	11
6	1	10	0	10
7	0	11	1	00
8	1	11	1	00

Fig. 46: The ASM transition table describing the component, on the left, and the generated symbol, on the right.

Now the component is ready to be imported in the d-DcS. We insert the component in a very simple way, loading it from file. An example of use of this component in the d-DcS is shown in the Fig. 47, where two instances of it are connected in a circuit composed also of standard gates.

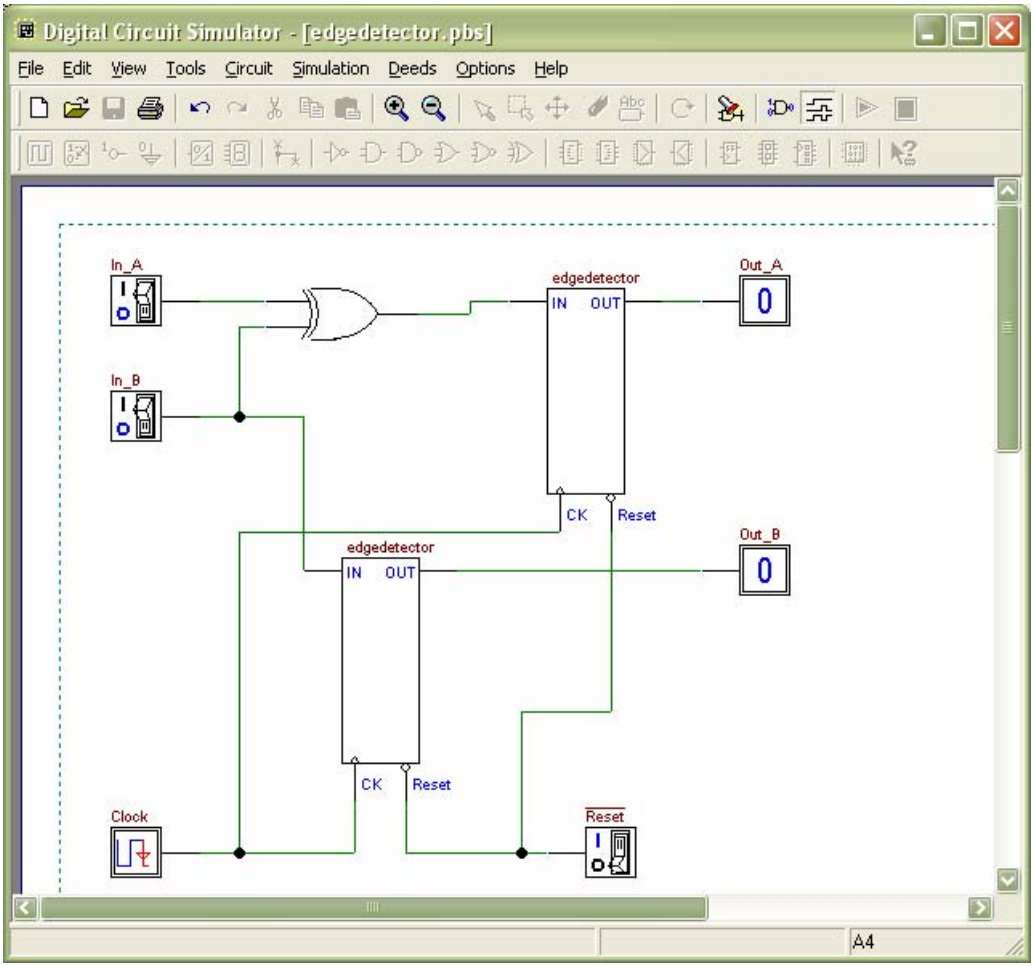


Fig. 47: Two instances of the component are connected in a circuit composed of standard gates, in the d-DcS.

Then the student could verify the correct behaviour of the network under test, comparing d-DcS simulation results with those expected, in particular with the functional simulation produced by the d-Fsm. In Fig. 48, you see a screen shot of the timing simulation obtained with the d-DcS.

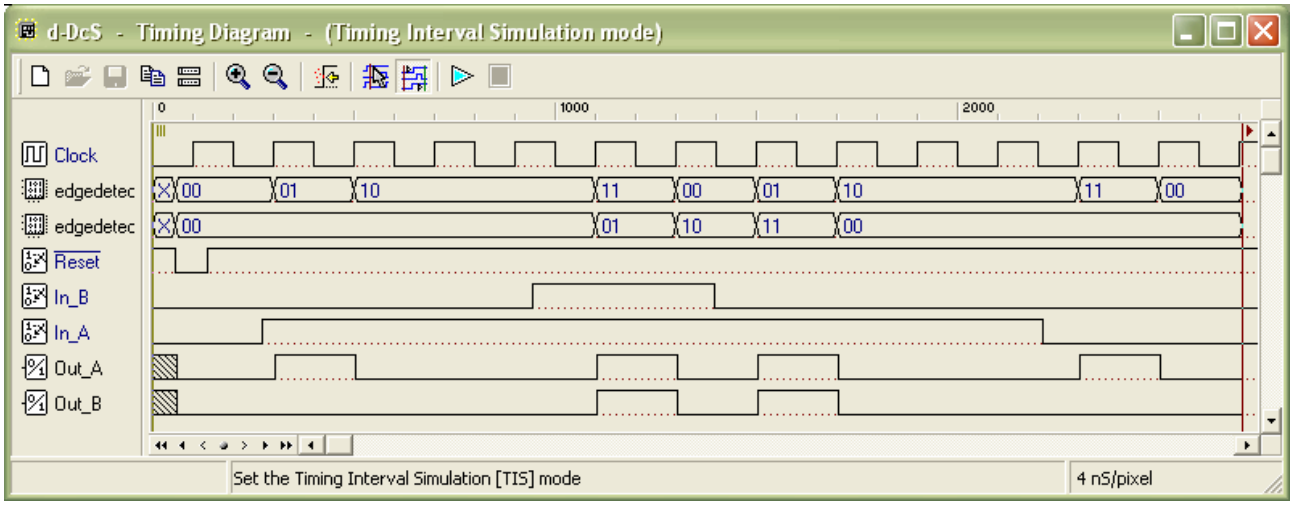


Fig. 48: Timing simulation of the previous network, obtained with the d-DcS.

This image from the Tapestry of Bayeux, Bayeux Cathedral, France

A simple example of interaction between Deeds browsers and d-FsM

As in the example applied to the Deeds with the d-DcS, in Fig. 49 a list of laboratory assignments is opened in the Deeds main browser.

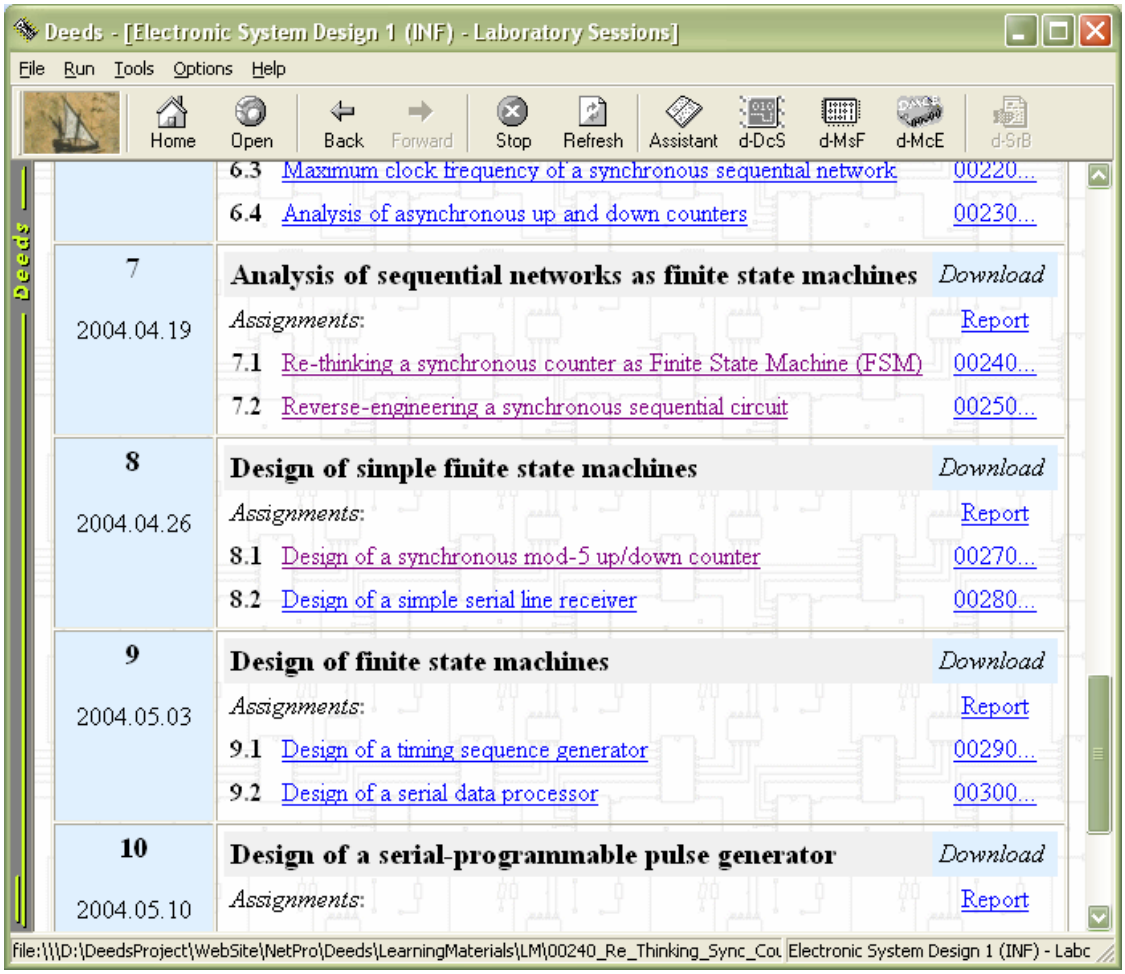


Fig. 49: A list of laboratory assignments, with use of d-FsM, opened in the Deeds main browser.

The student executes the assignment # 8.1: *"Design of a synchronous mod-5 up/down counter"*. As in the example related to the d-DcS, with a click of the user on the link, the specific assignment will be opened in the Assistant (Fig. 50a and 50b).

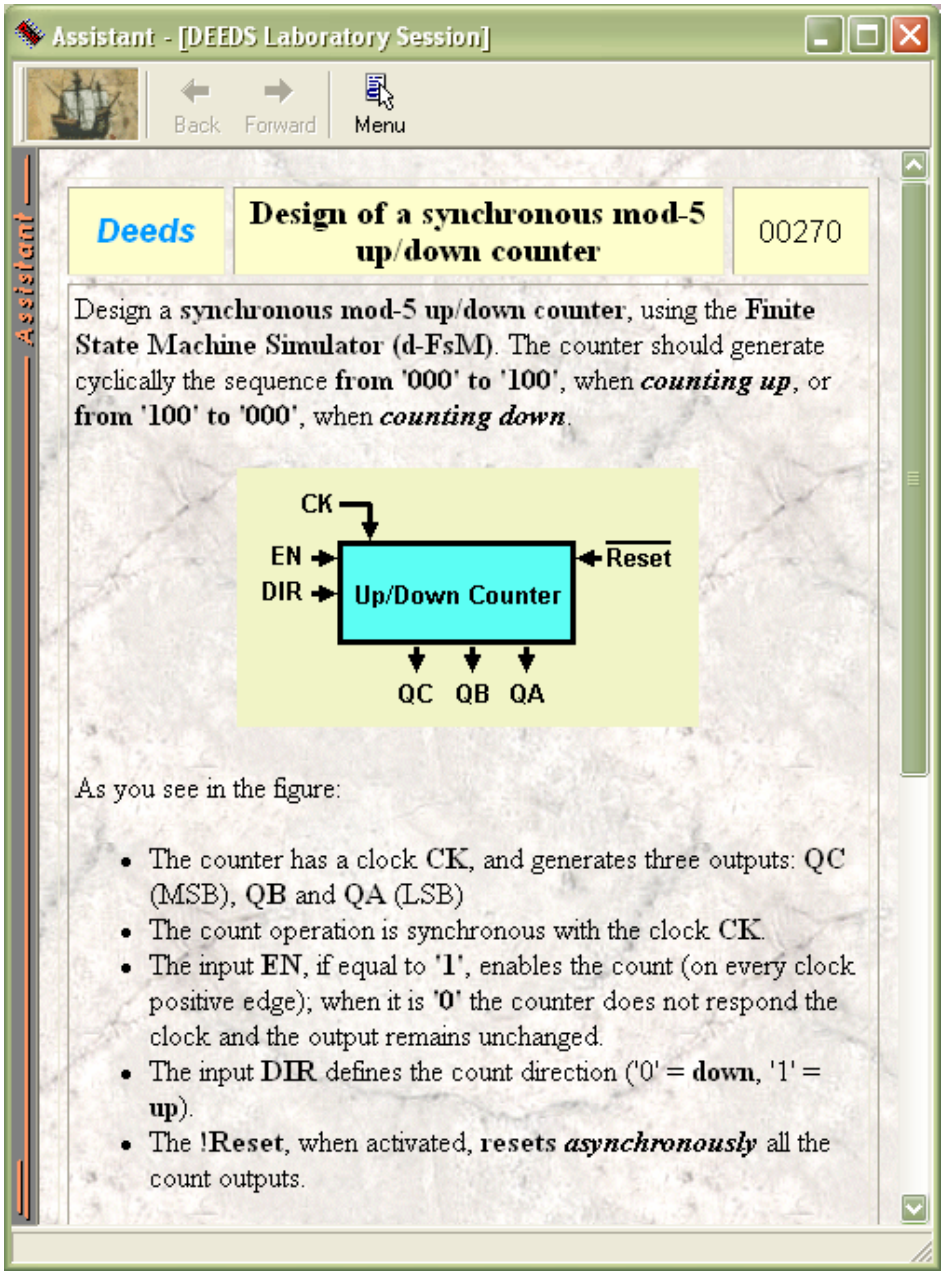


Fig. 50a: The specific laboratory assignment, opened in the Assistant browser (first page).

The assignment asks the user to design a synchronous mod-5 up/down counter, using the Finite State Machine Simulator.

In the laboratory assignment (Fig. 50a) is explained that the counter should generate a numerical sequence on the outputs QC, QB and QA, depending from the line input EN and DIR. The counter is synchronous with the clock CK and it is initialized by an asynchronous Reset input. In particular, the input DIR defines the count direction (up or down), and the input EN enables the count operation, that will take place on every clock positive edge.

In Fig. 50b, the assignment continues with a suggestion: to download an ASM diagram template, to be guided toward the solution. If the student use this option, he or she could concentrate better on the argument, instead of build from scratch the solution, bothering with the simulator details and spending time in less useful and distracting tasks. The option is not mandatory, however, and the student can freely activate the simulator without using the template.

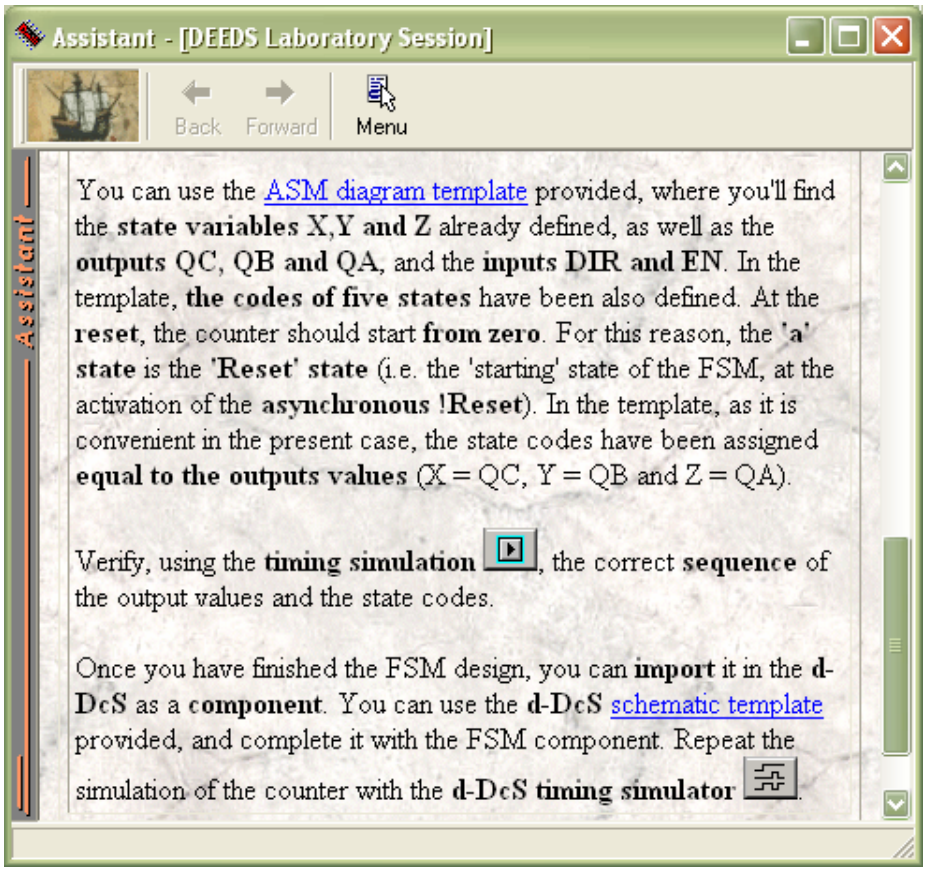


Fig. 50b: The specific laboratory assignment, opened in the Assistant browser (second page).

To download the template, it is necessary only a simple click on the link in the text. The d-FsM will be activated, and the file downloaded from the web site, automatically. In Fig. 51 you see the suggested template, as downloaded in the simulator.

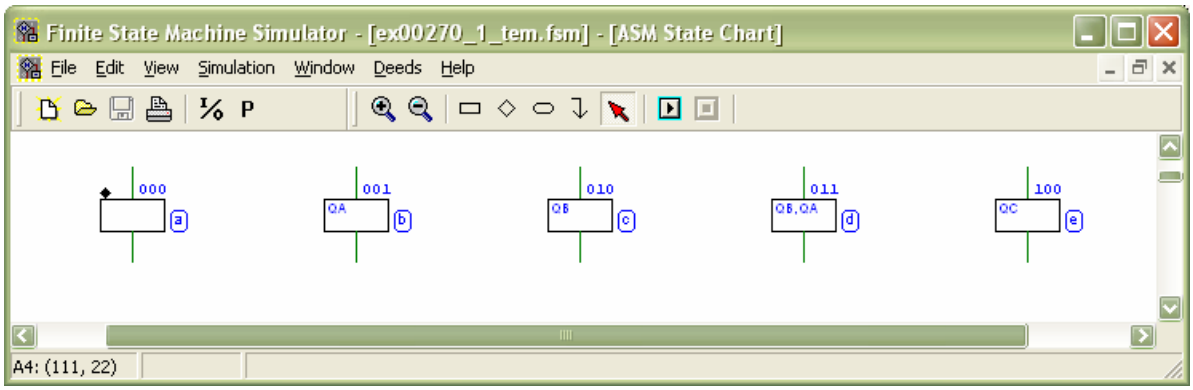


Fig. 51: The downloaded ASM diagram, template of the solution.

In the template, as the text of the assignment explains, the student will find some important definition already set: the state variables X,Y,Z, the outputs QC, QB, QA and the inputs DIR and EN. The necessary five state blocks are already drawn.

In Fig. 52a,b,c are displayed the pre-defined properties, as they appear in the Input/Output dialog windows, that the user activates with the tool bar command  .

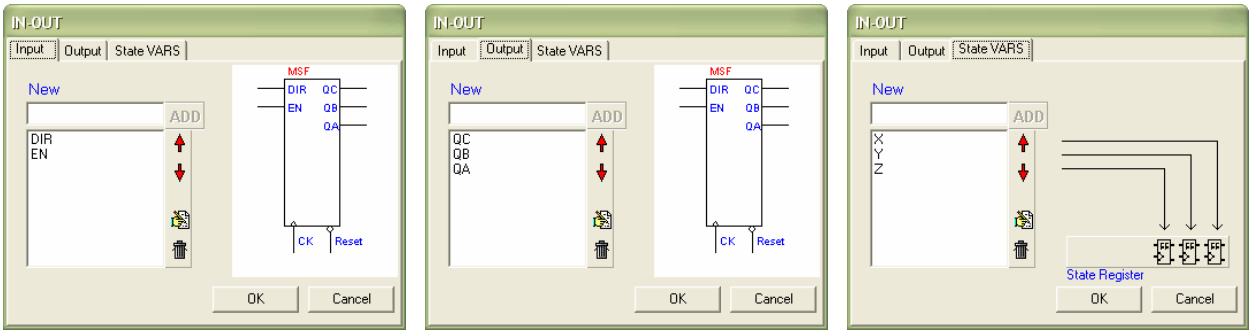


Fig. 52a,b,c: The three pages of the Input/Output dialog window, used to define inputs, outputs and state variables .

Note that the specification requires that the 'a' state will be the 'Reset' state, i.e. the 'starting' state of the component at the activation of the asynchronous !Reset. Also this characteristic has been pre-defined in the template, as the ‘a’ state appears in the drawing with a *little diamond* placed on it.

Actually, all the states properties have been pre-defined in the template. The user can modify this properties opening the Property Window. This can be left aside to the editor, during the operations (to open it, press the tool bar button ). In Fig. 53 you see the Property Window, as it appears when the user select the ‘a’ state block (with a mouse click on it).

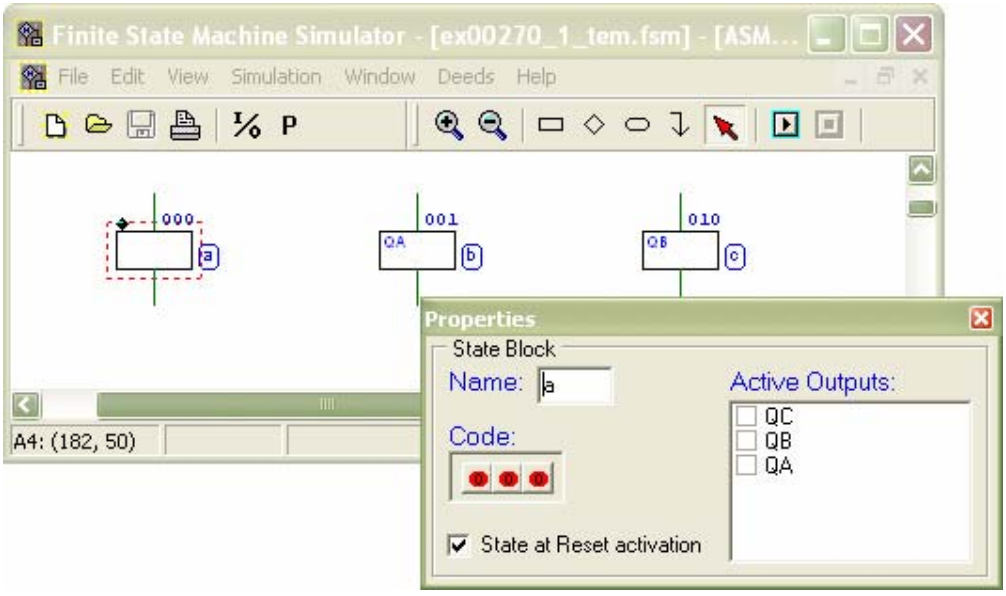


Fig. 53: The property window, displaying the properties of the ‘a’ state.

For a state block, the user can set or change the symbolic name (‘a’ in the present case), the state code (‘000’, here), and the active outputs (none, in the example). The check box on the left imposes this one as ‘Reset State’.

The user is asked to complete the ASM diagram and, using the timing simulation integrated in the d-FsM, to verify the correct sequence of output values and state codes. The user will start drawing, adding path lines and diamonds, as required by the requested functionality.

In Fig. 54 you see the Property Window, as it appears when the user select a condition block. The user can change the orientation of the diamond connections and the condition, chosen among the input variables ('DIR' in this example).

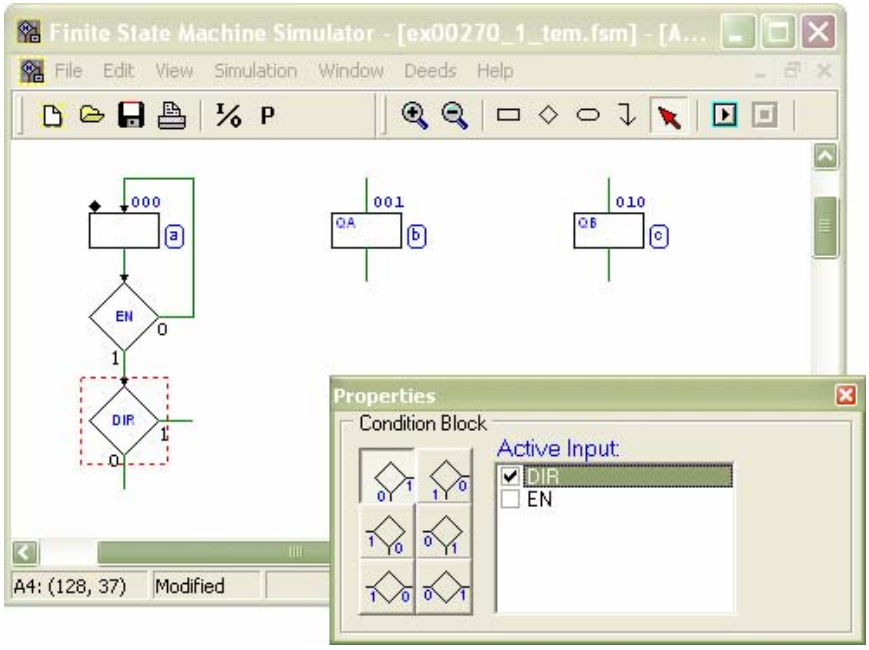


Fig. 54: The property window, displaying the properties of a condition block.

Once the student have finished the design, the next step required is to verify the behaviour of the counter with the timing simulator of the d-FsM itself (Fig. 55).

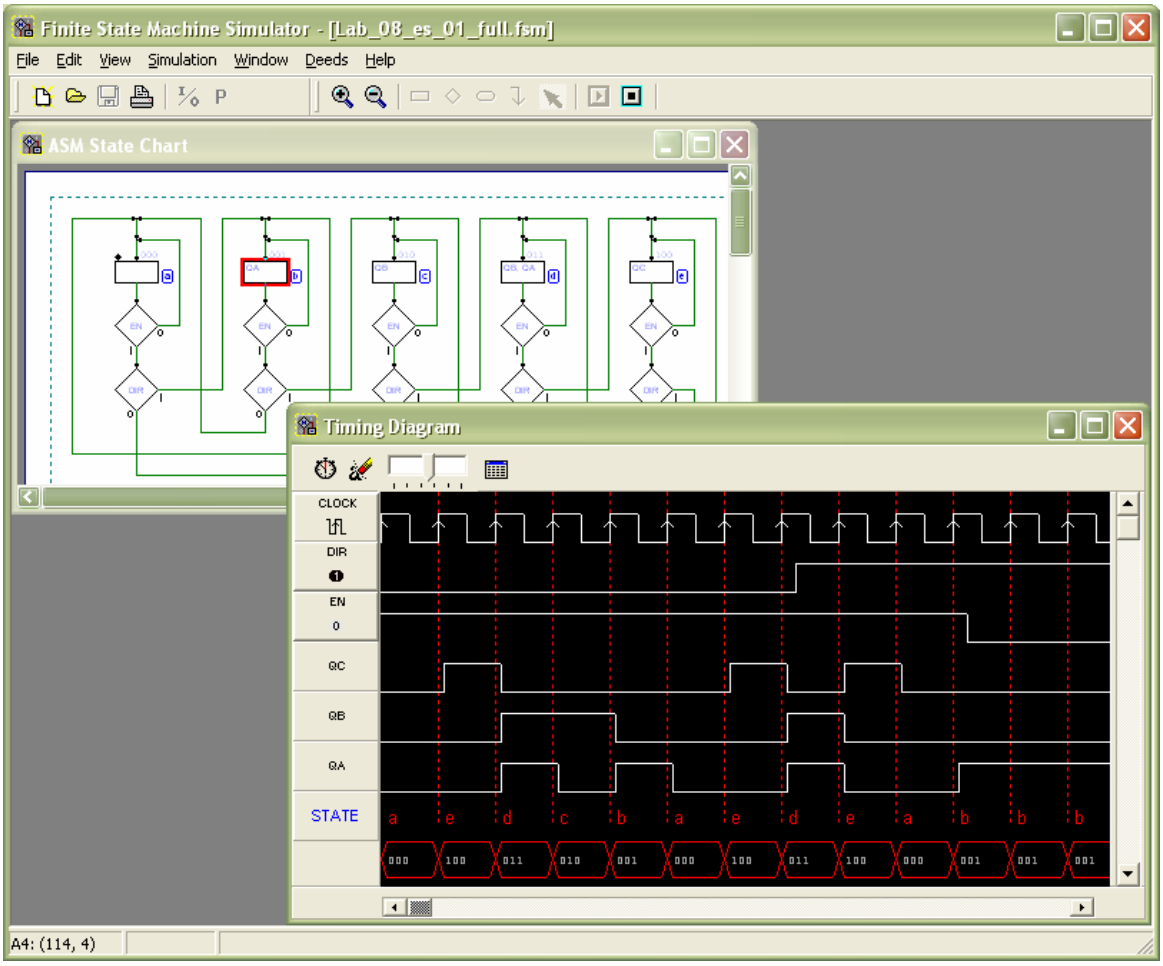


Fig. 55: The finished ASM diagram, and its timing simulation, in the d-FsM.

When the user clicks on the ‘Clock’ button, the internal simulator evaluates next state and outputs (according to the current input values) and displays the results on the time diagram.

At the same time, in the editor window, the corresponding new state is *highlighted* (with a coloured frame around it, see Fig. 55). This is an important feature, because a major difficulty, for a beginner, is to understand the correspondence between states and events time sequence.

Finally, when the behaviour of the component satisfies all the required specifications, the component could be imported in the d-DcS (see the assignment, Fig. 50b). Also in this case, a simple d-DcS schematic template is provided, to speed up the operations; it can be easy downloaded and opened in the d-DcS with a click on the hyperlink in the text. Once completed the schematic, the simulation of the counter could be repeated in the d-DcS timing simulator (Fig. 56).

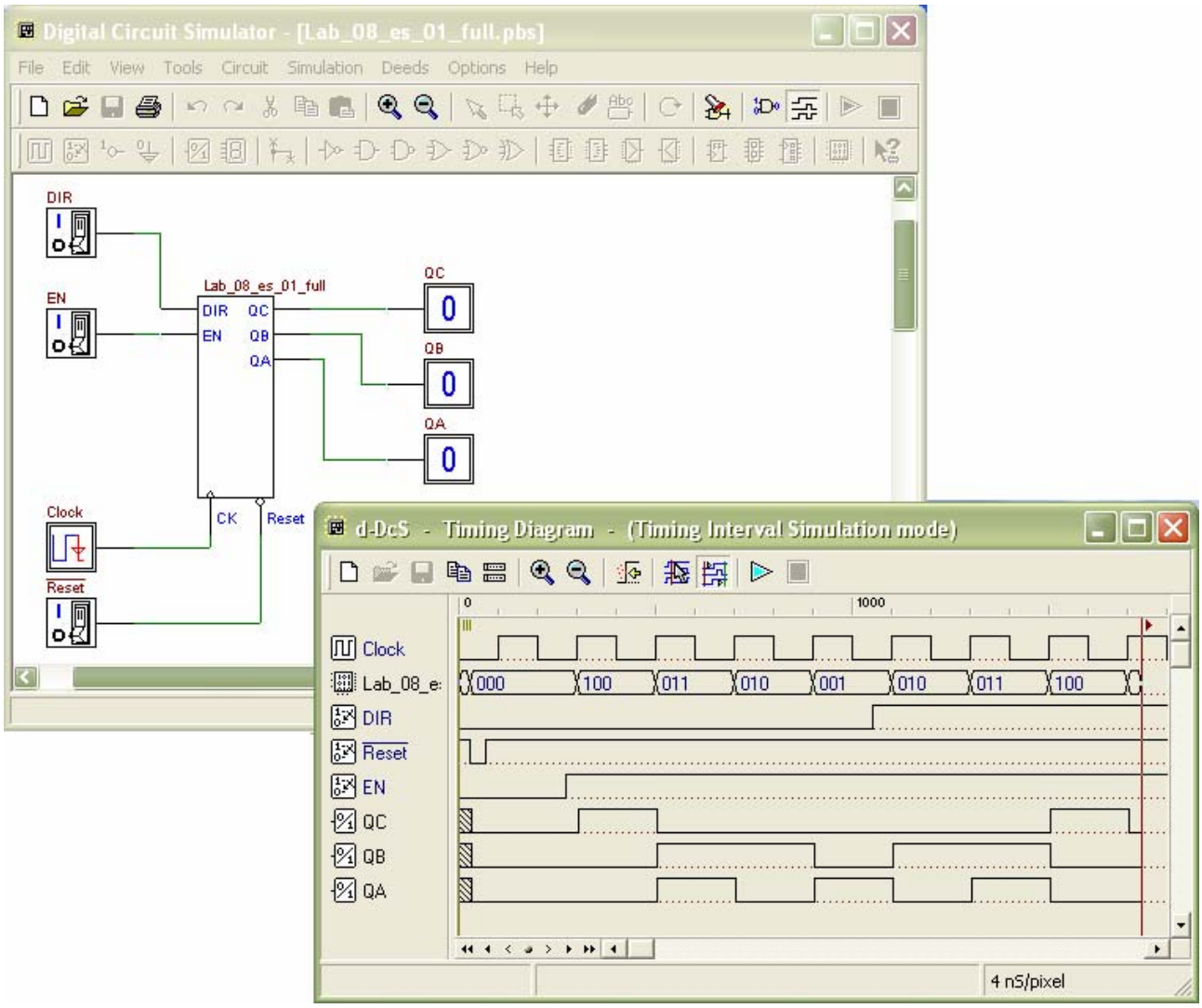


Fig. 56: The finished d-DcS schematic, and the timing simulation of the component, in the d-DcS.

As in the example related to the d-DcS, at this point the student will compile and deliver a report about its work. As already seen, in the assignments page, a link is set to download a report template file (Fig. 57).

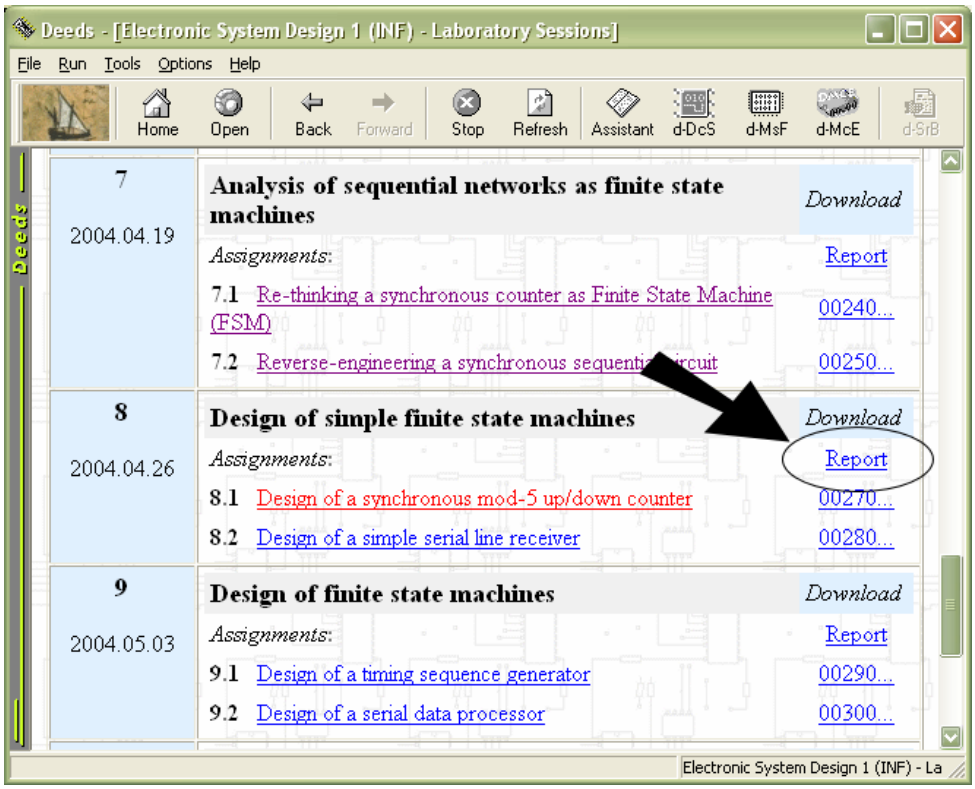


Fig. 57: Also in this case, the student will download the report template to speed up its compilation and delivering.

In Fig. 58 is displayed the report template prepared for this laboratory assignment, downloaded and ready to be edited.

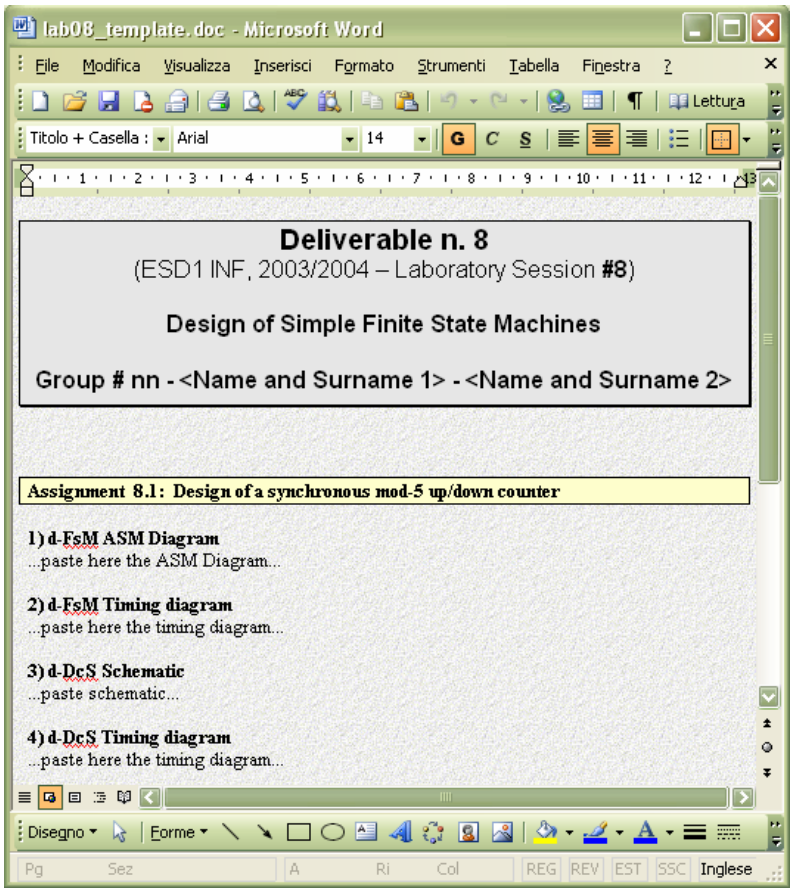


Fig. 58: The report template for this laboratory assignment assignment.

The timing diagram window

In this window (Fig. 59) the timing diagram of all the signals is constructed, during the simulation, in a interactive mode. The timing diagram displays the Input and Output signals and, at bottom, the current State, by symbolic name and by code.

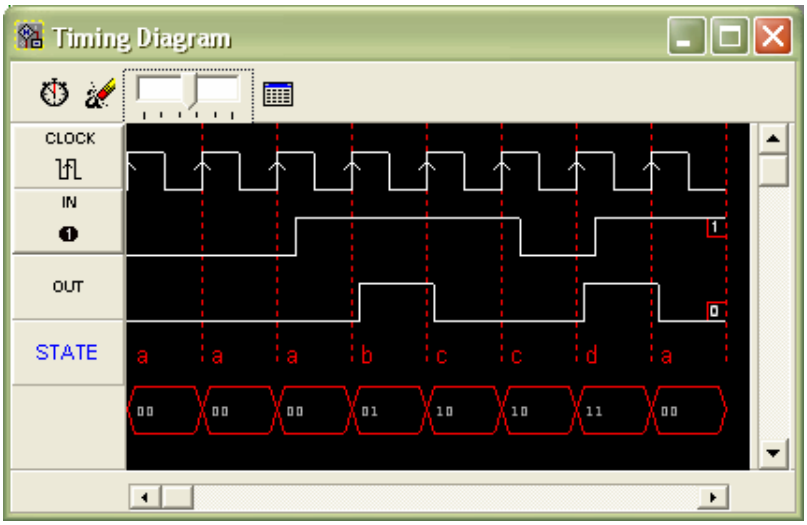
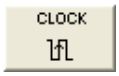


Fig. 59: The Timing Diagram window of the d-FsM.

In the default mode, the user clicks on the 'Clock' button  to advance the simulation by one step (a clock cycle).

If the tool bar button  is checked, the simulation step is automated (the execution speed is controlled by the tool bar cursor ).

To toggle Input signal values, the user clicks on the Input signal buttons, under the 'Clock' button. In the example of Fig. 59, the button is  .

The button  restarts simulation (from time = 0).

The button  activates the ASM Table window (Fig. 60). In this table Inputs, Outputs and current and next States are expressed in a compact, tabular form.

ASM Table

	IN	State	OUT	Next State
1	0	00	0	00
2	1	00	0	01
3	0	01	1	10
4	1	01	1	10
5	0	10	0	11
6	1	10	0	10
7	0	11	1	00
8	1	11	1	00

Fig. 60: The ASM Table window.

d-FsM: Menu Commands

The menu of the Finite State Machine Simulator allows the user to access all the function of the application. The ToolBars replicate most of the commands already in the menu, to speed up user operations.

File Menu

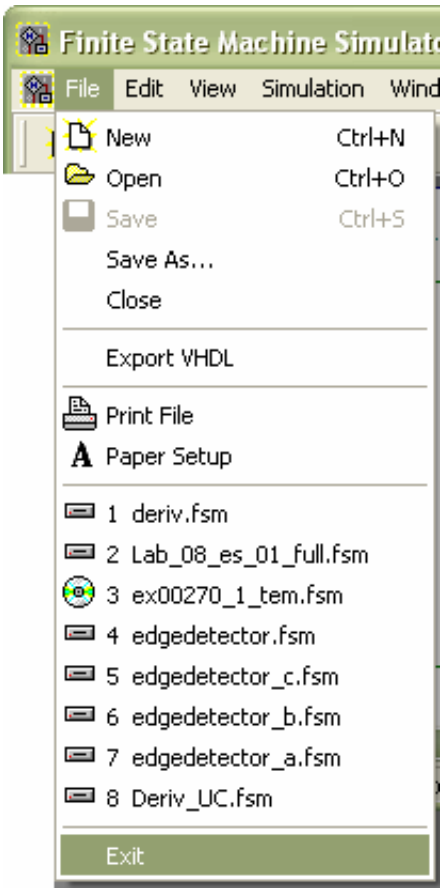


Fig. 61: The d-FsM "File" menu.

New

Command to create a new Finite State Machine file.

Open

Command to open a Finite State Machine file. The file can be also downloaded directly from a web site.

Save

Command to save current Finite State Machine file.

Save as

Command to save current Finite State Machine file with a different name or in a different position.

Close

Command to close the current Finite State Machine.

Export VHDL

Command to export the Finite State Machine ASM diagram in VHDL language. It shows a window with the equivalent VHDL code, generated from the internal data base (Fig. 62).

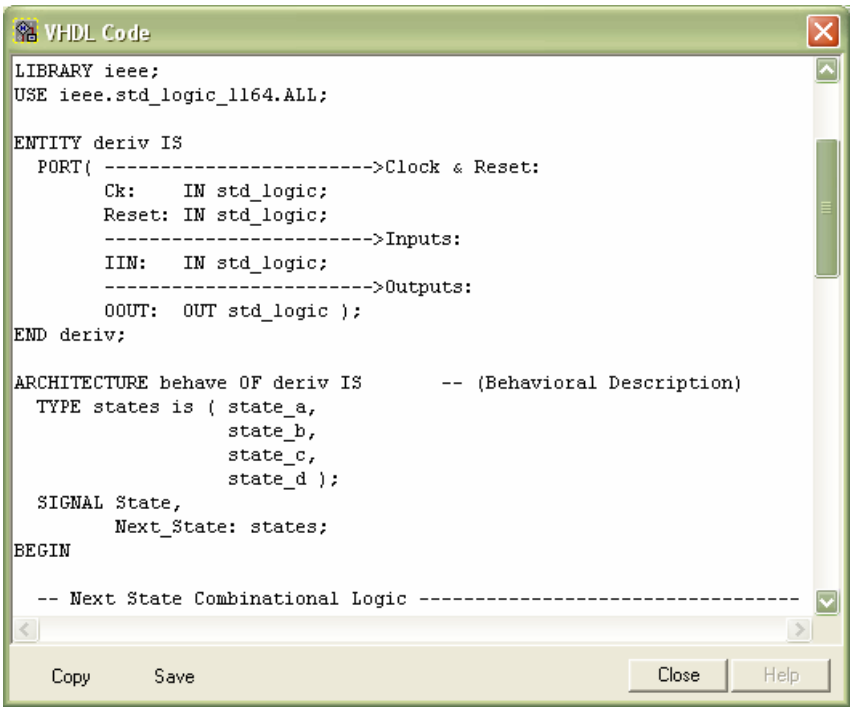


Fig. 62: The VHDL code window.

If you wish to save the generated code in a file, click on the 'Save' button: you will prompted to chose a name file, before to save it. If you want include the VHDL code in another text file, click on the 'Copy' button to pass all the VDHL code onto the 'clipboard', ready to be pasted in a code editor of your choice.

Print

Command to print the Finite State Machine ASM diagram.

Paper Setup

Command to define current paper format and orientation. It displays the Paper Setup dialog window (Fig. 63).

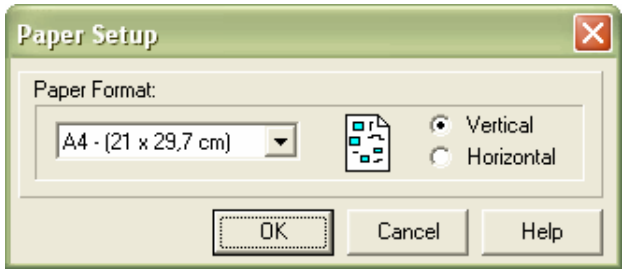


Fig. 63: The Paper Setup dialog window.

Recent Files List

Commands to re-open the most recent files. Up to 8 recent files can be reopened with this list. The symbol that is displayed on the left of the file name means that:

	The file has been stored by the user on the local disk or network.
	The file has been downloaded from a web site, but it has not been saved (yet) on the local disk or network.
	The file has been loaded from a local courseware, where it is read only and it has not been saved (yet) on the local disk or network.

Exit

Standard command to close the application.

Edit Menu

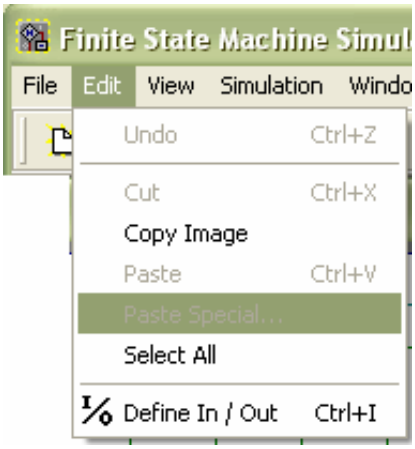


Fig. 64: The d-FsM "Edit" menu.

Undo

Command to undo the previous operation (command temporary inhibited).

Cut

Command to cut the selected part of the ASM diagram, and copy it on the clipboard (command temporary inhibited).

Copy Image

Command to copy the selection as a bitmap image and put it on the Clipboard.

Paste

Command to paste the clipboard content in the circuit (command temporary inhibited).

Select All

Command to select all the object of the drawing.

Define In / Out

Command to define or modify Inputs, Outputs and State Variables. It activates a modal dialog window (see Fig. 65), where the user can add, rename and delete the Input and Output lines (up to 8 lines), as well as the State Variables (up to 6). The dialog is divided in the specialized pages (Input, Output and State Vars).

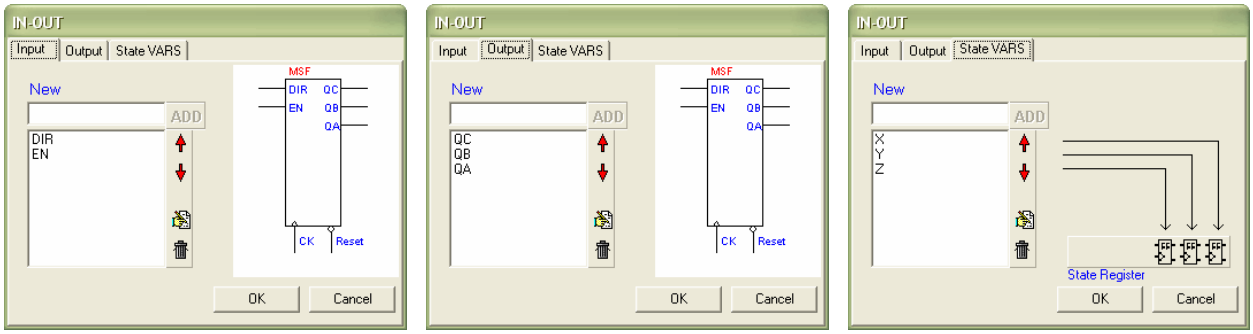


Fig. 65: The three pages of the Input/Output dialog window, used to define inputs, outputs and state variables .

View Menu

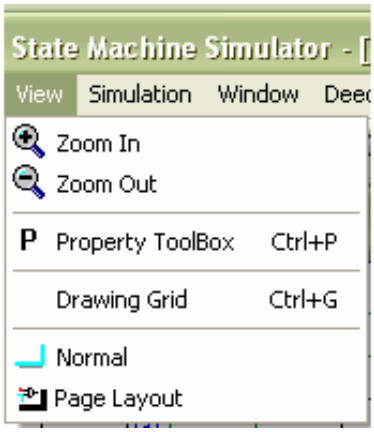


Fig. 66: The d-FsM "View" menu.

Zoom In, Out

Command to "zoom in" or "zoom out" the drawing.

Property ToolBox

Command to activate the "Property Window", that enables the user to set and modify the properties of the selected State Block, Conditional Block or Conditional Output Block. It shows four different "property pages", depending on the context (Fig. 67).

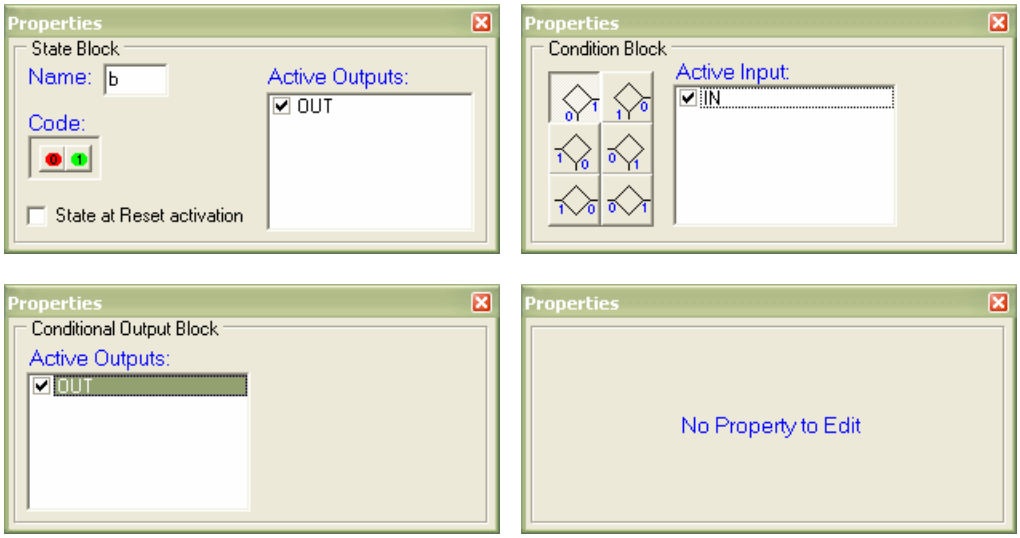


Fig. 67: The four pages of Property Window, used to define properties of state, conditional and conditional output blocks.

Normal

Command to set the "normal view" of drawing space (i.e. as uniform continuous background, only with the indication of drawing margins).

Page Layout

Command to set the view of the drawing space as a paper foil (i.e. with visible foil borders and shadows, together with drawing margins).

Tools Menu

Fig. 68: The d-FsM "Simulation" menu.

Start Simulation

Command to start the functional simulation of the finite state machine represented by the currently ASM diagram. During simulation, the editor commands are inhibited, and the "Timing Diagram" window is displayed (Fig. 59).

Stop Simulation

Command to stop simulation and return to the edit mode of the ASM diagram.
Four commands to rotate an object (during its insertion) to the specified direction.

Window Menu

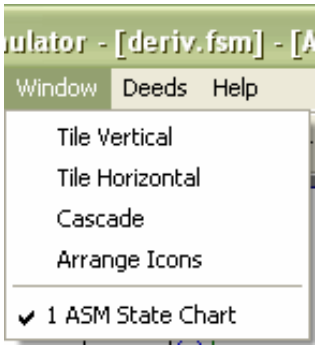


Fig. 69: The d-FsM "Window" menu.

Tile Vertical

Command to tile vertically the opened windows (the graphical editor, the timing diagram, the ASM table).

Tile Horizontal

Command to tile horizontally the opened windows (as above).

Cascade

Command to cascade diagonally the opened windows (as above).

Arrange Icons

Command to reorder the icons of the iconized windows, at the bottom of the main window.

Opened windows list

Command to switch focus among the opened windows within the main window.

Deeds Menu

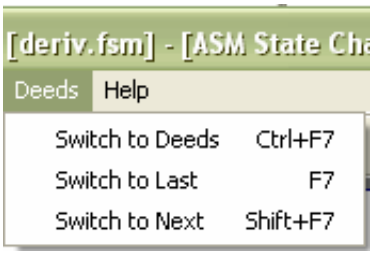


Fig. 70: The d-FsM "Deeds" menu.

Switch to Deeds

Command to switch focus to the Deeds main browser.

Switch to Last

Command to switch to the tool that was 'last on top' before switching to the currently opened instance of the d-DcS.

Switch to Next

Command to switch focus among all active Deeds applications, in order of activation.

Help Menu

Fig. 71: The d-FsM "Help" menu.

Index

Command to open the d-FsM Help System (disabled in this version).

License Agreement

Command to display the Licence Agreement.

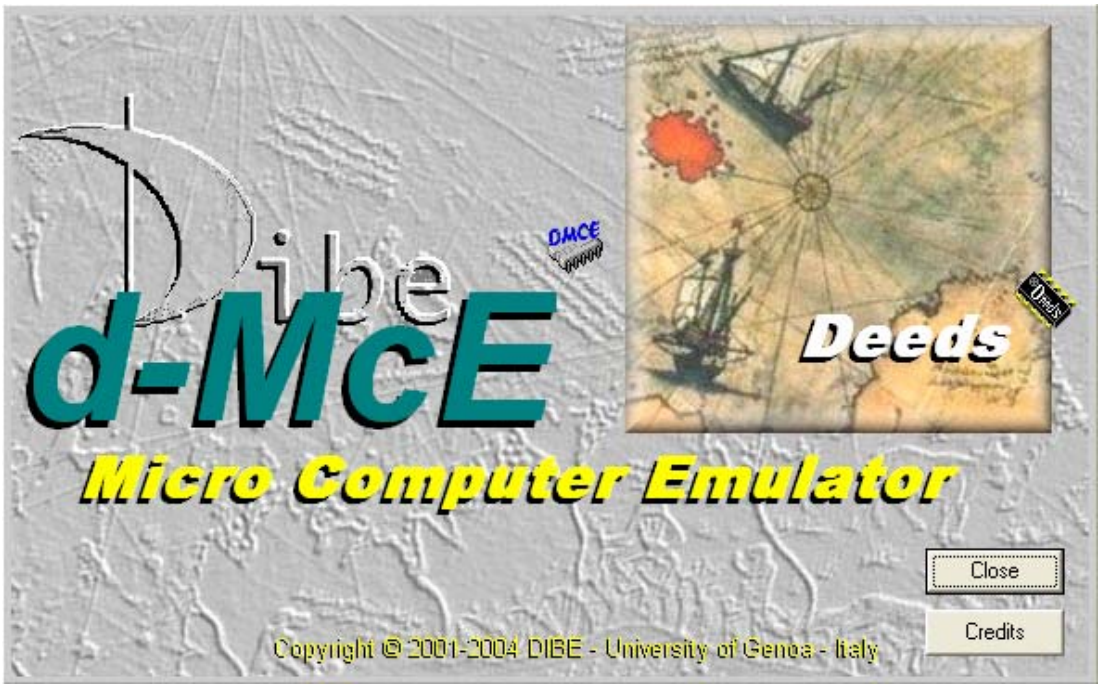
Version Notes

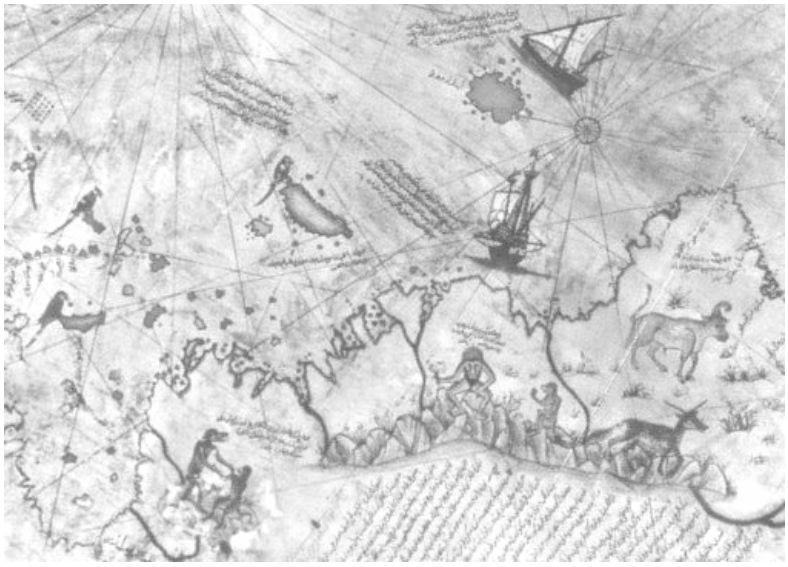
Command to display the Deeds "Version Notes" file.

About

Command to display the d-FsM 'splash' window dialog.

Deeds: The Micro Computer Emulator d-McE





This image from the ancient (and mysterious) Piri Reis map (1513)

Introduction

With the Micro Computer Emulator ***d-McE***, the user can practice programming at assembly language level (Fig. 72). It functionally emulates a board including a CPU, ROM and RAM memory, parallel I/O ports, reset circuitry and a simple interrupt logic. The custom 8 bit CPU, named DMC8, has been designed to suite our educational needs, and it is based on a simplified version of the well-known 'Z80-CPU' processor.

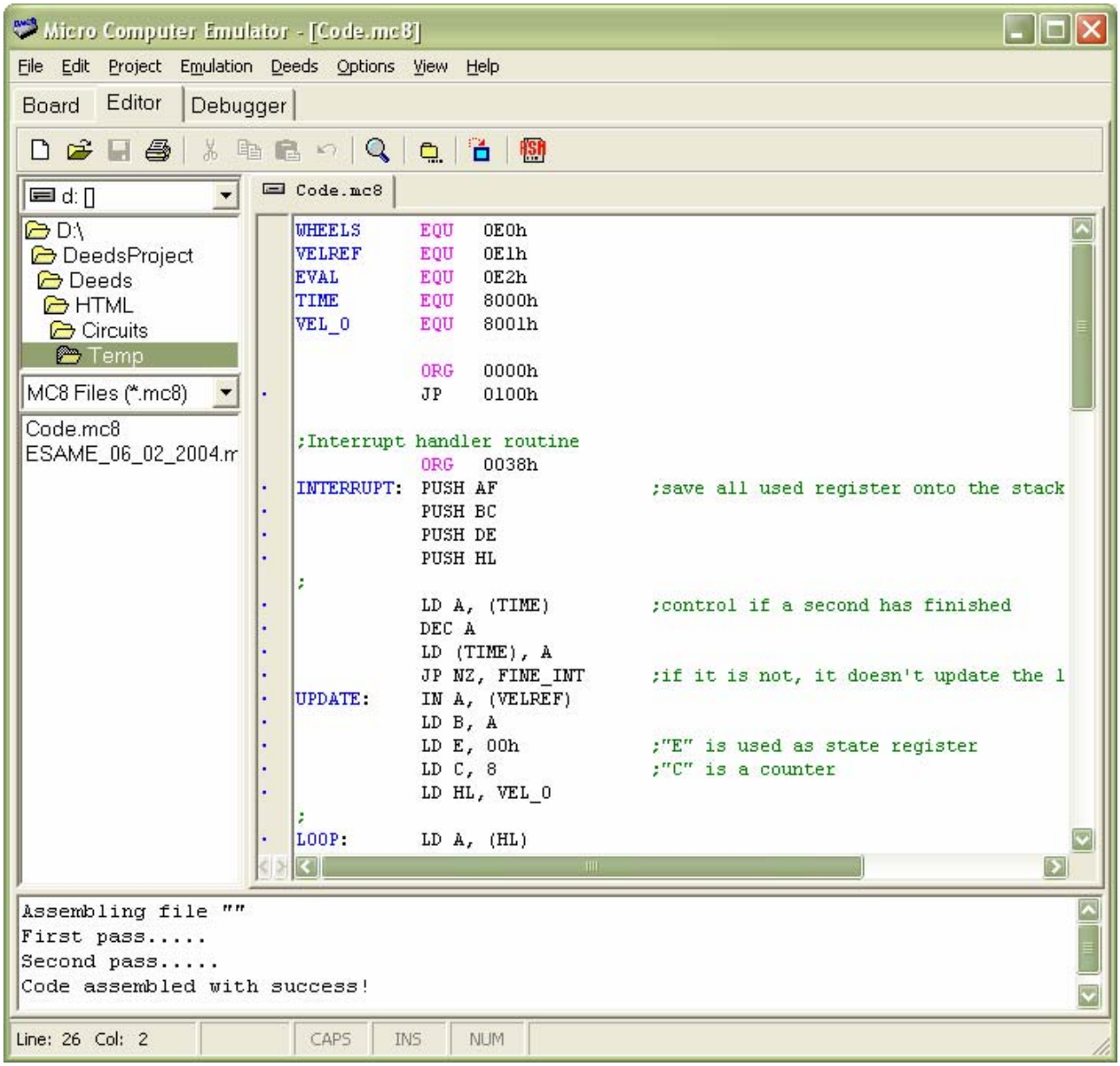


Fig. 72: The assembler code editor of the Micro Computer Emulator (d-McE).

The integrated source code editor enables user to enter assembly programs, and a simple command permits to assemble, link and load them in the emulated system memory.

The execution of the programs can be run step by step in the interactive debugger (Fig. 73). In the debugger, as in professional tools, the user can evaluate the contents of all the structures involved in the hardware / software system, by stepping the execution of the programs.

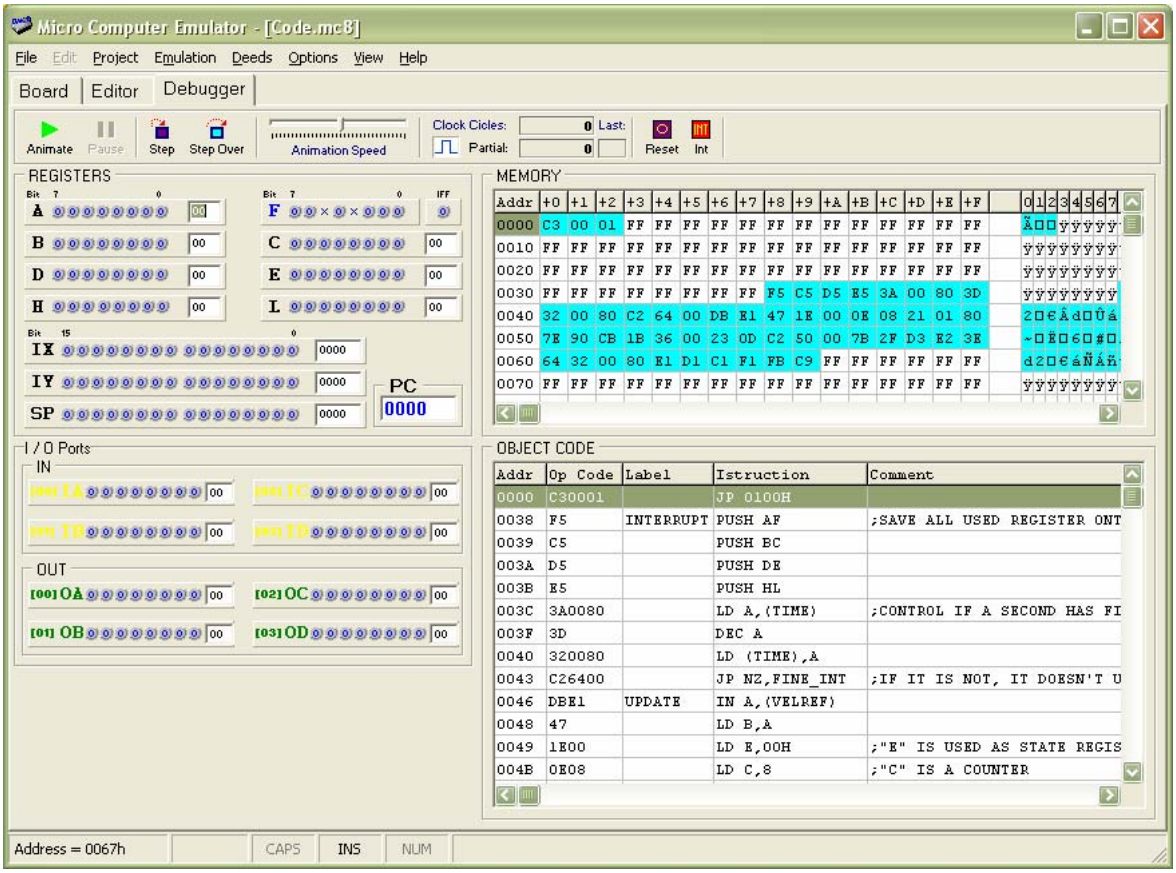


Fig. 73: The assembler-level debugger of the Micro Computer Emulator.

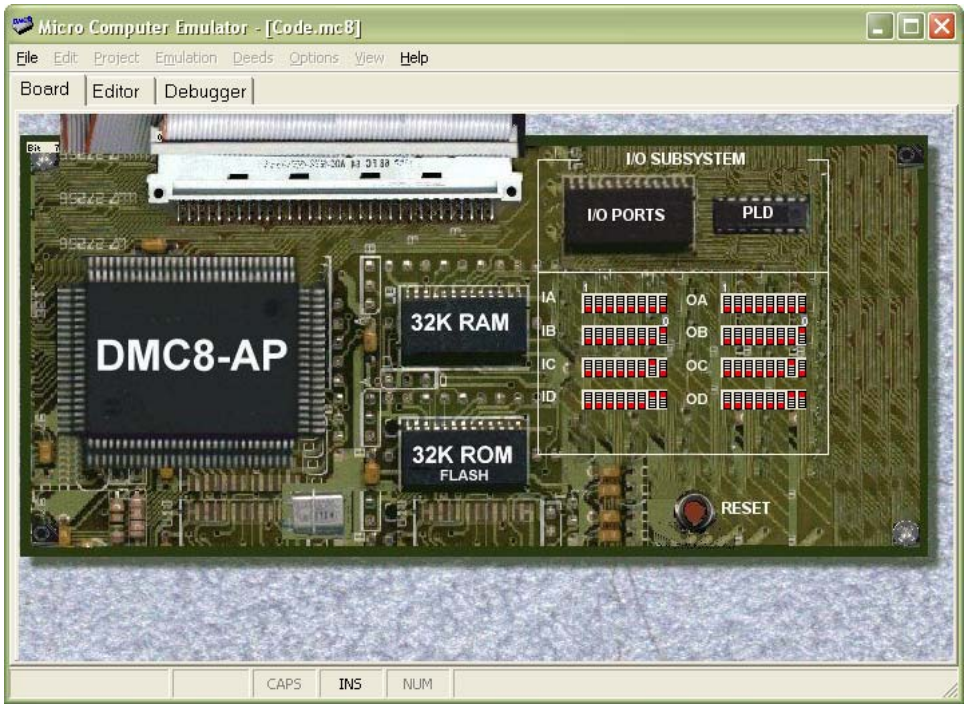


Fig. 74: The emulated board, as represented in the Micro Computer Emulator.

A simple example

In the following screen shot (Fig. 75) you can see an assembly program edited in the d-McE code editor. The code editor supports syntax highlighting. The code of the DMC8 microprocessor assembly is mainly the same of the well-known ‘Z80-CPU’ processor, but reduced of some instructions, to simplify and ‘linearize’ the instruction set.

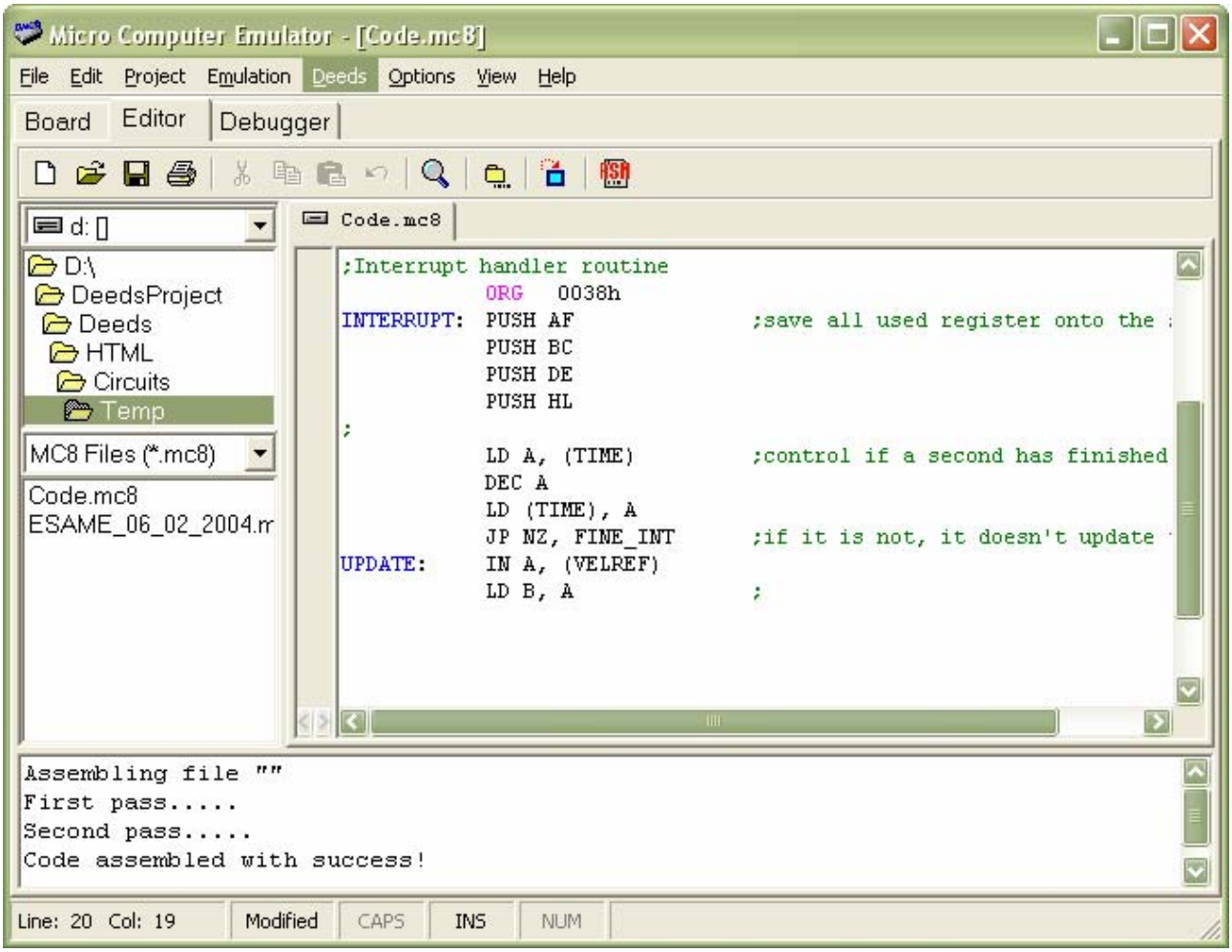


Fig. 75: The editing phase of an assembly program, in the d-McE.

The microprocessor architecture is documented in the help system. This presents topics to the user as a “multi-page” window (Fig. 76).

The instruction set is documented ‘on line’, to help the user in writing the assembly programs (examples in Fig. 77 and 78).

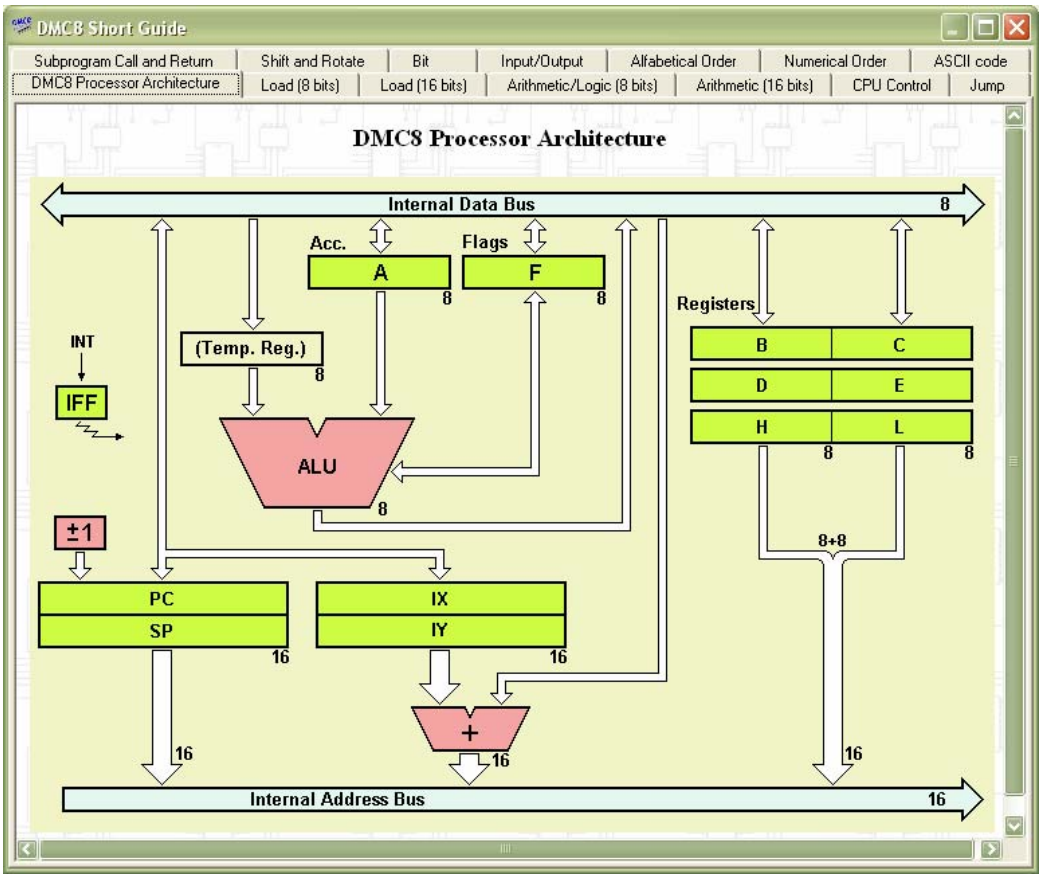
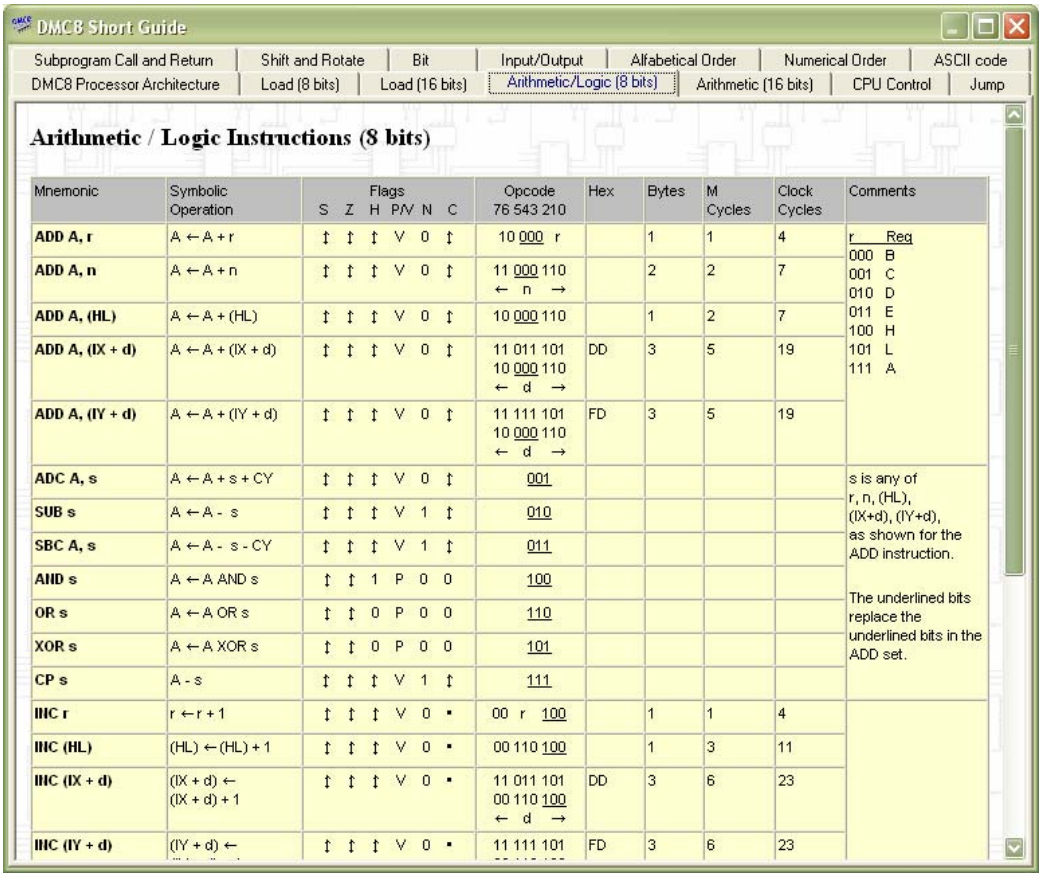


Fig. 76: The DMC8 “architecture”, as shown by the help-system.



Mnemonic	Symbolic Operation	Flags S Z H P/V N C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
ADD A, r	A ← A + r	↕ ↕ ↕ V 0 ↕	10 000 r		1	1	4	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A ← A + n	↕ ↕ ↕ V 0 ↕	11 000 110 ← n →		2	2	7	
ADD A, (HL)	A ← A + (HL)	↕ ↕ ↕ V 0 ↕	10 000 110		1	2	7	
ADD A, (IX + d)	A ← A + (IX + d)	↕ ↕ ↕ V 0 ↕	11 011 101 10 000 110 ← d →	DD	3	5	19	
ADD A, (IY + d)	A ← A + (IY + d)	↕ ↕ ↕ V 0 ↕	11 111 101 10 000 110 ← d →	FD	3	5	19	
ADC A, s	A ← A + s + CY	↕ ↕ ↕ V 0 ↕	001					s is any of r, n, (HL), (IX+d), (IY+d), as shown for the ADD instruction. The underlined bits replace the underlined bits in the ADD set.
SUB s	A ← A - s	↕ ↕ ↕ V 1 ↕	010					
SBC A, s	A ← A - s - CY	↕ ↕ ↕ V 1 ↕	011					
AND s	A ← A AND s	↕ ↕ 1 P 0 0	100					
OR s	A ← A OR s	↕ ↕ 0 P 0 0	110					
XOR s	A ← A XOR s	↕ ↕ 0 P 0 0	101					
CP s	A - s	↕ ↕ ↕ V 1 ↕	111					
INC r	r ← r + 1	↕ ↕ ↕ V 0 ▪	00 r 100		1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕ ↕ ↕ V 0 ▪	00 110 100		1	3	11	
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕ ↕ ↕ V 0 ▪	11 011 101 00 110 100 ← d →	DD	3	6	23	
INC (IY + d)	(IY + d) ←	↕ ↕ ↕ V 0 ▪	11 111 101	FD	3	6	23	

Fig. 77: An example of the ‘on line’ instruction set documentation: the Arithmetic and Logic instructions.

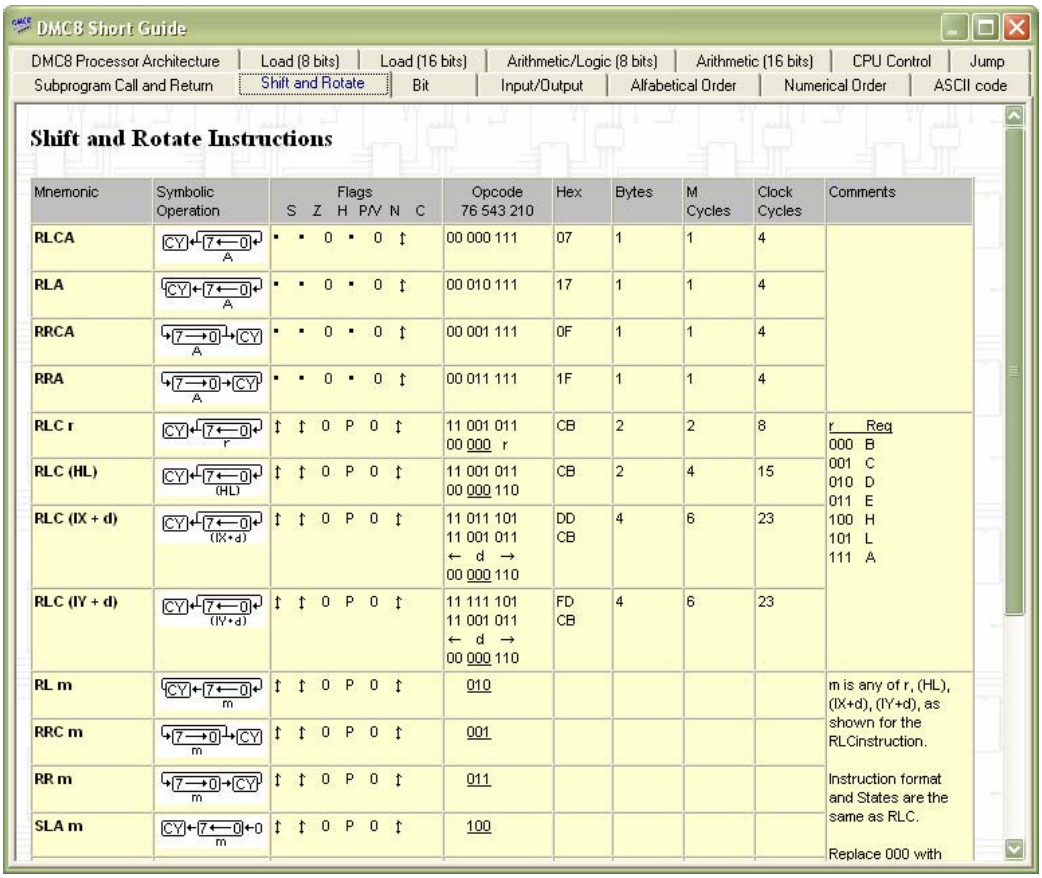


Fig. 78: Another example of the 'on line' instruction set documentation: the Shift and Rotate instructions.

When the user wishes to verify the correctness of the written code, or when the coding is finished, he or she can launch the Assembler module, using the tool bar button . In Fig. 79 an example of assembling report, in case of error, is shown (a unknown label was found, and the offending line is pointed by a little symbol).

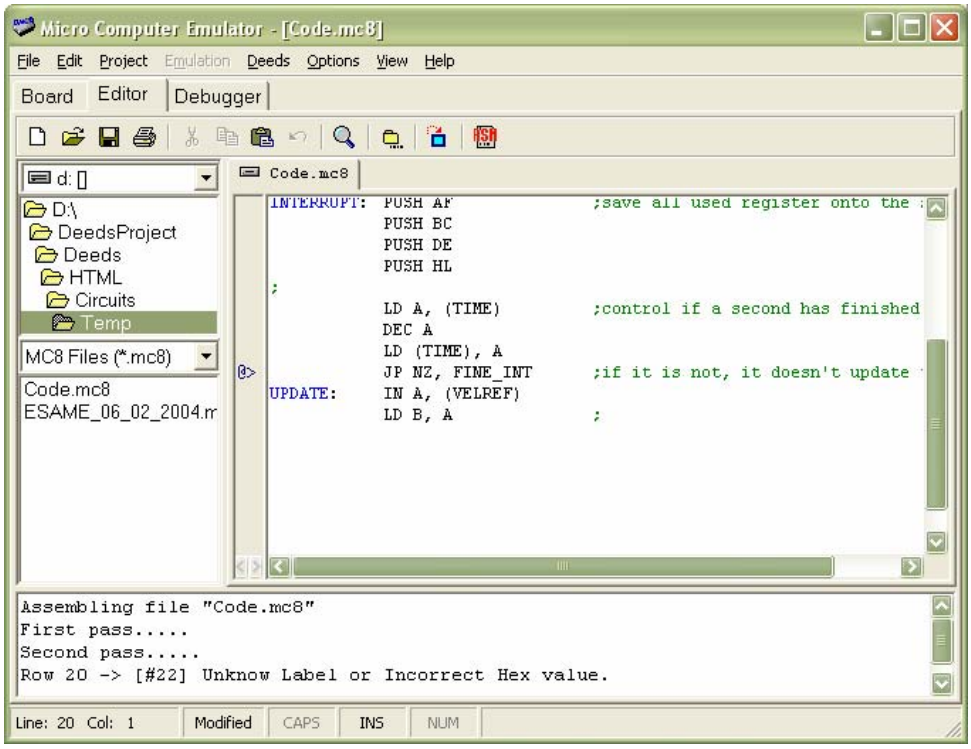


Fig. 79: The Assembler module reports an error in the source code.

When the code has been cleaned, and no syntax error is reported, the program can be tested in the debugger (Fig. 80).

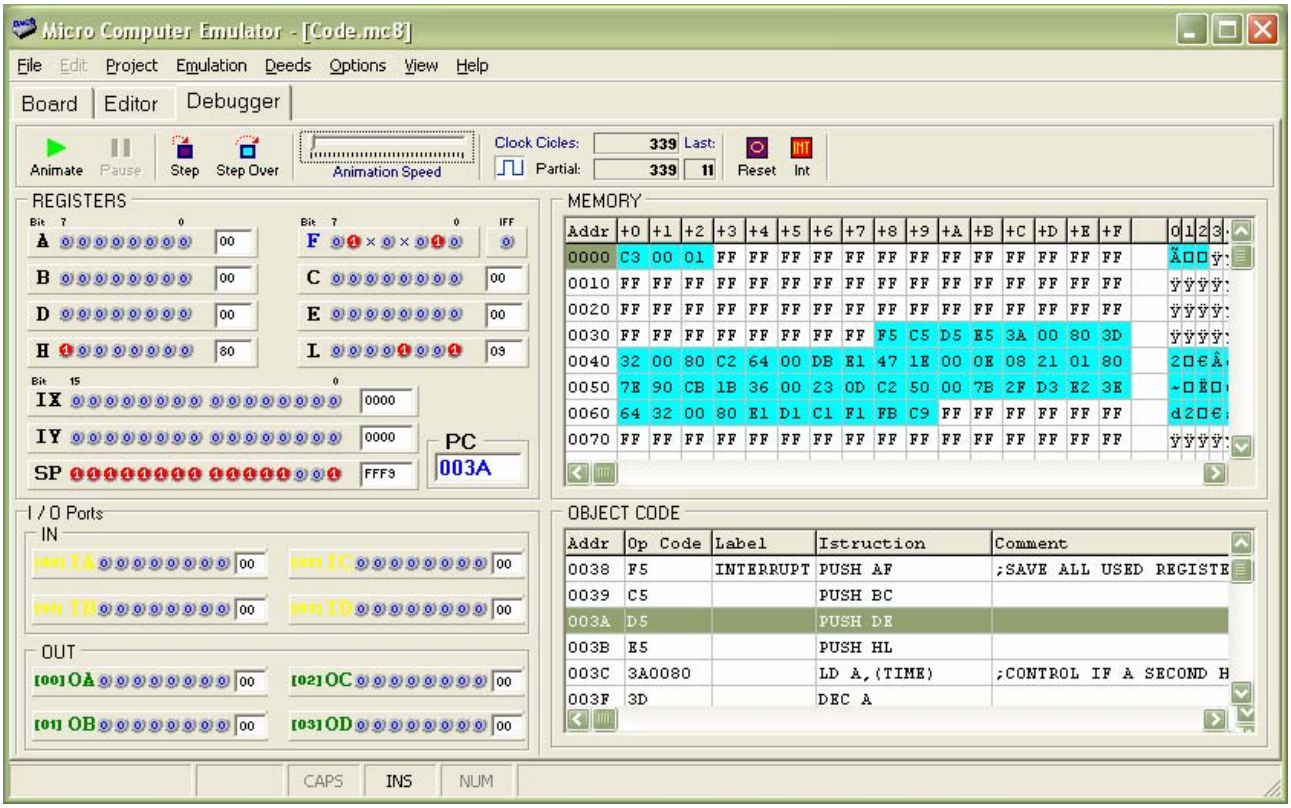


Fig. 80: The Debugger module shows the program under test, the memory, the CPU registers, the I/O ports.

The first ‘pane’ in the window shows the CPU internal registers. For instance, at this moment of the program execution, the Program Counter register contains the value 003Ah (as you can see also in the last pane, where the current instruction to be executed in actually at this address).

The second pane displays the memory contents. The used memory locations are highlighted: they correspond to the object code under execution. The user can change manually each memory location.

The third pane represents the Input / Output port contents. The user can interact with these ports, changing the Input values, by clicking on the little round buttons (corresponding to the port bits), or writing the value in the field aside (in decimal or hexadecimal coding).

The last pane presents to the user the object code in execution, as loaded in memory, in numerical format (on the left) and in assembly source format (on the right).

The student can execute the program step by step, or by animation, a modality that resembles the real execution, but at ‘human readable’ speed. A cursor permits to regulate the animation speed to the needs of the test.

A simple example of interaction between Deeds browsers and d-McE

In Fig. 81 a list of laboratory assignments is opened in the Deeds main browser. The student has to attend the assignment # 4.1 of the course on Microcomputer: *"Asynchronous serial communication"*.

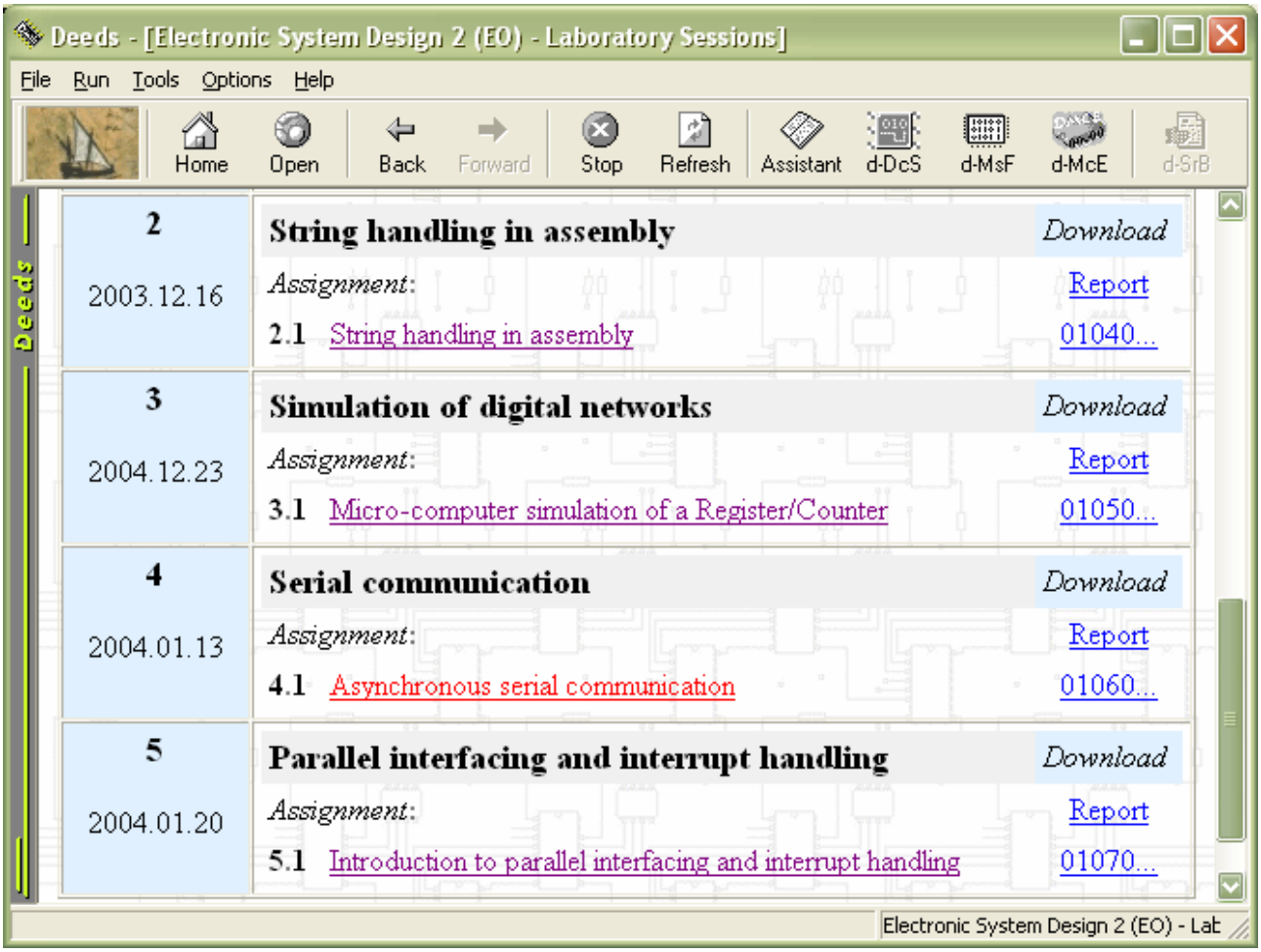


Fig. 81: A list of laboratory assignments, opened in the Deeds main browser.

With a click on the link, the assignment will open in the Assistant (see Fig. 82a and 82b).

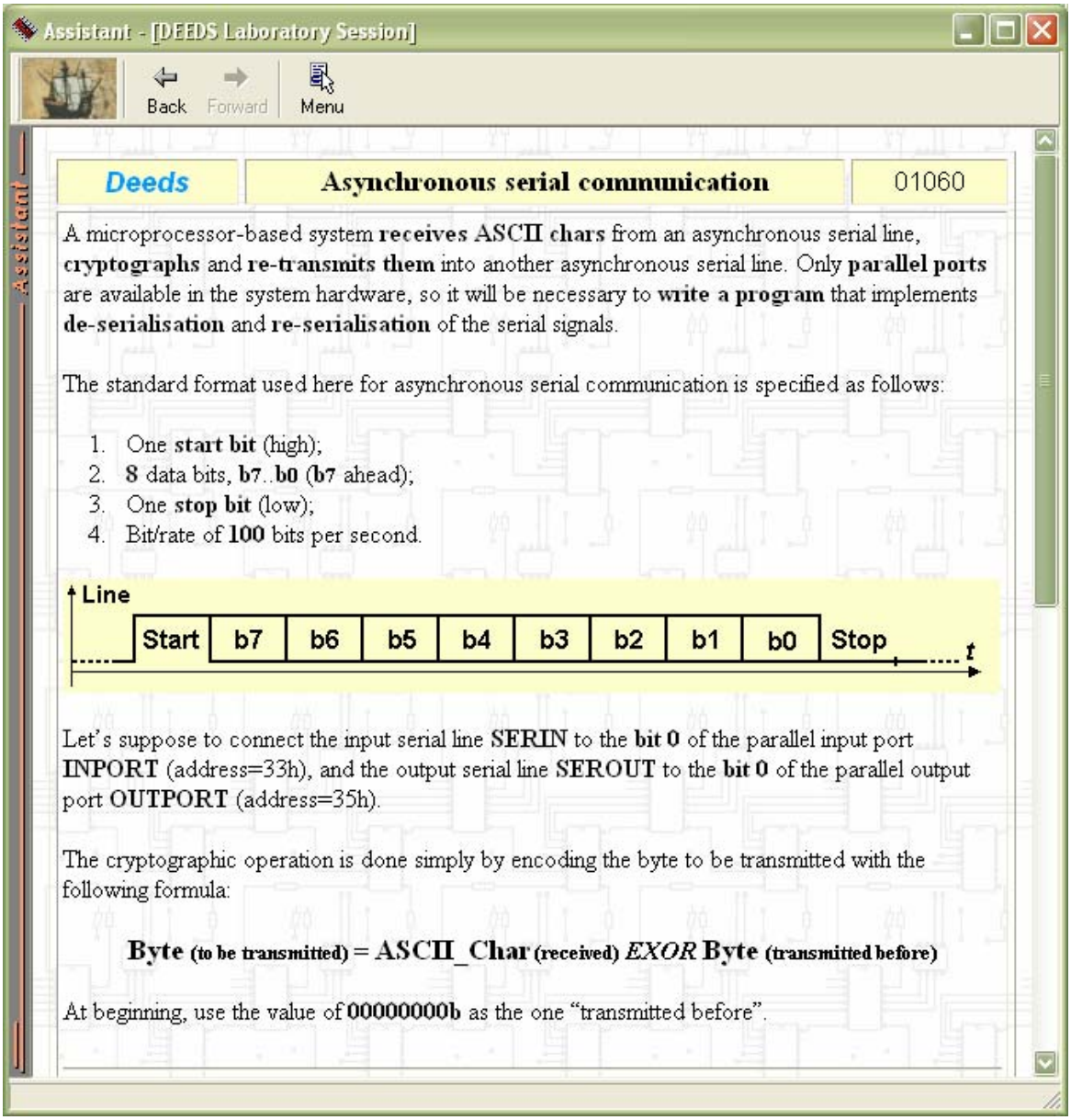


Fig. 82a: The specific laboratory assignment, opened in the Assistant browser (first part).

In this assignment (Fig. 82a), we require to the student to write a program to receive and retransmit serial asynchronous information, using the parallel ports available in the d-McE. The program should take in charge the operation of de-serializing and serializing data. Also a simple cryptographic method is applied to data before retransmitting it.

In the assignment is described the format of the serial data packet (standard 8 bit asynchronous serial communication, without parity control). That protocol previews one start bit at '1', eight data bits b7..b0 (b7 ahead), one stop bit at '0'. It is defined a low bit rate (100 bits per second), with the aim to let the user concentrate on the basic tasks, without bothering too attention to timing problems.

The text continues suggesting to connect the input and output serial lines to specific bits of the available input and output ports (INPORT and OUTPORT).

The simple cryptographic operation requires that the program remember the previous transmitted byte and combine it in a byte-wise EXOR operation with the currently received one.

Fig. 82b: The specific laboratory assignment, opened in the Assistant browser (second part).

The theme continue with the guidelines for a possible solution, as the student, at the moment of this laboratory session, faces this kind of problems for the first time (Fig. 82b).

The Deeds let to get a trace of the solution, with a simple click on the specific link. It will be automatically downloaded and opened in the source code editor of the d-McE (Fig. 83). As usual, this approach let the user simplify the operations necessary to start with the 'true' work.

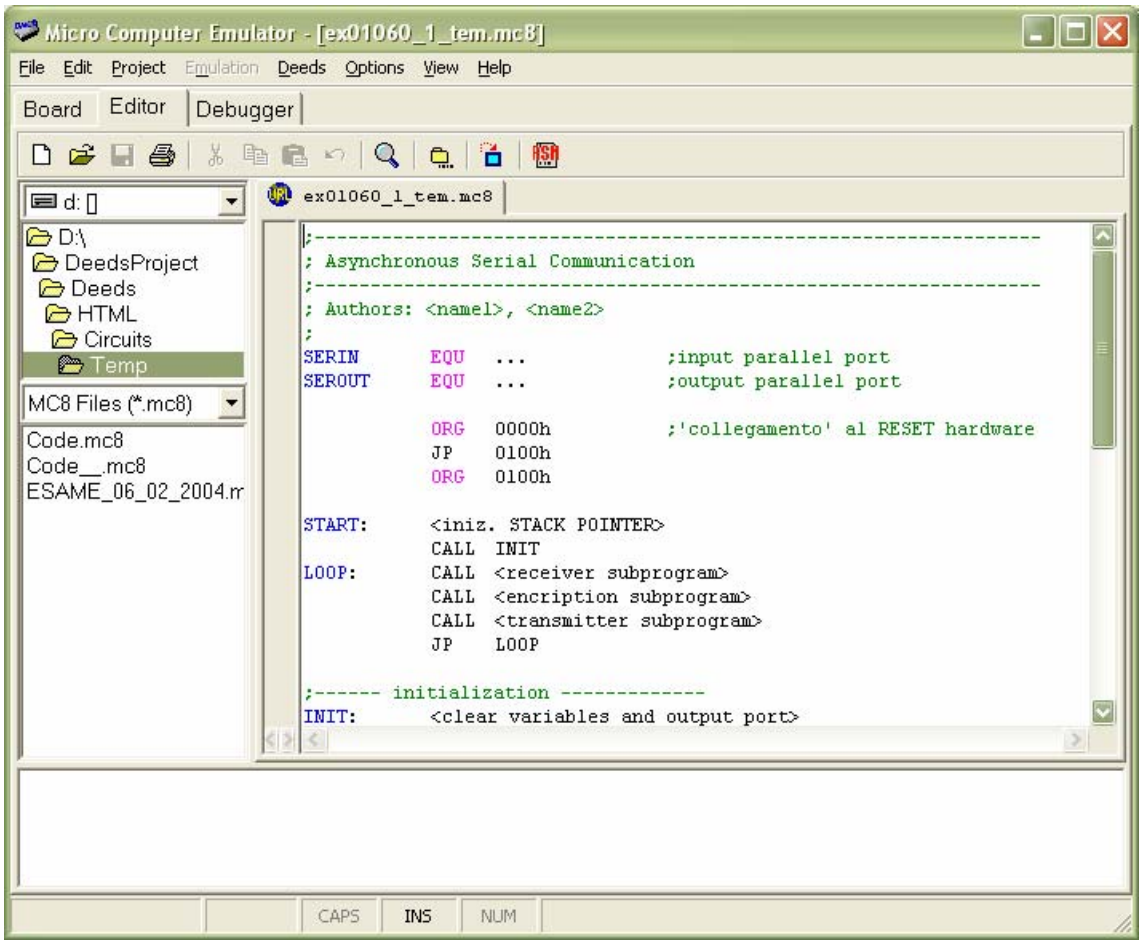


Fig. 83: The Micro Computer Emulator, opened by a click on the web page. The editor shows the trace of the solution, automatically downloaded from the courseware site.

Note the icon visible on top of the editor page: . In this case the symbol indicates that the file has been downloaded from the web. When the user will save it on the local disk, this little icon will change in .

Once completed the assembly coding of the program, the student will compile it. If no syntax error has been found, the verification of the program functionality can start (Fig. 84).

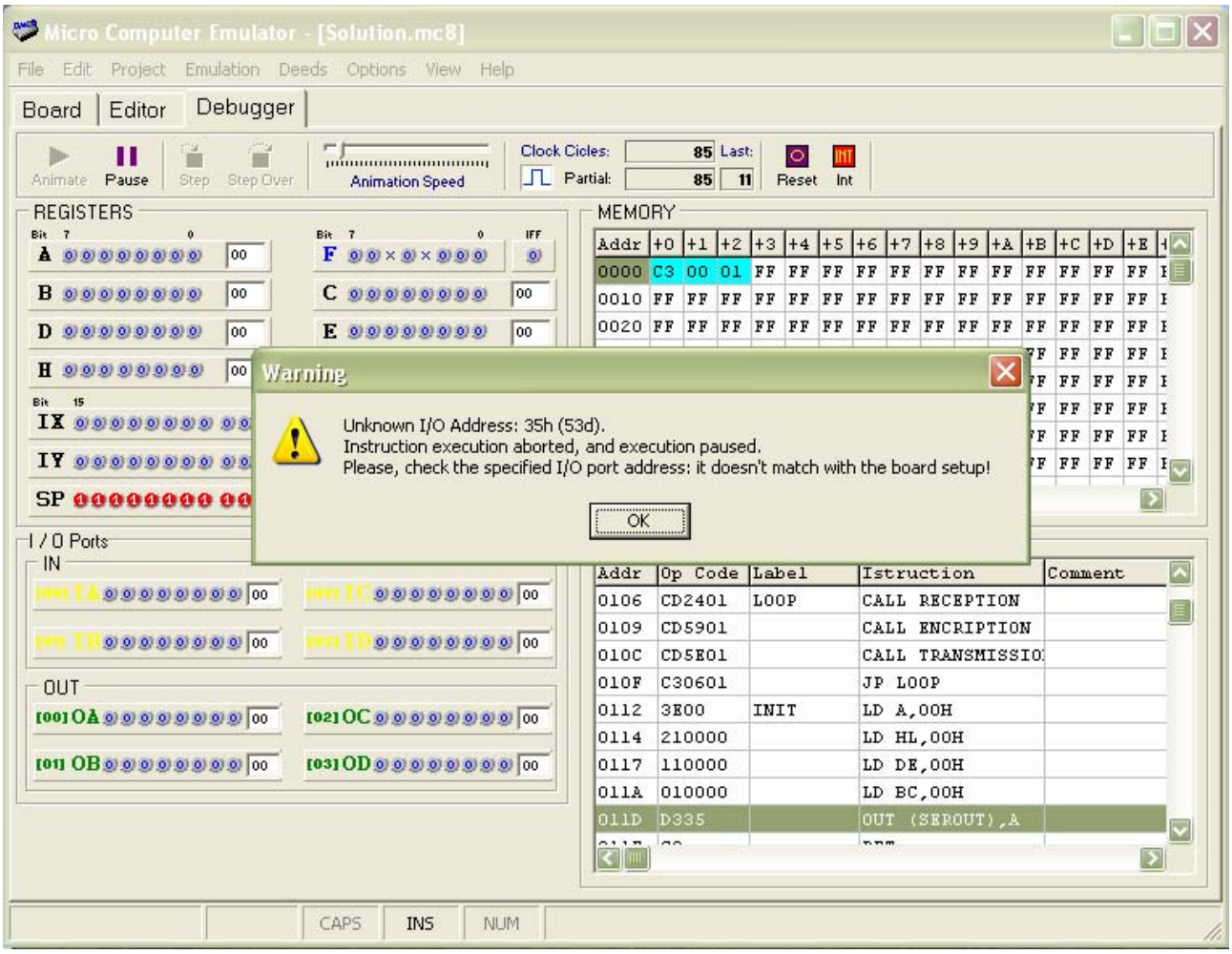


Fig. 84: The program under test in the interactive debugger of the d-McE: a Warning has be sent to the user.

In Fig. 84 the program is ‘Animated’ by the student, i.e. it is automatically executed step by step, at a ‘human readable’ speed. The speed is controlled by the cursor visible on the tool bar (“Animation Speed”).

In this example, a typical warning message is generated by the debugger. In a real case, if a port hardware address is not correctly instanced in the program code, unpredictable events could result. By the learner point of view, it could be very difficult realize what really happens in the system.

The d-McE debugger, instead, has been designed to track many common mistakes, reporting them to the student before then unwanted results could complicate the understanding of the wrong behaviour of the program.

In the present case (Fig. 84), the processor should execute the OUT instruction at address 011Dh. But the address instanced by the instruction is 35h, while no port has been set to respond to this address. So, the student has two possibilities: to return to the editor and change the source code, adapting it to the board setup, or to change the board setup.

To change the board setup, for instance, it is possible to activate (with a right-click on the port pane) the "I/O Ports Address Decoding" dialog window (Fig. 85).

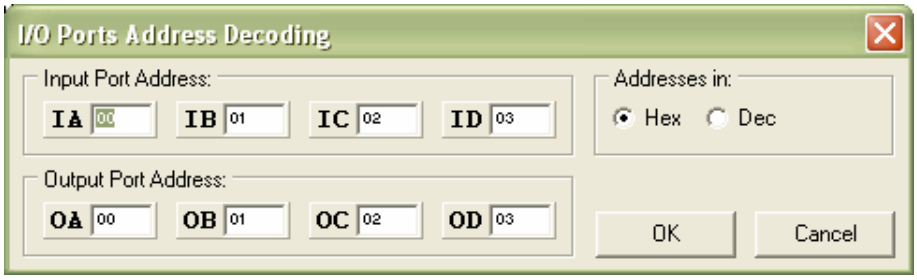


Fig. 85: Port addresses can be modified in the "I/O Ports Address Decoding" dialog window.

Another possibility, that resembles the real case, is to switch the current d-McE "page" and visualize the physical board, as seen in Fig. 74. Now it is possible to toggle, with a mouse click, the address 'dip-switches' that define the hardware address decoding (Fig. 86).
IA, IB, IC and ID are the addresses of the four parallel input ports available on board; OA, OB, OC and OD are those of the four output ports.

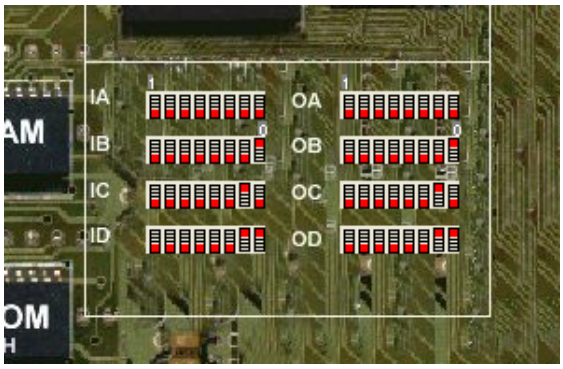


Fig. 86: Port addresses can be modified by a mouse click on the simulated 'on board' dip switches.

When finished, the student had to compile and deliver a report. A template file for the report is available in the assignment page (see Fig. 87).

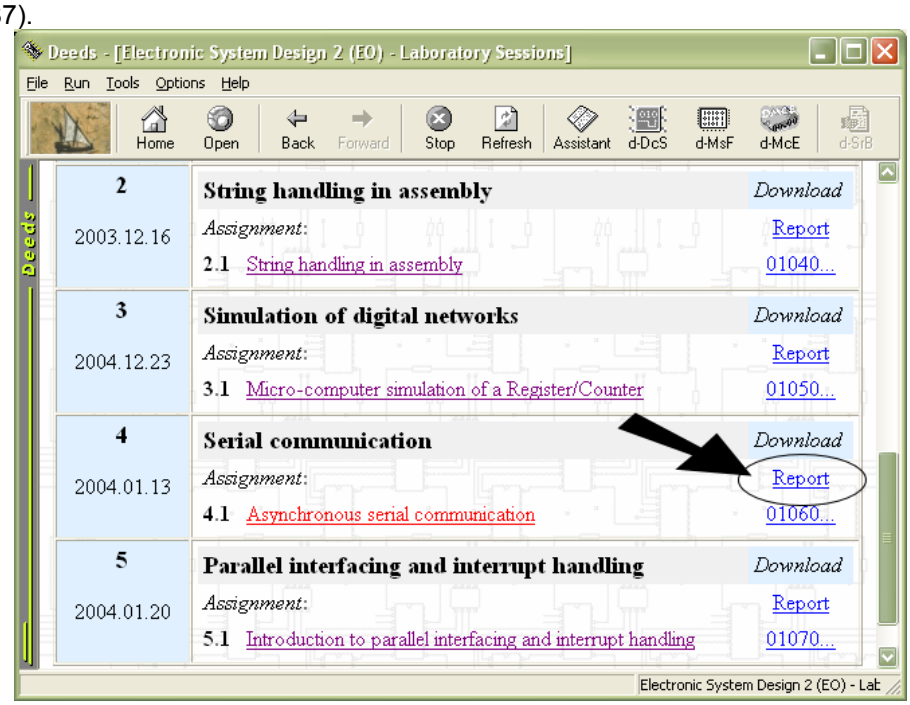


Fig. 87: The student can download the report template to speed up its compilation and delivering.

In this case, the template presents only a header that permit to uniform all the report styles, making easier the teacher task (Fig. 88).

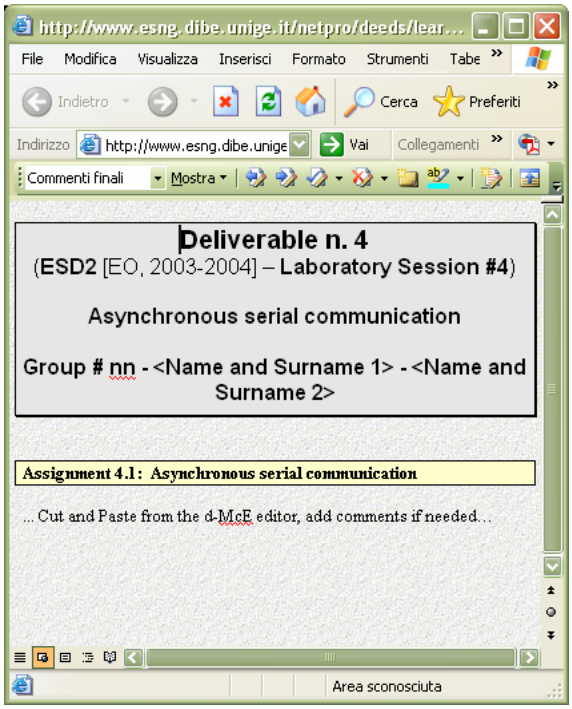


Fig. 88: The simple template provided on the web page, that the student can download.

In the next figure, an example of complete report is displayed (Fig. 89).

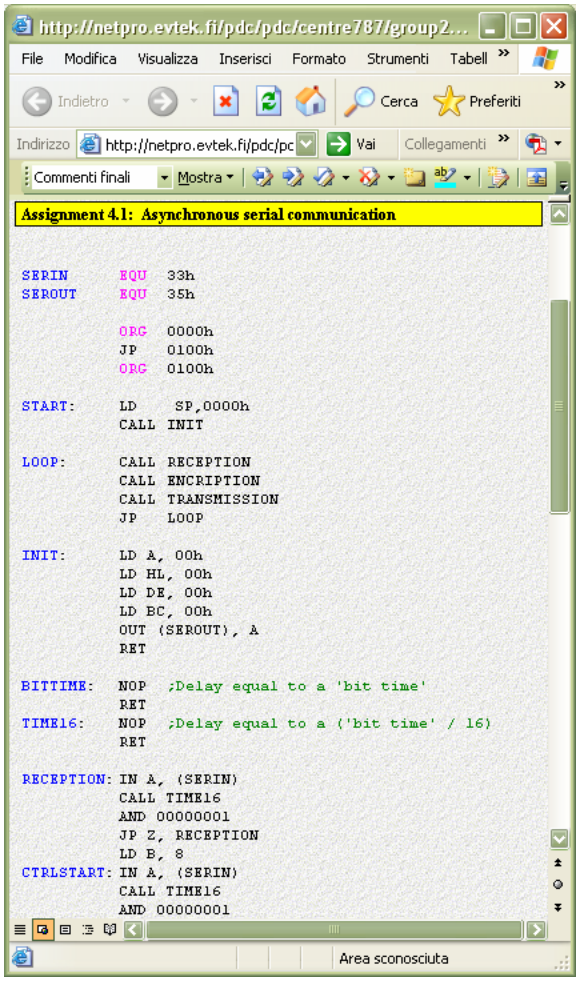


Fig. 89: A partial view of a 'final' student report.

d-McE: Menu Commands

The menu of the Micro Computer Emulator allows the user to access all the function of the application. The ToolBars replicate most of the commands already in the menu, to speed up user operations.

File Menu

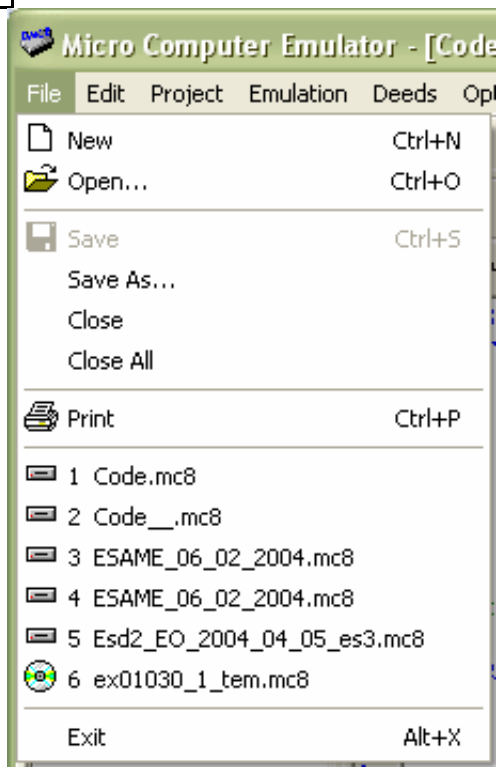


Fig. 90: The d-McE "File" menu.

New

Command to create a new (void) source file. If one or more files not void are already in the editor, a new editor page is created.

Open

Command to open a source file. If one or more files are already in the editor, a new editor page is created, and the file will be opened in it. The file can be downloaded directly from a web site.

Save

Command to save current source file.

Save as

Command to save current source file with a different name or in a different position.

Print

Command to print the source file.

Recent Files List

Commands to re-open the most recent files. Up to 8 recent files can be reopened with this list. The symbol that is displayed on the left of the file name means that:

	The file has been stored by the user on the local disk or network.
	The file has been downloaded from a web site, but it has not been saved (yet) on the local disk or network.
	The file has been loaded from a local courseware, where it is read only and it has not been saved (yet) on the local disk or network.

Exit

Standard command to close the application.

Edit Menu

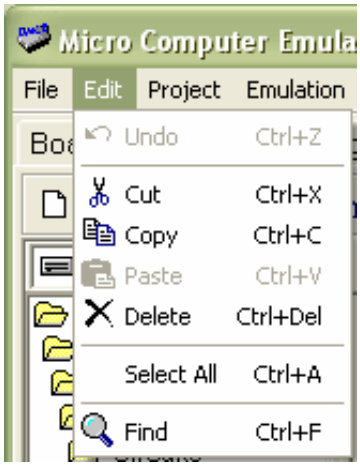


Fig. 91: The d-McE "Edit" menu.

Undo

Command to undo the previous operation (command temporary inhibited).

Cut

Command to cut the selected piece of text, and put it onto the clipboard.

Copy

Command to copy the selected piece of text onto the clipboard.

Paste

Command to paste the text from the clipboard.

Delete

Command to delete the selected piece of text.

Select All

Command to select all the text in the editor.

Find

Standard command to search strings in the text file opened in the editor.

Project Menu

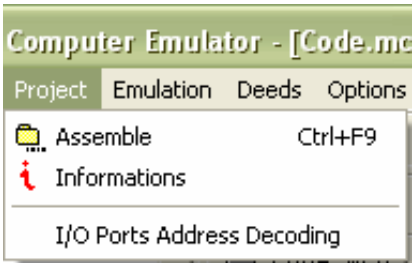


Fig. 92: The d-McE "Project" menu.

Assemble

Command to compile (assemble) the assembly source file opened in the editor.

Informations

Command to show statistical information about the previous compile (assemble) operation. It shows the "Source Info" dialog window (Fig. 93).

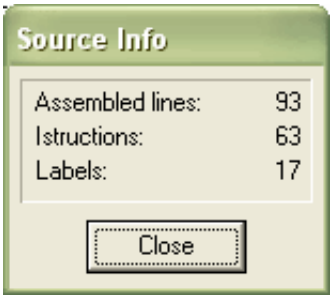


Fig. 93: The "Source Info" dialog window.

I/O Ports Address Decoding

Command to display the "I/O Ports Address Decoding" dialog window, that lets the user set the hardware addresses of the Input / Output ports (Fig. 94).

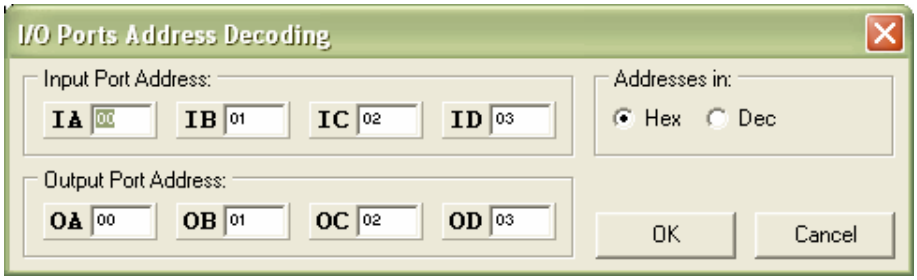


Fig. 94: The "I/O Ports Address Decoding" dialog window.

Emulation Menu

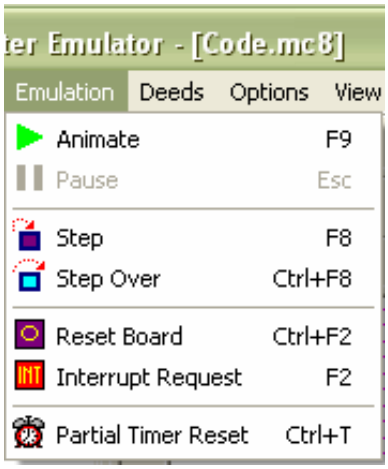


Fig. 95: The d-McE "Emulation" menu.

Animate

Debugger command to "Animate" the execution of the program.

Pause

Debugger command to pause the "Animation".

Step

Debugger command to execute one instruction (the one pointed by the Program Counter).

Step Over

This debugger command has the same effect of the previous "Step" command, except for a particular case, the execution of the **CALL**. When the Program Counter points to a **CALL** instruction, the *Step Over* command forces the execution of the program until the corresponding **RET** (return) instruction is found.

Reset Board

Debugger command to simulate the effect of a Hardware Reset.

Interrupt Request

Debugger command to simulate the effect of a Interrupt Request.

Partial Timer Reset

Debugger command to reset the 'partial' clock cycle.

Deeds Menu

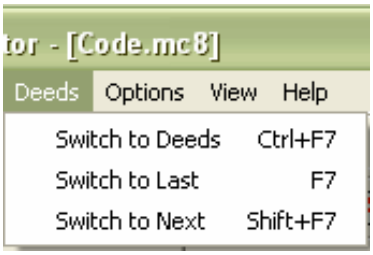


Fig. 96: The d-McE "Deeds" menu.

Switch to Deeds

Command to switch focus to the Deeds main browser.

Switch to Last

Command to switch to the tool that was 'last on top' before switching to the currently opened instance of the d-McE.

Switch to Next

Command to switch focus among all active Deeds applications, in order of activation.

Options Menu

Fig. 97: The d-McE "Options" menu.

Configuration

Command to change the application configuration (disabled in this version).

View Menu

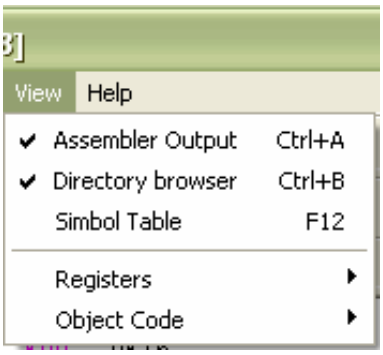


Fig. 98: The d-McE "View" menu.

Assembler Output

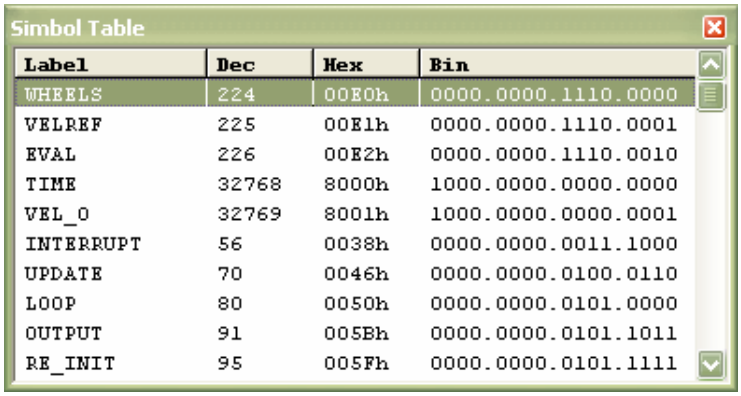
Command to hide / show the "Assembler Output" message list (at bottom).

Directory browser

Commands to hide / show the "Directory Browser" (to the left of main window)..

Symbol Table

Command to hide or show the assembler Symbol Table window (Fig. 99).



Label	Dec	Hex	Bin
WHEELS	224	00E0h	0000.0000.1110.0000
VELREF	225	00E1h	0000.0000.1110.0001
EVAL	226	00E2h	0000.0000.1110.0010
TIME	32768	8000h	1000.0000.0000.0000
VEL_0	32769	8001h	1000.0000.0000.0001
INTERRUPT	56	0038h	0000.0000.0011.1000
UPDATE	70	0046h	0000.0000.0100.0110
LOOP	80	0050h	0000.0000.0101.0000
OUTPUT	91	005Bh	0000.0000.0101.1011
RE_INIT	95	005Fh	0000.0000.0101.1111

Fig. 99: The Symbol Table window.

Registers

Command to change the user numerical format of the registers (Hexadecimal or Decimal).

Object Code

Command to change the display mode of the Object Code pane of the Debugger (Compact View or Extended View).

Help Menu

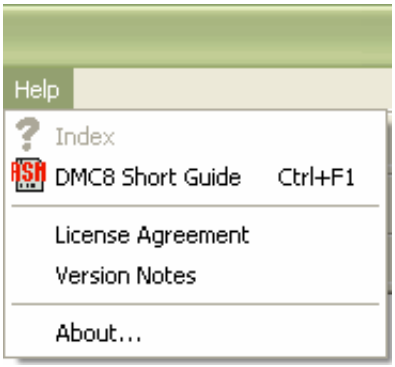


Fig. 100: The d-McE "Help" menu.

Index

Command to open the d-McE Help System (disabled in this version).

DMC8 Short Guide

Command to open the DMC8 short programming guide.

License Agreement

Command to display the Licence Agreement.

Version Notes

Command to display the Deeds "Version Notes" file.

About

Command to display the d-McE 'splash' window dialog.

DMC8 Instruction Set

In this chapter all the instructions implemented in the ***DMC8*** microprocessor are listed.

Load Instructions (8 bits)

Mnemonic	Symbolic Operation	Flags S Z H P/V N C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
LD r, r'	r ← r'	• • • • • •	01 r r'		1	1	4	r, r' Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
LD r, n	r ← n	• • • • • •	00 r 110 ← n →		2	2	7	
LD r, (HL)	r ← (HL)	• • • • • •	01 r 110		1	2	7	
LD r, (IX + d)	r ← (IX + d)	• • • • • •	11 011 101 01 r 110 ← d →	DD	3	5	19	
LD r, (IY + d)	r ← (IY + d)	• • • • • •	11 111 101 01 r 110 ← d →	FD	3	5	19	
LD (HL), r	(HL) ← r	• • • • • •	01 110 r		1	2	7	
LD (IX + d), r	(IX + d) ← r	• • • • • •	11 011 101 01 110 r ← d →	DD	3	5	19	
LD (IY + d), r	(IY + d) ← r	• • • • • •	11 111 101 01 110 r ← d →	FD	3	5	19	
LD (HL), n	(HL) ← n	• • • • • •	00 110 110 ← n →	36	2	3	10	
LD (IX + d), n	(IX + d) ← n	• • • • • •	11 011 101 00 110 110 ← d → ← n →	DD 36	4	5	19	
LD (IY + d), n	(IY + d) ← n	• • • • • •	11 111 101 00 110 110 ← d → ← n →	FD 36	4	5	19	
LD A, (BC)	A ← (BC)	• • • • • •	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	• • • • • •	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	• • • • • •	00 111 010 ← n → ← n →	3A	3	4	13	
LD (BC), A	(BC) ← A	• • • • • •	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	• • • • • •	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	• • • • • •	00 110 010 ← n → ← n →	32	3	4	13	
Notes: Flag Notation:	r, r' means any of the registers A, B, C, D, E, H, L. • = flag is not affected.							

Load Instructions (16 bits) (first section)

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
LD dd, nn	dd ← nn	•	•	•	•	•	•	00 dd0 001 ← n → ← n →		3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn	•	•	•	•	•	•	11 011 101 00 110 001 ← n → ← n →	DD 21	4	4	14	
LD IY, nn	IY ← nn	•	•	•	•	•	•	11 111 101 00 110 001 ← n → ← n →	FD 21	4	4	14	
LD HL, (nn)	L ← (nn) H ← (nn+1)	•	•	•	•	•	•	00 101 010 ← n → ← n →	2A	3	5	16	
LD dd, (nn)	dd_L ← (nn) dd_H ← (nn+1)	•	•	•	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	ED	4	6	20	
LD IX, (nn)	IX_L ← (nn) IX_H ← (nn+1)	•	•	•	•	•	•	11 011 101 00 101 010 ← n → ← n →	DD 2A	4	6	20	
LD IY, (nn)	IY_L ← (nn) IY_H ← (nn+1)	•	•	•	•	•	•	11 111 101 00 101 010 ← n → ← n →	FD 2A	4	6	20	
LD (nn), HL	(nn) ← L (nn+1) ← H	•	•	•	•	•	•	00 100 010 ← n → ← n →	22	3	5	16	
LD (nn), dd	(nn) ← dd_L (nn+1) ← dd_H	•	•	•	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	DD	4	6	20	
LD (nn), IX	(nn) ← IX_L (nn+1) ← IX_H	•	•	•	•	•	•	11 011 101 00 100 010 ← n → ← n →	DD 22	4	6	20	
LD (nn), IY	(nn) ← IY_L (nn+1) ← IY_H	•	•	•	•	•	•	11 111 101 00 100 010 ← n → ← n →	FD 22	4	6	20	
LD SP, HL	SP ← HL	•	•	•	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP ← IX	•	•	•	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LD SP, IY	SP ← IY	•	•	•	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	

Notes: dd is any of the register pair BC, DE, HL, SP.
qq is any of the register pair BC, DE, HL, AF.

Flag Notation: • = flag is not affected.

(continue)

Load Instructions (16 bits) (second section)

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
PUSH qq	SP ← SP - 1 (SP) ← qq_H SP ← SP - 1 (SP) ← qq_L	•	•	•	•	•	•	11 qq0 101		1	3	11	qq Pair 00 BC 01 DE 10 HL
PUSH IX	SP ← SP - 1 (SP) ← IX_H SP ← SP - 1 (SP) ← IX_L	•	•	•	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	11 AF
PUSH IY	SP ← SP - 1 (SP) ← IY_H SP ← SP - 1 (SP) ← IY_L	•	•	•	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	(SP) ← qq_L SP ← SP + 1 (SP) ← qq_H SP ← SP + 1	•	•	•	•	•	•	11 qq0 001		1	3	10	
POP IX	(SP) ← IX_L SP ← SP + 1 (SP) ← IX_H SP ← SP + 1	•	•	•	•	•	•	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	(SP) ← IY_L SP ← SP + 1 (SP) ← IY_H SP ← SP + 1	•	•	•	•	•	•	11 111 101 11 100 001	FD E1	2	4	14	
PUSH qq	SP ← SP - 1 (SP) ← qq_H SP ← SP - 1 (SP) ← qq_L	•	•	•	•	•	•	11 qq0 101		1	3	11	qq Pair 00 BC 01 DE 10 HL

Notes: dd is any of the register pair BC, DE, HL, SP.
qq is any of the register pair BC, DE, HL, AF.

Flag Notation: • = flag is not affected.

Arithmetic and Logic Instructions (8 bits)

Mnemonic	Symbolic Operation	Flags S Z H P/V N C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
ADD A, r	A ← A + r	↕ ↕ ↕ V 0 ↕	10 000 r		1	1	4	r Reg..
ADD A, n	A ← A + n	↕ ↕ ↕ V 0 ↕	11 000 110 ← n →		2	2	7	000 B 001 C
ADD A, (HL)	A ← A + (HL)	↕ ↕ ↕ V 0 ↕	10 000 110		1	2	7	010 D 011 E
ADD A, (IX + d)	A ← A + (IX + d)	↕ ↕ ↕ V 0 ↕	11 011 101 10 000 110 ← d →	DD	3	5	19	100 H 101 L 111 A
ADD A, (IY + d)	A ← A + (IY + d)	↕ ↕ ↕ V 0 ↕	11 111 101 10 000 110 ← d →	FD	3	5	19	
ADC A, s	A ← A + s + CY	↕ ↕ ↕ V 0 ↕	001					s is any of r, n, (HL), (IX+d), (IY+d), as shown for the ADD instruction. The underlined bits replace the underlined bits in the ADD set.
SUB s	A ← A - s	↕ ↕ ↕ V 1 ↕	010					
SBC A, s	A ← A - s - CY	↕ ↕ ↕ V 1 ↕	011					
AND s	A ← A AND s	↕ ↕ 1 P 0 0	100					
OR s	A ← A OR s	↕ ↕ 0 P 0 0	110					
XOR s	A ← A XOR s	↕ ↕ 0 P 0 0	101					
CP s	A - s	↕ ↕ ↕ V 1 ↕	111					
INC r	r ← r + 1	↕ ↕ ↕ V 0 •	00 r 100		1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕ ↕ ↕ V 0 •	00 110 100		1	3	11	
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕ ↕ ↕ V 0 •	11 011 101 00 110 100 ← d →	DD	3	6	23	
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕ ↕ ↕ V 0 •	11 111 101 00 110 100 ← d →	FD	3	6	23	
DEC m	M ← m - 1	↕ ↕ ↕ V 1 •	101					m is any of r, (HL), (IX+d), (IY+d), as shown for the INC instruction. DEC same format and states as INC. Replace 100 with 101 in opcode.
CPL	$A \leftarrow \overline{A}$	• • 1 • 1 •	00 101 111	2F	1	1	4	One's complement.
NEG	$A \leftarrow \overline{A} - 1$	↕ ↕ ↕ V 1 ↕	11 101 101 01 000 100	ED 44	2	2	8	Two's complement.

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the operation. Similarly the P symbol indicates parity.
r means any of the registers A, B, C, D, E, H, L.
CY means the carry flip-flop.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set,
↕ = flag is set according to the result of the operation.

Arithmetic Instructions (16 bits)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
ADD HL, ss	HL ← HL + ss	•	•	↕[2]	•	0	↕[1]	00 ss1 001		1	3	11	ss Reg. 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL + ss + CY	↕[1]	↕[1]	↕[2]	V[1]	0	↕[1]	11 101 101 01 ss1 010	ED	2	4	15	
SBC HL, ss	HL ← HL – ss – CY	↕[1]	↕[1]	↕[2]	V[1]	1	↕[1]	11 101 101 01 ss0 010	ED	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	↕[2]	•	0	↕[1]	11 011 101 00 pp1 001	DD	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	↕[2]	•	0	↕[1]	11 111 101 00 rr1 001	FD	2	4	15	
INC ss	ss ← ss + 1	•	•	•	•	•	•	00 ss0 011		1	1	6	
INC IX	IX ← IX + 1	•	•	•	•	•	•	11 011 101 00 100 011	DD 23	2	2	10	
INC IY	IY ← IY + 1	•	•	•	•	•	•	11 111 101 00 100 011	FD 23	2	2	10	rr Reg. 00 BC 01 DE 10 IY 11 SP
DEC ss	ss ← ss - 1	•	•	•	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX ← IX - 1	•	•	•	•	•	•	11 011 101 00 101 011	DD 2B	2	2	10	
DEC IY	IY ← IY - 1	•	•	•	•	•	•	11 111 101 00 101 011	FD 2B	2	2	10	

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the operation.
Ss means any of the registers BC, DE, HL, SP.
Pp means any of the registers BC, DE, IX, SP.
Rr means any of the registers BC, DE, IY, SP.
16 bit additions are performed by first adding the two low order eight bits, and then the two high order eight bits.
[1] Indicates the flag is affected by the 16 bit result of the operation.
[2] Indicates the flag is affected by the 8 bit addition of the high order eight bits.
CY means the carry flip-flop.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set,
↕ = flag is set according to the result of the operation.

CPU Control Instructions

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
CCF	CY ← $\overline{CY}$	•	•	X	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag.
SCF	CY ← 1	•	•	0	•	0	1	00 110 111	37	1	1	4	
NOP	No Operation	•	•	•	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	•	•	•	•	01 110 110	76	1	1	4	
DI[1]	IFF ← 0	•	•	•	•	•	•	11 110 011	F3	1	1	4	
EI[1]	IFF ← 1	•	•	•	•	•	•	11 111 011	FB	1	1	4	

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the operation.
Similarly the P symbol indicates parity.
[1] No interrupts are issued directly after a DI or EI.
CY means the carry flip-flop.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, X = flag is “don’t care”,
↕ = flag is set according to the result of the operation.

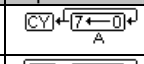
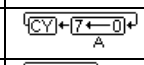
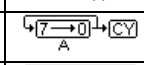
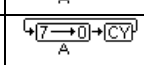
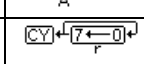
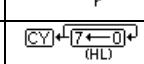
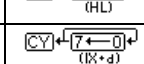
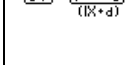
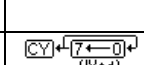
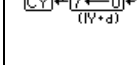
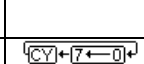
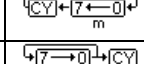
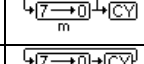
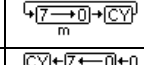
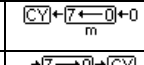
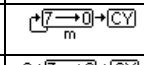
Jump Instructions

Mnemonic	Symbolic Operation	Flags S Z H P/V N C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
JP nn	PC ← nn	• • • • • •	11 000 011 ← n → ← n →	C3	3	3	10	
JP cc, nn	if cc is true, PC ← nn	• • • • • •	11 cc 010 ← n → ← n →		3	3	10	cc Condition 000 NZ non zero 001 Z zero 010 NC non carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JP (HL)	PC ← HL	• • • • • •	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	• • • • • •	11 011 101 11 101 001	DD E9	2	2	8	
JP (IY)	PC ← IY	• • • • • •	11 111 101 11 101 001	FD E9	2	2	8	
Notes: Flag Notation: • = flag is not affected.								

Call and Return Instructions

Mnemonic	Symbolic Operation	Flags S Z H P/V N C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
CALL nn	SP ← SP - 1 (SP) ← PC_H SP ← SP - 1 (SP) ← PC_L PC ← nn	• • • • • •	11 001 101 ← n → ← n →	CD	3	5	17	
CALL cc, nn	if cc is true, SP ← SP - 1 (SP) ← PC_H SP ← SP - 1 (SP) ← PC_L PC ← nn	• • • • • •	11 ccc 100 ← n → ← n →		3 3	3 5	10 17	if cc is false if cc is true
RET	PC_L ← (SP) SP ← SP + 1 PC_H ← (SP) SP ← SP + 1	• • • • • •	11 001 001	C9	1	3	10	
+RET cc	if cc is true, PC_L ← (SP) SP ← SP + 1 PC_H ← (SP) SP ← SP + 1	• • • • • •	11 ccc 000		1 1	1 3	5 11	if cc is false if cc is true
RST p	SP ← SP - 1 (SP) ← PC_H SP ← SP - 1 (SP) ← PC_L PC ← p	• • • • • •	11 t 111		1	3	11	t p 000 0000h 001 0008h 010 0010h 011 0018h 100 0020h 101 0028h 110 0030h 111 0038h
Notes: Flag Notation: • = flag is not affected.								

Rotate and Shift Instructions

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
RLCA		•	•	0	•	0	↕	00 000 111	07	1	1	4	
RLA		•	•	0	•	0	↕	00 010 111	17	1	1	4	
RRCA		•	•	0	•	0	↕	00 001 111	0F	1	1	4	
RRA		•	•	0	•	0	↕	00 011 111	1F	1	1	4	
RLC r		↕	↕	0	P	0	↕	11 001 011 00 000 r	CB	2	2	8	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (HL)		↕	↕	0	P	0	↕	11 001 011 00 000 110	CB	2	4	15	
RLC (IX + d)		↕	↕	0	P	0	↕	11 011 101 11 001 011 ← d → 00 000 110	DD CB	4	6	23	
RLC (IY + d)		↕	↕	0	P	0	↕	11 111 101 11 001 011 ← d → 00 000 110	FD CB	4	6	23	
RL m		↕	↕	0	P	0	↕	010					m is any of r, (HL), (IX+d), (IY+d), as shown for the RLC instruction. Instruction format and States are the same as RLC. Replace 000 with shown code.
RRC m		↕	↕	0	P	0	↕	001					
RR m		↕	↕	0	P	0	↕	011					
SLA m		↕	↕	0	P	0	↕	100					
SRA m		↕	↕	0	P	0	↕	101					
SRL m		↕	↕	0	P	0	↕	111					
RLD		↕	↕	0	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	
RRD		↕	↕	0	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

Notes: The P symbol in the P/V flag column indicates that the P/V flag contains the parity of the result.
r means any of the registers A, B, C, D, E, H, L.
CY means the carry flip-flop.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set,
↕ = flag is set according to the result of the operation.

Bit Handling Instructions

Mnemonic	Symbolic Operation	Flags S Z H P/V N C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
BIT b, r	$Z \leftarrow \overline{r_b}$	X ↕ 1 X 0 •	11 001 011 01 b r	CB	2	2	8	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	X ↕ 1 X 0 •	11 001 011 01 b 110	CB	2	3	12	
BIT b, (IX + d)	$Z \leftarrow \overline{(IX+ d)_b}$	X ↕ 1 X 0 •	11 011 101 11 001 011 ← d → 01 b 110	DD CB	4	5	20	
BIT b, (IY + d)	$Z \leftarrow \overline{(IY+ d)_b}$	X ↕ 1 X 0 •	11 111 101 11 001 011 ← d → 01 b 110	FD CB	4	5	20	
SET b, r	$r_b \leftarrow 1$	• • • • • •	11 001 011 11 b r	CB	2	2	8	b Bit. 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
SET b, (HL)	$(HL)_b \leftarrow 1$	• • • • • •	11 001 011 11 b 110	CB	2	4	15	
SET b, (IX + d)	$(IX+ d)_b \leftarrow 1$	• • • • • •	11 011 101 11 001 011 ← d → 11 b 110	DD CB	4	6	23	
SET b, (IY + d)	$(IY+ d)_b \leftarrow 1$	• • • • • •	11 111 101 11 001 011 ← d → 11 b 110	FD CB	4	6	23	
RES b, m	$m_b \leftarrow 0$ m ≡ r, (HL), (IX+d), (IY+d)	• • • • • •	10					To form new opcode replace 11 of SET b, s with 10. Flags and states are the same.

Notes: The notation m_b indicates bit b (0 to 7) of location m.
BIT instructions are performed by an bitwise AND.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, X = flag is "don't care",
↕ = flag is set according to the result of the operation.

Input and Output Instructions

Mnemonic	Symbolic Operation	Flags S Z H P/V N C	Opcode 76 543 210	Hex	Bytes	M Cycles	Clock Cycles	Comments
IN A, (n)	A ← (n)	• • • • • •	11 011 011 ← n →	DB	2	3	11	R Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
IN r, (C)	r ← (C)	↕ ↕ 0 P 0 •	11 101 101 01 r 000	ED	2	3	12	
OUT (n), A	(n) ← A	• • • • • •	11 010 011 ← n →	D3	2	3	11	
OUT (C), r	(C) ← r	• • • • • •	11 101 101 01 r 001	ED	2	3	12	

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the operation.
Similarly the P symbol indicates parity.
r means any of the registers A, B, C, D, E, H, L.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set,
↕ = flag is set according to the result of the operation.

DMC8 Instructions (in alfabetical order)

The instructions are 659, considering all the possible variations.

8E	ADC A, (HL)
DD8E d	ADC A, (IX + d)
FD8E d	ADC A, (IY + d)
8F	ADC A, A
88	ADC A, B
89	ADC A, C
8A	ADC A, D
8B	ADC A, E
8C	ADC A, H
8D	ADC A, L
CE n	ADC A, n
ED4A	ADC HL, BC
ED5A	ADC HL, DE
ED6A	ADC HL, HL
ED7A	ADC HL, SP
86	ADD A, (HL)
DD86 d	ADD A, (IX + d)
FD86 d	ADD A, (IY + d)
87	ADD A, A
80	ADD A, B
81	ADD A, C
82	ADD A, D
83	ADD A, E
84	ADD A, H
85	ADD A, L
C6 n	ADD A, n
9	ADD HL, BC
19	ADD HL, DE
29	ADD HL, HL
39	ADD HL, SP
DD09	ADD IX, BC
DD19	ADD IX, DE
DD29	ADD IX, IX
DD39	ADD IX, SP
FD09	ADD IY, BC
FD19	ADD IY, DE
FD29	ADD IY, IY
FD39	ADD IY, SP
A6	AND (HL)
DDA6 d	AND (IX + d)
FDA6 d	AND (IY + d)
A7	AND A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
E6 n	AND n
CB46	BIT 0, (HL)
DDCB d 46	BIT 0, (IX + d)
FDCB d 46	BIT 0, (IY + d)
CB47	BIT 0, A
CB40	BIT 0, B
CB41	BIT 0, C
CB42	BIT 0, D
CB43	BIT 0, E
CB44	BIT 0, H
CB45	BIT 0, L
CB4E	BIT 1, (HL)
DDCB d 4E	BIT 1, (IX + d)
FDCB d 4E	BIT 1, (IY + d)
CB4F	BIT 1, A
CB48	BIT 1, B
CB49	BIT 1, C
CB4A	BIT 1, D
CB4B	BIT 1, E
CB4C	BIT 1, H
CB4D	BIT 1, L
CB56	BIT 2, (HL)
DDCB d 56	BIT 2, (IX + d)
FDCB d 56	BIT 2, (IY + d)
CB57	BIT 2, A
CB50	BIT 2, B
CB51	BIT 2, C
CB52	BIT 2, D
CB53	BIT 2, E
CB54	BIT 2, H
CB55	BIT 2, L
CB5E	BIT 3, (HL)
DDCB d 5E	BIT 3, (IX + d)
FDCB d 5E	BIT 3, (IY + d)
CB5F	BIT 3, A
CB58	BIT 3, B
CB59	BIT 3, C
CB5A	BIT 3, D
CB5B	BIT 3, E
CB5C	BIT 3, H
CB5D	BIT 3, L
CB66	BIT 4, (HL)
DDCB d 66	BIT 4, (IX + d)
FDCB d 66	BIT 4, (IY + d)
CB67	BIT 4, A
CB60	BIT 4, B
CB61	BIT 4, C
CB62	BIT 4, D
CB63	BIT 4, E
CB64	BIT 4, H
CB65	BIT 4, L
CB6E	BIT 5, (HL)
DDCB d 6E	BIT 5, (IX + d)
FDCB d 6E	BIT 5, (IY + d)
CB6F	BIT 5, A
CB68	BIT 5, B
CB69	BIT 5, C
CB6A	BIT 5, D
CB6B	BIT 5, E
CB6C	BIT 5, H
CB6D	BIT 5, L
CB76	BIT 6, (HL)
DDCB d 76	BIT 6, (IX + d)
FDCB d 76	BIT 6, (IY + d)
CB77	BIT 6, A
CB70	BIT 6, B
CB71	BIT 6, C
CB72	BIT 6, D
CB73	BIT 6, E
CB74	BIT 6, H
CB75	BIT 6, L
CB7E	BIT 7, (HL)
DDCB d 7E	BIT 7, (IX + d)
FDCB d 7E	BIT 7, (IY + d)
CB7F	BIT 7, A
CB78	BIT 7, B
CB79	BIT 7, C
CB7A	BIT 7, D
CB7B	BIT 7, E
CB7C	BIT 7, H
CB7D	BIT 7, L
DC n n	CALL C, nn
FC n n	CALL M, nn
D4 n n	CALL NC, nn
CD n n	CALL nn
C4 n n	CALL NZ, nn
F4 n n	CALL P, nn
EC n n	CALL PE, nn
E4 n n	CALL PO, nn
CC n n	CALL Z, nn
3F	CCF
BE	CP (HL)
DDBE d	CP (IX + d)
FDBE d	CP (IY + d)
BF	CP A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
FE n	CP n
2F	CPL
35	DEC (HL)
DD35 d	DEC (IX + d)

FD35 d	DEC (IY + d)
3D	DEC A
5	DEC B
0B	DEC BC
0D	DEC C
15	DEC D
1B	DEC DE
1D	DEC E
25	DEC H
2B	DEC HL
DD2B	DEC IX
FD2B	DEC IY
2D	DEC L
3B	DEC SP
F3	DI
FB	EI
76	HALT
ED78	IN A, (C)
DB n	IN A, (n)
ED40	IN B, (C)
ED48	IN C, (C)
ED50	IN D, (C)
ED58	IN E, (C)
ED60	IN H, (C)
ED68	IN L, (C)
34	INC (HL)
DD34 d	INC (IX + d)
FD34 d	INC (IY + d)
3C	INC A
4	INC B
3	INC BC
0C	INC C
14	INC D
13	INC DE
1C	INC E
24	INC H
23	INC HL
DD23	INC IX
FD23	INC IY
2C	INC L
33	INC SP
E9	JP (HL)
DDE9	JP (IX)
FDE9	JP (IY)
DA n n	JP C, nn
FA n n	JP M, nn
D2 n n	JP NC, nn
C3 n n	JP nn
C2 n n	JP NZ, nn
F2 n n	JP P, nn
EA n n	JP PE, nn
E2 n n	JP PO, nn
CA n n	JP Z, nn
2	LD (BC), A
12	LD (DE), A
77	LD (HL), A
70	LD (HL), B
71	LD (HL), C
72	LD (HL), D
73	LD (HL), E
74	LD (HL), H
75	LD (HL), L
36 n	LD (HL), n
DD77 d	LD (IX + d), A
DD70 d	LD (IX + d), B
DD71 d	LD (IX + d), C
DD72 d	LD (IX + d), D
DD73 d	LD (IX + d), E
DD74 d	LD (IX + d), H
DD75 d	LD (IX + d), L
DD36 d n	LD (IX + d), n
FD77 d	LD (IY + d), A
FD70 d	LD (IY + d), B
FD71 d	LD (IY + d), C
FD72 d	LD (IY + d), D
FD73 d	LD (IY + d), E
FD74 d	LD (IY + d), H
FD75 d	LD (IY + d), L
FD36 d n	LD (IY + d), n
32 n n	LD (nn), A
ED43 n n	LD (nn), BC
ED53 n n	LD (nn), DE
22 n n	LD (nn), HL
ED63 n n	LD (nn), HL
DD22 n n	LD (nn), IX
FD22 n n	LD (nn), IY
ED73 n n	LD (nn), SP
0A	LD A, (BC)
1A	LD A, (DE)
7E	LD A, (HL)
DD7E d	LD A, (IX + d)
FD7E d	LD A, (IY + d)
3A n n	LD A, (nn)
7F	LD A, A
78	LD A, B
79	LD A, C
7A	LD A, D
7B	LD A, E
7C	LD A, H
7D	LD A, L
3E n	LD A, n
46	LD B, (HL)
DD46 d	LD B, (IX + d)
FD46 d	LD B, (IY + d)
47	LD B, A
40	LD B, B
41	LD B, C
42	LD B, D
43	LD B, E
44	LD B, H
45	LD B, L
06 n	LD B, n
ED4B n n	LD BC, (nn)
01 n n	LD BC, nn
4E	LD C, (HL)
DD4E d	LD C, (IX + d)
FD4E d	LD C, (IY + d)
4F	LD C, A
48	LD C, B
49	LD C, C
4A	LD C, D
4B	LD C, E
4C	LD C, H
4D	LD C, L
0E n	LD C, n
56	LD D, (HL)
DD56 d	LD D, (IX + d)
FD56 d	LD D, (IY + d)
57	LD D, A
50	LD D, B
51	LD D, C
52	LD D, D
53	LD D, E
54	LD D, H
55	LD D, L
16 n	LD D, n
ED5B n n	LD DE, (nn)
11 n n	LD DE, nn
5E	LD E, (HL)
DD5E d	LD E, (IX + d)
FD5E d	LD E, (IY + d)
5F	LD E, A
58	LD E, B
59	LD E, C
5A	LD E, D
5B	LD E, E
5C	LD E, H
5D	LD E, L
1E n	LD E, n
66	LD H, (HL)
DD66 d	LD H, (IX + d)
FD66 d	LD H, (IY + d)
67	LD H, A
60	LD H, B
61	LD H, C
62	LD H, D
63	LD H, E
64	LD H, H
65	LD H, L
26 n	LD H, n
2A n n	LD HL, (nn)
21 n n	LD HL, nn
DD2A n n	LD IX, (nn)
DD21 n n	LD IX, nn
FD2A n n	LD IY, (nn)
FD21 n n	LD IY, nn
6E	LD L, (HL)
DD6E d	LD L, (IX + d)
FD6E d	LD L, (IY + d)
6F	LD L, A
68	LD L, B
69	LD L, C
6A	LD L, D
6B	LD L, E
6C	LD L, H
6D	LD L, L
2E n	LD L, n
ED7B n n	LD SP, (nn)
F9	LD SP, HL
DDF9	LD SP, IX

FDF9	LD SP, IY
31 n n	LD SP, nn
ED44	NEG
0	NOP
B6	OR (HL)
DDB6 d	OR (IX + d)
FDB6 d	OR (IY + d)
B7	OR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
F6 n	OR n
ED79	OUT (C), A
ED41	OUT (C), B
ED49	OUT (C), C
ED51	OUT (C), D
ED59	OUT (C), E
ED61	OUT (C), H
ED69	OUT (C), L
D3 n	OUT (n), A
F1	POP AF
C1	POP BC
D1	POP DE
E1	POP HL
DDE1	POP IX
FDE1	POP IY
F5	PUSH AF
C5	PUSH BC
D5	PUSH DE
E5	PUSH HL
DDE5	PUSH IX
FDE5	PUSH IY
CB86	RES 0, (HL)
DDCB d 86	RES 0, (IX + d)
FDCB d 86	RES 0, (IY + d)
CB87	RES 0, A
CB80	RES 0, B
CB81	RES 0, C
CB82	RES 0, D
CB83	RES 0, E
CB84	RES 0, H
CB85	RES 0, L
CB8E	RES 1, (HL)
DDCB d 8E	RES 1, (IX + d)
FDCB d 8E	RES 1, (IY + d)
CB8F	RES 1, A
CB88	RES 1, B
CB89	RES 1, C
CB8A	RES 1, D
CB8B	RES 1, E
CB8C	RES 1, H
CB8D	RES 1, L
CB96	RES 2, (HL)
DDCB d 96	RES 2, (IX + d)
FDCB d 96	RES 2, (IY + d)
CB97	RES 2, A
CB90	RES 2, B
CB91	RES 2, C
CB92	RES 2, D
CB93	RES 2, E
CB94	RES 2, H
CB95	RES 2, L
CB9E	RES 3, (HL)
DDCB d 9E	RES 3, (IX + d)
FDCB d 9E	RES 3, (IY + d)
CB9F	RES 3, A
CB98	RES 3, B
CB99	RES 3, C
CB9A	RES 3, D
CB9B	RES 3, E
CB9C	RES 3, H
CB9D	RES 3, L
CBA6	RES 4, (HL)
DDCB d A6	RES 4, (IX + d)
FDCB d A6	RES 4, (IY + d)
CBA7	RES 4, A
CBA0	RES 4, B
CBA1	RES 4, C
CBA2	RES 4, D
CBA3	RES 4, E
CBA4	RES 4, H
CBA5	RES 4, L
CBAE	RES 5, (HL)
DDCB d AE	RES 5, (IX + d)
FDCB d AE	RES 5, (IY + d)
CBAF	RES 5, A
CBA8	RES 5, B
CBA9	RES 5, C
CBAA	RES 5, D
CBAB	RES 5, E
CBAC	RES 5, H
CBAD	RES 5, L
CBB6	RES 6, (HL)
DDCB d B6	RES 6, (IX + d)
FDCB d B6	RES 6, (IY + d)
CBB7	RES 6, A
CBB0	RES 6, B
CBB1	RES 6, C
CBB2	RES 6, D
CBB3	RES 6, E
CBB4	RES 6, H
CBB5	RES 6, L
CBBE	RES 7, (HL)
DDCB d BE	RES 7, (IX + d)
FDCB d BE	RES 7, (IY + d)
CBBF	RES 7, A
CBB8	RES 7, B
CBB9	RES 7, C
CBBA	RES 7, D
CBBB	RES 7, E
CBBC	RES 7, H
CBBD	RES 7, L
C9	RET
D8	RET C
F8	RET M
D0	RET NC
C0	RET NZ
F0	RET P
E8	RET PE
E0	RET PO
C8	RET Z
CB16	RL (HL)
DDCB d 16	RL (IX + d)
FDCB d 16	RL (IY + d)
CB17	RL A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
17	RLA
CB06	RLC (HL)
DDCB d 06	RLC (IX + d)
FDCB d 06	RLC (IY + d)
CB07	RLC A
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
7	RLCA
ED6F	RLD
CB1E	RR (HL)
DDCB d 1E	RR (IX + d)
FDCB d 1E	RR (IY + d)
CB1F	RR A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
1F	RRA
CB0E	RRC (HL)
DDCB d 0E	RRC (IX + d)
FDCB d 0E	RRC (IY + d)
CB0F	RRC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
0F	RRCA
ED67	RRD
C7	RST 00h
CF	RST 08h
D7	RST 10h
DF	RST 18h
E7	RST 20h
EF	RST 28h
F7	RST 30h
FF	RST 38h
9E	SBC A, (HL)
DD9E d	SBC A, (IX + d)

FD9E d	SBC A, (IY + d)
9F	SBC A, A
98	SBC A, B
99	SBC A, C
9A	SBC A, D
9B	SBC A, E
9C	SBC A, H
9D	SBC A, L
DE n	SBC A, n
ED42	SBC HL, BC
ED52	SBC HL, DE
ED62	SBC HL, HL
ED72	SBC HL, SP
37	SCF
CBC6	SET 0, (HL)
DDCB d C6	SET 0, (IX + d)
FDCB d C6	SET 0, (IY + d)
CBC7	SET 0, A
CBC0	SET 0, B
CBC1	SET 0, C
CBC2	SET 0, D
CBC3	SET 0, E
CBC4	SET 0, H
CBC5	SET 0, L
CBCE	SET 1, (HL)
DDCB d CE	SET 1, (IX + d)
FDCB d CE	SET 1, (IY + d)
CBCF	SET 1, A
CBC8	SET 1, B
CBC9	SET 1, C
CBCA	SET 1, D
CBCB	SET 1, E
CBCC	SET 1, H
CBCD	SET 1, L
CBD6	SET 2, (HL)
DDCB d D6	SET 2, (IX + d)
FDCB d D6	SET 2, (IY + d)
CBD7	SET 2, A
CBD0	SET 2, B
CBD1	SET 2, C
CBD2	SET 2, D
CBD3	SET 2, E
CBD4	SET 2, H
CBD5	SET 2, L
CBDE	SET 3, (HL)
DDCB d DE	SET 3, (IX + d)
FDCB d DE	SET 3, (IY + d)
CBDF	SET 3, A
CBD8	SET 3, B
CBD9	SET 3, C
CBDA	SET 3, D
CBDB	SET 3, E
CBDC	SET 3, H
CBDD	SET 3, L
CBE6	SET 4, (HL)
DDCB d E6	SET 4, (IX + d)
FDCB d E6	SET 4, (IY + d)
CBE7	SET 4, A
CBE0	SET 4, B
CBE1	SET 4, C
CBE2	SET 4, D
CBE3	SET 4, E
CBE4	SET 4, H
CBE5	SET 4, L
CBEE	SET 5, (HL)
DDCB d EE	SET 5, (IX + d)
FDCB d EE	SET 5, (IY + d)
CBEF	SET 5, A
CBE8	SET 5, B
CBE9	SET 5, C
CBEA	SET 5, D
CBEB	SET 5, E
CBEC	SET 5, H
CBED	SET 5, L
CBF6	SET 6, (HL)
DDCB d F6	SET 6, (IX + d)
FDCB d F6	SET 6, (IY + d)
CBF7	SET 6, A
CBF0	SET 6, B
CBF1	SET 6, C
CBF2	SET 6, D
CBF3	SET 6, E
CBF4	SET 6, H
CBF5	SET 6, L
CBFE	SET 7, (HL)
DDCB d FE	SET 7, (IX + d)
FDCB d FE	SET 7, (IY + d)
CBFF	SET 7, A
CBF8	SET 7, B
CBF9	SET 7, C
CBFA	SET 7, D
CBFB	SET 7, E
CBFC	SET 7, H
CBFD	SET 7, L
CB26	SLA (HL)
DDCB d 26	SLA (IX + d)
FDCB d 26	SLA (IY + d)
CB27	SLA A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB2E	SRA (HL)
DDCB d 2E	SRA (IX + d)
FDCB d 2E	SRA (IY + d)
CB2F	SRA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB3E	SRL (HL)
DDCB d 3E	SRL (IX + d)
FDCB d 3E	SRL (IY + d)
CB3F	SRL A
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
96	SUB (HL)
DD96 d	SUB (IX + d)
FD96 d	SUB (IY + d)
97	SUB A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
D6 n	SUB n
AE	XOR (HL)
DDAE d	XOR (IX + d)
FDAE d	XOR (IY + d)
AF	XOR A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
EE n	XOR n

DMC8 Instructions (in numerical order)

0	NOP	40	LD B, B	78	LD A, B
01 n n	LD BC, nn	41	LD B, C	79	LD A, C
2	LD (BC), A	42	LD B, D	7A	LD A, D
3	INC BC	43	LD B, E	7B	LD A, E
4	INC B	44	LD B, H	7C	LD A, H
5	DEC B	45	LD B, L	7D	LD A, L
06 n	LD B, n	46	LD B, (HL)	7E	LD A, (HL)
7	RLCA	47	LD B, A	7F	LD A, A
9	ADD HL, BC	48	LD C, B	80	ADD A, B
0A	LD A, (BC)	49	LD C, C	81	ADD A, C
0B	DEC BC	4A	LD C, D	82	ADD A, D
0C	INC C	4B	LD C, E	83	ADD A, E
0D	DEC C	4C	LD C, H	84	ADD A, H
0E n	LD C, n	4D	LD C, L	85	ADD A, L
0F	RRCA	4E	LD C, (HL)	86	ADD A, (HL)
11 n n	LD DE, nn	4F	LD C, A	87	ADD A, A
12	LD (DE), A	50	LD D, B	88	ADC A, B
13	INC DE	51	LD D, C	89	ADC A, C
14	INC D	52	LD D, D	8A	ADC A, D
15	DEC D	53	LD D, E	8B	ADC A, E
16 n	LD D, n	54	LD D, H	8C	ADC A, H
17	RLA	55	LD D, L	8D	ADC A, L
19	ADD HL, DE	56	LD D, (HL)	8E	ADC A, (HL)
1A	LD A, (DE)	57	LD D, A	8F	ADC A, A
1B	DEC DE	58	LD E, B	90	SUB B
1C	INC E	59	LD E, C	91	SUB C
1D	DEC E	5A	LD E, D	92	SUB D
1E n	LD E, n	5B	LD E, E	93	SUB E
1F	RRA	5C	LD E, H	94	SUB H
21 n n	LD HL, nn	5D	LD E, L	95	SUB L
22 n n	LD (nn), HL	5E	LD E, (HL)	96	SUB (HL)
23	INC HL	5F	LD E, A	97	SUB A
24	INC H	60	LD H, B	98	SBC A, B
25	DEC H	61	LD H, C	99	SBC A, C
26 n	LD H, n	62	LD H, D	9A	SBC A, D
29	ADD HL, HL	63	LD H, E	9B	SBC A, E
2A n n	LD HL, (nn)	64	LD H, H	9C	SBC A, H
2B	DEC HL	65	LD H, L	9D	SBC A, L
2C	INC L	66	LD H, (HL)	9E	SBC A, (HL)
2D	DEC L	67	LD H, A	9F	SBC A, A
2E n	LD L, n	68	LD L, B	A0	AND B
2F	CPL	69	LD L, C	A1	AND C
31 n n	LD SP, nn	6A	LD L, D	A2	AND D
32 n n	LD (nn), A	6B	LD L, E	A3	AND E
33	INC SP	6C	LD L, H	A4	AND H
34	INC (HL)	6D	LD L, L	A5	AND L
35	DEC (HL)	6E	LD L, (HL)	A6	AND (HL)
36 n	LD (HL), n	6F	LD L, A	A7	AND A
37	SCF	70	LD (HL), B	A8	XOR B
39	ADD HL, SP	71	LD (HL), C	A9	XOR C
3A n n	LD A, (nn)	72	LD (HL), D	AA	XOR D
3B	DEC SP	73	LD (HL), E	AB	XOR E
3C	INC A	74	LD (HL), H	AC	XOR H
3D	DEC A	75	LD (HL), L	AD	XOR L
3E n	LD A, n	76	HALT	AE	XOR (HL)
3F	CCF	77	LD (HL), A	AF	XOR A

B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
B6	OR (HL)
B7	OR A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
BE	CP (HL)
BF	CP A
C0	RET NZ
C1	POP BC
C2 n n	JP NZ, nn
C3 n n	JP nn
C4 n n	CALL NZ, nn
C5	PUSH BC
C6 n	ADD A, n
C7	RST 0h
C8	RET Z
C9	RET
CA n n	JP Z, nn
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
CB06	RLC (HL)
CB07	RLC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
CB0E	RRC (HL)
CB0F	RRC A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
CB16	RL (HL)
CB17	RL A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
CB1E	RR (HL)
CB1F	RR A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB26	SLA (HL)
CB27	SLA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB2E	SRA (HL)
CB2F	SRA A
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
CB3E	SRL (HL)
CB3F	SRL A
CB40	BIT 0, B
CB41	BIT 0, C
CB42	BIT 0, D
CB43	BIT 0, E
CB44	BIT 0, H
CB45	BIT 0, L
CB46	BIT 0, (HL)
CB47	BIT 0, A
CB48	BIT 1, B
CB49	BIT 1, C
CB4A	BIT 1, D
CB4B	BIT 1, E
CB4C	BIT 1, H
CB4D	BIT 1, L
CB4E	BIT 1, (HL)
CB4F	BIT 1, A
CB50	BIT 2, B
CB51	BIT 2, C
CB52	BIT 2, D
CB53	BIT 2, E
CB54	BIT 2, H
CB55	BIT 2, L
CB56	BIT 2, (HL)
CB57	BIT 2, A
CB58	BIT 3, B
CB59	BIT 3, C
CB5A	BIT 3, D
CB5B	BIT 3, E
CB5C	BIT 3, H
CB5D	BIT 3, L
CB5E	BIT 3, (HL)
CB5F	BIT 3, A
CB60	BIT 4, B
CB61	BIT 4, C
CB62	BIT 4, D
CB63	BIT 4, E
CB64	BIT 4, H
CB65	BIT 4, L
CB66	BIT 4, (HL)
CB67	BIT 4, A
CB68	BIT 5, B
CB69	BIT 5, C
CB6A	BIT 5, D
CB6B	BIT 5, E
CB6C	BIT 5, H
CB6D	BIT 5, L
CB6E	BIT 5, (HL)
CB6F	BIT 5, A
CB70	BIT 6, B
CB71	BIT 6, C
CB72	BIT 6, D
CB73	BIT 6, E
CB74	BIT 6, H
CB75	BIT 6, L
CB76	BIT 6, (HL)
CB77	BIT 6, A
CB78	BIT 7, B
CB79	BIT 7, C
CB7A	BIT 7, D
CB7B	BIT 7, E
CB7C	BIT 7, H
CB7D	BIT 7, L
CB7E	BIT 7, (HL)
CB7F	BIT 7, A
CB80	RES 0, B
CB81	RES 0, C
CB82	RES 0, D
CB83	RES 0, E
CB84	RES 0, H
CB85	RES 0, L
CB86	RES 0, (HL)
CB87	RES 0, A
CB88	RES 1, B
CB89	RES 1, C
CB8A	RES 1, D
CB8B	RES 1, E
CB8C	RES 1, H
CB8D	RES 1, L
CB8E	RES 1, (HL)
CB8F	RES 1, A
CB90	RES 2, B
CB91	RES 2, C
CB92	RES 2, D
CB93	RES 2, E
CB94	RES 2, H
CB95	RES 2, L
CB96	RES 2, (HL)
CB97	RES 2, A
CB98	RES 3, B
CB99	RES 3, C
CB9A	RES 3, D
CB9B	RES 3, E
CB9C	RES 3, H
CB9D	RES 3, L
CB9E	RES 3, (HL)
CB9F	RES 3, A
CBA0	RES 4, B

CBA1	RES 4, C
CBA2	RES 4, D
CBA3	RES 4, E
CBA4	RES 4, H
CBA5	RES 4, L
CBA6	RES 4, (HL)
CBA7	RES 4, A
CBA8	RES 5, B
CBA9	RES 5, C
CBAA	RES 5, D
CBAB	RES 5, E
CBAC	RES 5, H
CBAD	RES 5, L
CBAE	RES 5, (HL)
CBAF	RES 5, A
CBB0	RES 6, B
CBB1	RES 6, C
CBB2	RES 6, D
CBB3	RES 6, E
CBB4	RES 6, H
CBB5	RES 6, L
CBB6	RES 6, (HL)
CBB7	RES 6, A
CBB8	RES 7, B
CBB9	RES 7, C
CBBA	RES 7, D
CBBB	RES 7, E
CBBC	RES 7, H
CBBD	RES 7, L
CBBE	RES 7, (HL)
CBBF	RES 7, A
CBC0	SET 0, B
CBC1	SET 0, C
CBC2	SET 0, D
CBC3	SET 0, E
CBC4	SET 0, H
CBC5	SET 0, L
CBC6	SET 0, (HL)
CBC7	SET 0, A
CBC8	SET 1, B
CBC9	SET 1, C
CBCA	SET 1, D
CBCB	SET 1, E
CBCC	SET 1, H
CBCD	SET 1, L
CBCE	SET 1, (HL)
CBCF	SET 1, A
CBD0	SET 2, B
CBD1	SET 2, C
CBD2	SET 2, D
CBD3	SET 2, E
CBD4	SET 2, H
CBD5	SET 2, L
CBD6	SET 2, (HL)
CBD7	SET 2, A
CBD8	SET 3, B
CBD9	SET 3, C
CBDA	SET 3, D
CBDB	SET 3, E
CBDC	SET 3, H
CBDD	SET 3, L
CBDE	SET 3, (HL)
CBDF	SET 3, A
CBE0	SET 4, B
CBE1	SET 4, C
CBE2	SET 4, D
CBE3	SET 4, E
CBE4	SET 4, H
CBE5	SET 4, L
CBE6	SET 4, (HL)
CBE7	SET 4, A
CBE8	SET 5, B
CBE9	SET 5, C
CBEA	SET 5, D
CBEB	SET 5, E
CBEC	SET 5, H
CBED	SET 5, L
CBEE	SET 5, (HL)
CBEF	SET 5, A
CBF0	SET 6, B
CBF1	SET 6, C
CBF2	SET 6, D
CBF3	SET 6, E
CBF4	SET 6, H
CBF5	SET 6, L
CBF6	SET 6, (HL)
CBF7	SET 6, A
CBF8	SET 7, B
CBF9	SET 7, C
CBFA	SET 7, D
CBFB	SET 7, E
CBFC	SET 7, H
CBFD	SET 7, L
CBFE	SET 7, (HL)
CBFF	SET 7, A
CC n n	CALL Z, nn
CD n n	CALL nn
CE n	ADC A, n
CF	RST 8h
D0	RET NC
D1	POP DE
D2 n n	JP NC, nn
D3 n	OUT (n), A
D4 n n	CALL NC, nn
D5	PUSH DE
D6 n	SUB n
D7	RST 10h
D8	RET C
DA n n	JP C, nn
DB n	IN A, (n)
DC n n	CALL C, nn
DD09	ADD IX, BC
DD19	ADD IX, DE
DD21 n n	LD IX, nn
DD22 n n	LD (nn), IX
DD23	INC IX
DD29	ADD IX, IX
DD2A n n	LD IX, (nn)
DD2B	DEC IX
DD34 d	INC (IX + d)
DD35 d	DEC (IX + d)
DD36 d n	LD (IX + d), n
DD39	ADD IX, SP
DD46 d	LD B, (IX + d)
DD4E d	LD C, (IX + d)
DD56 d	LD D, (IX + d)
DD5E d	LD E, (IX + d)
DD66 d	LD H, (IX + d)
DD6E d	LD L, (IX + d)
DD70 d	LD (IX + d), B
DD71 d	LD (IX + d), C
DD72 d	LD (IX + d), D
DD73 d	LD (IX + d), E
DD74 d	LD (IX + d), H
DD75 d	LD (IX + d), L
DD77 d	LD (IX + d), A
DD7E d	LD A, (IX + d)
DD86 d	ADD A, (IX + d)
DD8E d	ADC A, (IX + d)
DD96 d	SUB (IX + d)
DD9E d	SBC A, (IX + d)
DDA6 d	AND (IX + d)
DDAE d	XOR (IX + d)
DDB6 d	OR (IX + d)
DDBE d	CP (IX + d)
DDCB d 06	RLC (IX + d)
DDCB d 0E	RRC (IX + d)
DDCB d 16	RL (IX + d)
DDCB d 1E	RR (IX + d)
DDCB d 26	SLA (IX + d)
DDCB d 2E	SRA (IX + d)
DDCB d 3E	SRL (IX + d)
DDCB d 46	BIT 0, (IX + d)
DDCB d 4E	BIT 1, (IX + d)
DDCB d 56	BIT 2, (IX + d)
DDCB d 5E	BIT 3, (IX + d)
DDCB d 66	BIT 4, (IX + d)
DDCB d 6E	BIT 5, (IX + d)
DDCB d 76	BIT 6, (IX + d)
DDCB d 7E	BIT 7, (IX + d)
DDCB d 86	RES 0, (IX + d)
DDCB d 8E	RES 1, (IX + d)
DDCB d 96	RES 2, (IX + d)
DDCB d 9E	RES 3, (IX + d)
DDCB d A6	RES 4, (IX + d)
DDCB d AE	RES 5, (IX + d)
DDCB d B6	RES 6, (IX + d)
DDCB d BE	RES 7, (IX + d)
DDCB d C6	SET 0, (IX + d)
DDCB d CE	SET 1, (IX + d)
DDCB d D6	SET 2, (IX + d)
DDCB d DE	SET 3, (IX + d)
DDCB d E6	SET 4, (IX + d)
DDCB d EE	SET 5, (IX + d)
DDCB d F6	SET 6, (IX + d)
DDCB d FE	SET 7, (IX + d)
DDE1	POP IX
DDE5	PUSH IX
DDE9	JP (IX)
DDF9	LD SP, IX

DE n	SBC A, n
DF	RST 18h
E0	RET PO
E1	POP HL
E2 n n	JP PO, nn
E4 n n	CALL PO, nn
E5	PUSH HL
E6 n	AND n
E7	RST 20h
E8	RET PE
E9	JP (HL)
EA n n	JP PE, nn
EC n n	CALL PE, nn
ED40	IN B, (C)
ED41	OUT (C), B
ED42	SBC HL, BC
ED43 n n	LD (nn), BC
ED44	NEG
ED48	IN C, (C)
ED49	OUT (C), C
ED4A	ADC HL, BC
ED4B n n	LD BC, (nn)
ED50	IN D, (C)
ED51	OUT (C), D
ED52	SBC HL, DE
ED53 n n	LD (nn), DE
ED58	IN E, (C)
ED59	OUT (C), E
ED5A	ADC HL, DE
ED5B n n	LD DE, (nn)
ED60	IN H, (C)
ED61	OUT (C), H
ED62	SBC HL, HL
ED63 n n	LD (nn), HL
ED67	RRD
ED68	IN L, (C)
ED69	OUT (C), L
ED6A	ADC HL, HL
ED6F	RLD
ED72	SBC HL, SP
ED73 n n	LD (nn), SP
ED78	IN A, (C)
ED79	OUT (C), A
ED7A	ADC HL, SP
ED7B n n	LD SP, (nn)
EE n	XOR n
EF	RST 28h
F0	RET P
F1	POP AF
F2 n n	JP P, nn
F3	DI
F4 n n	CALL P, nn
F5	PUSH AF
F6 n	OR n
F7	RST 30h
F8	RET M
F9	LD SP, HL
FA n n	JP M, nn
FB	EI
FC n n	CALL M, nn
FD09	ADD IY, BC
FD19	ADD IY, DE
FD21 n n	LD IY, nn
FD22 n n	LD (nn), IY
FD23	INC IY
FD29	ADD IY, IY
FD2A n n	LD IY, (nn)
FD2B	DEC IY
FD34 d	INC (IY + d)
FD35 d	DEC (IY + d)
FD36 d n	LD (IY + d), n
FD39	ADD IY, SP
FD46 d	LD B, (IY + d)
FD4E d	LD C, (IY + d)
FD56 d	LD D, (IY + d)
FD5E d	LD E, (IY + d)
FD66 d	LD H, (IY + d)
FD6E d	LD L, (IY + d)
FD70 d	LD (IY + d), B
FD71 d	LD (IY + d), C
FD72 d	LD (IY + d), D
FD73 d	LD (IY + d), E
FD74 d	LD (IY + d), H
FD75 d	LD (IY + d), L
FD77 d	LD (IY + d), A
FD7E d	LD A, (IY + d)
FD86 d	ADD A, (IY + d)
FD8E d	ADC A, (IY + d)
FD96 d	SUB (IY + d)
FD9E d	SBC A, (IY + d)
FDA6 d	AND (IY + d)
FDAE d	XOR (IY + d)
FDB6 d	OR (IY + d)
FDBE d	CP (IY + d)
FDCB d 06	RLC (IY + d)
FDCB d 0E	RRC (IY + d)
FDCB d 16	RL (IY + d)
FDCB d 1E	RR (IY + d)
FDCB d 26	SLA (IY + d)
FDCB d 2E	SRA (IY + d)
FDCB d 3E	SRL (IY + d)
FDCB d 46	BIT 0, (IY + d)
FDCB d 4E	BIT 1, (IY + d)
FDCB d 56	BIT 2, (IY + d)
FDCB d 5E	BIT 3, (IY + d)
FDCB d 66	BIT 4, (IY + d)
FDCB d 6E	BIT 5, (IY + d)
FDCB d 76	BIT 6, (IY + d)
FDCB d 7E	BIT 7, (IY + d)
FDCB d 86	RES 0, (IY + d)
FDCB d 8E	RES 1, (IY + d)
FDCB d 96	RES 2, (IY + d)
FDCB d 9E	RES 3, (IY + d)
FDCB d A6	RES 4, (IY + d)
FDCB d AE	RES 5, (IY + d)
FDCB d B6	RES 6, (IY + d)
FDCB d BE	RES 7, (IY + d)
FDCB d C6	SET 0, (IY + d)
FDCB d CE	SET 1, (IY + d)
FDCB d D6	SET 2, (IY + d)
FDCB d DE	SET 3, (IY + d)
FDCB d E6	SET 4, (IY + d)
FDCB d EE	SET 5, (IY + d)
FDCB d F6	SET 6, (IY + d)
FDCB d FE	SET 7, (IY + d)
FDE1	POP IY
FDE5	PUSH IY
FDE9	JP (IY)
FDF9	LD SP, IY
FE n	CP n
FF	RST 38h

Appendix: *Deeds* historical version notes

(Notes are reported in time reverse order).

20 - 01 - 2004

d-McE

- It has been solved a bug of the **"Save As"** command: now, if you press the **'Cancel'** button of the 'Save As' standard dialog, the **Close operations**, if running, are **aborted** as expected.
- The execution of the **RRCA instruction** has been fixed.
- The **"I/O Port Address" dialog**, on computers with the video card configured in a low-resolution mode, **did not appear**. The problem has been fixed, the dialog now will show always (centred on the main window).

11 - 11 - 2003

Deeds, d-AsT

- Now, when **a new version** of the Deeds is installed, the browser home pages are **reset to the defaults**, to avoid confusion between the different Deeds versions. However, the address of the previous user home page is not lost: it will be found in the history list of the opened pages, in the 'open' window.

d-DcS

- An error in simulation of **finite state machine** components has been fixed (the behaviour of the network when the FSM Reset input is activated at time=0).
- Now, when you start the **'interactive' simulation**, the input switches are initialised according to the assigned **names**: as in the 'timing' simulation mode, **the initial value** will be set to **'one'** if the name represents an **'active low'** signal (i.e. the name is negated). As a consequence, for instance, all the components that require a (low activated) reset now will start **un-initialised**, showing **'unknown' outputs** until the user will reset them expressly. When you exit the 'interactive' simulation all inputs an outputs reset to their default status.
- Now, **during** the **'timing'** simulation, the circuit in the editor shows **input and output value** coherently with **simulation results**. You can observe the input/output status of the circuit in the editor **before and after each simulation step**.
- Now, when you **print** the circuit, or **copy** it **as image** on the clipboard, the resulting **picture is coherent** with the **input/output values** currently displayed in the editor.
- The **maximum time for simulation** has been fixed, now it is no more limited to 32678 nS.

d-FsM

- Now, when timing simulation window is iconized and the simulation closed, the bar buttons are correctly enabled and updated.
- Now, when editor and timing windows are in iconized or maximized state, and the user closes them, their 'normal' position, instead of the currently one, is saved. In this way, when the user will re-open the windows, these will be placed in their 'normal' position. The correction has been done to reduce flickering and flashing of the windows on the screen.

04 - 11 - 2003

d-DcS

- An error in simulation of decoder components has been fixed.

d-McE

- In the debugger OBJECT CODE frame, the memory 'extended view' command has been fixed: now, in this mode, all the micro-processor memory space is shown.
- The ASCII table page, in the On Line Help, has been corrected.

15 - 10 - 2003

d-McE

- A new simulation tool has been added to the Deeds: the **Micro-Computer Emulator (d-McE)**.
- The functionally emulated board include a CPU, ROM and RAM memory, parallel I/O ports, reset circuitry and a simple interrupt logic.
- The custom 8 bit CPU, named DMC8, has been designed to suite our educational needs, and it is based on a simplified version of the well-known 'Z80' processor.
- The d-McE integrates a Code Editor, an Assembler and a machine-level interactive Debugger.
- The integrated source code editor enables user to enter assembly programs, and a simple command permits to assemble, link and load them in the emulated system memory.
- The execution of the programs can be run step by step in the interactive debugger, where the user can observe all the structures involved in the hardware/software system
- By now, the integration of this tool with the Deeds is not complete: the board can not be inserted in the d-DcS (yet).

d-DcS

- The simulation kernel code has been completely revised, and its code linked with the executable. The current version doesn't need the installation of the ActiveX that the previous versions do, and some mistake in the simulation of complex components has been fixed.
- The 'Delete' (by Selection) command has been fixed.
- Some other minor bugs have been corrected.

11 - 07 - 2003

d-FsM

- Now the d-FsM tool can **export** the finite state machine in **VHDL format**. The command is available under the 'File' menu item. The user can view the **VHDL code**, copy it on the clipboard, or save it on a text file ('.vhd').
- The '**State at Reset**' is **highlighted** with a little **diamond**, placed on the top-left of the state block. A 'starting' state block is now accepted (i.e., a state without a connection over it, normally used as 'State at Reset', or to drive the FSM, through the unused states, to a 'safe' state).
- The **graphic editor** has been **radically modified**. Now blocks and lines can't be inserted or moved over other blocks and lines: this is highlighted in the editor by displaying a red 'denied' signal when appropriate.
- The '**selection**' rectangle is now **re-sizable**, 'grip points' are available to move the four vertexes with the mouse.
- Now it is possible to show/hide the '**drawing grid**' (the command is under the 'View' menu item).
- In the editor the user can use the new **Zoom commands**; they permit an easier editing of the ASM diagrams (the commands are under the 'View' menu item and on the toolbar).

- A new feature of the editor permits the controls of insertion, moving and editing of lines. This feature automatically breaks (or links together) the lines, as they are inserted, moved or edited. The criteria are to connect line segments only on their vertexes. In this way all the previous bugs, related a 'lateral' ('T' shaped) connection between lines, have been fixed.
- The algorithm that **automatically** assigns the **code** to a 'newly created' state has been modified. Now it assigns to the state the first not-used code (checking it from the lowest code available). If no state is deleted, this mode of operation is equivalent to assign codes in up-order. If a state is deleted, its code will be re-used. If a code value is no more available, the user, when trying to insert a new state block, is prompted to add another variable to the state register.
- During insertion, moving or editing of ASM blocks and lines, if the user presses the **<escape>** key, **the current operation** is automatically **aborted**.
- The <delete> key now acts on the 'key-down' instead of the 'key-up' (conforming its behaviour to all the Windows application).
- In the IN/OUT dialog, pressing the <Return> key generates **no more** a tedious **beep**; also, in the same dialog, it is now possible to edit, as expected, the eighth Input (or Output) entry.
- Now it is possible to design **a FSM** having **no input signal** (for instance, a simple binary counter).

d-DcS

- In the previous version, a click on the 'Cancel' button of the message dialog that appears when you want to create a new Finite State Machine hanged the d-DcS. The problem has been fixed.

13 - 05 - 2003

d-FsM

- 'Cascade' connection of more than two conditional blocks do not hang the program anymore: the bug has been fixed. Now the program seems also to process correctly conditional blocks connected in a 'nonsense' mode.
- Some algorithmic optimization has been done, so the program now is faster that before (during redrawing, when the diagram is 'big').
- The **Properties** window now shows correctly for all the screen resolution. Now the user controls its **visibility** and the visibility is **remembered** between work sessions.
- The In/Out dialog now *remembers,* between work sessions, which page was last opened.
- The timing window doesn't ask the user anymore, if no simulation has been started; instead, now the program prompts the user on a "clear diagram" request, if data could be lost.
- Drawing of output names, in the state and conditional output blocks has been enhanced. They are displayed from left to right, on two lines. If some output name can not be displayed, for lack of space, a '+' sign appears after the last one, on the second line, to notify the user that more output have been assigned to the block, but that they can not be displayed. Anyway, the complete output list is visible on the bottom status bar, when the user points the mouse over the block of interest.
- The algorithm that shows the arrows on the lines has been enhanced, and the arrow shape modified.
- Drawing of the input name in the decisional blocks has been horizontally centered.

d-DcS

- Drawing of input and output pin names, in the FSM components, has been enhanced. To avoid overlapping of 'long' names, names too long are shorted (at display level).

24 - 02 - 2003

d-AsT

- A little bug was fixed: it occurred during page loading in the Assistant Browser.

20 - 02 - 2003

All

- ***The Deeds and Assistant browsers are now enabled*** and specialized to 'run' the Deeds learning material (as well ordinary HTML pages). During operation, these browsers decode the so-called ***Deeds Commands***, that the author of a lesson (or laboratory session) include in the HTML page to enable interactivity between the HTML page and each Deeds tool included in the suite.
- Now it is possible to **open a file downloading it from internet**. This command is intended to be driven by the Deeds browser, when the user clicks on an active link, to open a file.

Deeds

- Now, when a Tool is launched, the "Splash Form" is displayed only the firth time.
- The problem of 'double launch' of Deeds when you start it from the Application Bar has been solved.
- For debug reasons, a "hard close" command has been added. If could be necessary, you may close the Deeds main application (without closing also the other tools), activating the ordinary "File/Close" command while pressing the <Shift> and <Control> keys.

d-DcS

- Now the title of the Timing Diagram window shows the current timing simulation mode (the modes enabled are, by now: the **Incremental Interactive Simulation** mode [**IIS**], and the **Timing Interval Simulation** mode [**TIS**].
- Now it is possible to **open a file downloading it from internet**. This command is intended to be driven by the Deeds browser, when the user clicks on an active link, to open a file.
- The **warning messages of the simulator**, when needed, are displayed in a list at **the bottom of the main window** (if in 'animation' mode), **or at the bottom of the timing diagram window** (if in 'timing' mode). In this way the messages results more kind to the user than before, when each message was displayed by a dialog box.
- Now, **the last status bar message** can be displayed moving the mouse over the bar itself.

In the **Timing Diagram**, the **highlighting of the transitions** of a specific signal has been **enhanced**. By clicking the button corresponding to a specific signal, **you can toggle among four highlight modes**: **a)** vertical lines on 0→1 transitions only; **b)** vertical lines on 1→0 transitions only; **c)** vertical line on all transition; **d)** no highlight.

d-FsM

- **Drag and Drop** of FSM files from the file manager has been enabled.

10 - 05 - 2002

d-DcS

- Finite State Machine components, when not completed, cann't be loaded in the d-DcS. This is OK, but under some circumstances, the user message explaining that the file wasn't completed didn't appear. This problem has been fixed.

d-FsM

- The Print command has been fixed, now it is possible to print on paper ASM diagrams.
- Some file/save file/open bugs has been fixed.
- The “Save” file commands have been completed with automatic file backup generation: for instance, before saving file “name.fsm”, if a previous version of the file exists, this is renamed in “name.˜fsm“.
- The File/Close command has been fixed: it no more operational if no file is opened.
- A known problem has not been fixed yet: under some circumstances, ‘big’ ASM diagrams can show a sensible slowing of window repainting.

22 - 03 - 2002

d-DcS

- Now, in the schematic editor, labeling of the component is allowed only for Input/Output components. Attempt to label another kind of component results in a warning message on the status bar of the window. You can set a negation bar over Input/Output component labels placing a ‘!’ on front of the label string. The editor accepts also a leading ‘/’ or ‘\’, but the ‘!’ has to be preferred. Moreover, the negation bar is now displayed also on the signal name in the timing diagram window.
- Drawing of Finite State Machine components has been (partially) fixed for those components placed with “down” and “up” orientation. Before this fix, displayed name of inputs and outputs were not the right ones. Anyway, we suggest to not use “down” and “up” orientation for Finite State Machines, as name strings could easily overlap.
- In the Timing Diagram, now signal names are “buttons” evidenced by proper glyphs and colors, and negated signal are displayed up-lined.
- You can highlight the transitions of a specific signal with a click on its name button (in this way, you can relate its transitions with the behaviour of the network under simulation).
- If you click with the mouse right button on a signal name button, you activate a context menu. Context Menus allow to move up or down the signal traces and to set signal properties. For instance, you can change clock period and initial value for clock inputs, and initial value of the ordinary input signals can be set.
- In the Timing Diagram, activating simulation and/or signal editing without to release the “Time Stop” cursor is now inhibited, avoiding the bug of losing the cursor.
- Now the “F8” short cut not only sets the Timing Diagram Simulation mode, but also put the Timing Diagram window on the Top if already present. Instead, if the Timing Diagram window is on Top, the “F8” short cut gets the main window on Top again.
- The problem of redrawing of the vertical lines used as cursors in the timing diagram has been fixed (before, when “hint” messages of buttons were displayed and the mouse moved away through the diagram, some pieces of lines remained on the screen).
- When timing simulation is active, editing of circuit is now actually inhibited (the "out of bound" error has been eliminated).
- Now, before to simulate, the application asks the user for saving the file of the circuit, if the file has been modified.
- The internal ”simulation loop” has been enhanced, making timing simulation faster.
- Now it is possible to break simulation when tired to wait for long times, by clicking on the Stop button. You’ll be requested to confirm breaking.
- During long simulations, a percentage of the work is displayed on the status bar, at bottom.
- Finite State Machine simulation has been fixed and enhanced: now, at simulation start, their state is considered “undefined” until the Reset signal is activated, as expected. However, due to limitation of the model used, by now the outputs are considered always “unknown” until state stay undefined, without taking in count conditions from the inputs.

d-FsM

- Now it is possible to restore correctly the application windows after having closed them in the maximized state.
- Now, some commands no more generate errors when activated in absence of opened windows.
- Some error message revised, some others translated in English.
- The Print command has been disabled, waiting a major fix of the printing module.
- A few minor bugs have been fixed.

Deeds

- The main browser is not yet fully operational, but a link to the ***Deeds Web Site*** has been added in the demo page.

01 - 03 - 2002

d-DcS

- Added the ability to copy on the Clipboard the Timing Diagram current view.
- Now, in the Timing Diagram, you can highlight the transitions of a specific signal with a click on its button; in this way, you can relate its transitions with the behaviour of the network under simulation.
- A few bugs have been fixed.

22 - 02- 2002 (and before)

- Released for internal beta test only.