Interdisciplinary
Engineering Design Education Conference
IEDEC 2013
March 4-5, 2013
Santa Clara, CA USA

# From Gates to FPGA: Learning Digital Design with Deeds

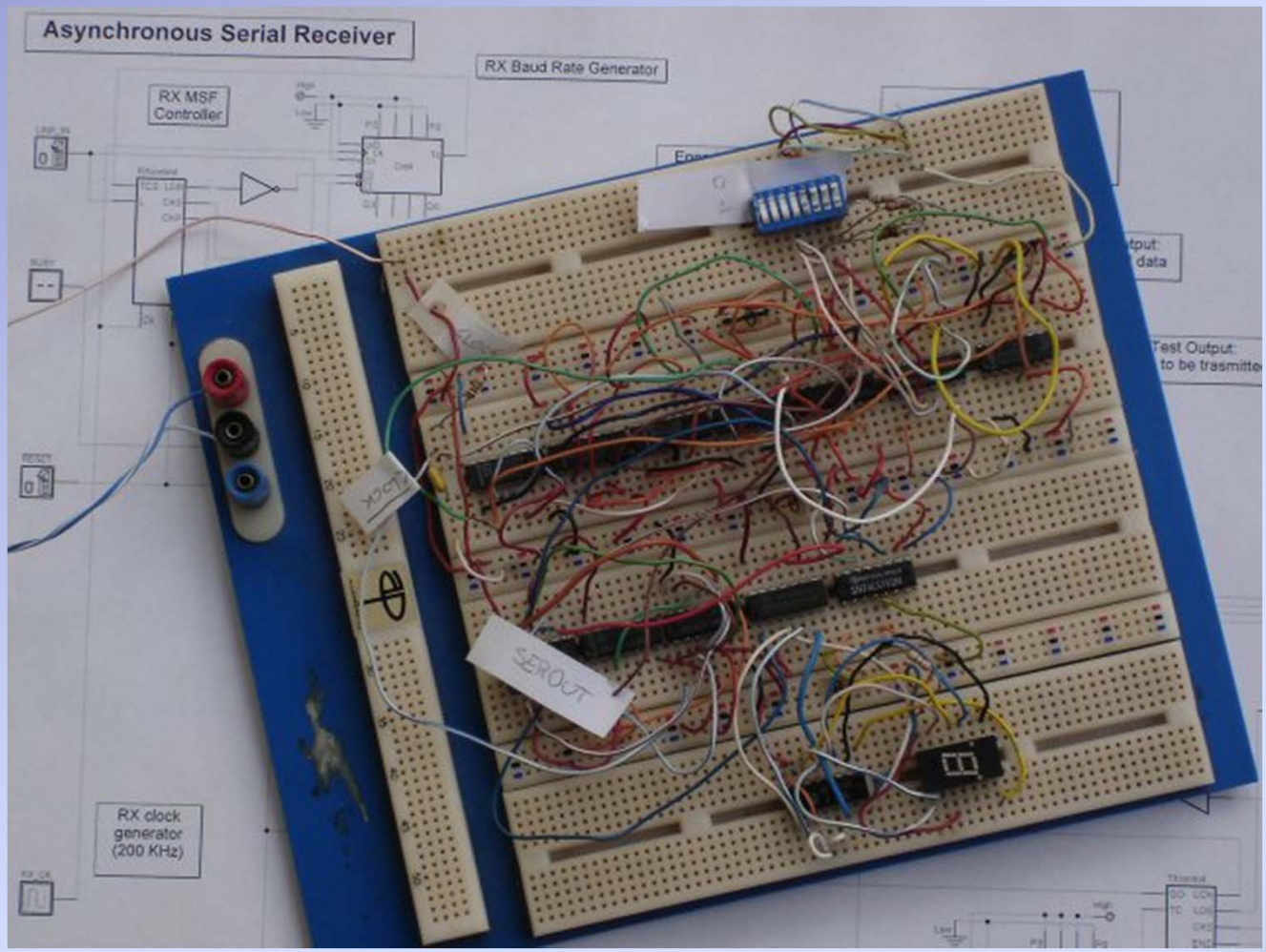
***Giuliano Donzellini, Domenico Ponta***

*giuliano.donzellini@unige.it* , *domenico.ponta@unige.it*

*DITEN – University of Genoa, Italy*

The software tools presented are available
free of charge to all interested parties at:
**http://www.esng.dibe.unige.it/deeds/**

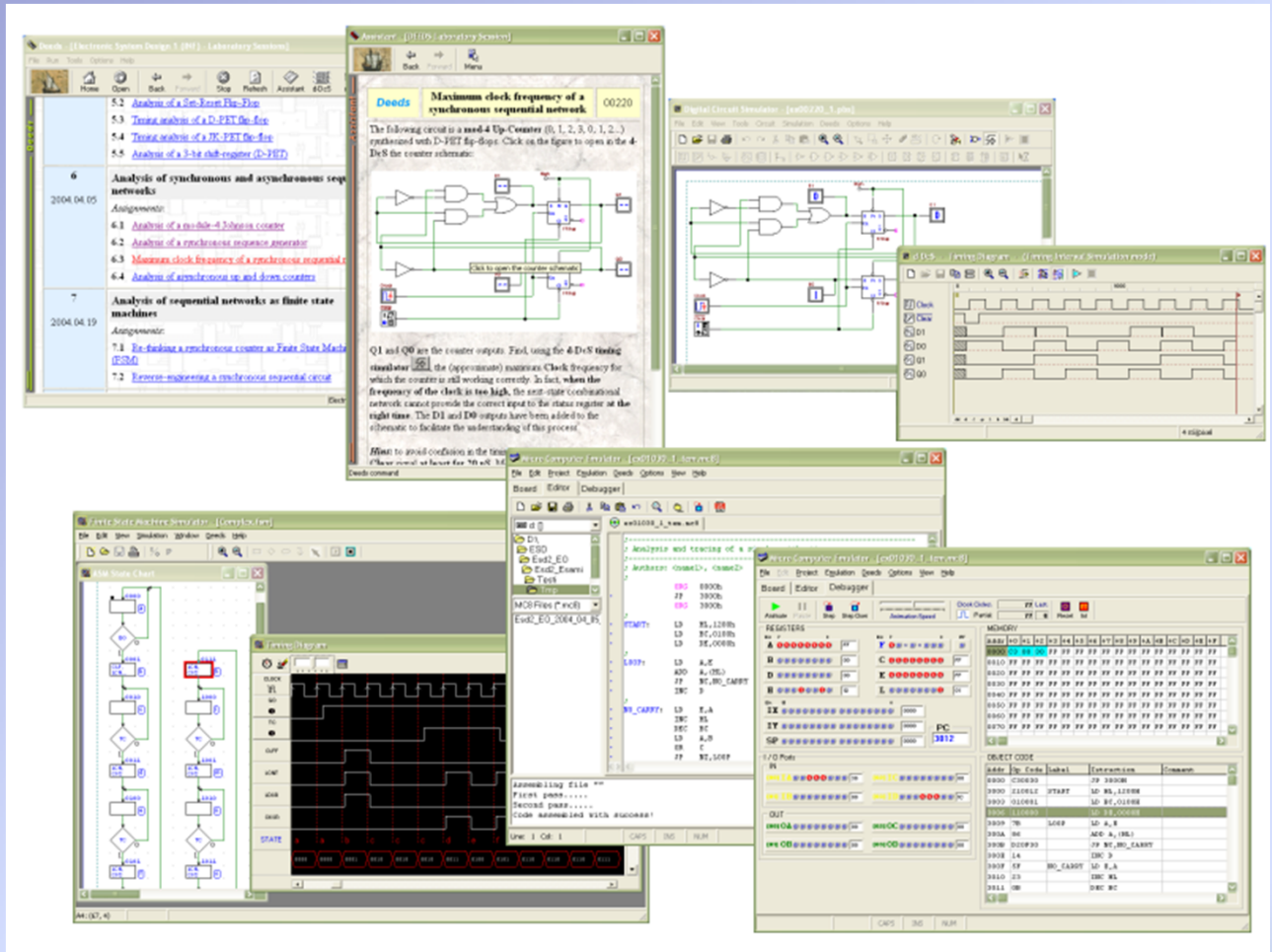
# *Learning digital design: yesterday*



# *The traditional "white breadboard"*

- In the 70' and later on, laboratory activities were based on the construction and testing of circuit using TTL ICs on a solderless breadboard.
- Most of the laboratory time was used for the construction of the circuit, often critical because of faulty components and bad electrical contacts.
- Required familiarity with lab instrumentation
- Design and implementation technologies are now obsolete

# *Next: circuit simulation (Virtual lab)*

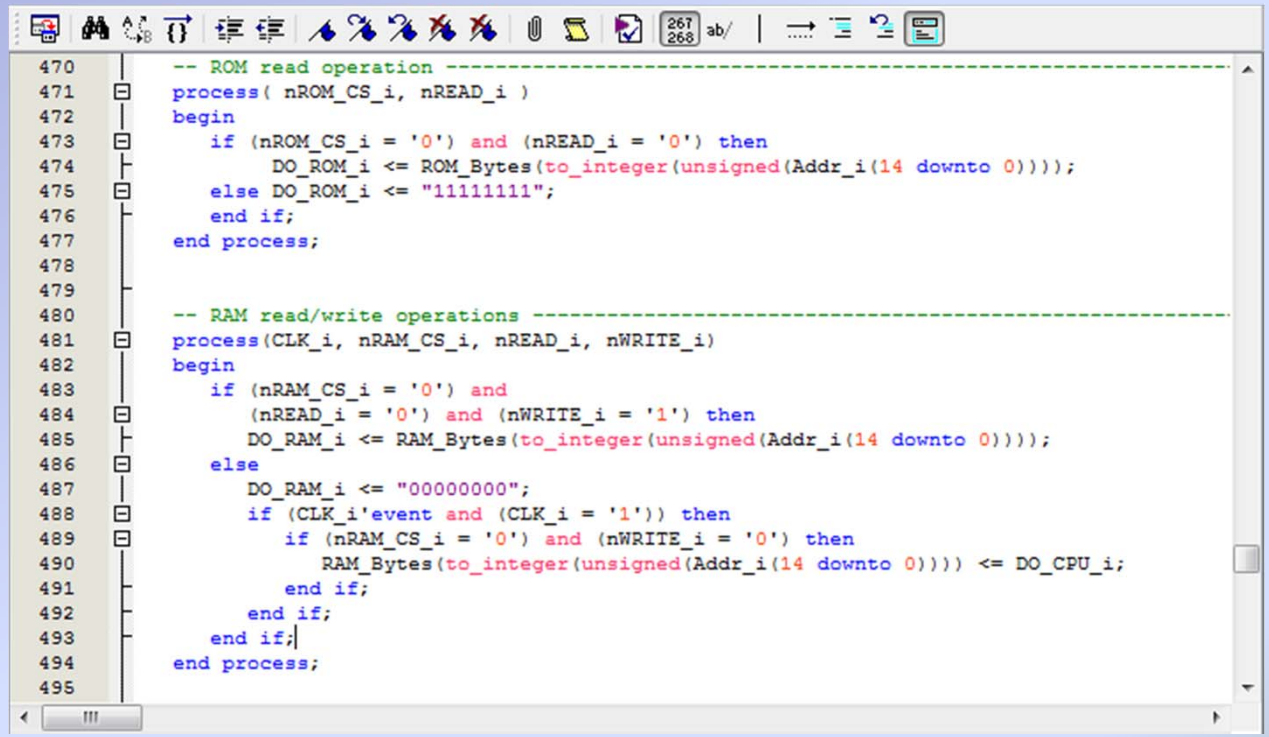


# *Digital circuit simulation (Virtual lab)*

- Faster and easier “construction” of the circuit
- Time in lab spent in a more productive way
- Possibility of remotization of the laboratory activities
- ***But: no physical implementation, no real measurements possible***

# *Digital systems today*

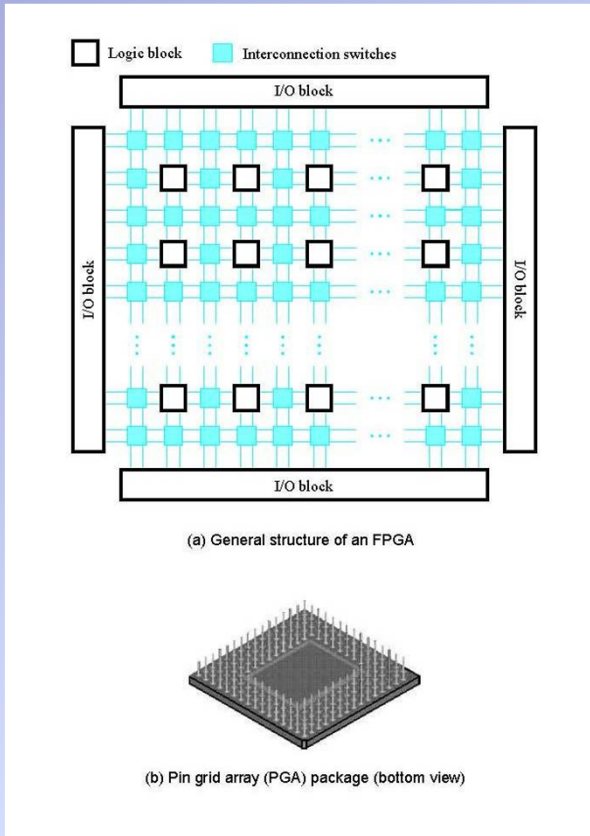
- Electronic Design Automation (EDA) and Hardware Description Languages (HDL) are the core of a digital system development process.

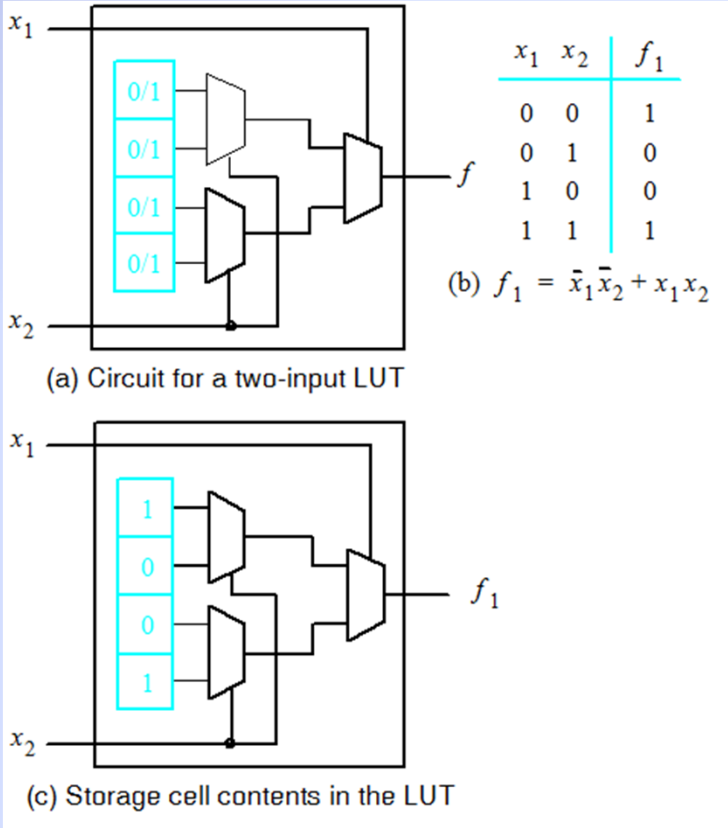


```
470   -- ROM read operation ----------------------------------------------------------
471   process( nROM_CS_i, nREAD_i )
472   begin
473      if (nROM_CS_i = '0') and (nREAD_i = '0') then
474           DO_ROM_i <= ROM_Bytes(to_integer(unsigned(Addr_i(14 downto 0))));
475      else DO_ROM_i <= "11111111";
476      end if;
477   end process;
478
479
480   -- RAM read/write operations -----------------------------------------------------
481   process(CLK_i, nRAM_CS_i, nREAD_i, nWRITE_i)
482   begin
483      if (nRAM_CS_i = '0') and
484         (nREAD_i = '0') and (nWRITE_i = '1') then
485         DO_RAM_i <= RAM_Bytes(to_integer(unsigned(Addr_i(14 downto 0))));
486      else
487         DO_RAM_i <= "00000000";
488         if (CLK_i'event and (CLK_i = '1')) then
489            if (nRAM_CS_i = '0') and (nWRITE_i = '0') then
490               RAM_Bytes(to_integer(unsigned(Addr_i(14 downto 0)))) <= DO_CPU_i;
491            end if;
492         end if;
493      end if;
494   end process;
495
```

# *FPGA and digital design (1)*

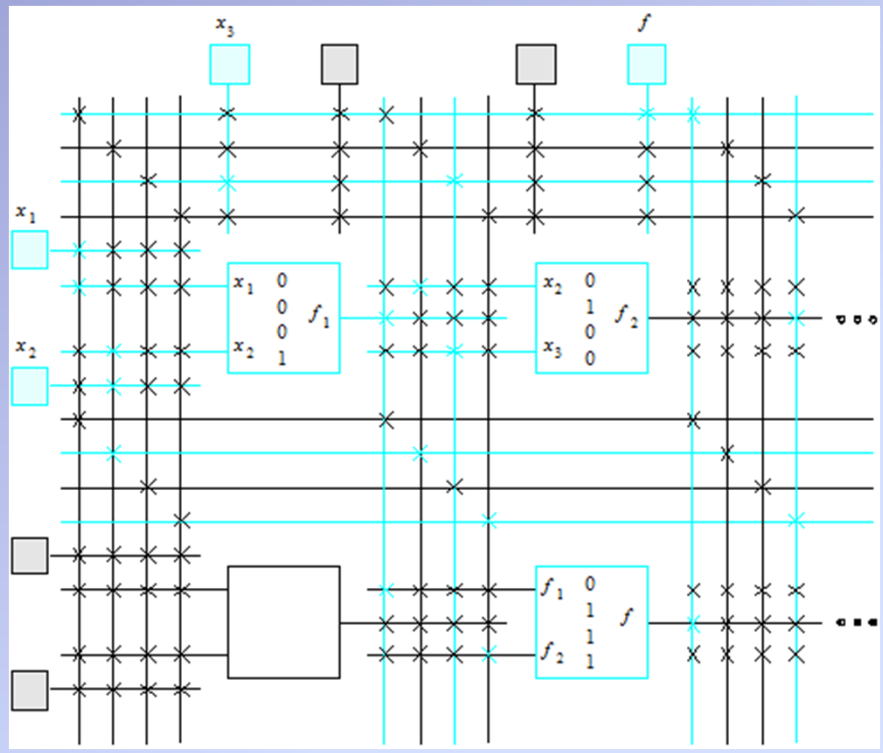
- A growing number of digital systems are based on FPGAs, very large arrays of simple logic elements (LUT) with programmable connections.

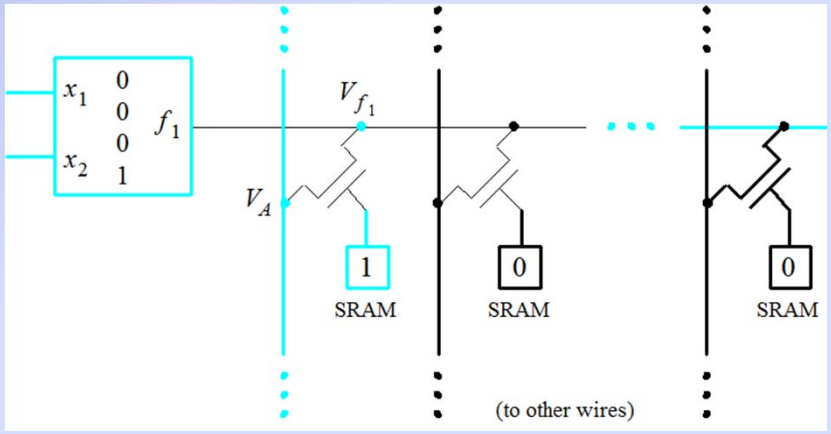


# *FPGA and digital design (2)*

- In the FPGA, connections among the blocks are defined by data stored in a SRAM. They can be configured to implement any hardware device, including processors.




# *Learning digital design today*

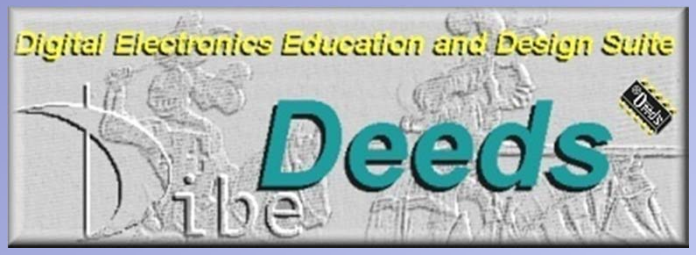
- The complexity of today's digital systems is putting new demands on education.
- A growing number of teachers introduce HDL, FPGA and professional EDA tools in basic, introductory courses.
- Such approach has many advantages but it may hide from learners important basic issues.

# *Learning digital design: the challenge*

- EDA tools are made for professionals, not for beginners.
- Introductory design education can still take advantage from the traditional approach based on logic symbols and schematics.
- But some familiarity with HDL and FPGA techniques is a necessary target of a digital design course, even an introductory one.
- We strive to achieve continuity between the traditional approach and the new design techniques.

# Deeds: Digital Electronics Education and Design Suite

Digital Electronics Education and Design Suite



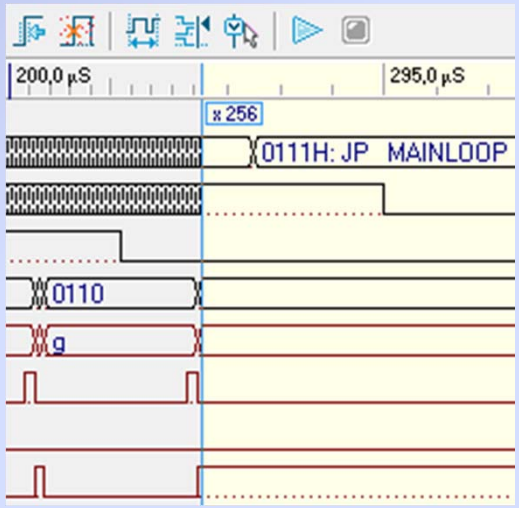

- ***Deeds*** is developed at DITEN (ex DIBE), University of Genoa
- The suite is composed by three simulators and a wide collection of associated ***learning material*** to learn-by-doing and practice with:
  - ***Combinational and sequential logic networks***
  - ***Finite state machine design***
  - ***Embedded microcomputer interfacing and programming***
  - ***FPGA programming (exporting projects to EDA tools)***

# *Learning the basics: the Deeds approach*

- ***Deeds*** embodies our pedagogical approach to teaching and learning digital design: building a solid understanding of the principle of digital design.

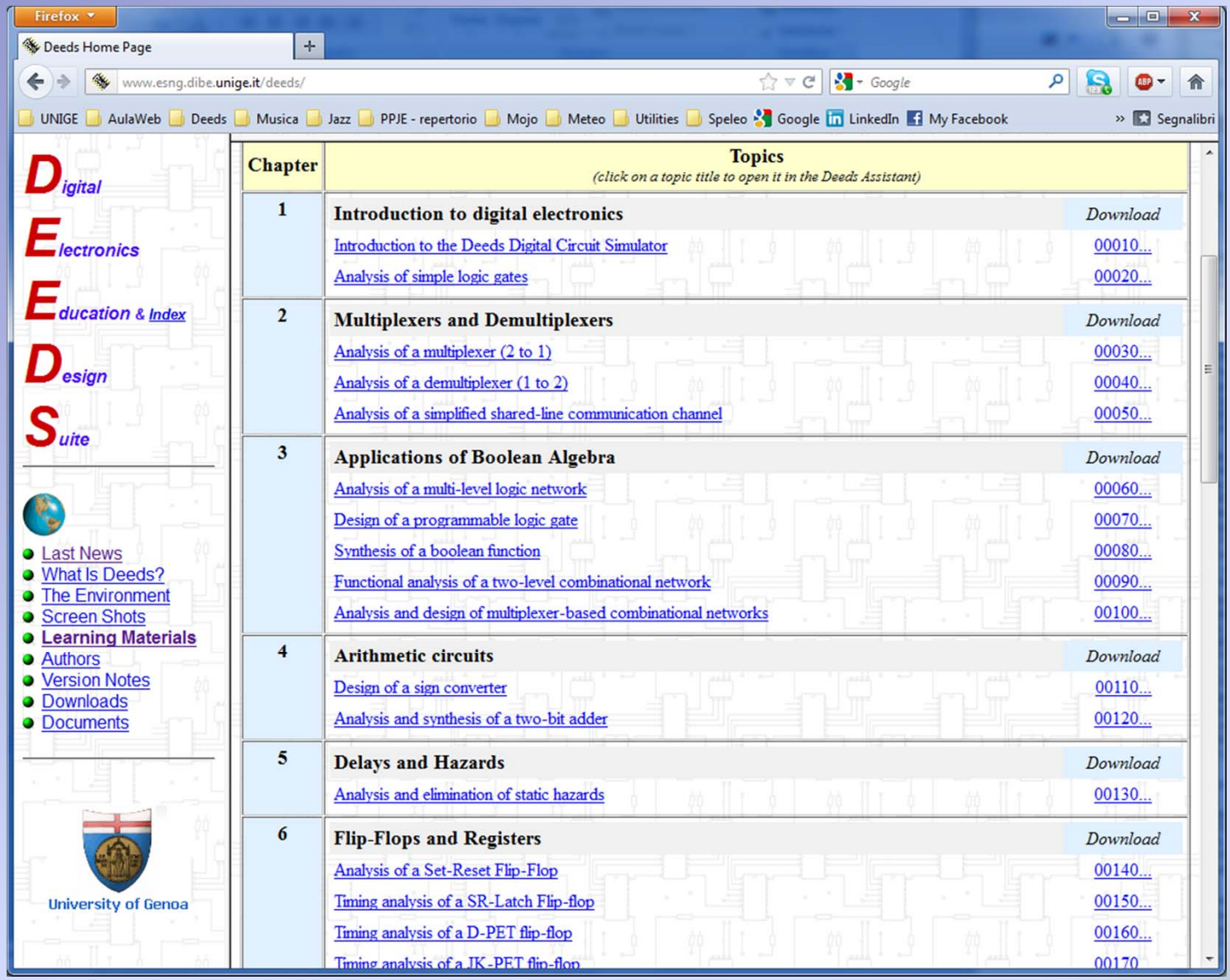
# Deeds: *the simulation tools*

- d-DcS Digital Circuit Simulator
- d-FsM Finite State Machine Simulator
- d-McE Microcomputer Emulator




- The three simulators are *fully integrated*
- It is possible to design and simulate *digital systems* with *standard logic*, *finite state machines* and *microcomputers*
- It allows the understanding of the interaction among the hardware and software components of *embedded systems*
- Deeds Projects can be exported in VHDL

# *Deeds: the learning material*



# Deeds: project assignment

Deeds - Exercises (by topic) - Mozilla Firefox

File Modifica Visualizza Cronologia Segnalibri Strumenti ?

http://www.esng.dibe.unige.it/deeds/Learnin

Google

| | |
|---|---|
| 8 | **Introduction to Finite State Machines** |
| | Re-thinking a synchronous counter as Finite State Mac |
| | Reverse-engineering a synchronous sequential circuit |
| | Design of a synchronous mod-5 up/down counter |
| | Design of a simple serial line receiver |
| 9 | **Design of finite state machines** |
| | Design of a timing sequence generator |
| | Design and synthesis of a 3-bit up-counter for signed r |
| | Design of a serial data processor |
| 10 | **Design of FSM-based digital systems** |
| | Design of a serial line receiver |
| | Design of a serial-programmable pulse generator |
| | Design of a programmable square-wave generator |
| 11 | **Micro-computer systems: introduction to** |
| | Introduction to the Deeds Micro Computer Emulator |
| | Analysis and tracing of a simple assembly program |
| | Analysis and tracing of a simple arithmetic program |
| | Analysis and tracing of a simple de-cryptographic prog |
| | Accessing tables in RAM memory |

Completato

DEEDS Laboratory Session - Mozilla Firefox

File Modifica Visualizza Cronologia Segnalibri Strumenti ?

http://www.esng.dibe.unige.it/deec

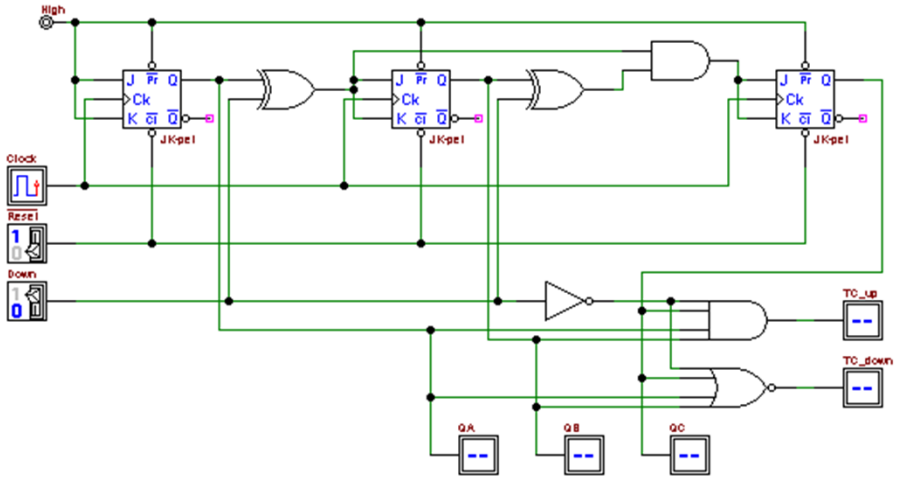
Google

**Introduction to Finite State Machines**

**Re-thinking a synchronous counter as Finite State Machine (FSM)**

This project requires the Deeds simulator installed on your PC. You can download Deeds here.

The following network is a **Synchronous Up/Down Binary Counter**. Click on the figure to open it in the **d-DcS**:

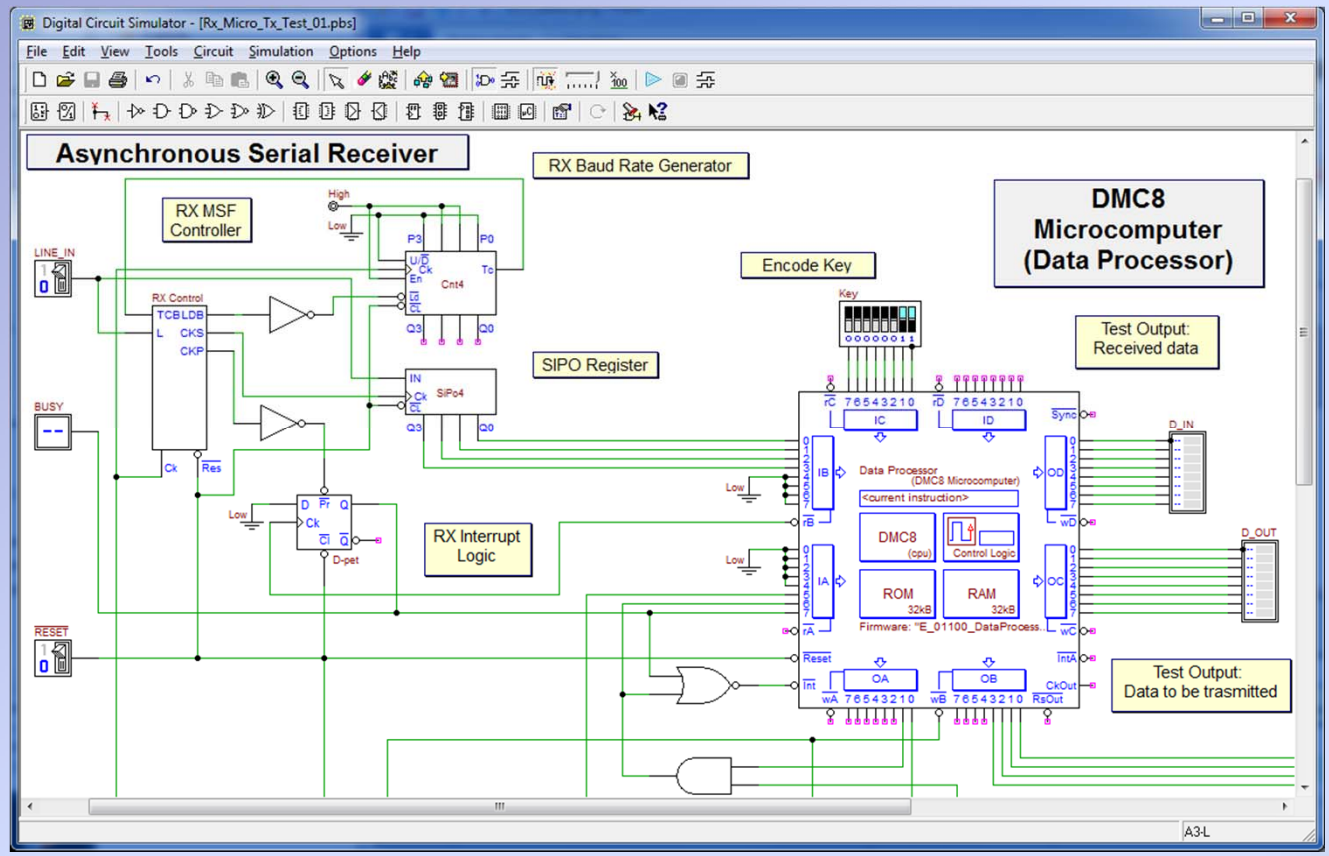



The input **Down** sets the direction of the count (*up* = '0', *down* = '1'). The output **TC_Up** (*Terminal Count, 'Up' direction*) is activated to **'1'** when the *up* count reaches the maximum number; instead, the output **TC_Down** (*Terminal Count, 'Down' direction*) is activated when the *down* count reaches zero. **Verify** such behaviour with the **timing simulator**.

Completato

# *Deeds: the d-DcS Digital Circuit Simulator*

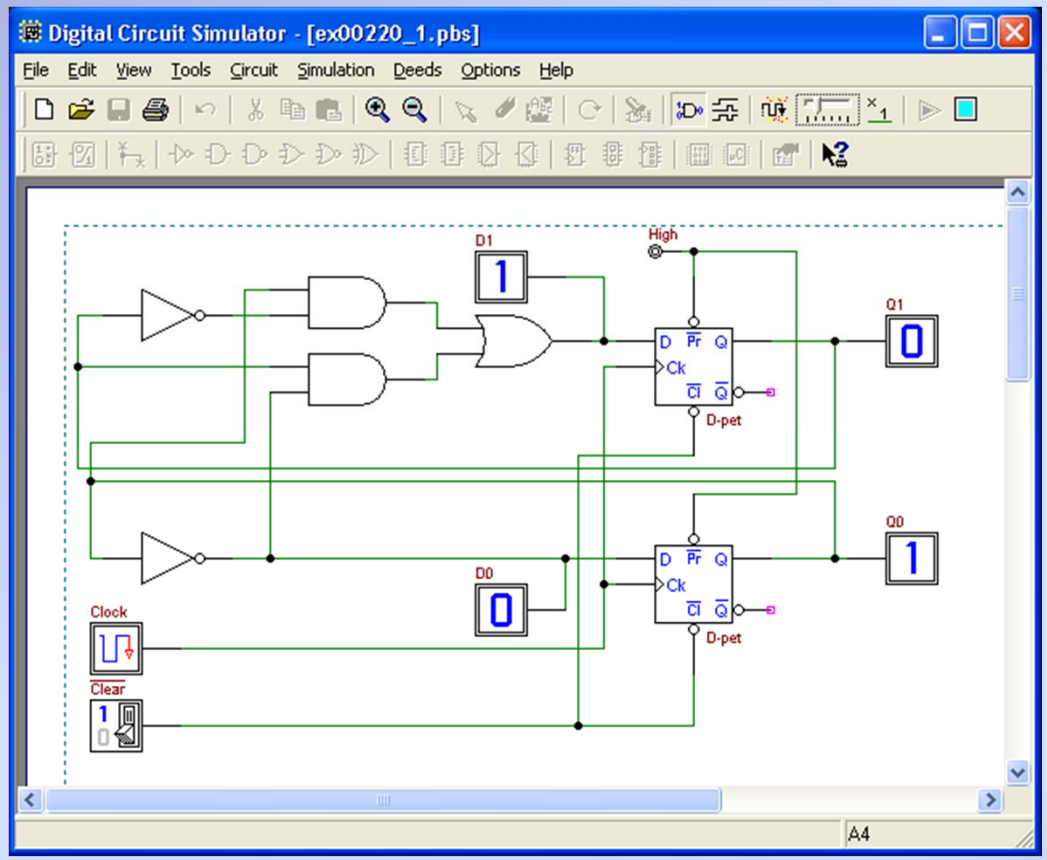
- The d-DcS is a digital simulator specifically developed with educational needs in mind
- The d-DcS has been designed to be easy to use, while maintaining quasi-professional features
- The user interface is intuitive
- The choice of digital components is based on their logical function, not on commercial lines
- Two simulation modes are available:
  a) *Interactive Animation*
  b) *Timing diagram*



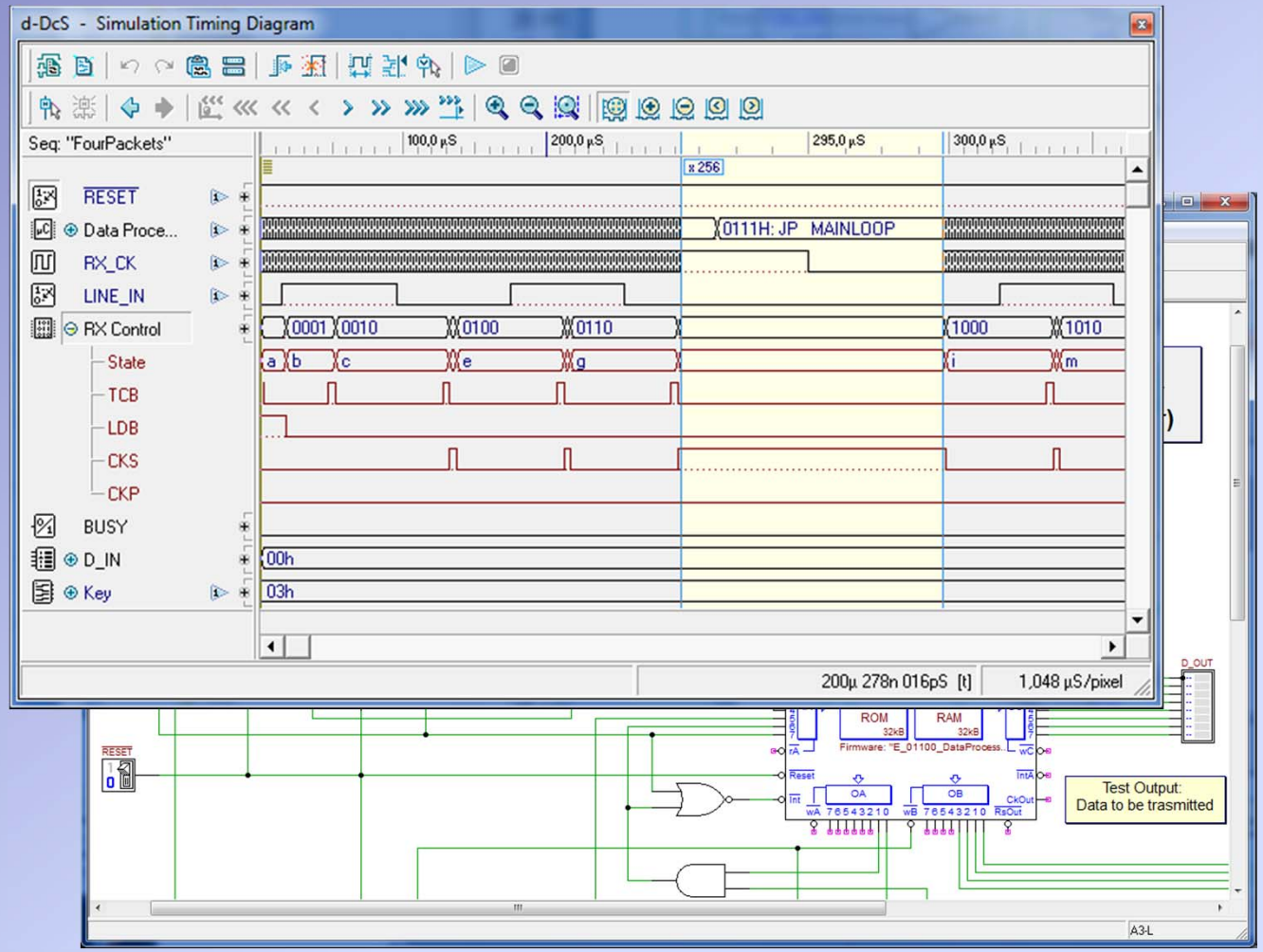

- The d-DcS can export in VHDL projects, including Finite State Machines and the Microcomputer

# *Deeds: the d-DcS Interactive Animation*

- In the interactive animation mode the simulator processes input commands and displays output values
- The input components, on the left of the figure, are a clock and a logical level generator
- The level generator is represented as a switch, toggled by a mouse click
- The clock can be activated edge by edge or continously
- Output components display the values of the selected nodes (four in the figure)



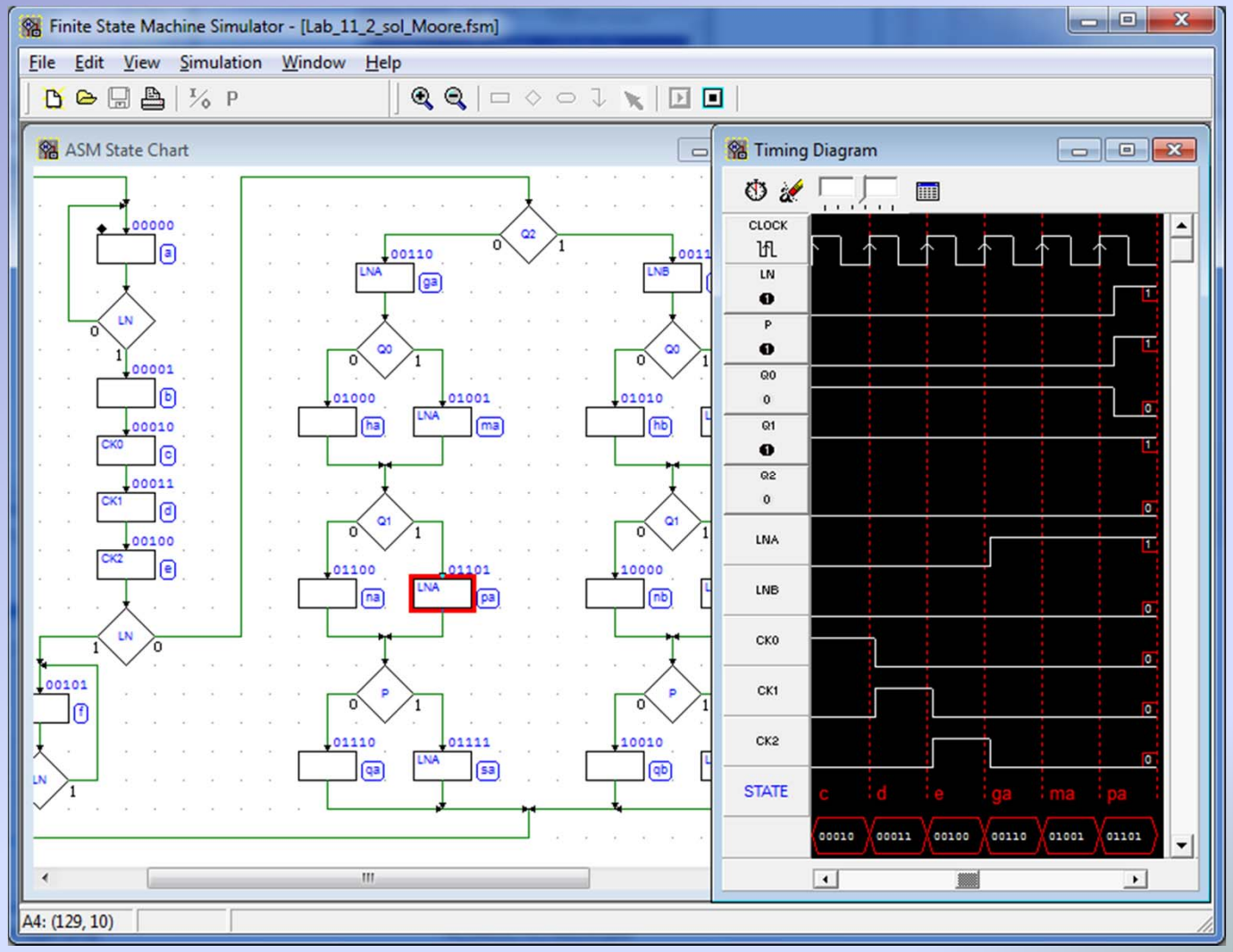
# *Deeds: the d-DcS Timing Diagram*




- Timing diagram simulation is the most common tool to check circuits' functionality, as in professional tools
- Clock and input signals can be easily edited
- Input sequences can be saved with the circuit file: this feature is useful for educational applications
- Analysis utilities complete the timing diagram, i.e. the horizontal magnifier (visible here), used to enlarge details while observing overall simulation results.

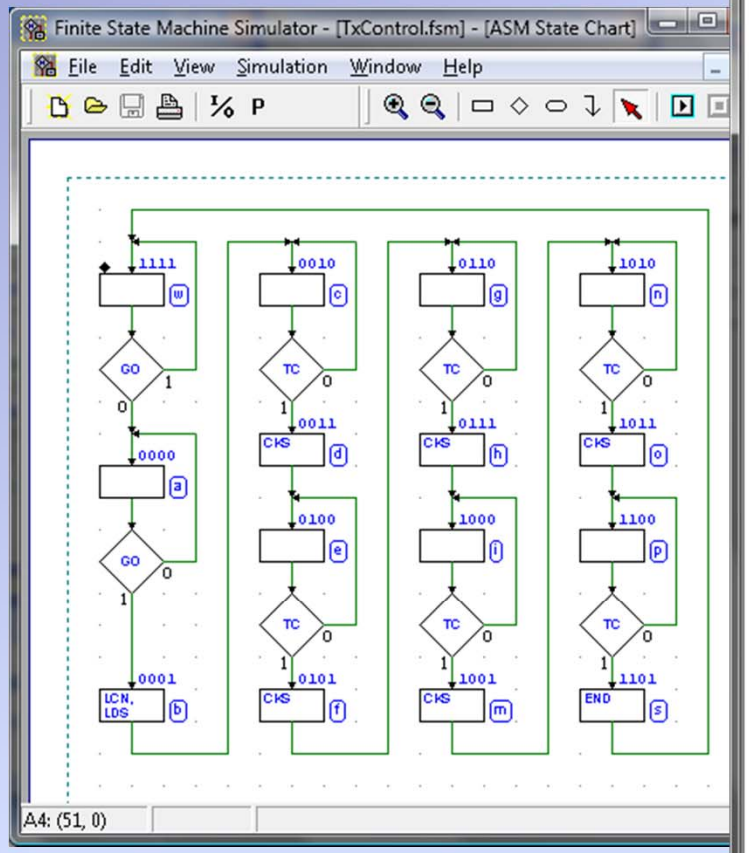
# *Deeds: the d-FsM Simulator*

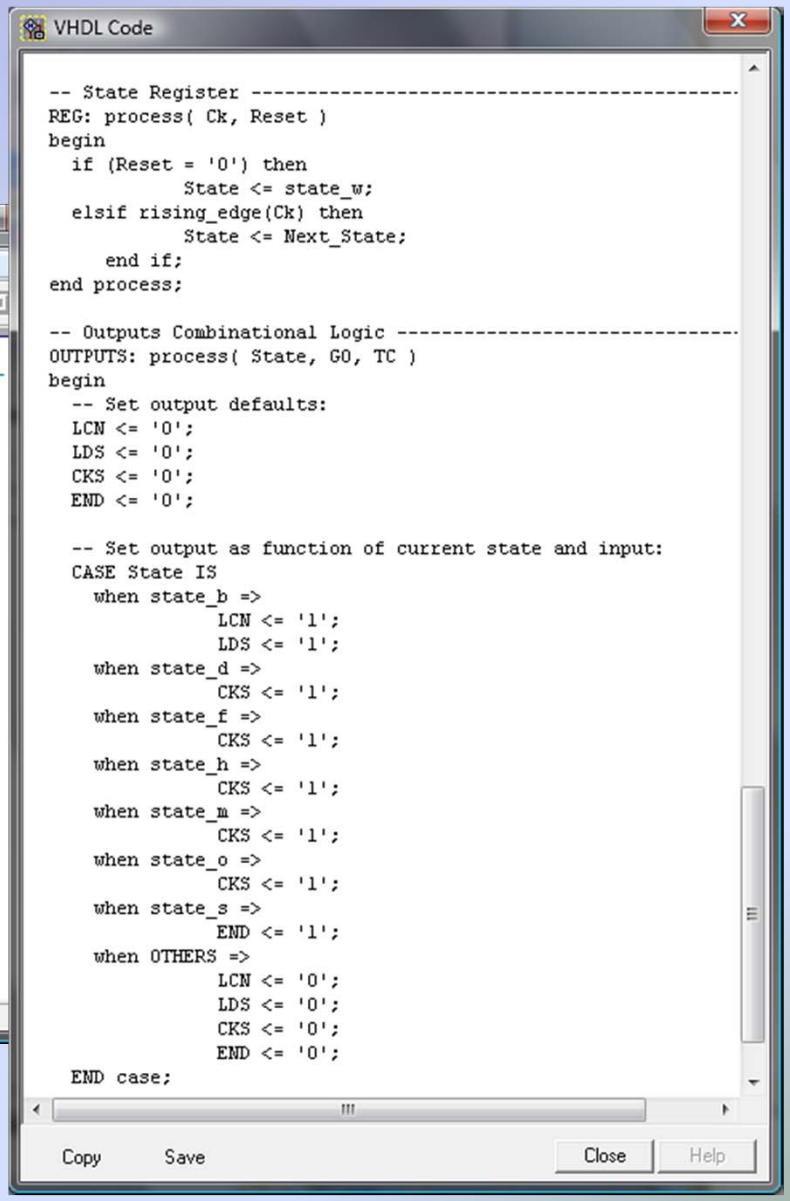
- Finite state machines (FSM) are designed with ASM charts
- Algorithms can be functionally tested (with a timing diagram), without circuit syntesis
- A FSM produced with the d-FsM tool can be used as a component by the d-DcS
- A FSM can also be exported in behavioral VHDL language, to reuse it in professional design tools



# *Deeds: the d-FsM to VHDL encoder*

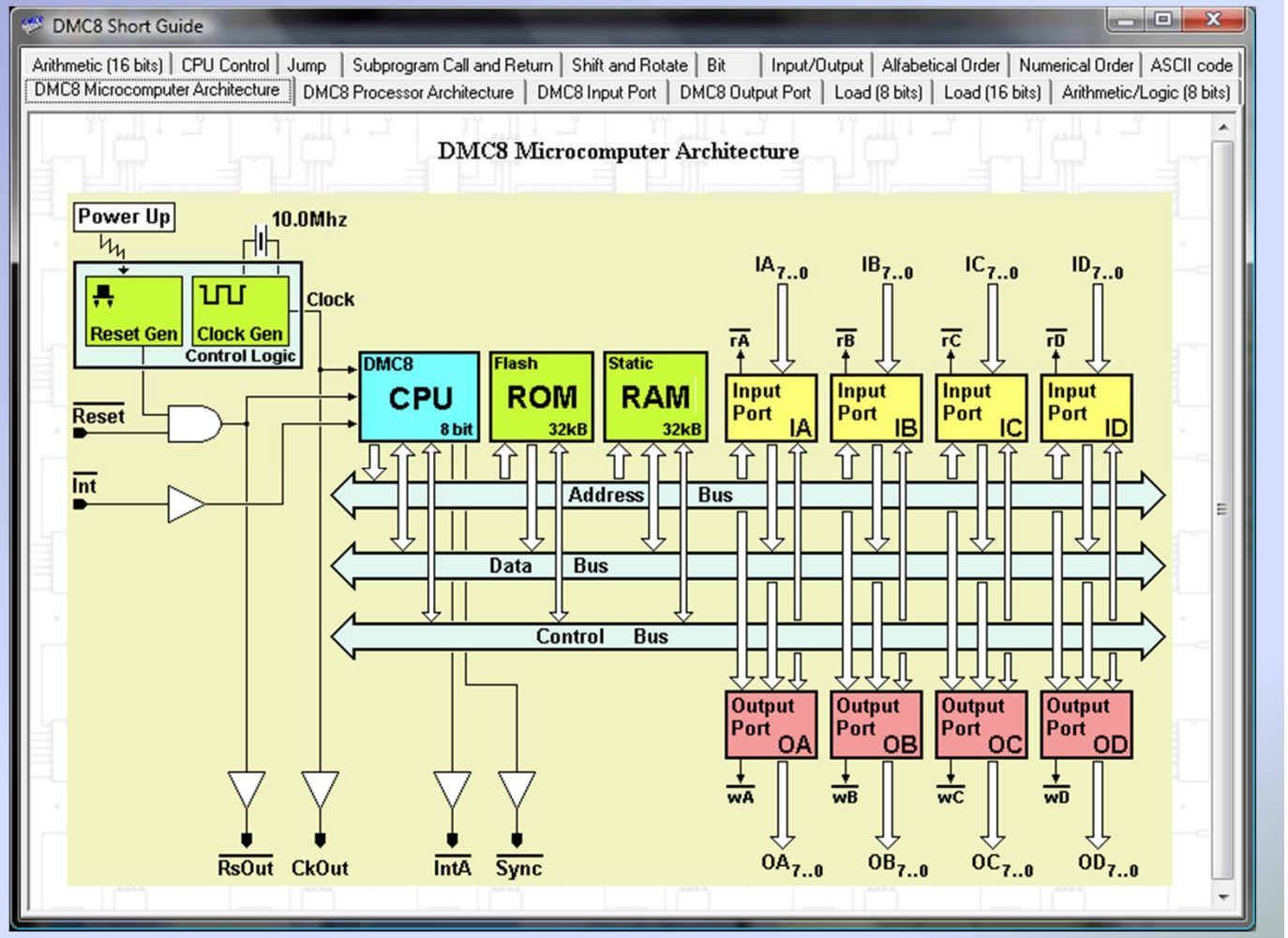
- A FSM can also be exported in VHDL language, to allow reusing it in professional design tools



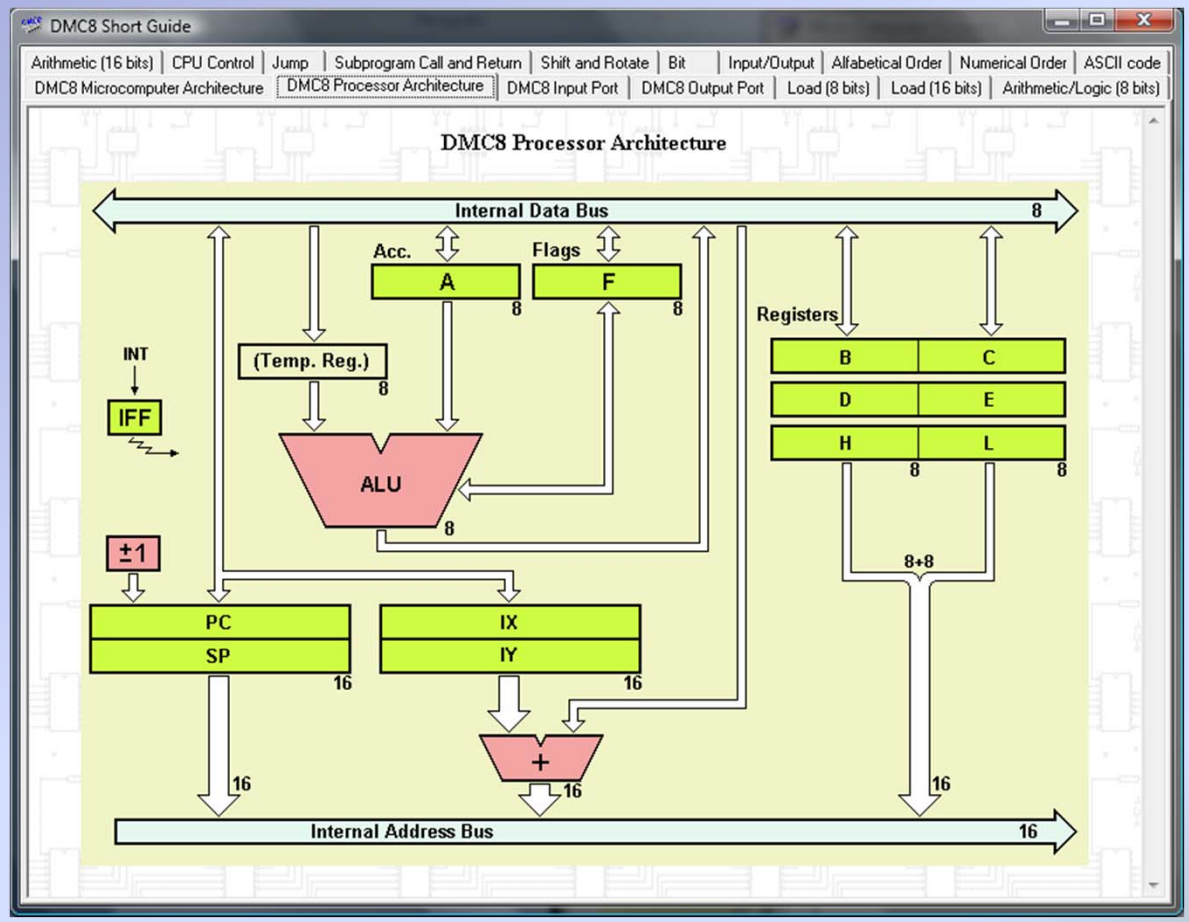

# *Deeds: the d-McE μC emulator*

- The ***d-McE*** micro-computer emulator is based on an 8-bits microprocessor, RAM and ROM and a simple parallel input/output port system
- It interfaces the external world through four input and four output parallel ports
- Address and data busses are not available outside, but port control signals are available to extend the ports
- Clock, Reset and Interrupt signal are available



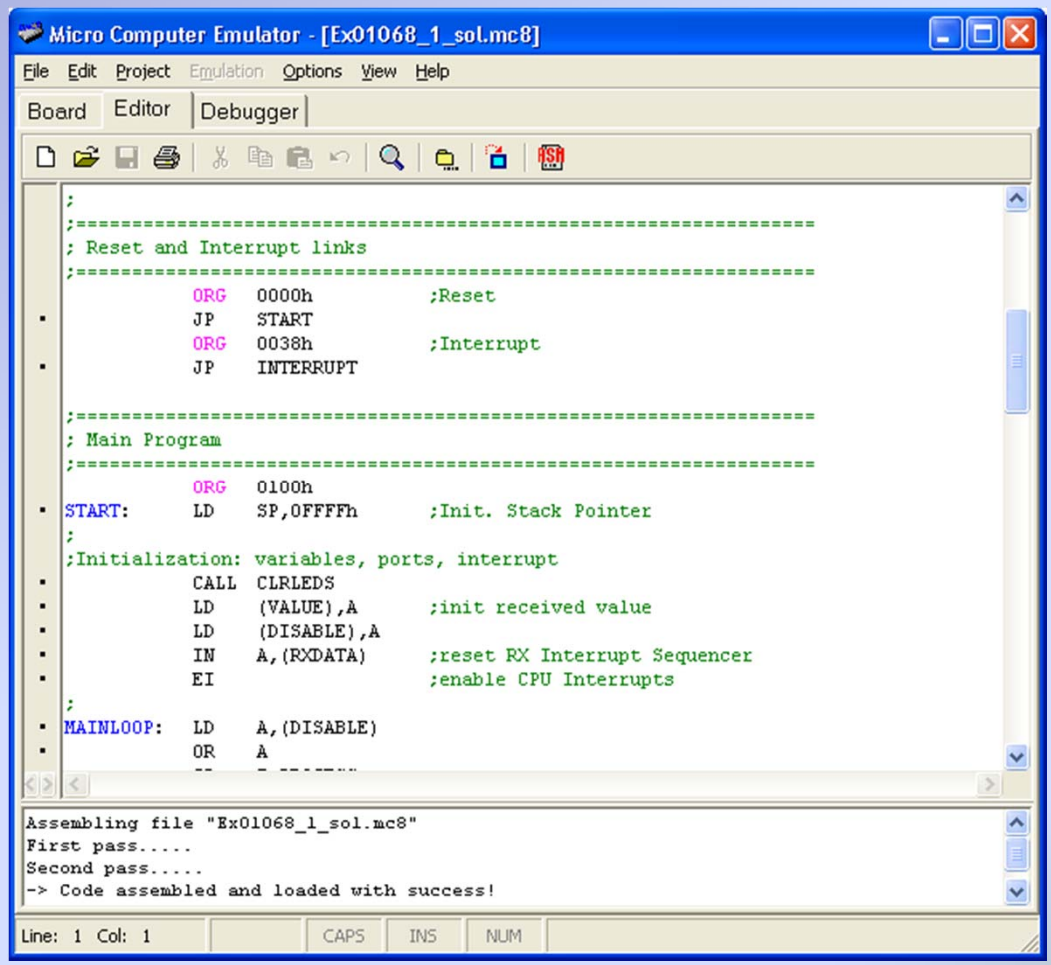
# *Deeds: the CPU architecture*

- The 8-bit micro-processor is the **DMC8**
- **DMC8** emulates a simplified version of the well-known Z80-CPU
- The use of a "state of the art" CPU is not compatible with our pedagogical targets



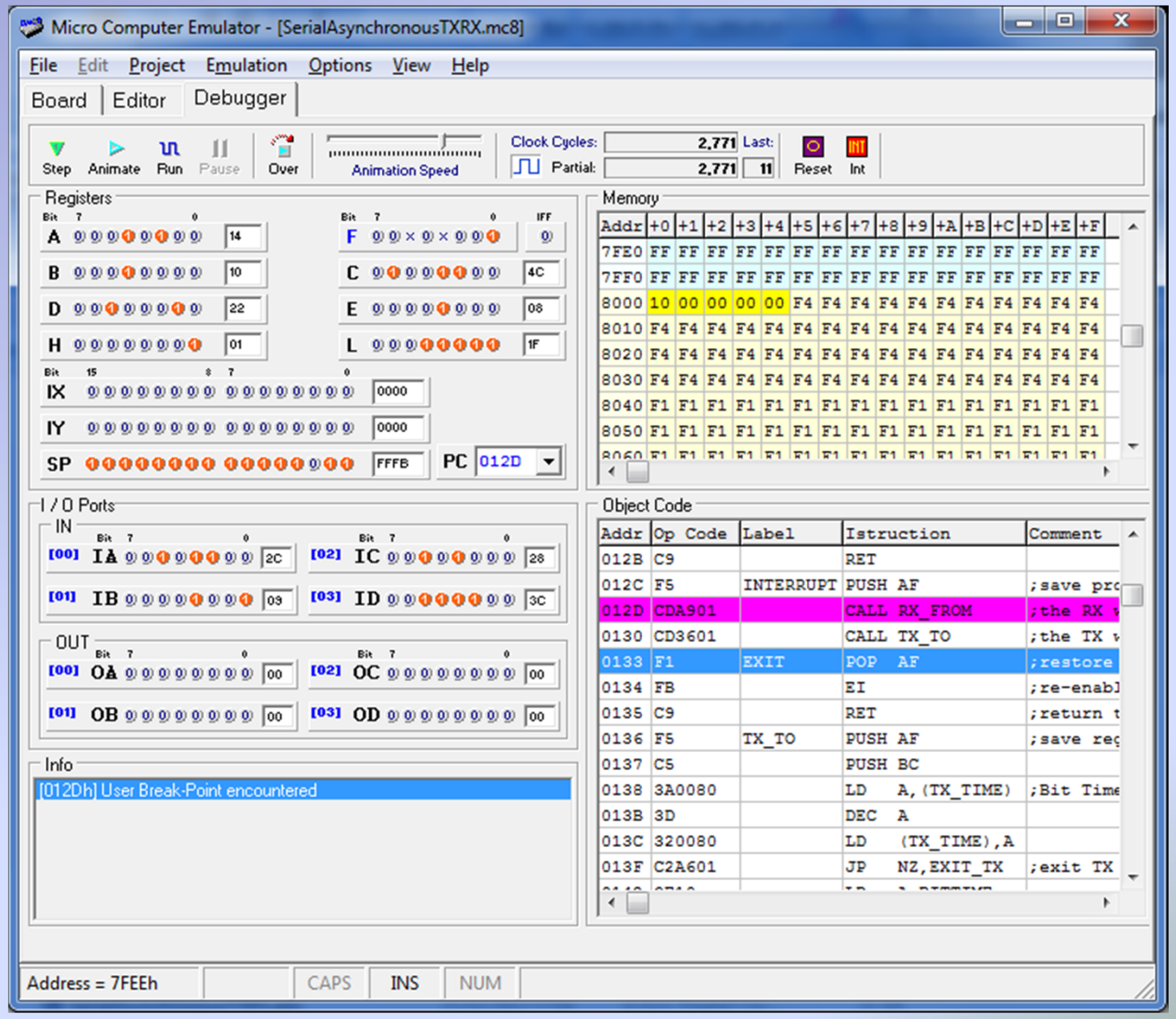
# *Deeds: the d-McE code editor*

- The micro-computer emulator allows to edit assembly code with syntax highlighting
- Assembling, linking and loading operation are transparent to the user



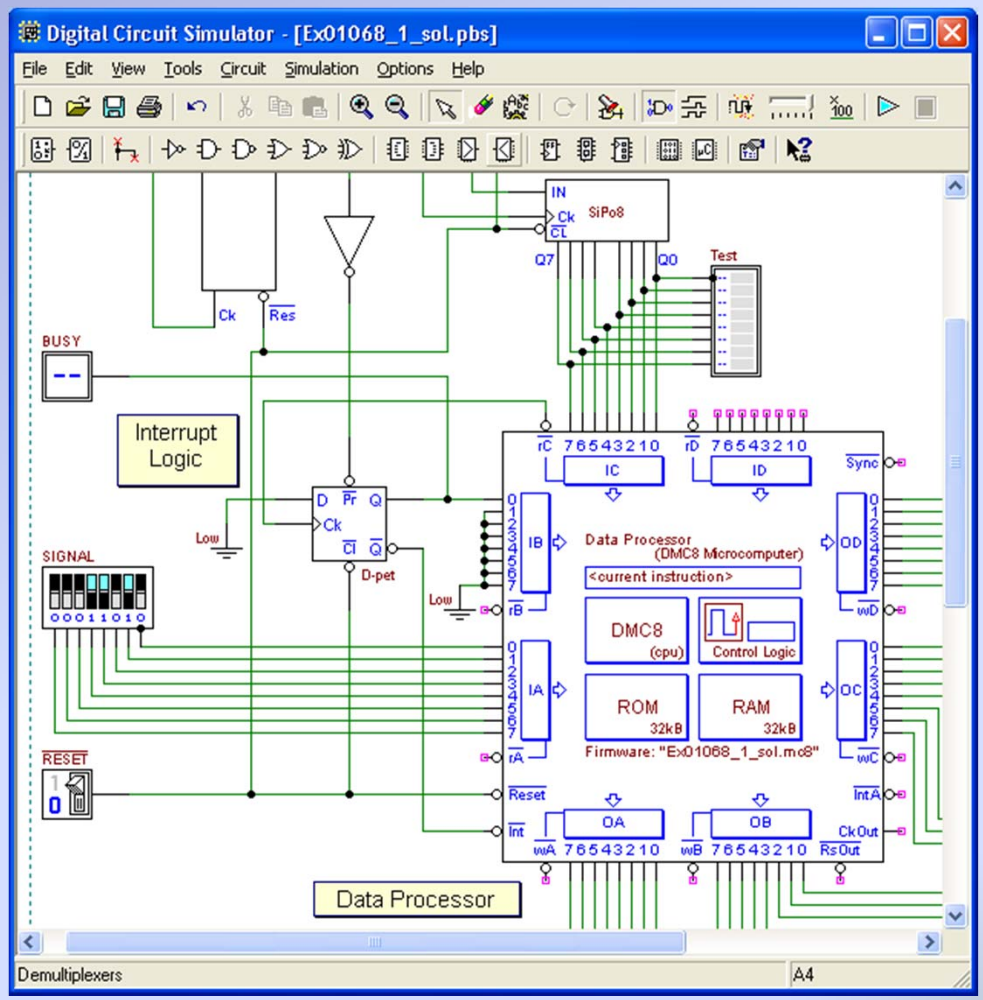
# Deeds: the d-McE debugger

- The micro-computer emulator allows to debug the program step-by-step or in animation mode
- The interactive visual debugger shows memory, registers and ports contents
- The tool allows a full control of the microcomputer, including I/O operations
- The emulation system has been designed with the student in mind
- Each instruction is executed against a preventive control of eligibility of all the contour parameters (valid program counter value, valid memory address, valid memory contents, valid port address, attempt to write outside of RAM space, and so on)
- User can setup one or more breakpoints, useful to halt the execution in order to verify intermediate results



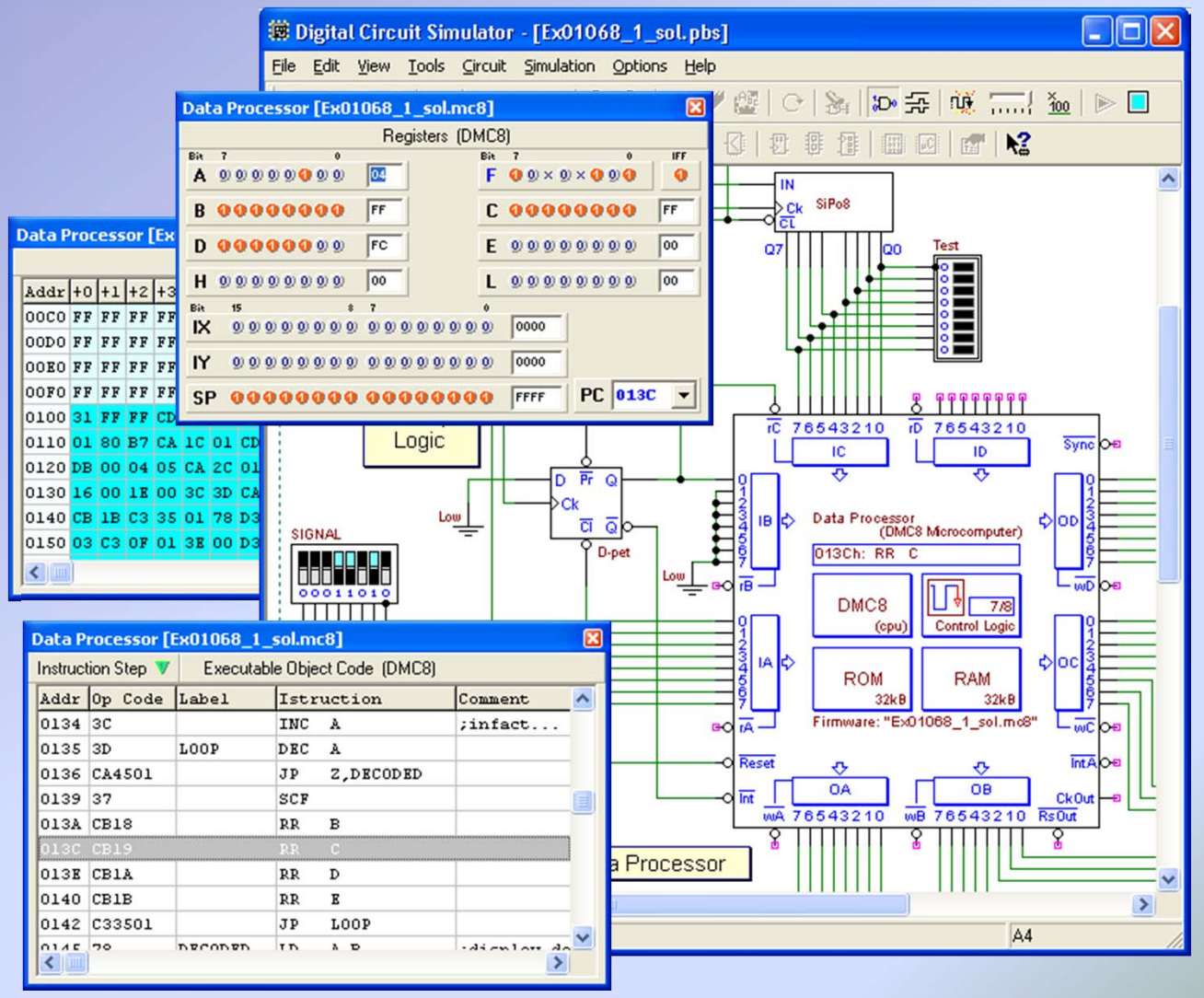
# *The micro-computer as d-DcS component*

- The DMC8 micro-computer is a component of the library of the d-DcS
- A digital circuit, embedding one or more DMC8 micro-computers, can be simulated by the d-DcS
- The embedded DMC8 micro-computer can be programmed with the d-McE tool
- The FPGA extension allows the export of the entire circuit, including the microcomputer and the user program




# *The micro-computer in the d-DcS*

- The functionality of the d-McE debugger is available in the d-DcS
- Memory, CPU registers, port status can be monitored during simulation of a digital system that embeds one or more micro-computer
- Through the debugger windows we can analyse the program logic while testing the hardware behaviour

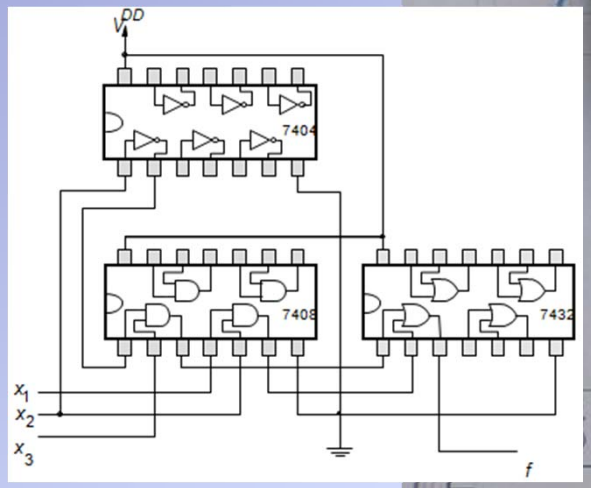


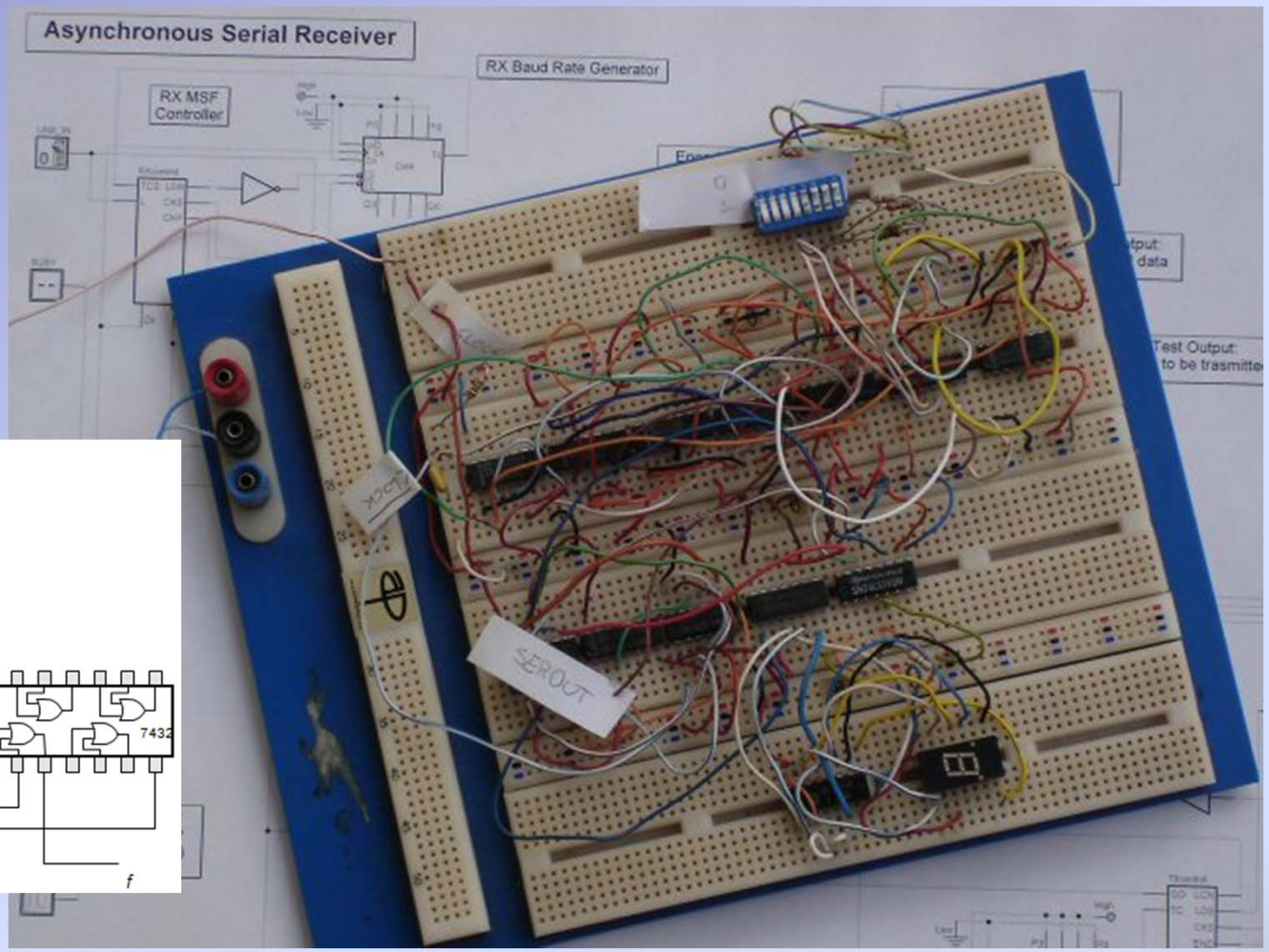
# *Deeds in practice at University of Genoa*



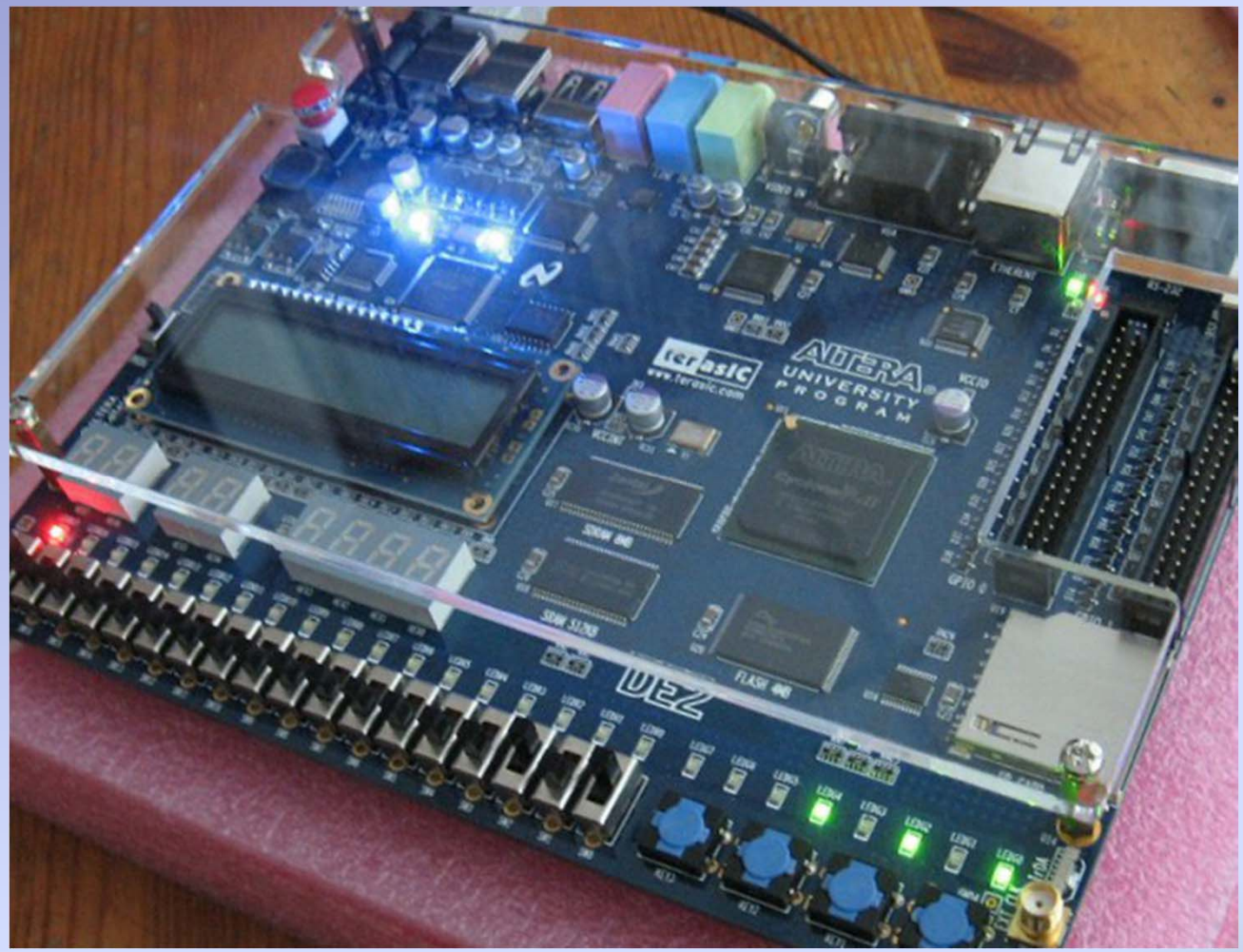

- ***Deeds***, as support to traditional teaching, has been extensively used in our institution by thousands of students of the first and second year of ***all the information engineering curricula***.
- This practice has been very successful, as demonstrated over the years by the evaluation procedures.
- Several colleagues from European universities have adopted Deeds in their teaching.
- Deeds is available free of charge to all interested parties

# *Deeds replaced a hand-wired breadboard with simulation*



# *But now breadboarding is here again with the Deeds FPGA extension!*



# *The "FPGA extension" (1)*

- The ***Deeds "FPGA extension"*** combines the pedagogical value and existing material of Deeds with the innovation represented by the FPGA
- It allows students to compile a project into an FPGA chip starting from Deeds, ***leaving in the background the operations performed by the FPGA-specific development software***.
- Our purpose is not to simplify FPGA programming via high-level synthesis languages, but to allow a direct implementation on FPGA of Deeds projects

# *The "FPGA extension" (2)*

- The project to be implemented on FPGA may be composed of:
  - Combinational and sequential components from Deeds library;
  - Finite State Machines;
  - DMC8 microcomputers.

# *The FPGA extension module: technical description*

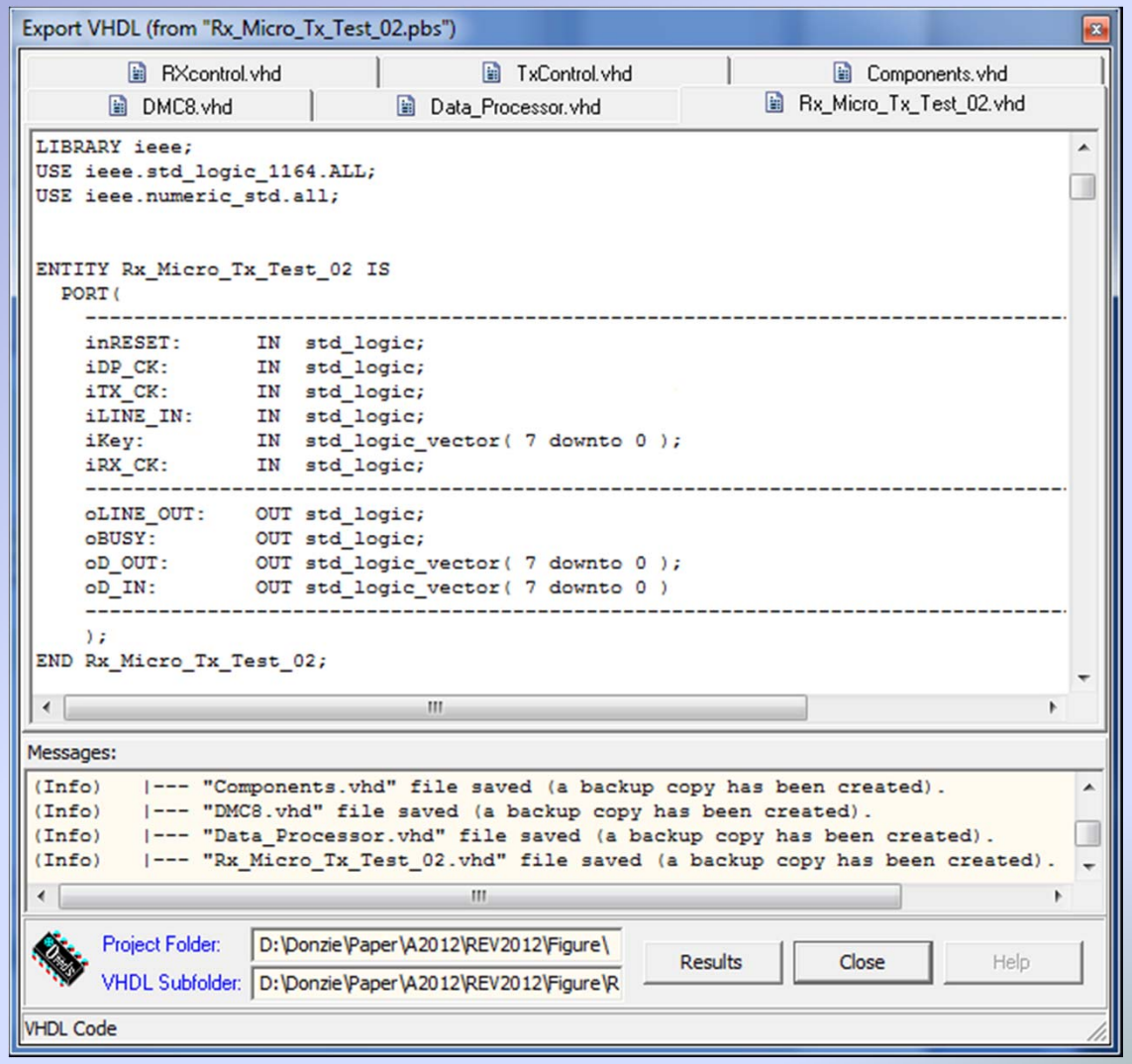
- The ***FPGA extension*** module has been integrated with the ***d-DcS*** tool, where two new commands are available:
  - The **"Export in VHDL"** command converts into VHDL conversion the digital system currently opened in the Deeds schematic editor
  - The **"Test on FPGA"** configures VHDL conversion for a particular FPGA development board, generating all the specific "Project Files"

# *"Export in VHDL" command (1)*

- All library components, FSM and Microcomputers are exported in behavioral VHDL code.
- The top level schematic is compiled in structural VHDL.
- The VHDL code produced is quite general, not specific for a particular FPGA implementation.
- At the end of the conversion, the VHDL files are presented to the user through a dedicated dialog window

# *"Export in VHDL" command (2)*

- The ***"Export VHDL" dialog window*** visualizes all the files produced
- It shows a short report on the conversion process
- In this example, part of the top level VHDL "Entity" defining all the input and outputs of the schematic
- Users with a good VHDL familiarity can ***export, edit and re-use the generated code*** in a professional design tool of their choice



# *"Test on FPGA" command (1)*

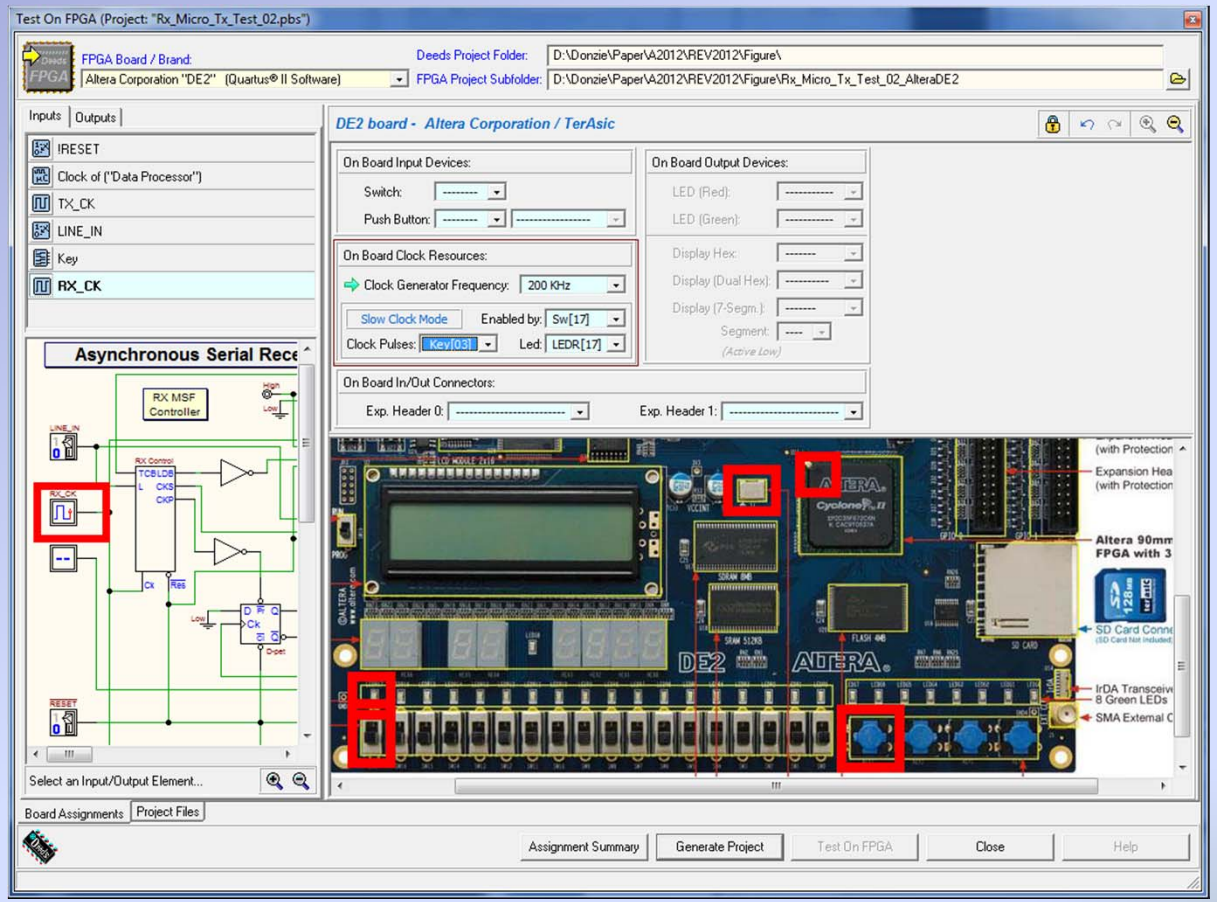
- The ***"Test on FPGA"*** command specializes the VHDL conversion for a particular FPGA development board.
- Selecting a particular FPGA board, the user selects also the specific FPGA professional tool to be used.
- The VHDL code conversion is executed with the same criteria described for the ***"Export in VHDL"*** command, but the top level VHDL code is generated according to the particular FPGA board features and connections.
- The conversion process generates not only the VHDL files from the current project, but also all the "Project Files" needed by the FPGA professional tool.

# *"Test on FPGA" command (2)*

- At the end of the conversion, all the ***Projects and VHDL files*** are ready to be opened directly in the specific FPGA professional tool
- By default, the students interaction with the FPGA tool can be reduced to a minimum: beginners students need only to compile the project and then download it to the FPGA board
- Advanced students, instead, can start from here to experiment the possibilities offered by the FPGA tool, and practicing with changes in VHDL code, adding functionalities, etc.
- All the ***"Test on FPGA"*** command operations are defined by the students using the ***"Test On FPGA Expert Window"***

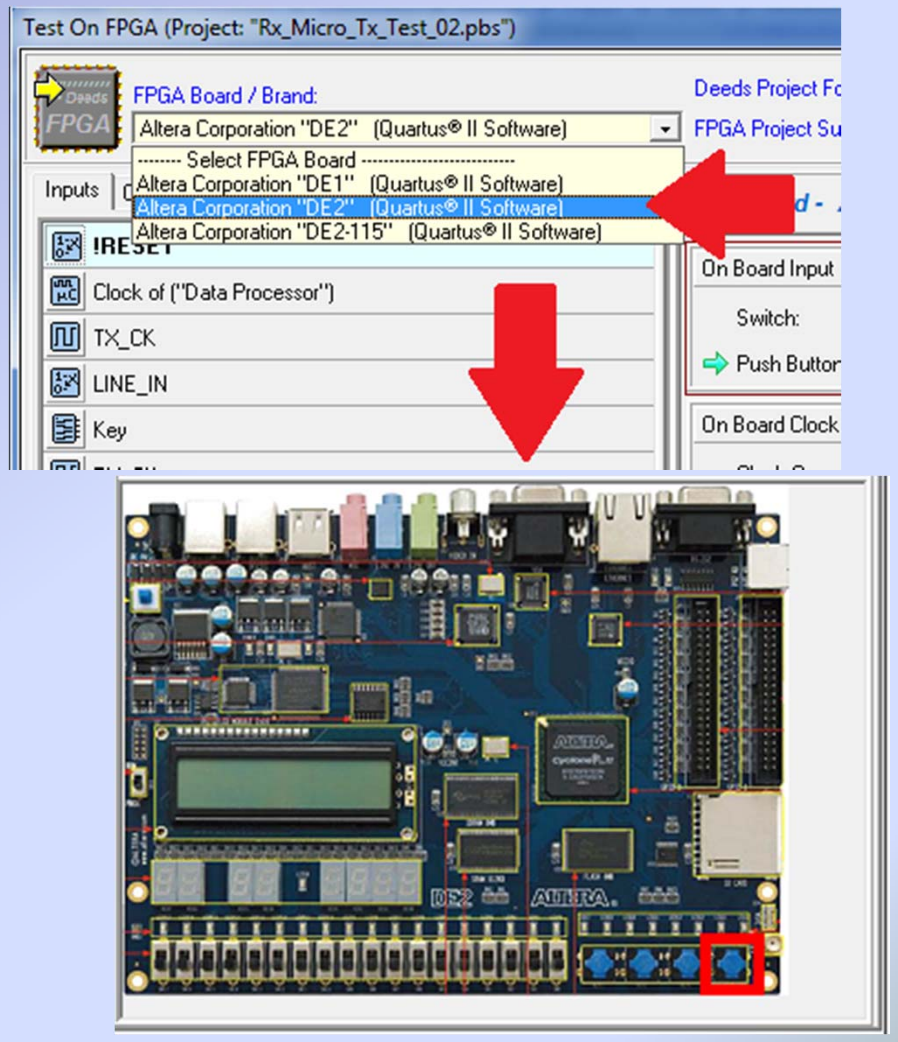
# *"Test on FPGA" Expert Window*

- The user interact with all the ***"Test on FPGA"*** processes by means of this multi-page window
- The first page contains the ***"Expert"*** that allows to define all the needed parameters
- Here the user can:
  - Target the conversion to a particular FPGA board
  - Make all the associations between the input and output of the schematic and the FPGA board devices and resources.

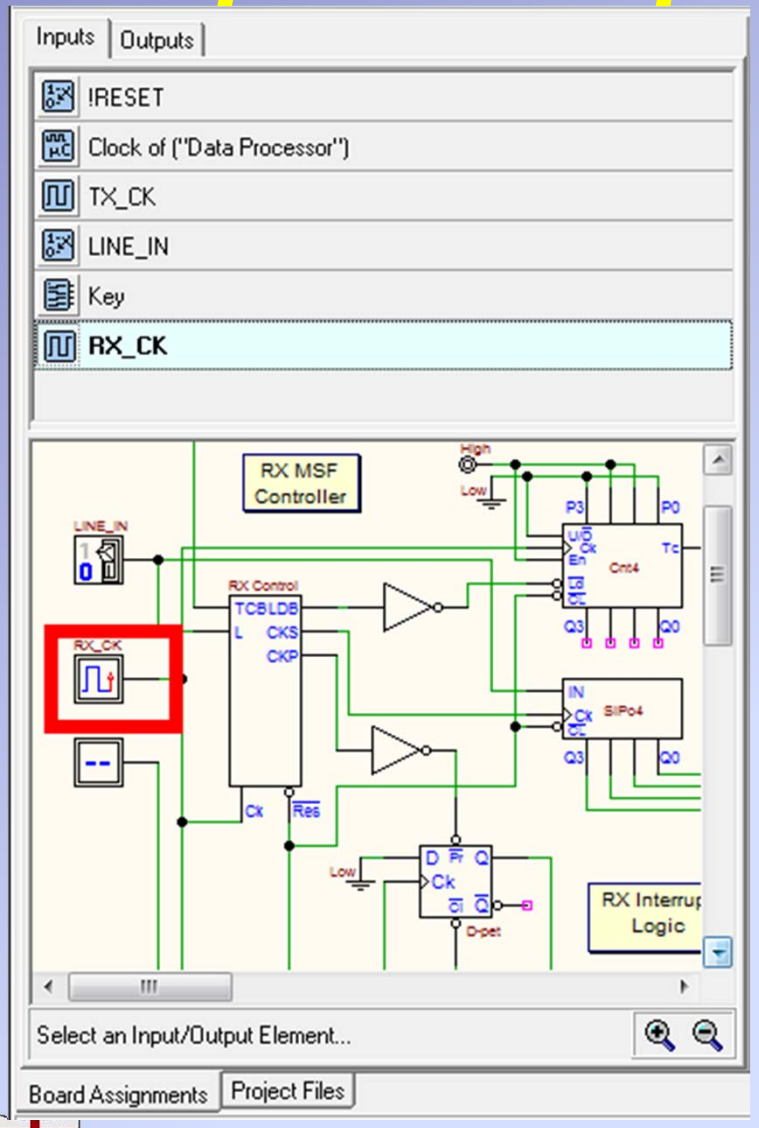


# The FPGA board selection

- The user selects one of the available FPGA boards: the choice is associated to the specific FPGA professional tool
- The board photo is displayed in the same window, to allow device highlighting of user associations
- In the present release of ***Deeds*** (v.1.60), we started with the support of the following board models: **DE1**, **DE2** and **DE2-115** by Terasic, based on Altera Corporation chips
- The ***FPGA extension*** has been already designed to extend support, from the next major release of ***Deeds***, also to Xilinx boards and devices



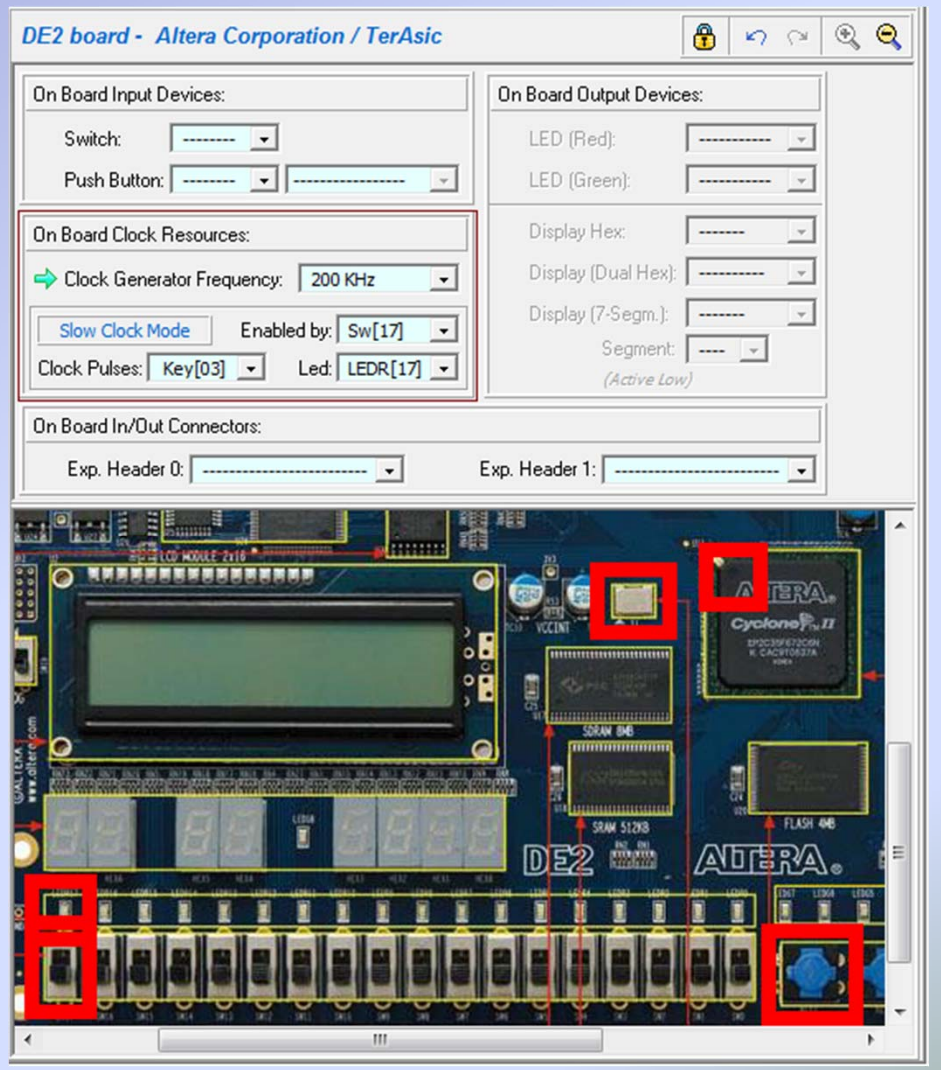
# Input / Output Signal Selection




- The user selects each ***Input*** or ***Output*** termination included in the ***Deeds schematic*** with the aim of to associate it to a ***on-board device or resource***
- The selection can be done in two ways: selecting the signal in the ***Input or Output lists***, or with ***a click on the schematic***
- When an ***Input*** or ***Output*** component of the ***Deeds schematic*** has been selected, on the opposed part of the window the compatible board devices became available and selectable (see next slide)

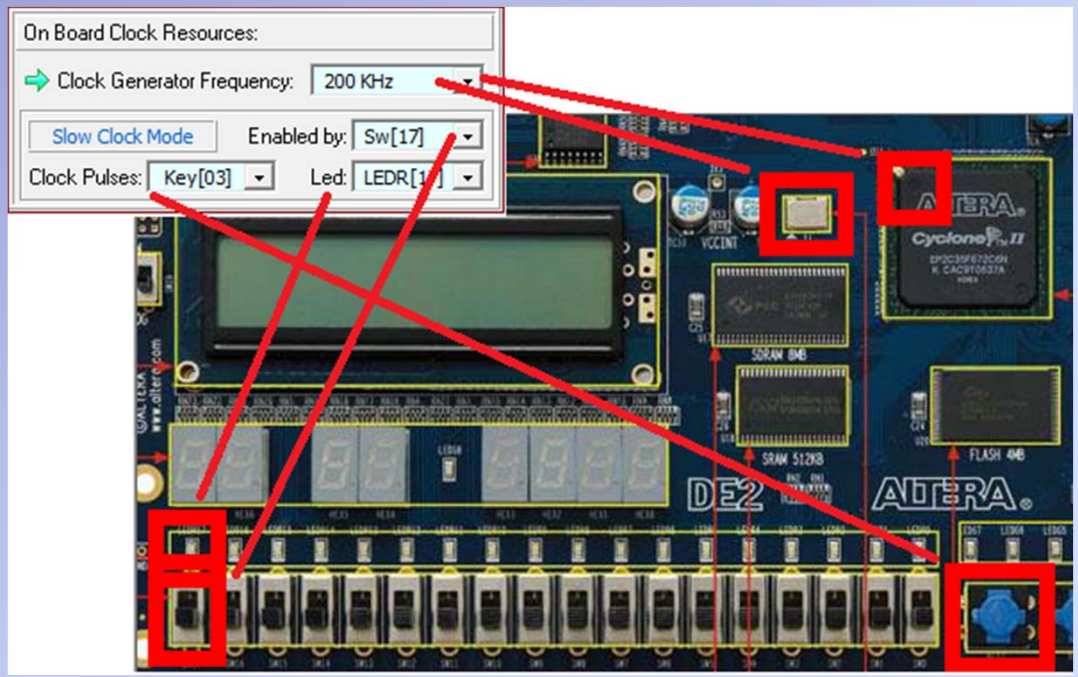
# On-board resources association

- For each ***Input*** or ***Output*** component of the ***Deeds schematic***, the compatible board devices became selectable
- In this example, a ***Clock Resource*** has been selected, on the board, assigning to it a frequency of 200KHz
- The ***graphical highlighting*** of the objects visible in the board image will help in this process
- Typical available devices are: ***switches***, ***push buttons***, ***LEDs***, ***hex displays***, and on-board ***connectors***
- ***Undo/Redo*** commands are provided



# *Clock and Slow Clock definition*

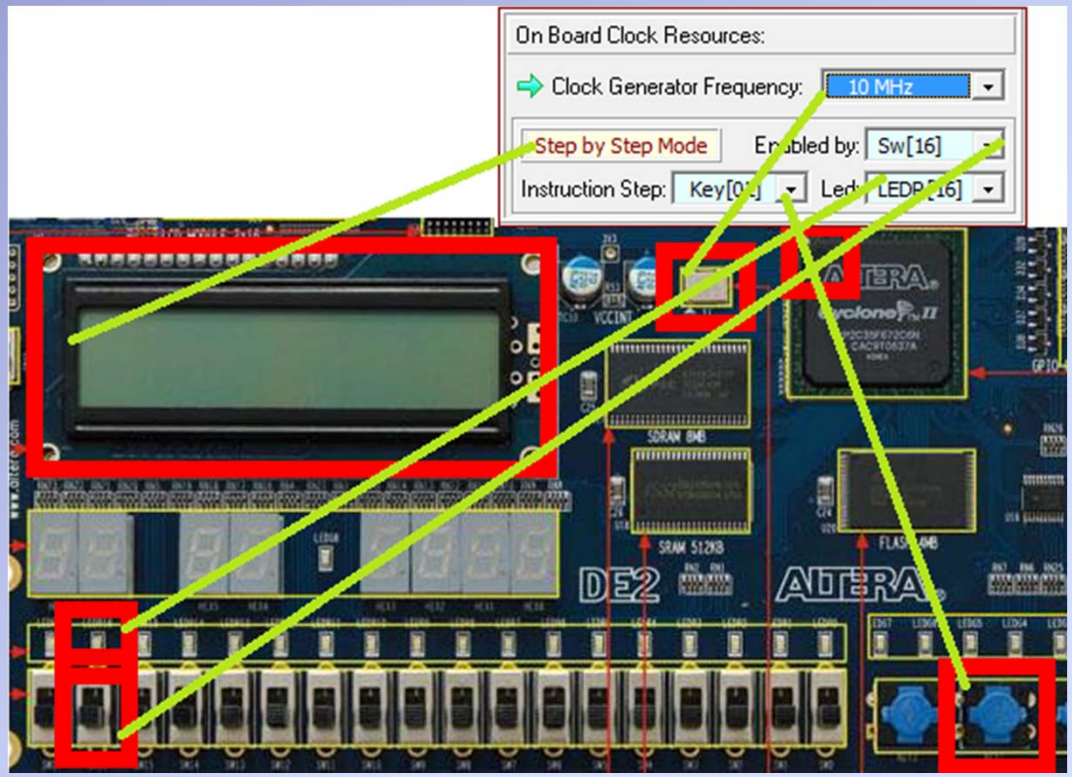
- The VHDL converter provides automatically for ***additional components to scale down clock frequencies***, starting from the on-board native ***clock oscillators***
- For a clock resource, we can optionally set the ***Slow Clock Mode***. When set, this option allows to slow down the clock, at run time, simply by turning on a switch




- In this example, an on-board clock resource of ***200 KHz*** has been set
- Also the ***Slow Clock Mode*** has been set, defining:
  - ***Sw[17]*** as “enabling switch”
  - ***LedR[17]*** as clock Led
  - ***Key[03]*** button as clock Pulser.
- If enabled by ***Sw[17]***, pushing the button ***Key[03]*** will generate manually one clock pulse

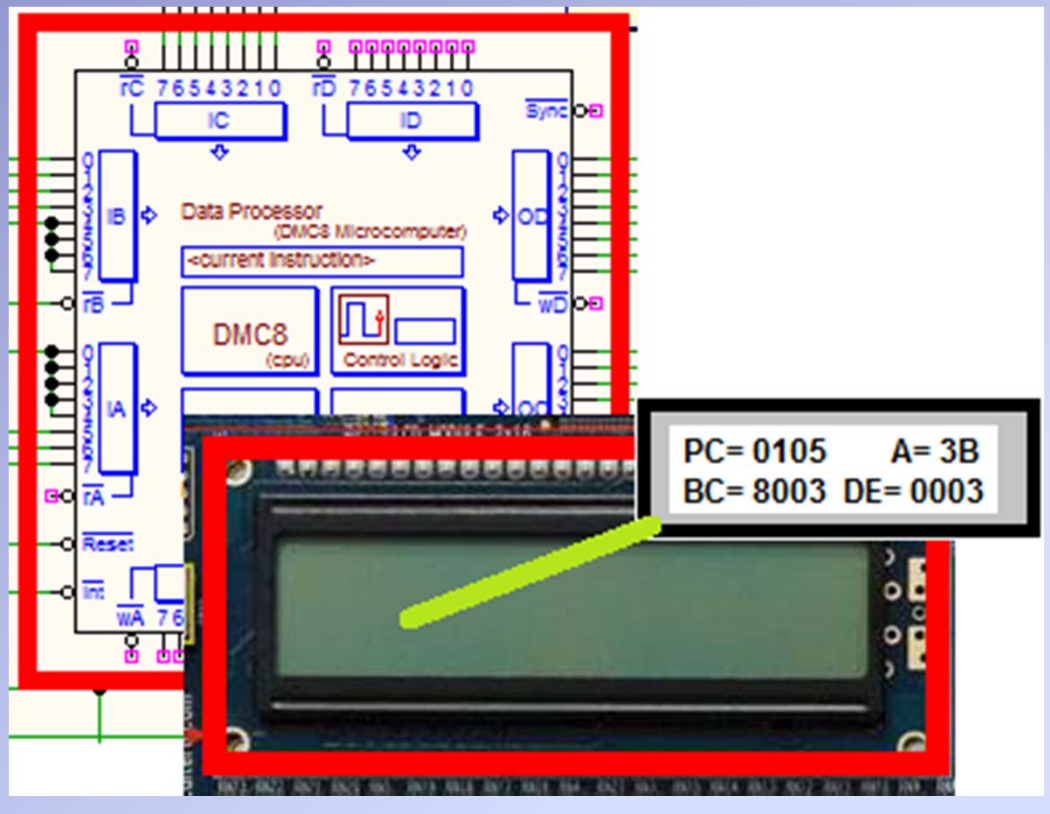
# *Micro Computer and Step by Step Mode*

- The VHDL converter provides automatically for ***additional components*** to ***debug program execution***, if the ***Step by Step Mode*** has been preset
- The *Step by Step Mode* is activated, at run time, simply by turning on a switch



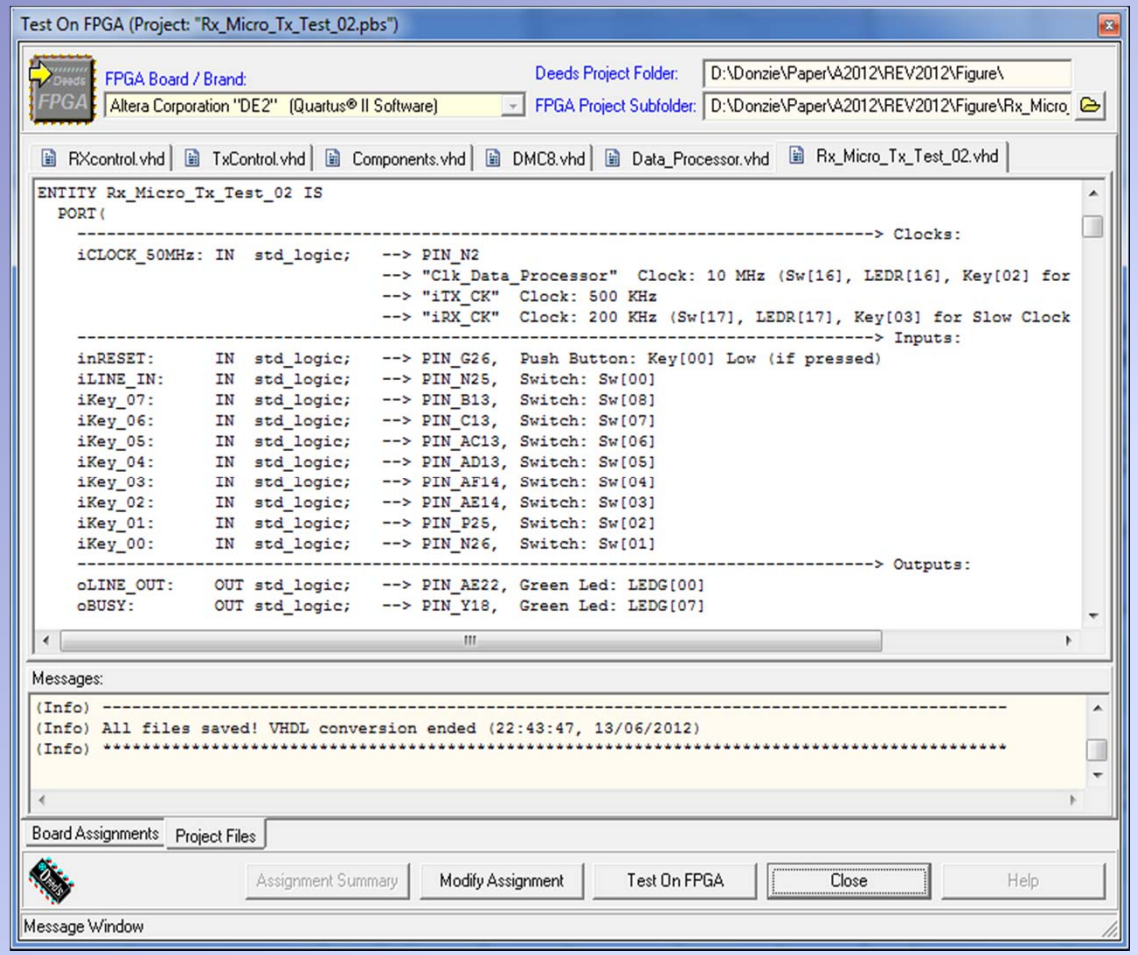

- In this example, a clock of ***10 MHz*** has been set for the microcomputer
- The ***Step by Step Mode*** has been set, defining:
  - ***Sw[16]*** as “enabling switch”
  - ***LedR[16]*** as clock Led
  - ***Key[02]*** button as instruction Stepper
- If enabled by ***Sw[16]***, pressing the push button ***Key[02]***, it will execute the ***next instruction***

# *Micro Computer: the Debugger*



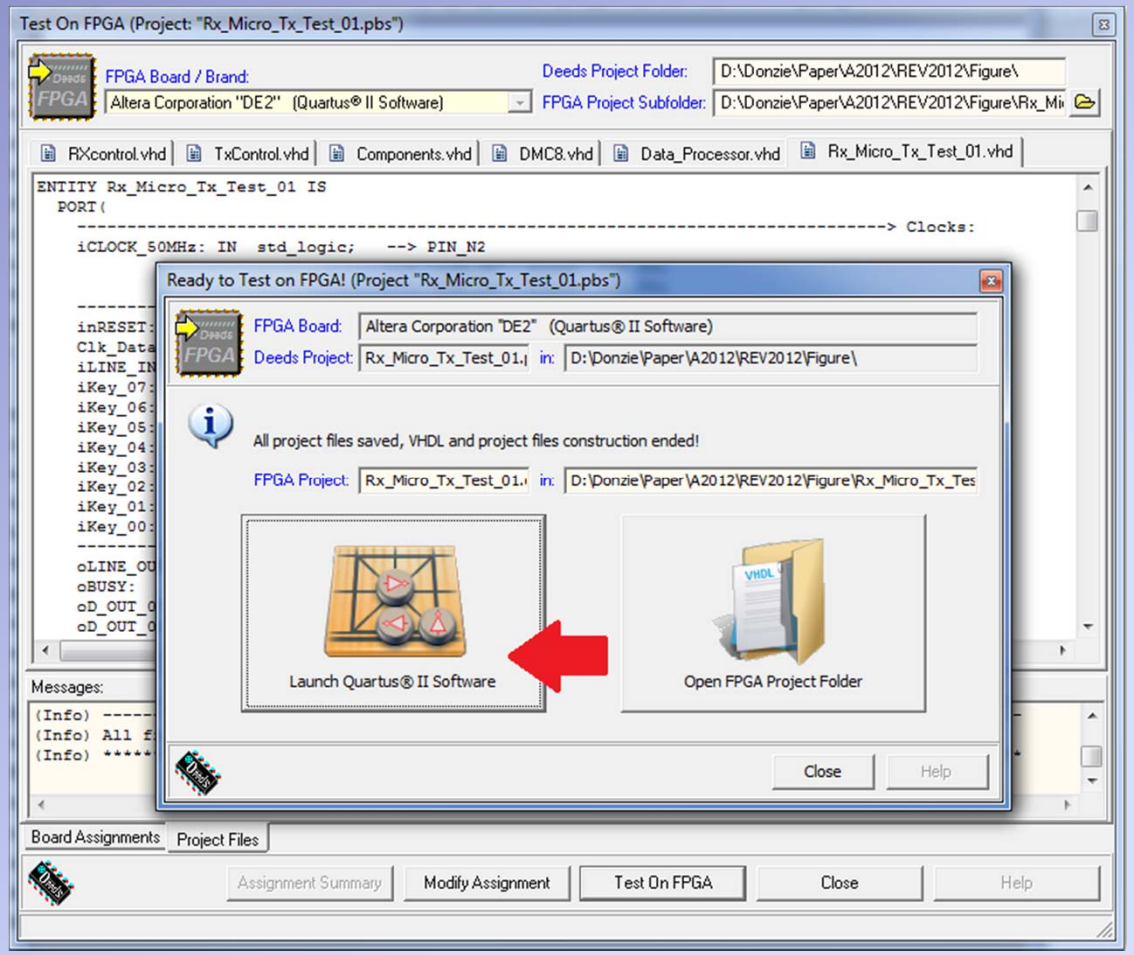

- If an ***alphanumerical display*** is available on the FPGA board, it can be used for ***microcomputer debugging***
- When the ***Step by Step Mode*** is activated, the debugger will update the ***processor status*** on the display, after the execution of each instruction
- The display has only two rows: pages are continuously rotated to show all registers contents.

# VHDL and Project Files generation



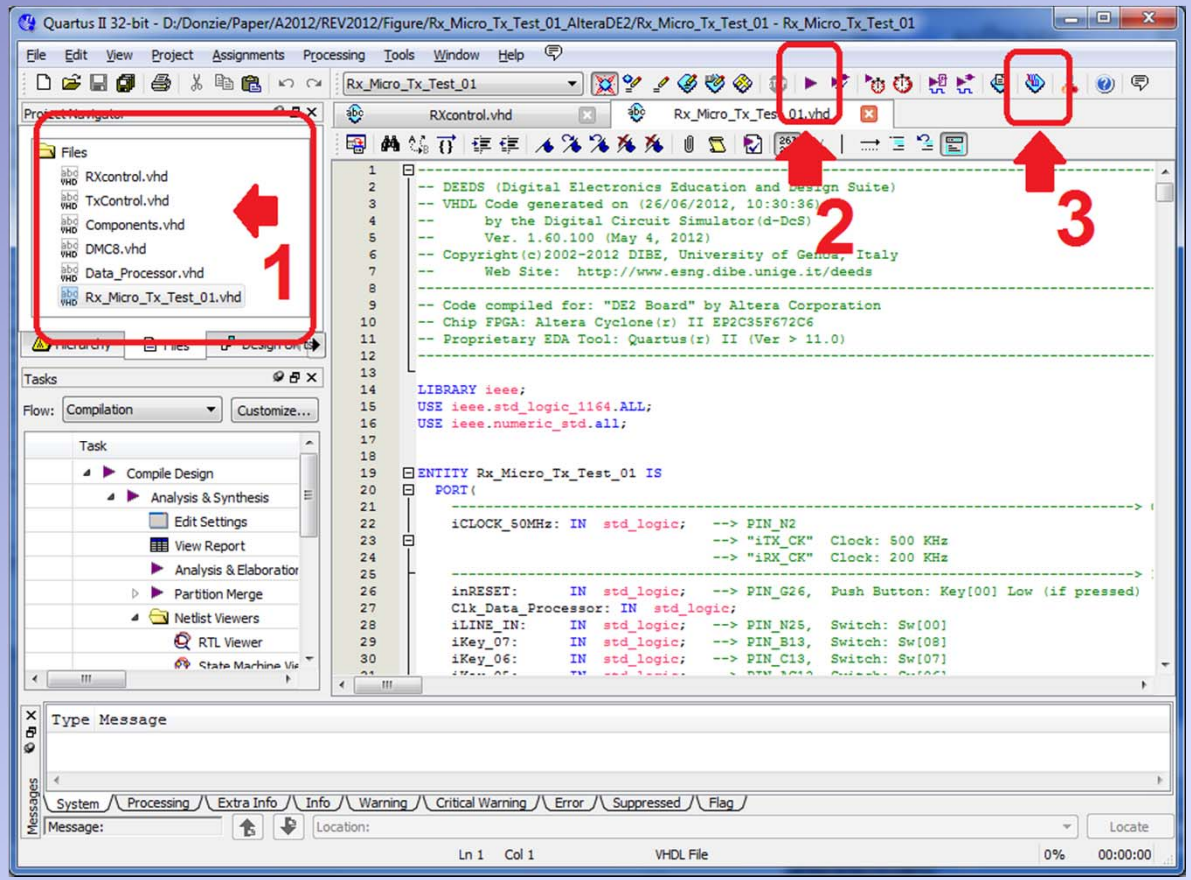

- The generated ***VHDL code*** is ***specialized*** for the selected board, and compliant to the user-defined ***associations***
- The top level Entity definition is more complex that the one generated by the ***"Export VHDL"*** command
- It declares unused inputs and outputs of the board, or modifies those inputs or outputs that require some adaptation because not immediately available (i.e. decoded hex displays).

# *Ready to test project on FPGA!*



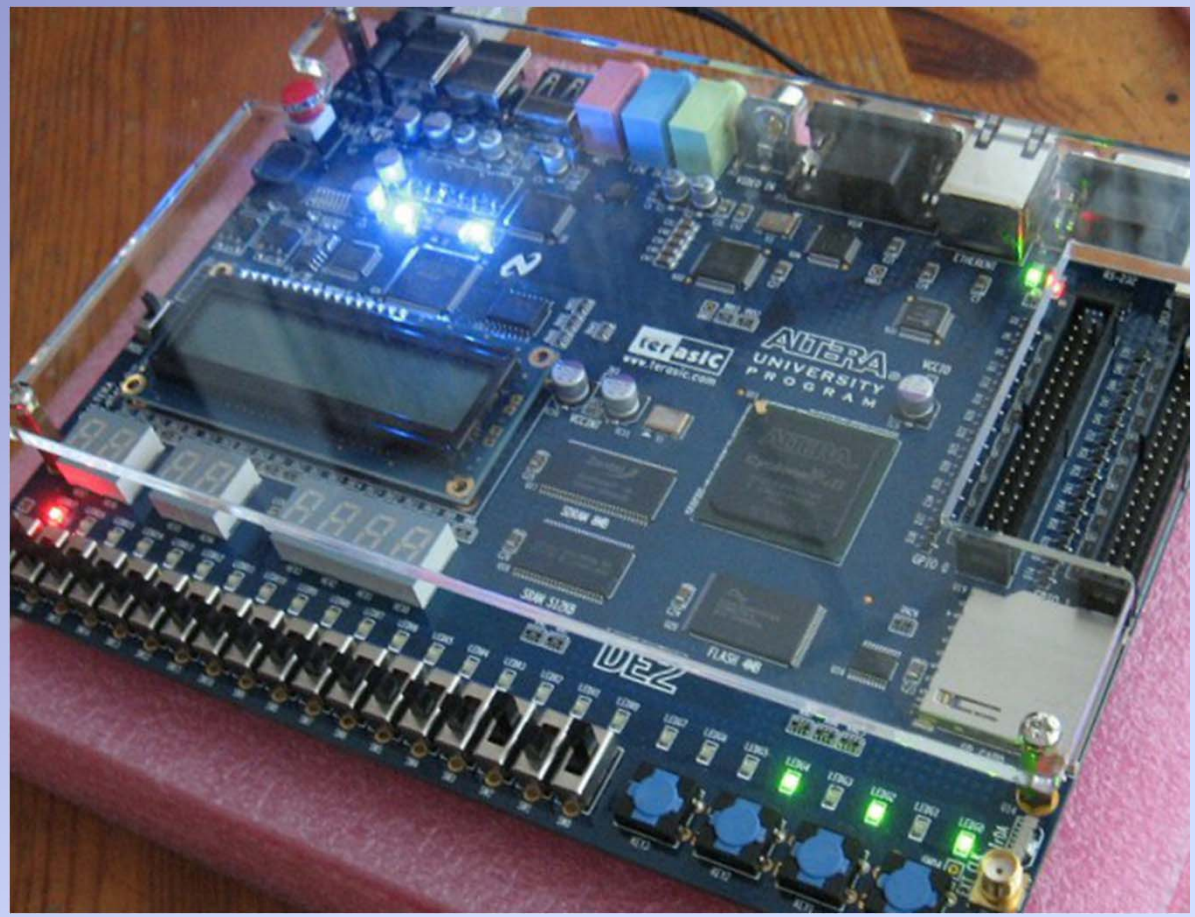

- The ***VHDL generator*** is in charge of building also the ***Project Files*** needed by the ***FPGA specific*** software.
- All the associations between the schematic and the board devices are coded in the ***Project Files***, defining all FPGA pins and board connections
- The user is prompted to launch the ***FPGA specific*** software (Altera Quartus II, in the example aside)

# *Using the FPGA specific software*




- The user has launched the ***FPGA specific*** software (Altera Quartus II, in this example)
- All the VHDL files are ready to be compiled **(1)**
- The user needs only to activate compilation **(2)** and, finally, to download the project binaries into the FPGA board **(3)**, ready to be tested

# *Testing the project on the FPGA board*




- The user, after the download of the programming files into the FPGA, is ready to test the functionality of the project
- The FPGA board has become the system that the user has designed!

# *The lab experiments (1)*

- On December 2010 we started the experimentation in the lab sessions. The laboratory had been equipped with 25 workstations, composed by a PC with Altera's Quartus® II and a DE2 board. Each station hosted two students.
- Preliminary experiments before the release of the FPGA extension were based on the development of a FSM, exported by Deeds as a VHDL file. Two lab sessions were executed by approximately 250 students (2 hours x 2 sessions x 250 students = 1000 lab hours total).
- In the following academic year (2011-12), using the beta version of the FPGA extension, we ran two lab sessions, consisting in FSM design assignments, executed by approximately 200 students (1200 lab hours total).

# *The lab experiments (2)*

- In May 2012, release 1.60 allowed to download in the FPGA also the DMC8 processor. We exploited this feature by adding a new lab with a simple embedded system.
- In the current academic year 2012-13, using the release 1.71 (with a new component library), three lab experiments have been executed by 90 students.
- In the second semester we plan to deliver three new labs using the DMC8 as embedded processor.

# *Report and results from the labs*

- The technical aspects of the labs were almost flawless: no problems with the compilation of the files and the operation of the DE2 boards.
- Students learned the operating procedures much quicker than we expected and were generally able to perform successfully the experiments.
- The most visible result of the labs was the very high level of acceptance by students.
- The number of lab attendees has increased noticeably and their interest has definitely been stronger than with the existing simulation-based labs.

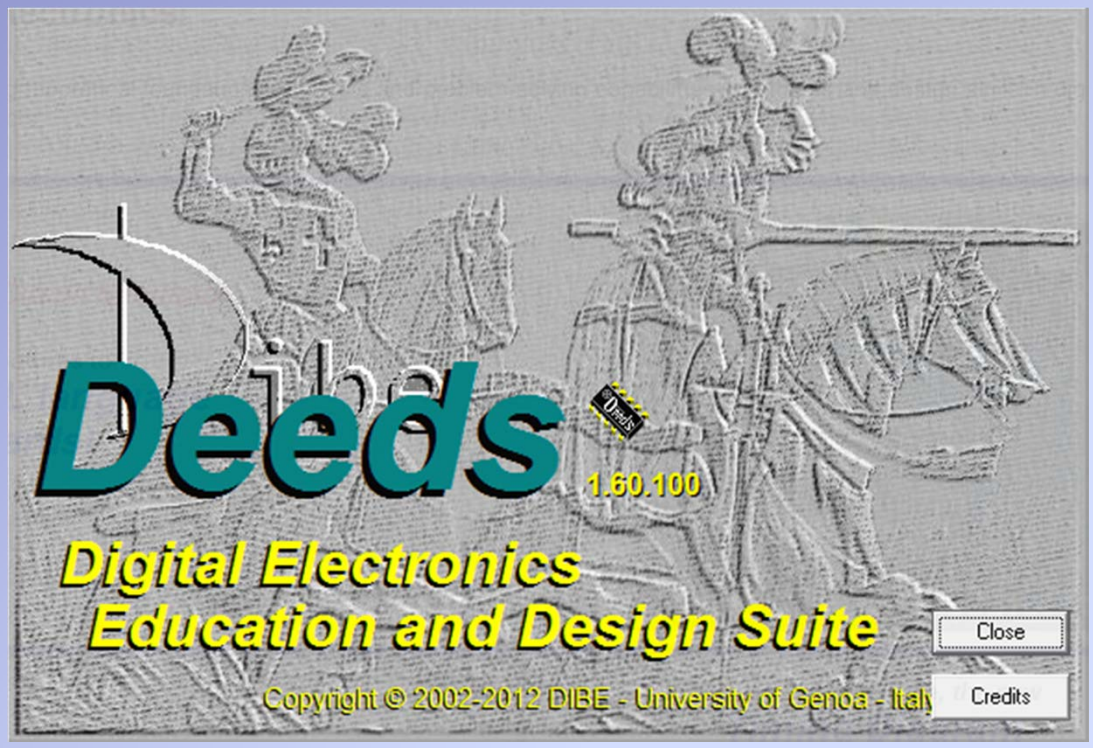
# *FPGA extension: conclusions (1)*

- The FPGA extension opens to Deeds the possibility to export and test project on FPGA boards
- Familiarity with FPGA and the professional FPGA software is not required.
- The FPGA extension focuses students' attention on the essential steps of the design: they can see the result of their work immediately, starting from the first laboratory sessions

# *FPGA extension: conclusions (2)*

- FPGA extension provides a new life to the large amount of simulation-based Deeds learning materials, by re-targeting them toward FPGA.
- FPGA extension ensures continuity between the classic, schematic entry based education and the new approach with HDL and FPGA.
- All Deeds projects (in English and Italian) are available at:
  - http://www.esng.dibe.unige.it/deeds/LearningMaterials
- FPGA projects are indexed here:
  - http://www.esng.dibe.unige.it/deeds/FPGA
- .

# *Thank you for your attention!*




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

*We thank Altera Corporation for the generous donation of the DE2 development boards that have made possible an extensive experimentation in the lab.*