Rapid Prototyping Modules for Remote Engineering Applications

K. Henke, H.-D. Wuttke and Tobias Braune
Ilmenau University of Technology, Ilmenau, Germany

Abstract—**This contribution describes a concept for an integration of microcontroller- and FPGA- based Rapid Prototyping modules into a Remote Lab system and its implementation. The implementation enables a Web-based access to control electro-mechanical models. The electro-mechanical models can be controlled by different platforms (microcontroller or FPGA-based) Students have the possibility to test a designed control algorithm directly within a hardware environment by uploading a source file to the Remote Lab server. That way the students can compare architectures of different control-platforms as well as different design approaches to control electro-mechanical models. The paper starts with an introduction into the design process and describes the tool set, which is provided to the students. After that the different platforms and their integration in the Web-server are introduced. Finally we would like to give examples of such complex design tasks and show, how the students can use different tools during several design steps.**

Index Terms—**Web-based education, Virtual and Remote Labs, Web-based design tools, Distance Learning**

I. INTRODUCTION

The development of Remote Engineering becomes more and more important in a knowledge based society and is used in a wide range of education and social important fields (virtual and Remote Laboratories, teleworking /e-working environments, remote engineering etc.).

New technologies in Remote Engineering have been widely deployed and explored in Europe in the last few years. The Internet is widely used in the whole world and is nowadays the most popular communication network. Therefore, new technologies and mobile networks have to be improved in the next future. This means, that the number of specialists in these fields has to increase dramatically in the next years.

The same trend we can see in overseas. This is related to the growing technical opportunities of the Internet (bandwidth) and new methods in e- and distance learning. The forerunners in this area are engineering disciplines and natural sciences.

Due to the:

- growing complexity of engineering tasks,
- more and more specialized and expensive equipment as well as software tools and simulators,
- required use of these equipment and software tools/simulators in short time projects,
- application of high tech equipment also in small and medium-sized enterprises (SME),
- need of high qualified staff to control recent equipment and
- demand of globalization and division of labour

It is more and more necessary to allow and organize a shared use of equipment, but also specialized software, for example simulators.

Users in the workplace can access the labs without travelling. This flexibility is important for tele-working, education and life long learning. Using online laboratories has the potential of removing the obstacles of cost, time-inefficient use of facilities, inadequate technical support and limited access to laboratories.

This also would benefit people with special needs and people working from their home, as they would not need to travel to their companies' facilities to perform their work. Even employees, working at their company's facilities, can use remote specialized equipment at another company without travelling. This may provide new opportunities and benefit for small and medium-sized enterprises (SME) that would not be able to use such equipment otherwise [1].

Our Department of Integrated Hardware and Software Systems at the Ilmenau University of Technology has 10 years experience dealing with Internet-supported teaching in the field of digital system design [2, 3]. We have developed a new teaching concept, called "Living Pictures" [4], that we use in several phases of the learning process. Living Pictures are highly interactive Java applets that can be used for demonstration, as well as for experimental purposes, and serve also as tools in certain steps of the design process of digital systems. To complete the teaching concept by own experiences, the students have to pass a practical examination in a laboratory. A task during this examination is to design a digital control system that controls one of various electro-mechanical models, for instance a model of a traffic light, a lift, a positioning table etc.

The design of complex digital control systems for Bachelor and Master Courses includes the following lectures:

- "Technical Informatics"
- "Design of Digital Control Systems"
- "Design and Validation of Complex Parallel Systems"

The lecture *Technical Informatics* is held for Bachelor students in the first semester. It deals with the basics of Boolean algebra, combinational logic and simple sequential circuits. Students learn different functional specification techniques (e.g. logical equation, truth table, schematic diagram and automaton based techniques). This course is supported by “Living Pictures” as well as the Web-based usage of a Remote Prototyping board [5,6].

Main topics in the lecture *Design of Digital Control Systems* are various minimization techniques for logical expressions, dynamic effects in combinational and sequential circuits and the design of digital control systems – mainly based on Finite State Machine (FSM) descriptions. This lecture is accompanied by a set of tools, which students can use in a 4-hour laboratory [7].

Students in Master level courses have the opportunity to deepen their knowledge in the lecture *Design and Validation of Complex Parallel Systems.* In this course, the students learn different methods and tool concepts to design and validate complex design tasks. For a better understanding, some accompanying practical designs are beneficial. Unfortunately such designs are too complex for a single lecture or a 4-hour laboratory lesson. That’s why, we organize “project seminars” during a whole semester. The time for the project seminar is calculated with a workload of 120 hours – 4 ECTS.

In our contribution, we would like to give examples of such a complex design task and how the students can use different tools during several design steps.

II. THE DESIGN PROCESS

The design process of Digital Control Systems usually consists of the conceptual formulation and the design of the control algorithm to finally achieve a validated control. Figure 1 gives an overview of this process and describes the design phases with more detail.

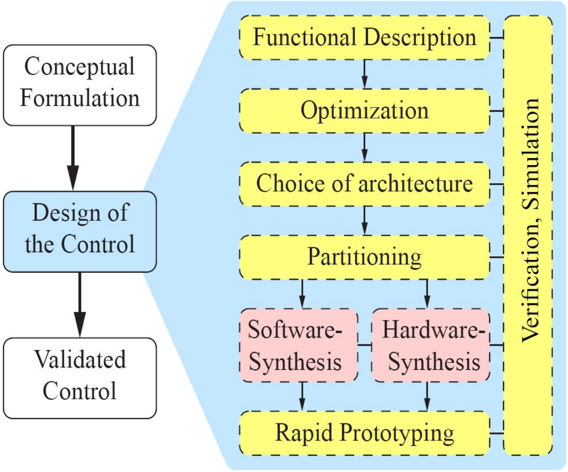


Figure 1. Design Process of Digital Control Systems

For the *functional description* we use e.g. parallel Finite State Machines (FSM), Hardware Description Languages (AHDL, VHDL) or software notations (Assembler, C) as description techniques [8,9].

These description methods allow a formal *optimization*, *verification* and *validation* of the design.

Simulation and visual prototyping help to find functional errors. Run-up to the practical, simulations and animations in “virtual worlds” are often used to verify the developed solutions. The physical behaviour of the actual model, and its environment as well, will be emulated (e. g. as simulation model). The user can influence his “virtual world” and analyse the caused reaction of his control algorithm. Dynamic activities will be animated and in some cases supported by corresponding engine sounds. Figure 2 shows an example of such a visual model - a Water level control, used for a design task.

These steps have to be executed until errors are detected. But there is an essential disadvantage in this method. *Real disruptive factors* (e. g. failure of single components, mechanical problems) can't be recognized by the underlying virtual environmental model.

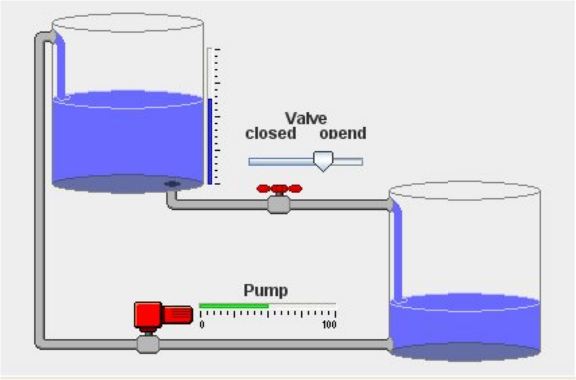


Figure 2. Visual model of the Water level control task

Generally, only a simulation of predetermined malfunctions is possible. After some times all these effects are well known in the student’s community. Unconsidered sources of errors lead to undetected failures of the control because the corresponding environmental situation was not simulated before [10].

That's why, a fault free design algorithm finally should be tested on real electro-mechanical models (e. g. the Water level control unit, shown in Figure 3) in the Remote Laboratory (*rapid prototyping*).

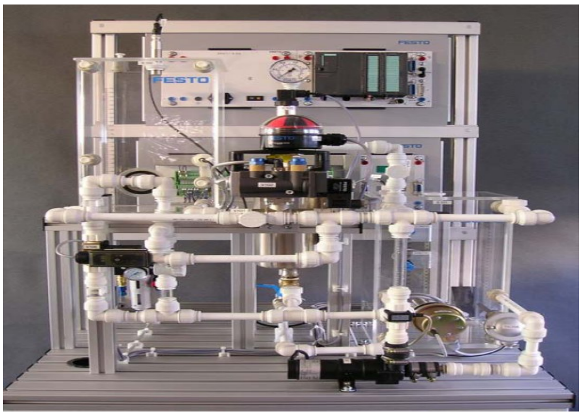


Figure 3. Hardware model of the Water level control unit

The *partitioning* of the control task into software and a hardware part is an essential step in the design process.

During the *synthesis* phase, code for the processor (software) and/or for the programmable logic (hardware) is generated by different design tools. Both parts (software and hardware) can be executed and evaluated on the hardware platform directly.

Assuming real laboratory conditions, disturbances can occur, causing special working conditions. Including such real disruptive factors to an algorithmic test leads in contrary to simulation to a distinctly closer relation to practical conditions. Furthermore, it should stimulate student’s interest in the design of safety critical control systems. Additionally, we would like to give the students the chance to verify their prepared control algorithms and to correct or modify them accordingly to the received results.

III. A TOOL SET FOR RAPID PROTOTYPING BASICS

Especially for Bachelor studies, we developed – in cooperation with the University of Cooperative Education Thuringia in Gera – a Rapid Prototyping board, shown in Figure 4 [6].

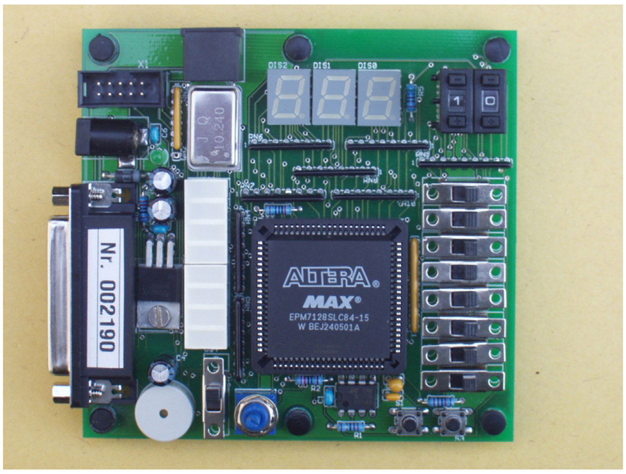


Figure 4. Stand-alone Rapid Prototyping board

Because of the special conceptual design of this board, students can test all the taught topics of the given lectures, e.g. basics of Boolean algebra, combinational logic and simple sequential circuits. Furthermore, they can apply and compare different functional specification techniques (e.g. logical equation, truth table, schematic diagram) with the ability for both local and Web-based remote access.

By using common design tools (e.g. MaxPlus+, QuartusII from Altera [11]), the student is able to specify his design via

- *text based* design methods,
- Here the student can enter his design by means of logical equations, truth tables or hardware description languages (AHDL, VHDL or Verilog).
- *graphically* based design methods,
- The student can use block or schematic diagrams to input his design.
- integrated *FSM editors*.
- In case of sequential design tasks he can directly enter the derived automaton graph (or graphs of parallel automata) with the built in FSM editor.
- The editor itself generates VHDL code for the further design steps.

Another use case of the Rapid Prototyping board is to identify the function of a given design (black box). The FPGA was programmed by the teacher - the student has no ability to reprogram the device.

By manipulating the inputs of the design (e.g. to enter all the input sets of a truth table or special input test vectors), the student attempts to analyse the response (the output signals) in order to find out the function of the given design.

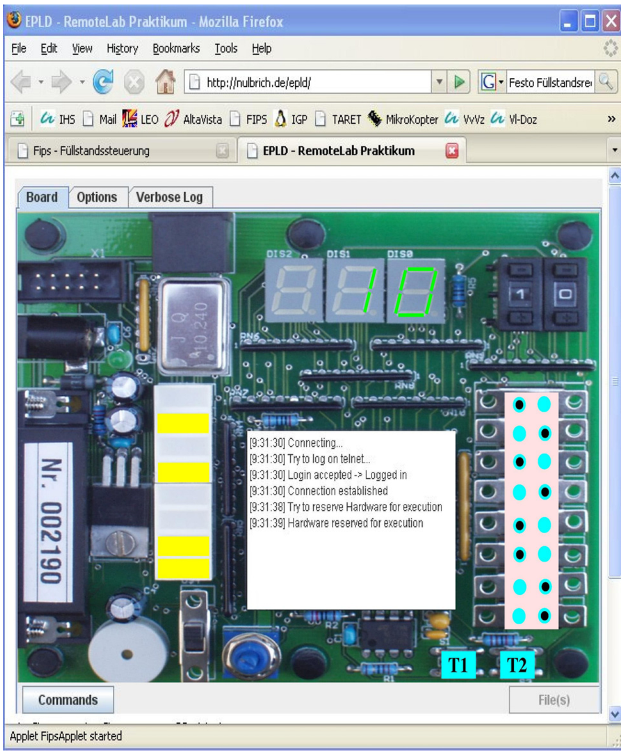
For the Web-based Rapid Prototyping ability we have developed a Web-interface as Java applet. By using this Web-interface, the student is able to

- upload his design,
- program the FPGA,
- handle the lab procedure.

The applet allows the student to manipulate all the inputs of the Rapid Prototyping board (slide switches, HEX-code switches, and pushbuttons).

He can observe the outputs of the board (7-segment displays, row of LEDs) virtual inside the Java applet.

For the look-and-feel of the applet, we use a "photo" of the board as background. All the inputs are realized as Java control elements and can be manipulated via the mouse interactively. The outputs are illuminated directly inside the "photo" elements. That's why, it is not necessary to use an additional Webcam to observe the board in the lab room. The control and communication between the applet and the Rapid Prototyping board is established via text based Telnet commands. Debug messages are displayed in a status window (placed over the FPGA circuit). Figure 5 gives an impression of the applet's Web-interface including the photo of the Rapid Prototyping board.



```
[9:31:30] Connecting...
[9:31:30] Try to log on telnet...
[9:31:30] Login accepted -> Logged in
[9:31:30] Connection established
[9:31:38] Try to reserve Hardware for execution
[9:31:39] Hardware reserved for execution
```

Figure 5. Web-interface of the Rapid Prototyping boar

IV. ARCHITECTURE OF A WEB-BASED RAPID PROTOTYPING PLATFORM

An example for a tool set, supporting all the above mentioned design steps for complex control tasks, is the REAL[1] system, developed at the Department of Integrated Hardware and Software Systems at the Ilmenau University of Technology [12] – see Figure 6. As mentioned in many papers (for instance [13-15]), Remote Labs can open opportunities, allowing an experimental approach for a wider audience and an independence of opening times of the laboratory as well.

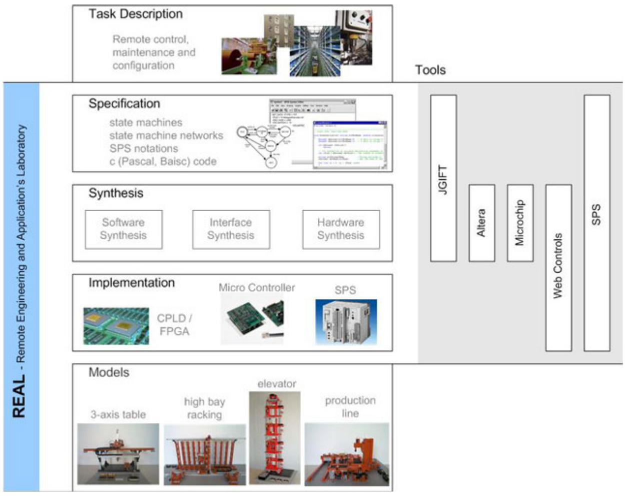


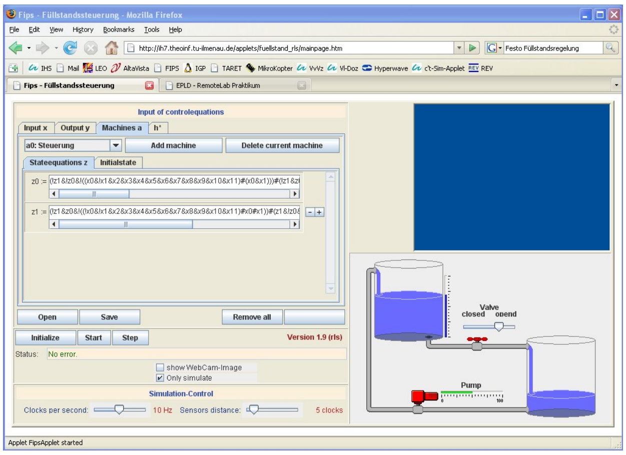
Figure 6. Overview of the *REAL* project

The *REAL* system offers various features. Visualization and animation allow observing and testing the properties of the design. In connection with formal design techniques, simulation and prototyping are used to establish a foundation for the development of a reliable system design during the first iterations of a cyclic design approach.

To check the functionality of the whole design, some special simulation and validation features are included as integral part of the *REAL* system. This offers various possibilities for the execution of simulations, e. g.:

- generation of executable visual prototypes from the general design and parts of it as well,
- properties for step by step and parallel execution of these prototypes,
- visualization of the simulation process with the tools, used for specification as well,
- features for test pattern generation,
- possibilities for analysing the simulation process offline afterwards,
- code generation for hardware and software synthesis as a result of the partitioning process.

REAL offers a Web-based environment including verification and simulation features to generate and execute a design within visual prototypes (e.g. the visual prototype of the Water level control, shown in Figure 7).


Fips - Füllstandssteuerung - Mozilla Firefox

http://ih7.theoinf.tu-ilmenau.de/applets/fuellstand_rls/mainpage.htm

a0: Steuerung

z0 := (!z1&!z0&!((x0&!x1&x2&x3&x4&x5&x6&x7&x8&x9&x10&x11)#(x0&x1)))#(!z1&z

z1 := (!z1&z0&!((!x0&!x1&x2&x3&x4&x5&x6&x7&x8&x9&x10&x11)#x0#x1))#(z1&!z0

Version 1.9 (rls)

Status: No error.

☐ show WebCam-Image

☑ Only simulate

Clocks per second: 10 Hz

Sensors distance: 5 clocks

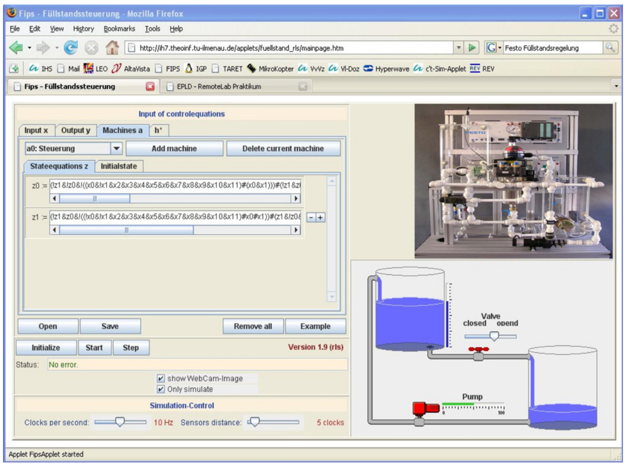
Applet FipsApplet started

Figure 7. Visual Prototyping (offline control) of the Water level control - without Internet connectivity to the Remote Lab

As discussed in the beginning, it is furthermore necessary for students to test their design under real environmental conditions in the Remote Lab. For these purposes, we have developed a Web-interface to control the tools and models. By using this Web-interface (as shown in Figure 8), the student is able to

- download the synthesized hardware and/or software design algorithms - generated and already validated by the *REAL* environment - to test his control algorithm on the real hardware model in the lab room,
- handle the laboratory procedure (start, stop, reset),
- change environmental variables if necessary and
- watch the laboratory procedure by supervising the model and environmental variables inside an I/O monitor or by observing the hardware model directly via a Web-Cam.

At any time the student has the chance to adjust his algorithm in case of fault. Therefore, he is able to achieve a fault free solution (a validated control algorithm) step by step. For more details see [3,10,19].


Fips - Füllstandssteuerung - Mozilla Firefox

http://ih7.theoinf.tu-ilmenau.de/applets/fuellstand_rls/mainpage.htm

a0: Steuerung

z0 := (!z1&!z0&!((x0&!x1&x2&x3&x4&x5&x6&x7&x8&x9&x10&x11)#(x0&x1)))#(!z1&z

z1 := (!z1&z0&!((!x0&!x1&x2&x3&x4&x5&x6&x7&x8&x9&x10&x11)#x0#x1))#(z1&!z0

Version 1.9 (rls)

Status: No error.

☑ show WebCam-Image

☑ Only simulate

Clocks per second: 10 Hz

Sensors distance: 5 clocks

Applet FipsApplet started

Figure 8. Online control of the Water level control - with Internet connectivity to the Remote Lab

---

[1] Goal of the project *REAL* (**Re**mote and **A**pplications **L**aboratory) is to show new ways and chances of remote controlling and remote observation of real processes (e. g. in the fields of control engineering, robotics, tele-control engineering), dealing with integrated and interactive usage of modern internet and intranet technologies, like WWW, HTML, Java etc.

To provide a single user access, a kind of “locking” has to be implemented. It is also required to restrict the access time to the server. This restriction offers an equal chance to work with the Remote Lab for students. That’s why, the following mechanisms will be examined:

- authentication,
- locking using a file,
- locking using a background process,
- locking interactions.,

Apart from the design of the hardware equipment, the focus is set to the “**Remote Lab Server**” (RLS), developed at the Department of Integrated Hardware and Software Systems [5]. It supports various ways of

- client or server based operation of the designed control algorithms,
- access controlling of different users (authentication, LDAP interface),
- global or local networking/communication,
- interactive process manipulation (starting/stopping of the system, influencing environmental variables),
- supervising the hardware equipment via Web-Cam (a video camera with internet connectivity) as well as
- additional support e.g. by mail, chat etc.

On the basis of the plug-in structure of the RLS, new lab tasks, new control mechanisms, new hardware models or even Remote Labs, located at other universities, can be integrated in a flexible way.

In order to ensure an even larger flexibility and utilization of the Remote Lab, it is necessary to connect various variants of the existing Web-control units (Beck-IPC, Microcontroller, FPGA etc. – see next section) directly to a hardware model via the Internet. This is done by an “IP-switch” – a Web-based multiplexer (see Figure 9). Furthermore, this switch can handle necessary support functions like lighting of the hardware setup for each model.

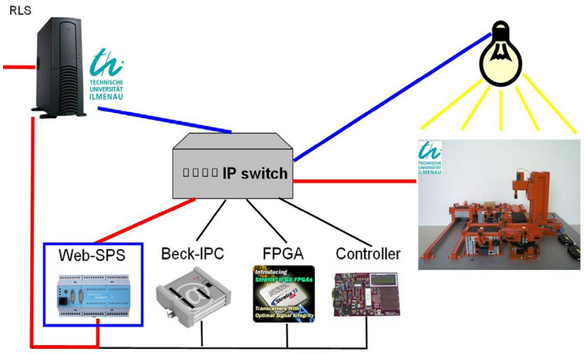


Figure 9. IP switch for the connection model – control unit

V. Fields of Application of the *REAL* System

As shown in Figure 10, the *REAL* system supports the design of control algorithms with different specification techniques by an integration and usage of proprietary design environments (e.g. the *JGIFT* system – see next section) as well as non-commercial software products (e.g. using Quartus II for a hardware-oriented VHDL design or using MPLAB respectively to develop Assembler or C software projects).

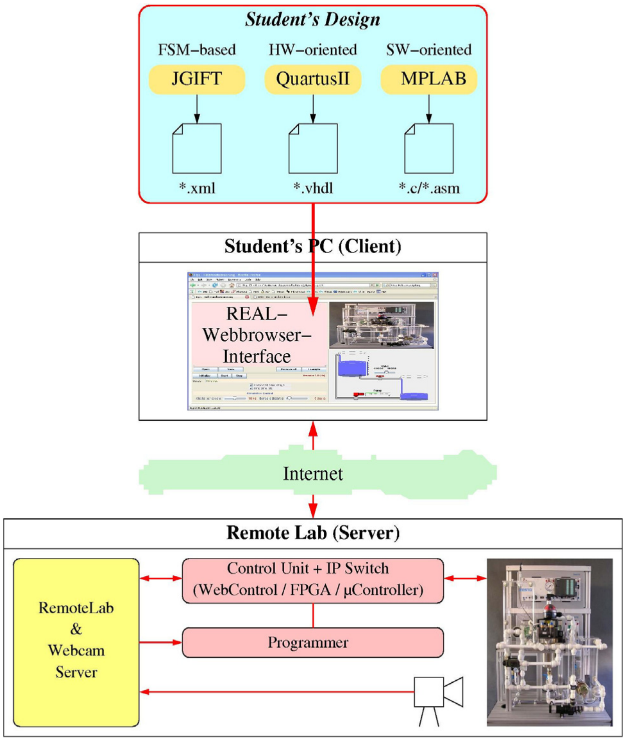


Figure 10. Usage of different description techniques for the design

The students only have to develop the actual control implementation – all additional tasks like

- model self protection mechanisms,
- interface declaration and
- voltage conversion 5/24V

are implemented in separate hardware/software modules, which will be included automatically and compiled within the whole hardware/software project.

One implementation problem, we had to solve, is to protect the Remote Lab against wrong control algorithms of unskilled students. We didn’t want to define so many constraints that only one solution of a design is possible. On the contrary, students should be free in their decisions and develop own creative solutions. They can create their own design strategies and can decide the granularity of control steps as well. The solution, to protect the Remote Lab, is to have a **reference design** and a method to check the students design against this reference design step by step [16] during the execution. Figure 11 shows this idea.

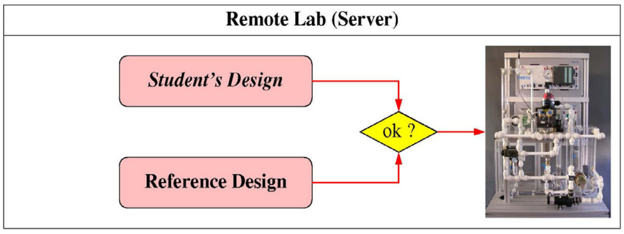


Figure 11. “Observation” of the Student’s design

All input values are inspected by the reference design and the student's design. If the student's design as well as the reference design produces the same output value, the result can be sent to the real hardware model.

The possibilities of the *REAL* system (as shown in Figure 10) we present here exemplarily. Some examples of the Student's design will be discussed in the following.

A. *Stand-alone usage of system components*

The simplest way to use various components of the *REAL* system is a stand-alone solution to control an application (without internet connectivity). This approach should not be focus of this article and is therefore not discussed.

B. *FSM oriented control*

In case of using Finite State Machines as specification technique, based upon an automaton graph, a student can use the *JGIFT* system – a built in design environment of the *REAL* platform.

The abbreviation *"JGIFT"* describes the objectives of this experimental system and stands for

- **J**ava based
- **G**raphical
- **I**nteractive
- **F**SM
- **T**ools

The *JGIFT* system offers different features. Visualization and animation allow observing and testing the properties of the design. In connection with formal design techniques, simulation and prototyping are used, leading to a foundation for the development of a reliable system design during the first iterations of a cyclic design approach.

To check the functionality of the whole design, some special simulation and validation features are included as integral part of the *JGIFT* system. This offers various possibilities for the execution of simulations, e.g.:

- generation of executable visual prototypes from the general design and parts of it as well,
- possibilities for step by step and parallel execution of these prototypes,
- visualization of the simulation process in the tools, used for specification (e.g., FSM editor),
- features for test pattern generation,
- possibilities for analysing the simulation process offline afterwards,
- code generation for hardware and software synthesis as a result of the partitioning process.

Figure 12 gives an impression of the integrated *JGIFT* design environment. The modules of the system can be used as stand-alone Java applet or via the Internet, embedded into any Web-browser. For more details see [7,18].

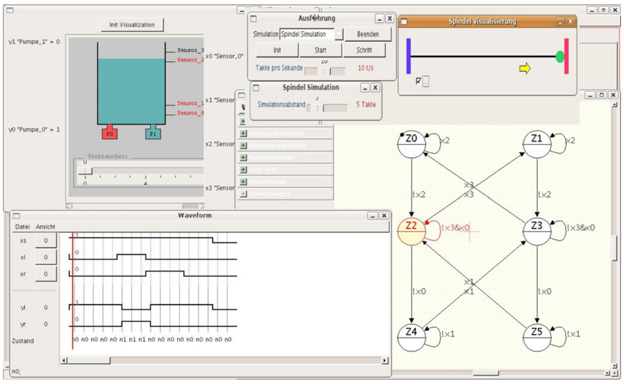


Figure 12. Some features of the *JGIFT* design environment

Assuming, the student achieved a validated design, he gets the required next state and the output equations. By accessing the Web browser interface of the *REAL* system (see Figure 10), he is able to enter his created control algorithm (the received equations), handle the laboratory procedure (start, stop, reset) and change environmental variables if necessary.

These equations for model control, are executed through an *interpreter* – located either on the client or server side.

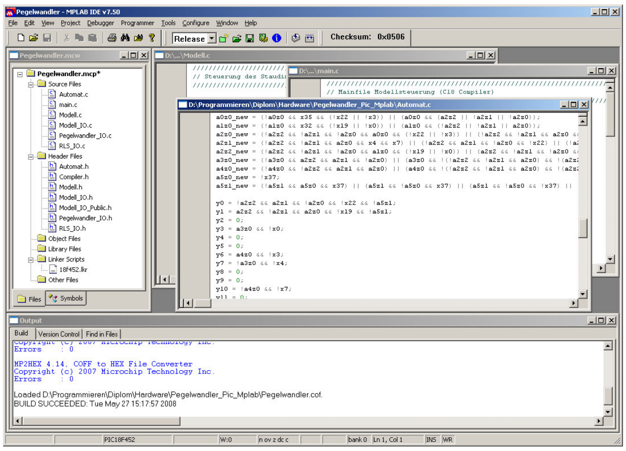
When using the „client based" remote control the hardware model will be controlled "from a distance" through the interpreter running inside the client (implemented e.g. as Java applet). In this case, only the input and output signals of the model will be transferred via Internet.

When executing the control algorithm "server based", the required equations have to be transferred from the browser via the Web server to the interpreter running inside (on the spot) the embedded system only once a time. In this case, the interpreter exchanges data with the model I/O interface directly.

For a detailed description see [4,5].

C. *Software oriented control*

Students can implement their design task directly into a microcontroller for a software oriented specification.



```
////////////////////////
// Steuerung des Staudi
////////////////////////
```

```
//////////////////////////////////////////////////////////////////////
// Mainfile Modellsteuerung (C18 Compiler)
```

```
a0z0_new = (!a0z0 && x35 && (!x22 || !x3)) || (a0z0 && (a2z2 || !a2z1 || !a2z0));
a1z0_new = (!a1z0 && x32 && (!x19 || !x0)) || (a1z0 && (!a2z2 || !a2z1 || a2z0));
a2z0_new = (!a2z2 && !a2z1 && !a2z0 && a0z0 && (!x22 || !x3)) || (!a2z2 && !a2z1 && a2z0 &&
a2z1_new = (!a2z2 && !a2z1 && a2z0 && x4 && x7) || (!a2z2 && a2z1 && !a2z0 && !x22) || (!a
a2z2_new = (!a2z2 && !a2z1 && !a2z0 && a1z0 && (!x19 || !x0)) || (a2z2 && !a2z1 && !a2z0 &&
a3z0_new = (!a3z0 && a2z2 && a2z1 && !a2z0) || (a3z0 && !(!a2z2 && !a2z1 && a2z0) && !(a2z
a4z0_new = (!a4z0 && !a2z2 && a2z1 && a2z0) || (a4z0 && !(!a2z2 && !a2z1 && a2z0) && !(a2z
a5z0_new = !x37;
a5z1_new = (!a5z1 && a5z0 && x37) || (a5z1 && !a5z0 && x37) || (a5z1 && !a5z0 && !x37) ||

y0 = !a2z2 && a2z1 && !a2z0 && !x22 && !a5z1;
y1 = a2z2 && !a2z1 && a2z0 && !x19 && !a5z1;
y2 = 0;
y3 = a3z0 && !x0;
y4 = 0;
y5 = 0;
y6 = a4z0 && !x3;
y7 = !a3z0 && !x4;
y8 = 0;
y9 = 0;
y10 = !a4z0 && !x7;
```

```
Copyright (c) 2007 Microchip Technology Inc.
Errors    : 0

MP2HEX 4.14, COFF to HEX File Converter
Copyright (c) 2007 Microchip Technology Inc.
Errors    : 0

Loaded D:\Programmieren\Diplom\Hardware\Pegelwandler_Pic_Mplab\Pegelwandler.cof.
BUILD SUCCEEDED: Tue May 27 15:17:57 2008
```

Figure 13. Usage of the MPLAB IDE for a software oriented design

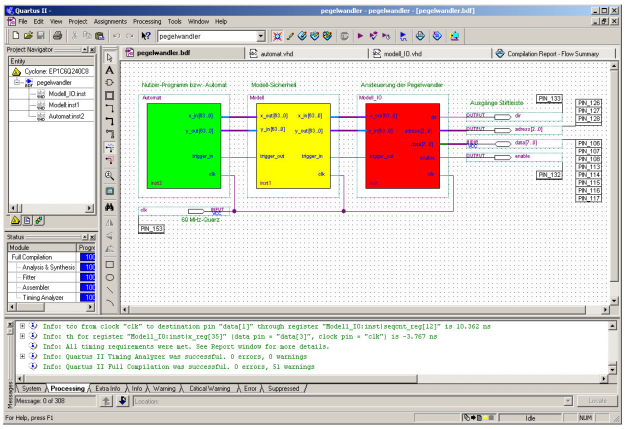
Therefore, they use common (non-commercial) development tools, (e. g. MPLAB IDE and/or C18 C compiler from Microchip [17] to develop Assembler and/or C software projects) shown in Figure 13.

The generated software control module (Assembler or C code) is then transferred via *REAL* Web-interface to the Remote Lab Server (see Figure 10). The Server automatically embeds the control module into a locally stored workbench project which ensures model safety etc. After compilation, the project is programmed into the microcontroller. Now, the student can begin with his lab work to proof, if his algorithm fulfils the requirements of the given task.

A software implementation of the control task is suitable for simple and non time-critical requirements.

D. Hardware oriented control

If a student prefers an exclusive hardware oriented design (without any software components), using a hardware description language like VHDL or AHDL as specification technique, he can prepare his design task with common development tools - e.g. MaxPlus+, Quartus II- shown in Figure 14. The generated hardware module (e.g. a VHDL module) is uploaded via the *REAL* Web-interface to the Remote Lab Server (see Figure 10) which will synthesize the code using a locally stored workbench project to ensure model safety, etc. After programming the connected FPGA, the student can begin his lab work.



```
Info: tco from clock "clk" to destination pin "data[1]" through register "Modell_IO:inst|seqcnt_reg[12]" is 10.362 ns
Info: th for register "Modell_IO:inst|x_reg[35]" (data pin = "data[3]", clock pin = "clk") is -3.767 ns
Info: All timing requirements were met. See Report window for more details.
Info: Quartus II Timing Analyzer was successful. 0 errors, 0 warnings
Info: Quartus II Full Compilation was successful. 0 errors, 51 warnings
```

Figure 14. Usage of the QuartusII IDE for a hardware orientated design

For time-critical as well as parallel applications, a hardware implementation of the control task is the preferred solution.

VI. CONCLUSION

A universal hardware platform has been discussed which can be used for Rapid Prototyping of digital control systems for complex control design tasks including the ability for both local and Web-based remote access.

In the projects present state, only one student at a time can gain access to the hardware model (single user mode). In future, this project will offer cooperative work features (multi user mode). Therefore, it is possible to manipulate the hardware model simultaneously via several parallel control algorithms of a distributed control prepared by various students.

For an effective usage of the *REAL* system within Learning Management Systems (LMS), the reference design and a method to check the students design against this reference design step by step will be traced by the LMS. Figure 15 shows this idea.

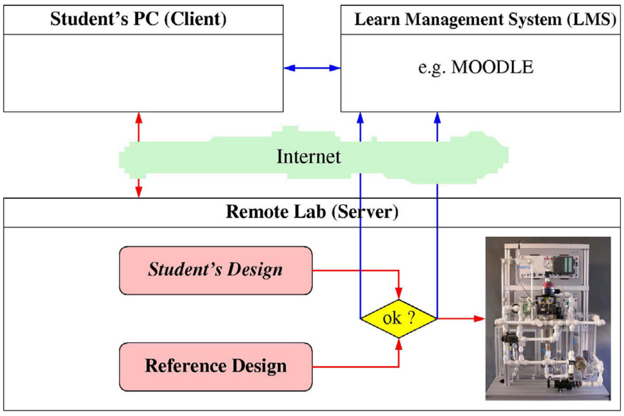


Figure 15. Observation of the student's design under LMS control

If the student's design as well as the reference design produces the same output value, the result can be sent to the Remote Lab. Otherwise, the LMS will be informed and the student gets some hints to correct his design. For a detailed description see [16,20].

Our Department of Integrated Hardware and Software Systems at the Ilmenau University of Technology is involved in different national and international e-Learning projects ([1,21]) where it is increasingly necessary to allow and organize a shared use of equipment. That's why, the main focus of the project *REAL* is a Web-wide usage of different design tools as well as Remote Labs for digital control tasks.

Using both, online tool support and laboratories, has the potential of removing the obstacles of cost, time-inefficient use of facilities, inadequate technical support and limited access to design and laboratory resources. It is also a benefit for students and researchers with special needs and students/researchers working from home, so they do not have to travel to their companies' facilities to perform their needed work. Even students/researchers working at their university's facilities can easily access remote specialized equipment located at another university without travelling.

Students will learn more about the opportunities and limits of remote control and observation via Internet on practical examples. The student – beside knowledge consolidation through design and practical experiments using these new Internet technologies – is forced to estimate the technologies critically.

During the implementation of the *REAL* system, great importance was attached to the

- reuse of the applied methods and tools for similar problems,
- reuse of the developed software modules,

which will finally lead to synergy effects in the production process of different Remote Engineering tools.

It is planned to use the *REAL* system within the TARET[i] summer school "Remote and virtual Applications" 2009 in Maribor/Slovenia. The goal is to mediate fundamentals, applications and experiences in these fields of remote engineering by an interdisciplinary approach in combination with “learning by doing” phases. In this way, we promote transnational cooperation and exchanges between students and teachers of five universities; encourage efficient and multinational teaching of special topics, which could not be taught in this quality in the partner universities of the MARE project [1].

We believe that Remote Engineering provides the ideal framework for exploring new ideas in collaborative virtual reality environments, advanced scientific visualization and advanced interfaces to high-performance computing systems and laboratories.

ACKNOWLEDGMENT

The authors would like to thank Prof. Arndt Seifert from the University of Cooperative Education Thuringia in Gera for the supply and the Web-based usage of the Water level control unit in his Automation Lab as well as Jürgen Schmidt and Friedemann Schoener for the inspiring discussions during the conception of the project.

Many thanks also to Stefan Ulbrich and Sascha Franke for the design and implementation of the Web-interfaces for the Rapid Prototyping board.

REFERENCES

[1] M.E. Auer, I. Grout, K. Henke, R. Safaric and D. Ursutiu: "A Joint Master Program in Remote Engineering", International Journal of Online Engineering (iJOE) [Online], Vol. 2, No. 2, 30 Apr (2006)

[2] S. Sire, F. Geoffroy and D. Gillet: “A Virtual Assistant for Sending Hints and Perturbations to Students based on an Electronic Laboratory Journal (eJournal)”, Proceedings of the ITHET’03, Marrakech, Morocco, July 7-9, 2003.

[3] K. Henke and H.-D. Wuttke: “Web-based educational tool access”, IASTED International Conference Computers and Advanced Technoly in Education – CATE 2003, Rhodes, Greece, 30 June-2 July, 2003.

[4] H.-D. Wuttke and K. Henke: “Living Pictures – tool-oriented learning modules and laboratory for teaching digital via internet”, Proceedings of the ICEE-2002 International Conference on Engineering Education UMIST, Manchester, Great Britain, August 18-22, 2002.

[5] T. Braune: “Rapid Prototyping components for Remote Engineering Applications” (in German), Master Thesis, Ilmenau University of Technology, Germany, 2007.

[6] St. Ulbrich and S. Franke: “Case studies for a Rapid Prototyping Development board” (in German), Study Thesis, Ilmenau University of Technology, Germany, 2008.

[7] JGIFT: “Java-based Graphical Interactive FSM Tools”, http://www.tu-ilmenau.de/fakia/Projekte.1749.0.html, TU Ilmenau

[8] E. Hopcroft, R. Motwani and J.D. Ullman: “Introduction to Automata Theory”, Addison-Wesley, 2001.

[9] J.O. Hamblen and M.D. Furman: “Rapid Prototyping of Digital Systems”, Kluwer Academic Publishers, 2002.

[10] K. Henke, H.-D. Wuttke and S. Hellbach: “Laboratory via Internet – new ways in education and research”, Int. Journal of Computers and Applications, vol. 25, ACTA press, 2002.

[11] Altera Corp., Homepage: www.altera.com

[12] REAL: “Remote Engineering and Applications Laboratory”, http://www.tu-ilmenau.de/fakia/Projekte.1749.0.html

[13] Y. Torroja, et al.: “A Modular Environment for Learning Digital Control Applications”, Microelectronics Education, Marcombo, S.A, 2002.

[14] T. A. Fjeldly, J. O. Strandman, R. Berntzen and M. S. Shur: “Advanced Solutions for Laboratory Experiments over the Internet”, Engineering Education and Research-2001, Begell House Publishing.

[15] H.-D. Wuttke, K. Henke and N. Ludwig: "Remote Labs versus Virtual Labs for Teaching Digital System Design", Proceedings of the Int. Conf. On Computer Systems and Technologies CompSysTech’05, Varna, 2005.

[16] H.-D. Wuttke, R. Ubar, K. Henke and A. Jutman: “Assessment of Student’s Design Results in E-Learning-Scenarios”, 8th Conference on Information Technology Based Higher Education and Training (ITHET2007), Kumamoto City, Japan, 10-13 July, 2007.

[17] Microchip Corp., Homepage: www.microchip.com

[18] K. Henke and H.-D. Wuttke: "A Mixed-Reality Environment for Digital Control Systems", International Conference on Remote Engineering and Virtual Instrumentation, REV2005, Brasov, Romania, 29 June-1 July, 2005.

[19] K. Henke, H.-D. Wuttke and T. Braune: “Virtual and Remote Labs in the Educational Process”, International Conference on Remote Engineering and Virtual Instrumentation, REV2007, Porto, Portugal, 25-27 June, 2007.

[20] K. Henke: “Reusable Assessment Objects for Learning Management Systems”, Computers and Advanced Technology in Education"(CATE 2007), Beijing, China, 8–10 October, 2007.

[21] TARET – The intensive program “Training in advanced remote technologies”, Homepage: http://www.online-lab.net/taret/

AUTHORS

K. Henke is with the Ilmenau University of Technology, Faculty of Informatics and Automation, Department of Integrated Hardware and Software Systems, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: karsten.henke@tu-ilmenau.de).

H.-D. Wuttke is with the Ilmenau University of Technology, Faculty of Informatics and Automation, Department of Integrated Hardware and Software Systems, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: dieter.wuttke@tu-ilmenau.de).

T. Braune is with the Ilmenau University of Technology, Faculty of Informatics and Automation. He obtained his MoS in Information Engineering in 2008 (e-mail: tobias.braune@stud.tu-ilmenau.de).

Manuscript received 01 june 2008. Published as submitted by the authors.

---

[i] TARET – The intensive program “Training in advanced remote technologies” is part of the implementation of the MARE (Master study program in Remote Engineering – see [21]) curriculum.