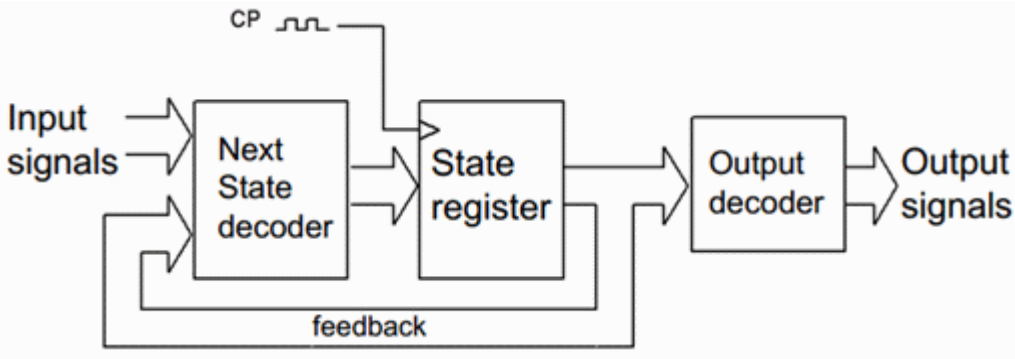
Laboratory VHDL introduction

Digital Design IE1204 (Note! *not* included for IE1205)



Attention! To access the laboratory experiment you must have:

- a booked lab time in the reservation system (Canvas).
- completed your personal knowledge control on the Web (Web-quiz).
- done all preparation tasks mentioned in the lab booklet.

During the lab you work in groups of two, but both students are responsible individually for their preparation and implementation.
Booth students should bring their lab booklets. This frontpage is used as your **receipt** that the lab is completed. Save the receipt until you have received the full course registered in the database (Ladok).

*Since this is your receipt you **must** fill in the table with ink.*

Name:			
Social Security Number:			
• Knowledge Control (Web-quiz)			
Bundle no:		Date:	
Lab-assistant receipt:			
• Preparation tasks in the lab booklet			
Lab-assistant receipt:			
• Lab implementation			
Laboration date:			
Lab-assistant receipt:			

Introduction

This lab is about how to design digital logic with VHDL language and modern CAD software. The idea is that you'll get a glimpse of how a "Digital" engineer work. VHDL language is a very complex programming language, and it is not reasonable to "learn" that this brief first Digital Design Course.
When you solve the lab assignments, you have therefore been given tutorials and template code on the course web.

The school has several good VHDL courses that can be chosen by those who want to know more, and who want to work with Digital Design in the future.

The best way to get to know a program is to install it on your own computer. Then you can in peace searche through menus and help pages, and can take the time it takes to sort out what you might have misunderstood.

If "computer hassles" threaten to consume too much time for you, you can also find the programs installed in the school's Windows computer rooms - as a backup solution.

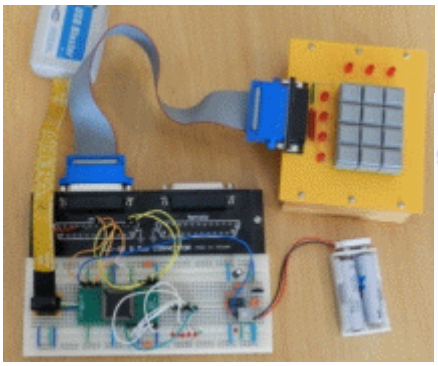
The goal of the lab

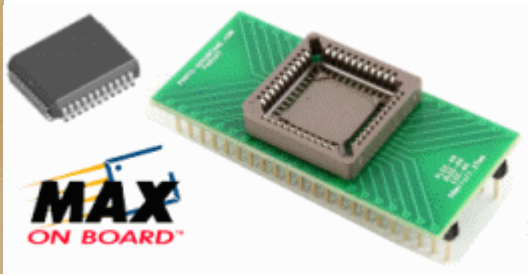
- Become familiar with modern CAD software, Quartus and ModelSim.
- Show how to simulate a digital design (a Moore-machine), how to generate input signals, stimuli, and how to observe the outputs and behavior (ModelSim-Wave).
- Orienting yourself on how a digital technician can write a VHDL "test bench" to ensure that the construction is completely correct.
- Practice the VHDL-construction of a state machine from a given state diagram (revise and expand a given template program).
- Practice how to tie together the design "signals" with target chip "pins".
- Show how you program the target chip (MAX3000), and trying the operation in reality.

Attention! Your lab time may be prior all course elements that may be needed for the lab has been lectured. You would then have to read the course material for yourself in advance - there are links to all slides for the lectures and exercises.

Attention! Lab equipment is completed. No wires should be changed, not added or removed.

A VHDL-code lock





Lab task is to construct a code lock that opens to a unique four-digit code, but we begin by studying a simpler template program, a code lock that opens to one key!

- **Preparation task 1 (done before the lab at home)**

Install the programs **Quartus II** and **ModelSim** on your own computer.

Follow the steps in the tutorial on the course web - Install the programs on your computer.

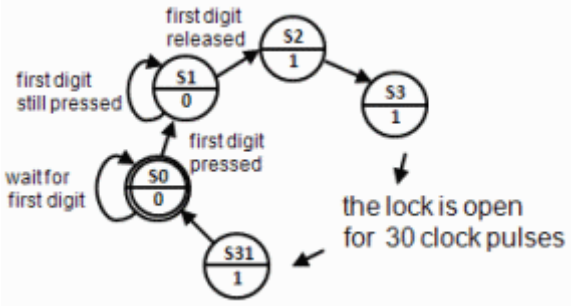
- **Laboratory task 1 (do at lab in school)**

Log on to the lab computer. On the school centrally managed lab computers You do not have rights to install software. **Quartus II** and **ModelSim** are already installed. You may not access the folders under `C:\`. At the laboratory, you should therefore use your "server" folder `H:\`.

- Create a folder `H:\MAXWork\` for the files in this lab.
- Important operating system setup. Show file extensions should be set at all programming courses!
  Windows 7 show fileextensions

- **Preparation task 2 (done before the lab at home)**

Start Quartus and create a project `codelock`. Bring the content of the file lockmall.vhd as the project VHDL-file and then compile the code.
Follow the steps in the tutorial on the course web - VHDL-program with Quartus.



Read about the template program VHDL code in the description on the course web
- VHDL for a codelock.

Read on how to tie the signal names to specific pins of the target chip in Quartus.
- Pin-planning in Quartus.

Read about how to use the Quartus programming function with a JTAG USB Blaster.
- Chip-programing with Quartus.

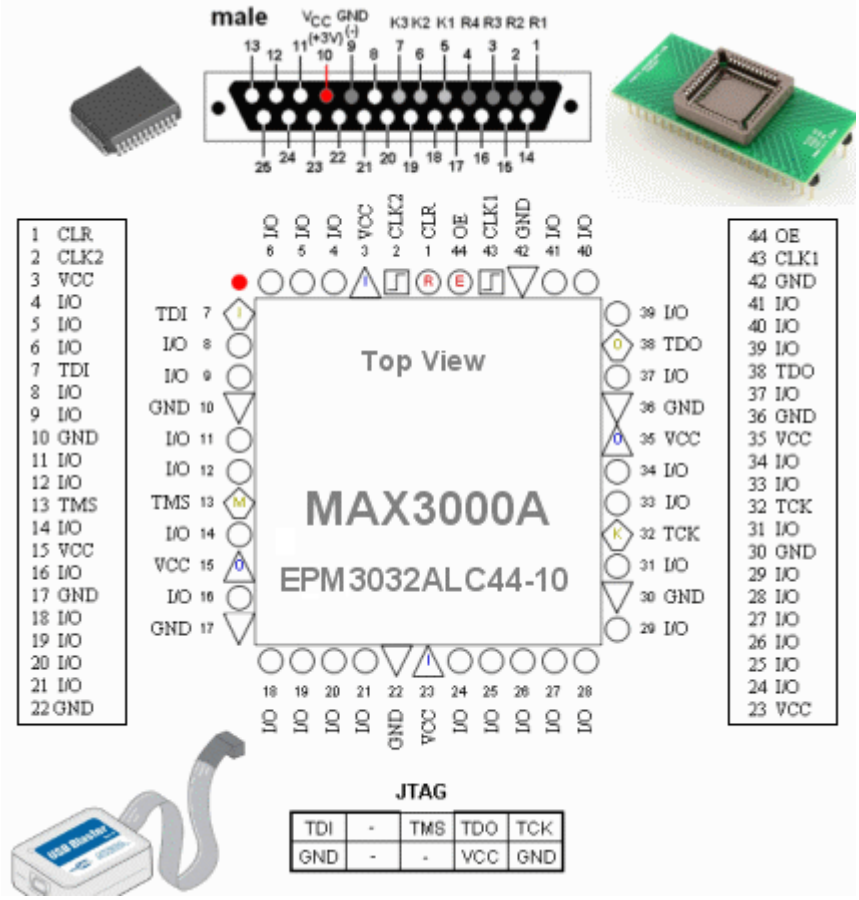
- **Laboratory task 2 (do at lab in school)**

- Start Quartus and create a project `codelock` in your server-folder `H:\MAXWork\`. Use the content of the file lockmall.vhd as the project VHDL-file and then compile the code.

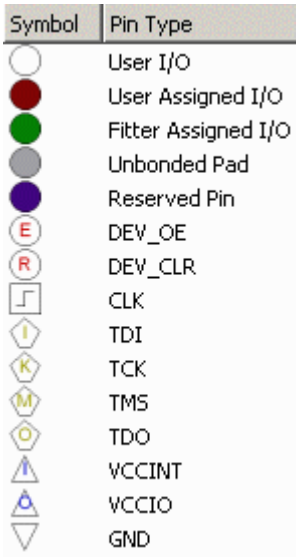
- Lab equipments have different wiring! Examine your lab equipment and enter the Pin-planning table, and thereafter **Pin Planner** in Quartus.

Follow the steps in the tutorial on the course web - Pin-planning in Quartus

Node Name	Direction	Location
clk	Input	PIN_
K[1..3]	Input	
K[1]	Input	PIN_
K[2]	Input	PIN_
K[3]	Input	PIN_
q[4..0]	Output	
q[4]	Output	PIN_
q[3]	Output	PIN_
q[2]	Output	PIN_
q[1]	Output	PIN_
q[0]	Output	PIN_
R[1..4]	Input	
R[1]	Input	PIN_
R[2]	Input	PIN_
R[3]	Input	PIN_
R[4]	Input	PIN_
UNLOCK	Output	PIN_



Description of the pin-symbols se figure.



- When you have completed the pin planner in Quartus so recompile the project.
- Program the device with the USB blaster.
- Check that the code lock opens when you press the "1" and then release the key.

- **Preparation task 3 (done before the lab at home)**

At home, you have no hardware, no laboratory equipment. In such situations one usually simulate the code to see if it is correct.

The leading simulation software **ModelSim** is available in a version for Altera's chips. Start ModelSim and simulate the VHDL-code with the content from `lockmall.vhd` as VHDL-file.

Follow the steps in the tutorial on the course web - Simulate with ModelSim.

lockmall.vhd lock.do

- **Laboratory task 3 (do at lab in school)**

Even when having access to the hardware, it is common to mix simulations with hardware test.

Carry out the same simulation in school that you practiced at home, ie show in the **wave window** that the lock opens for "1". Show Your lab assistant your "simulation expertise."

- **Preparation task 3 (done before the lab at home)**

Design of Digital hardware can often result in the production of an ASIC - An application specific Integrerated Circuit. It is then often several months of lead time and, manufacturing costs of the order of several million dollars.

Then you have to be sure that the design is absolutely correct!

(At the lab, we have a better starting point than the ASIC designer. If your design on a programmable CPLD chip is wrong, You get the chance to reprogram it - again and again.)

As you can see, it is the test engineer who is Digital Technology Hero!

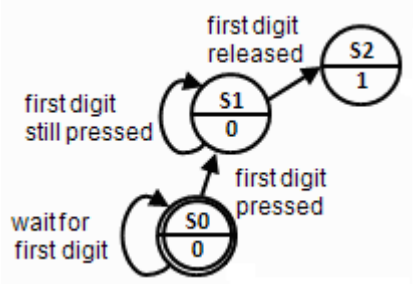
VHDL-language has various tools to enhance the ability to be able to write correct code.

- To reduce the risk of errors when transferring information from the data sheets, you can use index that runs up or down, to suit the method that was used in the data sheet.
- You has also the ability to create user-defined data types that fit the description of the construction. One can therefore often write VHDL code that is "obviously" correct!
- One can write a VHDL test bench. This is simulation code that can be used to test many/all signal combinations that the circuit may be exposed to.

Attention! A test bench is usually a more complicated program than the original design as it relates to test!

These lines from the template program is an example of code that *obviously* follow the given state diagram.

```
case state is
      when 0 => if (K = "001" and R ="0001")     then nextstate <= 1;
                else nextstate <= 0;
                end if;
      when 1 => if (K = "001" and R ="0001")     then nextstate <= 1;
                elsif (K = "000" and R = "0000") then nextstate <= 2;
                else nextstate <= 0;
                end if;
                . . .
```



However, if one takes over the code from someone else, even if that person *promise* that it works, the situation is different.

```
case state is
      when 0 => if(((R(2)='0') and (R(3)='0') and (K(2)='0') and
(K(3)='1')) and
                   ( not (( not ((K(1)='0') and (R(1)='0') and (R(4)='1')))
and
                   ( not ((K(1)='1') and (R(1)='1') and (R(4)='0'))))))
                then nextstate <= 1;
                else nextstate <= 0;
                end if;
                . . .
```

Here are the conditions written in such a way that it is no longer obvious what the code does - And therefore we do not know if it is correct or if, despite all the promises, it is incorrect?

Try now the code correctness with a (pre-written) test bench

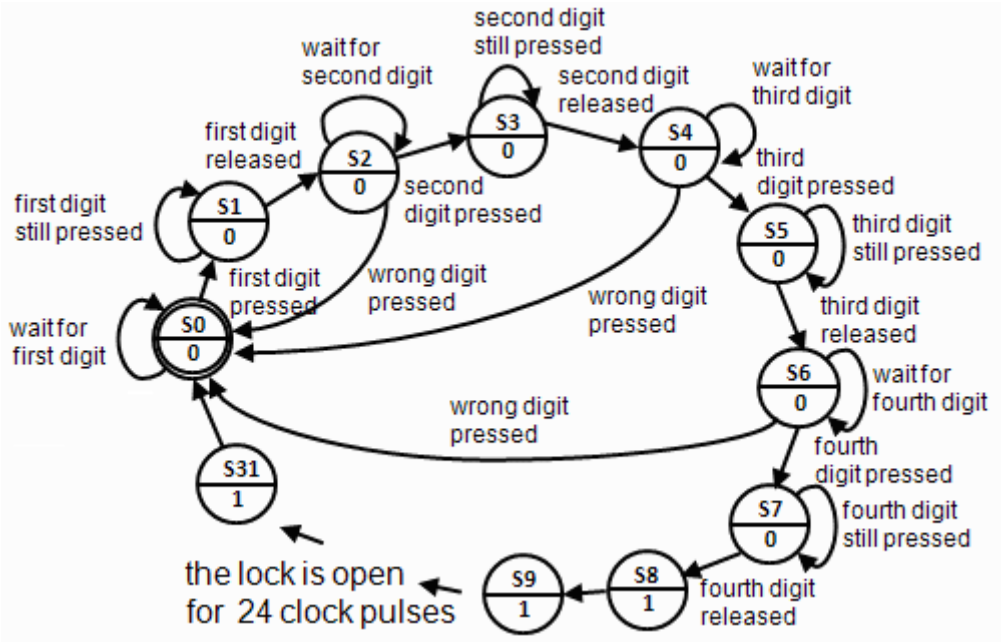
Follow the steps in the tutorial on the course web - Testbench in ModelSim.

lockmall.vhd lockmall_with_error.vhd tb_lockmall.vhd

- **Laboratory task 4 (do at lab in school)**

- Perform also the simulation of the test bench in the school, and show lab assistant your skills by simulating code with a test bench.
- Can you reveal anything wrong with the code? Show lab assistant.
- Close **ModelSim** and change to the program **Quartus II**. There you change the contents of the VHDL file from `lockmall.vhd` to `lockmall_with_error.vhd`. Compile and download the code to the MAX-chip.
- Check if the suspicious behavior from the simulation means that in practice You can open the code lock incorrectly?

- **Preparation task 5 (done before the lab at home)**



A code lock that opens to a single key press is of course ridiculously easy to force, normally you have a four digit code. Your task will be to develop the template program (`lockmall.vhd`) to such a lock with four digits - the last four digits of your civic number.

Prepare such a VHDL program at home. Check that it is possible to compile. Take with you the code to the school in any way, such as:

- Mail the text to yourself.
- Bring a USB flash drive with the code as a text file.
- Transfer the code to your server folder `H:\`.

Are you able to simulate the code at home, you'll increase the likelihood that you have the correct code from the start at school.
The time at the lab school will *not* be enough to write the program code from "scratch"!

- **Laboratory task 5 (do at lab in school)**

Make a code lock that opens to a four-digit code. Lab Assistant determines the digit combination so that - Two of the numbers taken from your preparation program, and the other two from your lab colleague preparation program.

- Show the working code lock for lab assistant.

Do you have time for more?

If you are well prepared for the lab, and if you are not suffering from intermittent connections or dead batteries, then you probably now have time for a "voluntary" task.

- Can you change the program so that it ***also opens*** for the previous *hidden* key combination?
  (so that there is a "hardware trojan" within the chip)

Good Luck!

When you are finished clean the lab desk.

Bill of materials

The "bill of material" for the lab equipment, could be helpful if you ever would need to use simple MAX-chips yourself.

```
Altera USB-blaster ELFA 73-898-90
ALTERA CPLD EPM3032ALC44-10
Proto Advantage PLCC-44 Socket to DIP-44 Adapter
Breadboard MB-85 ELFA 48-428-37
Contact breadboard to 25-pol D-sub ELFA 48-426-96
Pin contact (JTAG) ELFA 43-155-03
Lightdiode with series resistor 5V red  ELFA 75-012-59
Lightdiode with series resistor 5V yellow  ELFA 75-015-11
Electronic chip 555 ELFA 73-042-65
Trimming Potentiometer 500 kO with adjustment knob ELFA 64-635-25
```

William Sandqvist `william@kth.se`