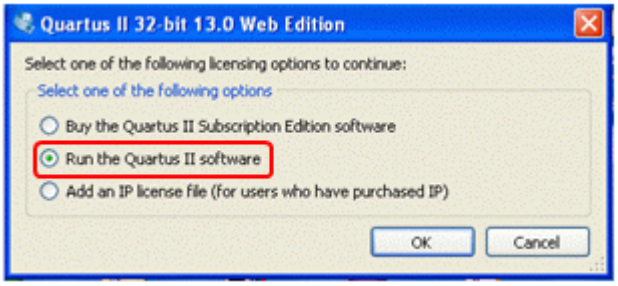
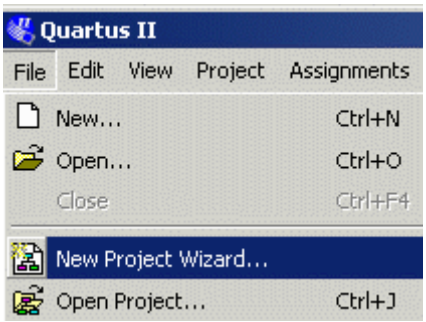
VHDL-program with Quartus

Choose the right program version from the school's start menu :

```
Altera 13.0.1.232 Web edition\
    Quartus II Web Edition 13.0.1.232\
         Quartus II 13.0sp1 (32bit)
```

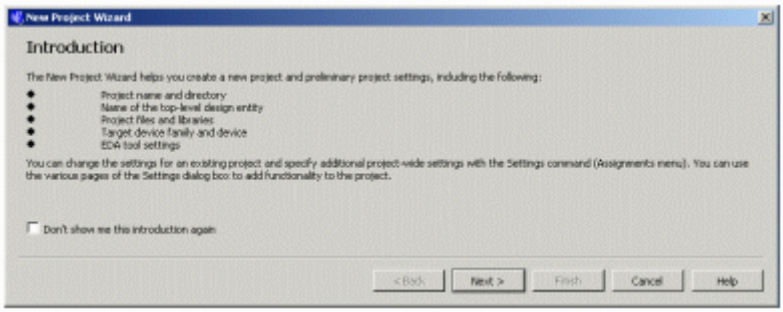


Start Quartus. You need no license and you do not need not buy anything.
If not be directly offered to start **New Project Wizard**, You may also select this option from the **File** menu.



Introduction

Clic on **Next**.

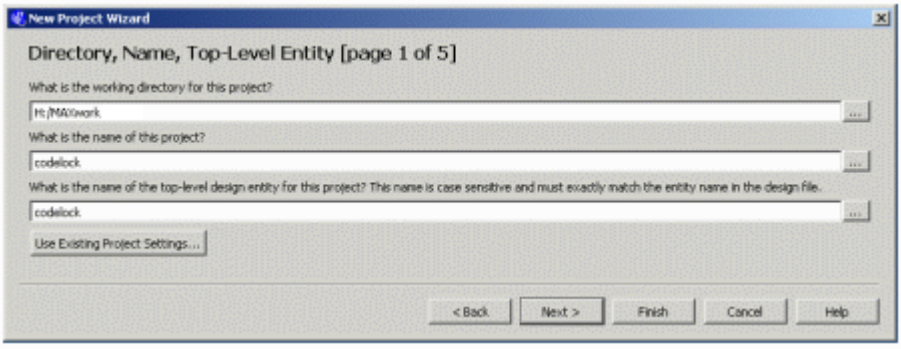


Project Name and Directory

In school, the entire project must be on your `H:\`, eg. `H:\MAXwork` (at home on `C:\`, eg. `C:\MAXwork`)

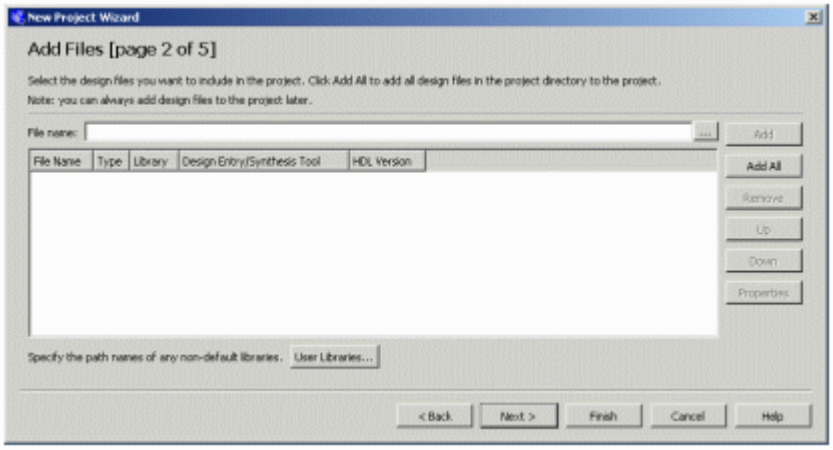
Name: `codelock`
Top-Level Entity: `codelock`
(Note! the name `codelock` must "match" the name you later on specify as entity in your VHDL-file)

Proceed with **Next**.



Add files

We have no files to add to the project, so we proceed with **Next**.

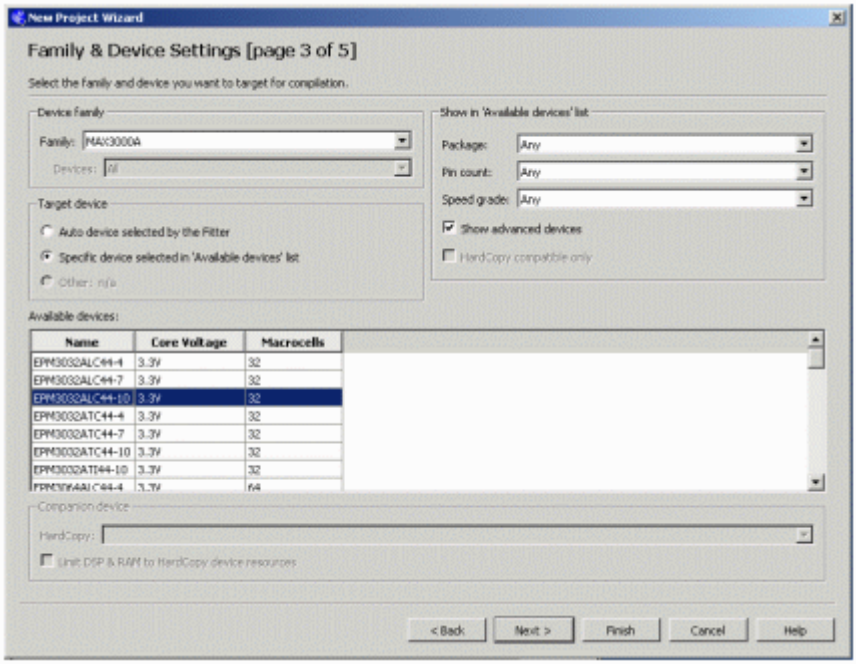


Family and Device Settings

Here we specify which chip we intend to use during the lab.

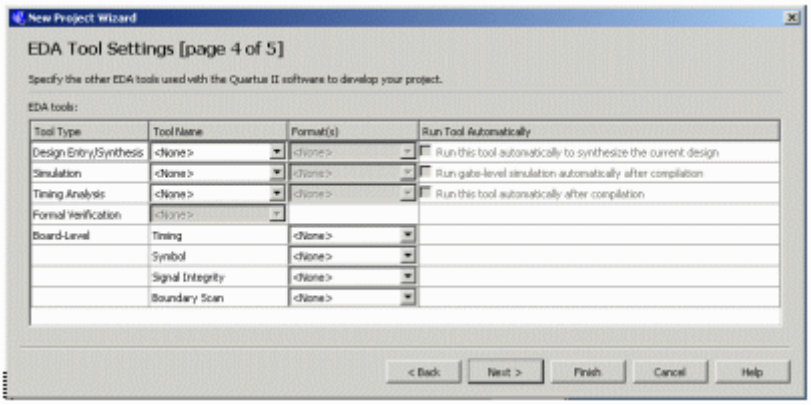
Family: `MAX3000A` **Available devices:** `EPM3032ALC44-10`

Proceed with **Next**.



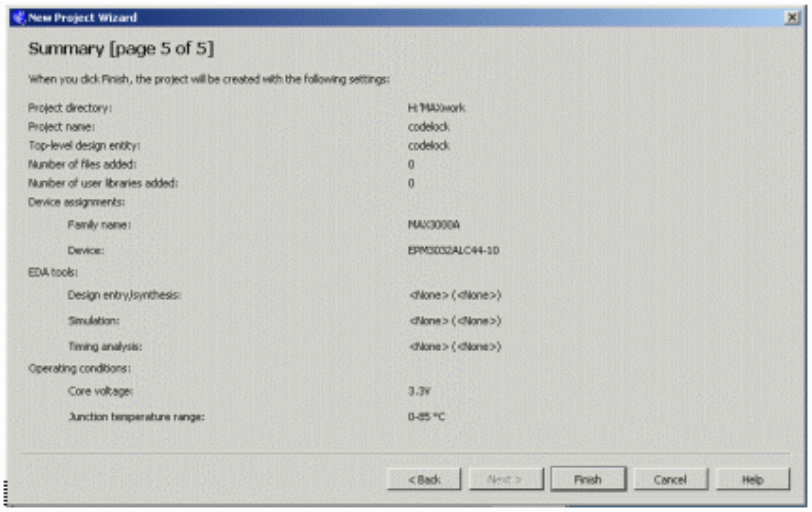
EDA tools setting

Here you can create context with software tools from other vendors. We will simulate with the ModelSim program but that we need not enter.
Proceed with **Next**.



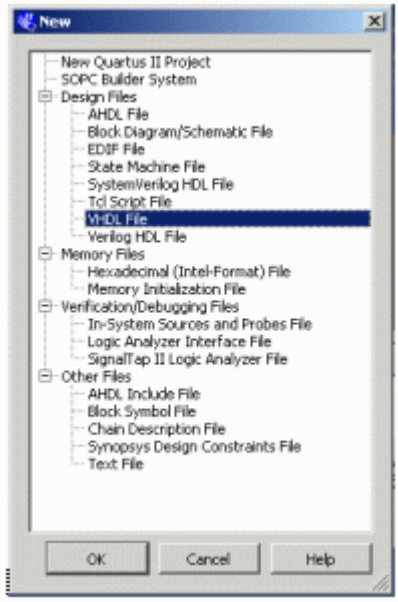
Summary.

Here you can see a summary of your selections, exit the "Wizard" with **Finish**.



The project has been created

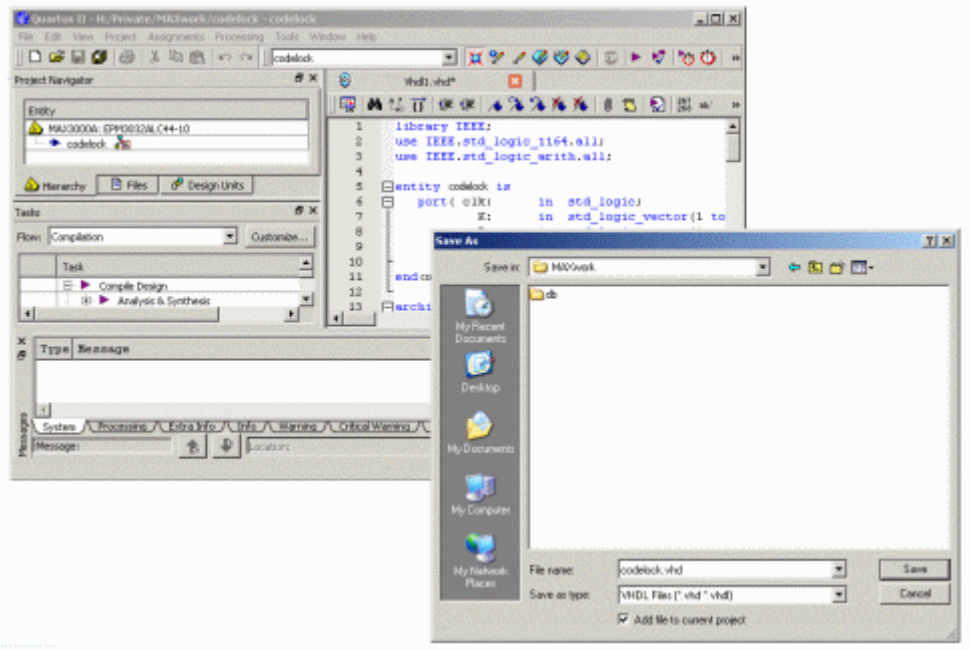
VHDL-code



Create a blank file for VHDL-code. **File**, **New**, **VHDL File**.

Copy the Template **`lockmall.vhd`** and paste it in Quartus text editor.

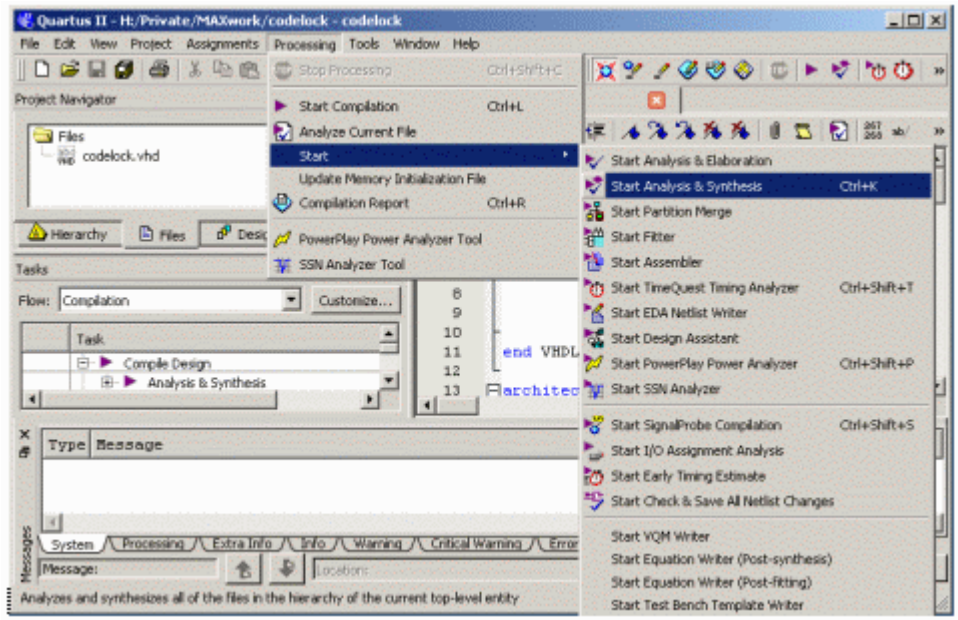
```
library IEEE;

use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity codelock is
   port( clk:      in  std_logic;
         K:      in  std_logic_vector(1 to 3);
         R:      in  std_logic_vector(1 to 4);
         q:      out std_logic_vector(4 downto 0);
         UNLOCK: out std_logic );
end codelock;

architecture behavior of codelock is
subtype state_type is integer range 0 to 31;
signal state, nextstate: state_type;

begin
nextstate_decoder: -- next state decoding part
process(state, K, R)
 begin
   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;
      when 2 to 30 => nextstate <= state + 1;
      when 31 => nextstate <= 0;
   end case;
end process;

debug_output:  -- display the state
q <= conv_std_logic_vector(state,5);

output_decoder: -- output decoder part
process(state)
begin
  case state is
     when 0 to 1  => UNLOCK <= '0';
     when 2 to 31 => UNLOCK <= '1';
  end case;
end process;

state_register: -- the state register part (the flipflops)
process(clk)
begin
  if rising_edge(clk) then
     state <= nextstate;
  end if;
end process;
end behavior;
```



Note that entity in the VHDL-file must "match" the project Top Level Entity!

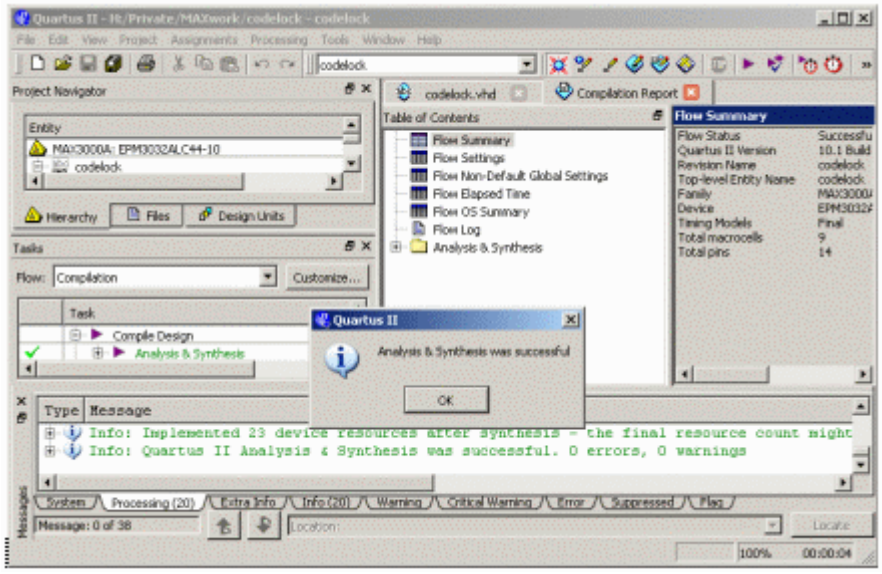
Save the file: **File**, **Save As** and as VHDL-file. The name could be `codelock.vhd` (or another).
Add File to current project shall be checked!

Analysis and Synthesis

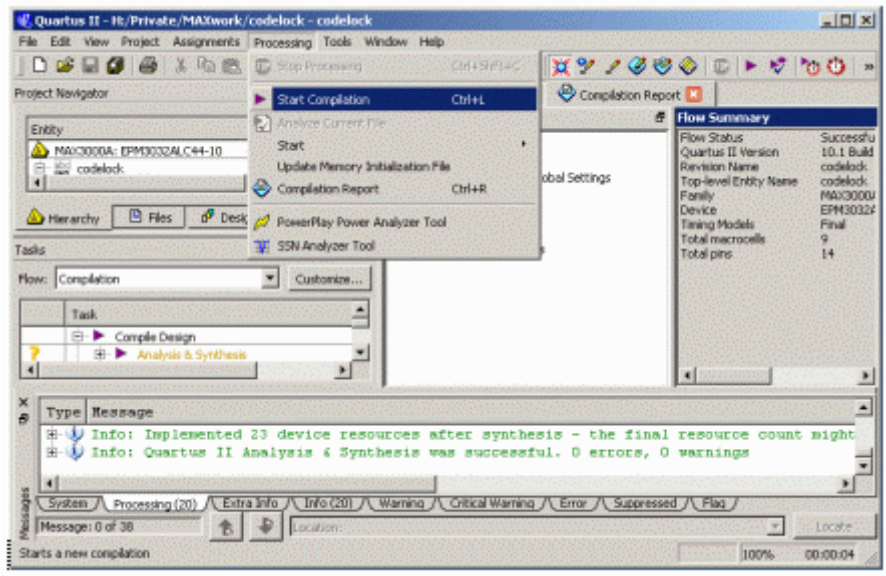


When you have new code, it is unnecessary to run the entire tool chain - chances are that there are errors along the way ...

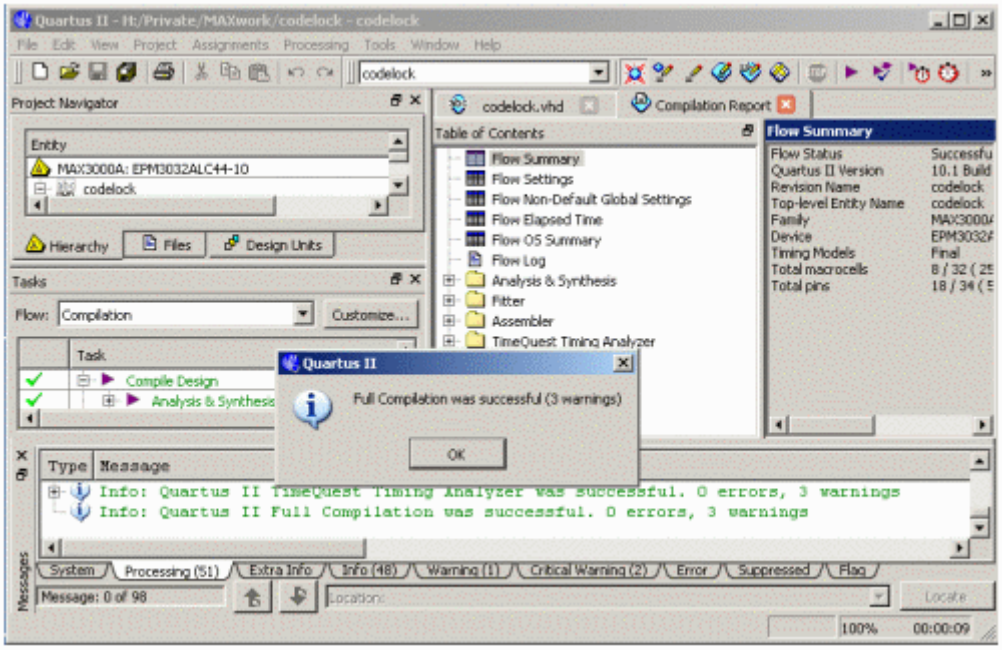
From the start you only do Analysis & Synthesis.



Start Compilation



Start Compilation runs the full tool chain.



The 3 warnings (moore with other program versions) are about "software tools" that are missing in our program version but we don’t need them.

William Sandqvist `william@kth.se`