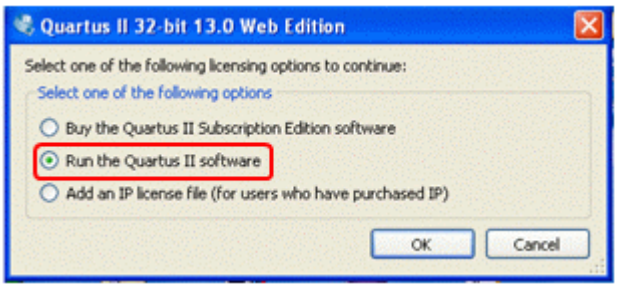
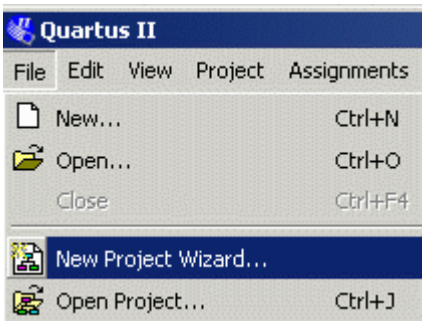
VHDL-program med Quartus

Välj rätt programversion - i skolan finns flera olika installerade under startmenyn!

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

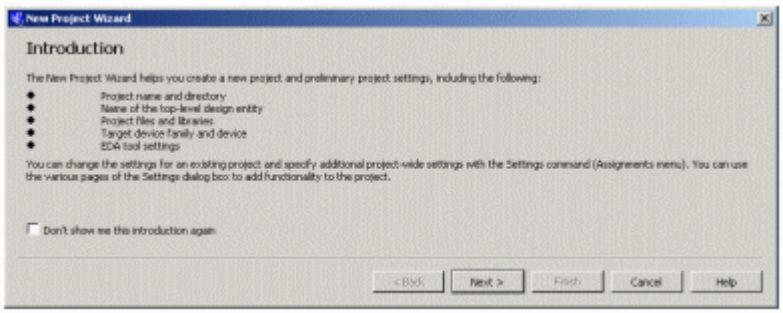


Starta Quartus. Du behöver ingen licens och inte heller köpa något.
Blir Du inte direkt erbjuden att starta **New Project Wizard**, så kan Du även välja det alternativet från **File**-menyn.



Introduction

Klicka på **Next**.

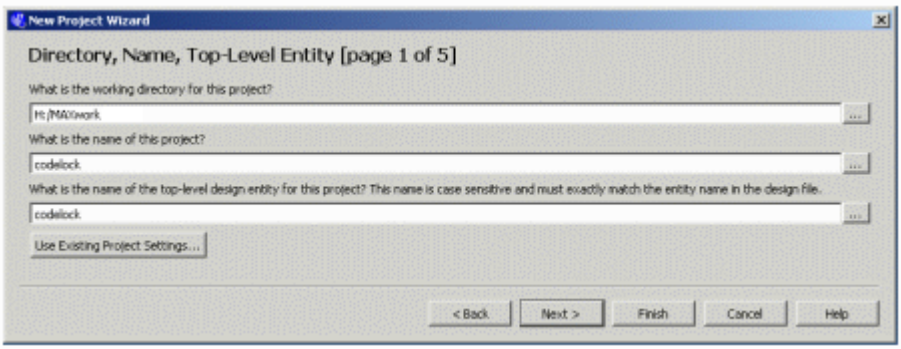


Project Name and Directory

I skolan måste hela projektet ligga i en mapp på ditt `H:\`, tex. `H:\MAXwork` (hemma på ditt `C:\`, tex. `C:\MAXwork`)

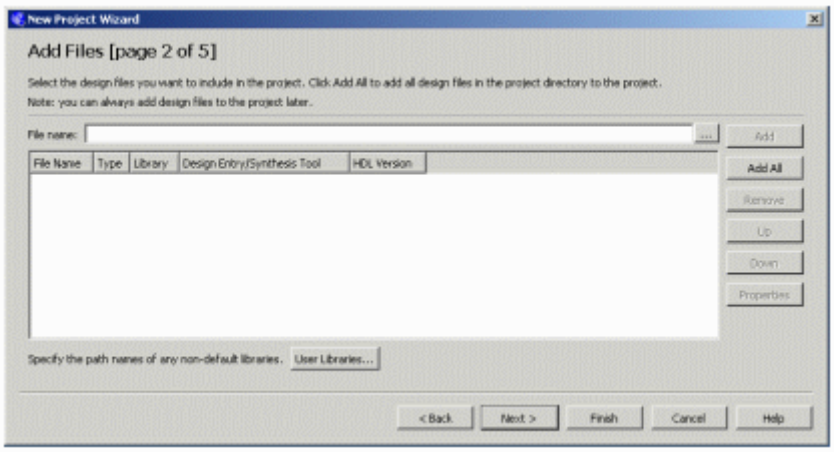
Name: `codelock`
Top-Level Entity: `codelock`
(OBS namnet `codelock` måste "matcha" det namn Du senare anger som entity i din VHDL-fil)

Gå vidare med **Next**.



Add files

Vi har inga filer att lägga till projektet, så vi går vidare med **Next**.

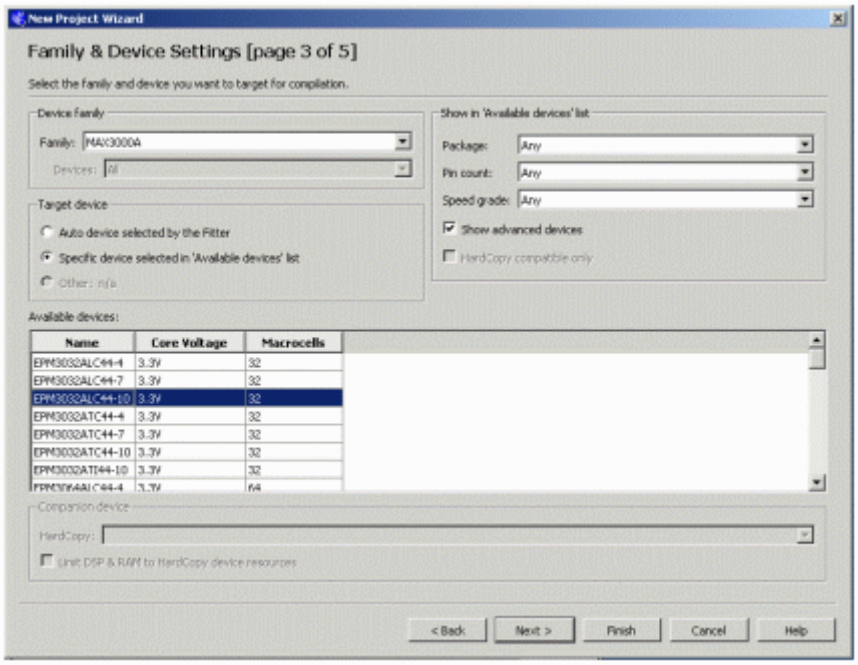


Family and Device Settings

Här anger vi vilket chip vi tänker använda under laborationen.

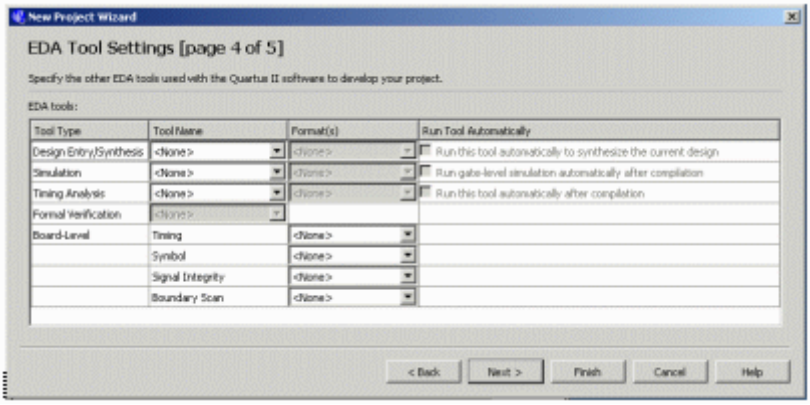
Family: `MAX3000A` **Available devices:** `EPM3032ALC44-10`

Gå vidare med **Next**.



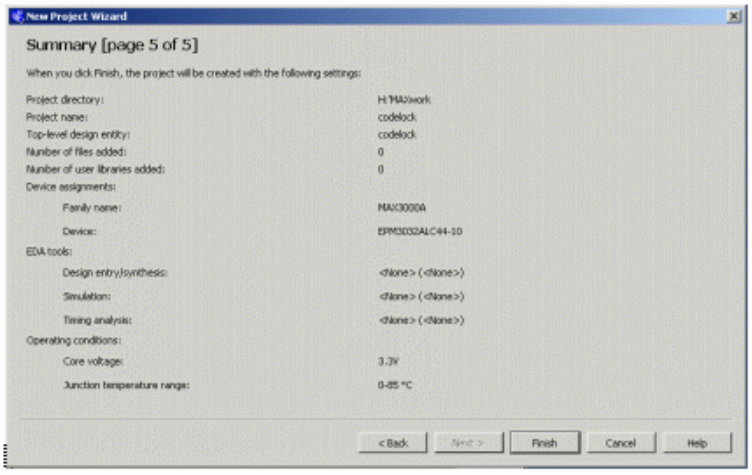
EDA tools setting

Här kan man skapa sammanhang med programverktyg från andra leverantörer. Vi kommer att simulera med programmet ModelSim men det behöver vi inte ange.
Gå vidare med **Next**.



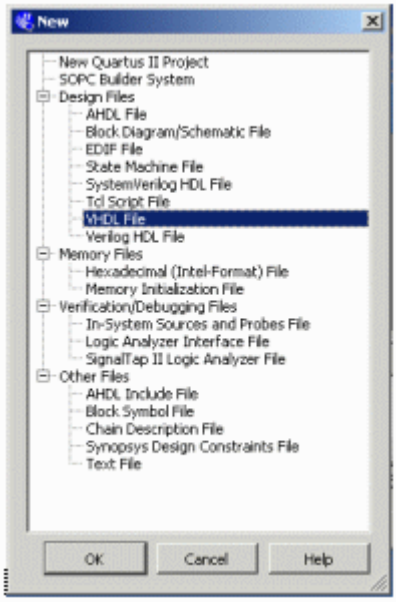
Summary, sammanfattning.

Här ser Du en sammanfattning av dina val, avsluta "Wizard" med **Finish**.



Projektet har skapats

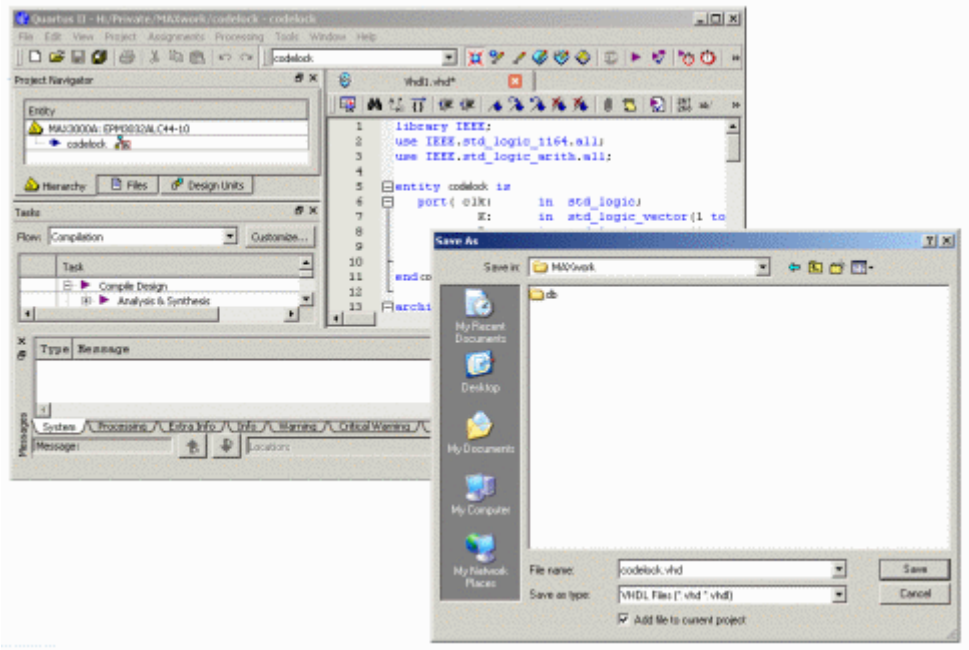
VHDL-koden



Skapa en blank fil för VHDL-koden. **File**, **New**, **VHDL File**.

Kopiera Mall-programmet **`lockmall.vhd`** och klistra in det i Quartus texteditor.

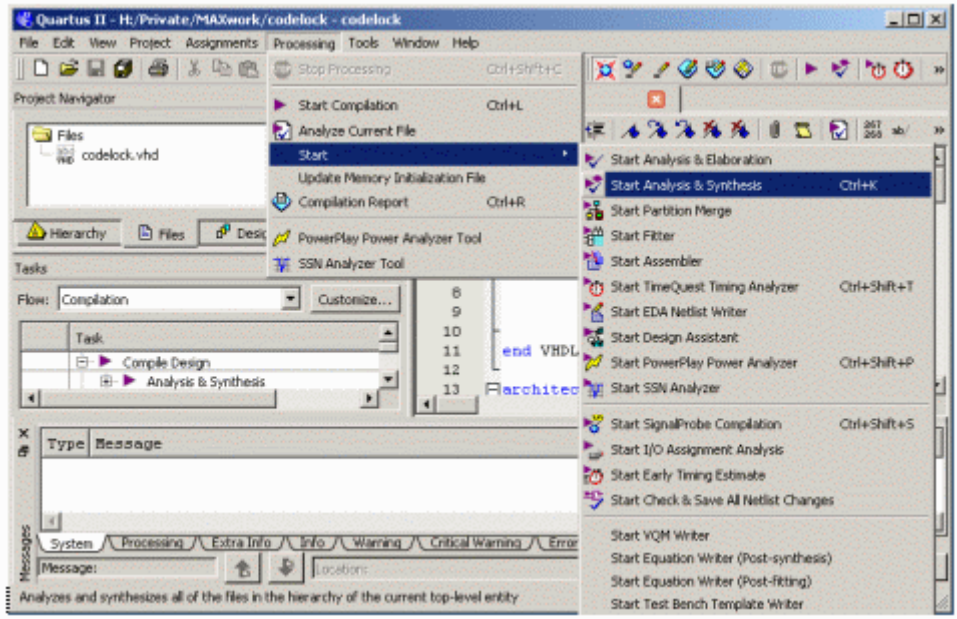
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

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;
```



Observera att entity i VHDL-filen ska matcha projektets Top Level Entity!

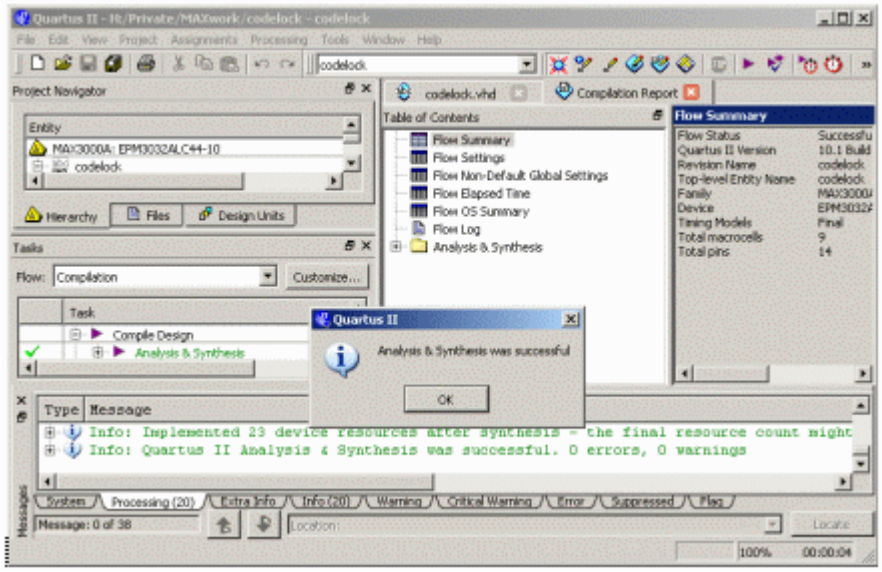
Spara filen med: **File**, **Save As** och som VHDL-fil. Namnet kan vara `codelock.vhd` (eller annat).
Add File to current project skall vara förbockat!

Analysis and Synthesis

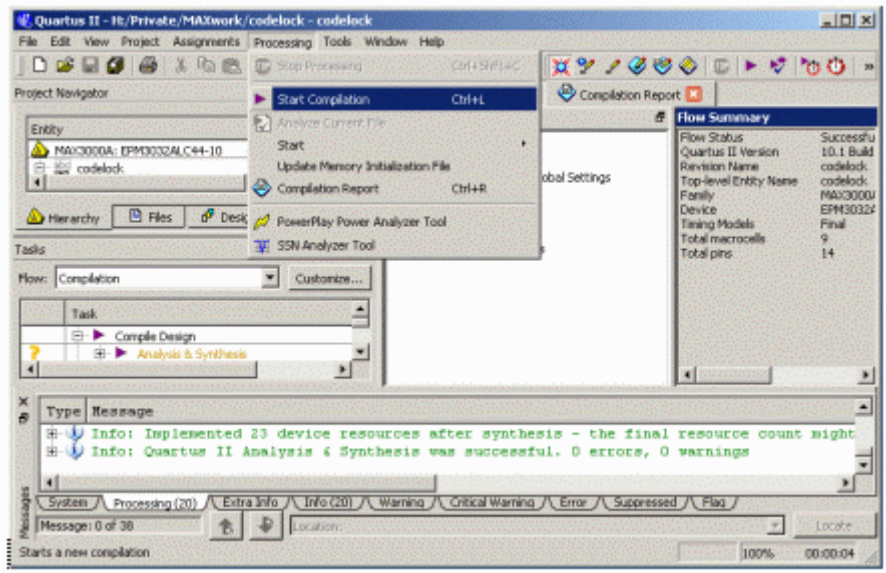


När man har nyskriven kod är det onödigt att köra hela verktygskedjan, risken är stor att det kan finnas felaktigheter längs vägen.

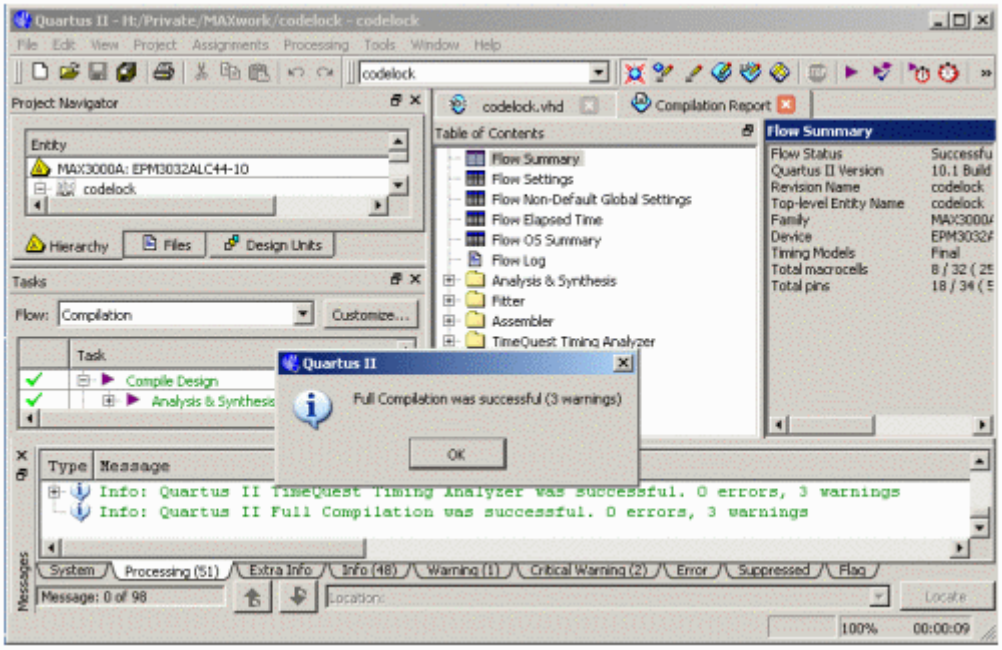
Från början kör man bara Analysis & Synthesis.



Start Compilation



Start Compilation kör hela verktygskedjan.



Varningarna handlar om "verktyg" som saknas i vår programversion men som vi inte behöver.
(Antalet varningar kan variera lite med programversion, detta är ingen anledning till oro).

© William Sandqvist `william@kth.se`