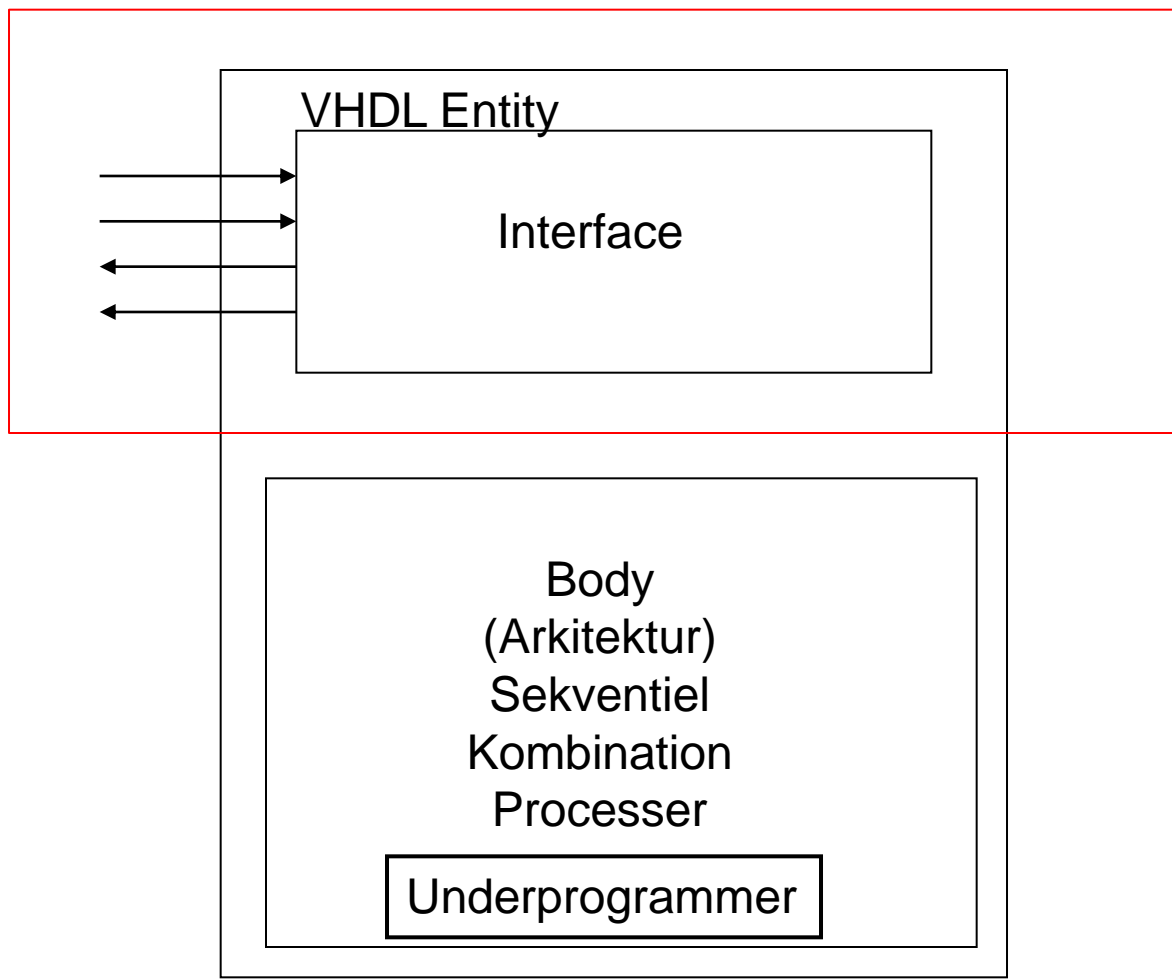


VHDL programmering H2

VHDL

- (**V**ery high speed Integrated circuits)
Hardware **D**escription **L**anguage
- IEEE standard 1076-1993 Den benytter vi!!
- Hvornår blev den frigivet som standard første gang??
- Ca. 1980!!

VHDL struktur



VHDL struktur

- ENTITY (enhed) ”Skal være det samme som projekt-navnet!!” Case sensitiv!!!
- VHDL kode skal starte biblioteker og derefter ”ENTITY” efterfulgt af (Projektnavnet)

VHDL kode

```

-- Quartus II VHDL Template
-- Configurable gate architecture
library ieee;use
ieee.std_logic_1164.all;
entity VHDL1 is
  port
  (
    i1 : in std_logic;
    i2 : in std_logic;
    o1 : out std_logic );
end VHDL1;
  
```

← TEXT

← Biblioteker

← Start på deklaration "ENTITY"

← Definition på porte

← ENTITY slut!!

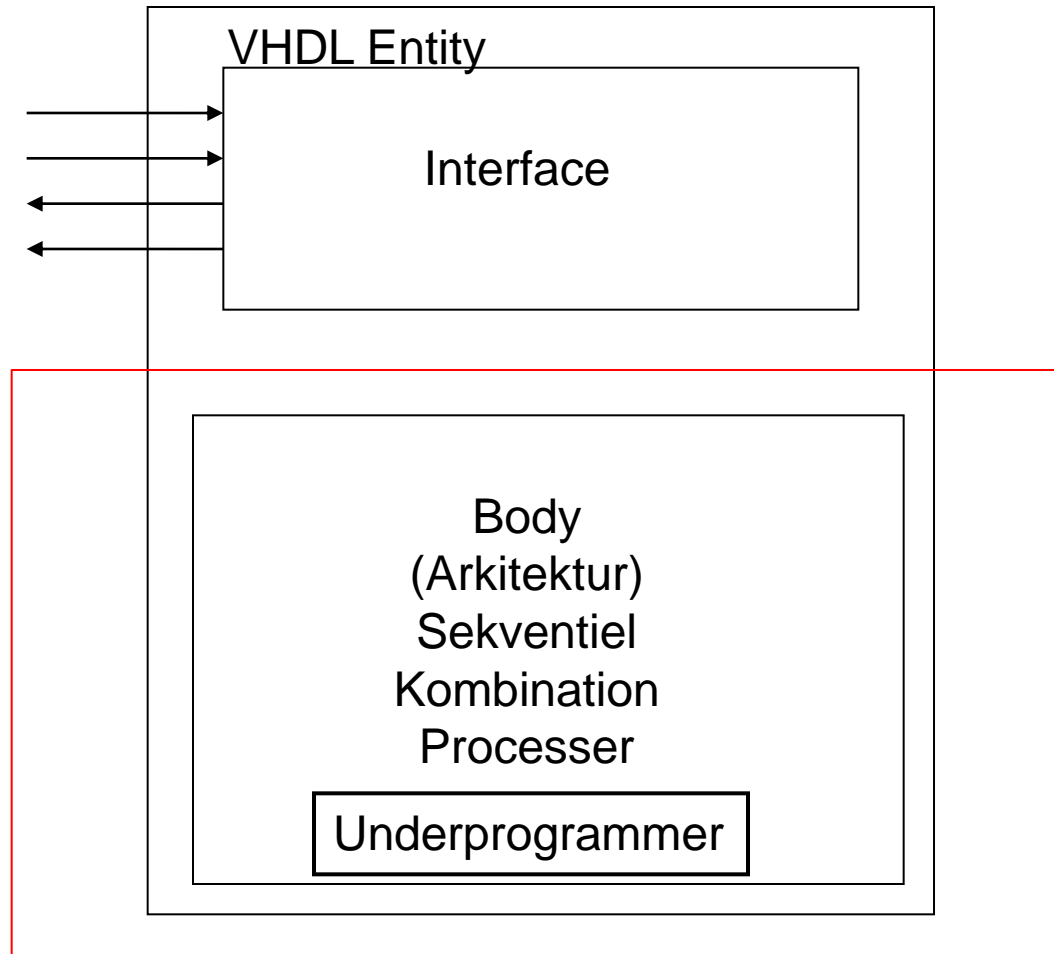
Porte

- **MODE**
 - IN: input signal
 - OUT: output signal
 - BUFFER: output der kan læses i VHDL koden
 - INOUT: både og

Type

- TYPE
 - *bit*: 0 og 1 værdier
 - *Bit_vector*: Kan eks. Være Bit_vector(0 to 7)
 - *Std_logic*, *std_ulogic*, *std_logic_vector*, *std_logic_ulogic*: Kan indtage 9 forskellige værdier!
Brug altid *std_logic* eller *std_logic_vector*!!!!
 - *Boolean*: false eller true
 - Integer: hele tal
 - *Real*: med komma
 - *Character*: Karakterer ikke tal værdier
 - *Time*: tid

VHDL struktur



Arkitektur

```
architecture and_gate of VHDL1 is
begin
  o1 <= i1 AND i2;
end and_gate;
```

Diagram illustrating the components of the VHDL architecture definition:

- Entity**: Points to the entity name `VHDL1`.
- Navn**: Points to the architecture name `and_gate`.
- Logisk udtryk!**: Points to the logic expression `o1 <= i1 AND i2;`.

Arkitektur

```
architecture or_gate of VHDL1 is  
begin  
    o1 <= i1 OR i2;  
end or_gate;
```

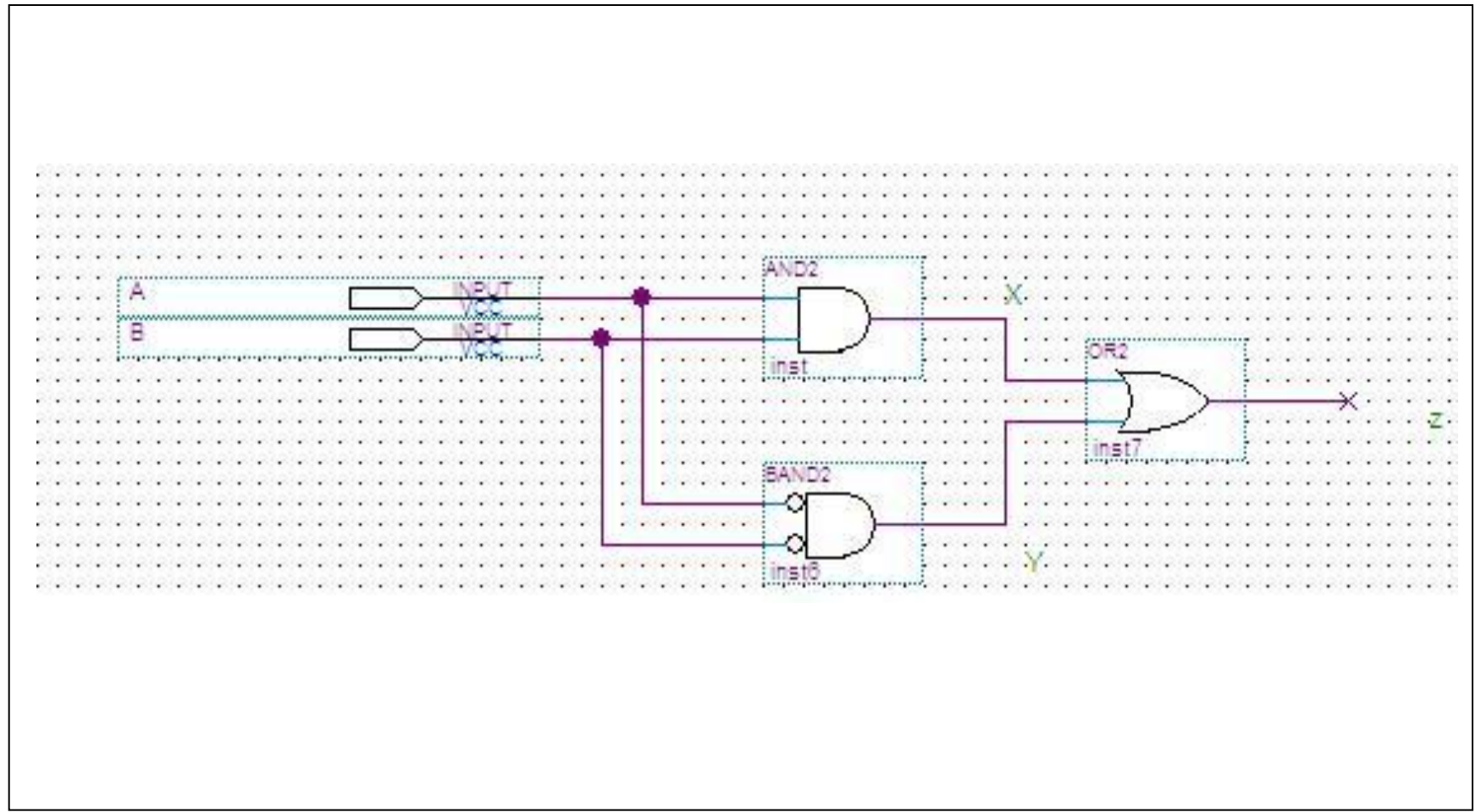
Arkitektur

```
architecture xor_gate of VHDL1 is  
begin  
    o1 <= i1 XOR i2;  
end xor_gate;
```

Arkitektur

```
Entity VHDL1 is
  port: (A,B:in std-logic;
         Z: out std_logic);
End;
architecture Xnor of VHDL1 is
  signal X,Y: std_logic; ← Interne signaler!!!
begin
  X <= A AND B;
  Y <= (not A) and (not B);
  Z <= X or Y
end Xnor;
```

Gates

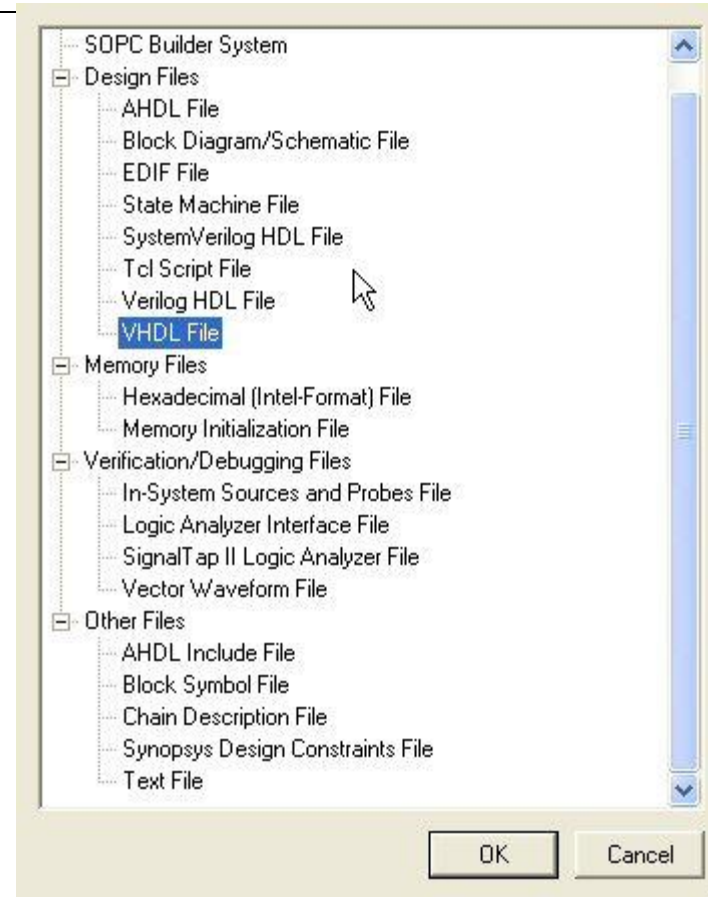


VHDL indtastning

- Start med at oprette en schmatic Entity!!!

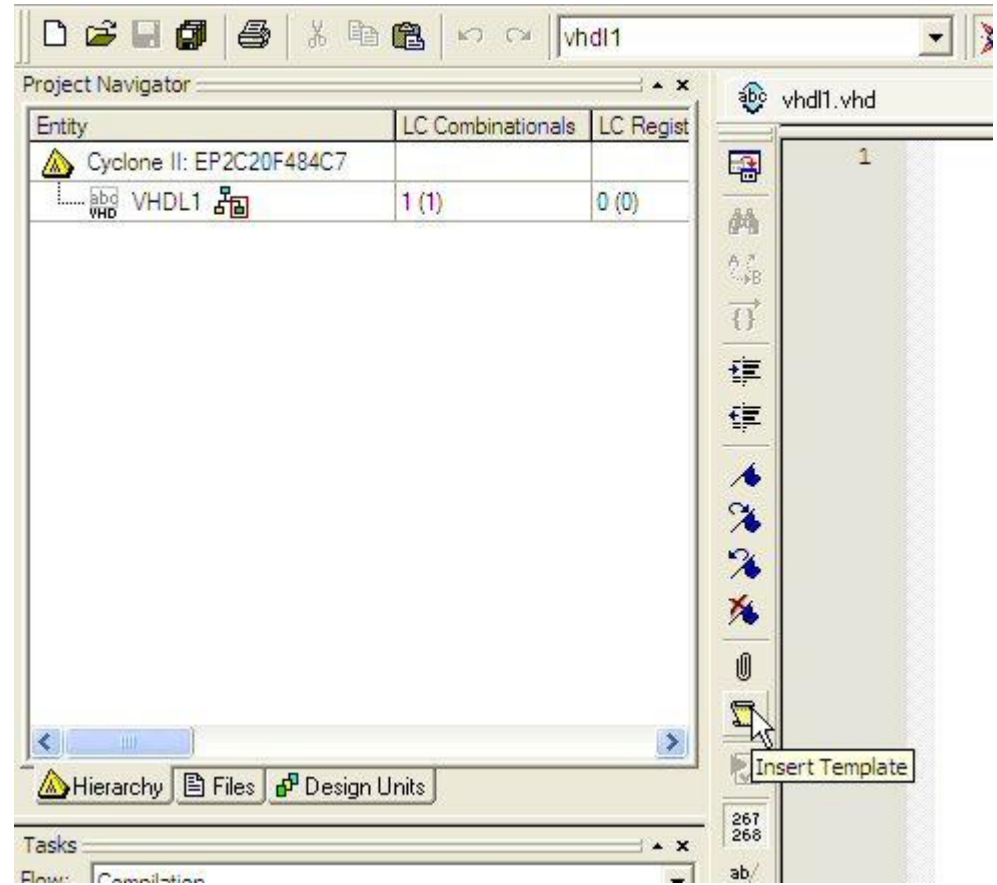
VHDL indtastning

- Tryk : "NEW"
- "VHDL File"



VHDL indtastning

- "Template"



VHDL indtastning

The screenshot shows the 'Insert Template' dialog in the Quartus II IDE. The left pane shows a tree view of language templates, with 'Configurable Gate Architecture' selected under 'VHDL' > 'Configurations' > 'Configuration Declarations'. The right pane shows the preview of the selected template, which is a VHDL code snippet for a configurable gate architecture.

```

-- Quartus II VHDL Template
-- Configurable gate architecture

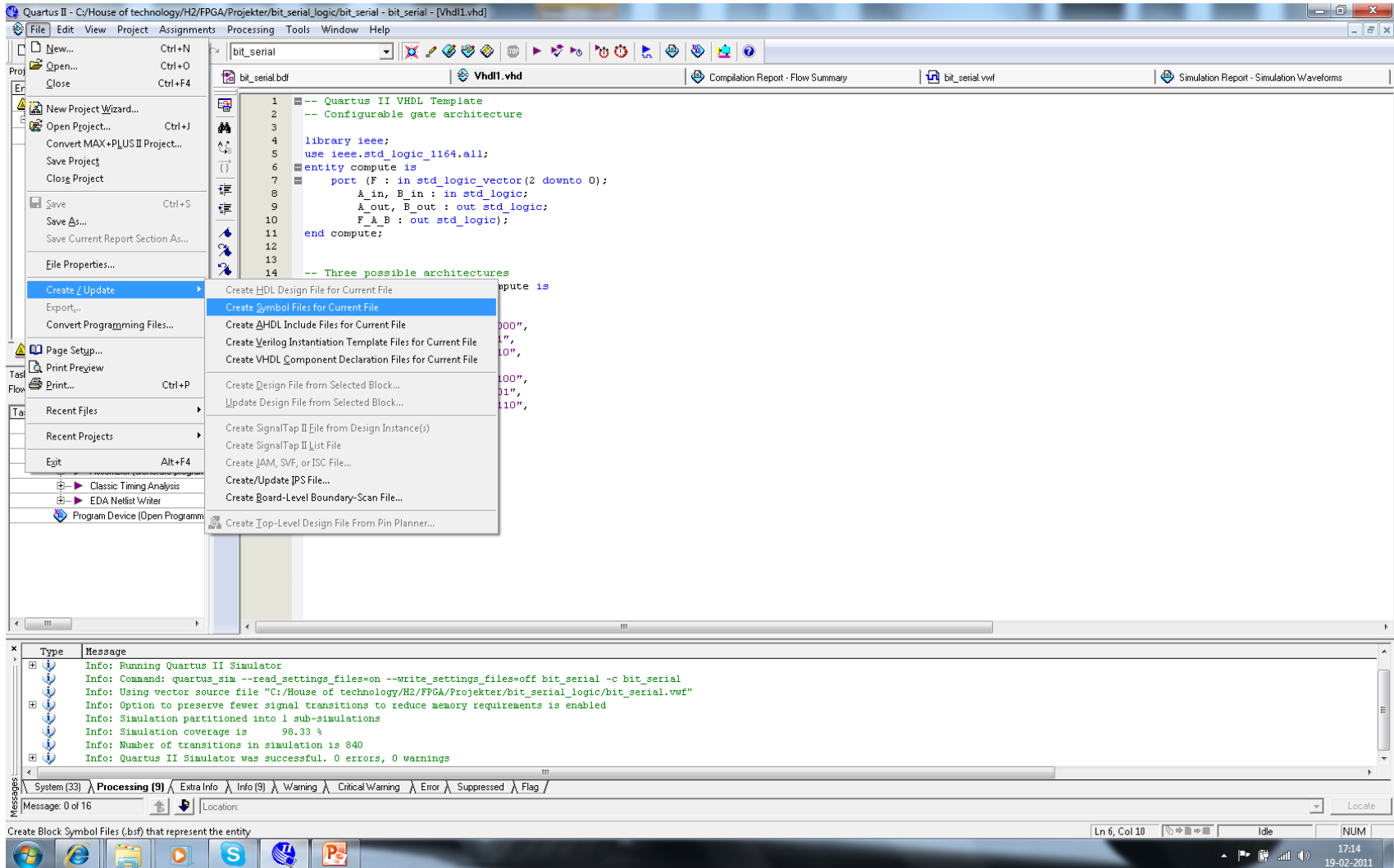
library ieee;
use ieee.std_logic_1164.all;
entity configurable_gate is
    port
    (
        i1 : in std_logic;
        i2 : in std_logic;
        o1 : out std_logic
    );
end configurable_gate;

-- Three possible architectures
architecture and_gate of configurable_gate is
begin
    o1 <= i1 AND i2;
end and_gate;

architecture or_gate of configurable_gate is
begin
    o1 <= i1 OR i2;
end or_gate;

```

VHDL ind i Schematic



The screenshot shows the Quartus II IDE with a VHDL file open. The code defines an entity named 'compute' with two input ports (A_in, B_in) and two output ports (A_out, B_out). The logic is a simple XOR operation: A_out is the XOR of A_in and B_in, and B_out is the XOR of B_in and A_in.

```

1  -- Quartus II VHDL Template
2  -- Configurable gate architecture
3
4  library ieee;
5  use ieee.std_logic_1164.all;
6  entity compute is
7      port ( F : in std_logic_vector(2 downto 0);
8            A_in, B_in : in std_logic;
9            A_out, B_out : out std_logic;
10           F_A_B : out std_logic);
11 end compute;
12
13
14 -- Three possible architectures
15
16 architecture compute is
17     signal A_xor_B : std_logic;
18     signal B_xor_A : std_logic;
19     signal F_A_B : std_logic;
20
21     A_xor_B <= A_in xor B_in;
22     B_xor_A <= B_in xor A_in;
23     F_A_B <= F(0) xor F(1) xor F(2);
24
25     A_out <= A_xor_B;
26     B_out <= B_xor_A;
27
28 end architecture compute;
  
```

The Message window at the bottom shows the following simulation results:

```

Info: Running Quartus II Simulator
Info: Command: quartus_sim --read_settings_files=on --write_settings_files=off bit_serial -c bit_serial
Info: Using vector source file "C:/House of technology/H2/FPGA/Projekter/bit_serial_logic/bit_serial.vwf"
Info: Option to preserve fewer signal transitions to reduce memory requirements is enabled
Info: Simulation partitioned into 1 sub-simulations
Info: Simulation coverage is 98.33 %
Info: Number of transitions in simulation is 840
Info: Quartus II Simulator was successful. 0 errors, 0 warnings
  
```

Opgave

- Opgave 1 laves med VHDL!
- Lav 7 seg. med VHDL, et element for hvert segment.
- Lave 7 segment med Logic Friday!!
 - Link til www adr på Mars!

Compute kode

F afgøre hvilken operator
Der skal benyttes

entity compute is

Port (F : in std_logic_vector(2 downto 0);

Data in som skal manipuleres!

A_In, B_In : in std_logic;

A_Out, B_Out : out std_logic;

F_A_B : out std_logic);

end entity compute;

Resultat af A_in og B_in

Compute kode

architecture Behavioral of compute is
begin
with F select

```

F_A_B <= A_In and B_In when "000",
      A_In or B_In when "001",
      A_In xor B_In when "010",
      '1' when "011",
      A_In nand B_In when "100",
      A_In nor B_In when "101",
      A_In xnor B_In when "110",
      '0' when others; -- must be included

A_Out <= A_In;
B_Out <= B_In;

```

end architecture Behavioral;

Når F = 000 så bliver A_in og B_in
And'et sammen

Opgave

- VHDL "Compute" kode analyseres?
- Hvordan virker den?

RS FF

- -- Quartus II VHDL Template
- -- Configurable gate architecture library ieee; use ieee.std_logic_1164.all;
- entity VHDL_test is
- port (s,r : in std_logic;
- q,nq : buffer std_logic);
- end VHDL_test;
- -- Three possible
- architectures architecture dataflow of VHDL_test is
- begin
- q<=r nor nq;
- nq<=s nor q;
- end dataflow;

Tæller op og ned

```
library IEEE;  
use IEEE.std_logic_1164.all;  
use IEEE.STD_LOGIC_ARITH.ALL;  
use IEEE.STD_LOGIC_UNSIGNED.ALL;  
  
entity up_down_counter is  
port (clk, enable, up_down : in std_logic;  
      synch_reset: in std_logic;  
      Q: out std_logic_vector(7 downto 0));  
end entity up_down_counter;
```


Architecture counter

```

architecture counter_behavior of up_down_counter is
signal count:std_logic_vector(7 downto 0);
begin
process(clk, synch_reset) -- sensitivity list
    begin
        if(synch_reset='1') then count <= "00000000";
            elsif (clk'event and clk='1') then
                if (enable='1') then
                    if (up_down='1') then count<=count+"0000001";
                        else count<=count-"0000001";
                    end if;
                end if;
            end if;
        end if;
        Q<=count;
    end process;
end architecture counter_behavior;

```

Opgave

- Analysere tælleren, hvordan virker den?