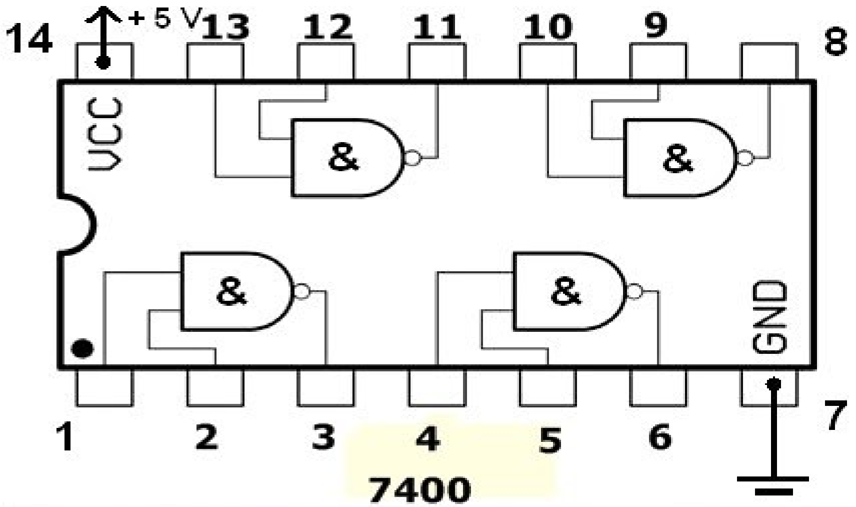
Lesson 5

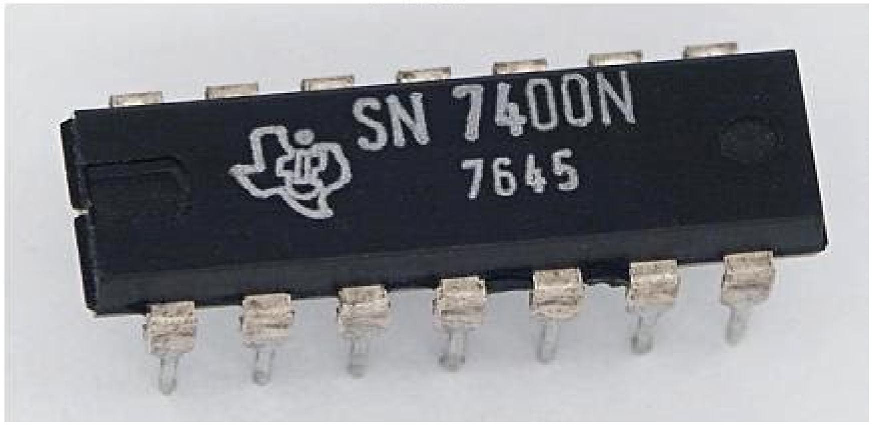
Entity, Architecture and Signals

Mohd Saufy Rohmad
Technical Trainer and Consultant

Think In Hardware

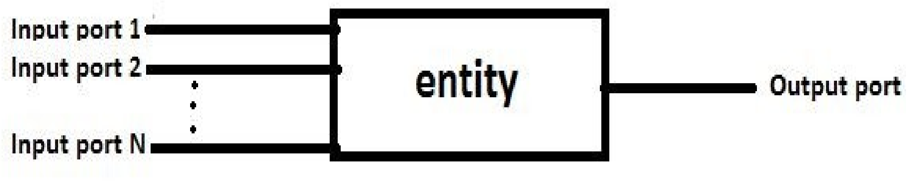
- Think about an IC, for example, a 7400N IC.
- It includes four NAND gates inside, while its provided in a black package.
- Lets say we want to write the VHDL code of the 7400N IC





Think In Hardware

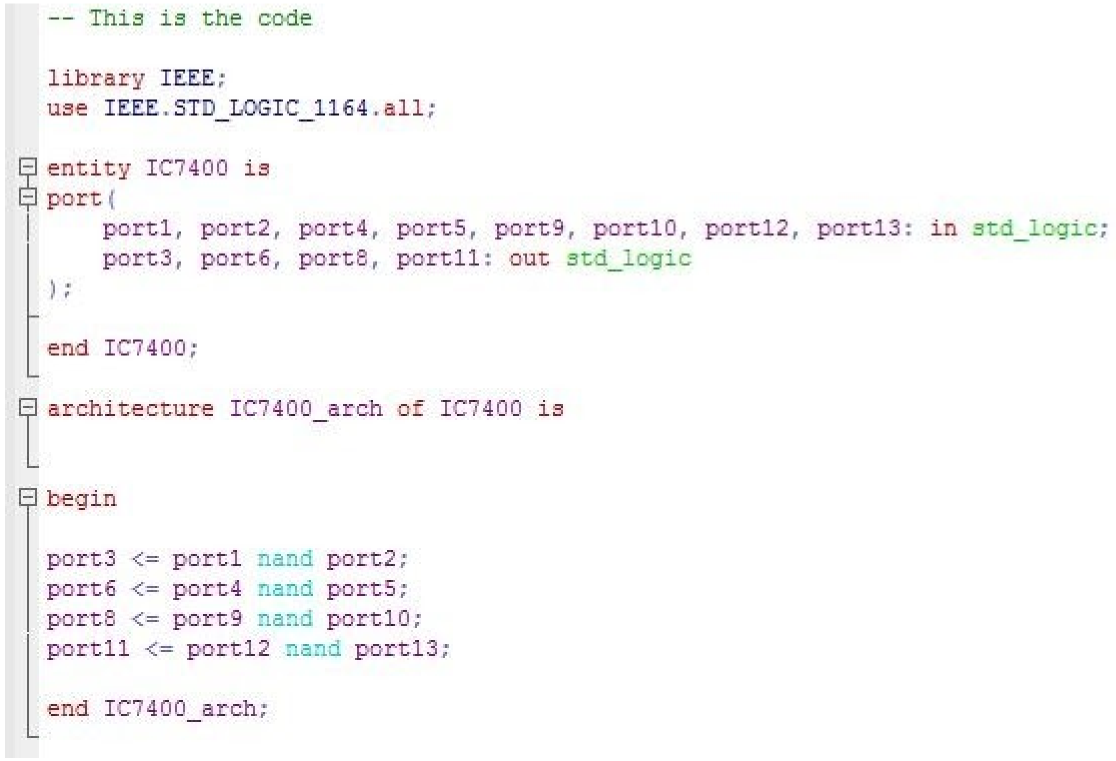
- It is easier to think that you are designing a black box when you are writing any VHDL code.
- This black box has inputs and outputs and something in between.
- This black boxes are called entity.



Think In Hardware

- For 7400N IC's VHDL code, we will call this black box aka entity as IC7400. The entity IC7400 should have 8 input ports and 4 output ports corresponding to the input and outputs of the original 7400N IC.
- This is how the VHDL code of IC7400 might look like

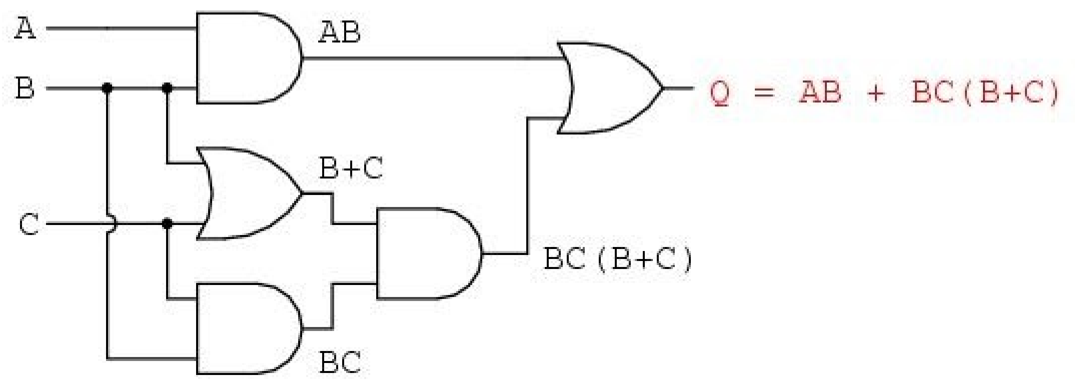
Think In Hardware



```
-- This is the code

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity IC7400 is
port(
    port1, port2, port4, port5, port9, port10, port12, port13: in std_logic;
    port3, port6, port8, port11: out std_logic
);

end IC7400;

architecture IC7400_arch of IC7400 is


begin

port3 <= port1 nand port2;
port6 <= port4 nand port5;
port8 <= port9 nand port10;
port11 <= port12 nand port13;

end IC7400_arch;
```

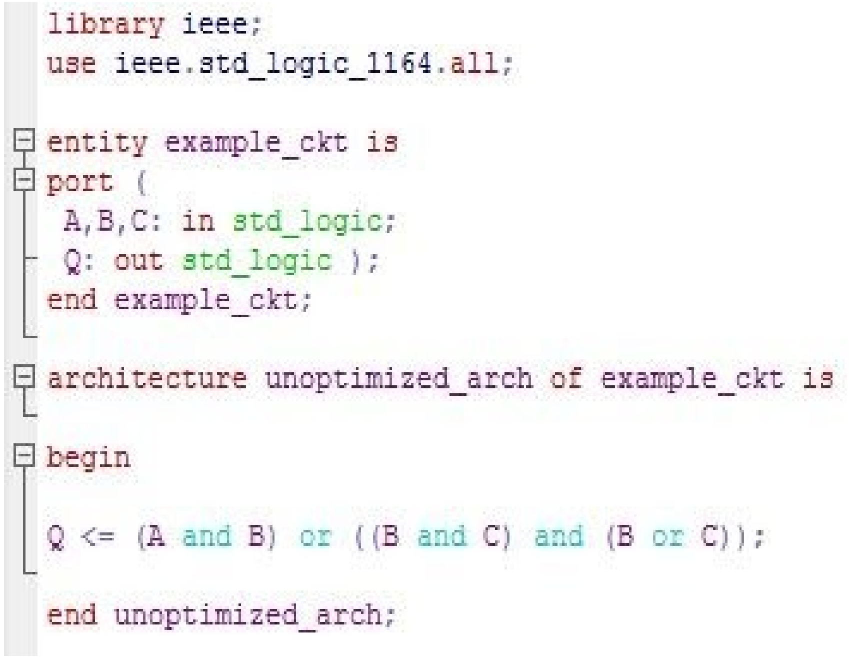
Architecture

- Every entity can have different architecture.
- Think about it in this way that any digital circuit can have different architectures, i.e. different logic, different number of gates etc.
- For example, The output of the following circuit have a boolean expression of AB + BC(B+C) which can be simplified as B(A+C).
- The VHDL code of the unoptimized architecture, i.e. AB + BC(B+C) can be written as



VHDL for 7400N

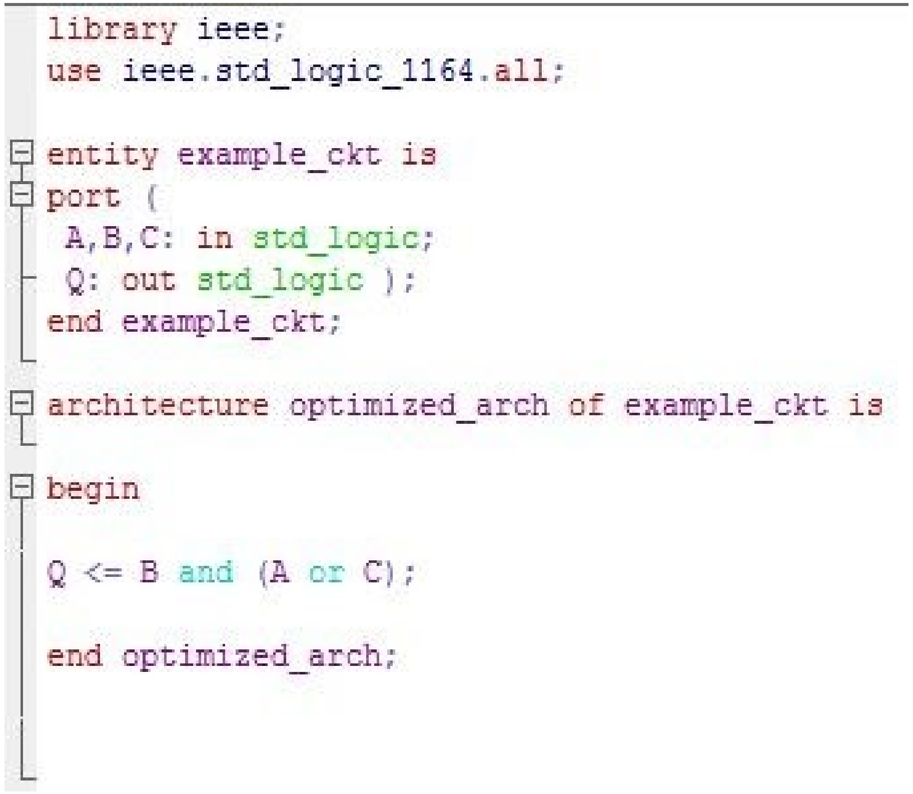
- The VHDL code of the unoptimized architecture, i.e. AB + BC(B+C) can be written as,



```
library ieee;
use ieee.std_logic_1164.all;

entity example_ckt is
port (
 A,B,C: in std_logic;
 Q: out std_logic );
end example_ckt;

architecture unoptimized_arch of example_ckt is

begin

Q <= (A and B) or ((B and C) and (B or C));

end unoptimized_arch;
```

VHDL for 7400N

- And the optimized architecture can be written as,



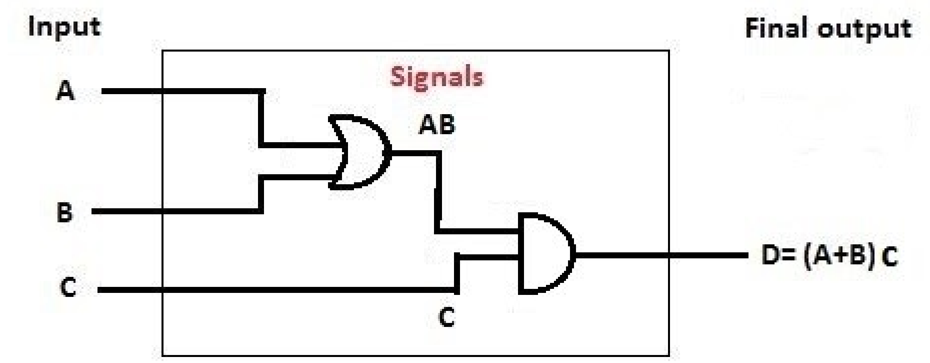
```
library ieee;
use ieee.std_logic_1164.all;

entity example_ckt is
port (
 A,B,C: in std_logic;
 Q: out std_logic );
end example_ckt;

architecture optimized_arch of example_ckt is

begin

Q <= B and (A or C);

end optimized_arch;
```

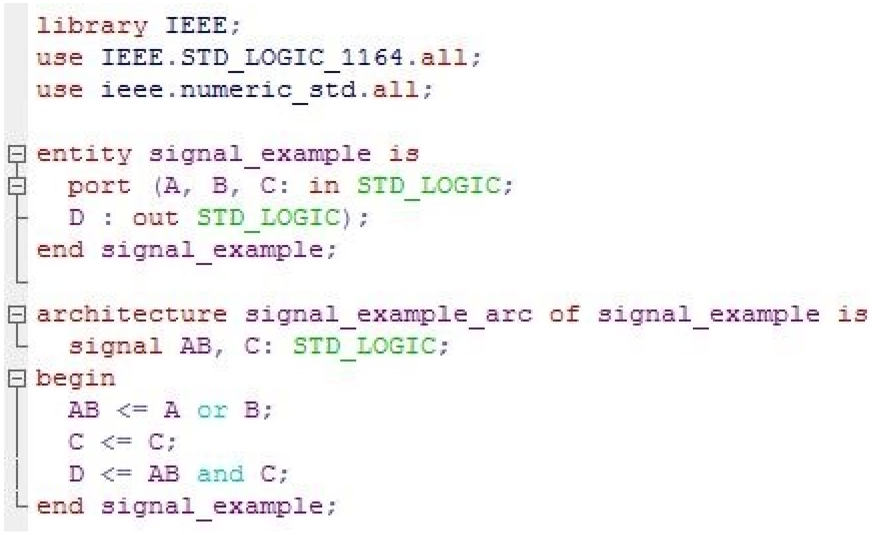
Signal

- So far we have only seen how to write input and output ports.
- We only learned to think the entity as a blackbox where the input and output ports are coming out of the box.
- But what happens when we have to write some ports or wires inside the blackbox.
- For example, consider the following circuit.
- The final output is (A or B) and C.
- It is possible to write the whole thing in a single sentence.
- But we might have a complex design where its not possible.
- We have to breakdown the entire operation then.
- In this case, lets say, we want to break down (A or B) and C into (A or B) first and then use the output to AND with C.
- This is where we introduce the concept of **signals**.

Signal

- The VHDL code of this circuit is written here.
- The signal here is AB that works as an intermediate signal inside the blackbox.
- We have to declare the signal after architecture and before begin.





```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;

entity signal_example is
  port (A, B, C: in STD_LOGIC;
  D : out STD_LOGIC);
end signal_example;

architecture signal_example_arc of signal_example is
  signal AB, C: STD_LOGIC;
begin
  AB <= A or B;
  C <= C;
  D <= AB and C;
end signal_example;
```