Lesson 8

Components

Mohd Saufy Rohmad
Technical Trainer and Consultant

Components

- Its nearly impossible to put every VHDL code inside one entity while designing a huge design.
- Its also not a clever idea as a modular design helps to debug and ease the development.
- So how can we divide a huge design in smaller parts and connect them together.
- Perhaps you have already guessed it?
- Yes, we will design smaller components of the bigger design as separate entities and connect them with a top entity

Components

- Lets assume it is difficult to design nand and and gate together in a single entity (which is certainly not true :p, but for the sake of understanding just assume its right).
- We can develop two separate entities first.
- One for the nand gate and another for the and gate.
- For example, an entity for the nand gate can be written as:

Components

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

entity comp1 is
port(
A,B: in std_logic;
C: out std_logic);
end comp1;

architecture comp1_arch of comp1 is

begin

C <= A nand B;

end comp1_arch;
```

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

entity comp2 is
port(
E,F: in std_logic;
G: out std_logic);
end comp2;

architecture comp2_arch of comp2 is

begin

G <= E and F;

end comp2_arch;
```

And another entity for the AND gate can be written as,

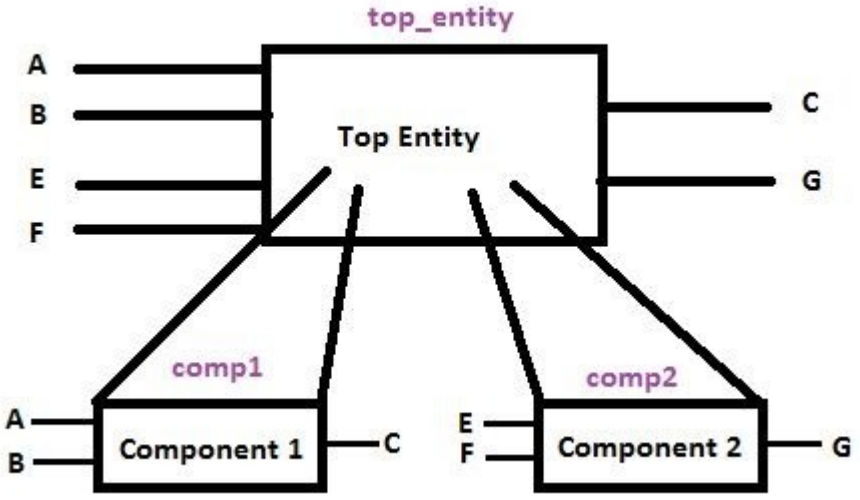
Components

- Now we want to put them together inside a top entity. The top entity would contain this two entities as components.
- Full code in lab

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

entity comp1 is
port(
A,B: in std_logic;
C: out std_logic);
end comp1;

architecture comp1_arch of comp1 is

begin

C <= A nand B;

end comp1_arch;


library ieee;
use ieee.std_logic_1164.all;

entity comp2 is
port(
E,F: in std_logic;
G: out std_logic);
end comp2;

architecture comp2_arch of comp2 is

begin

G <= E and F;

end comp2_arch;


library ieee;
use ieee.std_logic_1164.all;


entity top_entity is
port(
top_A,top_B,top_E,top_F: in std_logic;
top_C,top_G: out std_logic);
end top_entity;

top_C,top_G: out std_logic);
end top_entity;

architecture top_arch of top_entity is

component comp1 is
port(
A,B: in std_logic;
C: out std_logic);
end component;

component comp2 is
port(
E,F: in std_logic;
G: out std_logic);
end component;


begin

comp: comp1 port map(
A => top_A,
B => top_B,
C => top_C);

com2: comp2 port map(
E => top_E,
F => top_F,
G => top_G);

end top_arch;
```

Components

- It might look intimidating in the beginning, but don’t be scared.
- Its actually a pretty simple code.
- What we are doing here can be better understood by this diagram



Component port map:

- Now the biggest task is still left. We decided to use this smaller entities as components, but we have to connect the input and output ports of the smaller entities to some signal or ports of the top entity.
- You can actually visualize it in this way.

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


entity top_entity is
port(
top_A,top_B,top_E,top_F: in std_logic;
top_C,top_G: out std_logic);
end top_entity;

architecture top_arch of top_entity is

component comp1 is
port(
A,B: in std_logic;
C: out std_logic);
end component;

component comp2 is
port(
E,F: in std_logic;
G: out std_logic);
end component;


begin

comp: comp1 port map(
A => top_A,
B => top_B,
C => top_C);

com2: comp2 port map(
E => top_E,
F => top_F,
G => top_G);

end top_arch;
```

Components

- It's just like plugging ICs into a breadboard.
- For example, we can plug a 7400 IC (NAND) and 7408 IC (AND) gate in a breadboard in this similar way as shown in the figure above.
- We could also instantiate the components in this way, however the order of the instantiation is very important in that case.
- Otherwise the one part will be binded to another undesired part.