# Digital Design

# Generic

Generic variables are used to build generic modules.
Default values need to be specified when defining.

```
use IEEE. STD_LOGIC_ARITH. ALL;
use IEEE. STD_LOGIC_UNSIGNED. ALL;

entity counter is
 Generic (  data_width : integer := 8  );
 Port (
    enable_i : in STD_LOGIC; clk_i : in STD_LOGIC;
    rst_i : in STD_LOGIC;
    data_o : out STD_LOGIC_VECTOR ( data_width -1 downto  0)
 );
end counter ;
```

# Connecting modules

- We want to have more structured systems and reuse modules and VHDL code when possible
  - „using the code“ for a 4-bit counter to make a 5-bit counter
- Example: We can split a counter in 2 sub-modules
  - prescaler
    - input: clk_i, reset_i
    - output: enable_o
  - counter
    - input : clk_i, reset_i, enable_i
    - output : data_o

# Prescaler

```
entity prescaler is
  Generic (
          width : integer := 8;
          value  : integer :=255
         );
  Port (
          clk_i: in STD_LOGIC; reset_i: in STD_LOGIC;
          enable_o: out STD_LOGIC
         );
end prescaler ;
```

# Counter

```
entity counter is
Generic (
                width : integer := 8
            );
  Port (
        clk_i: in STD_LOGIC; reset_i: in STD_LOGIC;
        enable_i: in STD_LOGIC;
        data_o: in STD_LOGIC_VECTOR(width-1 downto 0)
       );
end counter ;
```

# Top Module

```
entity TopModule is
   Port (
    clk_i: in STD_LOGIC; reset_i: in STD_LOGIC;
    data_o: out STD_LOGIC_VECTOR(3 downto 0);
   );
end TopModule ;
```

# Connecting

- A module that is used inside another module is named a component

Declaration:

**component** name_of_component

**port** ( name: *direction signal_type* ...);

**end component**;

Connecting:

label: name_of_component **port map**

(component_signal_1 => top_module_signal_1, component_signal_1 => top_module_signal_2 ... );

# Connecting

```
architecture Behvioral of TopModule is
        component prescaler
                Generic (   width : integer := 8;      value  : integer :=255        );
                Port (      clk_i: in STD_LOGIC; reset_i: in STD_LOGIC;
                            enable_o: out STD_LOGIC;
                        );
        end component;

        component counter
                Generic (   width : integer := 8        );
                Port (      clk_i: in STD_LOGIC; reset_i: in STD_LOGIC;
                            enable_i: in STD_LOGIC;
                            data_o: out STD_LOGIC_VECTOR (width-1 downto 0);
                );
        end component;

        --inner signal to connect the two components
        signal enable: STD_LOGIC;
begin
```

# Connecting

```
pr : prescaler
generic map (
                    width => 12,
                    value => 4000
                )
port map        (
                    clk_i => clk_i,
                    reset_i => reset_i,
                    enable_o => enable
                );
```

# Connecting

```
cnt : counter
generic map
                (
                        width => 4
                )
port map
                (
                        clk_i => clk_i,
                        reset_i => reset_i,
                        enable_i => enable,
                        data_o => data_o
                );

end Behavioral;
```