Is it possible to create several instances of the same component using a loop?

I have a component:

```
Component CAU is
    port
    (
        CLK              : in std_logic;
        RESET            : in std_logic;
        START            : in std_logic;
        V_DIRECTION      : in vector_3d;
        P_ORIGIN         : in vector_3d;
        V_NORMAL_IN      : in vector_3d;
        DOT_A            : in vector_3d;
        DOT_B            : in vector_3d;
        DOT_C            : in vector_3d;
        ID_DATA_IN       : in scalar;

        ID_DATA_OUT      : out scalar;
        V_REFLECT        : out vector_3d;
        V_PASS           : out vector_3d;
        P_INTERSECT      : out vector_3d;
        V_NORMAL_OUT     : out vector_3d;
        T_OUT            : out scalar;
        SUCCESS          : out std_logic;
        FINISH           : out std_logic
    );
end  Component;
```

And I want to create 8 instances of it. each is called `CAU_inst0` , `CAU_inst1` , and so on. Each of the instances is connected the following way:

```
CAU_inst0 : CAU
 PORT MAP
 (
    CLK              => CLK_CAU,
    RESET            => RESET,
    START            => start_0_sig,
    V_DIRECTION      => v_direction_0_sig,
    P_ORIGIN         => p_origin_0_sig,
    V_NORMAL_IN      => v_normal_in_0_sig,
    DOT_A            => dot_a_0_sig,
    DOT_B            => dot_b_0_sig,
    DOT_C            => dot_c_0_sig,
    ID_DATA_IN       => id_data_in_0_sig,

    ID_DATA_OUT      => id_data_out_0_sig
    V_REFLECT        => v_reflect_0_sig,
    V_PASS           => v_pass_0_sig,
    P_INTERSECT      => p_intersect_0_sig,
    V_NORMAL_OUT     => v_normal_out_0_sig,
    T_OUT            => t_0_sig,
    SUCCESS          => success_0_sig,
    FINISH           => finish_0_sig
 );
```

where for each instance `i` the number `0` is replaced with `i` . What I did was to create a Matlab script that created the 8 different instances with the correct number. But it's an ugly solution as it takes 170 lines of just the same code with little changes. Is there a way to create the components (and if possible the appropriate signals) in a loop inside the code to reduce cluter and line?

vhdl

asked Nov 2 '12 at 11:34

SIMEL
3,645 17 55 89

1 Answer

What you are looking to use is a for...generate statement.

Here is an example, similar to what you want to achieve:

```
architecture GEN of REG_BANK is
  component REG
    port(D,CLK,RESET : in  std_ulogic;
         Q           : out std_ulogic);
  end component;
begin
   GEN_REG:
   for I in 0 to 3 generate
      REGX : REG port map
```

```
        (DIN(I), CLK, RESET, DOUT(I));
    end generate GEN_REG;
end GEN;
```

In your case, you will need to make all the signals that connect up to your block vectors and or vectors of vectors.

For example, if you signal is currently defined as:

```
signal v_normal_in_0_sig : std_logic_vector(7 downto 0);
```

You will need to change it to:

```
type vector16 is array (natural range <>) of std_logic_vector(15 downto 0);
signal v_normal_in_sig : vector16(7 downto 0);
```

In this case, you can now use your signal as `v_normal_in_sig(i)` to connect to the `i` th generated instanciation of your entity/component.

Note that if you are using VHDL-2008, you can do the following instead...

```
type vector_array is array (natural range <>) of std_logic_vector;
signal v_normal_in_sig : vector_array(7 downto 0)(15 downto 0);
```

answered Nov 2 '12 at 11:45

Josh
2,731 14 33

How are the instances named after the generation? – SIMEL Nov 2 '12 at 11:52

@IlyaMelamed not sure what you mean. There is the label for the loop: GEN_REG, the loop variable, I, and the instance, REGX. Do you want to know how to reference a specific instance generated by the loop? For what purpose? – Josh Nov 2 '12 at 12:07

2 @IlyaMelamed The generated instances of `REGX` in the above code would be named `GEN_REG(0)`, `GEN_REG(1)`, `GEN_REG(2)` when simulating the design. – Peter Bennett Nov 2 '12 at 13:02