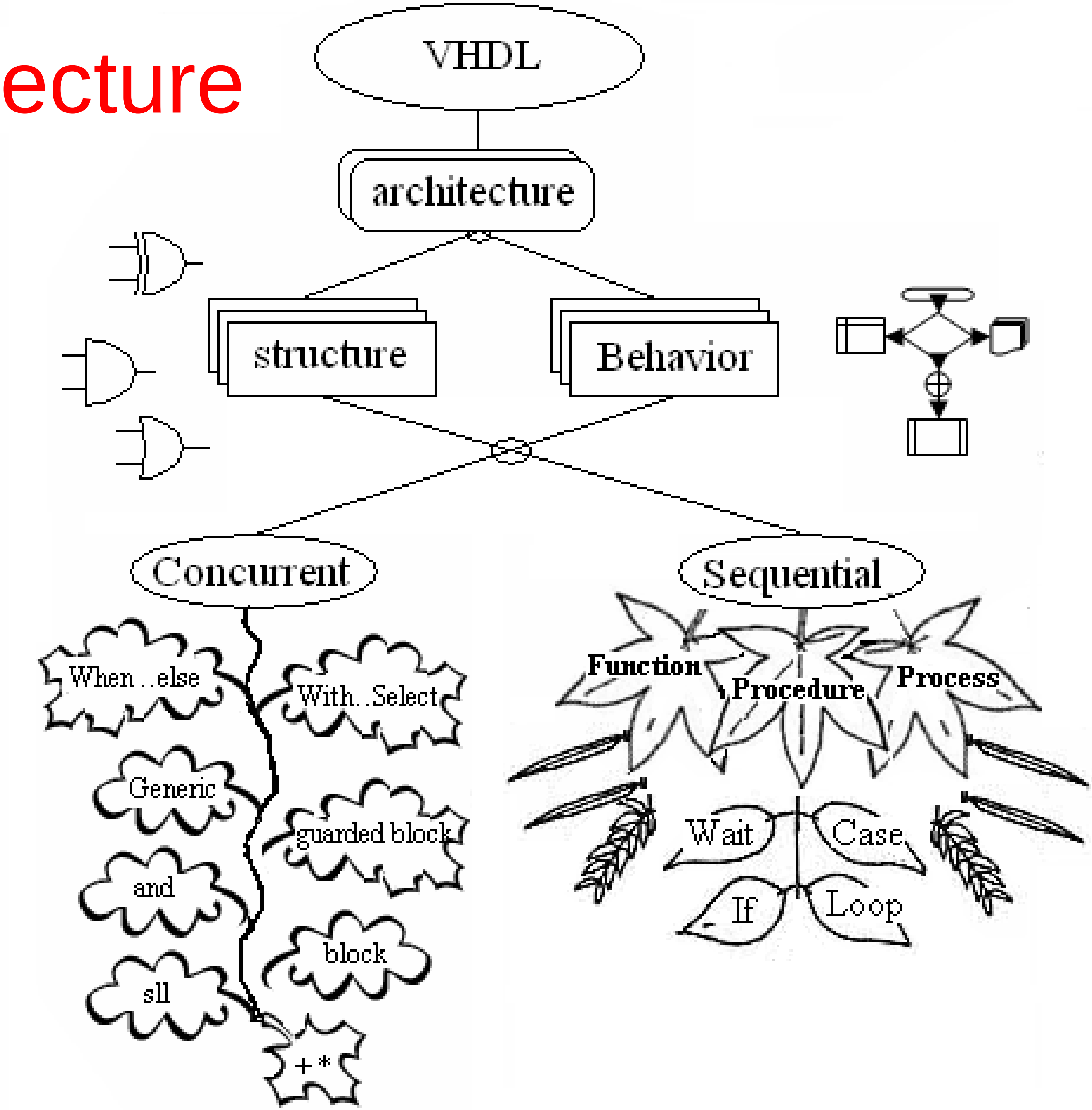
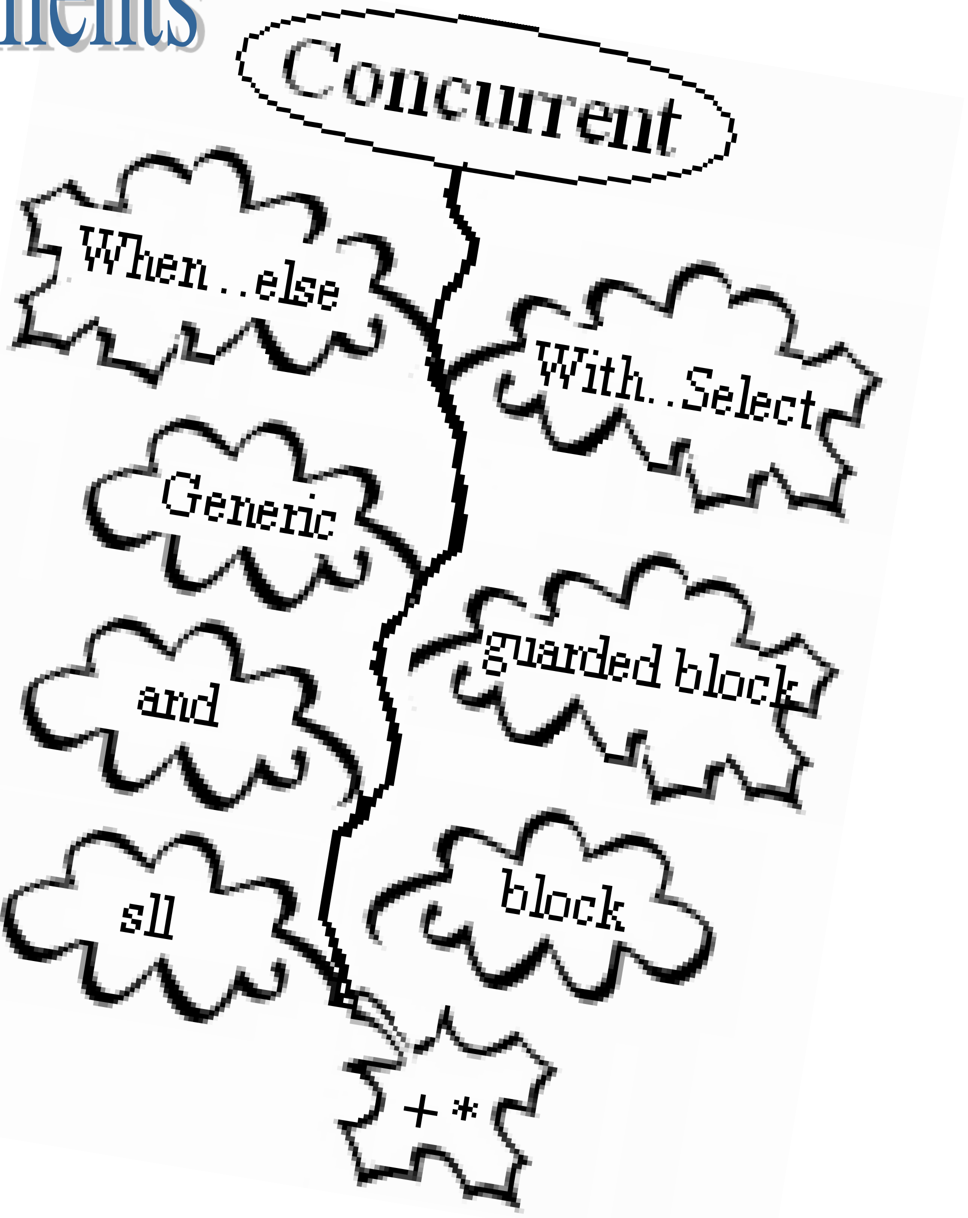
Concurrent statements

By: M. SHIPLE

Architecture



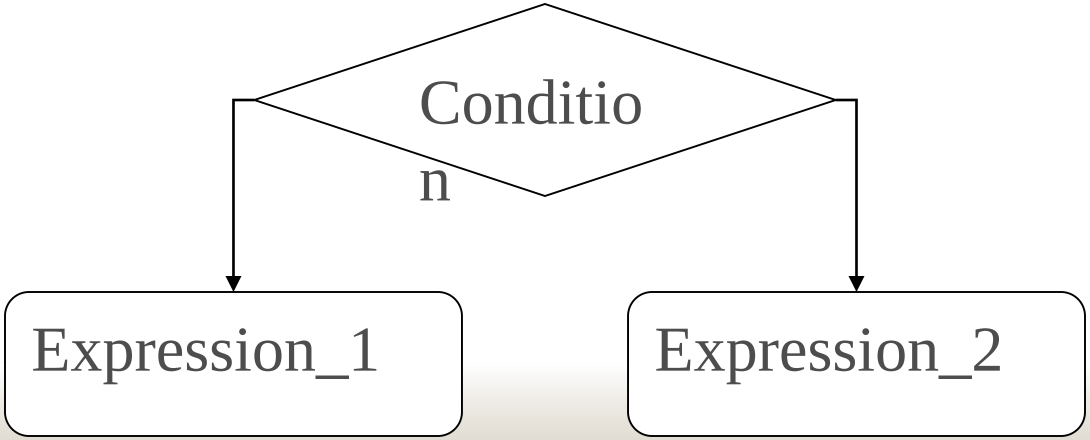
Concurrent statements



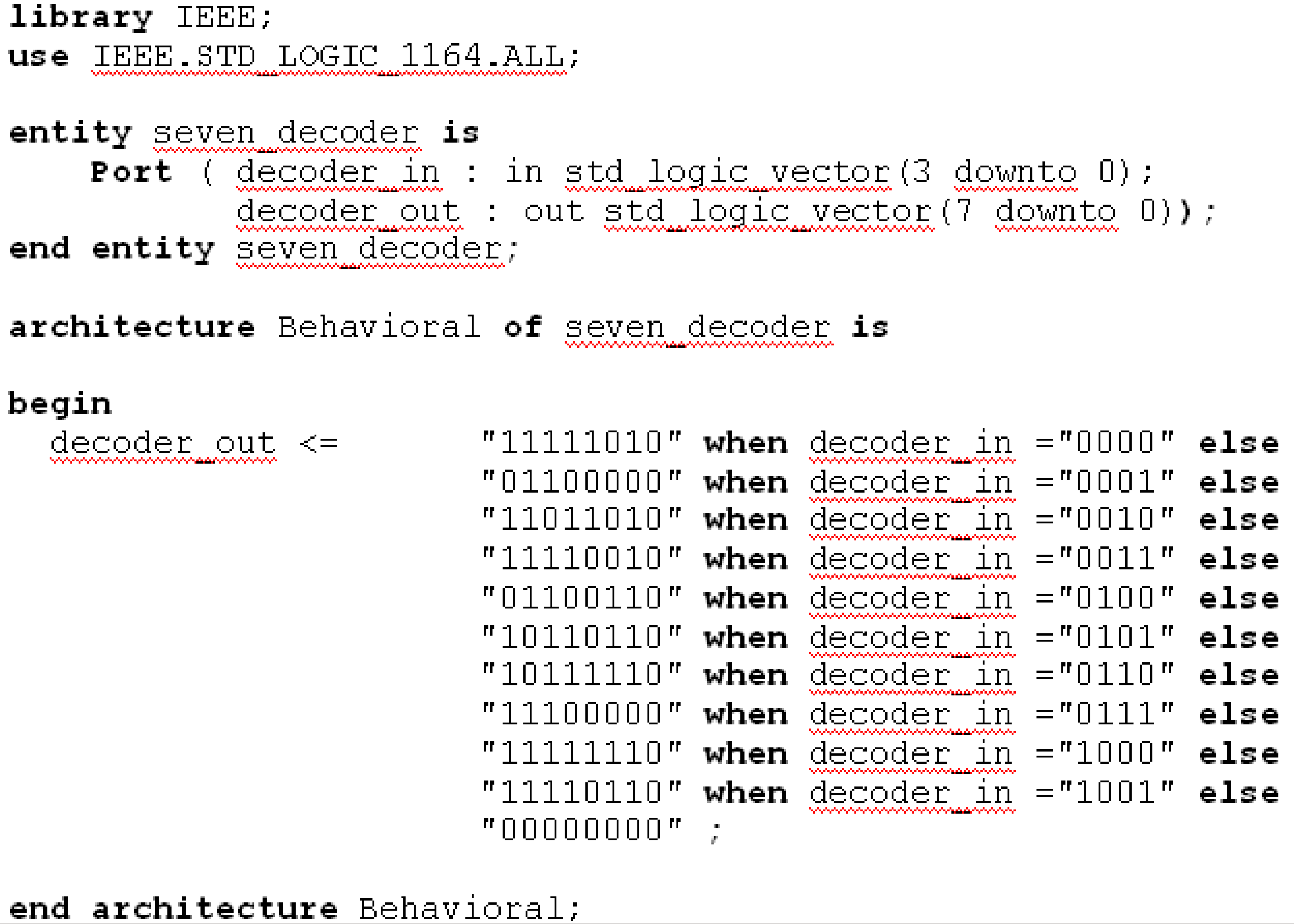
“When... else”

Syntax:

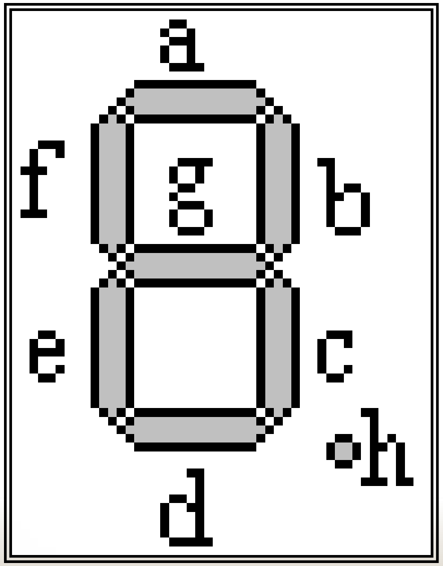
```
Signal_name <= { expression_1 when condition else }
                     expression_2;
```



Example



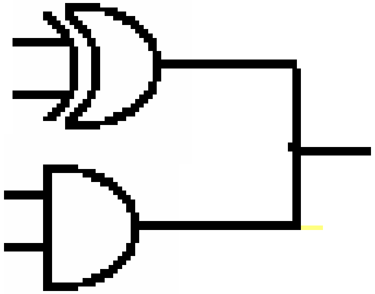
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity seven_decoder is
    Port ( decoder_in : in std_logic_vector(3 downto 0);
           decoder_out : out std_logic_vector(7 downto 0));
end entity seven_decoder;

architecture Behavioral of seven_decoder is

begin
  decoder_out <=       "11111010" when decoder_in ="0000" else
                       "01100000" when decoder_in ="0001" else
                       "11011010" when decoder_in ="0010" else
                       "11110010" when decoder_in ="0011" else
                       "01100110" when decoder_in ="0100" else
                       "10110110" when decoder_in ="0101" else
                       "10111110" when decoder_in ="0110" else
                       "11100000" when decoder_in ="0111" else
                       "11111110" when decoder_in ="1000" else
                       "11110110" when decoder_in ="1001" else
                       "00000000" ;

end architecture Behavioral;
```



Example

Multiple Driver

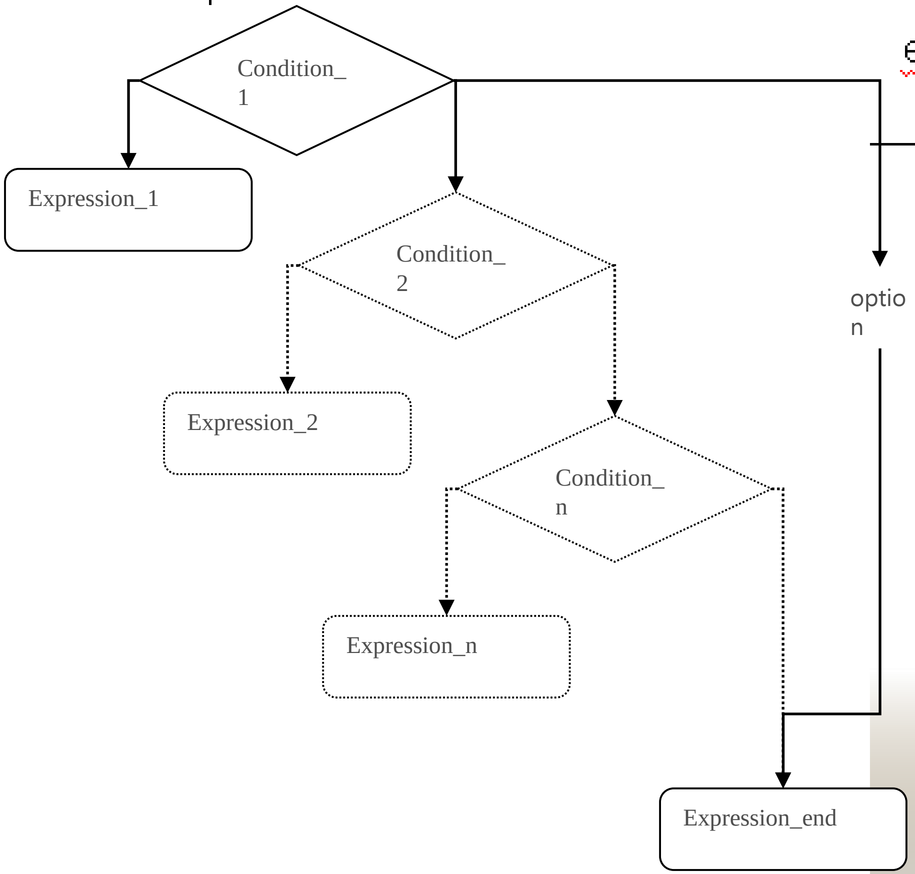
```
USE WORK.std_logic_1164.ALL;
ENTITY mux IS


END mux;
ARCHITECTURE bad OF mux IS
   BEGIN
         q <= i0 WHEN a = ‘0’ AND b = ‘0’ ELSE ‘0’;
         q <= i1 WHEN a = ‘1’ AND b = ‘0’ ELSE ‘0’;
         q <= i2 WHEN a = ‘0’ AND b = ‘1’ ELSE ‘0’;
         q <= i3 WHEN a = ‘1’ AND b = ‘1’ ELSE ‘0’;
END BAD;
```



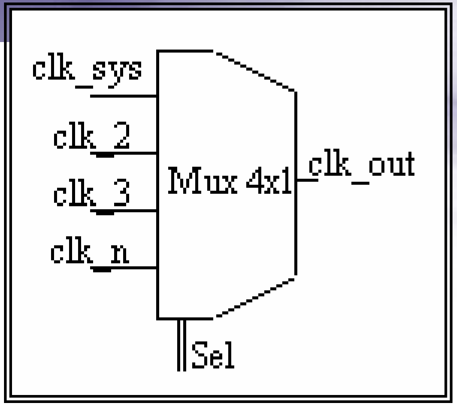
“With...select”

Syntax:

```
with selection_signal select
      signal_name <= { expression_1 when choice_1, }
                       expression_2 when choice_2,

                       expression_n when choice_n;
```



Example

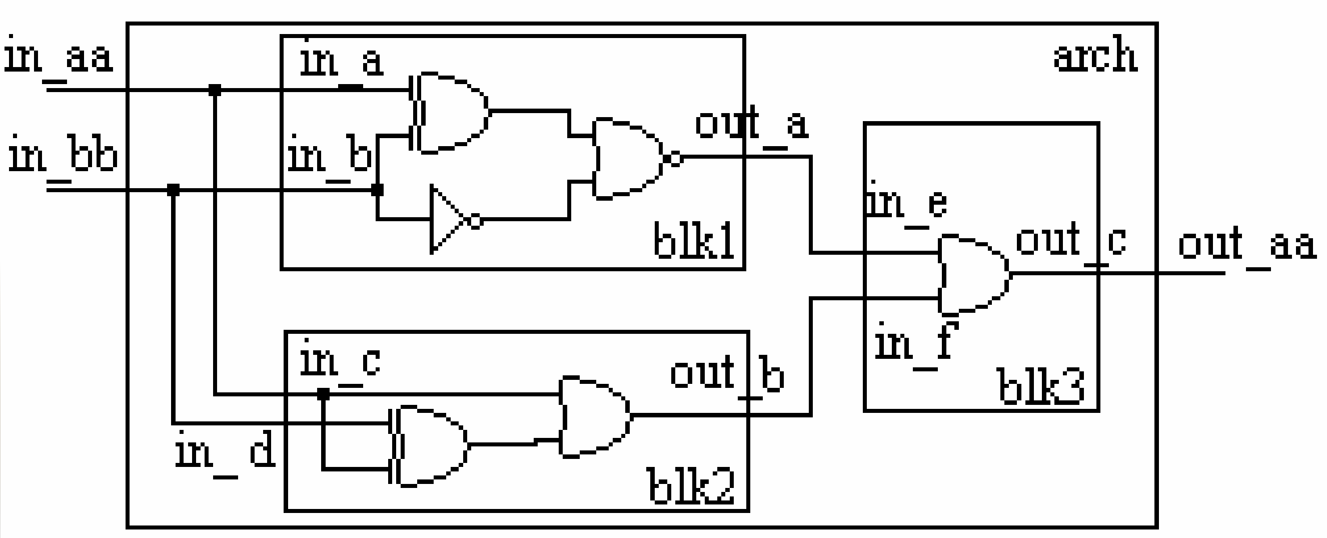


```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Mux4x1 is
    Port ( clk_sys  : in std_logic;
           clk_2    : in std_logic;
           clk_3    : in std_logic;
           clk_n    : in std_logic;
           Sel      : in std_logic_vector(1 downto 0);
           clk_out  : out std_logic);
end entity Mux4x1;

architecture concurrent of Mux4x1 is

begin
          with Sel select
          clk_out <= clk_sys when "00",
                     clk_2 when "01",
                     clk_3 when "10",
                     clk_n when OTHERS;

end architecture concurrent;
```

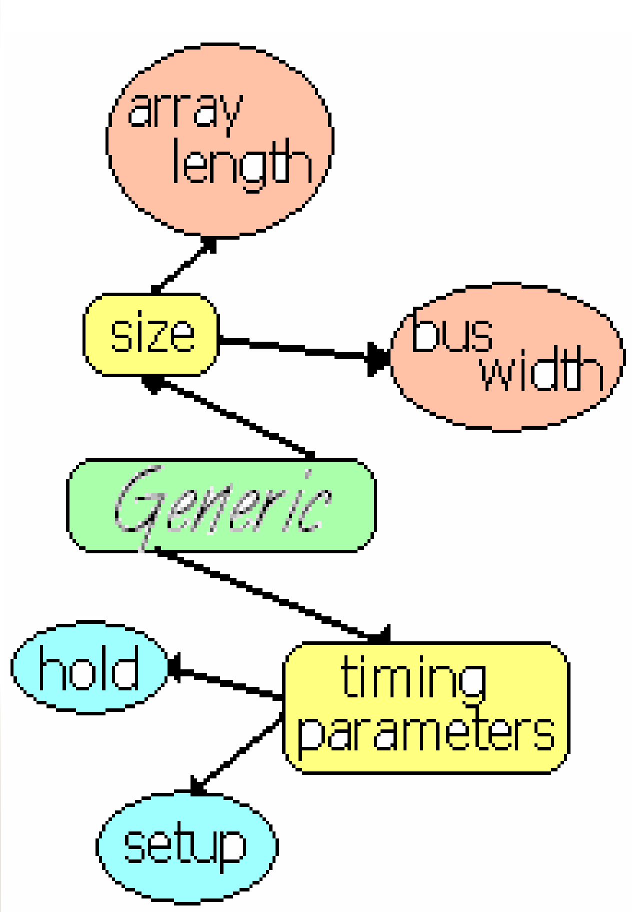
“Block” statement

Block Syntax:

```
label: block
   block_declarative domain;
begin
   concurrent statements;
end block  label ;
```



“Generic” statement

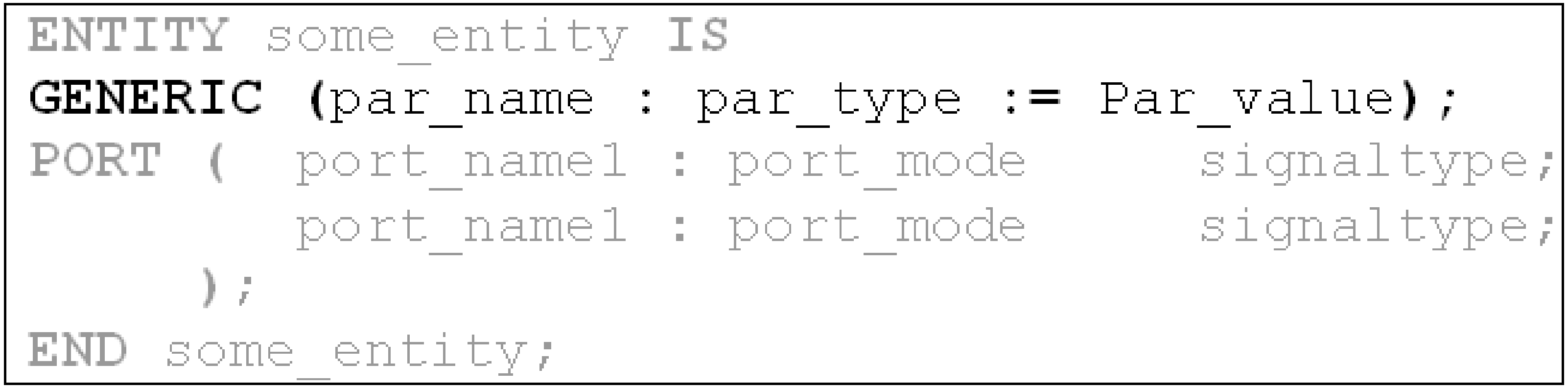


Synthesis

Simulation

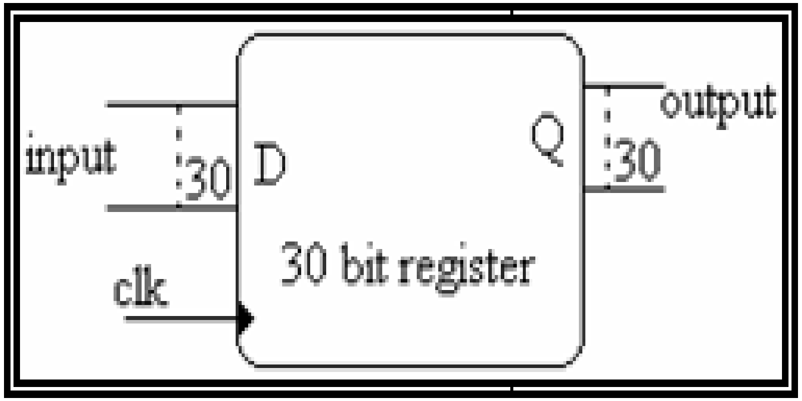
“Generic” statement

Syntax:



```
ENTITY some_entity IS
GENERIC (par_name : par_type := Par_value);
PORT (  port_name1 : port_mode     signaltype;
        port_name1 : port_mode     signaltype;
      );
END some_entity;
```

Exercise



```
LIBRARY IEEE ;
USE IEEE .Std_logic_1164 .ALL;

ENTITY example IS
generic (width : Integer := 30 );       -- top bit
    PORT (
          clk    : IN      std_logic ;
          input  : IN      std_logic_vector (width downto 0);
          output : OUT     std_logic_vector (width downto 0)
          ) ;
```

END