

Digital Systems - 1

Combinational Constructs

Pere Palà Schönwälder

iTIC <http://itic.cat>

February 2025

Boolean functions

► $f(A, B, C) = A \cdot B + \overline{A} \cdot \overline{C}$

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

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

architecture my_arch of boolean_function is
begin
    f <= (A and B) or ((not A) and (not C));
end;
```

► Precedence rules: Highest priority first

```
not
and
or
nand
nor
xor
xnor
```

Traffic light controller

- Define a bus: one-hot encoding (red, yellow, green)

```
entity light_controller is
    port( lights_in  : in  std_logic_vector (3 downto 1);
          enable     : in  std_logic;
          lights_out : out std_logic_vector (3 downto 1));
end entity light_controller;

architecture and_enable of light_controller is
begin
    lights_out(1) <= lights_in(1) and enable;
    lights_out(2) <= lights_in(2) and enable;
    lights_out(3) <= lights_in(3) and enable;
end architecture and_enable;
```

- Alternative

```
architecture conditional_enable of light_controller is
begin
    lights_out <= lights_in when enable = '1' else
                  "000";
end architecture conditional_enable;
```

Traffic light controller. Remarks

- ▶ Buses

- ▶ `std_logic_vector (3 downto 1)`

- ▶ `std_logic_vector (7 downto 0)`

- ▶ `when... else...`

```
signal_name <= value1 when condition1 else  
                value2 when condition2 else  
                value3 when condition3 else  
                default_value;
```

- ▶ The first true condition determines the assignment

- ▶ There is priority!

- ▶ Ensure to cover all cases, for instance, using `else`

7 segment decoder

- ▶ Input: BCD coded
- ▶ Output: 7 segment



```
entity BCDdecoder is
    port( BCD_in      : in  std_logic_vector (3 downto 0);
          segments_out : out std_logic_vector (6 downto 0));
end entity BCDdecoder;

architecture my_arch of BCDdecoder is
begin
    segments_out <= "1111110" when BCD_in = "0000" else
                    "0110000" when BCD_in = "0001" else
                    "1101101" when BCD_in = "0010" else
                    "1111001" when BCD_in = "0011" else
                    "0110011" when BCD_in = "0100" else
                    "1011011" when BCD_in = "0101" else
                    "1011111" when BCD_in = "0110" else
                    "1110000" when BCD_in = "0111" else
                    "1111111" when BCD_in = "1000" else
                    "1110011" when BCD_in = "1001" else
                    "-----";
end architecture my_arch;
```

7 segment decoder /2

- Less typing: with ... select ...

```
entity BCDdecoder is
    port( BCD_in      : in  std_logic_vector (3 downto 0);
          segments_out : out std_logic_vector (6 downto 0));
end entity BCDdecoder;

architecture my_arch of BCDdecoder is
begin
    with BCD_in select
        segments_out <= "1111110" when "0000",
                        "0110000" when "0001",
                        "1101101" when "0010",
                        "1111001" when "0011",
                        "0110011" when "0100",
                        "1011011" when "0101",
                        "1011111" when "0110",
                        "1110000" when "0111",
                        "1111111" when "1000",
                        "1110011" when "1001",
                        "-----" when others;
end architecture my_arch;
```

Equivalent Constructs

- ▶ Functionally equivalent
 - ▶ `with... select`
 - ▶ `when... else`
- ▶ Use `with... select` for multi-condition multiplexers (compact, clearer).
- ▶ Use `when... else` for simple conditional assignments or two-way selections (clearer and often more intuitive).

7 segment decoder with a blanking input

```
entity BCDdecoder is
    port( BCD_in      : in  std_logic_vector (3 downto 0);
          blank       : in  std_logic_vector;
          segments_out : out std_logic_vector (6 downto 0));
end entity BCDdecoder;

architecture my_arch of BCDdecoder is
    signal inner_segs : std_logic_vector (6 downto 0);
begin    --Internal, auxiliar signal
    with BCD_in select
        inner_segs <= "1111110" when "0000",
                       "0110000" when "0001",
                       "1101101" when "0010",
                       "1111001" when "0011",
    ...
                       "1111111" when "1000",
                       "1110011" when "1001",
                       "-----" when others;

    segments_out <= "0000000" when blank = '1' else
                    inner_segs;
end architecture my_arch;
```