DIGITAL LOGIC WITH VHDL
(Fall 2013)
Unit 2

- *Use of std_logic_vector.*

✓ *CONCURRENT DESCRIPTION*

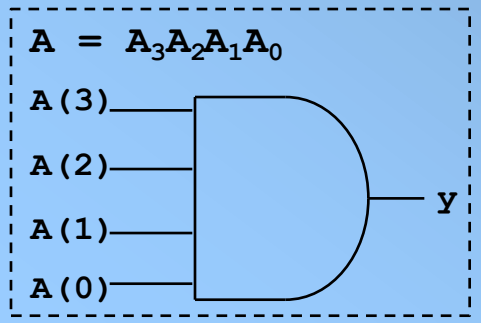
- *'with-select', 'when-else' statements*
- *Examples: multiplexor, decoder.*

Instructor: Daniel Llamocca

✓ std_logic_vector

- In the example, we use the *std_logic_vector* type for an input signal.

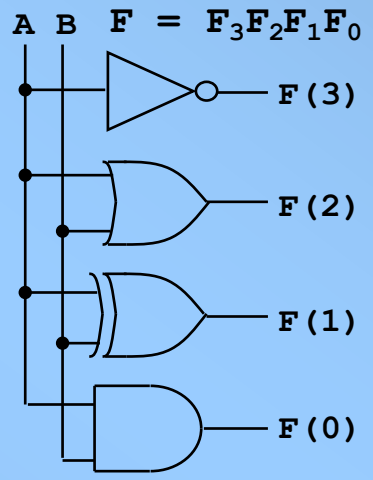
```
library ieee;
use ieee.std_logic_1164.all;

entity test is
  port ( A: in std_logic_vector (3 downto 0);
         -- A: |A3|A2|A1|A0|
         y: out std_logic);
end test;

architecture struct of test is

begin
    -- The circuit represents an AND gate
    -- with 4 inputs: A(3), A(2), A(1), A(0)
    y <= A(3) and A(2) and A(1) and A(0);

end struct;
```



✓ std_logic_vector

- In the example, we use the *std_logic_vector* type for an output signal.

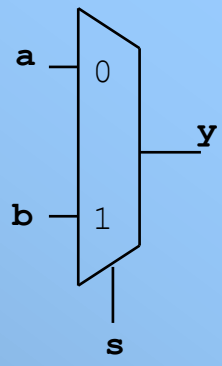
```
library ieee;
use ieee.std_logic_1164.all;

entity tst is
  port ( A,B: in std_logic;
         F: out std_logic_vector (3 downto 0);
         -- F: |F3|F2|F1|F0
end tst;

architecture struct of tst is

begin
    F(0) <= A and B; F(1) <= A xor B;
    F(2) <= A or B; F(3) <= not(A);

end struct;
```



✓ *CONCURRENT DESCRIPTION*

- In this description, circuits are specified in an structured fashion. The order of the statements is irrelevant: all the statements represent circuits that are working at the same time. This type of description is well-suited for combinatorial circuits.
- The use of sentences with ‘and’, ‘or’, ‘xor’, ‘nand’, ‘nor’, ‘xnor’, and ‘not’ is a basic instance of the concurrent description. It is sometimes called ‘horizontal description’.
- In Unit 4, the so-called ‘structural description’ is just a generalization of the concurrent description.
- Since we already know how to build circuits based on logic gates, we now present two concurrent assignment statements (*with – select*, *when – else*), which are far more powerful than the statements with logic gates when it comes to describe complex circuits.

- **CONCURRENT ASSIGNMENT STATEMENTS:**
  - **Selected Signal Assignment: WITH-SELECT:** This statement allows assigning values to a signal based on certain criterion.
  - **MUX 2-to-1:**



s	a	b	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

s	y
0	a
1	b

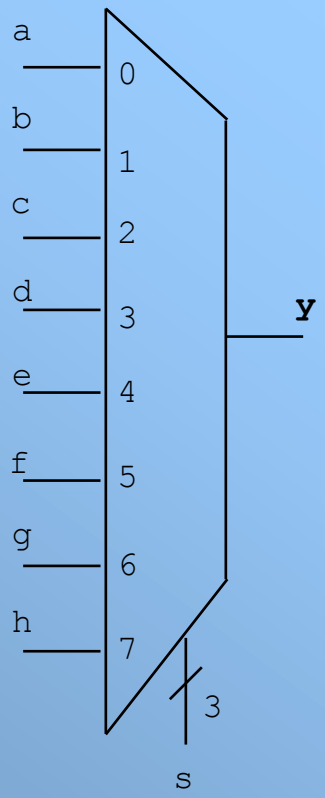
$$y = \bar{s}(a\,\bar{b} + a\,b) + s(\bar{a}\,b + a\,b)$$

$$y = \bar{s}a + sb$$

with s select: 's'specifies the selection criterion
when specifies the value assigned to 'y' for each value of 's'
when others: we need to include all the possible values of 's', that are 9 according to *std_logic* type.

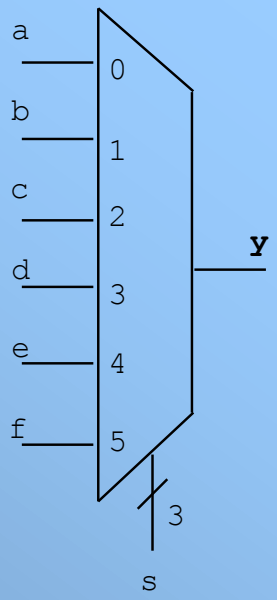
```
library ieee;
use ieee.std_logic_1164.all;

entity my_mux21 is
 port ( a, b, s: in std_logic;
        y: out std_logic);
end my_mux21;

architecture st of my_mux21 is
begin
  with s select
       y <= a when '0',
            b when others;
end st;
```

- Selected Signal Assignment (WITH - SELECT):
- MUX 8-to-1




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

entity my_mux81 is
 port ( a,b,c,d,e,f,g,h: in std_logic;
        s: in std_logic_vector (2 downto 0);
        y: out std_logic);
end my_mux81;

architecture struct of my_mux81 is
begin
 with s select
  y <= a when "000", -- note ',' instead of ';'
       b when "001",
       c when "010",
       d when "011",
       e when "100",
       f when "101",
       g when "110",
       h when others;
end struct;
```

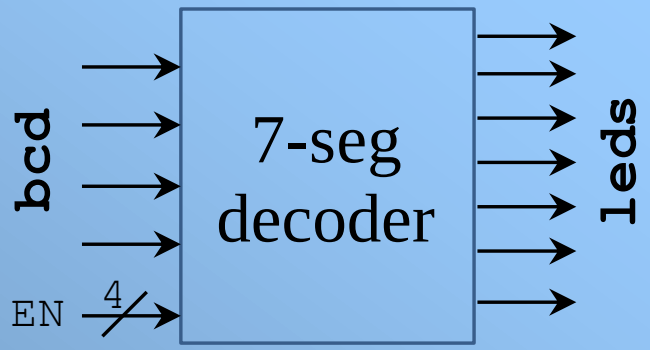
- Selected Signal Assignment(WITH - SELECT):
- MUX 6-to-1




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

entity my_mux61 is
 port ( a,b,c,d,e,f: in std_logic;
        s: in std_logic_vector (2 downto 0);
        y: out std_logic);
end my_mux61;

architecture struct of my_mux61 is
begin
  with s select
     y <= a when "000",
          b when "001",
          c when "010",
          d when "011",
          e when "100",
          f when "101",
          '-' when others;
end struct;
```

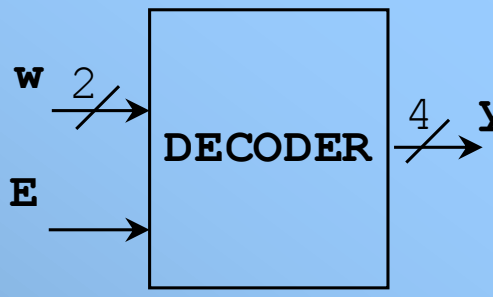
Value '–': Don't care output
It helps the synthesizer to optimize the circuit

- **7-segment DECODER** (outputs >= inputs)



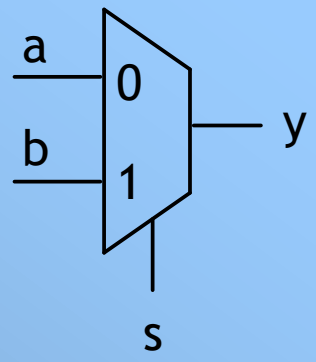

```
library ieee;
use ieee.std_logic_1164.all;

entity sevenseg is
 port ( bcd: in std_logic_vector (3 downto 0);
        sevenseg: out std_logic_vector (6 downto 0);
        EN: in std_logic_vector (3 downto 0));
end sevenseg;

architecture struct of sevenseg is
    signal leds: std_logic_vector (6 downto 0);
begin
-- |  a  | b   |  c  |  d  |  e  |  f  |  g  |
-- |leds6|leds5|leds4|leds3|leds2|leds1|leds0|
    with bcd select
       leds <= "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",
               "1111011" when "1001",
               "-------" when others;
    -- Nexys3: LEDs are active low.
    --   Each 7-seg display has an active-low enable
    EN <= "0111";
    sevenseg <= not(leds);
end struct;
```

- Decoder 2-to-4 with enable:




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

entity dec2to4 is
 port (w: in std_logic_vector (1 downto 0);
       E: in std_logic;
       y: out std_logic_vector (3 downto 0));
end dec2to4;

architecture struct of dec2to4 is
   signal Ew: std_logic_vector (2 downto 0);
begin
  Ew <= E & w; -- concatenation
  with Ew select
     y <= "0001" when "100",
          "0010" when "101",
          "0100" when "110",
          "1000" when "111",
          "0000" when others;
end struct;
```

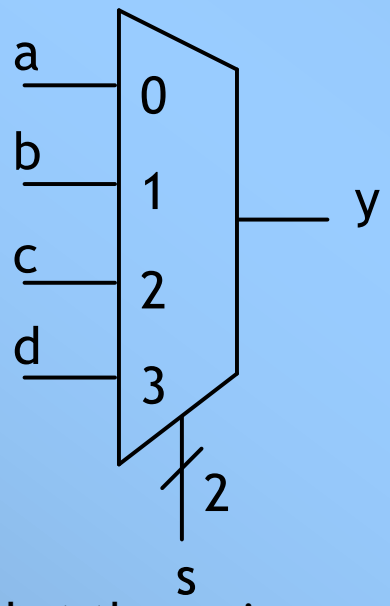
- **CONCURRENT ASSIGNMENT STATEMENTS :**
  - **Conditional signal assignment: WHEN - ELSE:** Similarly to the selected signal assignment, this statement allows a signal to take one out of many values based on a certain condition. The syntax however is different and it allows to describe circuits in a more compact fashion.
  - **Example:** MUX 2-to-1



- If the condition on when is FALSE, we assign a value to 'y' after else. This assignment can also be conditioned by another when-else clause (see next example)

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

entity mux21_cond is
 port ( a, b, s: in std_logic;
        y: out std_logic);
end mux21_cond;

architecture est of mux21_cond is
begin
   y <= a when s = '0' else b;
end est;
```

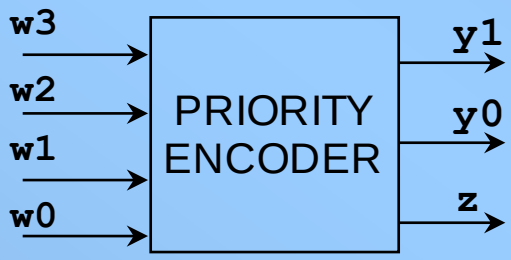
- **Conditional signal assignment (WHEN - ELSE):**
- **Example:** MUX 4-to-1




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

entity mux41_cond is
 port (a, b, c, d: in std_logic;
       s: in std_logic_vector (1 downto 0);
       y: out std_logic);
end mux41_cond;

architecture est of mux41_cond is
begin
   y <= a when s = "00" else
        b when s = "01" else
        c when s = "10" else
        d;
end est;
```

- Note that the assignment on ‘b’ is conditioned by another when-else clause. Same for ‘c’. Only the assignemnt of ‘d’ is not conditioned. There is no limit to the number of nested conditions.

- **Conditional signal assignment (WHEN - ELSE):**
- **Example:** Priority Encoder 4-to-2

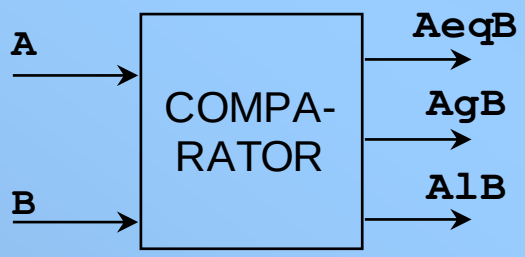


w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	0	0	0
1	x	x	x	1	1	1
0	1	x	x	1	0	1
0	0	1	x	0	1	1
0	0	0	1	0	0	1

when-else has a priority level, thus it is easy to describe a priority encoder. With with-select describing this circuit would be very tedious.

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

entity my_prienc is
 port ( w: in std_logic_vector (3 downto 0);
        y: out std_logic_vector (1 downto 0);
        z: out std_logic);
end my_prienc;

architecture struct of my_prienc is
begin
  y <= "11" when w(3) = '1' else
       "10" when w(2) = '1' else
       "01" when w(1) = '1' else
       "00";
  z <= '0' when w = "0000" else '1';
  -- If no input is '1', z is '0'
end struct;
```

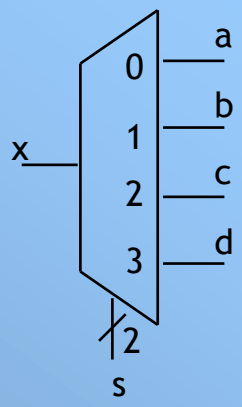
- **Conditional signal assignment (WHEN - ELSE):**
- **Example:** 4-bit comparator




```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all; -- unsigned #s

entity my_comp is
 port ( A,B: in std_logic_vector (3 downto 0);
        AeqB, AgB, AlB: out std_logic);
end my_comp;

architecture struct of my_comp is
begin

  AeqB <= '1' when A = B else '0';
  AgB  <= '1' when A > B else '0';
  AlB  <= '1' when A < B else '0';

end struct;
```

- Note the use of the operators '=, >, <' to compare numbers
- Always indicate what type of numbers we are working with. In the example we are using unsigned numbers. For 2's complement, use 'signed'.

- **Conditional Signal Assignment (WHEN - ELSE):**
- **Example:** Demultiplexor




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

entity my_demux is
 port ( s: in std_logic_vector (1 downto 0);
        x: in std_logic;
        a,b,c,d: out std_logic);
end my_demux;

architecture struct of my_demux is
begin

  a <= x when s = "00" else '0';
  b <= x when s = "01" else '0';
  c <= x when s = "10" else '0';
  d <= x when s = "11" else '0';

end struct;
```