VHDL Basics

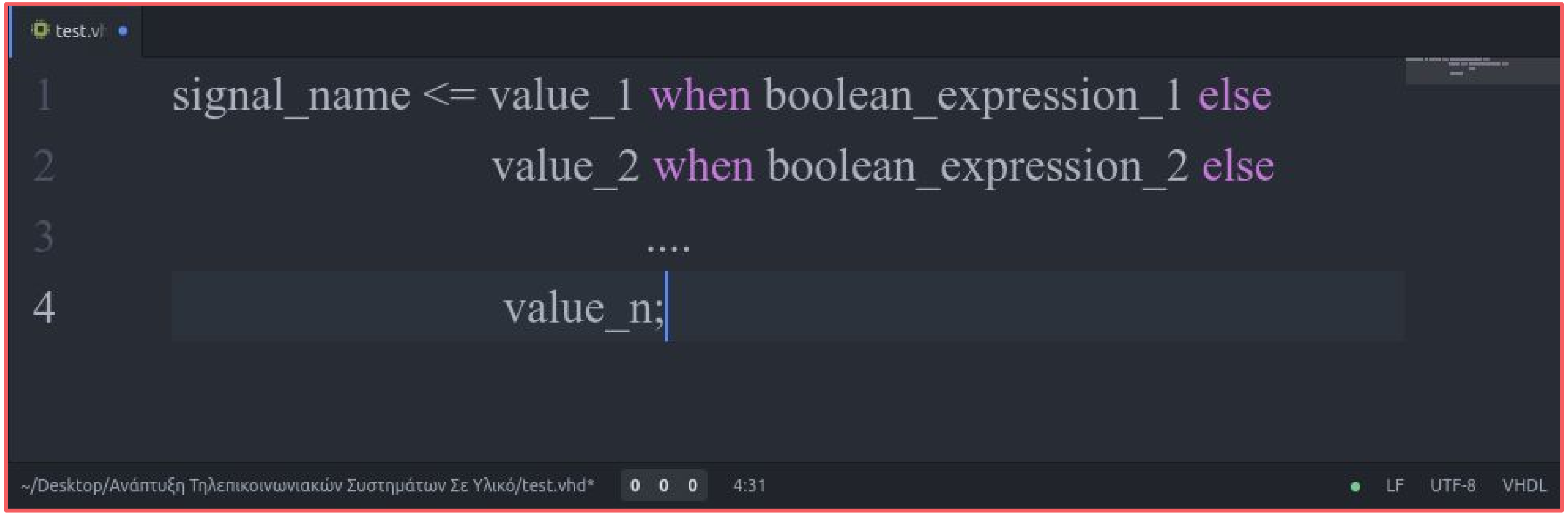
Δαδαλιάρης Αντώνιος
dadaliaris@cs.uth.gr

VHDL: Concurrent Signal Assignment Statements

- A signal can be present at both sides of a concurrent assignment statement.
  - my_signal <= my_signal AND another_signal;
  - Q <= ((NOT q) AND (NOT en)) OR x;
- Although the first statement is syntactically right, it is bad practice to use it this way since it can lead to obscure synthesis results.

VHDL: Concurrent Signal Assignment Statements (cont.)

- Conditional Signal Assignment Statements



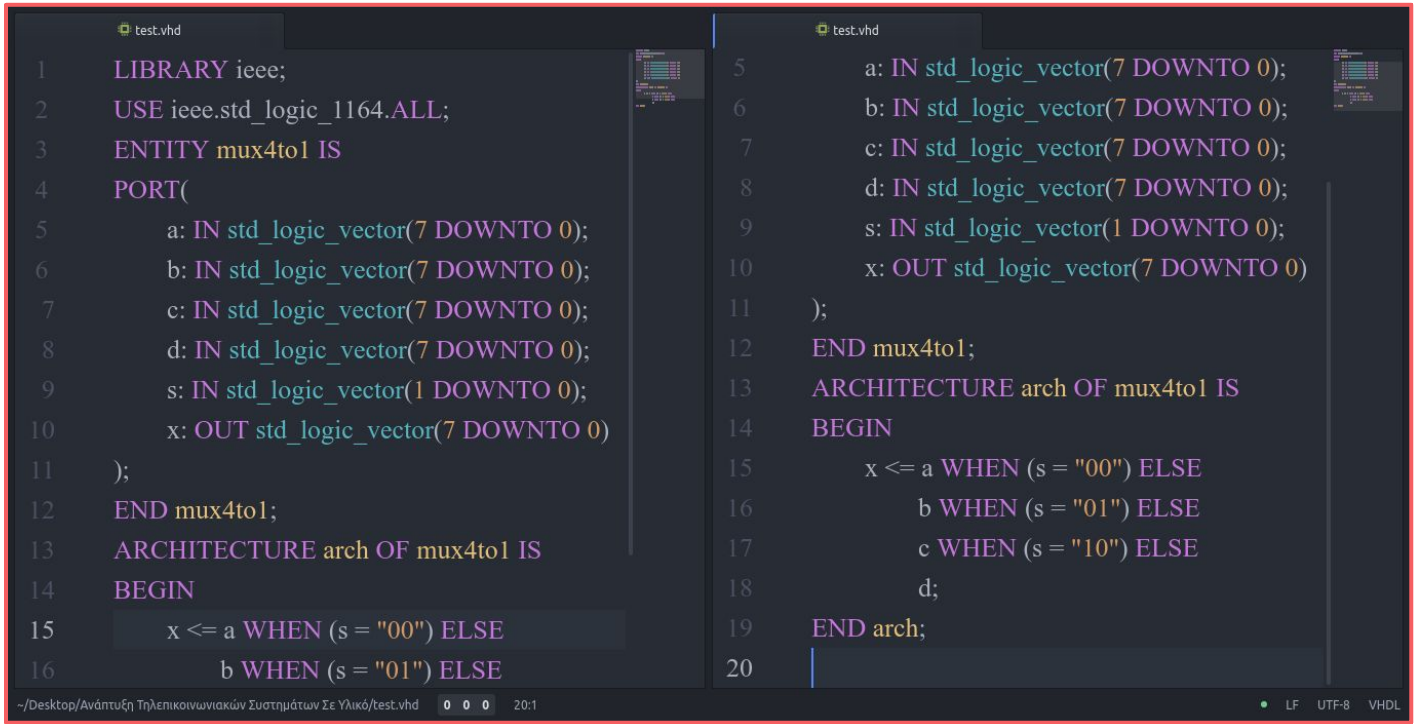
test.vhd

```
signal_name <= value_1 when boolean_expression_1 else
               value_2 when boolean_expression_2 else
                     ....
               value_n;
```

~/Desktop/Ανάπτυξη Τηλεπικοινωνιακών Συστημάτων Σε Υλικό/test.vhd* 0 0 0 4:31 LF UTF-8 VHDL

VHDL: Concurrent Signal Assignment Statements (cont.)

- Conditional signal assignment statements are represented by multiplexers.



test.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY mux4to1 IS
PORT(
        a: IN std_logic_vector(7 DOWNTO 0);
        b: IN std_logic_vector(7 DOWNTO 0);
        c: IN std_logic_vector(7 DOWNTO 0);
        d: IN std_logic_vector(7 DOWNTO 0);
        s: IN std_logic_vector(1 DOWNTO 0);
        x: OUT std_logic_vector(7 DOWNTO 0)
);
END mux4to1;
ARCHITECTURE arch OF mux4to1 IS
BEGIN
        x <= a WHEN (s = "00") ELSE
                b WHEN (s = "01") ELSE
```

test.vhd

```
        a: IN std_logic_vector(7 DOWNTO 0);
        b: IN std_logic_vector(7 DOWNTO 0);
        c: IN std_logic_vector(7 DOWNTO 0);
        d: IN std_logic_vector(7 DOWNTO 0);
        s: IN std_logic_vector(1 DOWNTO 0);
        x: OUT std_logic_vector(7 DOWNTO 0)
);
END mux4to1;
ARCHITECTURE arch OF mux4to1 IS
BEGIN
        x <= a WHEN (s = "00") ELSE
                b WHEN (s = "01") ELSE
                c WHEN (s = "10") ELSE
                d;
END arch;

```

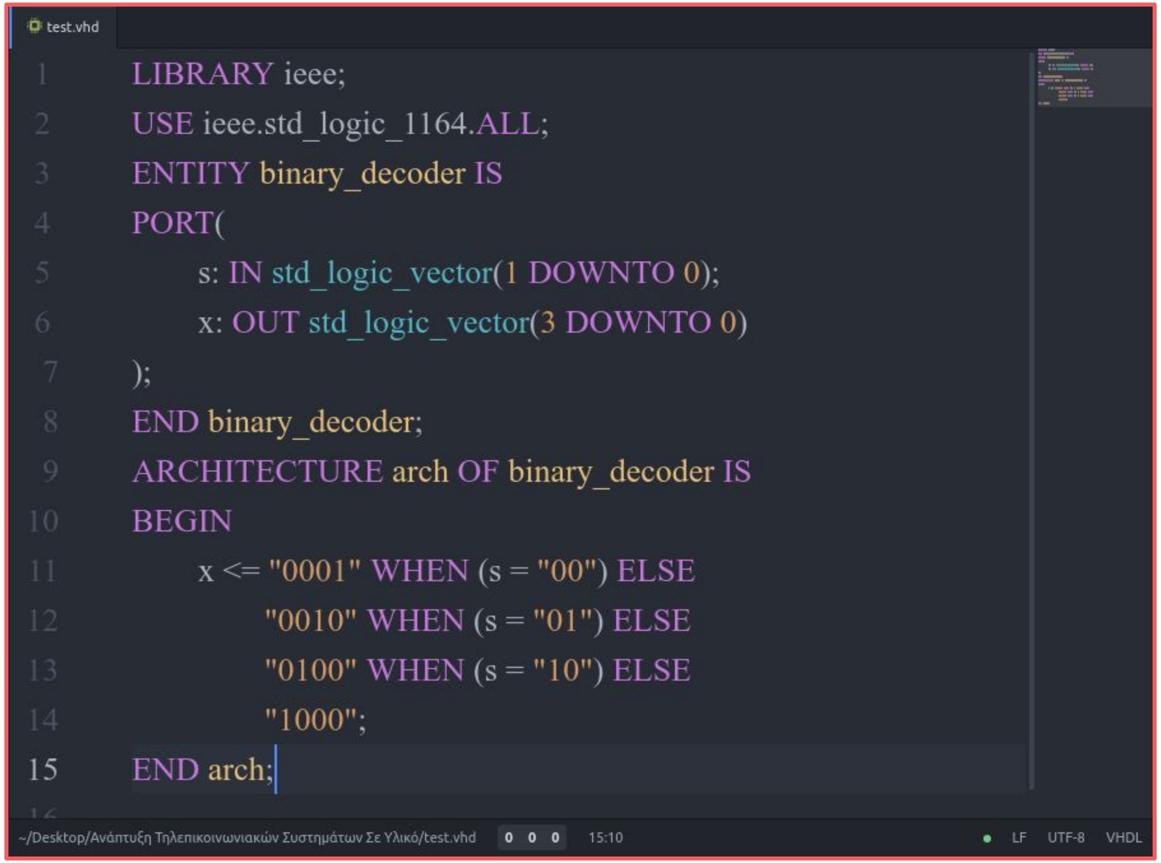
~/Desktop/Ανάπτυξη Τηλεπικοινωνιακών Συστημάτων Σε Υλικό/test.vhd 0 0 0 20:1 LF UTF-8 VHDL

VHDL: Concurrent Signal Assignment Statements (cont.)

- Binary Decoder:
  - Input: n-bit
  - Output: 2^n-bit
  - Truth Table:

s	x
00	0001
01	0010
10	0100
11	1000

VHDL: Concurrent Signal Assignment Statements (cont.)



```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY binary_decoder IS
PORT(
        s: IN std_logic_vector(1 DOWNTO 0);
        x: OUT std_logic_vector(3 DOWNTO 0)
);
END binary_decoder;
ARCHITECTURE arch OF binary_decoder IS
BEGIN
        x <= "0001" WHEN (s = "00") ELSE
                "0010" WHEN (s = "01") ELSE
                "0100" WHEN (s = "10") ELSE
                "1000";
END arch;
```

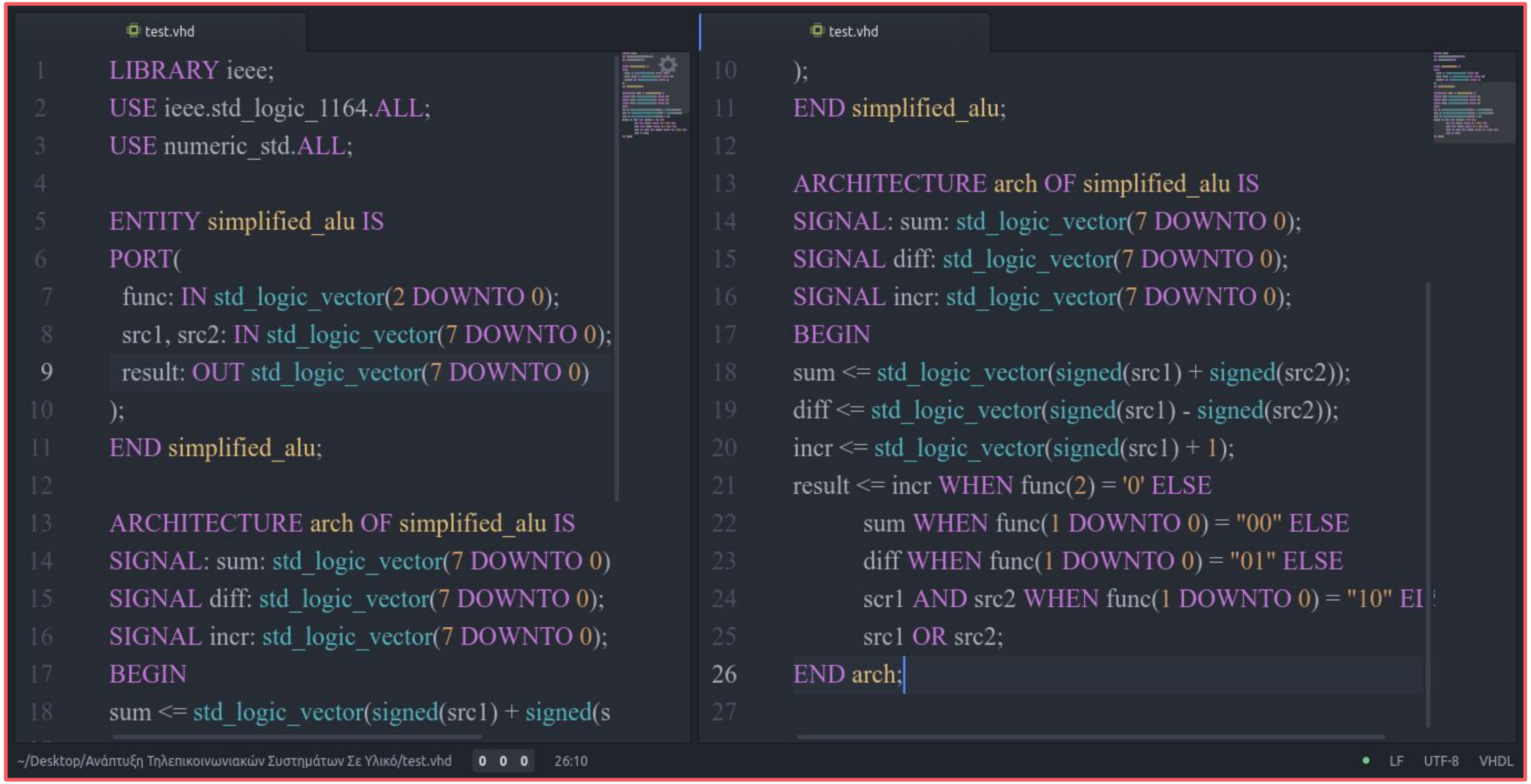
~/Desktop/Ανάπτυξη Τηλεπικοινωνιακών Συστημάτων Σε Υλικό/test.vhd 0 0 0 15:10 LF UTF-8 VHDL

VHDL: Concurrent Signal Assignment Statements (cont.)

- Arithmetic Logic Unit (ALU):
  - Functionality

0---	src1 + 1
100	src1 + src2
101	src1 - src2
110	src1 and src2
111	src1 or src2

VHDL: Concurrent Signal Assignment Statements (cont.)



test.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE numeric_std.ALL;

ENTITY simplified_alu IS
PORT(
  func: IN std_logic_vector(2 DOWNTO 0);
  src1, src2: IN std_logic_vector(7 DOWNTO 0);
  result: OUT std_logic_vector(7 DOWNTO 0)
);
END simplified_alu;

ARCHITECTURE arch OF simplified_alu IS
SIGNAL: sum: std_logic_vector(7 DOWNTO 0)
SIGNAL diff: std_logic_vector(7 DOWNTO 0);
SIGNAL incr: std_logic_vector(7 DOWNTO 0);
BEGIN
sum <= std_logic_vector(signed(src1) + signed(s
```

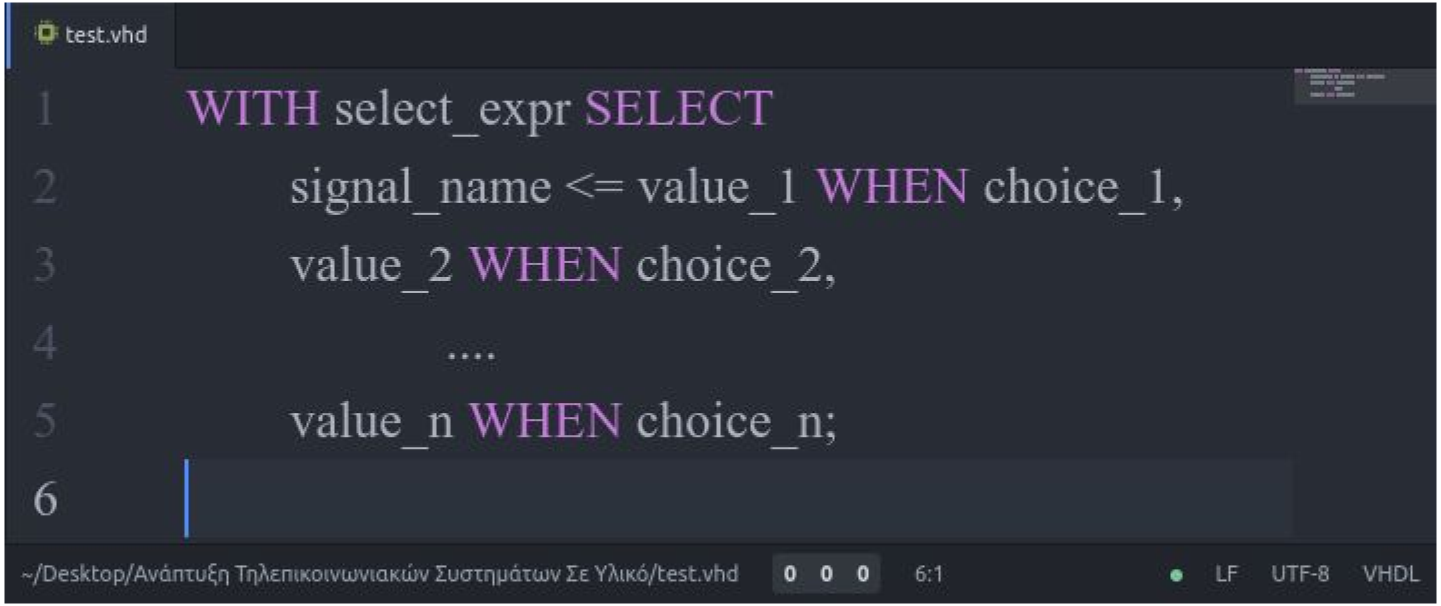
test.vhd

```
);
END simplified_alu;

ARCHITECTURE arch OF simplified_alu IS
SIGNAL: sum: std_logic_vector(7 DOWNTO 0);
SIGNAL diff: std_logic_vector(7 DOWNTO 0);
SIGNAL incr: std_logic_vector(7 DOWNTO 0);
BEGIN
sum <= std_logic_vector(signed(src1) + signed(src2));
diff <= std_logic_vector(signed(src1) - signed(src2));
incr <= std_logic_vector(signed(src1) + 1);
result <= incr WHEN func(2) = '0' ELSE
          sum WHEN func(1 DOWNTO 0) = "00" ELSE
          diff WHEN func(1 DOWNTO 0) = "01" ELSE
          scr1 AND src2 WHEN func(1 DOWNTO 0) = "10" EI
          src1 OR src2;
END arch;

```

```
~/Desktop/Ανάπτυξη Τηλεπικοινωνιακών Συστημάτων Σε Υλικό/test.vhd   0 0 0   26:10        LF   UTF-8   VHDL
```

VHDL: Concurrent Signal Assignment Statements (cont.)

- Selected Signal Assignment Statements

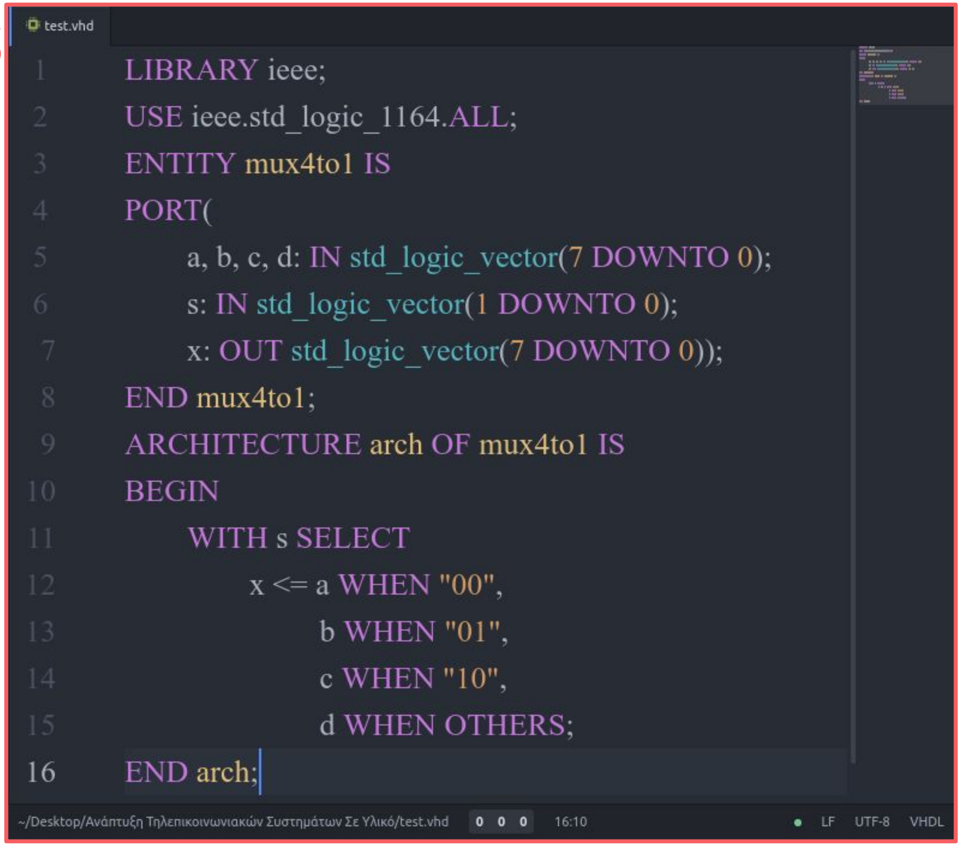


test.vhd

```
WITH select_expr SELECT
        signal_name <= value_1 WHEN choice_1,
        value_2 WHEN choice_2,
                ....
        value_n WHEN choice_n;
```

~/Desktop/Ανάπτυξη Τηλεπικοινωνιακών Συστημάτων Σε Υλικό/test.vhd 0 0 0 6:1 LF UTF-8 VHDL

VHDL: Concurrent Signal Assignment Statements



test.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY mux4to1 IS
PORT(
        a, b, c, d: IN std_logic_vector(7 DOWNTO 0);
        s: IN std_logic_vector(1 DOWNTO 0);
        x: OUT std_logic_vector(7 DOWNTO 0));
END mux4to1;
ARCHITECTURE arch OF mux4to1 IS
BEGIN
        WITH s SELECT
                x <= a WHEN "00",
                        b WHEN "01",
                        c WHEN "10",
                        d WHEN OTHERS;
END arch;
```

~/Desktop/Ανάπτυξη Τηλεπικοινωνιακών Συστημάτων Σε Υλικό/test.vhd 0 0 0 16:10 LF UTF-8 VHDL

VHDL: Sequential Statements

- In order to use sequential statements we need to have a process written in our VHDL code. A process includes a number of sequential statements that define the behavior of a circuit.

- Processes run concurrently with the rest of the statements defined in the code.
  - You can consider any process as a black box.

VHDL: Sequential Statements (cont.)

- Sequential Statements (process):
  - wait
  - sequential signal assignment
  - if-else
  - case
  - for
  - ....

- Processes with sensitivity list vs. Processes without sensitivity

VHDL: Sequential Statements (cont.)

- **Process with sensitivity list:**
    - Syntax:

```
PROCESS (sensitivity list)
    declarations
    BEGIN
        sequential statements;
            ....
        sequential statements;
END PROCESS;
```

VHDL: Sequential Statements (cont.)

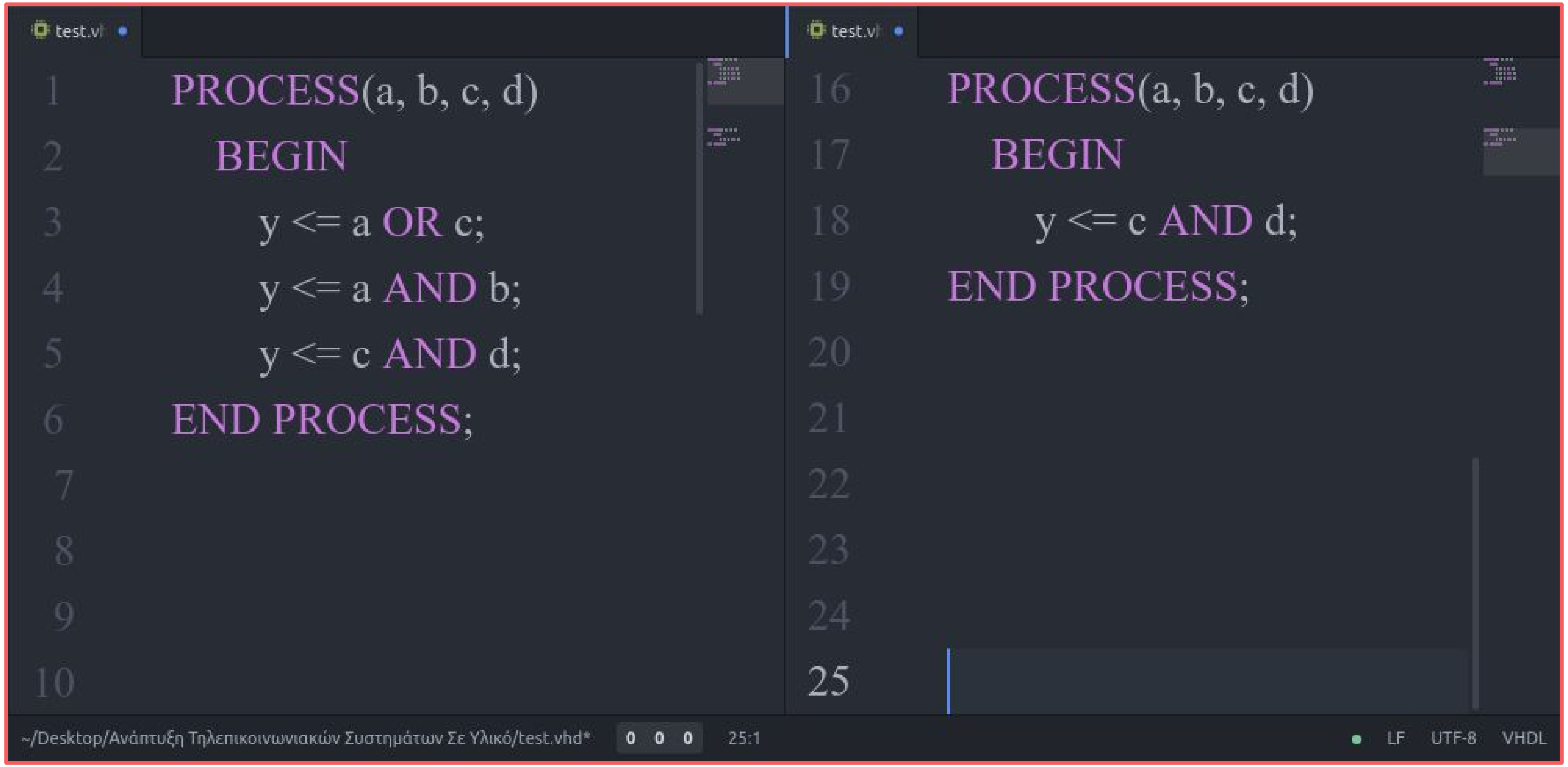
- **Process without sensitivity list:**
  - Syntax:

```
PROCESS (sensitivity list)

          declarations

          BEGIN

                    sequential statements;
                              ....
                    sequential statements;

                    WAIT ON/ WAIT UNTIL/ WAIT FOR statement

END PROCESS;
```

VHDL: Sequential Statements (cont.)

- **Process with sensitivity list:**
  - A process is either active or inactive (in a sense like functions in any programming language).
  - A process becomes active when a signal from the sensitivity list changes its value. It then becomes inactive until another change is reported.
  - When describing a sequential circuit, every input should be placed in the sensitivity list.
- **Process without sensitivity list:**
  - The process runs until a wait statement is reached, and then it becomes inactive.

VHDL: Sequential Statements (cont.)



test.vhd

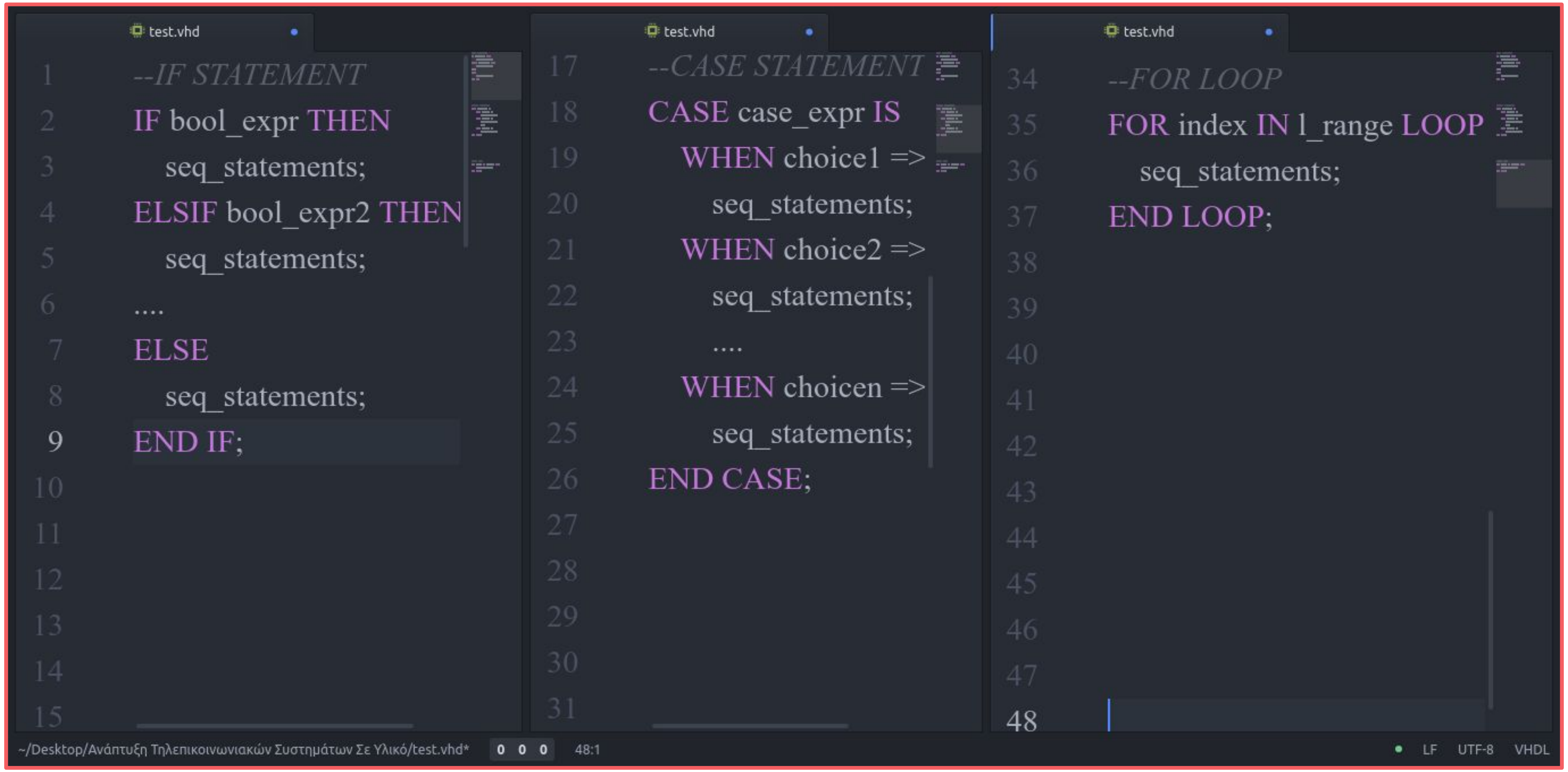
```
PROCESS(a, b, c, d)
    BEGIN
        y <= a OR c;
        y <= a AND b;
        y <= c AND d;
END PROCESS;
```

test.vhd

```
PROCESS(a, b, c, d)
    BEGIN
        y <= c AND d;
END PROCESS;
```

~/Desktop/Ανάπτυξη Τηλεπικοινωνιακών Συστημάτων Σε Υλικό/test.vhd*
0 0 0
25:1
LF
UTF-8
VHDL

VHDL: Sequential Statements (cont.)



```
--IF STATEMENT
IF bool_expr THEN
   seq_statements;
ELSIF bool_expr2 THEN
   seq_statements;
....
ELSE
   seq_statements;
END IF;
```

```
--CASE STATEMENT
CASE case_expr IS
   WHEN choice1 =>
      seq_statements;
   WHEN choice2 =>
      seq_statements;
      ....
   WHEN choicen =>
      seq_statements;
END CASE;
```

```
--FOR LOOP
FOR index IN l_range LOOP
   seq_statements;
END LOOP;
```

~/Desktop/Ανάπτυξη Τηλεπικοινωνιακών Συστημάτων Σε Υλικό/test.vhd* 0 0 0 48:1 LF UTF-8 VHDL

VHDL: Keywords

abs, configuration, impure, null, rem, type, access, in, of, report, unaffected, after, disconnect, if, inertial, return, units, alias, downto, inout, open, rol, all, else, is, or, ror, use, and, label, others, select, variable, end, library, on, severity, wait, array, entity, linkage, package, signal, assert, exit, literal, port, shared, while, attribute, loop, postponed, sla, out, with, begin, for, procedure, sll, xnor, block, function, mod, sra, xor, body, generate, nand, pure, srl, generic, new, range, subtype, bus, group, next, then, case, guarded, nor, register, to, when, not, reject, constant, transport, architecture, until, elsif, file, map, process, component, buffer, record