LD

21.02.2024

(laborator 1 - S1)

Sabina Cenda (sabina.cenda17@gmail.com)

Note:
- -> temă_lab
- -> test S7-8
- -> test S13

- maxim 2 laboratoare recuperate
- proiect pt mărire dacă e nevoie

Verilog HDL

```
1  module <nume_modul>(<tip_port_1>, <tip_port_2>, ...);
2
3      // se scrie direct numele, fara < >
4
5  endmodule
6
```

1) Numele modulului ar trebui să fie același cu numele fișierului.

2) Tipuri de porturi
- input
- output
- inout

3) Variabile locale
- wire
- reg

WIRE vs. REG

Sunt utilizate pentru legături între module și pentru asignarea de rezultate parțiale în circuite combinaționale

-> un element de tip WIRE își schimbă valoarea doar în blocuri de tip assign sau ca ieșire a unui modul.

-> un element de tip REG își schimbă valoarea doar în blocuri de tip always

-> nici WIRE, nici REG nu își pot schimba valoarea în mai mult de un bloc.

-> assign <-> combinaționale, reg <-> secvențiale

and_gate.v

```
// output trebuie pus prima data
// 'input a' == 'input [0:1]a'
// in mod normal, a va fi reprezentat pe un bit
module and_gate(output o, inpu  t a, input b);
    assign o=a&b; // or and(o,a,b);
    // where 'and' is a primitive in verilog
endmodule

```

and_gate_tb.v (tb = testbench)

```
module and_gate_tb;
    // inputs
    // folosim reg pt a asigna valori variabilelor
    reg a;
    reg b;

    // outputs
    // wire pt ca folosim always
    wire o;

    // 'main' este numele primul modul
    // 'uut' == unit under test
    // 'dut' == device under test
    // nu are importanta ce folosim aici
    and_gate uut(
        .o(o),
        .a(a),
        .b(b)
    );
    // .o referinta 'main' module 'o',
    // facem legatura cu o -ul declarat aici

    initial begin
        a = 0;
        b = 0;
        // asteptam 100 de nanosecunde sa fim
        //siguri ca operatiile s-au executat
        #100;
    end

    always begin
        #20 a = ~a;
        #30 b = ~b;
    end

endmodule

```

În fișierul run.txt:

```
# add all your source files to the sourcefiles list
# add the files separated by spaces
# Example:----------------------------------------

# set sourcefiles {mux_1s.v mux_2s.v mux2s_tb.v}
#adaugam toate fisierele verilog folosite cu spatiu intre ele

set sourcefiles {and_gate.v and_gate_tb.v}



# set name of the top module in variable topmodule
# Example:----------------------------------------


# set topmodule mux2s_tb
#adaugam numele modulului de instantiere
set topmodule and_gate_tb

##################################################
#####DO NOT MODIFY THE SCRIPT BELLOW THIS LINE#####
##################################################

# quit current simulation if any
quit -sim

# empty the work library if present
if [file exists "work"] {vdel -all}
#create a new work library
vlib work

# run the compiler
if [catch "eval vlog $sourcefiles"] {
    puts "correct the compilation errors"
    return
}

vsim -voptargs=+acc $topmodule
```

Pentru a simula:

1. do run.txt
2. add wave *
3. run -all

În fișierul pj.tcl :

-> vom avea de modificat numele fișierului + modulului

-> vom asigna led-uri și switch-uri pt variabile

↓ output ↓ input

```
project_new example1 -overwrite

set_global_assignment -name FAMILY MAX10
set_global_assignment -name DEVICE 10M50DAF484C7G

set_global_assignment -name BDF_FILE example1.bdf

//adaugam numele la fisierul verilog folosit

set_global_assignment -name VERILOG_FILE and_gate.v

set_global_assignment -name SDC_FILE example1.sdc

//adaugam numele la fisierul verilog folosit dar fara extensia '.v'

set_global_assignment -name TOP_LEVEL_ENTITY and_gate
set_location_assignment -to clk PIN_AH10

//aici scriem pinii pe care ii folosim pt switch-uri si led-uri

set_location_assignment PIN_C10 -to a   ;# SW[0]
set_location_assignment PIN_C11 -to b   ;# SW[1]

set_location_assignment PIN_A8  -to o  ;# LED[0]


load_package flow
execute_flow -compile

project_close

```