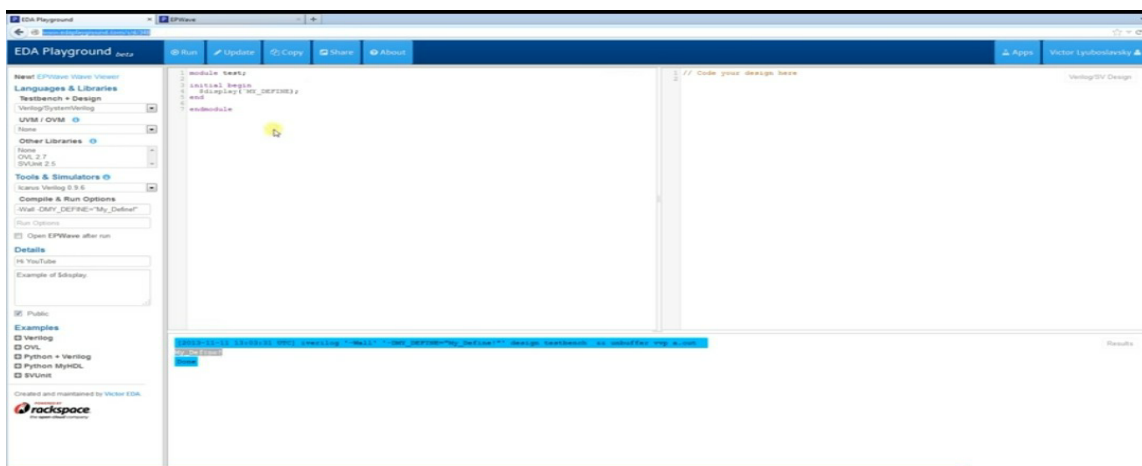


DAILY ASSESSMENT FORMAT

Date:	03-06-2020	Name:	K Muthu
Course:	DIGITAL DESIGN USING HDL	USN:	4a117ec038
Topic:	<ul style="list-style-type: none"> EDA Playground Online complier Task 	Semester & Section:	6 & 'A'
Github Repository:	K.Muthu-courses		

FORENOON SESSION DETAILS

Image of session



EDA Playground Introduction -- Simulate Verilog from a Web Browser

39K views · 6 years ago



111



1



Share



Download



Save



EDA Playground
5.12K subscribers

SUBSCRIBE

Report – Report can be typed or hand written for up to two pag

EDA Playground Online compiler :

The screenshot displays the EDA Playground Online compiler interface. The left sidebar contains navigation and configuration options:

- Languages & Libraries**
 - Testbench + Design
 - SystemVerilog/Verilog
 - UVM / OVM
 - UVM 1.2
 - Other Libraries
 - 0 selected
 - Tools & Simulators
 - Aldec Riviera Pro 2017.02
 - Compile & Run Options
 - timescale 1ns/1ns -sv2k9
 - +UVM_VERBOSITY=UVM_HIGH +a
 - Run Time: 10 ms
 - ☐ Use run.do Tcl file
 - ☐ Open EPWave after run
 - ☐ Download files after run
 - Examples
 - VHDL
 - Verilog/SystemVerilog
 - UVM
 - EasierUVM
 - SVAUnit
 - SVUnit
 - VUnit
 - TL-Verilog
 - e + Verilog
 - Python + Verilog
 - Python Only
 - C++/SystemC
 - Community
 - Collaborate
 - Forum
 - Follow @edaplayground

The main area is split into two panels:

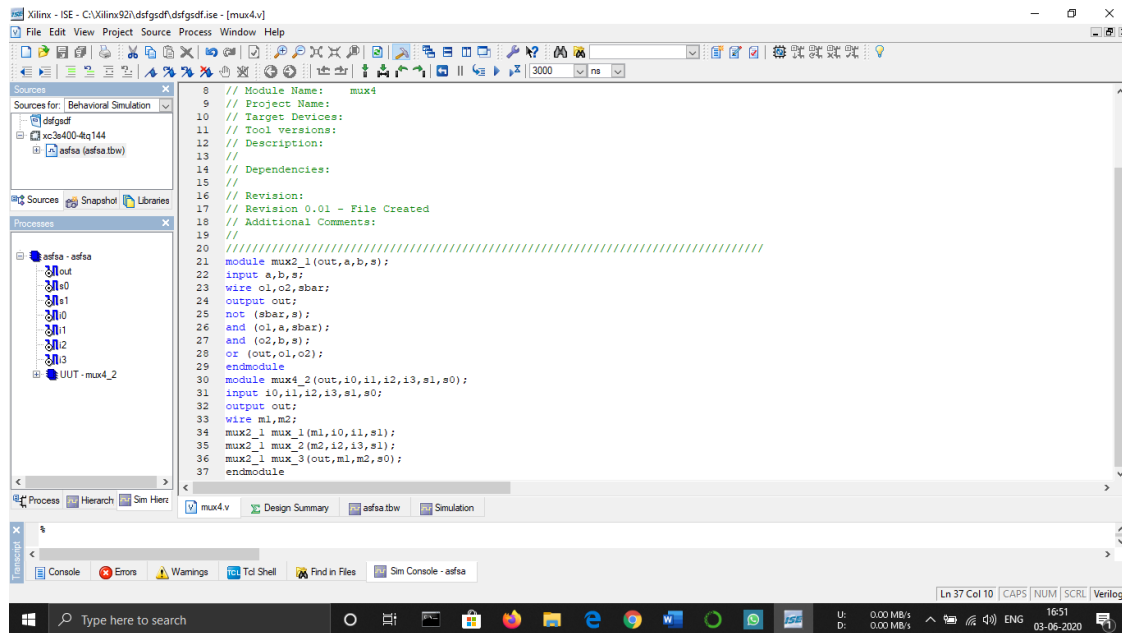
- testbench.sv** (SV/Verilog Testbench):

```
1 import uvm_pkg::*;
2 `include "uvm_macros.svh"
3
4 //-----
5 // environment env
6 //-----
7 class env extends uvm_env;
8
9     virtual add_sub_if m_if;
10
11     function new(string name, uvm_component
12         parent = null);
13         super.new(name, parent);
14     endfunction
15
16     function void connect_phase(uvm_phase
17         phase);
18         `uvm_info("LABEL", "Started connect
19             phase.", UVM_HIGH);
20         // Get the interface from the resource
21         database;
22         assert(uvm_resource_db#(virtual
23             add_sub_if)::read_by_name(
24                 get_full_name(), "add_sub_if", m_if));
25         `uvm_info("LABEL", "Finished connect
26             phase.", UVM_HIGH);
27         endfunction: connect_phase
28
29         task run_phase(uvm_phase phase);
30             phase.raise_objection(this);
31             `uvm_info("LABEL", "Started run phase.",
32                 UVM_HIGH);
33             begin
34                 int a = 8'h2, b = 8'h3;
35                 @(m_if.cb);
36                 m_if.cb.a <= a;
37                 m_if.cb.b <= b;
38                 m_if.cb.doAdd <= 1'b1;
39                 repeat(2) @(m_if.cb);
40                 `uvm_info("RESULT", $sprintf("%0d +
41                     %0d = %0d",
42                         a, b, m_if.cb.result), UVM_LOW);
43             end
44             `uvm_info("LABEL", "Finished run phase.",
45                 UVM_HIGH);
46             phase.drop_objection(this);
47         endtask: run_phase
48     endclass
49
50 //-----
51 // module top
52 //-----
53 module top;
54
55     bit clk;
56     env environment;
57     ADD_SUB dut(.clk (clk));
58
59     initial begin
60         environment = new("env");
61         // Put the interface into the resource
62         database;
63         uvm_resource_db#(virtual
64             add_sub_if)::set("env",
65                 "add_sub_if", dut.add_sub_if);
66     end
```
- design.sv** (SV/Verilog Design):

```
1 // Simple adder/subtractor mo:SV/Verilog Design
2 module ADD_SUB(
3     input clk,
4     input [7:0] a0,
5     input [7:0] b0,
6     // if this is 1, add; else subtract
7     input doAdd0,
8     output reg [8:0] result0
9 );
10
11 always @ (posedge clk)
12     begin
13         if (doAdd0)
14             result0 <= a0 + b0;
15         else
16             result0 <= a0 - b0;
17     end
18 endmodule: ADD_SUB
19
20 //-----
21 // Interface for the adder/subtractor DUT
22 //-----
23 interface add_sub_if(
24     input bit clk,
25     input [7:0] a,
26     input [7:0] b,
27     input doAdd,
28     input [8:0] result
29 );
30
31     clocking cb @(posedge clk);
32     output a;
33     output b;
34     output doAdd;
35     input result;
36     endclocking // cb
37 endinterface: add_sub_if
38
39 //-----
40 // Interface bind
41 //-----
42 bind ADD_SUB add_sub_if add_sub_if0(
43     .clk(clk),
44     .a(a0),
45     .b(b0),
46     .doAdd(doAdd0),
47     .result(result0)
48 );
49
50
51
```

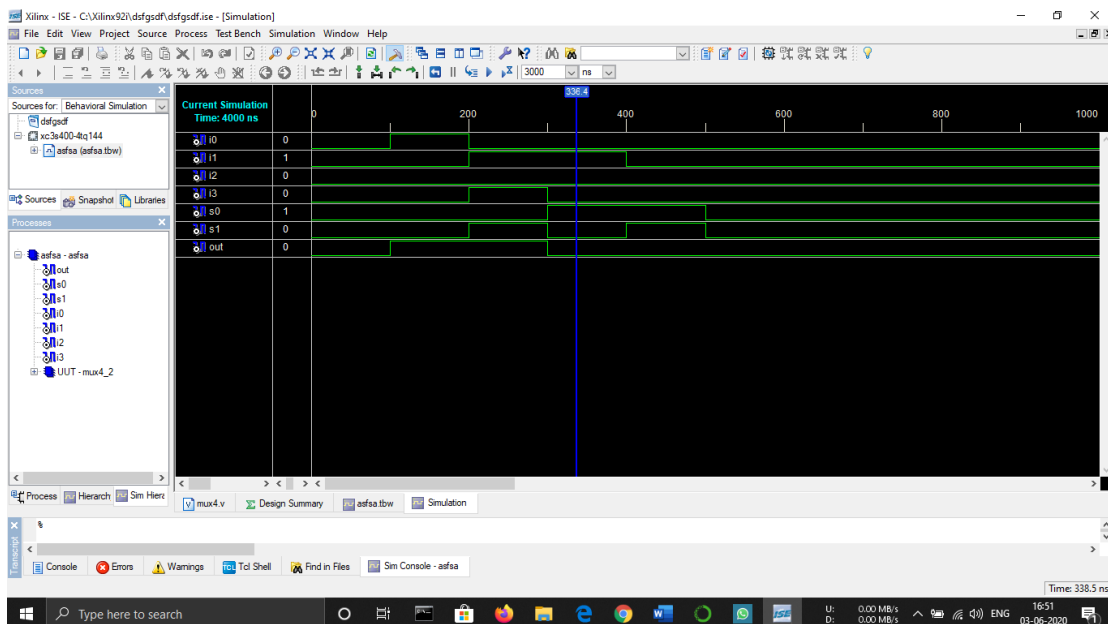
Task : Implement 4 to 1 MUX using two 2 to 1 MUX using structural modelling style and test the module in online/offline compiler.

Verilog code :



```
8 // Module Name: mux4
9 // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21 module mux2_1(out,a,b,s):
22 input a,b,s;
23 wire o1,o2,sbar;
24 output out;
25 not (sbar,s);
26 and (o1,a,sbar);
27 and (o2,b,s);
28 or (out,o1,o2);
29 endmodule
30 module mux4_2(out,i0,i1,i2,i3,s1,s0):
31 input i0,i1,i2,i3,s1,s0;
32 output out;
33 wire m1,m2;
34 mux2_1 mux_1(m1,i0,i1,s1);
35 mux2_1 mux_2(m2,i2,i3,s1);
36 mux2_1 mux_3(out,m1,m2,s0);
37 endmodule
```

Output :



Date: 03-06-2020

Name: K Muthu

Course: Python Bootcamp 2020 build 15
working applications and Games

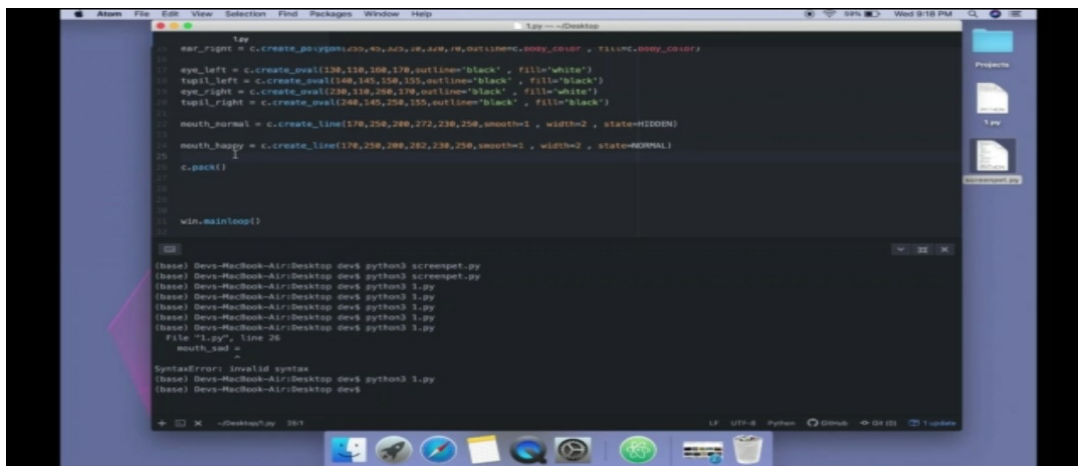
USN: 4a17ec038

Topic: Screen pet

Semester 6 & 'A'
& Section:

AFTERNOON SESSION DETAILS

Image of session



Lectures

More



Section 36 - Project-10 Screen pet



334 Introduction to this module

Video - 00:35 mins



335 Overview of project

Video - 01:32 mins



336 Creating body, eyes and ears

Video - 11:46 mins



337 Creating mouth tongue and cheeks



Report – Report can be typed or hand written for up to two pages.

Screen pet :

- Screen pet is programmed using Tkinter module package of python library.
- Tkinter is the standard GUI library for Python.
- Python when combined with Tkinter provides a fast and easy way to create GUI applications.
- Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.
- Creating a GUI application using Tkinter is an easy task.
- Steps involved in creating are,
 - ✓ Import the Tkinter module.
 - ✓ Create the GUI application main window.
 - ✓ Add the required widgets to the GUI application.
 - ✓ Enter the main event loop to take action against each event triggered by the user.

