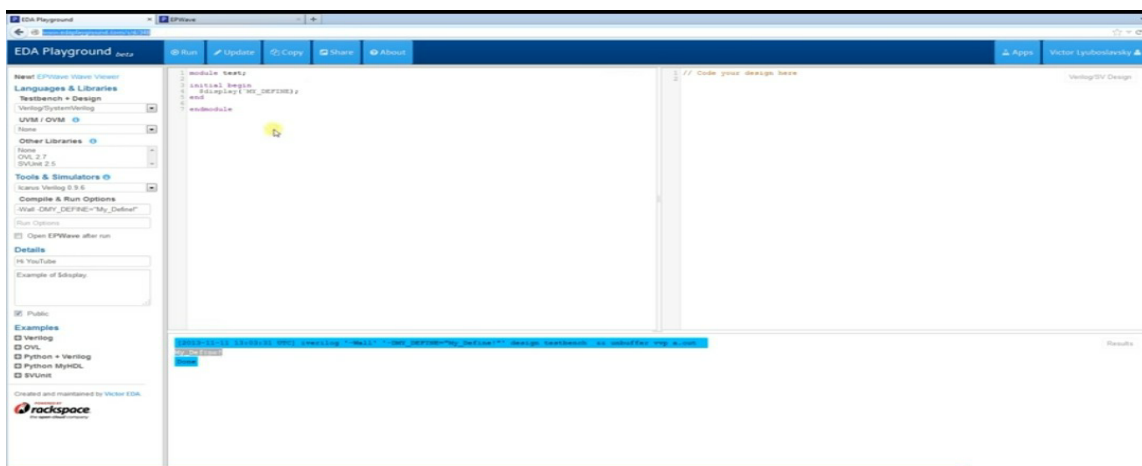


DAILY ASSESSMENT FORMAT

Date:	03-06-2020	Name:	Abhishek
Course:	DIGITAL DESIGN USING HDL	USN:	4a17ec001
Topic:	<ul style="list-style-type: none"> EDA Playground Online complier Task 	Semester & Section:	6 & 'A'
Github Repository:	Abhishek-online-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

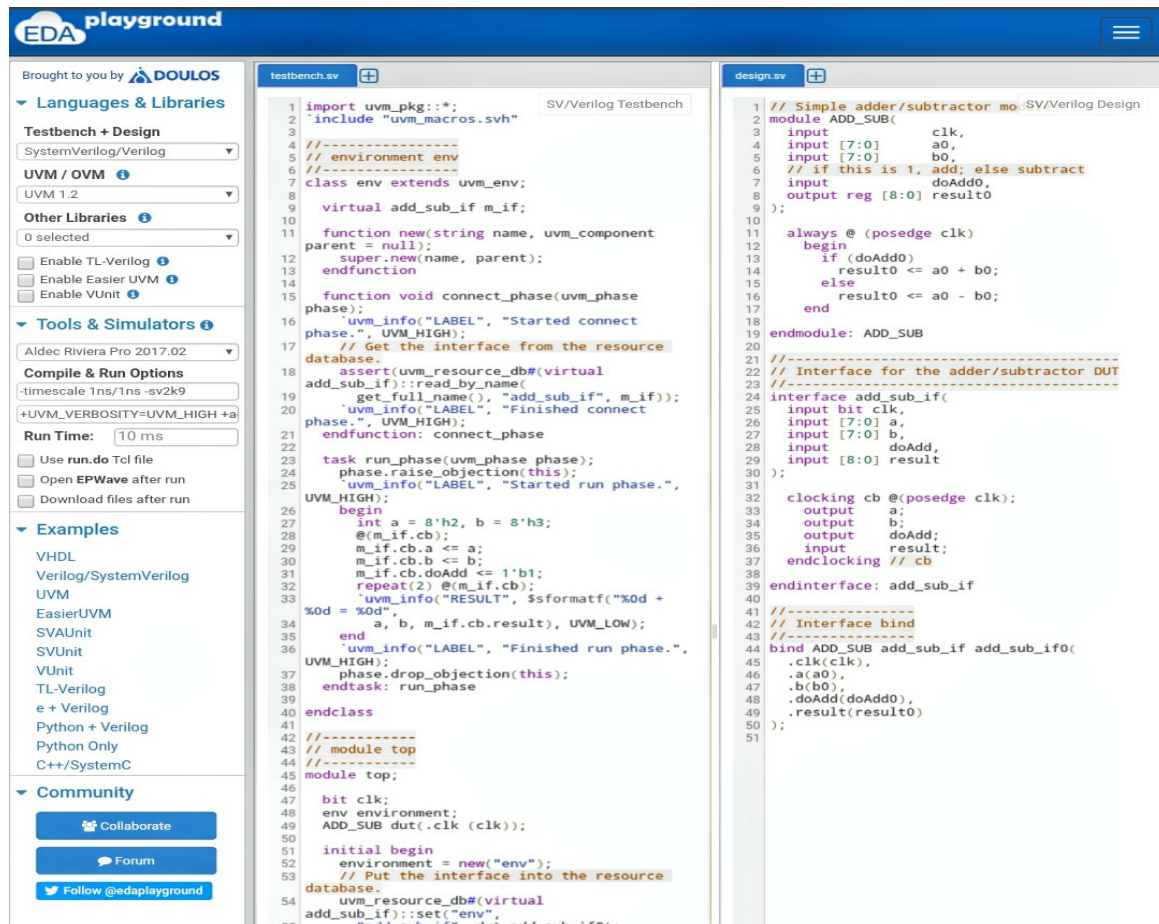


EDA Playground
5.12K subscribers

SUBSCRIBE

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

EDA Playground Online compiler :



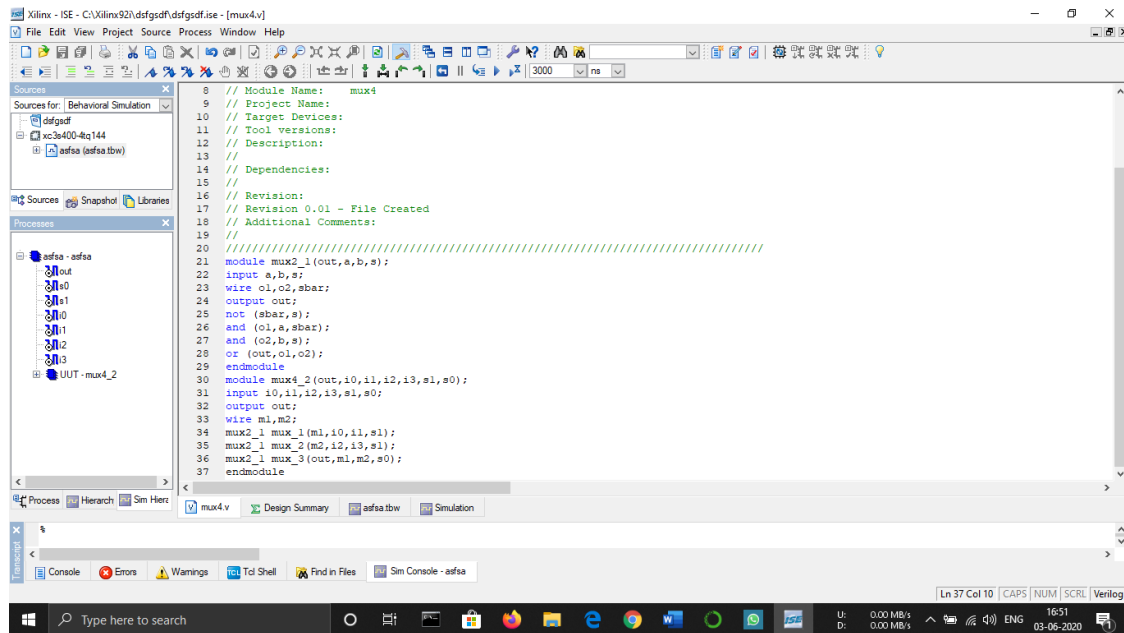
The screenshot displays the EDA Playground Online compiler interface. On the left, there is a sidebar with navigation options: Languages & Libraries, Tools & Simulators, Compile & Run Options, Examples, and Community. The main area is split into two panels: 'testbench.sv' and 'design.sv'. The 'testbench.sv' panel contains a UVM testbench for an adder/subtractor, including environment setup, phase definitions, and a top module instantiation. The 'design.sv' panel contains the implementation of the 'ADD_SUB' module, which is a simple adder/subtractor with 8-bit inputs and an 8-bit output. The code is written in Verilog and includes comments in English.

```
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", $sformatf("%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
67 endmodule
```

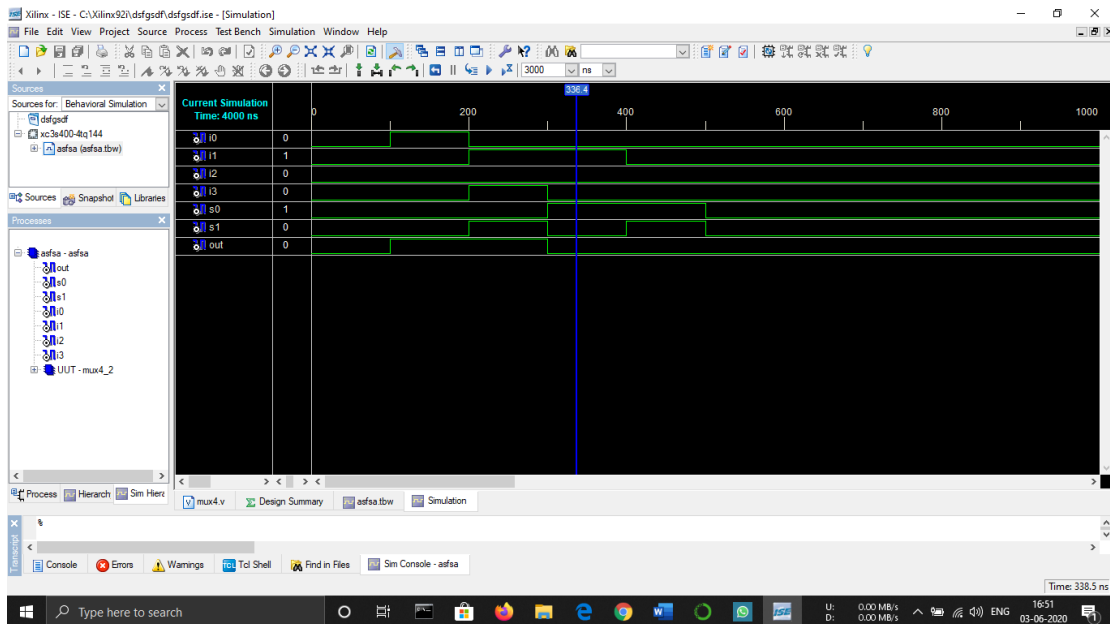
```
1 // Simple adder/subtractor module
2 module ADD_SUB(
3     input clk,
4     input [7:0] a0,
5     input [7:0] b0,
6     // if this is 1, add; else subtract
7     input doAdd0,
8     output reg [8:0] result0
9 );
10
11 always @ (posedge clk)
12 begin
13     if (doAdd0)
14         result0 <= a0 + 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 :



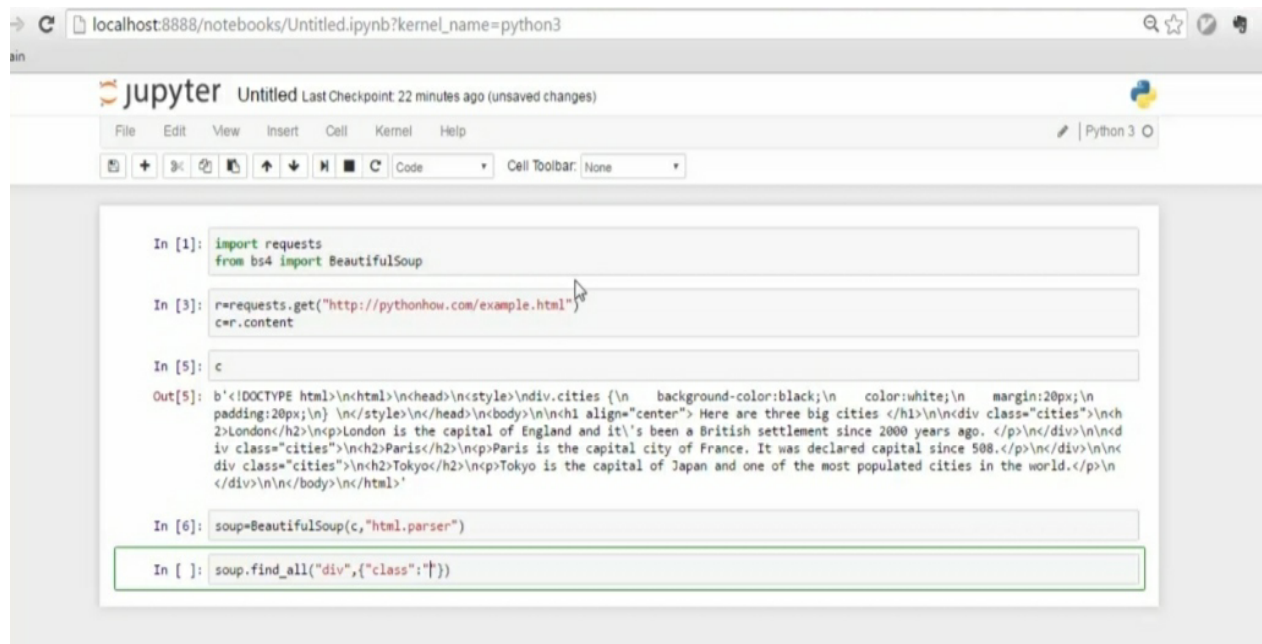
Output :



Date:	03-06-2020	Name:	Abhishek
Course:	The Python Mega Course: Build 10 Real World Applications	USN:	4al17ec001
Topic:	1] Web scraping with Python BeautifulSoup 2] Application 7: Scrape Real Estate Property Data from the Web	Semester & Section:	6 & 'A'

AFTERNOON SESSION DETAILS

Image of session



The screenshot shows a Jupyter Notebook interface in a web browser. The address bar indicates the URL is localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3. The Jupyter Notebook title bar shows 'Untitled' and 'Last Checkpoint 22 minutes ago (unsaved changes)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, and Help. The toolbar shows various icons for file operations and a dropdown menu set to 'Code'. The main area displays the following code and output:

```
In [1]: import requests
        from bs4 import BeautifulSoup

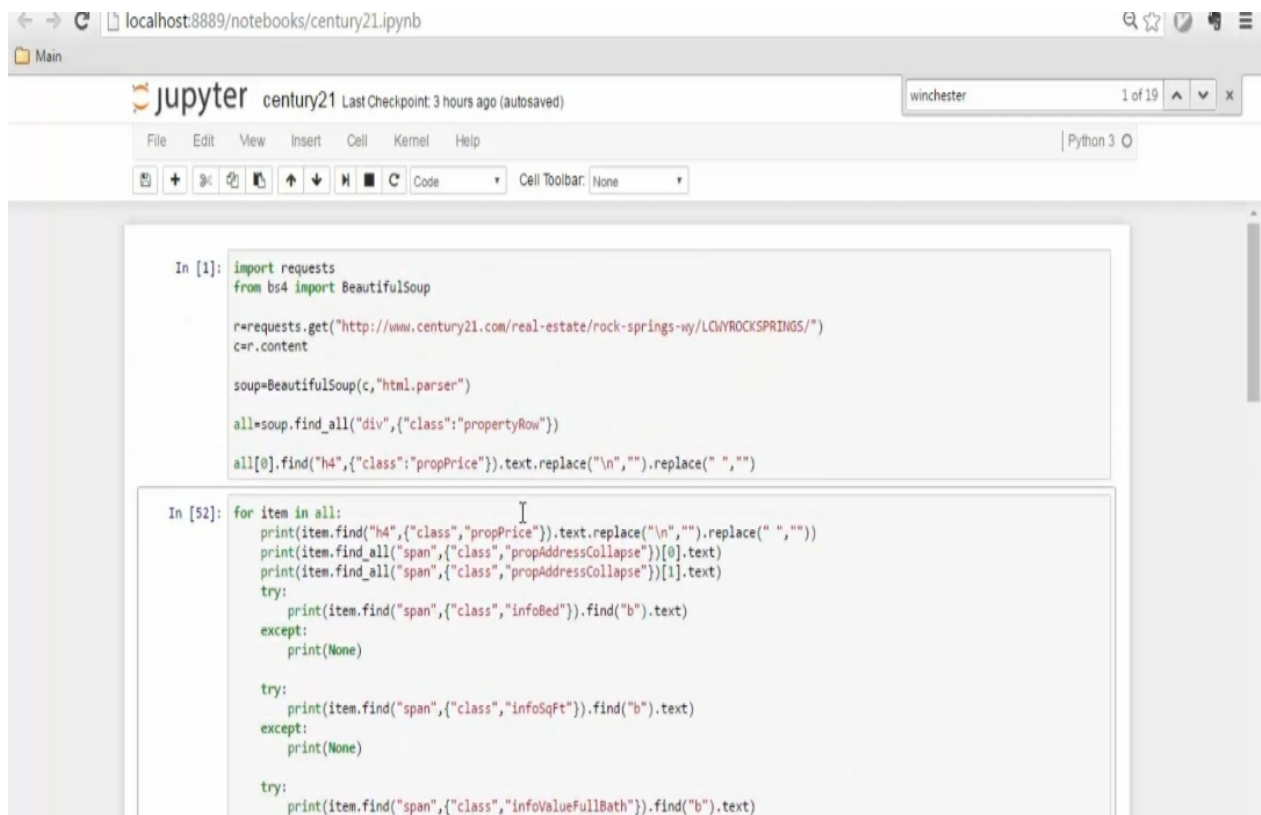
In [3]: r=requests.get("http://pythonhow.com/example.html")
        c=r.content

In [5]: c

Out[5]: b'<DOCTYPE html>\n<html>\n<head>\n<style>\n<div>.cities {\n  background-color:black;\n  color:white;\n  margin:20px;\n  padding:20px;\n} \n</style>\n</head>\n<body>\n<n<h1 align="center"> Here are three big cities </h1>\n<n<div class="cities">\n<n<h2>London</h2>\n<p>London is the capital of England and it\'s been a British settlement since 2000 years ago. </p>\n</div>\n<n<div class="cities">\n<n<h2>Paris</h2>\n<p>Paris is the capital city of France. It was declared capital since 508.</p>\n</div>\n<n<div class="cities">\n<n<h2>Tokyo</h2>\n<p>Tokyo is the capital of Japan and one of the most populated cities in the world.</p>\n</div>\n<n</body>\n<n</html>'
```

```
In [6]: soup=BeautifulSoup(c,"html.parser")

In [ ]: soup.find_all("div",{"class":"cities"})
```



localhost:8889/notebooks/century21.ipynb

Main

jupyter century21 Last Checkpoint: 3 hours ago (autosaved) winchester 1 of 19

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

```
In [1]: import requests
from bs4 import BeautifulSoup

r=requests.get("http://www.century21.com/real-estate/rock-springs-wy/LCwYROCKSPRINGS/")
c=r.content

soup=BeautifulSoup(c,"html.parser")

all=soup.find_all("div",{"class":"propertyRow"})

all[0].find("h4",{"class":"propPrice")).text.replace("\n","").replace(" ","")

In [52]: for item in all:
    print(item.find("h4",{"class","propPrice")).text.replace("\n","").replace(" ",""))
    print(item.find_all("span",{"class","propAddressCollapse"})[0].text)
    print(item.find_all("span",{"class","propAddressCollapse"})[1].text)
    try:
        print(item.find("span",{"class","infoBed"}).find("b").text)
    except:
        print(None)

    try:
        print(item.find("span",{"class","infoSqft"}).find("b").text)
    except:
        print(None)

    try:
        print(item.find("span",{"class","infoValueFullBath"}).find("b").text)
```

Report .

Web scraping with Python BeautifulSoup

- Introduction to web scraping using **BeautifulSoup** from **bs4** library and **requests** module.
- **BeautifulSoup** is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.
- The **requests** module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc.).
- **Web scraping** is a term used to describe the use of a program or algorithm to extract and process large amounts of data from the web. Whether you are a data scientist, engineer, or anybody who analyzes large amounts of datasets, the ability to scrape data from the web is a useful skill to have. Let's say you find data from the web, and there is no direct way to download it, web scraping using Python is a skill you can use to extract the data into a useful form that can be imported.

Application 7: Scrape Real Estate Property Data from the Web

- Python program to extract details of plots from century21 website and store the details in a **csv file** using **pandas** library.
- Some of the functions used under BeautifulSoup of bs4 library:
- The **find_all ()** method scans the entire document looking for results, but sometimes you only want to find one result. If you know a document only has one <body> tag, it's a waste of time to scan the entire document looking for more. Rather than passing in limit = 1 every time you call find_all, you can use the **find ()** method.
- The **prettify ()** method will turn a BeautifulSoup parse tree into a nicely formatted Unicode string, with a separate line for each tag and each string.
- To extract text from the body tag **.text** method is used.

