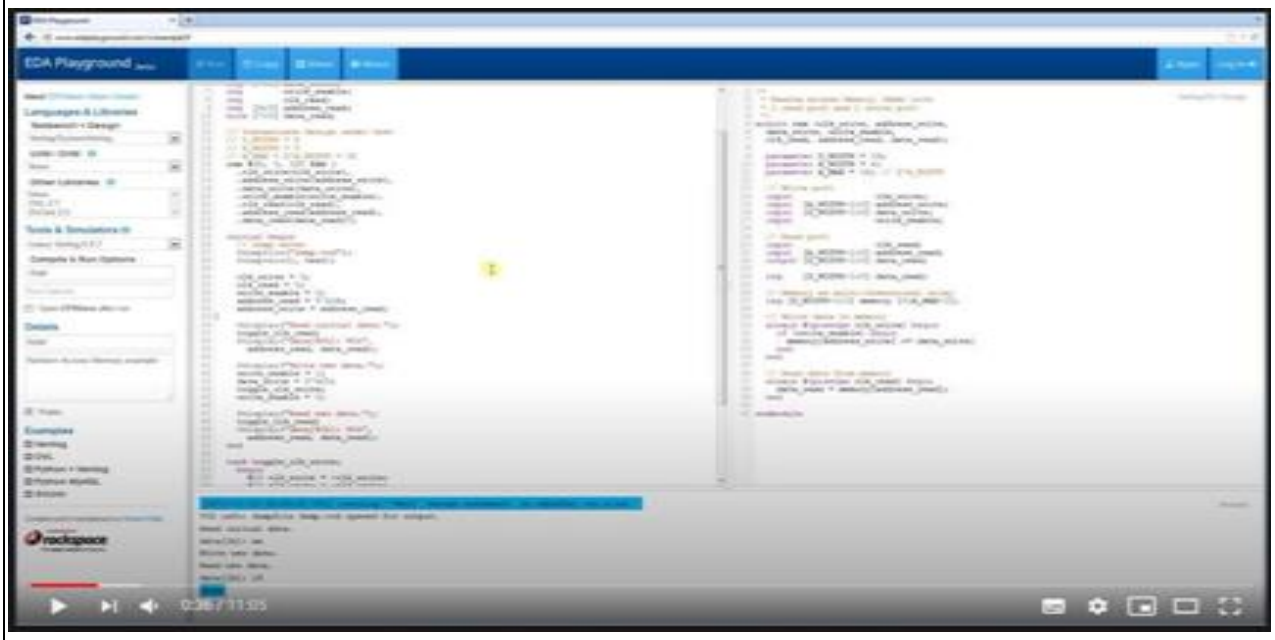
DAILY ASSESSMENT REPORT

Date:	**03 /6/2020**	**Name:**	**Chandana. R**
Course:	**DIGITAL DESIGN USING HDL**	**USN:**	**4AL16EC017**
Topic:	EDA Playground Online complier, EDA Playground Tutorial Demo Video, How to Download And Install Xilinx Vivado Design Suite, Vivado Design Suite for implementation of HDLcode	**Semester & Section:**	**8(A)**
Github Repository	**chandana-shaiva**		

FORENOON SESSION DETAILS

Image of session



Report:

- Vivado Design Suite for implementation of HDL code
- In a separate web browser window, log in to EDA Playground
- Log in Click the Log in button (top right) Then eitherclick on Google or Facebook or register by clicking on 'Register for a full account' (which enables all the simulators on EDA Playground)

- Select 'Aldec Riviera Pro' from the Tools & Simulators menu.
- This selects the Aldec Riviera Pro simulator, which can be used however you logged in. Using certain other simulators will require you to have registered for a full account.

- In either theDesignorTestbenchwindow pane, type in the following code:
- Module test;
- Initial
- $display("Hello World!");
- End module
- Click Run(top left)Yes, running a simulation is as simple as that!
- In the bottom pane, you should see realtime results as your code is being compiled and then run. A run typically takes 5seconds, depending on network traffic and simulator. Near the bottom of result output, you should see:
- Hello World!
- Now, let's save our good work. Click theSharetab near in the bottom pane and then type in a name and description. Then click Save
- The browser page will reload and the browser address bar will change. This is a persistent link to your saved code. You can send the link by email, post it on a web page, post it onStack Overflowforums, etc.
- Now, let's try modifying existing code. Load the following example: RAM
- On the left editor pane, before then do finite al block, add the following:
  - ✓ write_enable = 1;
  - ✓ data_write = 8'h2C;
  - ✓ toggle_clk_write;
  - ✓ toggle_clk_read;
  - ✓ $display("data[%0h]: %0h",
  - ✓ address_read, data_read);
    - Run the sim. In the results you should see this new message:
    - data[1b]: 2c
    - Optional. Click Copy to save a personal version of the modified RAM code, including the simulation results.

➢ Loading Waves from EDA Playground
➢ You can run a simulation on EDA
➢ Playground and load the resulting waves in EPWave. Loading Waves for System Verilog and Verilog Simulations
➢ Go to your code on EDA Playground. For example: RAM Design and Test Make sure your code contains appropriate function calls to create a *.vcd file.

✓ For example:

```
initial
begin
$dumpfile("dump.vcd");
$dumpvars(1);
End
```

Select a simulator and check the Open EPWave after run check box.

- Click Run.

- After the run completes, the resulting waves will load in a new EPWave window.
- Loading Waves for VHDL Simulations check the Open EPWave after run checkbox.
- Specify the top entity to simulate.
- Click Run

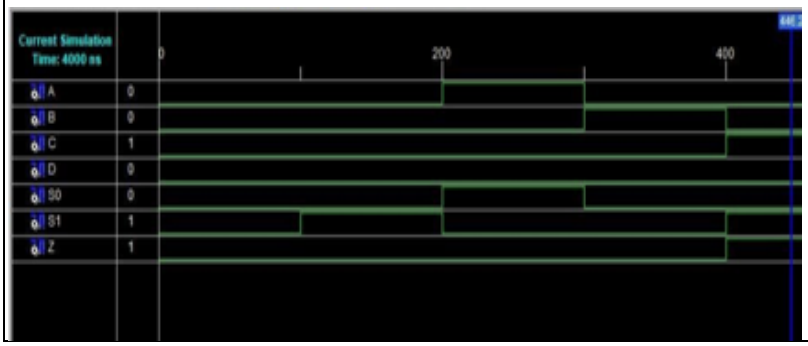
After the run completes, the resulting waves will load in a new EPWave window. (Popups must be enabled.)The waves for all signals in the specified top entity and any of its components will be dumped. In EPWave window, click get Signals to select the signals to view.

TASK:

Implement 4 to 1 MUX using two 2 to 1 MUX using structural modeling style and test the module in online/offline compiler

```
module mux4to1(a,sel,out);
input [3:0] a;
input [1:0] sel;
output out;
        wire mux[2:0];
mux2to1 m1 (a[3],a[2],sel[0],mux_1),
      m2 (a[1],a[4],sel[0],mux_2),
      m3 (mux_1,mux_2,sel[1],out);
endmodule
```

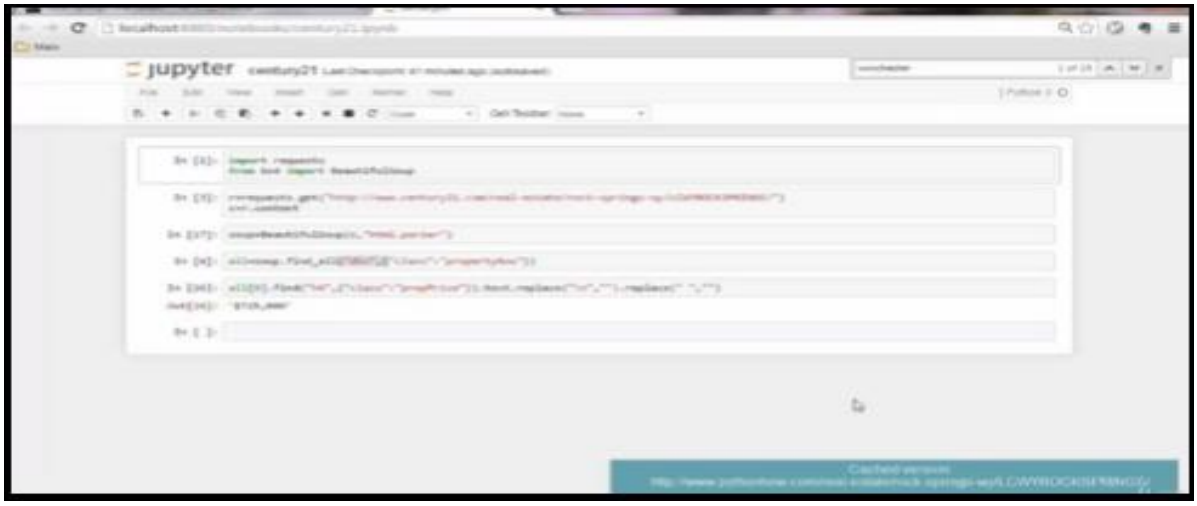
Output:



Date:	3/6/2020	Name:	Chandana.R
Course:	The Python Mega Course	USN:	4AL16EC017
Topic:	Application 8: Scrape Real Estate Property Data from the Web	Semester &Section:	8(A)

AFTERNOON SESSION DETAILS

Image of session:



Report:

Scrape Real Estate Property Data from the Web:

- In this application we learnt how to collect data from various websites using python.
- We learnt about loading the web pages in python.
- We learnt to extract the “div”tags.
- We learnt about extracting addresses and property details.
- We learnt about extracting elements without unique identifiers.
- We learnt how to save the obtained data in .csvformat.
- Andalsowelearnttoextractdatafromvariouswebsitesatatimeusingcrawlingthrough websites using python.

The output obtained from website is saved in excel sheet as shown below

WEBSCRAPING USING BEAUTIFUL SOUP

Beautiful Soup is a Python library designed for quick turnaround projects like screen scraping. Three features make it powerful:

Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application. Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application:

Beautiful Soup sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility

CODE –

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import urllib.request
import urllib.parse
import urllib.error
from bs4 import Beautiful Soup
import ssl
import json
import ast
import os
from urllib.request import Request, urlopen

# For ignoring SSL certificate errors

ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

# Input from user

url = input('Enter Trulia Product Url- ')

# Making the website believe that you are accessing it using a mozilla browser

req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
webpage = urlopen(req).read()

# Creating a BeautifulSoup object of the html page for easy extraction of data.

soup = BeautifulSoup(webpage, 'html.parser')
```

```
html = soup.prettify('utf-8')
product_json = {}

# This code block will get you a one liner description of the listed property

for meta in soup.findAll('meta', attrs={'name': 'description'}):
    try:
        product_json['description'] = meta['content']
        break
    except:
        pass

# This code block will get you the link of the listed property

for link in soup.findAll('link', attrs={'rel': 'canonical'}):
    try:
        product_json['link'] = link['href']
        break
    except:
        pass

# This code block will get you the price and the currency of the listed property

for scripts in soup.findAll('script',
                            attrs={'type': 'application/ld+json'}):
    details_json = ast.literal_eval(scripts.text.strip())
product_json['price'] = {}
product_json['price']['amount'] = details_json['offers']['price']
product_json['price']['currency'] = details_json['offers'
        ]['priceCurrency']

# This code block will get you the detailed description of the the listed property

for paragraph in soup.findAll('p', attrs={'id': 'propertyDescription'}):
    product_json['broad-description'] = paragraph.text.strip()
product_json['overview'] = []

# This code block will get you the important points regarding the listed property

for divs in soup.findAll('div',
                         attrs={'data-auto-test-id': 'home-details-overview'
                         }):
    for divs_second in divs.findAll('div'):
        for uls in divs_second.findAll('ul'):
            for lis in uls.findAll('li', text=True, recursive=False):
                product_json['overview'].append(lis.text.strip())

# Creates a json file with all the information that you extracted

with open('house_details.json', 'w') as outfile:
    json.dump(product_json, outfile, indent=4)
```

```
# Creates an html file in your local with the html content of the page you parsed.
with open('output_file.html', 'wb') as file:
    file.write(html)
print ('----------Extraction of data is complete. Check json file.----------')
```