**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **03/06/2020** | **Name:** | **Varshini MN** |
| **Course:** | **HDL design** | **USN:** | **4AL16EC089** |
| **Topic:** | **How to Download and Install Xilinx Vivado Design Suite**  **Vivado Design Suite for implementation of HDL code** | **Semester & Section:** | **8th B`** |
| **Github Repository:** | **varshinimn-test** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  First download Vivado (or at least the web installer for it) from Xilinx’s website, at <http://www.xilinx.com/support/download.html>.  Once you get to the download page, choose the appropriate installer for your system; I’m on a Windows 10 machine and don’t feel the need to get a universal, all OS installer, so I’ll choose the Windows Self Extracting Web Installer. **VHDL Code for Full Adder**  |  | | --- | | library IEEE; | | use IEEE.STD\_LOGIC\_1164.ALL; | |  |  | | --- | |  | | entity full\_adder\_vhdl\_code is | |  |  |  | | --- | --- | | Port ( A : in STD\_LOGIC; | | | B : in STD\_LOGIC; |  |  |  | | --- | --- | | Cin : in STD\_LOGIC; | | | S : out STD\_LOGIC; |  |  | | --- | | Cout : out STD\_LOGIC); | | end full\_adder\_vhdl\_code; | |  |  | | --- | |  | | architecture gate\_level of full\_adder\_vhdl\_code is | |  |  | | --- | |  | | begin | |  |  | | --- | |  | | S <= A XOR B XOR Cin ; | |  |  |  | | --- | --- | | Cout <= (A AND B) OR (Cin AND A) OR (Cin AND B) ; | | |  |  |  | | --- | | end gate\_level; | |  |  **Testbench VHDL Code for Full Adder**  |  | | --- | | LIBRARY ieee; | | USE ieee.std\_logic\_1164.ALL; | |  |  | | --- | |  | | ENTITY Testbench\_full\_adder IS | |  |  |  | | --- | --- | | END Testbench\_full\_adder; | | |  |  |  |  | | --- | --- | | ARCHITECTURE behavior OF Testbench\_full\_adder IS | | |  |  |  |  | | --- | --- | | -- Component Declaration for the Unit Under Test (UUT) | | |  |  |  |  | | --- | --- | | COMPONENT full\_adder\_vhdl\_code | | | PORT( |  |  | | --- | | A : IN std\_logic; | | B : IN std\_logic; |  |  |  | | --- | --- | | Cin : IN std\_logic; | | | S : OUT std\_logic; |  |  |  | | --- | --- | | Cout : OUT std\_logic | | | ); |  |  |  | | --- | --- | | END COMPONENT; | | |  |  |  | | --- | | --Inputs | | signal A : std\_logic := '0'; | |  |  | | --- | | signal B : std\_logic := '0'; | | signal Cin : std\_logic := '0'; | |  |  | | --- | |  | | --Outputs | |  |  | | --- | | signal S : std\_logic; | | signal Cout : std\_logic; | |  |  | | --- | |  | | BEGIN | |  |  | | --- | |  | | -- Instantiate the Unit Under Test (UUT) | |  |  |  | | --- | --- | | uut: full\_adder\_vhdl\_code PORT MAP ( | | | A => A, |  |  | | --- | | B => B, | | Cin => Cin, | |  |  | | --- | | S => S, | | Cout => Cout | |  |  | | --- | | ); |  **Output Waveform for full adder VHDL Code** [Testbench Waveform for full adder VHDL Code](http://allaboutfpga.com/wp-content/uploads/2014/04/Testbench-Waveform-for-full-adder-VHDL-Code.png) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **03/06/2020** | **Name:** | **Varshini MN** | |
| **Course:** | **Udemy** | **USN:** | **4AL16EC089** | |
| **Topic:** | **Python** | **Semester & Section:** | **8th B** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **REPORT**  The large property listing companies target an entire nation and work on millions of properties. But in case you are a real estate agent, or if you are setting up shop and targeting a specific state or region, it is better that instead of trying to gather data yourself, you scrape it from a major real estate listing website.  #!/usr/bin/python  # -\*- coding: utf-8 -\*-  import urllib.request  import urllib.parse  import urllib.error  from bs4 import BeautifulSoup  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 Zillow House Listing 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')  property\_json = {}  property\_json['Details\_Broad'] = {}  property\_json['Address'] = {}  # Extract Title of the property listing  for title in soup.findAll('title'):  property\_json['Title'] = title.text.strip()  break  for meta in soup.findAll('meta', attrs={'name': 'description'}):  property\_json['Detail\_Short'] = meta['content'].strip()  for div in soup.findAll('div', attrs={'class': 'character-count-truncated'}):  property\_json['Details\_Broad']['Description'] = div.text.strip()  for (i, script) in enumerate(soup.findAll('script',  attrs={'type': 'application/ld+json'})):  if i == 0:  json\_data = json.loads(script.text)  property\_json['Details\_Broad']['Number of Rooms'] = json\_data['numberOfRooms']  property\_json['Details\_Broad']['Floor Size (in sqft)'] = json\_data['floorSize']['value']  property\_json['Address']['Street'] = json\_data['address']['streetAddress']  property\_json['Address']['Locality'] = json\_data['address']['addressLocality']  property\_json['Address']['Region'] = json\_data['address']['addressRegion']  property\_json['Address']['Postal Code'] = json\_data['address']['postalCode']  if i == 1:  json\_data = json.loads(script.text)  property\_json['Price in $'] = json\_data['offers']['price']  property\_json['Image'] = json\_data['image']  break  with open('data.json', 'w') as outfile:  json.dump(property\_json, outfile, indent=4)  with open('output\_file.html', 'wb') as file:  file.write(html)  print ('----------Extraction of data is complete. Check json file.----------') | | | |