**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **02/06/2020** | | **Name:** | **Abhishek Vasudev Mahendrakar** | | |
| **Course:** | **Digital Design using HDL** | | **USN:** | **4AL17EC003** | | |
| **Topic:** | |  |  | | --- | --- | | 1. **FPGA Basics: Architecture, Applications and Uses**  |  | | --- | | 1. **Verilog HDL Basics by Intel** | | | 1. **Verilog Test bench code to verify the design under test (DUT)** | | | **Semester & Section:** | **6th-‘A’** | | |
| **Github Repository:** | **ECEAbhishekVMahendrakar** | | **E-mail:** | **abhi2244mahendrakar@gmail.com** | | |
| **FORENOON SESSION DETAILS** | | | | | |
| **Image of session** | | | | | |
| **Report – Report can be typed or hand written for up to two pages.**   1. **FPGA Basics: Architecture, Applications and Uses:**  * A basic FPGA architecture (Figure 1) consists of thousands of fundamental elements called configurable logic blocks (CLBs) surrounded by a system of programmable interconnects, called a fabric, that routes signals between CLBs. Input/output (I/O) blocks interface between the FPGA and external devices. * Depending on the manufacturer, the CLB may also be referred to as a logic block (LB), a logic element (LE) or a logic cell (LC).     **Application:**   * Many applications rely on the parallel execution of identical operations; the ability to configure the FPGA’s CLBs into hundreds or thousands of identical processing blocks has applications in image processing, artificial intelligence (AI), data center hardware accelerators, enterprise networking and automotive advanced driver assistance systems (ADAS). * Many of these application areas are changing very quickly as requirements evolve and new protocols and standards are adopted. FPGAs enable manufacturers to implement systems that can be updated when necessary. * A good example of FPGA use is high-speed search: Microsoft is using FPGAs in its data centres to run Bing search algorithms. The FPGA can change to support new algorithms as they are created. If needs change, the design can be repurposed to run simulation or modelling routines in an HPC application. This flexibility is difficult or impossible to achieve with an ASIC. * Other FPGA uses include aerospace and defense, medical electronics, digital television, consumer electronics, industrial motor control, scientific instruments, cybersecurity systems and wireless communications.  1. **Verilog HDL Basics by Intel:**  * Verilog is a HARDWARE DESCRIPTION LANGUAGE (HDL). It is a language used for describing a digital system like a network switch or a microprocessor or a memory or a flip−flop. * It means, by using a HDL we can describe any digital hardware at any level. Designs, which are described in HDL are independent of technology, very easy for designing and debugging, and are normally more useful than schematics, particularly for large circuits. * Behavioral level * Register-transfer level * Gate level * Lexical Tokens * Numbers * Identifiers * Operators * Data Types * Operators * Operands * Modules  1. **Verilog Test bench code to verify the design under test (DUT):**     **Verilog design:**  **module** boredF(**input** a, b, c,  **output** y);  **assign** y = ~b & ~c | a & ~b;  **endmodule**  **Testbench code:**  **module** testB();  **reg** a,b, c;  **wire** y;  boredF b1(.a(a), .b(b), .c(c), .y(y) );    **initial begin**  a= 0; b= 0; c= 0; #10;  c= 1; #10;  b= 1; c= 0; #10;  c= 1; #10;  a= 1; b= 0; c= 0; #10;  **end**  **endmodule**    **TASK:**  **Implement a 4:1 MUX and write the test bench code to verify the module: Multiplexer(4:1)**  **Verilog design:**  **module** mux41( input i0,i1,i2,i3,sel0,sel1, output reg y);  **always** @(\*) **begin**  **case** ({sel0,sel1}) 2'b00 : y = i0; 2'b01 : y = i1; 2'b10 : y = i2; 2'b11 : y = i3; **endcase**  **end**  **endmodule**  **TestBench:**    **module** tb\_mux41;  **reg** I0,I1,I2,I3,SEL0,SEL1; **wire** Y;  mux41 MUX (.i0(I0),.i1(I1),.i2(I2),.i3(I3),.sel0(SEL0),.sel1(SEL1),.y(Y));  **initial** **begin** I0 =1'b0; I1= 1'b0; I2 =1'b0;  I3 =1'b0; SEL0 =1'b0; SEL1 =1'b0; #45 **$finish**;  **end**  **always** #2 I0 = ~I0; **always** #4 I1 =~I1; **always** #6 I2 =~I1; **always** #8 I3 =~I1; **always** #3 SEL0 = ~SEL0; **always** #3 SEL1 = ~SEL1;  **always** @(Y)  **$display**( "time =%0t INPUT VALUES: \t I0=%b I1 =%b I2 =%b I3 =%b SEL0 =%b SEL1 =%b \t **output** **value** Y =%b ",$time,I0,I1,I2,I3,SEL0,SEL1,Y);  **endmodule**  **Output:**  time =0 INPUT VALUES: output value Y =0 time =2 INPUT VALUES: output value Y =1  time =3 INPUT VALUES: output value Y =0 time =6 INPUT VALUES: output value Y =1  time =8 INPUT VALUES: output value Y =0 time =14 INPUT VALUES: output value Y =1  time =15 INPUT VALUES: output value Y =0  I0=0I1=0I2=0I3=0SEL0=0SEL1=0 I0=1I1=0I2=0I3=0SEL0=0SEL1=0 I0=1I1=0I2=0I3=0SEL0=1SEL1=1 I0=1I1=1I2=0I3=0SEL0=0SEL1=0 I0=0I1=0I2=0I3=0SEL0=0SEL1=0  I0=1 I1 =1 I2 =1 I3 =0 SEL0 =0 SEL1 =0 I0=1 I1 =1 I2 =1 I3 =0 SEL0 =1 SEL1 =1 | | | | | |
| **Date:** | **02/06/2020** | **Name:** | | | **Abhishek Vasudev Mahendrakar** |
| **Course:** | **UDEMY-The Python Mega Course: Build 10 real world applications** | **USN:** | | | **4AL17EC003** |
| **Topic:** | **Scrape Real Estate**  **Property Data from the Web** | **Semester & Section:** | | | **6th-‘A’** |
| **AFTERNOON SESSION DETAILS** | | | | | |
| **Image of session** | | | | | |
| **Report – Report can be typed or hand written for up to two pages.**  #Get the first page to extract page numbers  import requests, re  from bs4 import BeautifulSoup  r=requests.get("http://www.pythonhow.com/real-estate/rock-springs-w  y/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(" ","")  page\_nr=soup.find\_all("a",{"class":"Page"})[-1].text  print(page\_nr,"number of pages were found")  l=[]  base\_url="http://www.pythonhow.com/real-estate/rock-springs-wy/LCWY  ROCKSPRINGS/t=0&s="  for page in range(0,int(page\_nr)\*10,10):  print( )  r=requests.get(base\_url+str(page)+".html")  c=r.content  #c=r.json()["list"]  soup=BeautifulSoup(c,"html.parser")  all=soup.find\_all("div",{"class":"propertyRow"})  for item in all:  d={}  d["Address"]=item.find\_all("span",{"class","propAddressCollapse"})[0].text  try:  d["Locality"]=item.find\_all("span",{"class","propAddressCollapse"})[1].text  except:  d["Locality"]=None  d["Price"]=item.find("h4",{"class","propPrice"}).text.replace("\n","").replace(" ","")  try:  d["Beds"]=item.find("span",{"class","infoBed"}).find("b").text  except:  d["Beds"]=None  try:  d["Area"]=item.find("span",{"class","infoSqFt"}).find("b").text  except:  d["Area"]=None  try:  d["Full  Baths"]=item.find("span",{"class","infoValueFullBath"}).find("b").text  except:  d["Full Baths"]=None  try:  d["Half Baths"]=item.find("span",{"class","infoValueHalfBath"}).find("b").text  except:  d["Half Baths"]=None  for column\_group in  item.find\_all("div",{"class":"columnGroup"}):  for feature\_group, feature\_name in  zip(column\_group.find\_all("span",{"class":"featureGroup"}),column\_g  roup.find\_all("span",{"class":"featureName"})):  if "Lot Size" in feature\_group.text:  d["Lot Size"]=feature\_name.text  l.append(d)  import pandas  df=pandas.DataFrame(l)  df  df.to\_csv("Output.csv") | | | | | |