REPORT JUNE 03

|  |  |  |  |
| --- | --- | --- | --- |
| Date: | 03-06-2020 | Name: | Bhoomika Hebbar |
| Course: | Digital Design Using HDL | USN: | 4AL17EC010 |
| Topic: | EDA Playground Online compiler , EDA Playground Tutorial Demo Video , How to Download And Install Xilinx Vivado Design Suite ,  Task for Day-3 | Semester & Section: | 6th SEM A sec |
| Github  Repository: | bhoomika\_python |  |  |

|  |
| --- |
| FORENOON SESSION DETAILS |
| Image of session |
| **INVERTER:** A power **inverter**, or **inverter**, is a power electronic device or circuitry that changes direct current (DC) to alternating current (AC). The input voltage, output voltage and frequency, and overall power handling depend on the design of the specific device or circuitry.  **Implement Inverter Using The EDA Tool:**  module inverter(y,a); output y;  input a; assign y=~a; endmodule  Testbench Code  timescale ins/lps |

|  |
| --- |
| module testbench(); reg a1;  wire y1;  inverter inv1(a1,y1); initial begin  a1=a’b1;  $display(“a=%b”,a1);  end endmodule  2.RCC: A **ripple counter** is an asynchronous **counter** where only the first flip-flop is clocked by an external clock. All subsequent flip-flops are clocked by the output of the preceding flip-flop. Asynchronous **counters** are also called **ripple**-**counters** because of the way the clock pulse **ripples** it way through the flip-flops.  Ripple Carry Counter  module ripple\_counter\_4\_bit(q,clk,reset); input clk,reset;  output[3:0]q;  T\_FF tff0(q[0],clk,reset); T\_FF tff1(q[1],q[0],reset);  T\_FF tff2(q[2],q[1],reset);  T\_FF tff3(q[3],q[2],reset); endmodule  module T\_FF(q,clk,reset); input clk,reset;  output q; wire d;  D\_FF dff0(q,d,clk,reset); not n1(d,q);  endmodule  module D\_FF(q,d,clk,reset); input d,clk,reset;  output reg q;  always@(negedge clk or posedge reset) begin  if(reset) q<=1'b0;  else q<=d; end  endmodule  TestBench Code  module test  reg clk,reset; wire(3:0)q;  ripple\_carry\_counter rcc(q,clk,reset);initial begin |

# $dumpfile(“dump.vcd”);

$dumpvars(1,test); clk=1’b0; reset=1’b1;

#10 reset=1’b0;

#200;

end

always #5 clk=~clk; endmodule

Implement 4 to 1 MUX using structural modelling style and test the module in an online/offline compiler.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mux2\_1 is

port(A,B : in STD\_LOGIC; S: in STD\_LOGIC;

Z: out STD\_LOGIC);

end mux2\_1;

architecture Behavioral of mux2\_1 is begin

process (A,B,S) is begin

if (S ='0') then Z <= A;

else

Z <= B;

end if; end process;

end behavioral;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mux4\_1 is port(

A,B,C,D : in STD\_LOGIC; S0,S1: in STD\_LOGIC; Z: out STD\_LOGIC

);

end mux4\_1;

architecture Behavioral of mux4\_1 is component mux2\_1

port( A,B : in STD\_LOGIC; S: in STD\_LOGIC;

Z: out STD\_LOGIC);

end component;

signal temp1, temp2: std\_logic; begin

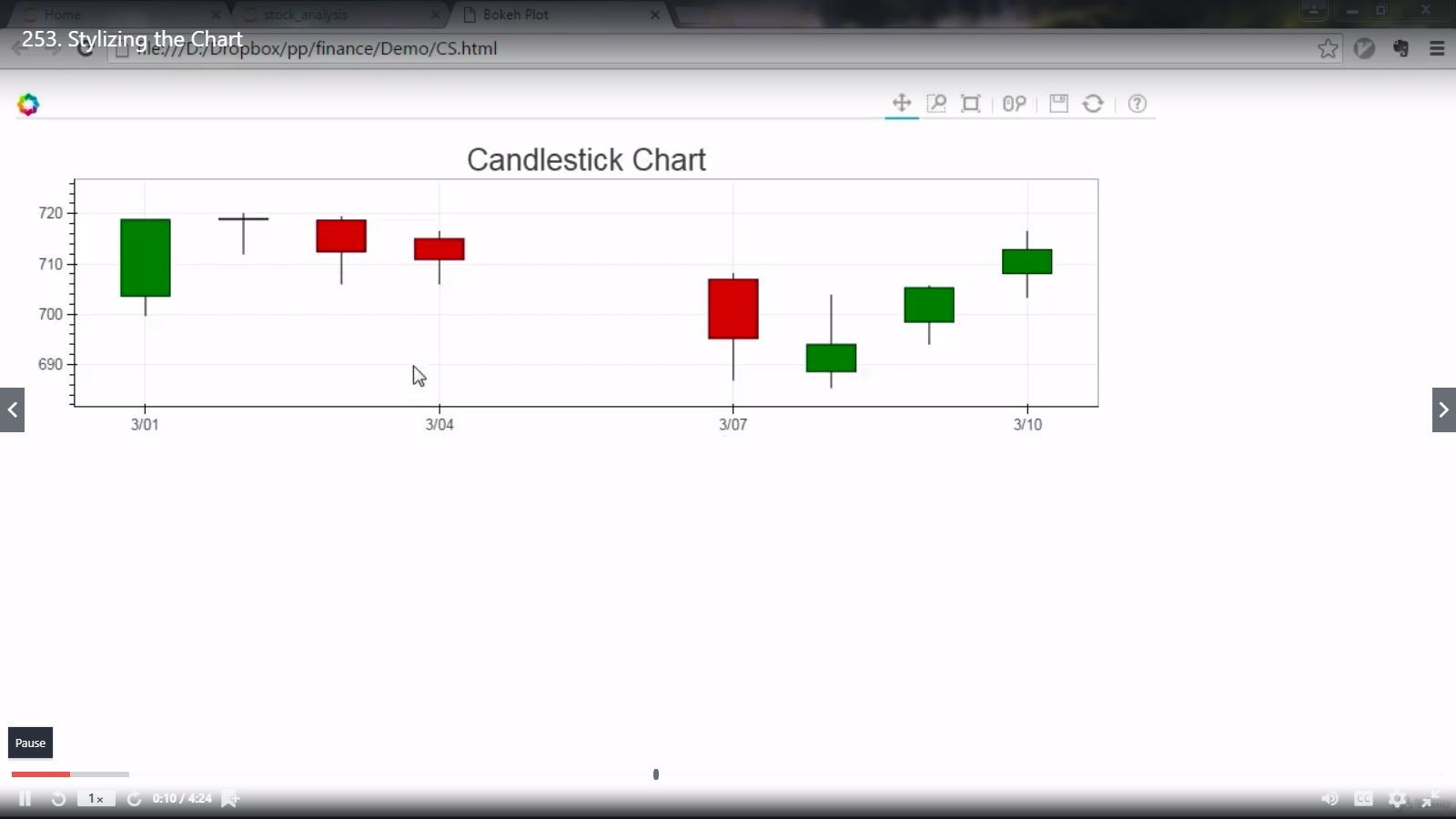
m1: mux2\_1 port map(A,B,S0,temp1); m2: mux2\_1 port map(C,D,S0,temp2); m3: mux2\_1 port map(temp1,temp2,S1,Z);

end behavioral;

# Date: 03-06-2020 Name: Bhoomika Hebbar

Course: Python by Udemy USN:4al17ec010

Topic: Build a Web-based Financial Graph Semester & Section:6 th A



**Image of session:Output**

**AFTERNOON SESSION DETAILS**

# Build a Web-based Financial Graph

from flask import Flask, render\_template app=Flask( name )

@app.route('/plot/') def plot():

from pandas\_datareader import data import datetime

import fix\_yahoo\_finance as yf yf.pdr\_override()

from bokeh.plotting import figure, show, output\_file from bokeh.embed import components

from bokeh.resources import CDN

start=datetime.datetime(2015,11,1) end=datetime.datetime(2016,3,10)

df=data.get\_data\_yahoo(tickers="GOOG", start=start, end=end)

def inc\_dec(c, o): if c > o:

value="Increase" elif c < o:

value="Decrease" else:

value="Equal" return value

df["Status"]=[inc\_dec(c,o) for c, o in zip(df.Close,df.Open)] df["Middle"]=(df.Open+df.Close)/2

df["Height"]=abs(df.Close-df.Open)

p=figure(x\_axis\_type='datetime', width=1000, height=300) p.title.text="Candlestick Chart" p.grid.grid\_line\_alpha=0.3

hours\_12=12\*60\*60\*1000

p.segment(df.index, df.High, df.index, df.Low, color="Black")

p.rect(df.index[df.Status=="Increase"],df.Middle[df.Status=="Increase"], hours\_12,

df.Height[df.Status=="Increase"],fill\_color="#CCFFFF",line\_color="black")

p.rect(df.index[df.Status=="Decrease"],df.Middle[df.Status=="Decrease"], hours\_12,

df.Height[df.Status=="Decrease"],fill\_color="#FF3333",line\_color="black")

script1, div1 = components(p) cdn\_js=CDN.js\_files[0] cdn\_css=CDN.css\_files[0]

return render\_template("plot.html", script1=script1,

div1=div1, cdn\_css=cdn\_css, cdn\_js=cdn\_js )

@app.route('/') def home():

return render\_template("home.html")

@app.route('/about/') def about():

return render\_template("about.html")

if name ==" main ": app.run(debug=True)