**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **02/06/2020** | **Name:** | **Namratha S Hipparagi** |
| **Course:** | **HDL design** | **USN:** | **4AL16EC040** |
| **Topic:** | **Analysis of clocked sequential circuits**  **Digital clock design** | **Semester & Section:** | **8 A** |
| **Github Repository:** | **namrathahipparagi\_1** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  This circuit was a school project and it was a 24h clock, but I decided to extend it to 12h and 24h with the transferring between them using a switch so you can choose whatever modeyouwant. However, if for whatever reason you run multiple output file() commands in the same execution, only the last one will be used for rendering. Anyway, I made this circuit just for fun so I just simulated it on proteus, i.e we're not going into pcb design.    The main parts of the circuit are as follows:  1- Timer 555: Responsible for generating the clock pulses for the counters, the frequency of the output should be 1 kHz which means 1 second for each pulse.  2- Counters: Responsible for generating the time in BCD (Binary Coded decimal).  3- Decoders: Takes the BCD of the counter as input and produces 7 segment output.  **Working:**  The functional parts of the 555 timer IC include flip-flop, voltage divider and a comparator. The main function of this IC is to generate an accurate timing pulse. In the monostable mode, the delay of this IC is controlled by the external components like a resistor and capacitor. 555 timer produces 1 second pulses to the clock input of the first counter which is responsible the first column of seconds, so its output will change every second. So the output of the first counter will count from 0 to 9 every second and that's exactly what we want from it, so we are done here. **The JK Flip Flop**   The J-K flip-flop is the most versatile of the basic flip-flops. It has the input- following character of the clocked D flip-flop but has two inputs, traditionally labeled J and K. If J and K are different then the output Q takes the value of J at the next clock edge. The JK Flip-flop is similar to the SR Flip-flop but there is no change in state when the J and K inputs are both LOW.  The basic S-R NAND flip-flop circuit has many advantages and uses in sequential logic circuits but it suffers from two basic switching problems.   * the Set = 0 and Reset = 0 condition (S = R = 0) must always be avoided * if Set or Reset change state while the enable (EN) input is high the correct latching action may not occur.   Then to overcome these two fundamental design problems with the SR flip-flop design, the JK flip Flop was developed. The two inputs labelled “J” and “K” are not shortened abbreviated letters of other words, such as “S” for Set and “R” for Reset, but are themselves autonomous letters chosen by its inventor Jack Kilby to distinguish the flip-flop design from other types.  The sequential operation of the JK flip flop is exactly the same as for the previous SR flip-flop with the same “Set” and “Reset” inputs. The JK flip flop is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level “1”. Due to this additional clocked input, a JK flip-flop has four possible input combinations, “logic 1”, “logic 0”, “no change” and “toggle”. The symbol for a JK flip flop is similar to that of an *SR Bistable Latch* as seen in the previous tutorial except for the addition of a clock input. The difference this time is that the “JK flip flop” has no invalid or forbidden input states of the SR Latch even when S and R are both at logic “1”. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **02/6/2020** | **Name:** | **Namratha S Hipparagi** | |
| **Course:** | **Python** | **USN:** | **4al16ec040** | |
| **Topic:** | **Interactive Data Visualization with Bokeh**  **Webscraping with Python Beautiful Soup** | **Semester & Section:** | **8 A** | |
| **AFTERNOON SESSION DETAILS** | | | |
| REPORT  It’s worth briefly mentioning how Bokeh differs from matplotlib, and when one might be preferred to the other. Bokeh is a library for creating interactive data visualizations in a web browser. If you’ve worked with visualization in Python before, it’s likely that you have used [matplotlib](https://matplotlib.org/" \t "_blank). It offers a concise, human-readable syntax, which allows for rapidly presenting data in an aesthetically pleasing manner. Bokeh Visualization Template  This template is a general outline for turning your data into a  visualization using Bokeh.  """  # Data handling  import pandas as pd  import numpy as np  # Bokeh libraries  from bokeh.io import output\_file, output\_notebook  from bokeh.plotting import figure, show  from bokeh.models import ColumnDataSource  from bokeh.layouts import row, column, gridplot  from bokeh.models.widgets import Tabs, Panel  # Prepare the data  # Determine where the visualization will be rendered  output\_file('filename.html') # Render to static HTML, or  output\_notebook() # Render inline in a Jupyter Notebook  # Set up the figure(s)  fig = figure() # Instantiate a figure() object  # Connect to and draw the data  # Organize the layout  # Preview and save  show(fig) # See what I made, and save if I like it Generating Your First Figure By calling both output\_file() and output\_notebook() in the same execution, the visualization will be rendered both to a static HTML file and inline in the notebook.  There are [multiple ways to output your visualization](https://bokeh.pydata.org/en/latest/docs/user_guide/concepts.html#output-methods) in Bokeh.   * output\_file('filename.html') will write the visualization to a static HTML file. * output\_notebook() will render your visualization directly in a Jupyter Notebook.   It’s important to note that neither function will actually show you the visualization. That doesn’t happen until show() is called. However, if for whatever reason you run multiple output\_file() commands in the same execution, only the last one will be used for rendering.  This is a great opportunity to give you your first glimpse at a default Bokeh figure() using output\_file():  # Bokeh Libraries  from bokeh.io import output\_file  from bokeh.plotting import figure, show  # The figure will be rendered in a static HTML file called output\_file\_test.html  output\_file('output\_file\_test.html',  title='Empty Bokeh Figure')  # Set up a generic figure() object  fig = figure()  # See what it looks like  show(fig) | | | |
|  | | | |