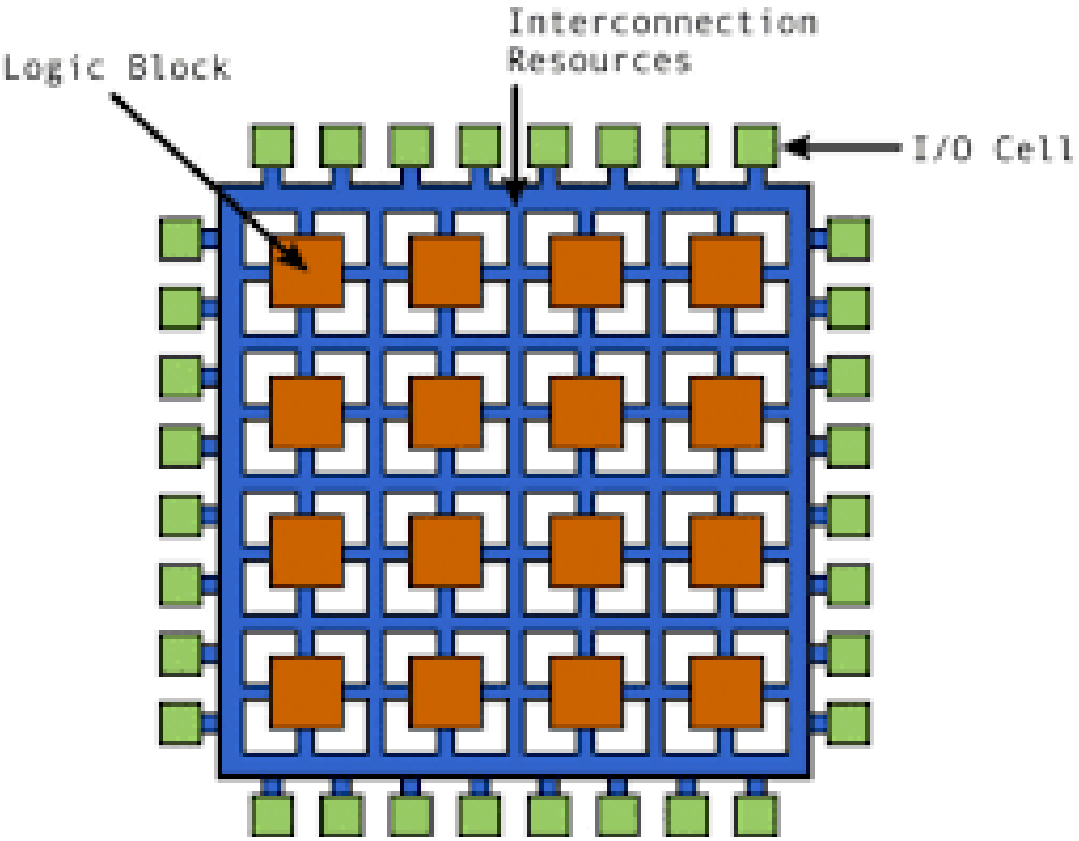
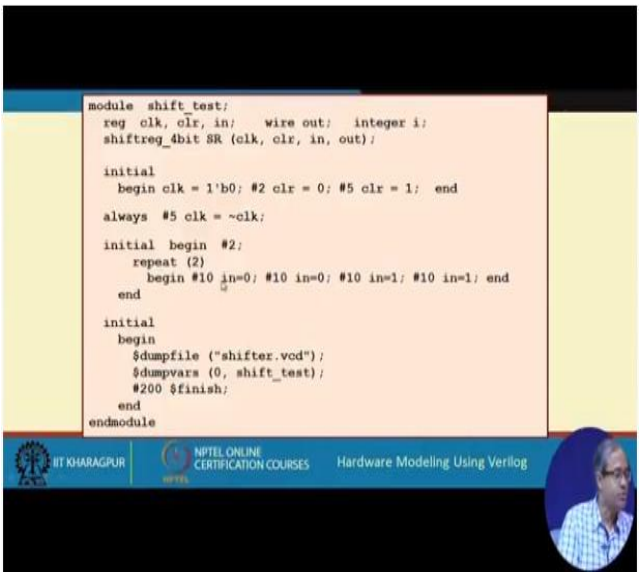
DAILY ASSESSMENT FORMAT

Date:	**02/06/2020**	**Name:**	**Apeksha S Shetty**
Course:	**DIGITAL DESIGN USING HDL**	**USN:**	**4AL15EC006**
Topic:	FPGA Basics: Architecture, Applications and Uses Verilog HDL Basics by Intel Verilog Testbench code to verify the design under test (DUT)	**Semester & Section:**	**8 A**
Github Repository:	**Apeksha-97**		

IMAGE SECTION



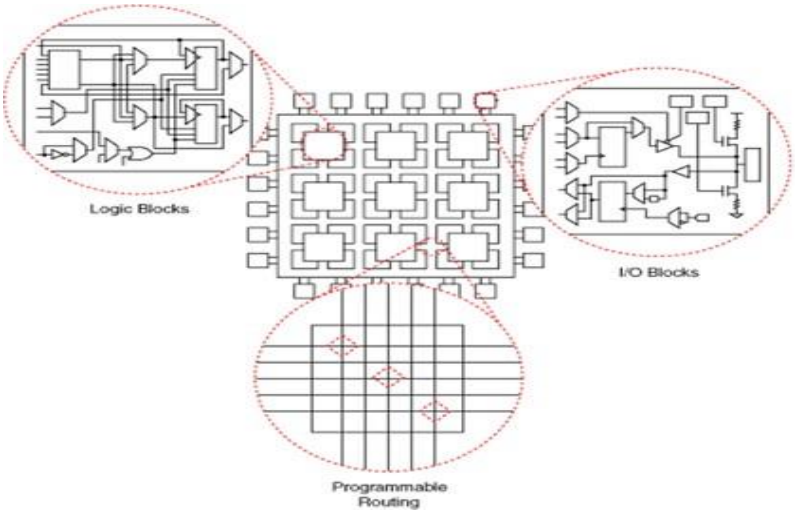

module shift_test;
reg clk, clr, in; wire out; integer i;
shiftreg_4bit SR (clk, clr, in, out);
initial
begin clk = 1'b0; #2 clr = 0; #5 clr = 1; end
always #5 clk = ~clk;
initial begin #2;
repeat (2)
begin #10 in=0; #10 in=0; #10 in=1; #10 in=1; end
end
initial
begin
$dumpfile ("shifter.vcd");
$dumpvars (0, shift_test);
#200 $finish;
end
endmodule
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Hardware Modeling Using Verilog

Report-

The field-programmable gate array (FPGA) is an integrated circuit that consists of internal hardware blocks with user-programmable interconnects to customize operation for a specific application.

What is FPGA?

- The field-programmable gate array (FPGA) is an integrated circuit that consists of internal hardware blocks with user-programmable interconnects to customize operation for a specific application.
- The interconnects can readily be reprogrammed, allowing an FPGA to accommodate changes to a design or even support a new application during the lifetime of the part.

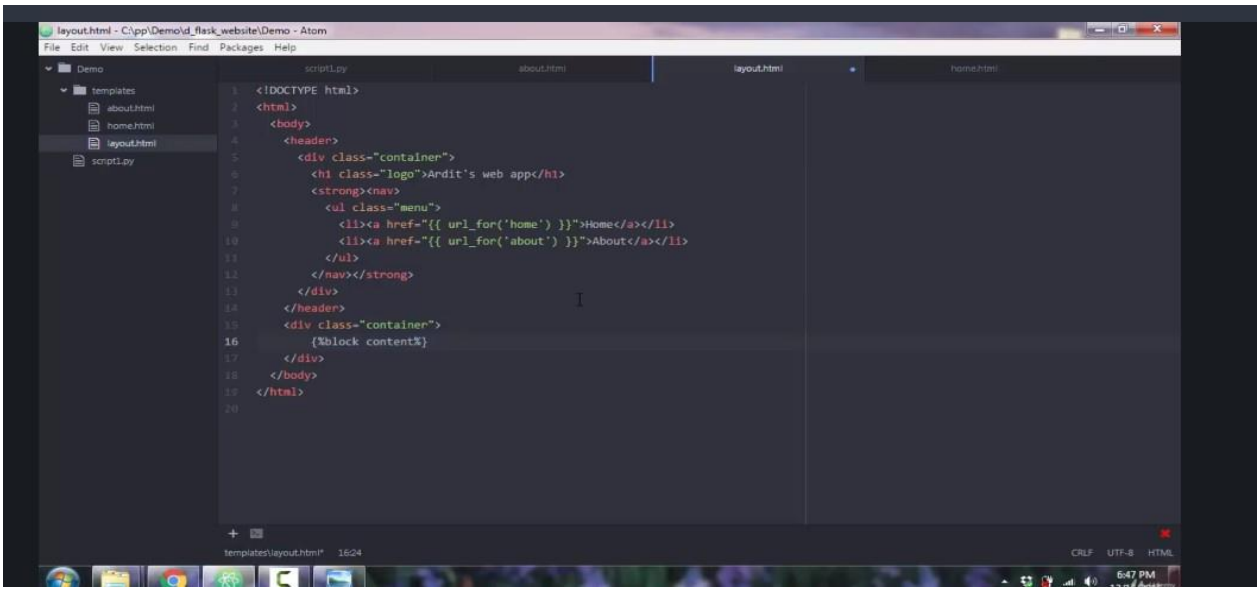


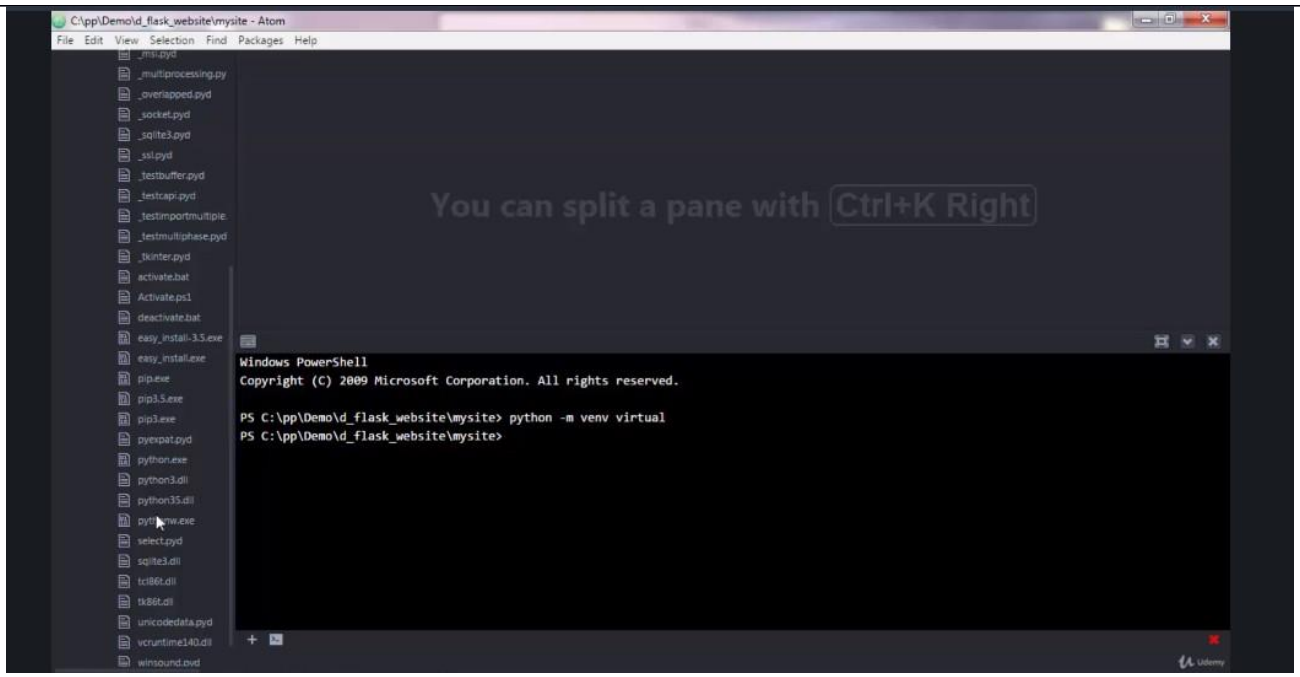
- The FPGA has its roots in earlier devices such as programmable read-only memories (PROMs) and programmable logic devices (PLDs).
- These devices could be programmed either at the factory or in the field, but they used fuse technology (hence, the expression “burning a PROM”) and could not be changed once programmed.
- In contrast, FPGA stores its configuration information in a re-programmable medium such as static RAM (SRAM) or flash memory. FPGA manufacturers

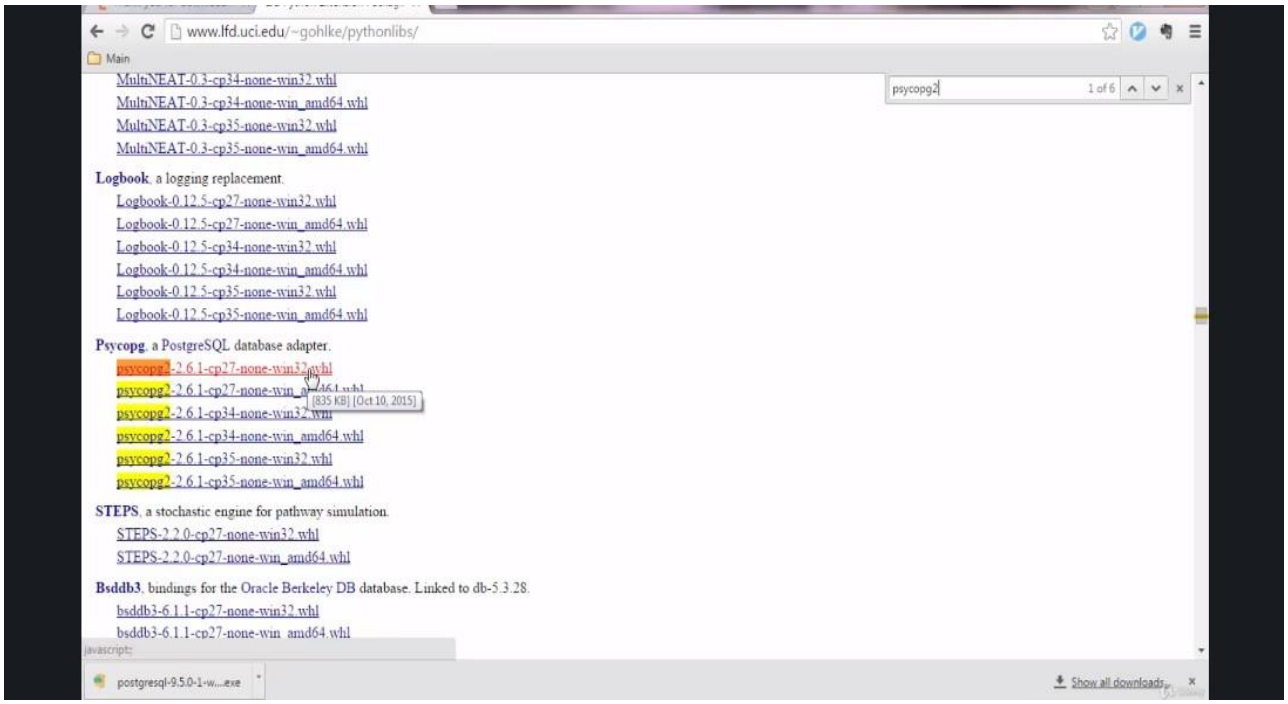
Date:	**02/06/2020**	**Name:**	**Apeksha S Shetty**
Course:	**Python**	**USN:**	**4AL15EC006**
Topic:	Interactive data visualization with bokeh, webscraping with python beautiful soup.	**Semester & Section:**	**8 A**
Github Repository:	**Apeksha-97**		

AFTERNOON SESSION DETAILS

Image of session




C:\pp\Demo\d_flask_website\mysite - Atom
File Edit View Selection Find Packages Help
_multiprocessing.py
_overlapped.pyd
_socket.pyd
_sqlite3.pyd
_ssl.pyd
_testbuffer.pyd
_testcapi.pyd
_testimportmultiple
_testmultiphase.pyd
_tkinter.pyd
activate.bat
Activate.ps1
deactivate.bat
easy_install-3.5.exe
easy_install.exe
pip.exe
pip3.5.exe
pip3.exe
pyexpat.pyd
python.exe
python3.dll
python35.dll
select.pyd
sqlite3.dll
tcl86t.dll
tk86t.dll
unicodedata.pyd
vcruntime140.dll
winsound.pyd
You can split a pane with Ctrl+K Right
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.
PS C:\pp\Demo\d_flask_website\mysite> python -m venv virtual
PS C:\pp\Demo\d_flask_website\mysite>


www.lfd.uci.edu/~gohlke/pythonlibs/
Main
psycopg2
1 of 6
MultiNEAT-0.3-cp34-none-win32.whl
MultiNEAT-0.3-cp34-none-win_amd64.whl
MultiNEAT-0.3-cp35-none-win32.whl
MultiNEAT-0.3-cp35-none-win_amd64.whl
Logbook, a logging replacement.
Logbook-0.12.5-cp27-none-win32.whl
Logbook-0.12.5-cp27-none-win_amd64.whl
Logbook-0.12.5-cp34-none-win32.whl
Logbook-0.12.5-cp34-none-win_amd64.whl
Logbook-0.12.5-cp35-none-win32.whl
Logbook-0.12.5-cp35-none-win_amd64.whl
Psycopg, a PostgreSQL database adapter.
psycopg2-2.6.1-cp27-none-win32.whl
[835 KB] [Oct 10, 2015]
psycopg2-2.6.1-cp34-none-win_amd64.whl
psycopg2-2.6.1-cp35-none-win32.whl
psycopg2-2.6.1-cp35-none-win_amd64.whl
STEPS, a stochastic engine for pathway simulation.
STEPS-2.2.0-cp27-none-win32.whl
STEPS-2.2.0-cp27-none-win_amd64.whl
Bsddb3, bindings for the Oracle Berkeley DB database. Linked to db-5.3.28.
bsddb3-6.1.1-cp27-none-win32.whl
bsddb3-6.1.1-cp27-none-win_amd64.whl
javascript:
postgresql-9.5.0-1-w....exe
Show all downloads

REPORT

If you haven't installed Bokeh yet, you can easily install it with pip from the terminal:

```
pip install bokeh
```

Or you use pip3:

```
pip3 install bokeh
```

Snippet producing the triangle based plot

```
#Making a basic Bokeh line graph

#importing Bokeh
from bokeh.plotting import figure
from bokeh.io import output_file, show

#prepare some data
x=[3,7.5,10]
y=[3,6,9]

#prepare the output file
```

```
output_file("Line.html")

#create a figure object
f=figure()

#create line plot
f.triangle(x,y)

#write the plot in the figure object
show(f)
```

#Snippet producing the circle based plot

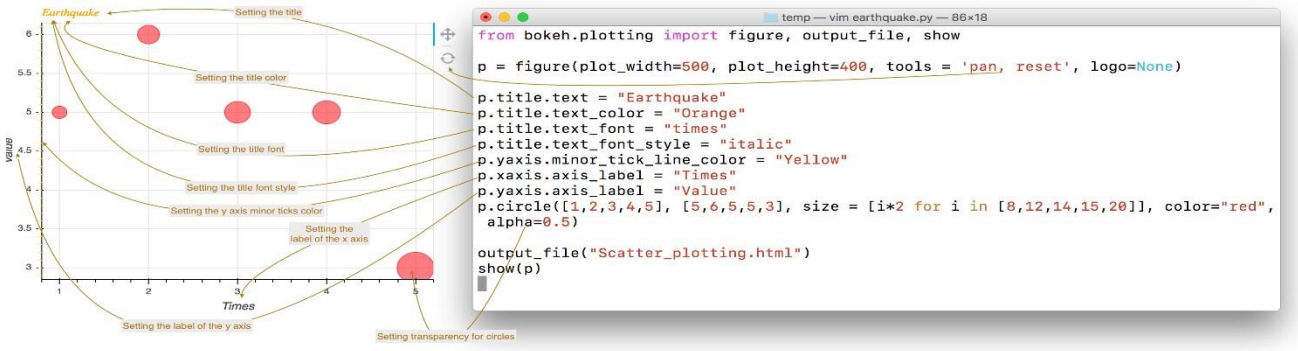
```
#Making a basic Bokeh line graph

#importing Bokeh
from bokeh.plotting import figure
from bokeh.io import output_file, show

```

```
#prepare some data
x=[3,7.5,10]
y=[3,6,9]

#prepare the output file
output_file("Line.html")
13.

#create a figure object
f=figure()
16.

#create line plot
f.circle(x,y)
19.

#write the plot in the figure object
show(f)
```

Visual Attributes

Once you have built a basic plot, you can customize its visual attributes including changing the `title` color and font, adding labels for `xaxis` and `yaxis`, changing the

color of the axis ticks, etc. All these properties are illustrated in the diagram below:



And here is the code if you want to play around with it:

```
from bokeh.plotting import figure, output_file, show

p = figure(plot_width=500, plot_height=400, tools = 'pan, reset')

p.title.text = "Earthquakes"

p.title.text_color = "Orange"

p.title.text_font = "times"

p.title.text_font_style = "italic"

p.yaxis.minor_tick_line_color = "Yellow"

p.xaxis.axis_label = "Times"

p.yaxis.axis_label = "Value"

p.circle([1,2,3,4,5], [5,6,5,5,3], size = [i*2 for i in [8,12,14,15,20]], color="red",
alpha=0.5)

output_file("Scatter_plotting.html")
```

12. show(p)

For a complete list of visual attributes, see the Styling Visual Attributes documentation page of Bokeh.