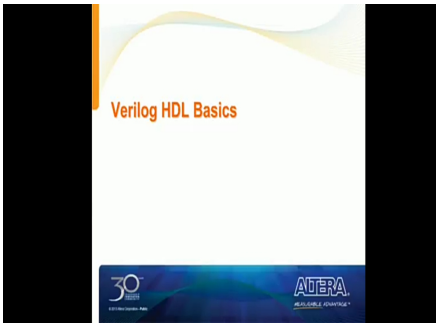
JUNE 2 REPORT

Date:	02/06/20	Name:	ANKITHA C C
Course:	DIGITAL DESIGN USING HDL	USN:	4AL16EC004
Topic:	FPGA	Semester & Section:	8TH & A
Github Repository:	ankitha-course		

FORENOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

Fpga basics

Architecture

FPGA Architecture

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).

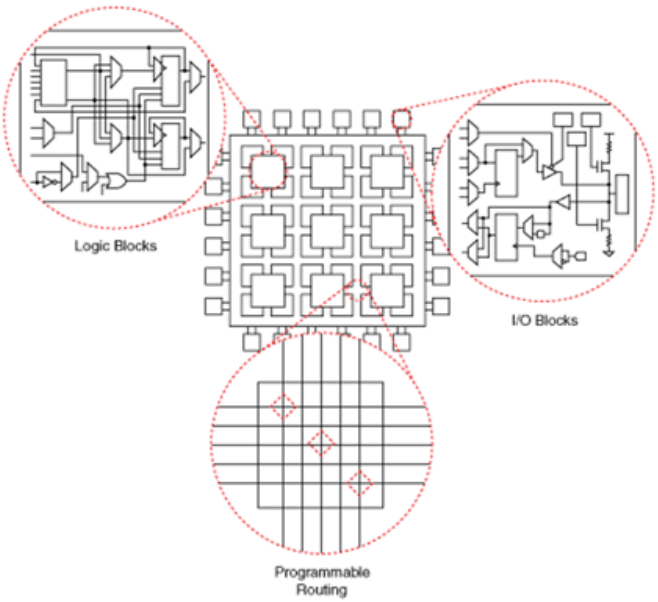


Figure 1: The fundamental FPGA architecture (Image Source: National Instruments)

An individual CLB (Figure 2) is made up of several logic blocks. A lookup table (LUT) is a characteristic feature of an FPGA. An LUT stores a predefined list of logic outputs for any combination of inputs: LUTs with four to six input bits are widely used. Standard logic functions such as multiplexers (mux), full adders (FAs) and flip-flops are also common.

FPGA Applications

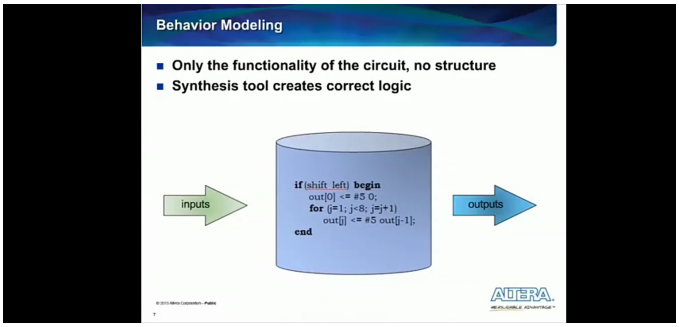
Many applications rely on the parallel execution of identical operations; the ability to configure the FPGA's CLBs into hundreds or thousands of identical processing blocks has applications in image processing, artificial intelligence (AI), data center hardware accelerators, enterprise networking and automotive advanced driver assistance systems (ADAS).

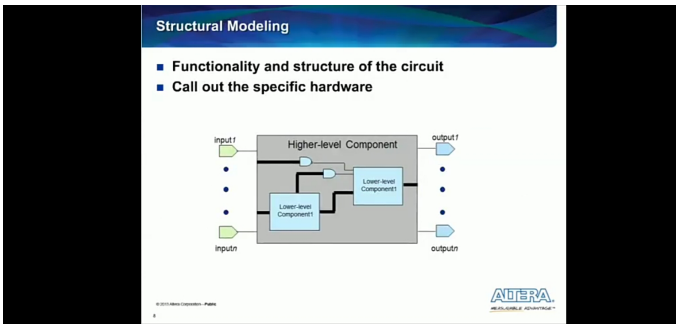
Many of these application areas are changing very quickly as requirements evolve and new protocols and standards are adopted. FPGAs enable manufacturers to implement systems that can be updated when necessary.

A good example of FPGA use is high-speed search: Microsoft is using FPGAs in its data centers 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 modeling routines in an HPC application. This flexibility is difficult or impossible to achieve with an ASIC.

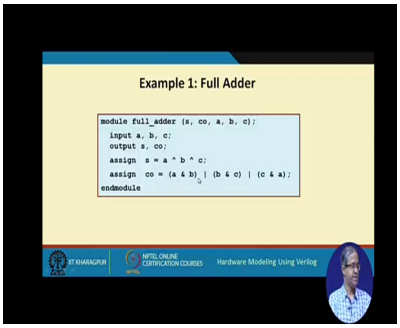
Other FPGA uses include aerospace and defense, medical electronics, digital television, consumer electronics, industrial motor control, scientific instruments, cybersecurity systems and wireless communications.

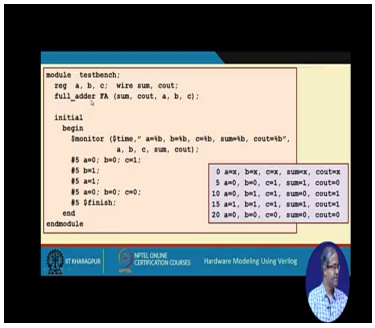
2. Verilog hdl basics by intel





3. Verilog test bench code to verify the design under test





Example

$y = (b' \cdot c') + (a \cdot b')$

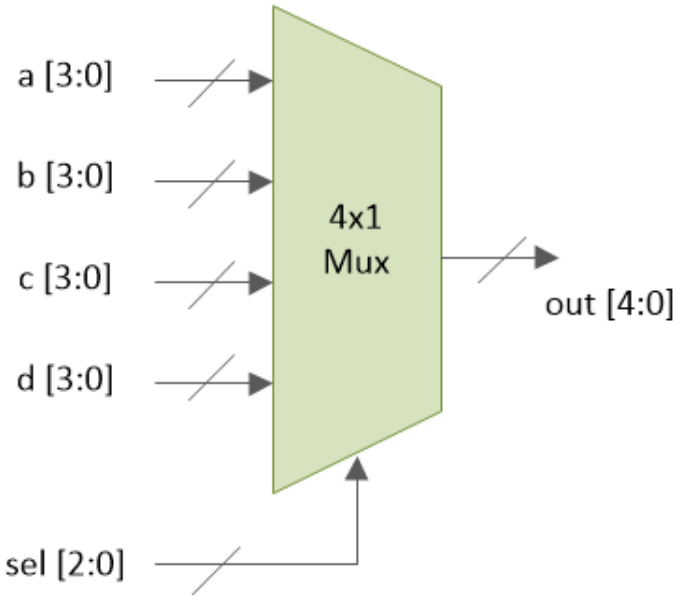
```
module sillyfunction(input a, b, c,
output y);
assign y = ~b & ~c | a & ~b;
endmodule
```

Test bench code

```
module testbench1();
reg a, b, c;
wire y;
sillyfunction dut (.a(a), .b(b), .c(c), .y(y) );d
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 2

Implement a 4:1 MUX and write the test bench code to verify the module

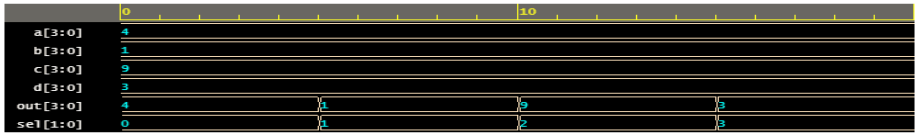



```
module mux_4to1_assign ( input [3:0] a,  b, c, d, input [1:0] sel, output [3:0] out);
  assign out = sel[1] ? (sel[0] ? d : c) : (sel[0] ? b : a);
```

```
endmodule
```

Testbench

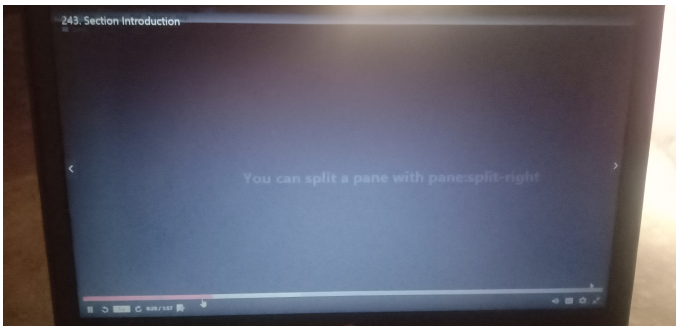
```
module tb_4to1_mux;
  reg [3:0] a, b, c,  d;
  wire [3:0] out;
  reg [1:0] sel;
  integer i;
 mux_4to1_case  mux0 (   .a (a),   .b (b),  .c (c), .d (d), .sel (sel), .out (out));
  initial begin
    $monitor ("[%0t] sel=0x%0h a=0x%0h b=0x%0h c=0x%0h d=0x%0h out=0x%0h", $time, sel, a, b, c, d, out);
   sel <= 0; a <= $random; b <= $random; c <= $random;  d <= $random;
    for (i = 1; i < 4; i=i+1) begin
      #5 sel <= i;
    end
 #5 $finish;
  end
endmodule
```



Date:	02/06/20	Name:	Ankitha c c
Course:	Python	USN:	4al16ec004
Topic:		Semester & Section:	8th & a

AFTERNOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

Installing Bokeh

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

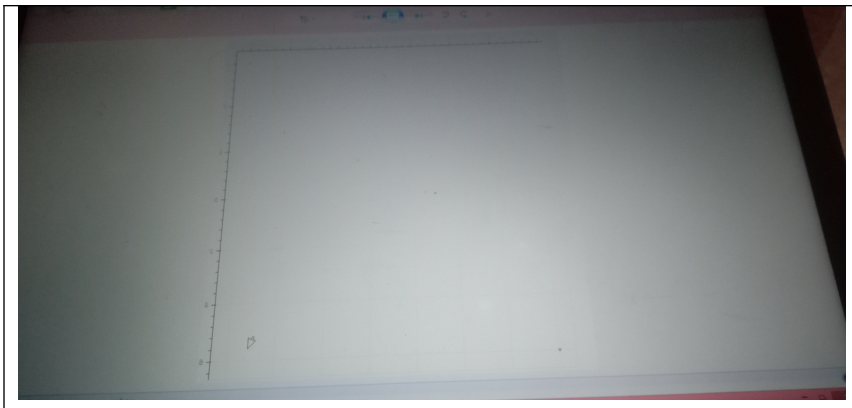
pip install bokeh

Or you use pip3:

pip3 install bokeh

Plotting Triangles and Circle Glyphs (Practice)

Write two code snippets, each producing the following graphs. The first graph has triangles as glyphs and the second graph has circles as glyphs. You can use triangle , and circle instead of line. You should have the same coordinates as shown in the plots below.



Solution

```
#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
```

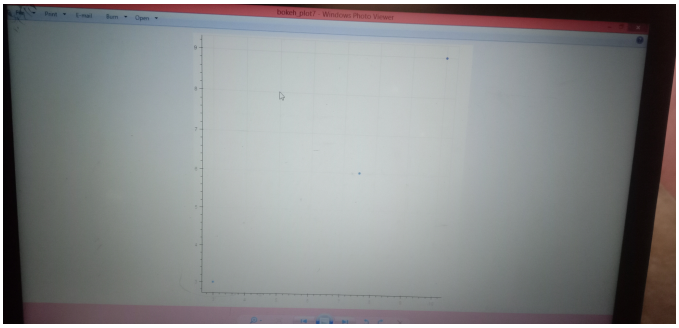
```
#prepare some data

x=[3,7.5,10]

y=[3,6,9]

#prepare the output file

output_file("Line.html")

#create a figure object

f=figure()

#create line plot

f.circle(x,y)

#write the plot in the figure object

show(f)
```

Plotting Education Data (Practice)

The following line graph shows the percentage of women who have received a bachelor's degree over the years in USA. The graph was produced from the Year and Engineering columns of the CSV file provided in the following link:

http://pythonhow.com/data/bachelors.csv

Try to reproduce the graph using Bokeh.



Solution

```
#Plotting percentage of women who received an engineering degree over years

#importing bokeh and pandas
```

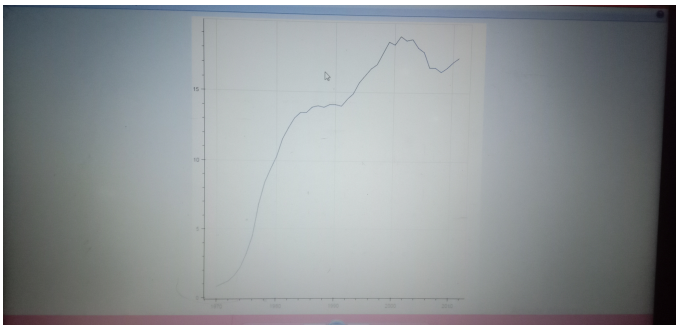
```
from bokeh.plotting import figure

from bokeh.io import output_file, show

import pandas

#prepare some data

df=pandas.read_csv("http://pythonhow.com/data/bachelors.csv")

x=df["Year"]

y=df["Engineering"]

#prepare the output file

output_file("Line_from_bachelors.html")

#create a figure object

f=figure()

#create line plot

f.line(x,y)

#write the plot in the figure object

show(f)
```

Plot Properties

You can add a title to the plot, set the figure width and height, change title font, etc. Below is a summary of properties which can be added to change the style of the plot:

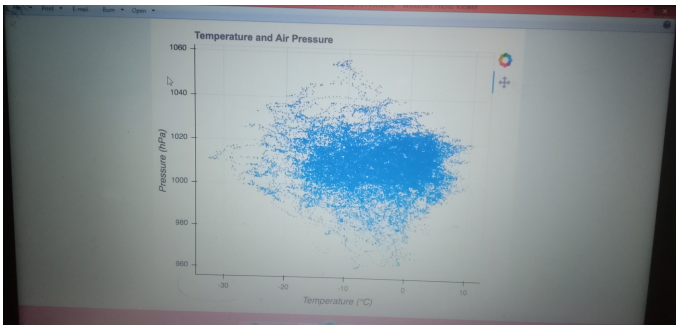
```
import pandas

from bokeh.plotting import figure, output_file, show


p=figure(plot_width=500,plot_height=400, tools='pan',logo=None)


p.title.text="Cool Data"

p.title.text_color="Gray"

p.title.text_font="times"

p.title.text_font_style="bold"
```

```
p.xaxis.minor_tick_line_color=None

p.yaxis.minor_tick_line_color=None

p.xaxis.axis_label="Date"

p.yaxis.axis_label="Intensity"


p.line([1,2,3],[4,5,6])

output_file("graph.html")

show(p)
```



Plotting Weather Data (Practice)

Produce the following graph using the data from this Excel file: http://pythonhow.com/data/verlegenhuken.xlsx



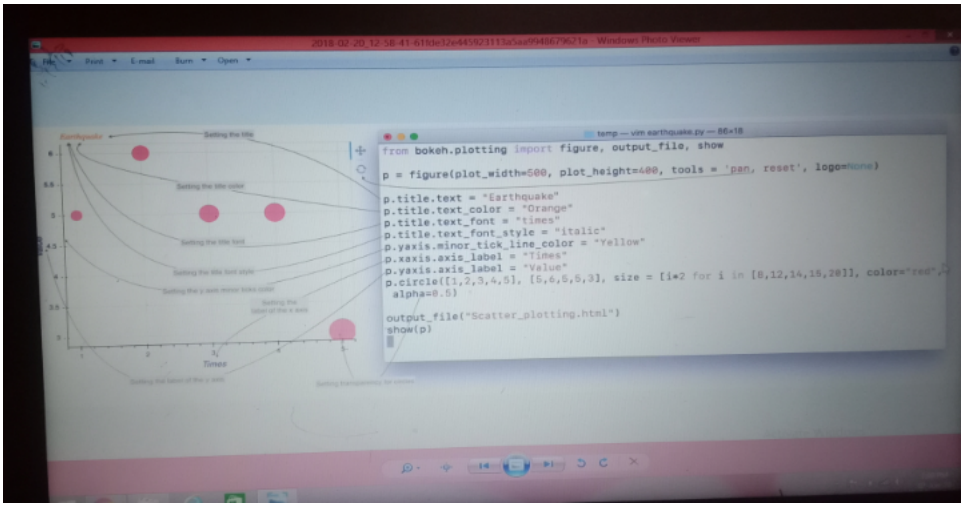
Solution

```
import pandas

from bokeh.plotting import figure, output_file, show

df=pandas.read_excel("http://pythonhow.com/data/verlegenhuken.xlsx",sheet_name=0)

df["Temperature"]=df["Temperature"]/10

df["Pressure"]=df["Pressure"]/10
```

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

p.title.text="Temperature and Air Pressure"

p.title.text_color="Gray"

p.title.text_font="arial"

p.title.text_font_style="bold"

p.xaxis.minor_tick_line_color=None

p.yaxis.minor_tick_line_color=None

p.xaxis.axis_label="Temperature (°C)"

p.yaxis.axis_label="Pressure (hPa)"

p.circle(df["Temperature"],df["Pressure"],size=0.5)

output_file("Weather.html")

show(p)
```

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")

show(p)
```

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

Request Headers

Note

When I use this code in the next video:

```
r = requests.get("http://www.pythonhow.com/example.html")
```

please use this instead:

```
r = requests.get("http://www.pyclass.com/example.html", headers={'User-agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefox/61.0'})
```

The rest of the code stays the same.

So, we're just changing the domain name from pythonhow to pyclass and we're adding a header argument. Some webpages don't like scripts sometimes, so adding a header allows the script to impersonate a web browser.