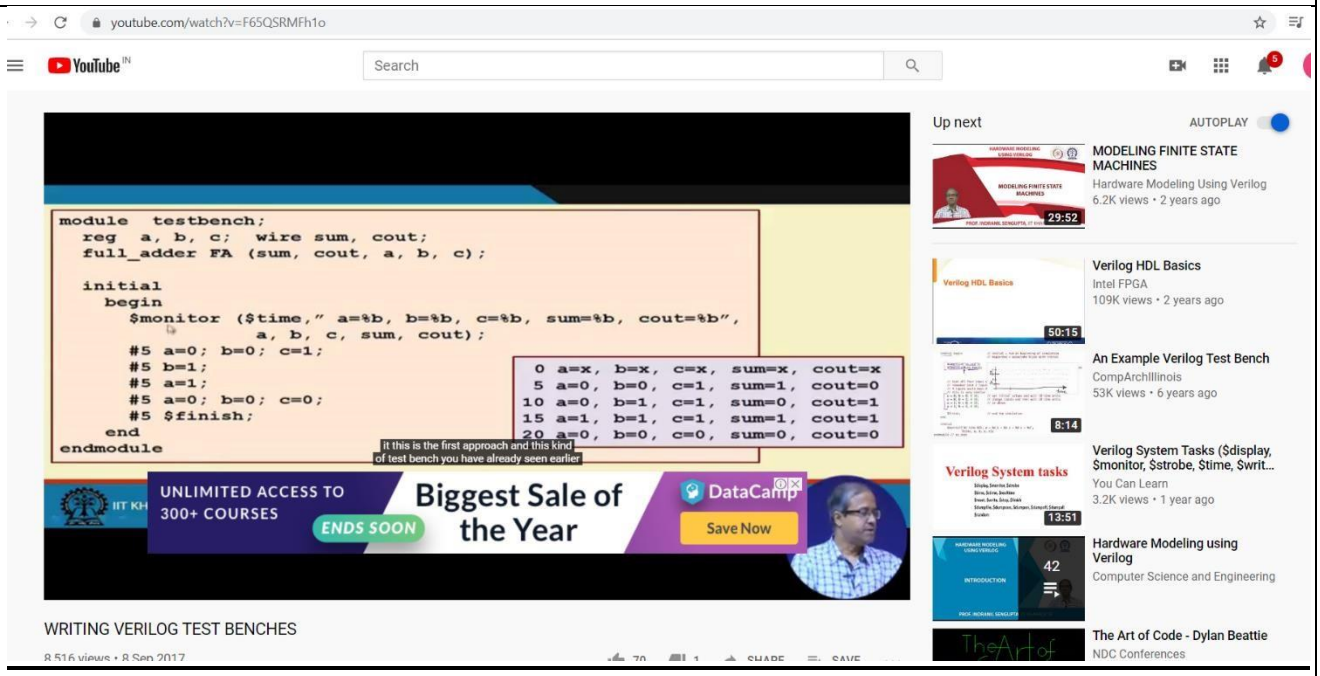
DAILY ASSESSMENT FORMAT

Date:	2-06-2020	**Name:**	Poorvi j
Course:	Digital design using hdl	**USN:**	4al17ec071
Topic:	1.fpga basics architechture ,application and uses. 2.verilog h dl basics 3.test bench waveform 4.task2	**Semester & Section:**	6th b
Github Repository:	Poorvi-2000		

FORENOON SESSION DETAILS



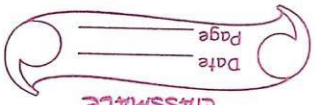
2-06-2020 :- FPGA BASICS Architecture, applications and uses.

FPGA is an internal circuit that consists of internal hardware blocks with the user programable interconnects to customize operation for a specific application

→ FPGA has 2 types → PROM's & PLD's

→ FPGA architecture consists of thousands of fundamental elements called configurable logic block.

→ FPGA use for high speed search & Microsoft is using FPGA's in its data centre - to run bing search algorithm



~~FPGA~~ verilog HDL BASICS

RTL → type of behavioural modeling for the purpose of synthesis

```
module module_name(port_list);
Port declarations
data type declarations
Circuit functionality
timing specifications
endmodule
```

variable data type → element to store data temporarily

net data type → physical interconnect between structure
- wire

```
ex:- module clk_gen
(
  output reg clk
);
  initial clk = 1'b0;
  always
    #(period/2) clk = ~clk;
    initial #100 $finish
endmodule
```

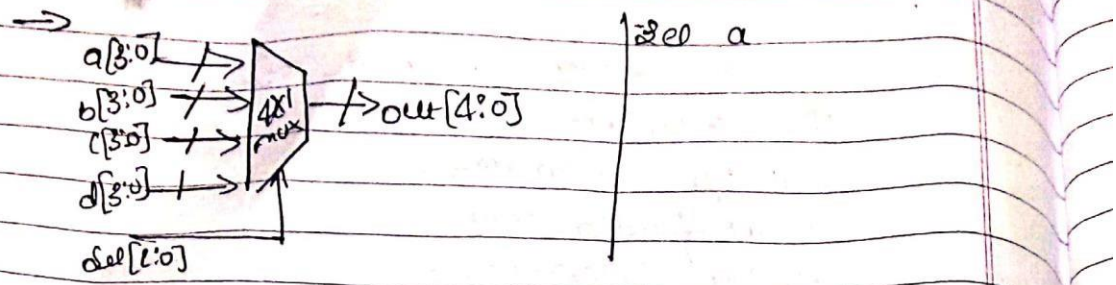
Procedural assignments: ① Blocking (=) ② Non-blocking (<=)

→ verilog testbench code to verify the design under test

```
module full_adder(s, co, a, b, c);
  input a, b, c;
  output s, co;
  assign s = a^b^c;
  assign co = (a&b)|(b&c)|(c&a);
endmodule
```

```
module testbench
  reg a, b, c; wire sum, cout;
  full_adder FA(sum, cout, a, b, c)
  initial
  begin
  $monitor($time, "a=%b, b=%b,
  c=%b, sum=%b, cout=%b",
    a, b, c, sum, cout);
  #5 a=0; b=0; c=1;
  #5 b=1;
  #5 a=1;
  #5 a=0; b=0; c=0;
  end initial
endmodule
```

Task day 2 Implement a 4:1 mux and write the test bench code to verify the module



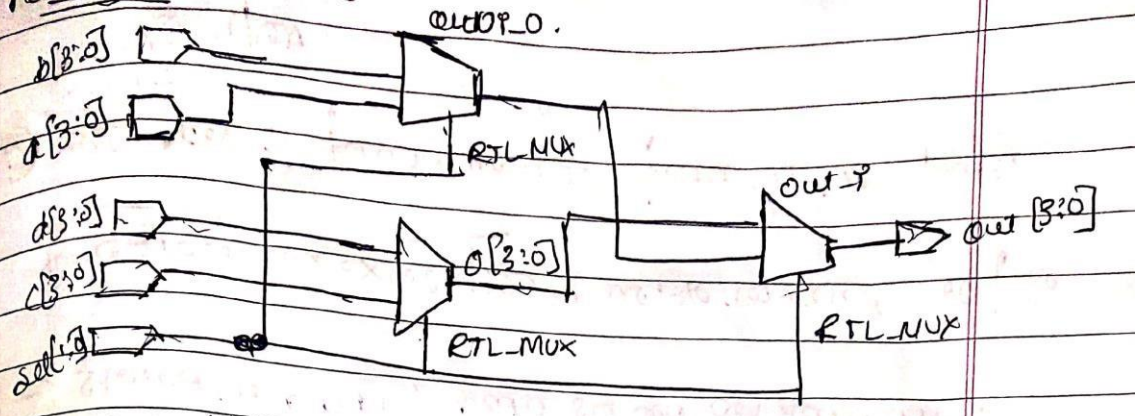
- using assign statement

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

using case statement

```
module mux_4to1_case (input [3:0] a,
                      input [3:0] b,
                      input [3:0] c,
                      input [3:0] d,
                      input [1:0] sel,
                      output reg [3:0] out);
always @(a or b or c or d or sel) begin.
  case (sel)
    2'b00 : out <= a;
    2'b01 : out <= b;
    2'b10 : out <= c;
    2'b11 : out <= d;
  endcase
end
endmodule.
```

Hardware Schematic

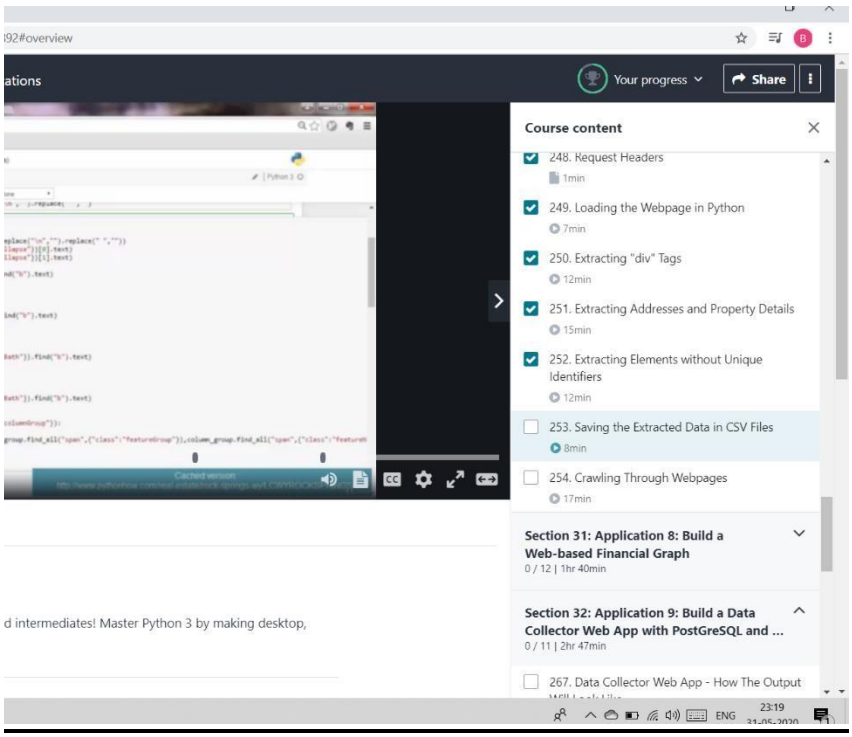



```
module tb_4to1_mux;
reg [3:0] a;
reg [3:0] b;
reg [3:0] c;
reg [3:0] d;
wire [3:0] out;
reg [1:0] sel;
integer i;
mux_4to1_raw mux0 ( .a(a)
                    .b(b),
                    .c(c),
                    .d(d),
                    .sel(sel)
                    .out(out));
Initial begin
    $monitor ("[%0t] sel =0x%0h a=0x%0h b=0x%0h c=0x%0h
              d=0x%0h out=0x%0h", $time, sel, a,
    sel <=0;
    a <= $random;
    b <= $random;
    c <= $random;
    d <= $random;
    for(i=1; i<4 ; i=i+1) begin.
    end
    #5 $finish;
end module
```

Date:2june2020 **Name:poorvi j**

Course: python **USN:4al17ec071**

Topic: sec30 **Sem&Sec:6th b**

AFTERNOON SESSION DETAILS
Image of session 

8-06-2020 Sec: 30. Scrape Real Estate property Data from the web

→ Jupyter Notebook

In[1] 1. import request
2. from BS4 import BeautifulSoup

In[2] r= requests.get ("http://www.century21.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/")
C= r.content

In[3] Soup = BeautifulSoup (C, "html.parser")
Print

In[4] all = Soup.find_all ("div", {"class": "propertyRow"})

In[5] all[0].find ("h4", {"class": "propPrice"}).text.replace ("\n", "").replace (" ", "")

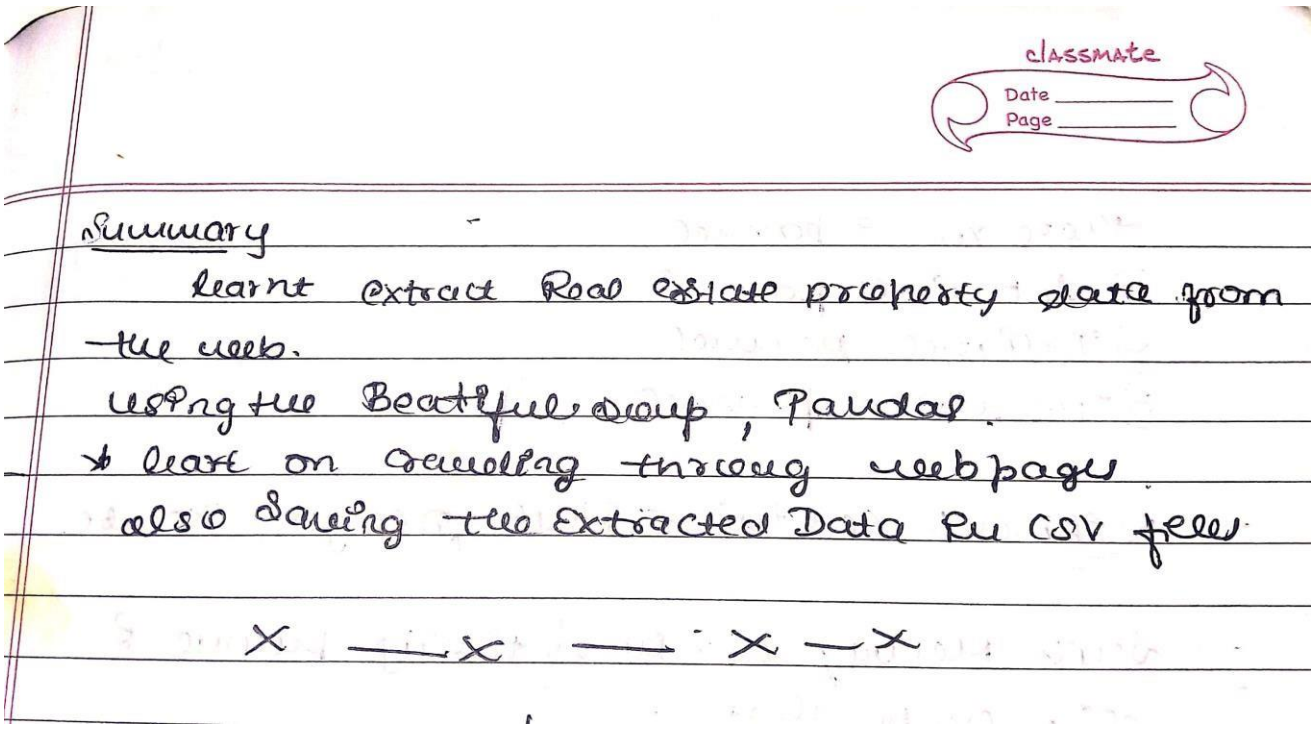
Op: $725,000

In[ ]: for item in all:

```
print(item.find("h4", {"class": "propPrice"}).
      text.replace("\n", " ").replace(" ", " "))
print(item.find_all("span", {"class": "propAddressCollapse"})
      [0].text)
print(item.find_all("span", {"class": "propAddressCollapse"})
      [1].text)
try:
    print(item.find("span", {"class": "RufooBeg"}).
          find("b").text)
except:
    print(None)
print("")
```

* Extracting Elements without unique.

```
for column_group in item.find_all("div", {"class":
    "column-group"}):
    print(column_group)
    for feature_group, feature_name in zip(
    column_group.find_all("span", {"class":
    "feature-group"}), column_group.find_all
    ("span", {"class": "feature-name"})):
    print(feature_group.text, feature_name.text)
    if "Hot Spot" in feature_group.text:
        print(feature_name.text)
print("")
```



Python excersise program:

```
import cv2, time

first_frame = None

video = cv2.VideoCapture(0)

while True:
    check, frame = video.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray,(21,21),0)

    if first_frame is None:
        first_frame = gray
```

```
    delta_frame = cv2.absdiff(first_frame, gray)
    thresh_frame = cv2.threshold(delta_frame, 30, 255, cv2.THRESH_BINARY)[1]
    thresh_frame = cv2.dilate(thresh_frame, None, iterations = 2)

    (cnts, _) = cv2.findContours(thresh_frame.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    for contour in cnts:
        if cv2.contourArea(contour) < 1000:
            continue
        (x, y, w, h) = cv2.boundingRect(contour)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)

    cv2.imshow("Gray Frame", gray)
    cv2.imshow("Delta Frame", delta_frame)
    cv2.imshow("Threshold Frame", thresh_frame)
    cv2.imshow("Color Frame", frame)

    key = cv2.waitKey(1)
    print(gray)
    print(delta_frame)

    if key == ord('q'):
        break

video.release()
cv2.destroyAllWindows
```