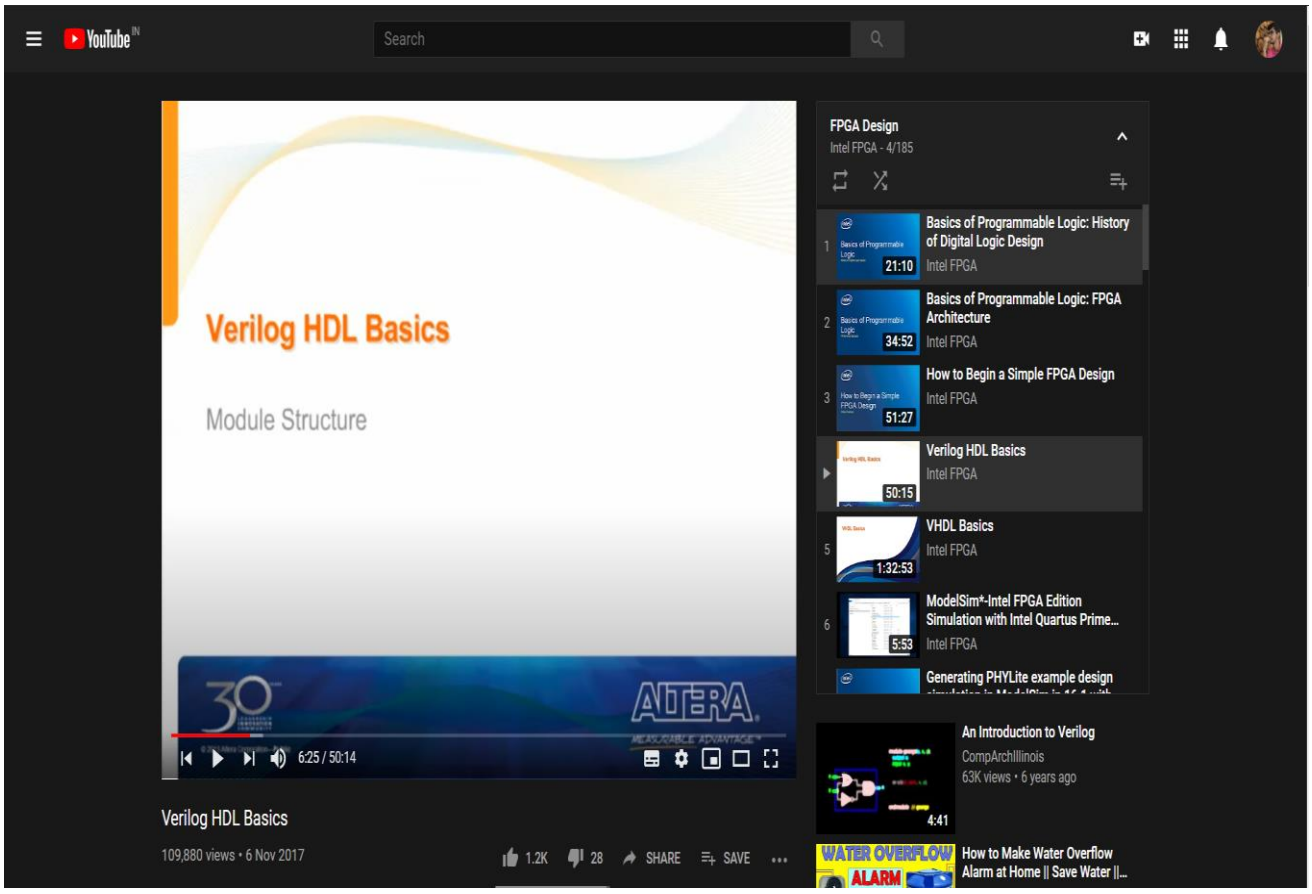
DAILY ASSESSMENT REPORT

Date:	**02 June 2020**	**Name:**	**Gagan M K**
Course:	**DIGITAL DESIGN USING HDL**	**USN:**	**4AL17EC032**
Topic:	• **FPGA Basics: Architecture, Applications and Uses** • **Verilog HDL Basics by Intel** • **Verilog Test bench code to verify the design under test (DUT)**	**Semester & Section:**	**6th sem & 'A' sec**
Github Repository:	**Alvas-education-foundation/Gagan-Git**		

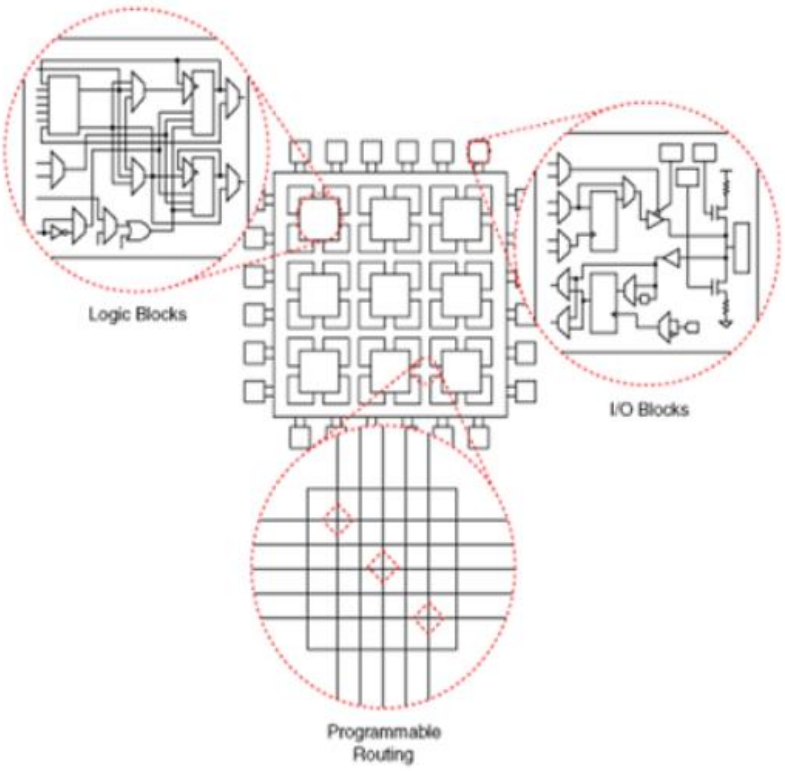
FORENOON SESSION DETAILS
Image of session



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

FPGA Basics: Architecture, Applications and Uses:

- **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).**



Application:

- **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.**

Verilog HDL Basics by Intel:

- **Verilog is a HARDWARE DESCRIPTION LANGUAGE (HDL). It is a language used for describing a digital system like a network switch or a microprocessor or a memory or a flip–flop.**
- **It means, by using a HDL we can describe any digital hardware at any level. Designs, which are described in HDL are independent of technology, very easy for designing and debugging, and are normally more useful than schematics, particularly for large circuits.**
- **Behavioral level**
- **Register-transfer level**
- **Gate level**
- **Lexical Tokens**
- **Numbers**
- **Identifiers**
- **Operators**
- **Data Types**
- **Operators**
- **Operands**
- **Modules**

Verilog Test bench code to verify the design under test (DUT):

```
module sillyfunction(input  a, b, c,
                     output y);


  assign y = ~b & ~c | a & ~b;
endmodule
```

```
module testbench1(); // Testbench has no inputs, outputs
  reg  a, b, c;      // Will be assigned in initial block
  wire y;

  // instantiate device under test
  sillyfunction dut (.a(a), .b(b), .c(c), .y(y) );d

  // apply inputs one at a time
  initial begin                  // sequential block
    a = 0; b = 0; c = 0; #10;    // apply inputs, wait 10ns
    c = 1; #10;                  // apply inputs, wait 10ns
    b = 1; c = 0; #10;           // etc .. etc..
    c = 1; #10;
    a = 1; b = 0; c = 0; #10;
  end
endmodule
```

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

Multiplexer(4:1)

Verilog design:

```
module mux41(
   input i0,i1,i2,i3,sel0,sel1,
   output reg y);

   always @(*)
   begin
     case ({sel0,sel1})
     2'b00 : y = i0;
     2'b01 : y = i1;
     2'b10 : y = i2;
     2'b11 : y = i3;
     endcase
   end

endmodule
```

TestBench:

```
module tb_mux41;

  reg I0,I1,I2,I3,SEL0,SEL1;
  wire Y;

  mux41 MUX (.i0(I0),.i1(I1),.i2(I2),.i3(I3),.sel0(SEL0),.sel1(SEL1),.y(Y));

  initial begin
     I0 =1'b0;
     I1= 1'b0;
     I2 =1'b0;
```

```
    I3 =1'b0;
    SEL0 =1'b0;
    SEL1 =1'b0;
    #45 $finish;
  end

  always #2 I0 = ~I0;
  always #4 I1 =~I1;
  always #6 I2 =~I1;
  always #8 I3 =~I1;
  always #3 SEL0 = ~SEL0;
  always #3 SEL1 = ~SEL1;

  always @(Y)
  $display( "time =%0t INPUT VALUES: \t I0=%b I1 =%b I2 =%b I3 =%b SEL0 =%b SEL1
=%b \t output value Y =%b ",$time,I0,I1,I2,I3,SEL0,SEL1,Y);

endmodule
```

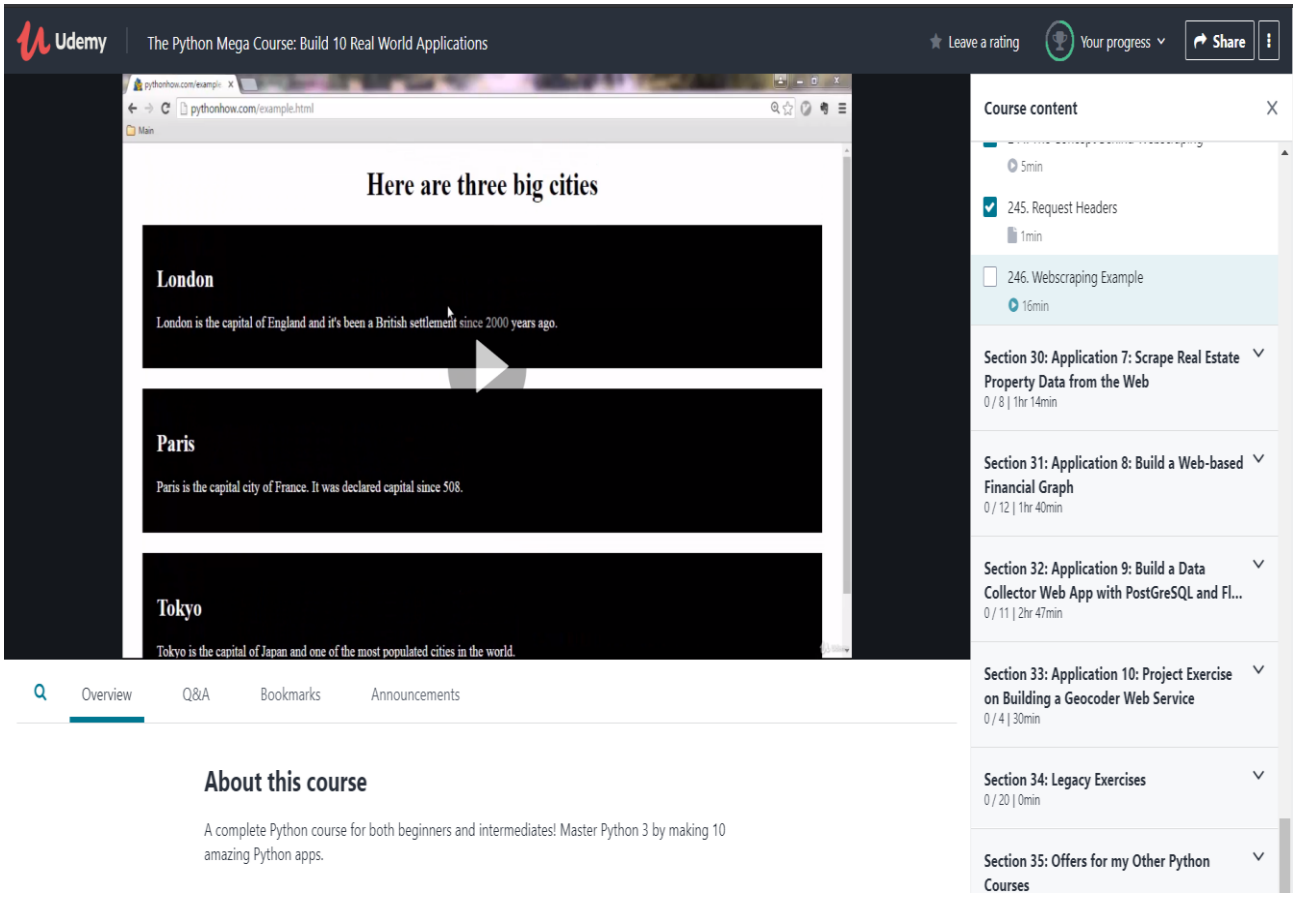
output:

```
time =0 INPUT VALUES:    I0=0 I1 =0 I2 =0 I3 =0 SEL0 =0 SEL1 =0
output value Y =0
time =2 INPUT VALUES:    I0=1 I1 =0 I2 =0 I3 =0 SEL0 =0 SEL1 =0
output value Y =1
time =3 INPUT VALUES:    I0=1 I1 =0 I2 =0 I3 =0 SEL0 =1 SEL1 =1
output value Y =0
time =6 INPUT VALUES:    I0=1 I1 =1 I2 =0 I3 =0 SEL0 =0 SEL1 =0
output value Y =1
time =8 INPUT VALUES:    I0=0 I1 =0 I2 =0 I3 =0 SEL0 =0 SEL1 =0
output value Y =0
time =14 INPUT VALUES:   I0=1 I1 =1 I2 =1 I3 =0 SEL0 =0 SEL1 =0
output value Y =1
time =15 INPUT VALUES:   I0=1 I1 =1 I2 =1 I3 =0 SEL0 =1 SEL1 =1
output value Y =0
```

Date:	02 June 2020	Name:	Gagan M K
Course:	The Python Mega Course	USN:	4AL17EC032
Topic:	• Interactive Data Visualization with Bokeh • Webscraping with Python Beautiful Soup	Semester & Section:	6th sem & 'A' sec

AFTERNOON SESSION DETAILS

Image of session:



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

Interactive Data Visualization with Bokeh:

- **If you haven't installed Bokeh yet, you can easily install it with pip from the terminal:**
- **pip install bokeh**
- **Using this we learnt to plot bokeh graph which are interactive in nature.**
- **Plotted line, Triangle, Circle graphs.**
- **Plotted Educational graph using data provided.**
- **Plotted Weather graph using given data.**
- **Some pictures of graphs obtained are shown below.**