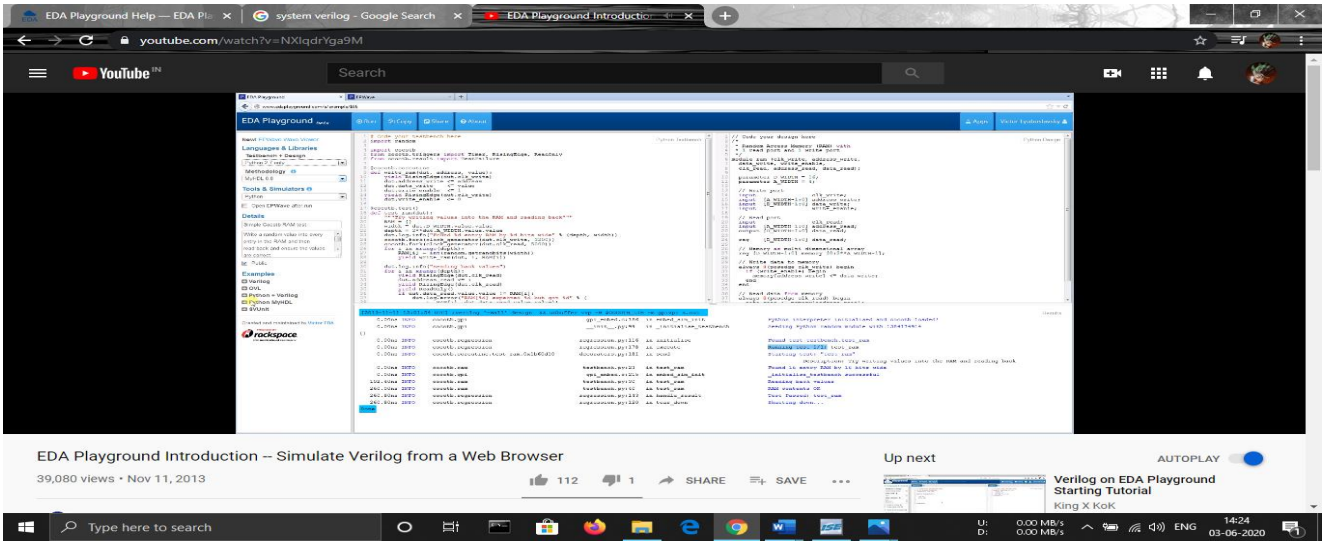
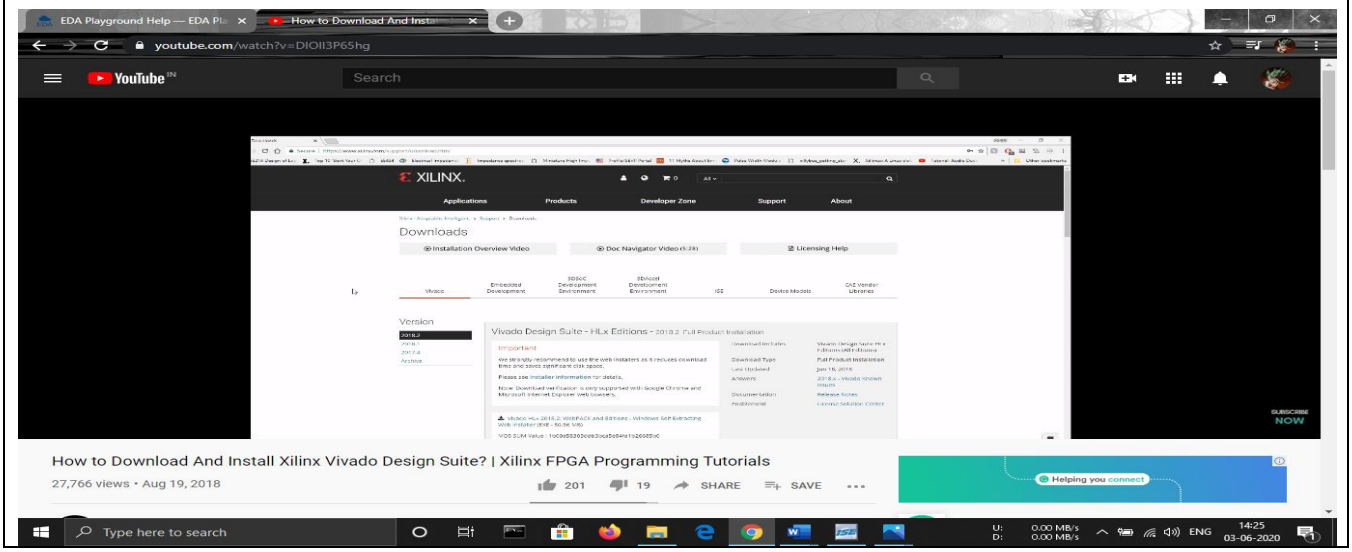
DAILY ASSESSMENT REPORT

Date:	**03/06/2020**	**Name:**	**Abhishek M Shastry K**
Subject:	**Digital Design Using HDL**	**USN:**	**4AL17EC002**
Topic:	**1] EDA Playground Online complier 2] EDA Playground Tutorial Demo Video 3] How to Download and Install Xilinx Vivado Design Suite 4] Vivado Design Suite for implementation of HDL code**	**Semester & Section:**	**6th 'A'**
Github Repository:	**AbhishekShastry-Courses**		

FORENOON SESSION DETAILS

Image of session





Report

EDA Playground

- EDA Playground gives engineers immediate hands-on exposure to simulating SystemVerilog, Verilog, VHDL, C++/SystemC, and other HDLs. All you need is a web browser. The goal is to accelerate learning of design/testbench development with easier code sharing and simpler access to EDA tools and libraries.
    - ✓ With a simple click, run your code and see console output in real time.
    - ✓ View waves for your simulation using EPWave browser-based wave viewer.
    - ✓ Save your code snippets (“Playgrounds”).
    - ✓ Share your code and simulation results with a web link. Perfect for web forum discussions or emails. Great for asking questions or sharing your knowledge.
    - ✓ Try out a language feature with a small example.
    - ✓ Try out a library that you’re thinking of using.
- **Tools and simulators in EDA Playground**
- **Simulators**
    - ✓ Synopsys VCS.
    - ✓ Cadence Incisive.
    - ✓ Aldec Riviera-PRO.
    - ✓ Incisive Specman Elite.
    - ✓ GHDL.
    - ✓ Icarus Verilog.
    - ✓ GPL Cver.
    - ✓ VeriWell.
- **Compilers and Interpreters**
    - ✓ C++.
    - ✓ Perl.
    - ✓ Python.
    - ✓ Csh (C Shell).

- **Synthesis Tools**
  - ✓ Yosys.
  - ✓ The Verilog-to-Routing (VTR) Project.

Xilinx Vivado Design Suite

- The Vivado® Design Suite offers a new approach for ultra-high productivity with next generation C/C++ and IP-based design. The new HLx editions include HL System Edition, HL Design Edition and HL WebPACK™ Edition. When coupled with the new UltraFast™ High-Level Productivity Design Methodology Guide, users can realize a 10-15X productivity gain over traditional approaches.
- Unlike traditional RTL-based design where the majority of the design effort is spent in the backend of the design process, C and IP based design allows for reduced development cycles in verification,
- Implementation and design convergence, so designers can focus on their differentiated logic. This flow includes:
  - ✓ Rapid generation of the platform connectivity design, along with the necessary software stack.
  - ✓ Rapid differentiated logic development using high-level design. This also enables superior design reuse capabilities.
  - ✓ Dramatically shortened verification times from high-level languages, compared to RTL.
- Using high levels of abstraction, design teams can quickly get overall better or equal Quality of Results (performance, power, utilization).
- The new Vivado HLx Editions offers a new approach for ultra-high productivity with next generation platform design automation, C/C++ programming of the differentiated logic, with graphical system assembly. This approach, described in the UltraFast HighLevel Productivity Design Methodology Guide (UG1197), is proven to accelerate design creation and verification by 15x over RTL-based methodologies.

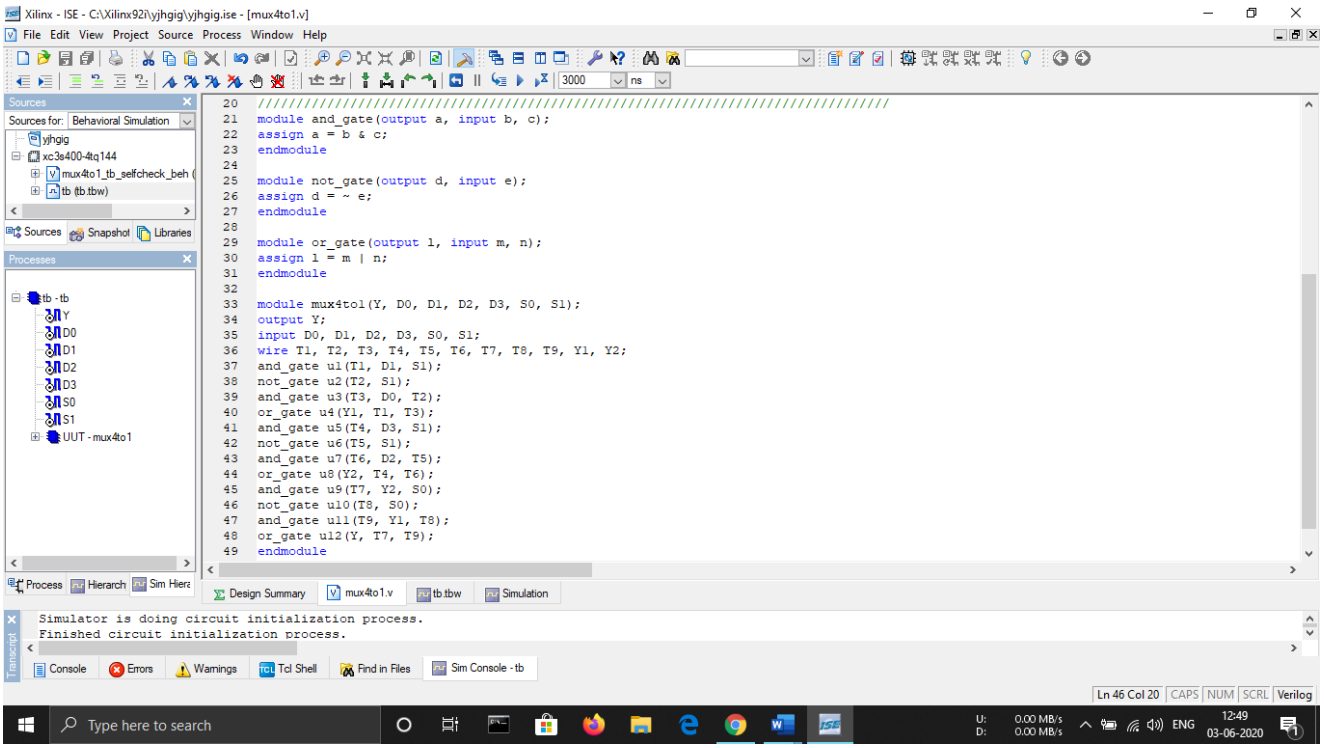
Task (DAY - 3)

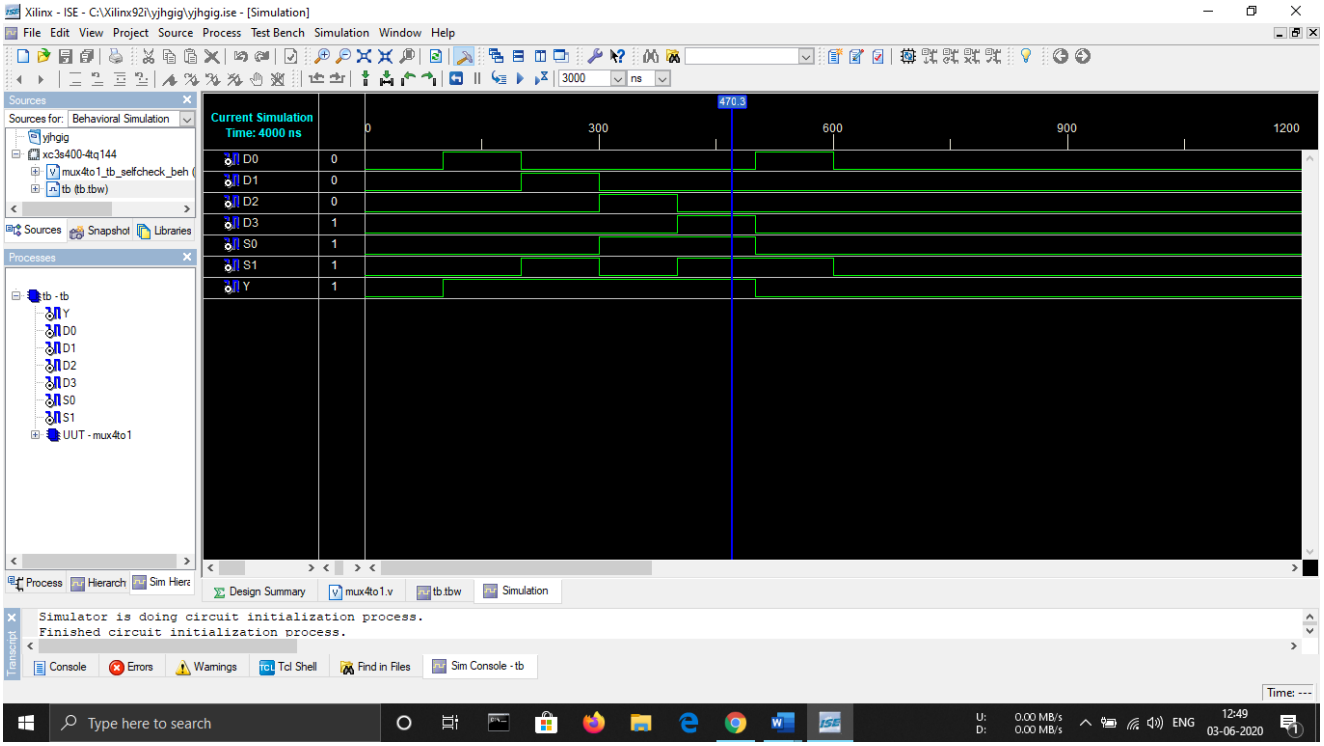
Implement 4 to 1 MUX using two 2 to 1 MUX using structural modelling style and test the module in online/offline compiler.

Verilog Code:

```
module and_gate (output a, input b, c);
assign a = b & c;
endmodule

module not_gate (output d, input e);
assign d = ~ e;
endmodule

module or_gate (output l, input m, n);
assign l = m | n;
endmodule

module mux4to1 (Y, D0, D1, D2, D3, S0, S1);
output Y;
input D0, D1, D2, D3, S0, S1;
wire T1, T2, T3, T4, T5, T6, T7, T8, T9, Y1, Y2;

//Mux 2:1 - 1
and_gate u1 (T1, D1, S1);
not_gate u2 (T2, S1);
and_gate u3 (T3, D0, T2);
or_gate u4 (Y1, T1, T3);

//Mux 2:1 - 2
and_gate u5 (T4, D3, S1);
not_gate u6 (T5, S1);
and_gate u7 (T6, D2, T5);
or_gate u8 (Y2, T4, T6);

//Mux 2:1 - 3
and_gate u9 (T7, Y2, S0);
not_gate u10 (T8, S0);
and_gate u11 (T9, Y1, T8);
or_gate u12 (Y, T7, T9);
endmodule
```

Compiler Output:

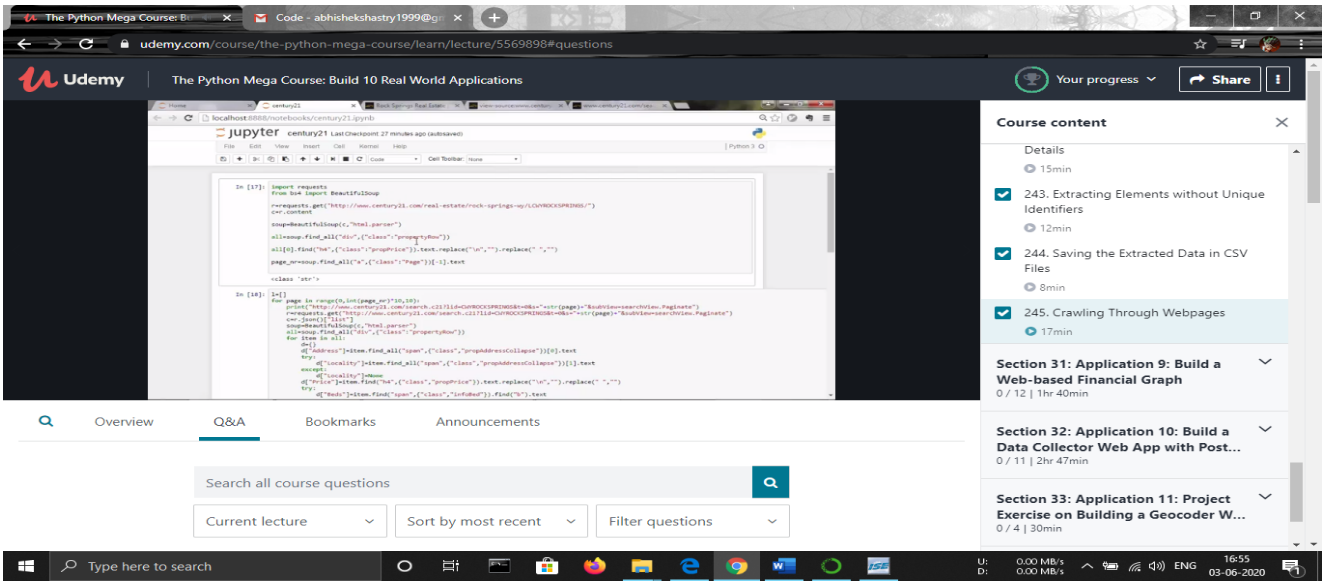


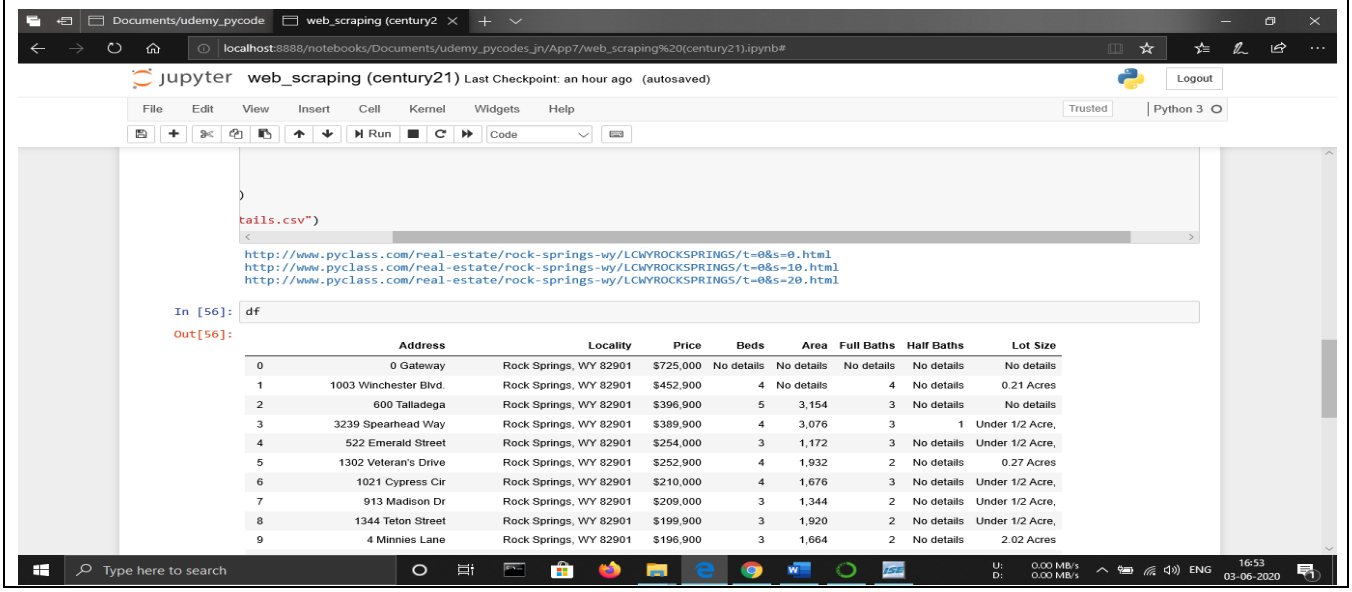


Date:	03/06/2020	Name:	Abhishek M Shastry K
Course:	The Python Mega Course: Build 10 Real World Applications	USN:	4AL17EC002
Topic:	1] Web scraping with Python Beautiful Soup 2] Application 7: Scrape Real Estate Property Data from the Web	Semester & Section:	6th 'A'
Github Repository:	AbhishekShastry-Courses		

AFTERNOON SESSION DETAILS

Image of session





Report

Web scraping with Python Beautiful Soup

- Introduction to web scraping using **BeautifulSoup** from **bs4** library and **requests** module.
- **BeautifulSoup** is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.
- The **requests** module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc.).
- **Web scraping** is a term used to describe the use of a program or algorithm to extract and process large amounts of data from the web. Whether you are a data scientist, engineer, or anybody who analyzes large amounts of datasets, the ability to scrape data from the web is a useful skill to have. Let's say you find data from the web, and there is no direct way to download it, web scraping using Python is a skill you can use to extract the data into a useful form that can be imported.

Application 7: Scrape Real Estate Property Data from the Web

- Python program to extract details of plots from century21 website and store the details in a **csv file** using **pandas** library.
- Some of the functions used under BeautifulSoup of bs4 library:
    - ✓ The **find_all ()** method scans the entire document looking for results, but sometimes you only want to find one result. If you know a document only has one <body> tag, it's a waste of time to scan the entire document looking for more. Rather than passing in limit = 1 every time you call find_all, you can use the **find ()** method.
    - ✓ The **prettify ()** method will turn a Beautiful Soup parse tree into a nicely formatted Unicode string, with a separate line for each tag and each string.
    - ✓ To extract text from the body tag **.text** method is used.