**3 June 2020**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **03 June 2020** | **Name:** | **Srinidhi J C** |
| **Course:** | **Digital designing using hdl** | **USN:** | **4al16ec078** |
| **Topic:** | **About EDA** | **Semester & Section:** | **8th & b** |
| **Github Repository:** | **SrinidhiJC078** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report – Report can be typed or hand written for up to two pages.**  **EDA Playground Help**  **Quick Start**  **Log in. Click the Log in button (top right) Then either**  **click on Google or Facebook or**  **register by clicking on ‘Register for a full account’ (which enables all the simulators on EDA Playground)**  **Select your language from the Testbench + Design menu.**  **Select your simulator from the Tools & Simulators menu. Using certain simulators will require you to supply additional identifcation information.**  **Type in your code in the testbench and design windows.**  **Click Run.**  **Tutorial <http://eda-playground.readthedocs.io/en/latest/tutorial.html>**  **EDA Playground on YouTube - Tutorials for Verilog, SystemVerilog, UVM, and VHDL, interview questions, news and features, etc.**  **What is 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.**  **Quickly try something out**  **Try out a language feature with a small example.**  **Try out a library that you’re thinking of using.**    **EDA Playground Tutorial Demo Video**    **How to Download And Install Xilinx Vivado Design Suite**  **Vivado Design Suite - HLx Editions Update 1 - 2019.2**  **Important Information**  **This is a common updater. You do not need to re-run it for Vitis if you have already run it for Vivado and vice versa.**  **Vivado Design Suite 2019.2.1 is now available with support for:**  **Additional Zynq UltraScale+ RFSoCs devices enabled:- (XCZU46DR, XCZU47DR, XCZU48DR, XCZU49DR)**  **For customers using these devices, Xilinx recommends installing Vivado 2019.2.1. For other devices, please continue to use Vivado 2019.2.**  **Note: Download verification is only supported with Google Chrome and Microsoft Internet Explorer web browsers.**    **Vivado Design Suite for implementation of HDL code**  **Introduction**  **Synthesis is the process of transforming an RTL-specified design into a gate-level**  **representation. Vivado® synthesis is timing-driven and optimized for memory usage and**  **performance. Vivado synthesis supports a synthesizeable subset of:**  **• SystemVerilog: IEEE Standard for SystemVerilog-Unified Hardware Design,**  **Specification, and Verification Language (IEEE Std 1800-2012)**  **• Verilog: IEEE Standard for Verilog Hardware Description Language (IEEE Std 1364-2005)**  **• VHDL: IEEE Standard for VHDL Language (IEEE Std 1076-2002)**  **• VHDL 2008**  **• Mixed languages: Vivado supports a mix of VHDL, Verilog, and SystemVerilog.**  **In most instances, the Vivado tools also support Xilinx design constraints (XDC), which is**  **based on the industry-standard Synopsys design constraints (SDC).**  **Synthesis Methodology**  **The Vivado IDE includes a synthesis and implementation environment that facilitates a push button flow with synthesis and implementation runs. The tool manages the run data automatically, allowing repeated run attempts with varying Register Transfer Level (RTL) source versions, target devices, synthesis or implementation options, and physical or timing constraints.**  **Within the Vivado IDE, you can do the following:**  **• Create and save a strategy. A strategy is a configuration of command options that you can apply to design runs for synthesis or implementation. See Creating Run Strategies.**  **• Queue the synthesis and implementation runs to launch sequentially or simultaneously with multi-processor machines. See Running Synthesis.**  **• Monitor synthesis or implementation progress, view log reports, and cancel runs. See Monitoring the Synthesis Run.**    **Task 3**  **Implement 4 to 1 MUX using two 2 to 1 MUX using structural modelling style and test the module in online/offline compiler.**    **module mux4to1(a,sel,out);**  **input [3:0] a;**  **input [1:0] sel;**  **output out;**  **wire mux[2:0];**  **mux2to1 m1 (a[3],a[2],sel[0],mux\_1),**  **m2 (a[1],a[4],sel[0],mux\_2),**  **m3 (mux\_1,mux\_2,sel[1],out);**  **endmodule.** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **03 June 2020** | **Name:** | **Srinidhi J C** | |
| **Course:** | **Python** | **USN:** | **4al16ec078** | |
| **Topic:** |  | **Semester & Section:** | **8th & b** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Report – Report can be typed or hand written for up to two pages.**  **Request Headers**  **Note**  **Whenever I use this code in the next videos:**  **r = requests.get("http://www.pythonhow.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/")**  **please use this instead:**  **r = requests.get("http://www.pyclass.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/", 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.** | | | |