**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **2/06/2020** | **Name:** | **Yalpi Nandika** |
| **Course:** | **Digital Design Using HDL** | **USN:** | **4AL17EC096** |
| **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**  **B section** |
| **GitHub Repository** | **Yalpi-Online-Courses** |  |  |

|  |
| --- |
| **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 Verilog HDL Basics by intel Verilog Test bench code to verify the design under test (DUT).**  **\* Specific application of an FPGA includes digital signal processing, bioinformatics, device controllers, software-defined radio, random logic, ASIC prototyping, medical imaging, computer hardware emulation, integrating multiple SPLDs, voice recognition, cryptography, filtering and communication encoding and many more.**  **\* FPGAs are particularly useful for prototyping application-specific integrated circuits (ASICs) or processors. An FPGA can be reprogrammed until the ASIC or processor design is final and bug-free and the actual manufacturing of the final ASIC begins. Intel itself uses FPGAs to prototype new chips.**  **\* The FPGA is Field Programmable Gate Array. It is a type of device that is widely used in electronic circuits. FPGAs are semiconductor devices which contain programmable logic blocks and interconnection circuits. It can be programmed or reprogrammed to the required functionality after manufacturing.**  **\* The Device Under Test (D.U.T.) In this example, the DUT is behavioral Verilog code for a 4-bit counter found in Appendix A. This is also known as a Register Transfer Level or RTL description of the design. In the HDL source, all the input and output signals are declared in the port list.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **2/06/2020** | **Name:** | **Yalpi Nandika** |
| **Course:** | **Python** | **USN:** | **4AL17EC096** |
| **Topic:** | **Timing your code-timeit**  **Regular Expressions**  **StrinIO** | **Semester & Section:** | **6th sem**  **B section** |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session**      **Timing your code-timeit Regular Expressions StrinIO**  **\* Python provides the timeit module for measuring the execution time of small code snippits. This can be called from the command line, or by importing it into an exisiting program. The timeit function will run the code a set number of times (in this case 100000) and then report how long it took to run (in seconds).**  **\* timeit(stmt, setup, timer, number) accepts four arguments: stmt which is the statement you want to measure; it defaults to 'pass'. setup which is the code that you run before running the stmt; it defaults to 'pass'. We generally use this to import the required modules for our code.** | | | |