**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **04/06/2020** | **Name:** | **Akshata Madiwalar** |
| **Course:** | **HDL** | **USN:** | **4AL17EC046** |
| **Topic:** | **1. Hardware modelling using verilog**  **2. FPGA and ASIC Interview questions** | **Semester & Section:** | **6th sem & A sec** |
| **Github Repository:** | **Akshata-course** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Report – Report can be typed or hand written for up to two pages.** |
| **1) Write a verilog code to swap contents of two registers with and without a temporary register?**  **With temp reg ;**  **always @ (posedge clock)**  **begin**  **temp=b;**  **b=a;**  **a=temp;**  **end**  **Without temp reg;**  **always @ (posedge clock)**  **begin**  **a <= b;**  **b <= a;**  **end**  **2) Difference between blocking and non-blocking?(Verilog interview questions that is most commonly asked)**  **The Verilog language has two forms of the procedural assignment statement: blocking and non-blocking. The two are distinguished by the = and <= assignment operators. The blocking assignment statement (= operator) acts much like in traditional programming languages. The whole statement is done before control passes on to the next statement. The non-blocking (<= operator) evaluates all the right-hand sides for the current time unit and assigns the left-hand sides at the end of the time unit. For example, the following Verilog program**  **// testing blocking and non-blocking assignment**  **module blocking;**  **reg [0:7] A, B;**  **initial begin: init1**  **A = 3;**  **#1 A = A + 1; // blocking procedural assignment**  **B = A + 1;**  **$display("Blocking: A= %b B= %b", A, B ); A = 3;**  **#1 A <= A + 1; // non-blocking procedural assignment**  **B <= A + 1;**  **#1 $display("Non-blocking: A= %b B= %b", A, B );**  **end**  **endmodule**  **produces the following output:**  **Blocking: A= 00000100 B= 00000101**  **Non-blocking: A= 00000100 B= 00000100**  **The effect is for all the non-blocking assignments to use the old values of the variables at the beginning of the current time unit and to assign the registers new values at the end of the current time unit. This reflects how register transfers occur in some hardware systems.**  **blocking procedural assignment is used for combinational logic and non-blocking procedural assignment for sequential**  **3) Difference between task and function?**  **Function:**  **A function is unable to enable a task however functions can enable other functions.**  **A function will carry out its required duty in zero simulation time. ( The program time will not be incremented during the function routine)**  **Within a function, no event, delay or timing control statements are permitted**  **In the invocation of a function their must be at least one argument to be passed.**  **Functions will only return a single value and can not use either output or inout statements.**  **Tasks:**  **Tasks are capable of enabling a function as well as enabling other versions of a Task**  **Tasks also run with a zero simulation however they can if required be executed in a non zero simulation time.**  **Tasks are allowed to contain any of these statements.**  **A task is allowed to use zero or more arguments which are of type output, input or inout.**  **A Task is unable to return a value but has the facility to pass multiple values via the output and inout statements** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **04/06/2020** | **Name:** | **Akshata Madiwalar** |
| **Course:** | **Python** | **USN:** | **4AL17EC046** |
| **Topic:** | **Build a web based financial graph** | **Semester & Section:** | **6th sem & A sec** |
| **AFTERNOON SESSION DETAILS** | | | |
|  | | | |
|  | | | |