**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **05-06-2020** | **Name:** | **Anand kumar k** |
| **Course:** |  | **USN:** | **4al16ec002** |
| **Topic:** | **D flipflop** | **Semester & Section:** | **8thsem ‘A’ sec** |
| **Github Repository:** | **Anand-courses** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**      **Introduction**  Verilog is a HARDWARE DESCRIPTION LANGUAGE (HDL). A hardware description Language is a language used to describe a digital system, for example, a network switch, a microprocessor or a memory or a simple flip−flop. This just means that, by using a HDL one can describe any hardware (digital ) at any level.  // D flip−flop Code  module d\_ff ( d, clk, q, q\_bar);  input d ,clk;  output q, q\_bar;  wire d ,clk;  reg q, q\_bar;  always @ (posedge clk)  begin  q <= d;  q\_bar <= !d;  end  endmodule  One can describe a simple Flip flop as that in above figure as well as one can describe a complicated designs having 1 million gates. Verilog is one of the HDL languages available in the industry for designing the Hardware. Verilog allows us to design a Digital design at Behavior Level,Register Transfer Level (RTL), Gate level and at switch level. Verilog allows hardware designers to express their designs with behavioral constructs, deterring the details of implementation to a later stage of design in the final design.  Many engineers who want to learn Verilog, most often ask this question, how much time it will take to learn Verilog?, Well my answer to them is "It may not take more then one week, if you happen to know at least one programming language".  Design Styles Verilog like any other hardware description language, permits the designers to design a design in either Bottom−up or Top−down methodology.  module num\_zero(input [15:0]A, output reg [4:0]zeros);  integer i;  always@(A)  begin  zeros=0;  for(i=0;i<16;i=i+1)  zeros=zeros+A[i];  end  endmodule  **test bench code**  module test;  reg [15:0]A;  wire [4:0] zeros;  num\_zero out (.A(A), .zeros(zeros));  initial begin  $dumpfile("dumo.vcd");  $dumpvars(1,test);  A=16'hFFFF; #100;  A=16'hF56F; #100;  A=16'h3FFF; #100;  A=16'h0001; #100;  A=16'hF10F; #100;  A=16'hF822; #100;  A=16'h7ABC; #100;  end  endmodule |
|  |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **05-06-2020** | **Name:** | **Anand kumar k** | |
| **Course:** |  | **USN:** | **4al16ec002** | |
| **Topic:** | **python** | **Semester & Section:** | **8thsem ‘A’ sec** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| Flask startup and configuration Like most widely used Python libraries, the Flask package is installable from the Python Package Index (PPI). First create a directory to work in (something like flask\_todo is a fine directory name) then install the flask package. You'll also want to install flask-sqlalchemy so your Flask application has a simple way to talk to a SQL database.A good way to get moving is to turn the codebase into an installable Python distribution. At the project's root, create setup.py and a directory called todo to hold the source code. The setup.py should look something like this:  requires = [  'flask',  'flask-sqlalchemy',  'psycopg2',  ]  setup(  name='flask\_todo',  version='0.0',  description='A To-Do List built with Flask',  author='<Your actual name here>',  author\_email='<Your actual e-mail address here>',  keywords='web flask',  packages=find\_packages(),  include\_package\_data=True,  install\_requires=requires  )  This way, whenever you want to install or deploy your project, you'll have all the necessary packages in the requires list. You'll also have everything you need to set up and install the package in sitepackages. For more information on how to write an installable Python distribution, check out the docs on setup.py.Within the todo directory containing your source code, create an app.py file and a blank \_\_init\_\_.py file. The \_\_init\_\_.py file allows you to import from todo as if it were an installed package. The app.py file will be the application's root. This is where all the Flask application goodness will go, and you'll create an environment variable that points to that file. If you're using pipenv (like I am), you can locate your virtual environment with pipenv --venv and set up that environment variable in your environment's activate script. | | | |