**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | 5 June 2020 | **Name:** | Anupama J S |
| **Course:** | Digital design using HDL | **USN:** | 4AL16EC005 |
| **Topic:** | 1. Verilog tutorials & practice programs 2. Building/ demo projects using FPGA | **Semester & Section:** | 8th sem “A”section |
| **Github Repository:** | AnupamaJS |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **C:\Users\User\Downloads\WhatsApp Image 2020-06-05 at 7.54.40 PM.jpegC:\Users\User\Downloads\WhatsApp Image 2020-06-05 at 7.54.41 PM.jpeg**  **C:\Users\User\Downloads\WhatsApp Image 2020-06-05 at 7.54.41 PM (1).jpegC:\Users\User\Downloads\WhatsApp Image 2020-06-05 at 7.54.41 PM (2).jpeg**  FPGA stands for field-programmable gate array. That’s quite a mouthful, so let’s start with a basic definition. Essentially, an FPGA is a hardware circuit that a user can program to carry out one or more logical operations. Taken a step further, FPGAs are integrated circuits, or ICs, which are sets of circuits on a chip—that’s the “array” part. Those circuits, or arrays, are groups of programmable logic gates, memory, or other elements. With a standard chip, such as the Intel Curie module in an Arduino board or a CPU in your laptop, the chip is fully baked. It can’t be programmed; you get what you get. With these chips, a user can write software that loads onto a chip and executes functions. That software can later be replaced or deleted, but the hardware chip remains unchanged. With an FPGA, there is no chip. The user programs the hardware circuit or circuits. The programming can be a single, simple logic gate (an AND or OR function), or it can involve one or more complex functions, including functions that, together, act as a comprehensive multi-core processor. Why Use an FPGA? You might use an FPGA when you need to optimize a chip for a particular workload, or when you are likely to need to make changes at the chip level later on. Uses for FPGAs cover a wide range of areas—from equipment for video and imaging, to circuitry for computer, auto, aerospace, and military applications, in addition to electronics for specialized processing and 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. In fact, Intel recently acquired a company called eASIC as a way to accelerate its designing and prototyping process. eASIC produces something called a “structured ASIC,” which relies on a model that is in between an ASIC and an FPGA. As this [AnandTech article](https://www.anandtech.com/show/13075/intel-acquires-easic-lower-cost-asics-in-fpga-design-time) explains, with a structured ASIC:  “Engineers can create a design using an FPGA, then rather than spending time optimizing the circuit layout, they bake the fixed layout into a single design mask for manufacturing. By being a fixed design like an ASIC, it is faster than a variable design, but without the die area benefits of ASIC-like power savings. However, it was designed in FPGA time, rather than ASIC time (up to six months saved), and saves power through its fixed design.”  **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.  **Bottom−Up Design**  The traditional method of electronic design is bottom−up. Each design is performed at the gate−level using the standard gates ( Refer to the Digital Section for more details) With increasing complexity of new designs this approach is nearly impossible to maintain. New systems consist of ASIC or microprocessors with a complexity of thousands of transistors. These traditional bottom−up designs have to give way to new structural, hierarchical design methods. Without these new design practices it would be impossible to handle the new complexity.  **Top−Down Design**  The desired design−style of all designers is the top−down design. A real top−down design allows early testing, easy change of different technologies, a structured system design and offers many other advantages. But it is very difficult to follow a pure top−down design. Due to this fact most designs are mix of both the methods, implementing some key elements of both design styles.  Figure shows a Top−Down design approach.  C:\Users\User\Downloads\WhatsApp Image 2020-06-05 at 7.54.41 PM (3).jpeg C:\Users\User\Downloads\WhatsApp Image 2020-06-05 at 7.54.41 PM (4).jpeg  **C:\Users\User\Downloads\WhatsApp Image 2020-06-05 at 7.54.41 PM (5).jpeg**  **Types of Nets**  Each net type has functionality that is used to model different types of hardware (such as PMOS,NMOS, CMOS, etc)  **Register Data Types**  Registers store the last value assigned to them until another assignment statement  changes their value.  • Registers represent data storage constructs.  • You can create arrays of the regs called memories.  • register data types are used as variables in procedural blocks.  • A register data type is required if a signal is assigned a value within a procedural block  • Procedural blocks begin with keyword initial and always.  Some of the FPGA projects can be FPGA tutorials such as [What is FPGA Programming](https://www.fpga4student.com/2017/08/what-is-fpga-programming.html), [image processing on FPGA](https://www.fpga4student.com/2016/11/image-processing-on-fpga-verilog.html), [matrix multiplication](https://www.fpga4student.com/2016/11/matrix-multiplier-core-design.html) on FPGA Xilinx using Core Generator, [Verilog vs VHDL: Explain by Examples](https://www.fpga4student.com/2017/08/verilog-vs-vhdl-explain-by-example.html) and [how to load text files or images into FPGA](https://www.fpga4student.com/2016/11/two-ways-to-load-text-file-to-fpga-or.html). Many others FPGA projects provide students with full Verilog/ VHDL source code to practice and run on FPGA boards. Some of them can be used for another bigger FPGA projects.  **TASK FOR THE DAY**  **VERILOG MODULE TO COUNT NUMER OF 0’S IN A 16 BIT NUMBER IN COMPUTER**  **code**  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:** | 5 June 2020 | **Name:** | Anupama J S |
| **Course:** | Python | **USN:** | 4AL16EC005 |
| **Topic:** | 1. Application 9: Build a Data Collector Web App with PostGreSQL and Flask | **Semester & Section:** | 8th sem “A”section |
| **Github Repository:** | AnupamaJS |  |  |
| **AFTERNOON SESSION DETAILS** | | | |
| C:\Users\User\Downloads\WhatsApp Image 2020-06-05 at 8.25.30 PM.jpegC:\Users\User\Downloads\WhatsApp Image 2020-06-05 at 8.27.59 PM.jpegThis lesson’s goals Learn:   * Create a users table in the database. It will have user names, passwords, and permission flags. * Good passwords have lowercase letters, uppercase letters, digits, and special characters. They don’t correspond to a dictionary word.  The users table Let’s add a table called users to the DogToys database. It will have the data we need for authentication and permissions. Here are the fields:  users table  Figure 1. users table  user\_id is the primary key, an unsigned integer. auto\_increment means that MySQL will automatically supply values when new records are added to the users table.  user\_name is, er, the user name. It can be up to 20 characters long.  password is the user’s password. Like user\_name, it can be up to 20 characters long.  We’re storing the password in clear text. You wouldn’t do that in a secure application.  permission\_add is a one-character field that is either y or n. If it’s y, the user has permission to add products.  permission\_edit is a one-character field as well. If it’s y, the user can change data about existing products.  permission\_delete controls whether users are allowed to delete product records. Sample data Here’s some data. I broke the image into pieces so it would fit on this page.  Users data  Figure 2. Users data  You can see that Kieran and Louise are allowed to do anything to the data. But Renata can only edit data. She can’t add products or delete them.  Look at the passwords. Renata’s password is terrible. If a hacker learned Renata’s user name, one of the first things s/he would try is using the same thing for the password.  Loiuse’s password isn’t very good either. The characters are all the same.  CC’s password is bad as well. It’s a single word. It’s vulnerable to “dictionary” attacks, where a program tries to log in using English words as passwords. Eventually, it would hit “sheltie.” The word is in [dictionary.com](http://dictionary.reference.com/browse/sheltie).  Kieran’s password is the only good one. It uses four different character types:   * Lowercase letters * Uppercase letters * Digits * Special characters   It’s not vulnerable to dictionary attacks, and is hard to guess. Maintaining user data We need to be able to:   * Add new user records * Let users change their passwords * Delete records when people leave the company * Change permissions when people change jobs   In a real Web application, we’d write Web pages for these tasks. We’d assign someone as an administrator. We’d add a permissions field that would let an administrator access those the pages that change user data.  To keep things simple, we don’t do that in this chapter. It’s not core.  You can change the user data with phpMyAdmin. Exercise: DogJokes users table Add a users table to the DogJokes database you [created in the previous chapter](http://coredogs.com/lesson/starting-jokes-project.html). Use the same fields I used for the DogToys users table. Add some records.  Can't find the 'comment' module! [Was it selected](http://coredogs.com/admin/build/modules)?  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  ) | | | |