**DAY 13 ASSIGNMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **01-06-2020** | **Name:** | **Ashish Shanbhag** |
| **Course:** | **DIGITAL DESIGN USING HDL** | **USN:** | **4AL16EC008** |
| **Topic:** | 1. **Industry Applications of FPGA** 2. **FPGA Business Fundamentals** 3. **FPGA vs ASIC Design Flow** | **Semester & Section:** | **8th A** |
| **Github Repository:** | **Ashish Shanbhag** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS**      **Industry Applications of FPGA**  An FPGA is a (mostly) digital, (re-)configurable ASIC.  I say mostly because there are analog and mixed-signal aspects to modern FPGAs.  For example, some have A/D converters and PLLs.  I put re- in parenthesis because there are actually one-time-programmable FPGAs, where once you configure them, that’s it, never again.  However, most FPGAs you’ll come across are going to be re-configurable.  Many applications rely on the parallel execution of identical operations; the ability to configure the FPGA’s CLBs into hundreds or thousands of identical processing blocks has applications in image processing, artificial intelligence (AI), data center hardware accelerators, enterprise networking and automotive advanced driver assistance systems (ADAS).  Many of these application areas are changing very quickly as requirements evolve and new protocols and standards are adopted. FPGAs enable manufacturers to implement systems that can be updated when necessary.  A good example of FPGA use is high-speed search: Microsoft is using FPGAs in its data centers to run Bing search algorithms. The FPGA can change to support new algorithms as they are created. If needs change, the design can be repurposed to run simulation or modeling routines in an HPC application. This flexibility is difficult or impossible to achieve with an ASIC.  Other FPGA uses include aerospace and defense, medical electronics, digital television, consumer electronics, industrial motor control, scientific instruments, cybersecurity systems and wireless communications.  **Write a verilog code to implement NAND gate in all different styles.**  **Gate Level Modeling**  module nand\_gate(c,a,b);  input a,b;  output c;  nand (c,a,b);  endmodule  **Data Flow Modeling**  module nand\_data(c,a,b);  input a,b;  output c;  assign c =~(a&b);  endmodule  **Behavioral Modeling**  module nand\_beh(c,a,b);  input a,b;  output c;  reg c;  always@(a,b)  begin  if (a==1 & b==1)  c=0;  else  c=1;  end  endmodule  **Test Bench**  module nand\_test;  reg a,b;  wire c;  nand\_gate nand\_test(c,a,b);  initial  begin  #000 a=0;b=0;  #100 a=0;b=1;  #100 a=1;b=0;  #100 a=1;b=1;  end  initial  begin  $monitor($time,"a=%b,b=%b,c=%b",a,b,c);  end  endmodule |

**DAY 13 ASSIGNMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **01-06-2020** | **Name:** | **Ashish Shanbhag** |
| **Course:** | **PYTHON** | **USN:** | **4AL16EC008** |
| **Topic:** | 1. **Interactive Data Visualization with Bokeh** 2. **Webscraping with Python Beautiful Soup** | **Semester & Section:** | **8th A** |
| **Github Repository:** | **Ashish Shanbhag** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS**      **Interactive Data Visualization with Bokeh**  Bokeh prides itself on being a library for *interactive* data visualization. Unlike popular counterparts in the Python visualization space, like Matplotlib and Seaborn, Bokeh renders its graphics using HTML and JavaScript. This makes it a great candidate for building web-based dashboards and applications. However, it’s an equally powerful tool for exploring and understanding your data or creating beautiful custom charts for a project or report.  Building a visualization with Bokeh involves the following steps:   * Prepare the data * Determine where the visualization will be rendered * Set up the figure(s) * Connect to and draw your data * Organize the layout * Preview and save your beautiful data creation   from bokeh.plotting import figure  from bokeh.io import output\_file, show  import pandas    #prepare some data  df=pandas.read\_csv("http://pythonhow.com/data/bachelors.csv")  x=df["Year"]  y=df["Engineering"]    #prepare the output file  output\_file("Line\_from\_bachelors.html")    #create a figure object  f=figure()    #create line plot  f.line(x,y)    #write the plot in the figure object  show(f)  **Webscraping with Python Beautiful Soup**  Web Scraping is the process of downloading data from websites and extracting valuable information from that data. The need for Web Scraping is increasing, and so it’s the perfect time to get comfortable using it. The process of web scraping and cleaning the scraped data is logical and can therefore be implemented easily and become second nature after a few attempts. Getting Started Library wise we have a few different choices, including:   * Request * Beautiful Soup * Scrapy * Selenium   Scrapy is a complete web scraping framework which takes care of everything from getting the HTML, to processing the data. Selenium is a browser automation tool that can for example enable you to navigate between multiple pages. These two libraries have a steeper learning curve than Request which is used to get HTML data and BeautifulSoup which is used as a parser for the HTML. |