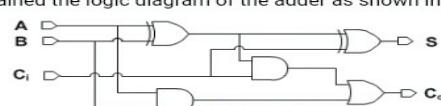


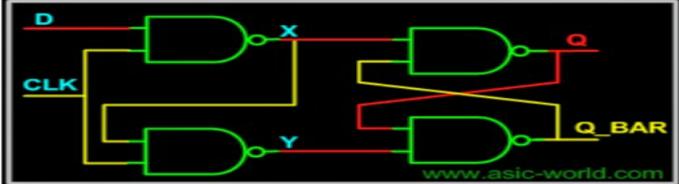
DAILY ASSESSMENT FORMAT

Date:	05-06-2020	Name:	K Muthu
Course:	DIGITAL DESIGN USING HDL	USN:	4al17ec038
Topic:	<ul style="list-style-type: none"> • Verilog Tutorials and practice programs • Building/ Demo projects using FPGA • Task 	Semester & Section:	6 & 'A'
Github Repository:	K.Muthu-courses		

FORENOON SESSION DETAILS	
<p>Image of session</p>  <p>The screenshot shows the homepage of fpga4student.com. The main title is "What is an FPGA?". Below it, there is a brief explanation of what an FPGA is, mentioning it's a Field Programmable Gate Array. It then lists two definitions: "Field-Programmable" and "Gate-Array". The page also features two images of FPGA boards: a smaller one on the left and a larger one on the right, both with red arrows pointing to specific components.</p> <p>What is an FPGA?</p> <p>What is FPGA? FPGA stands for Field Programmable Gate Array. Let's analyze the term: 1. Field-Programmable: An FPGA is manufactured to be easily reconfigured by developers, designers or customers. To program an FPGA as a specific configuration, Verilog HDL or VHDL (Hardware Description Language) is used as the standard language for FPGA programming. 2. Gate-Array: An FPGA consists of an array of programmable logic gates/ blocks such as AND, OR, XOR, NOT, memory elements, DSP components, etc., and reconfigurable interconnects which are to connect logic gates together for performing a specific function.</p> <p>What is an FPGA?</p> <p>Thus, FPGAs are nothing, but logic blocks and interconnects that can be programmable by Hardware Description Languages (Verilog HDL/ VHDL) to perform different complex functions. In fact, FPGAs can be used to implement almost any DSP algorithm. Some FPGAs also obtain embedded soft-core processors such as Xilinx's MicroBlaze, Altera's Nios II, etc. so that we can use C, C++, etc. to program the processor like what we do with a microcontroller. Besides, the soft processors can communicate with hardware accelerators to speed up complex DSP operations so that we can obtain a better flexible embedded system for niche applications.</p> <p>Let's take a very basic example on how to use an FPGA. Let's assume that you are designing a 1-bit full adder and you already obtained the logic diagram of the adder as shown in the figure below.</p>  <p>The logic diagram illustrates a 1-bit full adder. It takes three inputs: A, B, and C_i (carry-in). The sum (S) is produced by an OR gate that receives the outputs of two AND gates. The first AND gate has inputs A and B, while the second AND gate has inputs A and C_i. The carry-out (C_o) is produced by an OR gate that receives the outputs of two AND gates. The first AND gate has inputs B and C_i, while the second AND gate has inputs A and B. The website watermark "fpga4student.com" is visible at the bottom of the diagram.</p>	

Report -

Verilog Tutorials and practice programs :

 D-Flip flop from NAND Gate

Verilog Code

```
1 moduledff_from_nand();
2 wire Q,Q_BAR;
3 reg D,CLK;
4 
5 hand U1 (X,D,CLK);
6 hand U2 (Y,X,CLK);
7 hand U3 (Q,Q_BAR,X);
8 hand U4 (Q_BAR,Q,Y);
9 
10 // Testbench of above code
11 initial begin
12   $monitor( "CLK = %b D = %b Q = %b Q_BAR = %b" ,CLK, D, Q, Q_BAR);
13   CLK = 0;
14   D = 0;
15   #3 D = 1;
16   #3 D = 0;
17   #3 $finish;
18 end
19 
20 always #2 CLK = ~CLK;
21 
22 endmodule
```

Building/ Demo projects using FPGA :

Some of the FPGA projects can be FPGA tutorials such as [What is FPGA Programming, image proc on FPGA, matrix multiplication](#) on FPGA Xilinx using Core Generator, [Verilog vs VHDL: Examples](#) and [how to load text files or images into FPGA](#). Many others FPGA projects provide source code with full Verilog/ VHDL source code to practice and run on FPGA boards. Some of them can be used for other bigger FPGA projects.

- Following are the [FPGA projects](#) on [fpga4student.com](#):
1. [What is an FPGA? How does FPGA work?](#)
 2. [Basys 3 FPGA OV7670 Camera](#)
 3. [How to load text file or image into FPGA](#)
 4. [Image processing on FPGA using Verilog](#)
 5. [License Plate Recognition on FPGA](#)
 6. [Alarm Clock on FPGA using Verilog](#)
 7. [Digital Clock on FPGA using VHDL](#)
 8. [Simple Verilog code for debouncing buttons on FPGA](#)
 9. [Traffic Light Controller on FPGA](#)
 10. [Car Parking System on FPGA in Verilog](#)
 11. [VHDL code for comparator on FPGA](#)
 12. [Verilog code for Multiplier on FPGA](#)
 13. [N-bit Ring Counter in VHDL on FPGA](#)
 14. [Verilog implementation of Microcontroller on FPGA](#)
 15. [Verilog Carry Look Ahead Multiplier on FPGA](#)
 16. [VHDL Matrix Multiplication on FPGA Xilinx](#)
 17. [Fixed Point Matrix Multiplication on FPGA using Verilog](#)
 18. [Verilog Divider on FPGA](#)
 19. [VHDL code for Microcontroller on FPGA](#)
 20. [VHDL code for FIR Filter on FPGA](#)
 21. [Verilog code for Digital logic components on FPGA](#)
 22. [Delay Timer Implementation on FPGA using Verilog](#)
 23. [Single-Cycle MIPS processor on FPGA using Verilog](#)
 24. [FIFO Verilog Implementation on FPGA](#)
 25. [FIFO VHDL Implementation on FPGA](#)
 26. [Verilog D Flip Flop on FPGA](#)
 27. [Comparator Design on FPGA using Verilog](#)
 28. [D Flip Flop on FPGA using VHDL](#)
 29. [Full Adder Design on FPGA using Verilog](#)
 30. [Full Adder Design on FPGA using VHDL](#)

Task : Implement a verilog module to count number of 0's in a 16 bit number in compiler.

- Verilog code :

The screenshot shows the EDA playground interface. On the left, there's a sidebar with settings for languages (SystemVerilog/Verilog), tools (Icarus Verilog 0.9.6), and run options. The main area has two tabs: 'testbench.sv' and 'design.sv'. The 'testbench.sv' tab contains a Verilog testbench that generates four different 16-bit input values ('0', '100', '1afaf', and '18bd0') and monitors the number of zeros in each. The 'design.sv' tab contains a Verilog module 'count_0' that takes an 16-bit input and returns a 4-bit output representing the count of zeros. Below the code tabs is a log window showing the command 'iverilog -Wall design.sv testbench.sv && vcd' and its output, which lists the inputs and their zero counts.

```

// Code your SV/Verilog Testbench
// or browse Examples
module stimulus;
reg [15:0] in;
wire [4:0] out;
count_0 c0(out,in);
initial
begin
$dumpfile("dump1.vcd");
$dumpvars(1,stimulus);
in=16'h0000;
#100;
in=16'hAFAF;
#100;
in=16'h8BD0;
#100;
in=16'hFFFF;
#200;
end
initial $monitor($time," -
Input=%b No of
zeroes=%d",in,out);
endmodule

```

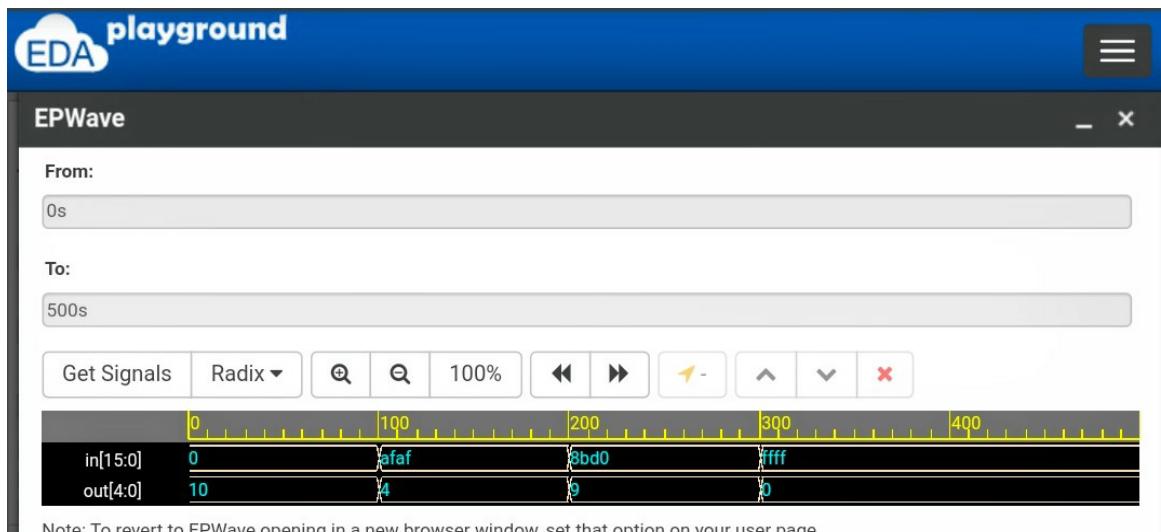
```

// Code your SV/Verilog Design
module count_0(out,in);
input [15:0] in;
output reg [4:0] out;
integer i;
always@(in)
begin
out=0;
for(i=0;i<16;i=i+1)
if (in[i]==1'b0)
out=out+1;
end
endmodule

```

[2020-06-05 06:00:14 EDT] iverilog '-Wall' design.sv testbench.sv && vcd
VCD info: dumpfile dump1.vcd opened for output.
0 - Input=0000000000000000 No of zeroes=16
100 - Input=101011110101111 No of zeroes= 4
200 - Input=1000101111010000 No of zeroes= 9
300 - Input=1111111111111111 No of zeroes= 0

- Output :



Note : The values displayed in EPWave output are in hexadecimal base.

Date: 05-06-2020

Name: K Muthu

Course: Python Bootcamp 2020 build 15
working applications and Games

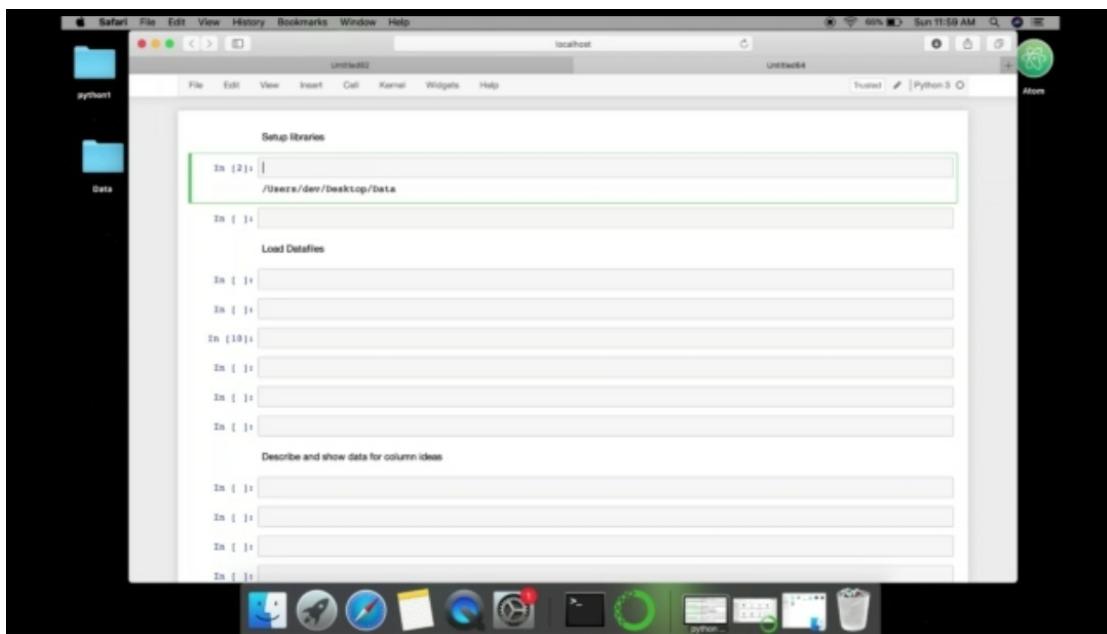
USN: 4al17ec038

Topic: Data Cleaning

**Semester 6 & 'A'
& Section:**

AFTERNOON SESSION DETAILS

Image of session



Lectures More

170



Loading data using pandas



170 Video - 15:50 mins

171

Pandas documentation

Article

Report –

Data Cleaning :

- Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.
- This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results.
- There are several methods for cleaning data depending on how it is stored along with the answers being sought.
- Data cleaning is a vital step to ensure that the answers you generate are accurate.
- Steps that are performed in data cleaning are,
 - ✓ Load the dataset
 - ✓ Identify the number of NULL values in the dataset
 - ✓ Modify the dataset by converting the NULL values into maximum occurred element of that particular column.
 - ✓ Repeat the process until all the NULL values are modified