

## DAILY ASSESSMENT FORMAT

Date:	04/06/2020	Name:	Davis S. Patel
Course:	DIGITAL DESIGN USING HDL	USN:	4AL16EC045
Topic:	Hardware modelling using Verilog FPGA and ASIC Interview questions	Semester & Section:	8 <sup>th</sup> - A
GitHub Repository:	Davis		

### FORENOON SESSION DETAILS

Image of session

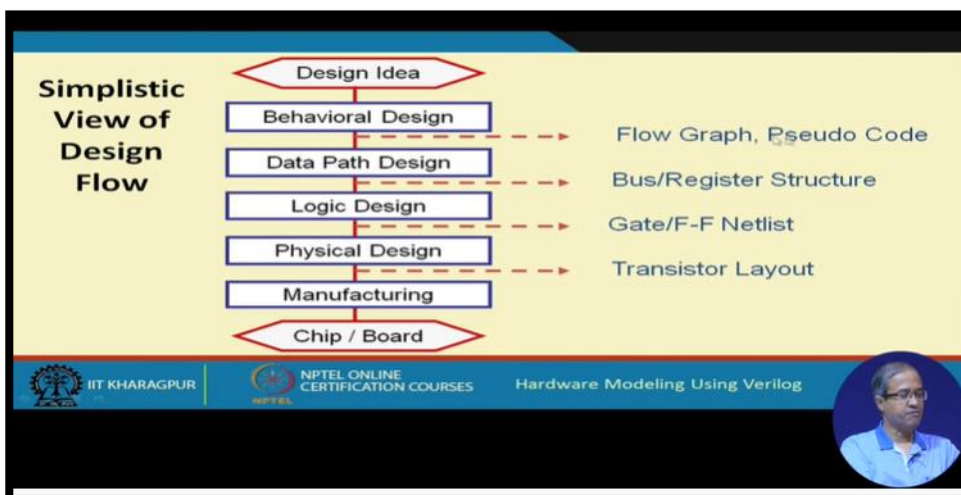
**Two Competing HDLs**

1. Verilog
2. VHDL

*Designs are created typically using HDLs, which get transformed from one level of abstraction to the next as the design flow progresses.*

There are other HDLs like SystemC, SystemVerilog, and many more ...


IIT KHARAGPUR
 NPTEL ONLINE CERTIFICATION COURSES
 Hardware Modeling Using Verilog



• Physical design and Manufacturing

- Generate the final layout that can be sent for fabrication.
- The layout contains a large number of regular geometric shapes corresponding to the different fabrication layers.
- Alternatively, the final target may be Field Programmable Gate Array (FPGA), where technology mapping from the gate level netlist is used.
  - Can be programmed in-field.
  - Much greater flexibility, but less speed.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog



## **REPORT –**

HDL is an abbreviation of Hardware Description Language. Any digital system can be represented in a REGISTER TRANSFER LEVEL (RTL) and HDLs are used to describe this RTL. Verilog is one such HDL and it is a general-purpose language –easy to learn and use. Its syntax is similar to C. The idea is to specify how the data flows between registers and how the design processes the data. To define RTL, hierarchical design concepts play a very significant role. Hierarchical design methodology facilitates the digital design flow with several levels of abstraction. Verilog HDL can utilize these levels of abstraction to produce a simplified and efficient representation of the RTL description of any digital design. For example, an HDL might describe the layout of the wires, resistors and transistors on an Integrated Circuit (IC) chip, i.e., the switch level or, it may describe the design at a more micro level in terms of logical gates and flip flops in a digital system, i.e., the gate level. Verilog supports all of these levels.

## **Hierarchy of design methodologies**

**Bottom-Up Design** The traditional method of electronic design is bottom-up (designing from transistors and moving to a higher level of gates and, finally, the system). But with the increase in design complexity traditional bottom-up designs have to give way to new structural, hierarchical design methods.

**Top-Down Design** For HDL representation it is convenient and efficient to adapt this design-style. A real top-down design allows early testing, fabrication technology independence, 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.

## **Hierarchical design concept and Verilog**

To follow the hierarchical design concepts briefly mentioned above one has to describe the design in terms of entities called MODULES.

**Modules** -A module is the basic building block in Verilog. It can be an element or a collection of low level design blocks. Typically, elements are grouped into modules to provide common functionality used in places of the design through its port interfaces, but hides the internal implementation

## **Abstraction Levels**

- Behavioral level
- Register-Transfer Level
- Gate Level
- Switch level

### **Behavioral or algorithmic Level**

This level describes a system by concurrent algorithms (Behavioral). Each algorithm itself is sequential meaning that it consists of a set of instructions that are executed one after the other. 'initial', 'always', 'functions' and 'tasks' blocks are some of the elements used to define the system at this level. The intricacies of the system are not elaborated at this stage and only the functional description of the individual blocks is prescribed. In this way the whole logic synthesis gets highly simplified and at the same time more efficient.

### **Register-Transfer Level Designs using the Register**

Transfer Level specify the characteristics of a circuit by operations and the transfer of data between the registers. An explicit clock is used. RTL design contains exact timing possibility, operations are scheduled to occur at certain times. Modern definition of a RTL code is "Any code that is synthesizable is called RTL code".

### **Gate Level**

Within the logic level the characteristics of a system are described by logical links and their timing properties. All signals are discrete signals. They can only have definite logical values ('0', '1', 'X', 'Z'). The usable operations are predefined logic primitives (AND, OR, NOT etc gates). It must be indicated here that using the gate level modeling may not be a good idea in logic design. Gate level code is generated by tools like synthesis tools in the form of netlists which are used for gate level simulation and for backend.

### **Switch Level**

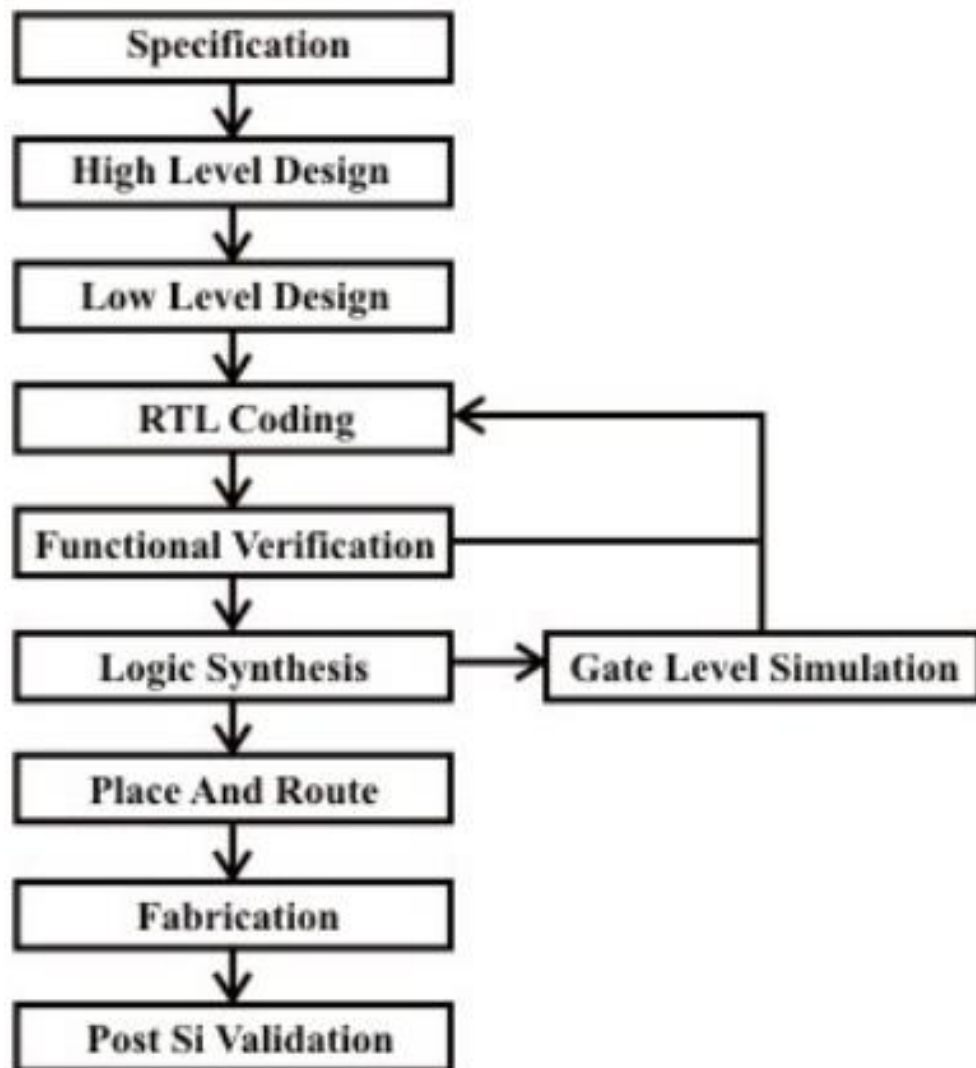
This is the lowest level of abstraction. A module can be implemented in terms of switches, storage nodes and interconnection between them. However, as has been mentioned earlier, one can mix and match all the levels of abstraction in a design. RTL is frequently used for Verilog description that is a combination of behavioral and dataflow while being acceptable for synthesis.

## **Instances**

A module provides a template from where one can create objects. When a module is invoked Verilog creates a unique object from the template, each having its own name, variables, parameters and I/O interfaces. These are known as instances.

## **The Design Flow**

This block diagram describes a typical design flow for the description of the digital design for both ASIC and FPGA realizations.



LEVEL OF FLOW	TOOLS USED
Specification	Word processor like Word, Kwriter, AbiWord, Open Office
High Level Design	Word processor like Word, Kwriter, AbiWord, for drawing waveform use tools like waveformr or testbencher or Word, Open Office.
Micro Design/Low level design	Word processor like Word, Kwriter, AbiWord, for drawing waveform use tools like waveformr or testbencher or Word. For FSM StateCAD or some similar tool, Open Office
RTL Coding	Vim, Emacs, conTEXT, HDL TurboWriter
Simulation	Modelsim, VCS, Verilog-XL, Veriwell, Finsim, iVerilog, VeriDOS
Synthesis	Design Compiler, FPGA Compiler, Synplify, Leonardo Spectrum. You can download this from FPGA vendors like Altera and Xilinx for free
Place & Route	For FPGA use FPGA' vendors P&R tool. ASIC tools require expensive P&R tools like Apollo. Students can use LASI, Magic
Post Si Validation	For ASIC and FPGA, the chip needs to be tested in real environment. Board design, device drivers needs to be in place

#### **Task 4 - Implement a simple T Flip-flop and test the module using a compiler.**

##### **Design**

```

module tff ( input clk,
            input rstn,
            input t,
            output reg q);

always @ (posedge clk) begin

if (!rstn)

    q <= 0;

else

    if (t)

```

```
    q <= ~q;

else

    q <= q;

end

endmodule
```

### **Test bench**

```
module tb;

reg clk;

reg rstn;

reg t;

tff u0 ( .clk(clk),

        .rstn(rstn),

        .t(t),

        .q(q));

always #5 clk = ~clk;

initial begin

    {rstn, clk, t} <= 0;

    $monitor ("T=%0t rstn=%0b t=%0d q=%0d", $time, rstn, t, q);

    repeat(2) @(posedge clk);

    rstn <= 1;

    for (integer i = 0; i < 20; i = i+1) begin

        reg [4:0] dly = $random;

        #(dly) t <= $random;

    end

end
```

#20 \$finish;

end

endmodule

## DAILY ASSESSMENT FORMAT

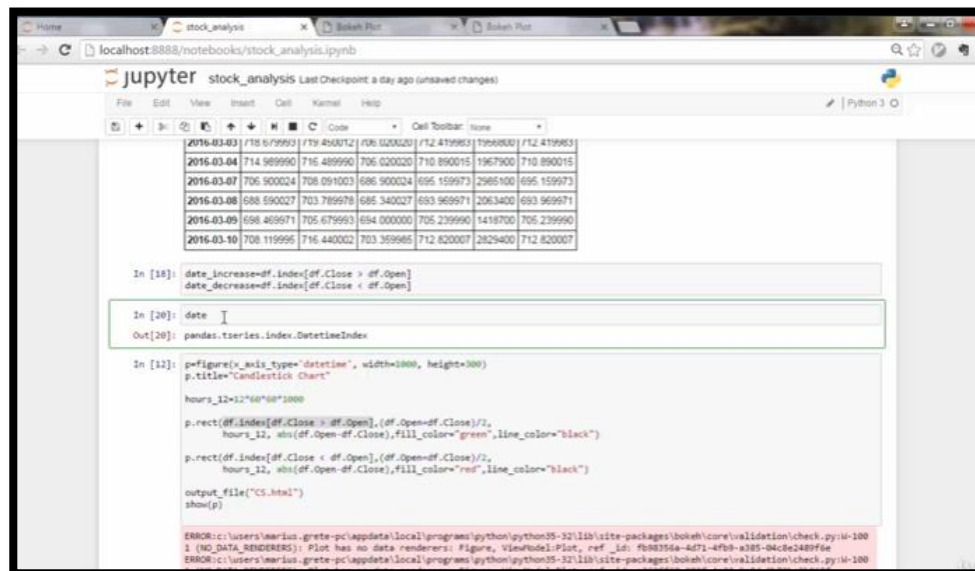
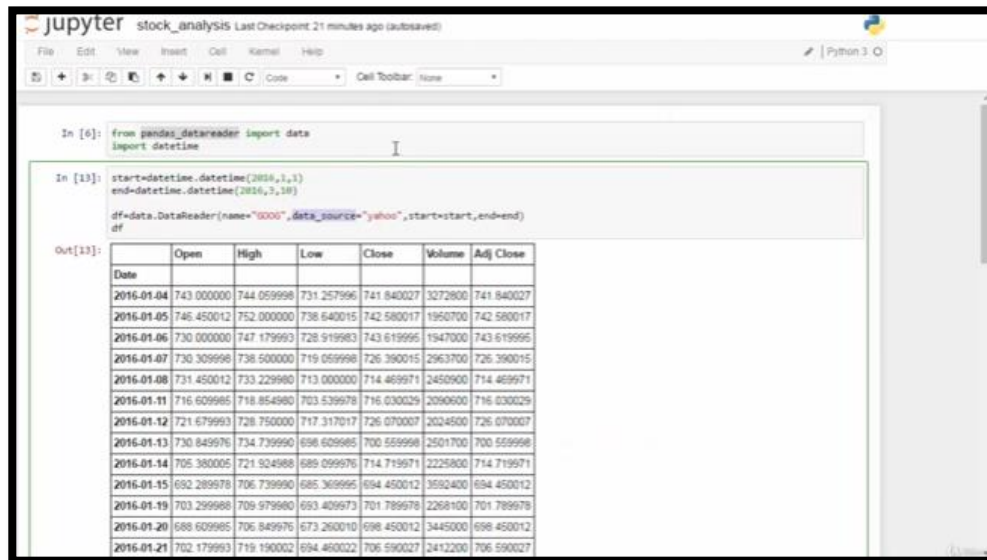
Date:	04/06/2020	Name:	Davis S. Patel
Course:	Python Course	USN:	4AL16EC045
Topic:	Application 8: Build a Web-based Financial Graph	Semester & Section:	8 <sup>th</sup> - A
GitHub Repository:	Davis		

### AFTERNOON SESSION DETAILS

Image of Session





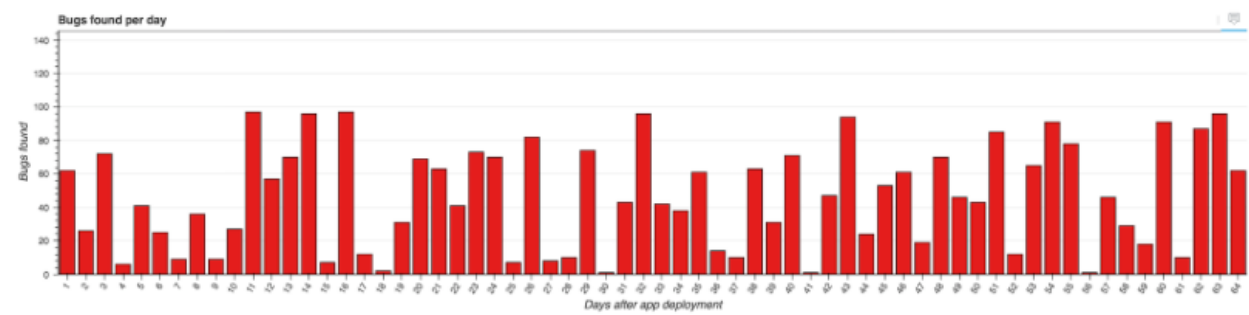


## **REPORT –**

Bokeh is a powerful open source Python library that allows developers to generate JavaScript data visualizations for their web applications without writing any JavaScript. While learning a JavaScript-based data visualization library like d3.js can be useful, it's often far easier to knock out a few lines of Python code to get the job done.

With Bokeh, we can create incredibly detailed interactive visualizations, or just traditional ones like the following bar chart.

**Bugs found over the past 64 days**



Bokeh provides a variety of ways to embed plots and data into HTML documents. First, a reminder of the distinction between standalone documents and apps:

### **Standalone Documents**

These are Bokeh documents that are not backed by a Bokeh server. They may have many tools and interactions (e.g. from CustomJS callbacks) but are self-contained HTML, JavaScript, and CSS. They can be embedded into other HTML pages as one large document, or as a set of sub-components templated individually.

### **Bokeh Applications**

These are Bokeh documents that are backed by a Bokeh Server. In addition to all the features of standalone documents, it is also possible to connect events and tools to real Python callbacks that execute in the Bokeh server.

## HTML files

Bokeh can generate complete HTML pages for Bokeh documents using the `file_html()` function. This function can emit HTML from its own generic template, or a template you provide. These files contain the data for the plot inline and are completely transportable, while still providing interactive tools (pan, zoom, etc.) for your plot. Here is an example:

```
from bokeh.plotting import figure

from bokeh.resources import CDN

from bokeh.embed import file_html


plot = figure()

plot.circle([1,2], [3,4])


html = file_html(plot, CDN, "my plot")
```

The returned HTML text can be saved to a file using standard python file operations. You can also provide your own template and pass in custom, or additional, template variables. See the `file_html()` documentation for more details. This is a fairly low-level, explicit way to generate an HTML file, which may be useful for use from a web application, e.g. a Flask app. When using the `bokeh.plotting` interface in a script or Jupyter notebook, users will typically call the function `output_file()` in conjunction with `show()` or `save()` instead.

Plots and data in the form of standalone documents as well as Bokeh applications can be embedded in HTML documents.

Standalone document is a Bokeh plot or document not backed by Bokeh server. The interactions in such a plot is purely in the form of custom JS and not Pure Python callbacks.

Bokeh plots and documents backed by Bokeh server can also be embedded. Such documents contain Python callbacks that run on the server.

In case of standalone documents, a raw HTML code representing a Bokeh plot is obtained by `file_html()` function.

```
from bokeh.plotting import figure
from bokeh.resources import CDN
from bokeh.embed import file_html
fig = figure()
fig.line([1,2,3,4,5], [3,4,5,2,3])
string = file_html(plot, CDN, "my plot")
```

Return value of `file_html()` function may be saved as HTML file or may be used to render through URL routes in Flask app.

In case of standalone document, its JSON representation can be obtained by `json_item()` function.

```
from bokeh.plotting import figure
from bokeh.embed import file_html
import json
fig = figure()
fig.line([1,2,3,4,5], [3,4,5,2,3])
item_text = json.dumps(json_item(fig, "myplot"))
```

This output can be used by the `Bokeh.embed.embed_item` function on a webpage –

```
item = JSON.parse(item_text);  
Bokeh.embed.embed_item(item);
```

Bokeh applications on Bokeh Server may also be embedded so that a new session and Document is created on every page load so that a specific, existing session is loaded. This can be accomplished with the `server_document()` function. It accepts the URL to a Bokeh server application, and returns a script that will embed new sessions from that server any time the script is executed.

The **`server_document()` function** accepts URL parameter. If it is set to 'default', the default URL `http://localhost:5006/` will be used.

```
from bokeh.embed import server_document  
script = server_document("http://localhost:5006/sliders")
```

The `server_document()` function returns a script tag as follows –

```
<script  
  src="http://localhost:5006/sliders/autoload.js?bokeh-autoload-element=1000&bokeh-  
app-path=/sliders&bokeh-absolute-url=https://localhost:5006/sliders"  
  id="1000">  
</script>
```