DAILY ASSESSMENT

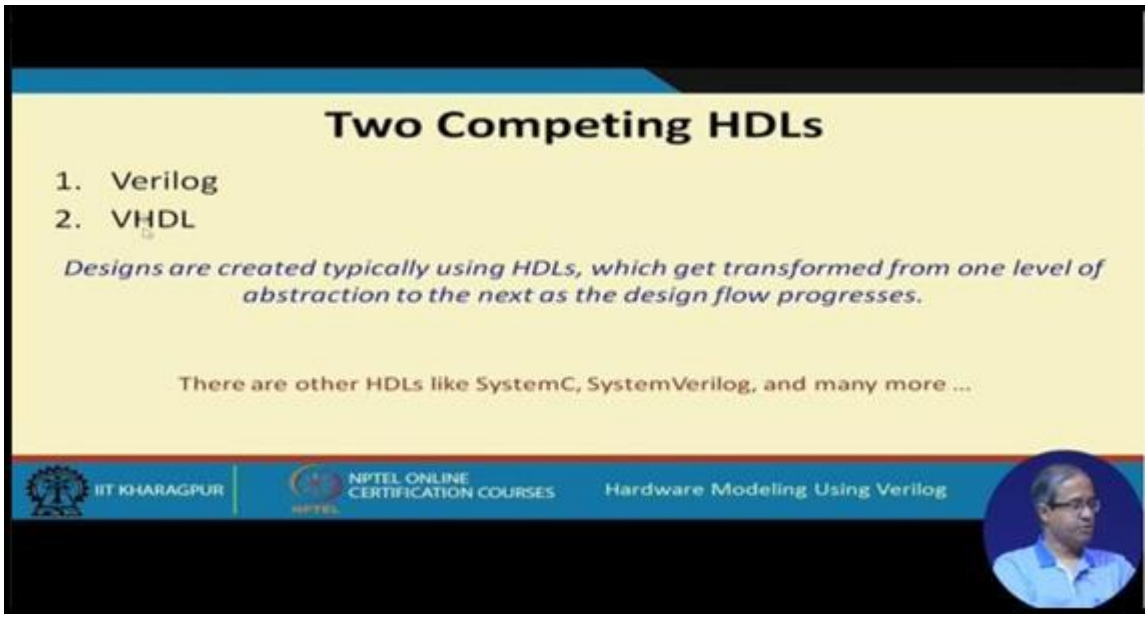
Date:	**04/06/2020**	**Name:**	**CHANDANA.R**
Course:	HDL	**USN:**	**4AL16EC017**
Topic:	Hardware modelling Using Verilog FPGA and ASIC Interview questions	**Semester & Section:**	**8(A)**
Github Repository:	**chandana-shaiva**		

FORENOON SESSION DETAILS

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.

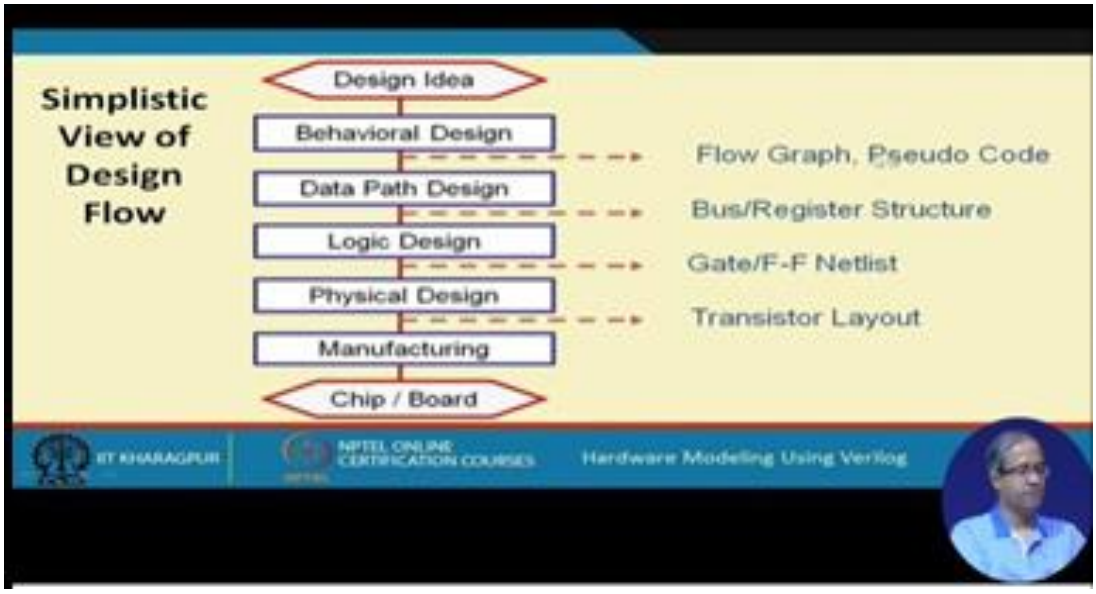
To follow the hierarchical design concepts briefly mentioned above one has to describe the design in terms of entities called MODULES.

- **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

➢ The Design Flow:



➢ 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 specifies 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 net lists 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.

➢ **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.

Task 4:

Implement a simple T Flip-flop and test the module using a compiler:

```
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
end module



Test bench:
module tb;
reg clk;
reg rstn;
reg t;
tffu0(.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= ++)
begin
reg[4:0]
delay= $random;
#(delay)t<= $random;
end
#20 $finish;
end
end module
```

DAILY ASSIGEMENTS DETAILS:

Date:	**04/06/2020**	**Name:**	**CHANDANA.R**
Course:	Python	**USN:**	**4AL15EC017**
Topic:	Application 8: Build a Web-based Financial Graph	**Semester & Section:**	**8(A)**
Github Repository:	**chandana-shaiva**		
AFTERNOON SESSION			

Session image:

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.
- 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:
- These are Bokeh documents that are not backed by a Bokeh server. They may have many tools and interactions (e.g. from Customs 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 template 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")
```

- 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>
```