

## DAILY ASSESSMENT FORMAT

<b>Date:</b>	<b>4-06-2020</b>	<b>Name:</b>	<b>BINDUSHRI</b>
<b>Course:</b>	<b>Digital design using HDL</b>	<b>USN:</b>	<b>4AL17EC011</b>
<b>Topic:</b>	<b>1.hardware modeling using Verilog 2.interview questions 3.TASK</b>		<b>6<sup>th</sup> A</b>
<b>Github Repository:</b>	<b>Bindushri</b>		

## FORENOON SESSION DETAILS

[Verilog FAQ](#)  
[Synthesis FAQ](#)  
[Digital FAQ](#)  
[Timing FAQ](#)  
[ASIC FAQ](#)  
[Cmos FAQ](#)  
[Misc FAQ](#)  
[Home](#)

### 2) Difference between blocking and non-blocking?(Verilog interview questions that is most commonly asked)

The Verilog language has two forms of the procedural assignment statement: blocking and non-blocking. The two are distinguished by the = and <= assignment operators. The blocking assignment statement (= operator) acts much like in traditional programming languages. The whole statement is done before control passes on to the next statement. The non-blocking (<= operator) evaluates all the right-hand sides for the current time unit and assigns the left-hand sides at the end of the time unit. For example, the following Verilog program

```
// testing blocking and non-blocking assignment

module blocking;
reg [0:7] A, B;
initial begin: init1
A = 3;
#1 A = A + 1; // blocking procedural assignment
B = A + 1;
$display("Blocking: A= %b B= %b", A, B ); A = 3;
#1 A = A + 1; // non-blocking procedural assignment
B = A + 1;
#1 $display("Non-blocking: A= %b B= %b", A, B );
end
endmodule
```

produces the following output:  
Blocking: A= 00000100 B= 00000101  
Non-blocking: A= 00000100 B= 00000100

The effect is for all the non-blocking assignments to use the old values of the variables at the beginning of the current time unit and to assign the registers new values at the end of the current time unit. This reflects how register transfers occur in some hardware systems.  
blocking procedural assignment is used for combinational logic and non-blocking procedural assignment for sequential

[Click to view more](#)

☆ B ⋮

and for (System)Verilog. ? 🧪 ⚡ Playgrounds Profile

design.sv +

h

```
1 module tff (    input clk,
2                input rstn,
3                input t,
4                output reg q);
5
6     always @ (posedge clk) begin
7         if (!rstn)
8             q <= 0;
9         else
10            if (t)
11                q <= ~q;
12            else
13                q <= q;
14        end
15    endmodule
16
```

SV/Verilog Design

log | help | map

EDA edit code - EDA Playground

edaplayground.com/x/5cPU

playground ⚙️ Save Copy Cadence Xcelium 19.0 is here! BTW. Mentor Precision examples: for VHDL and for (System)Verilog. ? 🧪 ⚡ Playgrounds

you by DOULOS

Images & Libraries

ch + Design

Verilog/Verilog

VM

Variables

1

TL-Verilog

Easier UVM

VUnit

& Simulators

ilog 0.9.7

& Run Options

ons

PWave after run

testbench.sv +

SV/Verilog Testbench

```
1 reg clk;
2 reg rstn;
3 reg t;
4
5 tff u0 ( .clk(clk),
6         .rstn(rstn),
7         .t(t),
8         .q(q));
9
10 always #5 clk = ~clk;
11
12 initial begin
13     {rstn, clk, t} <= 0;
14
15     $monitor ("T=%0t rstn=%0b t=%0d q=%0d", $time, rstn, t, q);
16     repeat(2) @(posedge clk);
17     rstn <= 1;
18
19     for (integer i = 0; i < 20; i = i+1) begin
20         reg [4:0] dly = $random;
21         #(dly) t <= $random;
22     end
23     #20 $finish;
24 end
25 endmodule
26
27
```

Log Share

design.sv +

SV

```
1 module tff (    input clk,
2                input rstn,
3                input t,
4                output reg q);
5
6     always @ (posedge clk) begin
7         if (!rstn)
8             q <= 0;
9         else
10            if (t)
11                q <= ~q;
12            else
13                q <= q;
14        end
15    endmodule
16
```

## Day-4 Hardware modeling using verilog.

→ Design complexity increasing rapidly

- Increased size & complexity
- Fabrication technology improving
- CAD tools are essential

### Moores Law

- Exponential growth
- Design complexity increase rapidly

### Standardized design procedure

- Starting from the design idea down to the actual implementation

### Logic Design

- Generate a network of gates / flip-flops or standard cells

### Interview question

- 1) Verilog code to swap contents of two registers with & without a temporary register?

→ with temp reg:

always @ (posedge clock)

begin

temp = b;

b = a;

a = temp;

end

without temp reg;

always @ (posedge clock)

begin

a <= b;

b <= a;

end

② Difference b/w blocking & non blocking.

→ the program has two forms of procedural assignment statement  
blocking & non-blocking. ' $=$ ' & ' $<=$ '

Blocking: whole statement is done before control pass next statement

Non-block: evaluates all the right-hand side for the current time unit and assigns the left-hand side at end of the time unit

③ Tell about verilog file I/O?

OPEN A FILE

integer ffile

ffile = \$fopen("filename");

ffile = \$fopenw("filename");

ffile = \$fopenw("filename");

the function \$fopen opens an existing file for reading

\$fopenw opens a new file for writing.

\$fopenw opens a new file for writing where any data would be needed to be appended at end of time.

CLOSE FILE

integer ffile, r;

r = \$fclose(ffile)

r = \$fclosew(ffile);

Task-4 Implement a simple TFF ~~latch~~

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

always @ (posedge clk) begin

if (!rstn)

q <= 0;

else if (t)

q <= ~q;

end  
endmodule



testbench

module tb;

reg clk;

reg rstn;

reg t;

ttl u0( .clk(clk),  
      .rstn(rstn),  
      .t(t),  
      .q(q));

always #5 clk = ~clk;

initial begin

  rstn, clk, t <= 0;

\$monitor (rstn=%0d t=%0d q=%0d",

          \$tmo, 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

#20 \$finish;

end

endmodule

Date:4june2020

Course: python  
Topic:sec 32

Name:Bindushri  
USN:4AL17EC011  
Sem&Sec:6<sup>th</sup> A

## AFTERNOON SESSION DETAILS

The screenshot displays the Udemy interface for the course "The Python Mega Course: Build 10 Real World Applications". The video player is active, showing a code editor with the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<title>Data Collector App</title>
<head>
<link href="static/main.css">
</head>
</html>
```

The course content list on the right side of the page includes the following lectures:

- 260. Frontend: HTML Part (15min)
- 261. Frontend: CSS Part (10min)
- 262. Backend: Getting User Input (18min)
- 263. Backend: The PostgreSQL Database Model (18min)
- 264. Backend: Storing User Data to the Database (19min)
- 265. Backend: Emailing Database Values Back to the User (11min)
- 266. Backend: Sending Statistics to Users (16min)
- 267. Deploying the Web Application to a Live Server (30min)
- 268. Bonus Lecture: Implementing Download

The "About this course" section is visible at the bottom of the page.

Python 4-06-2020

Data collector webapp

Frontend HTML part

templates. ↳ two folders.  
static.

- \* Inside template folder create one file index.html, success.html
- \* Inside static folder create one file main.css

index.html.

<!DOCTYPE html>

<html lang="en">

<title>Data Collector App</title>

<head>

<link href="/static/main.css" rel="stylesheet">

</head>

<body>

<div class="container">

<h1>collecting height</h1>

<h2>please fill the entries to get population  
statistics on height</h2>

<form action="/success" method="post">

<input title="You email be safe with us"

placeholder="Enter your email address"

type="email" name="email" required>

<br>

<input title="your data will be safe with us"

placeholder="Enter your height in cm" type="number"

min="50", max="300" name="height"

required><br>

<button type="submit">submit</button>



```
</form>
</div>
</body>
</html>
```

success.html

```
<!DOCTYPE html>
<html lang="en">
  <title> Data collector App </title>
  <head>
    <link href=".../static/main.css" rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <p> Thank you for Submission <br>
        You will email with survey results shortly
      </p>
    </div>
  </body>
</html>
```

main.css

```
html, body { height: 100%;
              margin: 0; }
```

container {

margin: 0 auto;

width: 100%;

height: 100%;

background-color: #060606;

color: #E6FFFF;

overflow: hidden;

text-align: center;



- container h1
  - font-family: Arial, sans-serif;
  - font-size: 30px;
  - color: #000000;
  - margin-top: 80px;

- container form
  - margin: 20px;

- container input
  - width: 350px;
  - height: 15px;
  - font-size: 15px;
  - margin: 0px;
  - padding: 0px;
  - transition: all 0.2s ease-in-out;

- container button
  - width: 70px;
  - height: 30px;
  - background-color: #000000;
  - margin: 4px;

- container p
  - margin: 100px;

Backend : getting user input



cmd → Python -m ~~venv~~ venv virtual.  
↓  
-this creates empty folder.

future note folder  
create one more  
app.py.

cmd → [virtual] scripts pip install flask.

→ app.py

```
1. from flask import Flask, render_template, request
2. app = Flask(__name__)
3.
4. @app.route("/")
5. def index():
6.     return render_template("index.html")
7. @app.route("/success", methods = ['POST'])
8. def success():
9.     return render_template("success.html")
10.     if request.method == 'POST':
11.         email = request.form["email name"]
12.         height = request.form["height name"]
13.         print(request.form["height"])
14.         return render_template("success.html")
15.
16. if __name__ == "__main__":
17.     app.debug = True
18.     app.run()
```

Suman

