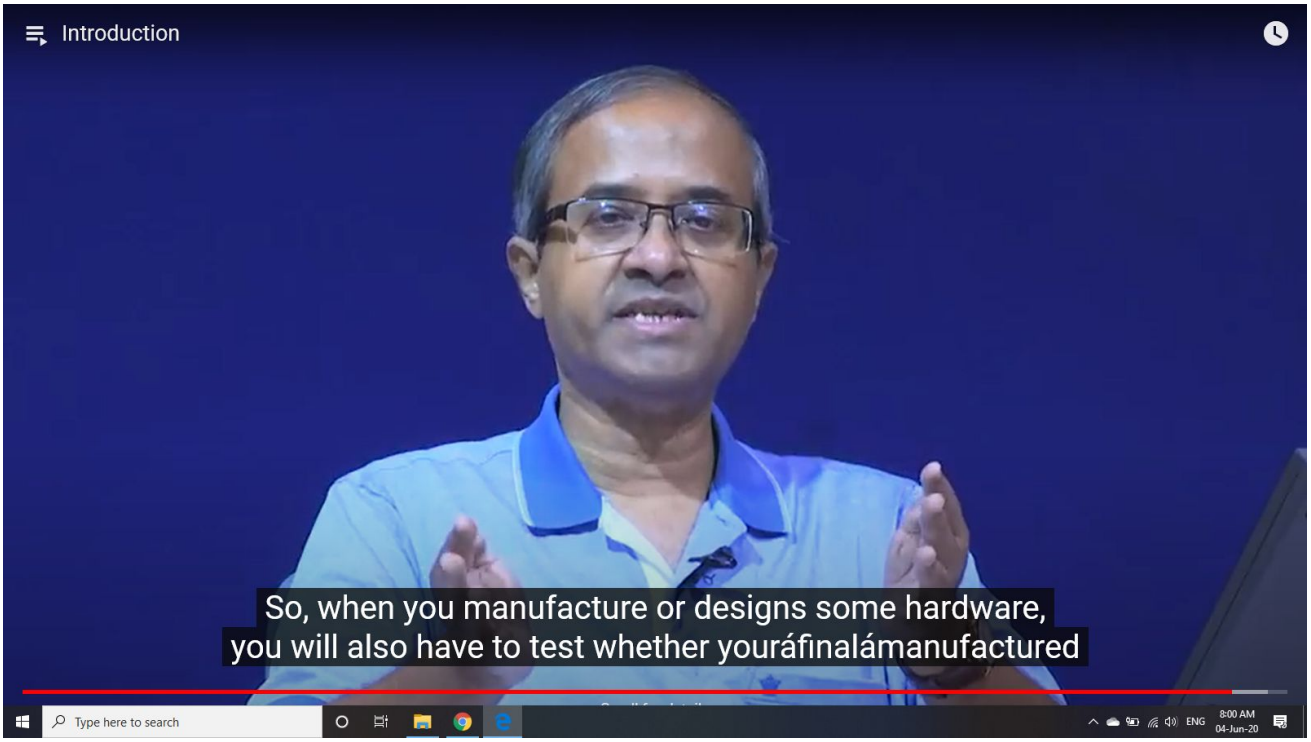
# REPORT JUNE 04

| | | | |
|---|---|---|---|
| **Date:** | **04 JUNE 2020** | **Name:** | **Rakshith B** |
| **Course:** | **Digital Design Using HDL** | **USN:** | **4AL16EC409** |
| **Topic:** | **Hardware Modeling using Verilog,Implement T Flip-Flop** | **Semester & Section:** | **6th SEM B** |
| **Github Repository:** | **Rakshith-B** | | |

**FORENOON SESSION DETAILS**

**Image of session**




**Report –Hardware Modeling Using Verilog**

**Objective of Hardware Modeling Using Verilog**

- **Learn about the Verilog hardware description language.**
- **Understand the difference between behavioral and structural design styles.**
- **Learn to write test benches and analyze simulation results.**
- **Learn to model combinational and sequential circuits,**
- **Distinguish between good and bad coding practices.**
- **Case studies with some complex designs.**

**VLSI Design Process**

- **Design complexity increasing rapidly**
  - ❑ **Increased size and complexity**
  - ❑ **Fabrication technology improving**
  - ❑ **CAD tools are essential**
  - ❑ **Conflicting requirements like area, speed, and energy consumption**

- **The present trend**
  - ❑ **Standardize the design flow**
  - ❑ **Emphasis on low-power design, and increased performance**

**Moore's Law**

- **Exponential growth**
- **Design complexity increases rapidly**
- **Automated tools are essential**
- **Must follow well defined design flow**

**Standardized design procedure**

- **Starting from the design idea down to the actual implementation.**

**Encompasses many steps:**

- **Specification Synthesis**
- **Simulation**
- **Layout Testability analysis**
- **and many more**

**Need to use Computer Aided Design (CAD) tools.**

- **Hardware Description Language (HDL)**
- **Based on HDL provide formats for representing the outputs of various design steps**
- **A CAD tool transforms its HDL input into a HDL output that contains more detailed information about the hardware.**
  - ❑ **Behavioral level to register transfer level**
  - ❑ **Register transfer level to gate level**
  - ❑ **Gate level to transistor level**
  - ❑ **Transistor to the layout level**

**Two Competing HDL's**

- **Verilog**
- **VHDL**

**Behavioral design**

- **Specify the functionality of the design in terms of its behavior.**
- **Various ways of specifying:**
  - ❑ **Boolean expression or truth table.**
  - ❑ **Finite-state machine behavior (e.g. state transition diagram or table).**
  - ❑ **In the form of a high-level algorithm.**

- **Needs to be synthesized into more detailed specifications for hardware realization,**

**Data path design**

- **Generate a netlist of register transfer level components, like registers, adders, multipliers, multiplexers, decoders, etc.**
- **A netlist is a directed graph, where the vertices indicate components, and the edges indicate interconnections.**

- **A netlist specification is also referred to as structural design.**
  - **Netlist may be specified at various levels, where the components may be functional modules, gates or transistors.**
  - **Systematically transformed from one level to the next**

**Logic design**

- **Generate a netlist of gates/flip-flops or standard cells.**
- **A standard cell is a pre-designed circuit module (like gates, flip-flops, multiplexer, etc.) at the layout level.**
- **Various logic optimization techniques are used to obtain a cost effective design.**

**There may be conflicting requirements during optimization:**

- **Minimize the number of gates.**
- **Minimize number of gate levels (i.e. delay).**
- **Minimize signal transition activities (i.e. dynamic power).**

**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.**

**Other Steps in the Design Flow**

- **Simulation for verification**
  - **At various levels: logic level, switch level, circuit level**
- **Formal verification**
  - **Used to verify the designs through formal techniques**
- **Testability analysis and Test pattern generation**
  - **Required for testing the manufactured devices**

**Verilog Code for T Flip-Flop**

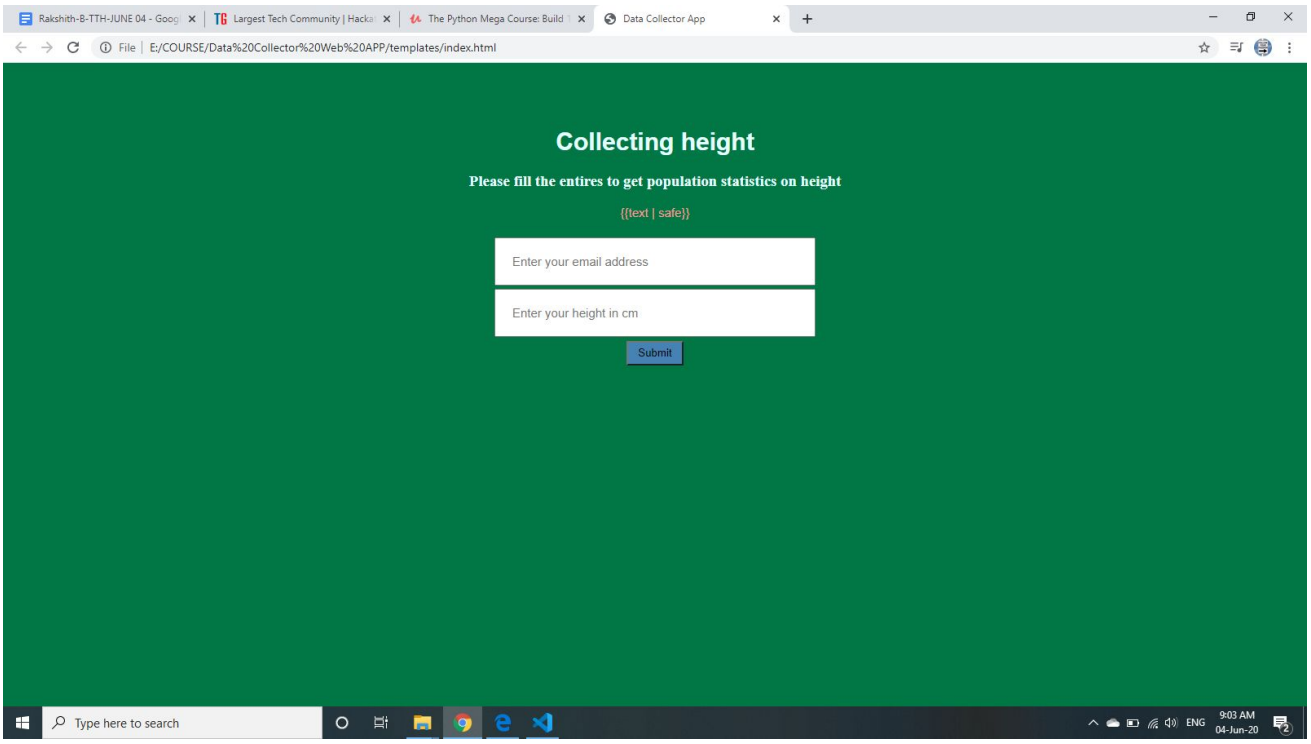
```
module tff (   input clk,
              input rstn,
              input t,
            output reg q);

  always @ (posedge clk) begin
    if (!rstn)
      q <= 0;
    else
      if (t)
          q <= ~q;
      else
          q <= q;
  end
endmodule
```

**Date:** 04 JUNE 2020 **Name:RAKSHITH B**
**Course:** Python On Udemy **USN:4AL16EC409**
**Topic:** Data Collector Web App **Semester & Section:6 B**

## AFTERNOON SESSION DETAILS

**Image of session:Output**



app.py

```
from flask import Flask, render_template, request
from flask.ext.sqlalchemy import SQLAlchemy
from send_email import send_email
from sqlalchemy.sql import func

app=Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI']='postgresql://postgres:postgr
es123@localhost/height_collector'
db=SQLAlchemy(app)

class Data(db.Model):
    __tablename__="data"
    id=db.Column(db.Integer, primary_key=True)
    email_=db.Column(db.String(120), unique=True)
    height_=db.Column(db.Integer)

    def __init__(self, email_, height_):
        self.email_=email_
        self.height_=height_

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/success", methods=['POST'])
def success():
    if request.method=='POST':
        email=request.form["email_name"]
        height=request.form["height_name"]
        print(email, height)
        if db.session.query(Data).filter(Data.email_ ==
email).count()== 0:
            data=Data(email,height)
            db.session.add(data)
            db.session.commit()

average_height=db.session.query(func.avg(Data.height_)).scalar()
            average_height=round(average_height, 1)
            count = db.session.query(Data.height_).count()
            send_email(email, height, average_height, count)
            print(average_height)
            return render_template("success.html")
    return render_template('index.html', text="Seems like we got
something from that email once!")

if __name__ == '__main__':
```

```
app.debug=True
app.run(port=5005)
```

**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>
        <h3>Please fill the entires to get population statistics on
height</h3>
        <div class="email"> {{text | safe}} </div>
        <form action="{{url_for('success')}}" method="POST">
          <input title="Your email will be safe with us" placeholder="Enter
your email address" type="email" name="email_name" 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_name"
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 class="success-message"> Thank you for your submission! <br>
          You will receive an email with the survey results shortly.
        </p>
      </div>
```

```
    </body>
</html>
```