Date:	05-06-2020	Name:	Rajeshwari Gadagi
Course:	HDL	USN:	4AL17EC076
Topic:	Verilog tutorial,demo projects,task	Semester and section:	6th sem and B sec

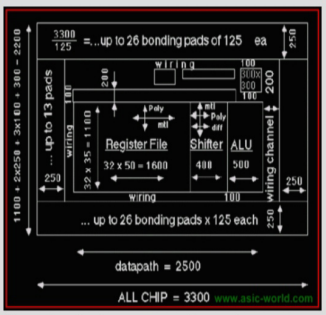


Figure : Sample micro-processor placement

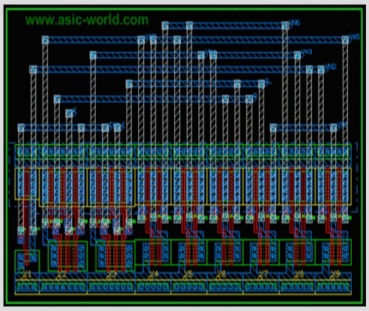


Figure : J-K Flip-Flop

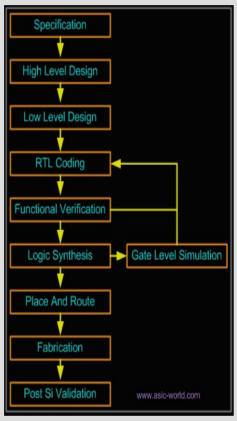
Post Silicon Validation

Once the chip (silicon) is back from fab, it needs to put in real environment and tested before it can be released into Market. Since the speed of simulation with RTL is very slow (number clocks per second), there is always possibility to find a bug in Post silicon validation.

Top-Down Design

The desired design-style of all designers is the top-down design. A real top-down design allows early testing, easy change of different technologies, 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.

Figure shows a Top-Down design approach.

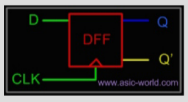


Abstraction Levels of Verilog

Verilog supports a design at many different levels of abstraction. Three of them are very important:

Introduction

Verilog is a **HARDWARE DESCRIPTION LANGUAGE (HDL)**. A hardware description Language is a language used to describe a digital system, for example, a network switch, a microprocessor or a memory or a simple flip-flop. This just means that, by using a HDL one can describe any hardware (digital) at any level.




```
// D flip-flop Code
module d_ff ( d, clk, q, q_bar);
input d ,clk;
output q, q_bar;
wire d ,clk;
reg q, q_bar;

always @ (posedge clk)
begin
   q <= d;
   q_bar <= !d;
end

endmodule
```

One can describe a simple Flip flop as that in above figure as well as one can describe a complicated designs having 1 million gates. Verilog is one of the HDL languages available in the industry for designing the Hardware. Verilog allows us to design a Digital design at Behavior Level, Register Transfer Level (RTL), Gate level and at switch level. Verilog allows hardware designers to express their designs with behavioral constructs, deterring the details of implementation to a later stage of design in the final design.

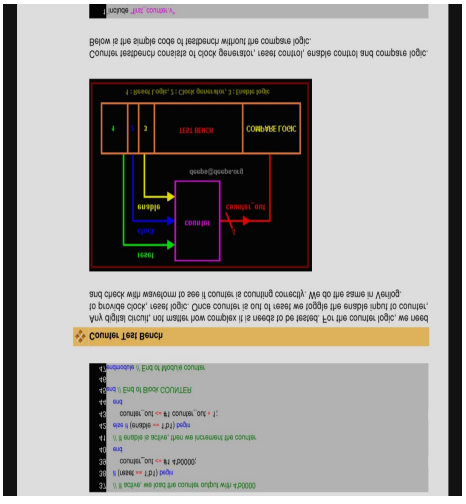
Many engineers who want to learn Verilog, most often ask this question, how much time it will take to learn Verilog?, Well my answer to them is **"It may not take more then one week, if you happen to know at least one programming language"**.

Design Styles

Verilog like any other hardware description language, permits the designers to design a design in either Bottom-up or Top-down methodology.

Bottom-Up Design

The traditional method of electronic design is bottom-up. Each design is performed at the gate-level using the standard gates (Refer to the Digital Section for more details) With increasing



Verilog Tutorial

- Verilog is a Hardware description language (HDL), which is used to describe a digital system.
- Design Methodologies → Top-down and Bottom up
- Abstraction levels of Verilog :-
  Behavioral level, Register-Transfer level, Gate level
- Various Stages of ASIC / FPGA -
  Specification, High level design, Micro Design / low level design, RTL Coding, Simulation, Synthesis, Place & Route, Post SI Validation
- Code -

```
module hello_world;
Initial begin
    $display ("Hello world");
    #10 $finish;
end
endmodule
```

- Verilog language has two primary data types -
  Nets - Represents structural connections between components
  Registers - Represents variables used to store data
- Verilog HDL Abstraction levels -
  Behavioral Models
  RTL Models
  Structural Models

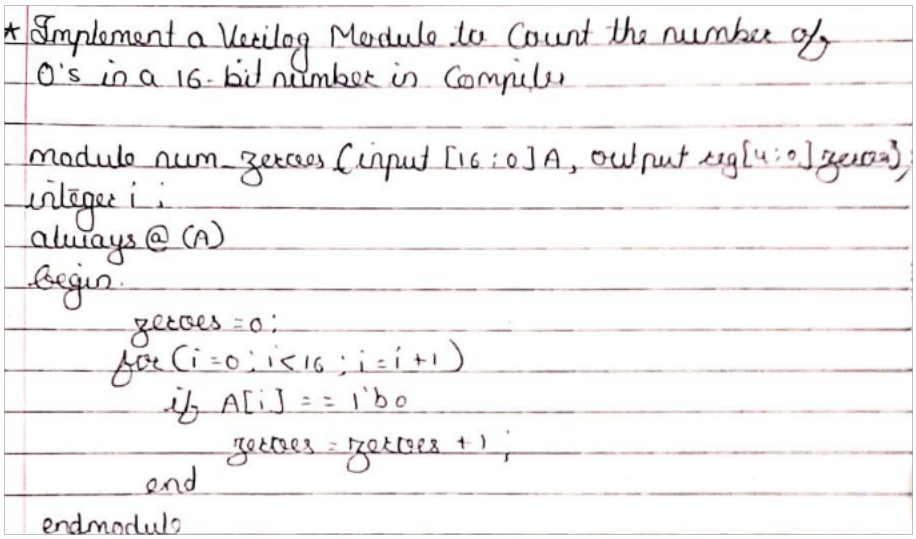
Building / Demo projects Using FPGA

- FPGAs are nothing but logic Blocks and interconnects that can be programmable by Hardware description languages (Verilog HDL / VHDL) to perform different complex functions.

Verilog code for Adder on FPGA :-

```
module fpga_adder (input A, B, Ci, output S, Co);
wire tmp1, tmp2, tmp3;
xor u1 (tmp1, A, B);
and u2 (tmp2, A, B);
and u3 (tmp3, tmp1, Ci);
or u4 (Co, tmp2, tmp3);
xor u5 (S, tmp1, Ci);
endmodule
```

```
library ieee;
use ieee.std_logic_1164.all;
entity fpga_adder is
    port (A, B, Ci : in std_logic;
          S, Co : out std_logic);
end fpga_adder;
architecture structural of fpga_adder is
signal tmp1, tmp2, tmp3 : std_logic;
begin
    tmp1 <= A xor B;
    tmp2 <= A and B;
    tmp3 <= tmp1 and Ci;
    Co <= tmp2 or tmp3;
    S <= tmp1 xor Ci;
end structural;
```



Date:	05-06-2020	Name:	Rajeshwari Gadagi
Course:	Python programming	USN:	4AL17EC076
Topic:	Application 10:build a data collector web app with PostgreSQL and flask.	Semester and section:	6th sem and B sec

Application-10: Build a Data Collector Web App with PostgreeSQL and Flask:

PostgreSQL Database web App with Flask: Step
Develop a HTML code for generating a webpage
Backend & frontend is developed
frontend sends the data to backend

FRONTEND → HTML

```
<!DOCTYPE html>
<html long="en">
<tittle> Data Collector App </tittle>
<head>
<link href="../static/main.css" rel="stylesheet">
</head>
<body>
<div class="container">
<h1> collecting height </h1>
<h3> Please fill the entries to get population statistics
                of height </h3>
<form action="success.html" method="POST">
<input tittle="Your email will be safe with us", place
        holdr="Enter your email address" type="email":
        name="email_name" required ><br>
<input tittle="your data will be safe with us" placeholder
        = enter height in cm" type="number" min="50",
        max="300", name="height_name" <br>
<button type="submit"> Submit </button>
</form>
</div>
</body>
</html>
```