Daily Assessment format

Name: Jyothi S Kumar
USN: 4AL17EC081

Date: 21/6/2020

Course: Digital design using HDL

Topic: FPGA Basics Architecture Applications
- Verilog HDL Basics by intel
- Verilog testbench code to verify the design under test (DUT)

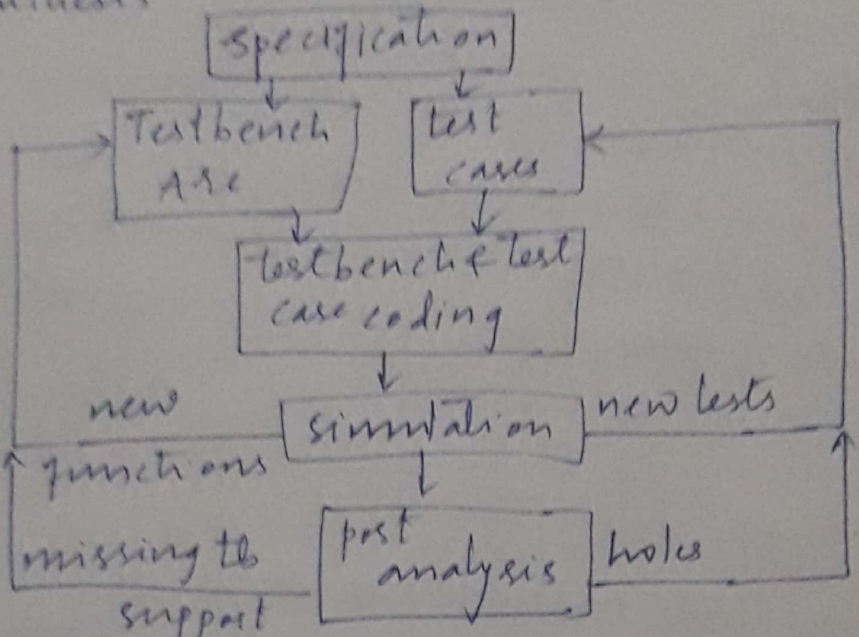
Github repository: jyothi-courses

forenoon session details

Image of session

Typical RTL synthesis & RTL simulation flows

synthesis



Report

what is FPGA?

the field-programmable gate array (FPGA) is an integrated ckt that consists of internal hardware blocks with user-progsammable interconnects to customize operation for a specific application. the interconnects can readily be reprogsammed. allowing an FPGA to accomodate changes to a design or even support a new applications during the lifetime of the part

The FPGA has its roots in earlier devices such as programmable read-only memories & programmable logic devices such as these devices could be programmed either at the factory or in the field, but they used fused technology & could not be changed once programmed. In contrast, FPGA stores its configuration information in a re-programmable medium such as static RAM (SRAM) or flash memory. FPGA manufactures include intel, xilinx Lattice semiconductor, microchip, technology & microsemi

FPGA Architecture

A basic FPGA Architecture consists of thousands of fundamental elements called configurable logic blocks (CLBs) surrounded by a system of programmable interconnects, called a fabric, that routes signals blw CLBs. input/output blocks interface blw the FPGA & external devices depending on the manufacturer. the CLB may also be referred to as a logic block (LB), a logic element or a logic cell. An individual CLB is made up of several logic blocks. A lookup table is a characteristic feature of an FPGA

CPLD vs FPGA:

Originally, FPGAs included the blocks in fig1. & little else but now designers can choose from products with a large range of features. less complex devices such as simple programmable logic devices (SPLDs) & complex programmable logic devices bridge the gap blw discrete logic devices. & entry-level FPGAs. entry-level FPGAs emphasize low power consumption, low logic density & low complexity per chip. Higher function devices add functional blocks dedicated to specific functions.

FPGA design

How do we transform this collection of thousands of hardware blocks into the correct configuration to execute the application? An FPGA based design begins by defining the required computing tasks in the development tool, then compiling them into a configuration file that contains information on how to hook up the CLBs & other modules. The process is similar to a software development cycle except that the goal is to architect the hardware itself.

FPGA Applications

Many applications rely on the parallel execution of identical operations; the ability to configure the FPGA's CLBs into hundreds or thousands of identical processing block has applications in image processing, artificial intelligence, data center hardware accelerators enterprise networking & automotive advanced driver assistance systems. Many of these application areas are changing very quickly as requirements evolve & new protocols & standards are adopted.

Basics of HDL

Verilog is a HARDWARE DESCRIPTION LANGUAGE (HDL). it is a language used for describing a digital system like a network switch or microprocessor or a memory or a flip-flop. It means, by using a HDL we can describe any digital hardware at any level.

Verilog supports a design of many levels of abstraction the major three are -

- Behavioural level
- Register-transfer level
- Gate level

Behavioral level

this level describes a system by concurrent algorithm every algorithm in sequential which means it consist of a set of instructions that are executed one by one. functions, tasks & blocks

Register_transfer level

Designs using the Register_transfer level specify the characteristics of a ckt using operations & the transfer of data

Gate level

within the Logical level, the characteristics of a syste are described by Logical links & their timing properties All signals are discrete signals. they can only have definite logical values.

Task for Day2

implement a 4:1 mux & write the test bench code to verify the module

```
module tb_4to1_mux;
reg[3:0] a,b,c,d;
wire[3:0] out;
reg[1:0] sel;
integer i;
mux_4to1_case mux0 (.a(a), .b(b), .c(c), d(d), .sel(sel),
                    .out(out));

initial begin
$monitor("[%0t]sel=0x%0h a=0x%0h b=0x%0h c=0x%0h
        d=0x%0h out=0x%0h", $time, sel, a,b,c,d);

sel<=0;
a<=$random;
b<=$random;
c<=$random;
d<=$ random;
```

```
for(i=1;i<=4;i=i+1) begin
#5 sel<=1;
end
#15 $finish;
end
endmodule
```

Date: 02/oct/2020
Course: python
Topic: Interactive data visualization with bokeh, webscraping with python beautiful soup

Name: Jyothi S Donur
Usn: 4AL17EC037

Github repository: jyoti-course

Afternoon session details

Report

Snippet producing the triangle based plot

```
# making a basic bokeh line graph
# importing Bokeh
from bokeh.plotting import figure
from bokeh.io import output_file, show
#prepare some data
x=[3,7,5,10]
y=[3,6,9]
#prepare the output file
output_file("Line.html")
# create a figure object
f=figure()
# create line plot
f.triangle(x,y)
show(f)
```

Snippet producing the circle based plot

```
# Making a basic bokeh line graph
# importing bokeh
from bokeh.plotting import figure
from bokeh.io import output_file, show

x=[3,7,5,10]
y=[3,6,9]
output_file("Line.html")
f=figure()
f.circle(x,y)
show(f)
```

Plot properties

You can add a title to the plot, set the figure width & height, change title font, etc below is a summary of properties which can be added to change the style of the plot

```
import pandas
from bokeh.plotting import figure, output_file, show
p=figure(plot_width=500, plot_height=400, tools='pan', logo=
 none)
p.title.text = "cool data"
p.title.text_color = "Gray"
p.title.text_font = "times"
p.title.text_font_style="bold"
p.xaxis.minor_tick_line_color=none
p.yaxis.minor_tick_line_color= none
p.xaxis.axis_label="Date"
p.yaxis.axis_label="Intensity"
p.line([1,2,3],[4,5,6])
output_file("graph.html")
show(p)
```

visual attributes

```
from bokeh.plotting import figure, output_file, show
p = figure(plot_width=400, plot_height=400, tools='pan,
        reset')
p.title.text = "Earthquakes"
p.title.text_color = "Orange"
p.title.text_font = "times"
p.title.text_font_style = "italic"
p.yaxis.minor_tick_line_color = "yellow"
p.xaxis.axis_label = "Times"
p.yaxis.axis_label = "value"
p.circle([1,2,3,4,5],[2,5,8,2,7], size=[i*2 for i in [8,12,4,6,5]],
    color="red", alpha=0.5)
output_file("scatter_plotting.html")
show(p)
```