EC2357-VLSI DESIGN LABORATORY
LABORATORY MANUAL
FOR SIXTH SEMESTER B.E (ECE)
(FOR PRIVATE CIRCULATION ONLY)
ACADEMIC YEAR (2013-2014)
ANNA UNIVERSITY, CHENNAI-25

DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING
DR. NAVALAR NEDUNCHEZHIYAN COLLEGE OF ENGINEERING
THOLUDUR – 606303, CUDDALORE DISTRICT.
SOFTWARE REQUIREMENTS:

XILINX ISE9.1V/CADENCE /MAGMA / TANNER.

UNIVERSITY PRACTICAL EXAMINATION

ALLOTMENT OF MARKS

| INTERNAL ASSESSMENT | 20 MARKS |
| PRACTICAL ASSESSMENT | 80 MARKS |

TOTAL | 100 MARKS

INTERNAL ASSESSMENT (20 MARKS)

Staff should maintain the assessment and the head of the department should monitor it.

SPLIT UP OF INTERNAL MARKS

| OBSERVATION | 3 MARKS |
| RECORD NOTE | 7 MARKS |
| MODEL EXAM | 5 MARKS |
| ATTENDANCE | 5 MARKS |

TOTAL | 20 MARKS

UNIVERSITY EXAMINATION

The Exam will be conducted for 100 marks. Then the marks will be converted to 80 marks.

ALLOCATION OF MARKS

| AIM AND RESULT | 10 MARKS |
| ALGORITHM & FLOWCHART | 20 MARKS |
| PROGRAM | 30 MARKS |
| EXECUTION | 30 MARKS |
| VIVA VOCE | 10 MARKS |

TOTAL | 100 MARKS
ANNA UNIVERSITY SYLLABUS
LIST OF EXPERIMENTS

1. Design Entry and simulation of combinational logic circuits (8 bit adders, 4 bit multipliers, address decoders, multiplexers), Test bench creation, functional verification, and concepts of concurrent and sequential execution to be highlighted.

2. Design Entry and simulation of sequential logic circuits (counters, PRBS generators, Accumulators) Test bench creation, functional verification, and concepts of concurrent and sequential execution to be highlighted.

3. Synthesis, P&R and Post P&R simulation for all the blocks/codes developed in Expt.No. 1 and No. 2 given above. Concepts of FPGA floor plan, critical path, design gate Count I/O configuration and pin assignment to be taught in this experiment.

4. Generation of configuration/fuse files for all the blocks/codes developed as part of Expt.1. And Expt. 2. FPGA devices must be configured and hardware tested for the blocks/codes developed as part of Expt. 1. and Expt. 2. The correctness of the inputs and outputs for each of the blocks must be demonstrated at least on oscilloscopes (logic analyser preferred).

5. Schematic Entry and SPICE simulation of MOS differential amplifier. Determination of gain, bandwidth, output impedance and CMRR.

6. Layout of a simple CMOS inverter, parasitic extraction and simulation.

7. Design of a 10 bit number controlled oscillator using standard cell approach, simulation followed by study of synthesis reports.

8. Automatic layout generation followed by post layout extraction and simulation of the circuit studied in Expt. No.7
<table>
<thead>
<tr>
<th>Ex. No</th>
<th>Name of the Experiment</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Design Entry and Simulation of Combinational Logic Circuits</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Design Entry and simulation of sequential logic circuit</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>Study of Synthesis Tools</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>Study of place and root annotation</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>Schematic Entry and SPICE Simulation</td>
<td>28</td>
</tr>
<tr>
<td>6</td>
<td>Layout of a CMOS Inverter</td>
<td>36</td>
</tr>
<tr>
<td>8</td>
<td>Design of a 10 bit number controlled oscillator</td>
<td>39</td>
</tr>
<tr>
<td>9</td>
<td>Automatic Layout Generation</td>
<td>41</td>
</tr>
</tbody>
</table>

**BEYOND THE SYLLABUS**

8. Implementation of Flip-flops

9. Implementation of Counters

10. Implementation of Registers

Question Bank

53

Signature of Staff Incharge
1) Study of Simulation using tools.

2) Design Entry and Simulation of Combinational Logic Circuits
   a) Basic logic gates
   b) Half adder and full adder
   c) Half Sub tractor and full sub tractor
   d) 8 bit adder
   e) 4 bit multiplier
   f) Encoder and Decoder
   g) Address Decoder
   h) Multiplexer

3) Design Entry and Simulation of Sequential Logic Circuits
   a) Flip-Flops
   b) Counter
   c) PRBS generator
   d) Accumulator

4) Study of Synthesis tools

5) Place and Route and Back annotation for FPGAs

6) Schematic Entry and SPICE Simulation
   a) CMOS Inverter
   b) Universal Gate
   c) Differential Amplifier

7) Layout of a CMOS Inverter

8) Design of a 10 bit number controlled oscillator

9) Automatic Layout Generation
ASIC DESIGN FLOW

SYSTEM REQUIREMENTS

SPECIFICATION

SPECIFICATION

LOGIC DESIGN

MODELLING

LOGIC DESIGN

SYNTHESIS

VERIFICATION

TEST GENERATION

PHYSICAL

SIGN-OFF / Mapping

MANUFACT / Place & Route

/ Configuration data

PROTOTYPE

PROTO VERIFICATION

SYSTEM TESTING
Experiment Number: 1

Title of the experiment: Design Entry and Simulation of Combinational Logic Circuits

Date of the experiment: 

OBJECTIVE OF THE EXPERIMENT

To study about the simulation tools available in Xilinx project navigator using Verilog tools.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Xilinx Project navigator – ISE 8.1</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double click the project navigator and select the option File-New project.</td>
</tr>
<tr>
<td>2</td>
<td>Give the project name.</td>
</tr>
<tr>
<td>3</td>
<td>Select Verilog module.</td>
</tr>
<tr>
<td>4</td>
<td>Type your Verilog coding.</td>
</tr>
<tr>
<td>5</td>
<td>Check for syntax.</td>
</tr>
<tr>
<td>6</td>
<td>Select the new source of test bench waveform</td>
</tr>
<tr>
<td>7</td>
<td>Choose behavioral simulation and simulate it by Xilinx ISE simulator.</td>
</tr>
<tr>
<td>8</td>
<td>Verify the output.</td>
</tr>
</tbody>
</table>
c) Verilog coding:

**Logic gates:**

**AND GATE:**

```verilog
module gl(a,b,c);
    input a;
    input b;
    output c;
    and(c,a,b);
end module
```

**OR GATE:**

```verilog
module gl(a,b,c);
    input a;
    input b;
    output c;
    or(c,a,b);
end module
```

**XOR GATE:**

```verilog
module gl(a,b,c);
    input a;
    input b;
    output c;
    xor (c,a,b);
end module
```

**NAND GATE:**

```verilog
module gl(a,b,c);
```
input a;
input b;
output c;
nand(c,a,b);
end module

**NOR GATE:**

module gl(a,b,c);
input a;
input b;
output c
nor(c,a,b);
end module

**HALF ADDER:**

module half adder(a,b,c,s);
input a;
input b;
output c;
output s;
xor(s,a,b);
and(c,~a,b);
end module

**HALF SUBTRACTOR:**

module half sub(a,b,c,s);
input a;
input b;
output c;
output s;

xor(s,a,b);

and(c,~a,b);

end module

**ENCODER**

module Encd2to4(i0, i1, i2, i3, out0, out1);

input i0,i1, i2, i3;

output out0, out1;

reg out0,out1;

always@(i0,i1,i2,i3)

case({i0,i1,i2,i3})

4'b1000: {out0,out1}=2'b00;

4'b0100: {out0,out1}=2'b01;

4'b0010: {out0,out1}=2'b10;

4'b0001: {out0,out1}=2'b11;

default: $display("Invalid");

de case
end module

**DECODER:**

// Module Name: Decd2to4

module Decd2to4(i0, i1, out0, out1, out2, out3);

input i0, i1;

output out0, out1, out2, out3;

reg out0,out1,out2,out3;

always@(i0,i1)

case({i0,i1})

2'b00:

{out0,out1,out2,out3}=4'b1000;

2'b01:

{out0,out1,out2,out3}=4'b0100;

2'b10:
\{out0, out1, out2, out3\} = 4'b0010;
2'b11:
\{out0, out1, out2, out3\} = 4'b0001;
default:
$display("Invalid");
endcaseendmodule

MULTIPLEXER:

// Module Name: Mux4to1
module Mux4to1(i0, i1, i2, i3, s0, s1, out);
input i0, i1, i2, i3, s0, s1;
output out;
wire s1n, s0n;
wire y0, y1, y2, y3;
not (s1n, s1);
not (s0n, s0);
and (y0, i0, s1n, s0n);
and (y1, i1, s1n, s0);
and (y2, i2, s1, s0n);
and (y3, i3, s1, s0);
or (out, y0, y1, y2, y3);
endmodule

DEMULTIPLEXER:

// Module Name: Dux1to4
module Dux1to4(in, s0, s1, out0, out1, out2, out3);
input in, s0, s1;
output out0, out1, out2, out3;
wire s0n, s1n;
not (s0n, s0);
not (s1n, s1);
and (out0, in, s1n, s0n);
and (out1,in,s1n,s0);
and (out2,in,s1,s0n);
and (out3,in,s1,s0);
endmodule

8 BIT ADDER
module adder(a,b, s,c);
input [7:0] a,b;
output [7:0] s,c;
assign {c,s} = a + b;
endmodule

MULTIPLIER
module multi(a,b, c);
input [3:0] a,b;
output [7:0] c;
assign c = a * b;
endmodule

RESULT:
Thus the program for study of simulation using tools and the output also verified successfully.
Experiment Number: 2

Title of the experiment : Design Entry and simulation of sequential logic circuits

Date of the experiment :

OBJECTIVE OF THE EXPERIMENT

To study about the simulation tools available in Xilinx project navigator using Verilog tools.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Xilinx Project navigator – ISE 8.1</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double click the project navigator and select the option File-New project.</td>
</tr>
<tr>
<td>2</td>
<td>Give the project name</td>
</tr>
<tr>
<td>3</td>
<td>Select Verilog module.</td>
</tr>
<tr>
<td>4</td>
<td>Type your verilog coding</td>
</tr>
<tr>
<td>5</td>
<td>Check for syntax</td>
</tr>
<tr>
<td>6</td>
<td>Select (view RTL schematic) from the synthesis-xst menu.</td>
</tr>
<tr>
<td></td>
<td>Verify the logic circuit and equivalent parameters.</td>
</tr>
</tbody>
</table>
Verilog coding:

**PRBS GENERATORS**

module prbs(a,clk,clr);
output [3:0] a;
input clk,clr;
reg [3:0] tmp;
always @(posedge clk or posedge clr)
begin
if(clr)
begin
tmp = 4'b1111;
end
else
begin
tmp = { tmp[0]^tmp[1],tmp[3],tmp[2],tmp[1]};
end
end
assign a=tmp;
endmodule

**ACCUMULATOR:**

module acc(indata, clk,clr, outdata);
input [3:0] indata;
input clk,clr;
output [3:0] outdata;
reg [3:0] outdata;
always @(posedge clk or posedge clr)
begin
if(clr)
outdata <= 4'd0;
else
outdata <= indata;
2- BIT COUNTER:

// Module Name: Count2Bit
module Count2Bit(Clock, Clear, out);
input Clock, Clear;
output [1:0] out;
reg [1:0] out;
always@(posedge Clock, negedge Clear)
if((~Clear) || (out>=4))
  out=2'b00;
else
  out=out+1;
endmodule

RESULT:

Thus the program for study of simulation using tools and the output also verified successfully.
Experiment Number: 3

Title of the experiment: Study of Synthesis Tools

Date of the experiment: 

OBJECTIVE OF THE EXPERIMENT

To study about synthesis tools available in Xilinx project navigator.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Xilinx Project navigator – ISE 8.1</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double click the project navigator and select the option File-New project.</td>
</tr>
<tr>
<td>2</td>
<td>Give the project name</td>
</tr>
<tr>
<td>3</td>
<td>Select Verilog module.</td>
</tr>
<tr>
<td>4</td>
<td>Type your verilog coding</td>
</tr>
<tr>
<td>5</td>
<td>Check for syntax</td>
</tr>
<tr>
<td>6</td>
<td>Select (view RTL schematic) from the synthesis-xst menu.</td>
</tr>
<tr>
<td></td>
<td>Verify the logic circuit and equivalent parameters.</td>
</tr>
</tbody>
</table>
THEORY:

Now that you have created the source files, verified the design behaviour with simulation, and added constraints, you are ready to synthesize and implement the design.

**Implementing the Design:**

1. Select the **counter** source file in the Sources in Project window.
2. In the Processes for Source window, click the “+” sign next to **Implement Design**. The Translate, Map, and Place & Route processes are displayed. Expand those processes as well by clicking on the “+” sign. You can see that there are many sub-processes and options that can be run during design implementation.
3. Double-click the top level **Implement Design** process. ISE determines the current state of your design and runs the processes needed to pull your design through implementation. In this case, ISE runs the Translate, Map and PAR processes. Your design is now pulled through to a placed-and-routed state. This feature is called the “pull through model.”
4. After the processes have finished running, notice the status markers in the Processes for Source window. You should see green checkmarks next to several of the processes, indicating that they ran successfully. If there are any yellow exclamation points, check the warnings in the Console tab or the Warnings tab within the Transcript window. If a red X appears next to a process, you must locate and fix the error before you can continue.
Figure 1: Floor planner View - Detailed View

Figure 2: Design Summary View
VERIFICATION OF SYNTHESIS:

Your synthesized design can be viewed as a schematic in the Register Transfer Level (RTL) Viewer. The schematic view shows gates and elements independent of the targeted Xilinx® device.

1. In the Processes for Source window, double-click View RTL Schematic found in the Synthesize - XST process group. The top level schematic representation of your synthesized design opens in the workspace.

2. Right-click on the symbol and select Push Into the Selected Instance to view the schematic in detail. The Design tab appears in the Sources in Project window, enabling you to view the design hierarchy. In the schematic, you can see the design components you created in the HDL source, and you can “push into” symbols to view increasing levels of detail.

3. Close the schematic window.

RESULT:

Thus the program to study about synthesis tools available in Xilinx project navigator and the output also verified successfully.
Experiment Number: 4

Title of the experiment : Study of Place and Root-Back Annotation

Date of the experiment :

OBJECTIVE OF THE EXPERIMENT

To study about place and back annotation in xilinx project navigator using verilog coding.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Xilinx ISE – 9.1 navigator.</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment:

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double click the project navigator and select the option File-New project.</td>
</tr>
<tr>
<td>2</td>
<td>Give the project name.</td>
</tr>
<tr>
<td>3</td>
<td>Select Verilog module</td>
</tr>
<tr>
<td>4</td>
<td>Type your verilog coding</td>
</tr>
<tr>
<td>5</td>
<td>Check for syntax.</td>
</tr>
<tr>
<td>6</td>
<td>Assign package pins and view floor planner diagram of the FPGA.</td>
</tr>
<tr>
<td>7</td>
<td>Make necessary changes in the diagram if required.</td>
</tr>
<tr>
<td>8</td>
<td>Save the changes, back annotate the changed constraints</td>
</tr>
<tr>
<td>9</td>
<td>Verify the UCF that changes are updated</td>
</tr>
<tr>
<td>10</td>
<td>Get the length of the sequence as N.</td>
</tr>
</tbody>
</table>

THEORY:

After implementation is complete, you can verify your design before downloading it to a device.

Viewing Placement:

In this section, you will use the Floor planner to verify your pin outs and placement. Floor planner is also very useful for creating area groups for designs.

1. Select the counter source file in the Sources in Project window.
2. Click the “+” sign to expand the **Place & Route** group of processes.

3. Double-click the **View/Edit Placed Design (Floor planner)** process. The Floor planner view opens.

4. Select **View _ Zoom _ To Box** and then use the mouse to draw a box around the counter instance, shown in green on the right side of the chip.

5. This Fig 8 shows where the entire design was placed. Click on any of the components listed in the Design Hierarchy window to see where each component is placed.

6. Zoom in to the right side of the chip even more, and place your mouse over the K13pad. You can see that your pin out constraint was applied - the DIRECTION pin is placed at K13.

7. Close the Floor planner without saving.

**Viewing Resource Utilization in Reports:**

Many ISE processes produce summary reports which enable you to check information about your design after each process is run. Detailed reports are available from the Processes for Source window. You can also view summary information and access most often-utilized reports in the Design Summary.

1. Click on the **Design Summary** tab at the bottom of the window. If you closed the summary during this tutorial, you can reopen it by double clicking the **View Design Summary** process.

![Timing Analyzer - Timing Summary](image)

**Figure 1:** **Timing Analyzer - Timing Summary**

2. In the Device Utilization Summary section, observe the number of Slice Flip Flops that were used during implementation. You should see 4 flip flops, since you implemented a 4-bit counter.

3. To see other reports, scroll to the bottom of the Design Summary. You can click on a report from here to view it in the ISE Text Editor.
Timing Closure:

In this section, you will run timing analysis on your design to verify that your timing constraints were met. Timing closure is the process of working on your design to ensure that it meets your necessary timing requirements. ISE provides several tools to assist with timing closure.

1. In the Processes for Source window, under the Place & Route group of processes, expand the Generate Post-Place & Route Static Timing group by clicking the “+” sign.

2. Double-click the Analyse Post-Place & Route Static Timing process. The Timing Analyser opens.

3. To analyse the design, select Analyse Against Timing Constraints. The Analyse with Timing Constraints dialog box opens.

4. Click OK. When analysis is complete, the timing report opens.

5. Select Timing summary from the Timing Report Description tree in the left window. This displays the summary section of the timing report, where you can see that no timing errors were reported. Close the Timing Analyser without saving.

Figure 2: FPGA Editor - Detailed View
Viewing the Placed and Routed Design:

In this section, you will use the FPGA Editor to view the design. You can view your design on the FPGA device, as well as edit the placement and routing with the FPGA Editor.

1. Double-click the View/Edit Routed Design (FPGA Editor) process found in the Place & Route group of processes. Your implemented design opens in the FPGA Editor.

2. Look in the List window to examine your design components.

3. Click on the COUNT_OUT K12 IOB in the List window to select the row. This is one of the outputs in your design.

4. With the COUNT_OUT K12 row selected, select View _ Zoom Selection. In the editor window, you can see the COUNT_OUT<0> IOB highlighted in red.

5. Push into (double-click) the red-highlighted COUNT_OUT K12 IOB. You should see Fig 2.

6. Enlarge the window and zoom in so you can see more detail. This view shows the inside of an FPGA at the lowest viewable level. The blue line shows the route that is used through the IOB. The red lines show the routes that are available.

7. Verify that the signal goes to the pad as an output.

8. Close the FPGA Editor.
Figure 3: Simulator Processes for Test Bench
Figure 4: **Timing Simulation in ISE Simulator**

Timing Simulation (ISE Simulator):

can verify that your design meets your timing requirements by running a timing simulation. You can use the same test bench waveform that was used earlier in the design flow for behavioral simulation. When running timing simulation, the ISE tools create a structural HDL file which includes timing information available after Place and Route is run. The simulator will run on a model that is created based on the design to be downloaded to the FPGA. If you are using ISE Base or Foundation, you can simulate your design with the ISE Simulator. To simulate your design with Modalism, skip to the “Timing Simulation (ModelSim)” section.

To run the integrated simulation processes:

1. Select the test bench waveform in the Sources in Project window. You can see the ISE Simulator processes in the Processes for Source window.
2. Double-click the Simulate Post-Place & Route Model process. This process generates a timing-annotated net list from the implemented and simulates it. The resulting simulation is displayed in the Waveform Viewer. These results look different than those you saw in the behavioural simulation earlier in this tutorial. These results show timing delays.
3. To see your simulation results, zoom in on the transitions and view the area between 300 ns and 900 ns to verify that the counter is counting up and down as directed by the stimulus on the DIRECTION port.

4. Zoom in again to see the timing delay between a rising clock edge and output transition.

5. Click the Measure Marker button and then click near the 300 ns mark. Drag the second marker to the point where the output becomes stable to see the time delay between the clock edge and the transition.

6. Close the waveform view window. You have completed timing simulation of your design using the ISE Simulator. Skip past the ModelSim section below, and proceed to the “Creating Configuration Data” section.

RESULT

Thus the program for perform the place and root-back annotation was studied and the output also verified successfully.


**Experiment Number: 5**

**Title of the experiment**: CMOS INVERTER

**Date of the experiment**: 

---

**OBJECTIVE OF THE EXPERIMENT**

To perform the functional verification of the CMOS Inverter through schematic entry.

**FACILITIES REQUIRED AND PROCEDURE**

**a) Facilities required to do the experiment**

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S-Edit using cadance Tool.</td>
<td></td>
</tr>
</tbody>
</table>

**b) Procedure for doing the experiment**

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Draw the schematic of CMOS Inverter using S-edit.</td>
</tr>
<tr>
<td>2</td>
<td>Perform Transient Analysis of the CMOS Inverter.</td>
</tr>
<tr>
<td>3</td>
<td>Obtain the output waveform from W-edit</td>
</tr>
<tr>
<td>4</td>
<td>Obtain the spice code using T-edit</td>
</tr>
</tbody>
</table>

**c) THEORY:**

Inverter consists of nMOS and pMOS transistor in series connected between VDD and GND. The gate of the two transistors are shorted and connected to the input. When the input to the inverter \( A = 0 \), nmos transistor is OFF and pMOS transistor is ON. The output is pull-up to VDD. When the input \( A = 1 \), nMOS transistor is ON and pMOS transistor is OFF. The Output is Pull-down to GND.
SCHEMATIC DIAGRAM:

SIMULATED WAVEFORM:

RESULT

Thus the functional verification of the CMOS Inverter through schematic entry and the output also verified successfully.
Experiment Number: 6

Title of the experiment : UNIVERSAL GATES

Date of the experiment : 

OBJECTIVE OF THE EXPERIMENT
To perform the functional verification of the universal gate through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S-Edit using CadanceTool.</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Draw the schematic of CMOS Inverter using S-edit.</td>
</tr>
<tr>
<td>2</td>
<td>Perform Transient Analysis of the CMOS Inverter.</td>
</tr>
<tr>
<td>3</td>
<td>Obtain the output waveform from W-edit.</td>
</tr>
<tr>
<td>4</td>
<td>Obtain the spice code using T-edit</td>
</tr>
</tbody>
</table>
NAND GATE

SIMULATED WAVEFORM:

<table>
<thead>
<tr>
<th>TSPICE 15.10</th>
<th>C:\DOCUME<del>1\VLSI\LOCALS</del>1\Temp\nand.sp</th>
<th>12:36:25 PM 11/30/2011</th>
</tr>
</thead>
</table>
RESULT:

Thus the functional verification of the NAND & NOR Gate through schematic entry and the output also verified successfully.
Experiment Number: 7

Title of the experiment : DIFFERENTIAL AMPLIFIER
Date of the experiment : 

OBJECTIVE OF THE EXPERIMENT

To calculate the gain, bandwidth and CMRR of a differential amplifier through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S-Edit using CadanceTool.</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Draw the schematic of differential amplifier using S-edit and generate the symbol.</td>
</tr>
<tr>
<td>2</td>
<td>Draw the schematic of differential amplifier circuit using the generated symbol.</td>
</tr>
<tr>
<td>3</td>
<td>Perform AC Analysis of the differential amplifier.</td>
</tr>
<tr>
<td>4</td>
<td>Obtain the frequency response from W-edit.</td>
</tr>
<tr>
<td>5</td>
<td>Obtain the spice code using T-edit.</td>
</tr>
</tbody>
</table>
SCHEMATIC DIAGRAM:
RESULT

Thus the functional verification of the **Differential Amplifier** through schematic entry, and the output also verified successfully.
Experiment Number: 8

Title Of The Experiment : LAYOUT OF CMOS INVERTER

Date of the experiment :

OBJECTIVE OF THE EXPERIMENT

To draw the layout of CMOS Inverter using L-Edit and extract the SPICE code.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L-Edit using CadanceTool.</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Draw the CMOS Inverter layout by obeying the Lamda Rules using Ledit.</td>
</tr>
<tr>
<td>2</td>
<td>Poly - 2\lambda.</td>
</tr>
<tr>
<td></td>
<td>ii. Active contact - 2 \lambda.</td>
</tr>
<tr>
<td></td>
<td>iii. Active Contact – Metal - 1 \lambda.</td>
</tr>
<tr>
<td></td>
<td>iv. Active Contact – Active region - 2 \lambda.</td>
</tr>
<tr>
<td></td>
<td>v. Active Region – Pselect - 3 \lambda.</td>
</tr>
<tr>
<td></td>
<td>vi. Pselect – nWell - 3\lambda.</td>
</tr>
<tr>
<td>3</td>
<td>Check DRC to verify whether any region violate the lamda rule.</td>
</tr>
<tr>
<td>4</td>
<td>Setup the extraction and extract the spice code using T-spice.</td>
</tr>
</tbody>
</table>
CMOS INVERTER:

SIMULATED WAVEFORM:
RESULT:

Thus the layout of CMOS Inverter using L-Edit and extract the SPICE code, and the output also verified successfully.
Experiment Number: 9

Title of the Experiment: DESIGN OF A 10 BIT NUMBER CONTROLLED OSCILLATOR

Date of the experiment:

OBJECTIVE OF THE EXPERIMENT

To perform the functional verification of the design of a 10 bit number controlled oscillator through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S-Edit using CadenceTool</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Draw the schematic of CMOS Inverter using S-edit.</td>
</tr>
<tr>
<td>2</td>
<td>Perform Transient Analysis of the CMOS Inverter</td>
</tr>
<tr>
<td>3</td>
<td>Obtain the output waveform from W-edit</td>
</tr>
<tr>
<td>4</td>
<td>Obtain the spice code using T-edit.</td>
</tr>
</tbody>
</table>

SCHEMATIC DIAGRAM:
RESULT:

Thus the functional verification of the design of a 10 bit number controlled oscillator through schematic entry.
Experiment Number: 10

Title of The Experiment : AUTOMATIC LAYOUT GENERATION

Date of the experiment :

OBJECTIVE OF THE EXPERIMENT

To generate the Layout from the schematic using the Tanner tool and verify the layout using simulation.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S-Edit, L-Edit using Tanner Tool</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Draw the schematic using S Edit and verify the output in W Edit.</td>
</tr>
<tr>
<td>2</td>
<td>Extract the schematic and store it in another location</td>
</tr>
<tr>
<td>3</td>
<td>Open the L Edit, open the design in Ring VCO</td>
</tr>
<tr>
<td>4</td>
<td>Create the new cell</td>
</tr>
<tr>
<td>5</td>
<td>Open the schematic file (.sdl) using the SDL Navigator</td>
</tr>
<tr>
<td>6</td>
<td>Do the necessary connections as per the design.</td>
</tr>
<tr>
<td>7</td>
<td>Name the ports and check the design for the DRC Rules</td>
</tr>
<tr>
<td>8</td>
<td>Locate the Destination file in the setup Extract window and extract the layout.</td>
</tr>
<tr>
<td>9</td>
<td>Include the Library and the print voltage statements in the net list which is obtained.</td>
</tr>
<tr>
<td>10</td>
<td>Verify the layout design using W Edit.</td>
</tr>
</tbody>
</table>
SCHEMATIC DIAGRAM:
LAYOUT:

[Diagram of a circuit layout with labels VDD, IN, GND, S, G, D, and OUT.]
RESULT:

Thus the layout from the schematic using the Cadance tool and verify the layout using simulation and the output also verified successfully.
BEYOND THE SYLLABUS
Experiment Number: 11

Title of the experiment : IMPLEMENTATION OF FLIP-FLOPS

Date of the experiment :

OBJECTIVE OF THE EXPERIMENT

To implement Flip-flops using Verilog HDL.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Xilinx Project navigator – ISE 8.1</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double click the project navigator and select the option File-New project.</td>
</tr>
<tr>
<td>2</td>
<td>Give the project name.</td>
</tr>
<tr>
<td>3</td>
<td>Select Verilog module.</td>
</tr>
<tr>
<td>4</td>
<td>Type your verilog coding.</td>
</tr>
<tr>
<td>5</td>
<td>Check for syntax.</td>
</tr>
<tr>
<td>6</td>
<td>Select the new source of test bench waveform</td>
</tr>
<tr>
<td>7</td>
<td>Choose behavioral simulation and simulate it by xilinx ISE simulator.</td>
</tr>
<tr>
<td>8</td>
<td>Verify the output.</td>
</tr>
</tbody>
</table>
c) Verilog coding:

PROGRAM:

D Flip-Flop:
// Module Name: DFF

module DFF(Clock, Reset, d, q);
    input Clock;
    input Reset;
    input d;
    output q;
    reg q;
    always@(posedge Clock or negedge Reset)
        if (~Reset) q=1'b0;
        else q=d;
endmodule

T Flip-Flop:
// Module Name: TFF

module TFF(Clock, Reset, t, q);
    input Clock;
    input Reset;
    input t;
    output q;
    reg q;
    always@(posedge Clock, negedge Reset)
        if (~Reset) q=0;
        else if (t) q=¬q;
endmodule
else q=q;
endmodule

JK Flip-Flop:

Program:

// Module Name: JKFF
module JKFF(Clock, Reset, j, k, q);
    input Clock;
    input Reset;
    input j;
    input k;
    output q;
    reg q;
    always@(posedge Clock, negedge Reset)
    if(~Reset)q=0;
    else
        begin
            case({j,k})
                2'b00: q=q;
                2'b01: q=0;
                2'b10: q=1;
                2'b11: q=~q;
            endcase
        end
endmodule

RESULT:

Thus the flip-flops program was implemented using tools and the output also verified successfully.
Experiment Number: 12

Title of the experiment : IMPLEMENTATION OF COUNTERS

Date of the experiment :

OBJECTIVE OF THE EXPERIMENT

To implement Counters using Verilog HDL.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Xilinx Project navigator – ISE 8.1</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double click the project navigator and select the option File-New project.</td>
</tr>
<tr>
<td>2</td>
<td>Give the project name.</td>
</tr>
<tr>
<td>3</td>
<td>Select Verilog module.</td>
</tr>
<tr>
<td>4</td>
<td>Type your verilog coding.</td>
</tr>
<tr>
<td>5</td>
<td>Check for syntax.</td>
</tr>
<tr>
<td>6</td>
<td>Select the new source of test bench waveform</td>
</tr>
<tr>
<td>7</td>
<td>Choose behavioral simulation and simulate it by xilinx ISE simulator.</td>
</tr>
<tr>
<td>8</td>
<td>Verify the output.</td>
</tr>
</tbody>
</table>
c) **Verilog coding:**

**PROGRAM:**

2- Bit Counter:

// Module Name:  Count2Bit

module Count2Bit(Clock, Clear, out);

    input Clock;
    input Clear;
    output [1:0] out;

    reg [1:0]out;

    always@(posedge Clock, negedge Clear)
        if((~Clear) || (out>=4))out=2'b00;
    else out=out+1;

endmodule

**RESULT:**

Thus the counters program was implemented using tools and the output also verified successfully.
Experiment Number: 13

Title of the experiment : IMPLEMENTATION OF REGISTERS

Date of the experiment :

OBJECTIVE OF THE EXPERIMENT

To implement Registers using Verilog HDL.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

<table>
<thead>
<tr>
<th>S.No.</th>
<th>SOFTWARE REQUIREMENTS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Xilinx Project navigator – ISE 8.1</td>
<td>1</td>
</tr>
</tbody>
</table>

b) Procedure for doing the experiment

<table>
<thead>
<tr>
<th>S.No</th>
<th>Details of the step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double click the project navigator and select the option File-New project.</td>
</tr>
<tr>
<td>2</td>
<td>Give the project name.</td>
</tr>
<tr>
<td>3</td>
<td>Select Verilog module.</td>
</tr>
<tr>
<td>4</td>
<td>Type your verilog coding.</td>
</tr>
<tr>
<td>5</td>
<td>Check for syntax.</td>
</tr>
<tr>
<td>6</td>
<td>Select the new source of test bench waveform</td>
</tr>
<tr>
<td>7</td>
<td>Choose behavioral simulation and simulate it by xilinx ISE simulator.</td>
</tr>
<tr>
<td>8</td>
<td>Verify the output.</td>
</tr>
</tbody>
</table>
c) **Verilog coding:**

**PROGRAM:**

2 – Bit Register:

// Module Name: Reg2Bit

module Reg2Bit(Clock, Clear, in, out);

    input Clock;
    input Clear;
    input [0:1] in;
    output [0:1] out;

    reg [0:1] out;

    always@(posedge Clock, negedge Clear)
        if(~Clear) out=2'b00;
        else out=in;

endmodule

**RESULT:**

Thus the Registers program was implemented using tools and the output also verified successfully.
**VIVA VOCE QUESTION & ANSWERS**

1. **What are four generations of Integration Circuits?**
   
   1) SSI (Small Scale Integration) 2) MSI (Medium Scale Integration) 3) LSI (Large Scale Integration) 4) VLSI (Very Large Scale Integration)

2. **Give the advantages of IC?**
   
   1) Size is less 2) High Speed 3) Less Power Dissipation

3. **Give the variety of Integrated Circuits?**
   
   1) More Specialized Circuits 2) Application Specific Integrated Circuits (ASICs) Systems-On-Chips

4. **Give the basic process for IC fabrication?**
   
   Silicon wafer Preparation, Epitaxial Growth, Oxidation, Photolithography, Diffusion Ion Implantation, Isolation technique, Metallization, Assembly processing & Packaging

5. **What are the various Silicon wafer Preparation?**
   
   Crystal growth & doping, Ingot trimming & grinding, Ingot slicing, Wafer polishing & etching, Wafer cleaning.

6. **Different types of oxidation?**
   
   Dry & Wet Oxidation

7. **What is the transistors CMOS technology provides?**
   
   n-type transistors & p-type transistors.

8. **What are the different layers in MOS transistors?**
   
   Drain, Source & Gate

9. **What is Enhancement mode transistor?**
   
   The device that is normally cut-off with zero gate bias.

10. **What is Depletion mode Device?**
    
    The Device that conduct with zero gate bias.

11. **When the channel is said to be pinched –off?**
    
    If a large Vds is applied this voltage with deplete the Inversion layer. This Voltage
effectively pinches off the channel near the drain.

12. Give the different types of CMOS process? p-well process, n-well process, Silicon-On-Insulator Process, Twin-tub Process

13. What are the steps involved in twin-tub process?
   Tub Formation, Thin-oxide Construction, Source & Drain Implantation, Contact cut definition, Metallization.

14. What are the advantages of Silicon-on-Insulator process?
   No Latch-up, Due to absence of bulks transistor structures are denser than bulk silicon.

15. Define Short Channel devices?
   Transistors with Channel length less than 3-5 microns are termed as Short channel devices. With short channel devices the ratio between the lateral & vertical dimensions are reduced. Non-Saturated Region Saturated Region

16. Define Threshold voltage in CMOS?
   The Threshold voltage, VT for a MOS transistor can be defined as the voltage applied between the gate and the source of the MOS transistor below which the drain to source current, IDS effectively drops to zero.

17. What is Body effect?
   The threshold voltage VT is not a constant w. r. to the voltage difference between the substrate and the source of MOS transistor. This effect is called substrate-bias effect or body effect.

18. What is Verilog?
   Verilog is a general purpose hardware descriptor language. It is similar in syntax to the C programming language. It can be used to model a digital system at many levels of abstraction ranging from the algorithmic level to the switch level.
19. What are the various modeling used in Verilog


20. What is the structural gate-level modeling?

Structural modeling describes a digital logic networks in terms of the components that make up the system. Gate-level modeling is based on using primitive logic gates and specifying how they are wired together.

21. What is Switch-level modeling?

Verilog allows switch-level modeling that is based on the behavior of MOSFETs. Digital circuits at the MOS-transistor level are described using the MOSFET switches.

22. What are identifiers?

Identifiers are names of modules, variables and other objects that we can reference in the design. Identifiers consists of upper and lower case letters, digits 0 through 9, the underscore character(_) and the dollar sign($). It must be a single group of characters.

Examples: A014, a ,b, in_o, s_out