DEPARTMENT
OF
ELECTRONICS & COMMUNICATION ENGINEERING

ADVANCED DIGITAL DESIGN LABORATORY MANUAL

VI Semester (21EC62)

Autonomous Course2023-2024

HOD : Dr. Manjunath T C
In -Charge:Dr.Madhura.R

Name of the Student	**:**	
Semester /Section	**:**	
USN	**:**	
Batch	**:**	

Dayananda Sagar College of Engineering

Shavige Malleshwara Hills, Kumaraswamy Layout,
Banashankari, Bangalore-560078, Karnataka
Tel : +91 80 26662226 26661104 Extn : 2731 Fax : +90 80 2666 0789
Web - http://www.dayanandasagar.edu Email : hod-ece@dayanandasagar.edu
(An Autonomous Institute Affiliated to VTU, Approved by AICTE & ISO 9001:2008 Certified)
(Accredited by NBA, National Assessment & Accreditation Council (NAAC) with 'A' grade)

Dayananda Sagar College of Engineering

Dept. of E & C Engg

Name of the Laboratory	**:**	ADD LAB
Semester/Year	**:**	VI/ 2024 (Autonomous)
No. of Students/Batch	**:**	20
No. of Equipment's	**:**	20
Major Equipment's	**:**	Model sim, Cadence NC cadence, INTEL FPGA kits
Area in square meters	**:**	109 Sq Mts
Location	**:**	Level – 2
Total Cost of Lab	**:**	Rs. 37, 77,535/-

Lab In charge/s: Dr Madhura R

Instructor : Mr. Pradyumna & Mr. Vijay

HOD : Dr. T.C Manjunath

About the College & the Department

The Dayananda Sagar College of Engineering was established in 1979, was founded by Sri R. Dayananda Sagar and is run by the Mahatma Gandhi Vidya Peetha Trust (MGVP). The college offers undergraduate, post-graduates and doctoral programmes under Visvesvaraya Technological University & is currently autonomous institution. MGVP Trust is an educational trust and was promoted by Late. Shri. R. Dayananda Sagar in 1960. The Trust manages 28 educational institutions in the name of "Dayananda Sagar Institutions" (DSI) and multi – Specialty hospitals in the name of Sagar Hospitals - Bangalore, India. Dayananda Sagar College of Engineering is approved by All India Council for Technical Education (AICTE), Govt. of India and affiliated to Visvesvaraya Technological University. It has widest choice of engineering branches having 16 Under Graduate courses & 17 Post Graduate courses. In addition, it has 21 Research Centres in different branches of Engineering catering to research scholars for obtaining Ph.D under VTU. Various courses are accredited by NBA & the college has a NAAC with ISO certification. One of the vibrant & oldest dept is the ECE dept. & is the biggest in the DSI group with 70 staffs & 1200+ students with 10 Ph.D.'s & 30^{+} staffs pursuing their research in various universities. At present, the department runs a UG course (BE) with an intake of 240 & 2 PG courses (M.Tech.), viz., VLSI Design Embedded Systems & Digital Electronics & Communications with an intake of 18 students each. The department has got an excellent infrastructure of 10 sophisticated labs & dozen class room, R & D centre, etc…

Vision & Mission of the Institute

Vision of the Institute

To impart quality technical education with a focus on Research and Innovation emphasizing on Development of Sustainable and Inclusive Technology for the benefit of society.

Mission of the Institute

- To provide an environment that enhances creativity and Innovation in pursuit of Excellence.
- To nurture teamwork in order to transform individuals as responsible leaders and entrepreneurs.
- To train the students to the changing technical scenario and make them to understand the importance of Sustainable and Inclusive technologies.

Vision & Mission of the Department

Vision:

To prepare the students for global competence, with core knowledge in Electronics and communication engineering having focus on research to meet the needs of industry and society.

Mission:

- To provide in-depth knowledge of Electronics and Communication Engineering, ensuring the effective teaching learning process.
- To train students to take-up innovative projects in group with emerging technology relevant to the industry and social needs.
- To imbibe professional ethics, research culture and development of skills.

Program Educational Objectives

Graduate students must be able to:

PEO1: Have Core knowledge of ECE with strong Teaching learning Process who are ready to apply the state-of-art technology to solve Industry and socially relevant problems.

PEO2: Engage in team with work technical knowledge and effective communication skills to address the practical issues in industry and society.

PEO3: Be technologists who can analyze and design innovative projects through research and emerging technology.

PEO4: Professionals, with capabilities for pursuing higher studies in technical/managerial courses.

Program Specific Outcomes

PSO1: Promote Electronics and Communication engineering in meeting technical, environmental and societal challenges.

PSO2: Integrate design and development of electronic systems and circuits using current practices and standards.

PSO3: Apply knowledge of Electronics and Communication Engineering in building projects, developing good communication skills, incorporating ethical behavior for project management, with financial prudence.

List of Experiments in IPCC Course

Integrated Lab
Simulation experiments using software tools like – Modelsim, Cadence

Expt. No.	Contents of the Experiments (2-3 expts from each module / unit)	Hrs	CO Level
Module 1			
1.	Write a Verilog code for modelling Flip-Flops using always block E,g:J.K flipflop,Tff,2:1 mux	2	1,2
2.	Write a Verilog code for modelling always blocks using Event Control Statements	2	1,2
3.	Write a Verilog code to show the usage of intra delay and inter delay assignment	2	1,2
Module 2			
4.	Write a Verilog code for single port SRAM using task	2	1,2,
5.	Write a Verilog code to show the usage of User defined primitives(UDP)	2	1,2
6.	Write a Verilog code to indicate the difference between blocking and non-blocking statements	2	1,2
Module 3			
7.	Execute the synthesis flow for the given counter design RTL	2	1,2,3
	a. Constrain the design with the clock frequency of 50 MHZ		
	b. Budget the input and output delay of inputs and output ports		
	c. Synthesize the design		
	d. Write out the gate level netlist and output constraints file(.sdc)		
	e. Generate the reports of timing power and area?		
Module 4			
8.	Write a Verilog Code for booth multiplier	2	1,2,3,4
9.	Write a Verilog code for asynchronous FIFO design	2	1,2,3,4
Module 5			
10.	Write Verilog code for SPI protocol	2	1,2,3,4
11.	Design RTL for APB protocol based memory (1kx32) and develop test bench to verify? (Open ended Experiment)	2	1,2,3,4

Integrated Laboratory Experiment Reference Text-Books :

1	Peter J. Ashenden, "Digital Design: An Embedded Systems Approach Using VERILOG", *Elesvier*, 2010.
2	chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://in.ncu.edu.tw/ncume_ee/digilogi/vhdl/Verilog_Reference_Guide.pdf
3	J. Bhaskar, "A Verilog HDL Primer", *BS Publications*, India.
4	Ming-Bo Lin, "Digital System designs and Practices using Verilog HDL and FPGAs", *John Wiley & Sons*, 2008

On-Line Materials & Resources (Video experiments / You-tube Videos / Virtual labs / Demo experiments videos, etc…) :

1	https://onlinecourses.nptel.ac.in/noc24_cs61/preview
2	https://archive.nptel.ac.in/courses/106/105/106105165/
3	https://www.youtube.com/watch?v=2JQ_rDLWr_g
4	https://www.slideshare.net/TonyLisko/advanced-digital-design-with-the-verilog-hdl-259648199
5	https://www.maven-silicon.com/best-vlsi-courses/online-vlsi-course

DO's

- Students should follow the dress code of the laboratory compulsorily.
- Keep your belongings in the corner of the laboratory.
- Students have to enter their name, USN, time in/out and signature in the log register maintained in the laboratory.
- Students are required to enter components in the components register related to the experiment and handle the equipment's smoothly.
- Check the components, range and polarities of the meters before connecting to the circuit.
- Come prepare for the experiment and background theory.
- Before connecting to the circuit refer the designed circuit diagram properly. Debug the circuit for proper output.
- Students should maintain discipline in the laboratory and keep the laboratory clean and tidy.
- Observation book and Record book should be complete in all respects and get it corrected by the staff members.
- Clarify the doubts with staff members and instructors.
- Experiment once conducted, in the next lab, the entire record should be complete in all respects, else the student will lose the marks.
- For programming lab, show the output to the concerned faculty.
- All the students should come to LAB on time with proper dress code and identity cards
- Keep your belongings in the corner of laboratory.
- Students have to enter their name, USN, time-in/out and signature in the log register maintained in the laboratory.
- All the students should submit their records before the commencement of Laboratory experiments.
- Students should come to the lab well prepared for the experiments which are to be performed in that particular session.
- Students are asked to do the experiments on their own and should not waste their precious time by talking, roaming and sitting idle in the labs.

- Observation book and record book should be complete in all respects and it should be corrected by the staff member.
- Before leaving the laboratory students should arrange their chairs and leave in orderly manner after completion of their scheduled time.
- Prior permission to be taken, if for some reasons, they cannot attend lab.
- Immediately report any sparks/ accidents/ injuries/ any other untoward incident to the faculty /instructor.
- In case of an emergency or accident, follow the safety procedure.
- Switch OFF the power supply after completion of experiment.

DONT's

- Do not switch on the power supply before verification of the connected circuits by concerned staff.
- Do not feed higher voltages than rated to the device.
- Do not upload, delete or alter any software on the laboratory PC's.
- Do not write or mark on the equipment's.
- Usage of mobile phone is strictly prohibited.
- Ragging is punishable.
- If student damages the equipment or any of the component in the lab, then he / she is solely responsible for replacing that entire amount of the equipment or else, replace the equipment.
- The use of mobile/ any other personal electronic gadgets is prohibited in the laboratory.
- Do not make noise in the Laboratory & do not sit on experiment table.
- Do not make loose connections and avoid overlapping of wires.
- Don't switch on power supply without prior permission from the concerned staff.
- Never point/touch the CRO/Monitor screen with the tip of the open.

Experiment No: 1 Date: _____

Flipflops

Aim: To write a Verilog code for modelling Flip-Flops using always block

1. JK flipflop
2. T flipflop
3. 2:1 mux

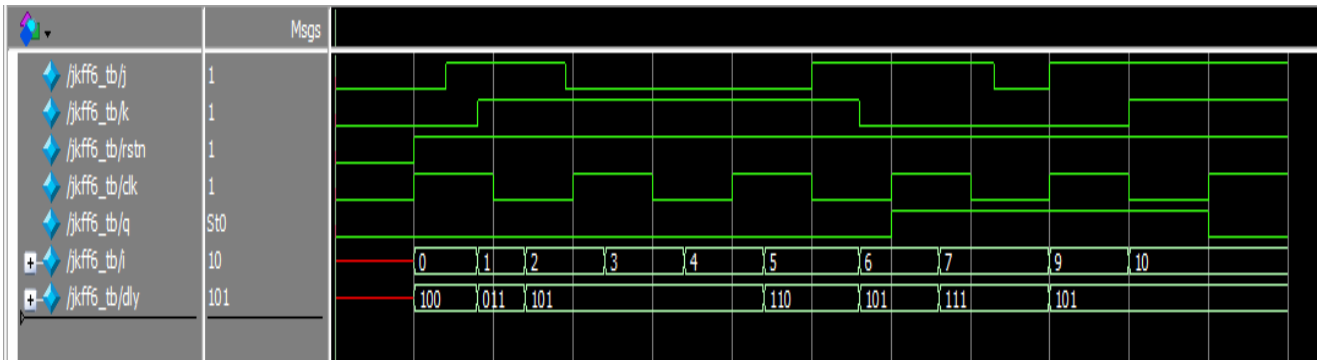
Verilog

```
module jk_ff (input j,input k, input rstn, input clk, output reg q);
always @ (posedge clk or negedge rstn)
begin
if(!rstn)
begin
q<=0;
end
else
begin
q <=(j&~q)|(~k&q);
end
end
endmodule
```

Testbench

```
module tb;
reg j,k,rstn,clk;
wire q;
integer i;
reg [2:0] dly;
always #10 clk=~clk;
jk_ff jk_instance ( .j(j),.k(k),.clk(clk),.rstn(rstn),.q(q));
initial
begin
{j,k,rstn,clk}<=0;
#10 rstn<= 1;
for(i=0;i<10;i=i+1)
begin
dly= $random;
#(dly)j<= $random;
#(dly) k<=$random;
end
#20 $finish;
end
endmodule
```

Results:



2. T flipflop

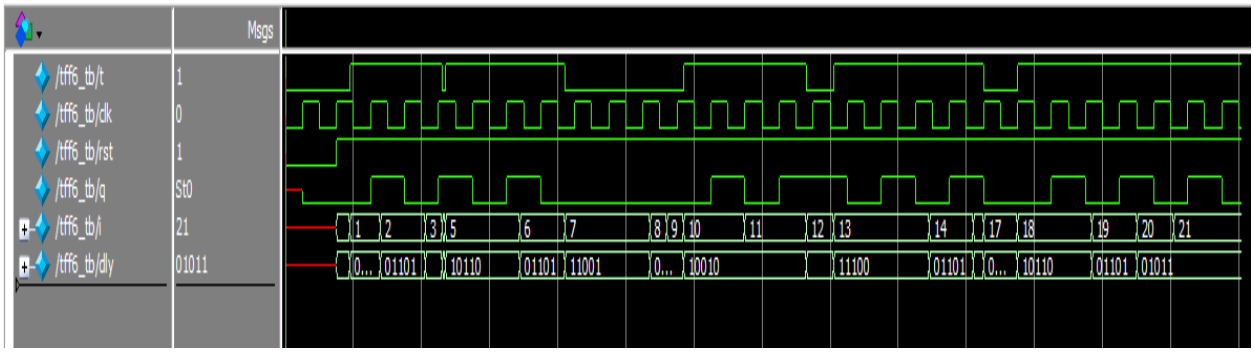
Verilog

```
module t_ff(t,clk,rst,q);
  input t,clk,rst;
  output reg q;
  always@(posedge clk or negedge rst)
   begin
   if(!rst)
    q<=1'b0;
   else
    begin
    if(t==1)
    q <=~q;
    if(t==0)
     q <=q;
    end
end
endmodule
```

Testbench

```
module t_ff_tb;
  reg t,clk,rst;
  wire q;
  integer i;
  reg [4:0]dly;
  t_ff uut(.clk(clk),.rst(rst),.t(t),.q(q));
  always #5 clk = ~clk;
  initial begin
   {rst,clk,t} <= 0;
   $monitor("T=%0t,rst = %0b,t=%0d,q=%0d",$time,rst,t,q);
   repeat(2)@(posedge clk);
   rst <= 1;
   for(i=0;i<=20;i=i+1)
   begin
    dly = $random;
    #(dly)t<=$random;
   end
  #20 $finish;
  end
endmodule
```

Results:



3. 2:1 mux

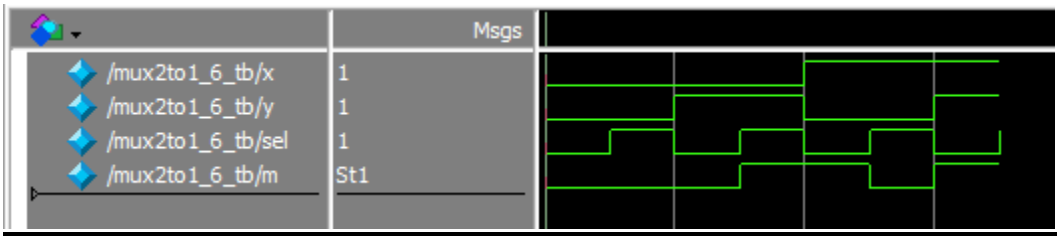
Verilog

```
module muxbehv(x,y,sel,m);
   input x,y,sel;
  output reg m;
  always@(x or y or sel)
   begin
   m =0;
   if(sel==0)
   begin
    m = x;
   end
   else
    begin
    m = y;
   end
  end
endmodule
```

Testbench

```
module muxbehav_tb;
reg x,y,sel;
wire m;
muxbehav uut(x,y,sel,m);
initial begin
x = 1'b0; y=1'b0;sel=1'b0;
 #10 x=1'b0;y=1'b0;sel=1'b1;
#10 x=1'b0;y=1'b1;sel=1'b0;
#10 x=1'b0;y=1'b1;sel=1'b1;
#10 x=1'b1;y=1'b0;sel=1'b0;
#10 x=1'b1;y=1'b0;sel=1'b1;
#10 x=1'b1;y=1'b1;sel=1'b0;
#10 x=1'b1;y=1'b1;sel=1'b1;
$finish;
end
endmodule
```

Result:



Application

1. **Flip flop**
    - Counters
    - Shift register
    - Frequency divider
    - Control Circuits
    - Toggle switches
2. **MUX**
    - Data selection
    - Parallel to series conversion
    - Function generation
    - Logic function implementation
    - Signal switching

Remarks:

Signature of Staff Incharge with date:

Experiment No: 2 Date: _____

Intra delay and Inter delay

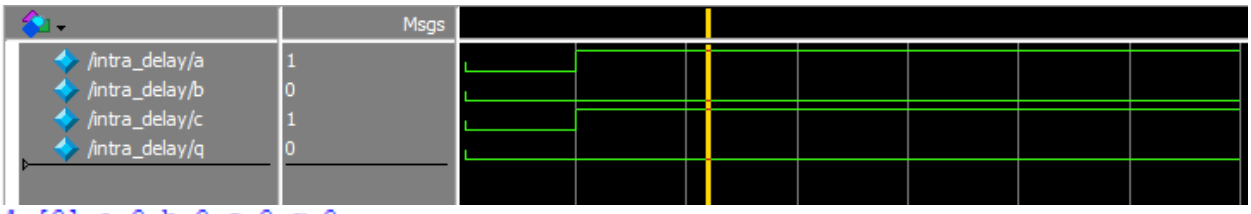
Aim: To write a Verilog code to show the usage of intra delay and inter delay assignment

I. Intra delay

Verilog

```
module intra_delay;
   reg a,b,c,q;
   initial begin
      $monitor("[%0t] a=%0b b=%0b c=%0b q=%0b",$time,a,b,c,q);
      a<=0;b<=0;c<=0;q<=0;
      #5 a<=1;c<=1;
      q<=#5 a&b|c;
      #30; $finish;
   end
endmodule
```

Results:



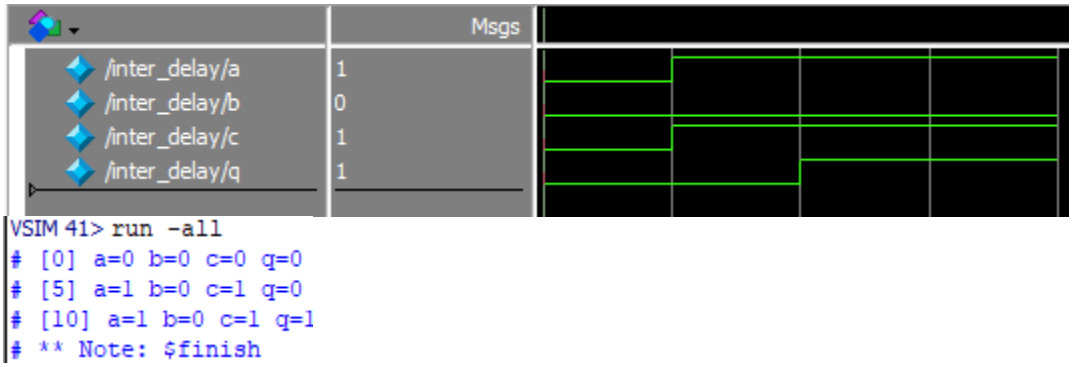

```
# [0] a=0 b=0 c=0 q=0
# [5] a=1 b=0 c=1 q=0
# ** Note: $finish
```

II. Inter delay

Verilog

```
module inter_delay;
   reg a,b,c,q;
   initial begin
      $monitor("[%0t] a=%0b b=%0b c=%0b q=%0b",$time,a,b,c,q);
      a<=0;b<=0;c<=0;q<=0;
      #5 a<=1;c<=1;
      #5 q<=a&b|c;
      #10; $finish;
   end
endmodule
```

Results:



Application

- Modelling propagation delay
- Test stimulus generation
- Behavioral modelling
- Clock generation
- Delay chain for event sequence

Remarks:

Signature of Staff Incharge with date

Experiment No: 3 Date: _____

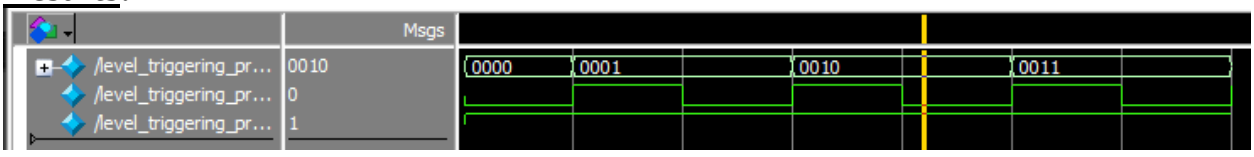
Event Controlled Statement

Aim: To write a Verilog code for modelling always blocks using Event Control Statements

Verilog

```
module tb;
   reg [3:0] ctr;
   reg clk;
   reg rst;
   always #10 clk=~clk;
   always begin
      rst =1;
      @(posedge clk)
      ctr <=ctr + 1;
   end
   initial begin
      {ctr,clk}<=0;
      wait (ctr);
      $display("T= %0t Counter reached non zero value 0x%0h",$time,ctr);
      wait(ctr==4);
      $display("T=%0t counter reached 0x%0h",$time,ctr);
      $finish;
   end
endmodule
```

Results:




```
# T= 10 Counter reached non zero value 0x1
# T=70 counter reached 0x4
```

Application

- Clock driven sequential logic
- State machines
- Edge detection
- Testbenches
- Combinational logic with sensitivity list

Remarks :

Signature of Staff Incharge with date:

Experiment No: 4 Date: _____

Single port SRAM

Aim: To write a Verilog code for single port SRAM using task.

Verilog

```
module memory(addr, cs, re, we, clk, reset, data_in, data_out);
   parameter addr_width = 16;
   parameter data_width = 32;

   input [addr_width-1:0] addr;
   input cs, re, we;
   input clk, reset;
   input [data_width-1:0] data_in;
   output [data_width-1:0] data_out;

   reg [data_width-1:0] data_out;
   reg [data_width-1:0] mem[0:2**addr_width-1];
   integer i;

   always @(posedge clk) begin
      if (reset) begin
         mem[addr] <= 0;
      end else begin
         if (cs) begin
            if (we) begin
               mem[addr] <= data_in;
            end
            if (re) begin
```

```
                data_out <= mem[addr];
            end
        end
    end
endmodule
```

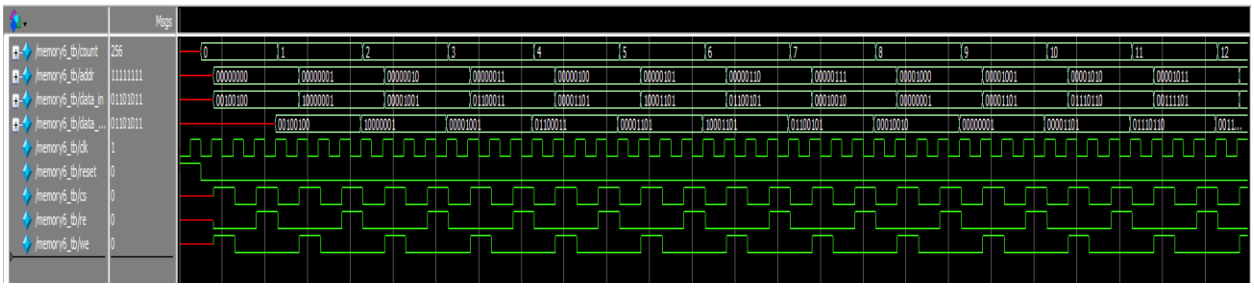
Testbench

```
module memory_tb();
    parameter addr_width = 8;
    parameter data_width = 8;
    integer count;
    reg [addr_width-1:0] addr;
    reg [data_width-1:0] data_in;
    wire [data_width-1:0] data_out;
    reg clk, reset, cs, re, we;

    memory #(addr_width, data_width) memory_inst(addr, cs, re, we, clk, reset, data_in,
data_out);
    always #5 clk = ~clk;

    initial begin
        clk = 0;
        reset = 1;
        #10 reset = 0;
        for (count = 0; count < 'h100; count = count + 1) begin
            write_task(count);
            read_task(count);
            if (data_in != data_out)
                $display($time, "FAIL: Data Miss Match for addr=%0h, data_in=%0h,
data_out=%0h\n", count, data_in, data_out);
```

```
else
            $display($time, "PASS: Data Match for addr=%0h, data_in=%0h, data_out=%0h\n",
count, data_in, data_out);
        end
        $finish;
    end
    task write_task;
        input [addr_width-1:0] addr_in;
        begin
            @(posedge clk);
            #1;
            cs = 1; we = 1; re = 0;
            data_in = $random;
            addr = addr_in;
            @(posedge clk);
            #1;
            we = 0; cs = 0;
        end
    endtask
    task read_task;
        input [addr_width-1:0] addr_in;
        begin
            @(posedge clk);
            #1;
            cs = 1; we = 0; re = 1;
            addr = addr_in;
            @(posedge clk);
            #1;
            re = 0; cs = 0;
        end
    endtask
```

endmodule

Results:



```
VSIM 4> run -all
#                      46PASS: Data Match for addr=0, data_in=24, data_out=24
#
#                      86PASS: Data Match for addr=1, data_in=81, data_out=81
#
#                     126PASS: Data Match for addr=2, data_in=9, data_out=9
#
#                     166PASS: Data Match for addr=3, data_in=63, data_out=63
#
#                     206PASS: Data Match for addr=4, data_in=d, data_out=d
#
#                     246PASS: Data Match for addr=5, data_in=8d, data_out=8d
#
#                     286PASS: Data Match for addr=6, data_in=65, data_out=65
#
#                     326PASS: Data Match for addr=7, data_in=12, data_out=12
#
```

Application:

- **Image and video processing**
- **Signal processing**
- **Embedded systems**

Remarks

Signature of Staff Incharge with date

Experiment No:5 Date: _____

User Defined Primitives (UDP)

Aim: To write a Verilog code to show the usage of User defined primitives (UDP)

1. 2:1 Mux
2. D latch
3. D flipflop

1. 2:1 MUX

Verilog

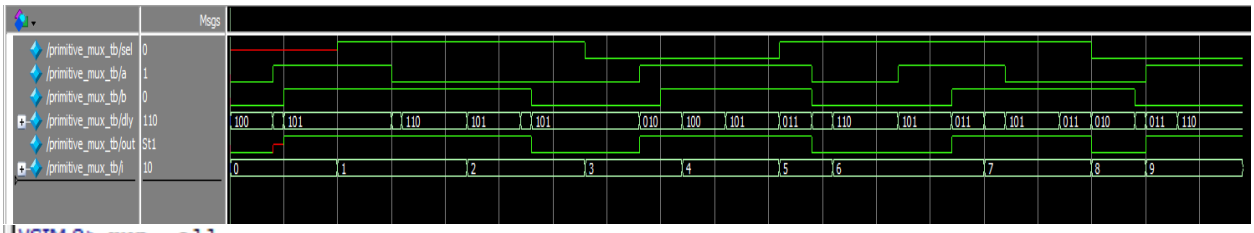
```
primitive mux(out,sel,a,b);
output out;
input sel,a,b;

table
0 1 ? : 1;
0 0 ? : 0;
1 ? 0 : 0;
1 ? 1 : 1;
x 0 0 : 0;
x 1 1 : 1;
endtable
endprimitive
```

Test Bench

```
module mux_tb;
reg sel,a,b;
reg [2:0]dly;
wire out;
integer i;
mux u_mux(out,sel,a,b);
```

```
initial begin
a<=0;
b<=0;
$monitor("[T=%0t] a=%0b b=%0b sel=%0b out=%0b",$time,a,b,sel,out);
for(i=0; i <10 ;i=i+1) begin
dly=$random;
#(dly)a<=$random;
dly=$random;
#(dly)b<=$random;
dly=$random;
#(dly)sel<=$random;
end
end
endmodule
```

Results:




```
VSIM 8> run -all
# [T=0] a=0 b=0 sel=x out=0
# [T=4] a=1 b=0 sel=x out=x
# [T=5] a=1 b=1 sel=x out=1
# [T=10] a=1 b=1 sel=1 out=1
# [T=15] a=0 b=1 sel=1 out=1
# [T=28] a=0 b=0 sel=1 out=0
# [T=33] a=0 b=0 sel=0 out=0
# [T=38] a=1 b=0 sel=0 out=1
# [T=40] a=1 b=1 sel=0 out=1
# [T=51] a=1 b=1 sel=1 out=1
# [T=54] a=0 b=0 sel=1 out=0
# [T=62] a=1 b=0 sel=1 out=0
# [T=67] a=1 b=1 sel=1 out=1
# [T=72] a=0 b=1 sel=1 out=1
# [T=80] a=0 b=1 sel=0 out=0
# [T=84] a=0 b=0 sel=0 out=0
# [T=85] a=1 b=0 sel=0 out=1
```

2. D latch

Verilog:

```
primitive D_latch(q,clk,d);
output q;
input clk,d;
reg q;
table
// clk d q q+
11 : ? : 1;
10 : ? : 0;
0? : ? : -;
endtable
endprimitive
```

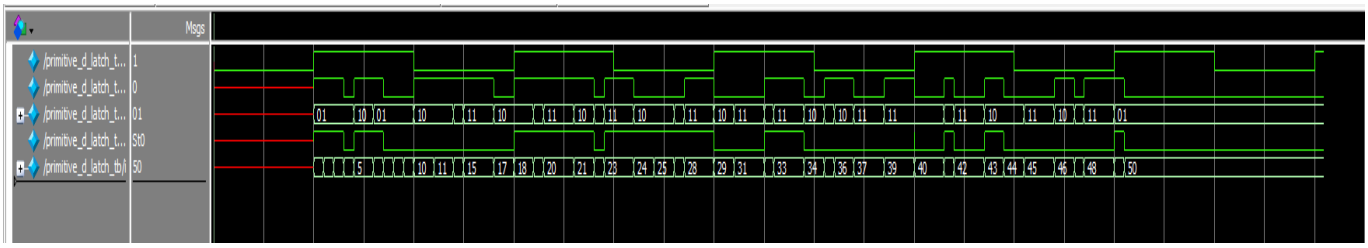
Test Bench

```
module Dlatch_tb1;
reg clk,d;
reg [1:0]dly;
wire q;
integer i;
D_latch u_latch(q,clk,d);
always #10 clk=~clk;
initial begin
clk=0;
$monitor ("[T=%0t] clk=%0b d=%0b q=%0b",$time,clk,d,q);
#10;
for(i=0;i<50;i=i+1)begin
dly=$random;
#(dly)d<=$random;
end
#20 $finish;
```

end

endmodule

Results:




```
VSIM 13> run -all
# [T=0] clk=0 d=x q=x
# [T=10] clk=1 d=1 q=1
# [T=13] clk=1 d=0 q=0
# [T=14] clk=1 d=1 q=1
# [T=17] clk=1 d=0 q=0
# [T=20] clk=0 d=1 q=0
# [T=28] clk=0 d=0 q=0
# [T=30] clk=1 d=1 q=1
# [T=38] clk=1 d=0 q=0
# [T=39] clk=1 d=1 q=1
# [T=40] clk=0 d=1 q=1
# [T=42] clk=0 d=0 q=1
# [T=47] clk=0 d=1 q=1
# [T=50] clk=1 d=0 q=0
# [T=55] clk=1 d=1 q=1
# [T=59] clk=1 d=0 q=0
# [T=60] clk=0 d=0 q=0
# [T=61] clk=0 d=1 q=0
# [T=64] clk=0 d=0 q=0
# [T=67] clk=0 d=1 q=0
# [T=70] clk=1 d=0 q=0
# [T=73] clk=1 d=1 q=1
# [T=74] clk=1 d=0 q=0
# [T=77] clk=1 d=1 q=1
# [T=79] clk=1 d=0 q=0
# [T=80] clk=0 d=0 q=0
# [T=84] clk=0 d=1 q=0
# [T=86] clk=0 d=0 q=0
# [T=87] clk=0 d=1 q=0
# [T=90] clk=1 d=1 q=1
# [T=91] clk=1 d=0 q=0
# [T=100] clk=0 d=0 q=0
# [T=110] clk=1 d=0 q=0
# ** Note: $finish    :
```

3. D flipflop

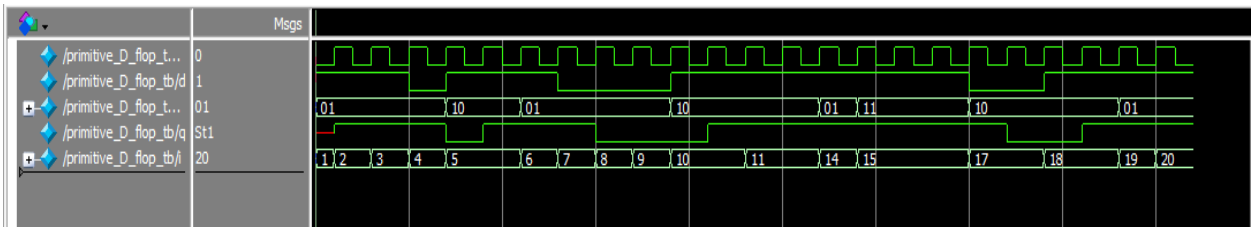
```
primitive D_flop(q,clk,d);
output q;
input clk,d;
reg q;
table
//clk d q q+
//obtain output on rising edeg of clk
(01)0 : ? : 0;
(01)1 : ? : 1;
(0?)1 : 1 : 1;
(0?)0 : 0 : 0;
(?0)? : ? : -;
?(??) : ? : -;
endtable
endprimitive
```

Test bench

```
module D_flop_tb;
reg clk,d;
reg [1:0]dly;
wire q;
integer i;
D_flop u_flop(q,clk,d);
always #10 clk=~clk;
initial begin
clk=0;
$monitor("[T=%0t] clk=%0b d=%0b q=%0b" ,$time,clk,d,q);
for(i=0;i<20;i=i+1)begin
dly=$random;
repeat(dly)@(posedge clk);
```

```
d<=$random;
end
#20;
$finish;
end
endmodule
```

Results:




```
# [T=0] clk=0 d=1 q=x
# [T=10] clk=1 d=1 q=1
# [T=20] clk=0 d=1 q=1
# [T=30] clk=1 d=1 q=1
# [T=40] clk=0 d=1 q=1
# [T=50] clk=1 d=0 q=1
# [T=60] clk=0 d=0 q=1
# [T=70] clk=1 d=1 q=0
# [T=80] clk=0 d=1 q=0
# [T=90] clk=1 d=1 q=1
# [T=100] clk=0 d=1 q=1
```

Applications:

- Counters
- Shift register
- Frequency divider
- Control Circuits
- Data selection
- Logic function implementation

Remarks:

Signature of Staff Incharge with date:

Experiment No: 6 Date: _____

Blocking and non-blocking statements

Aim: To write a Verilog code to indicate the difference between blocking and non-blocking statements.

Verilog

```
module blocking_nonblocking;
   reg[0:7] A,B,C;
   initial begin
      A=3;
      A=A+1;
      B=A+1;
      $display($time,"Blocking: A=%h B=%h",A,B);
      A=8'h3;B=8'h0;
      A<=A+'h1;
      B<=A+1;
      $display($time,"Non-blocking : A=%h B=%h",A,B);
      #1;
      $display($time,"End of time unit: A=%h B=%h",A,B);
   end
endmodule
```

Results:

```
0Blocking: A=04 B=05
0Non-blocking : A=03 B=00
1End of time unit: A=04 B=04
```

Application:

- Combinational logic
- Procedural testbench
- Sequential logic
- State machine
- Pipeline register

Remarks

Signature of Staff Incharge with date

Experiment No: 6 Date: _____

Synthesis flow

Aim: For the given counter design RTL

i) To Constrain the design with the clock frequency of 50 MHZ
ii) Budget the input and output delay of inputs and output ports
iii) Synthesize the design
iv) Write out the gate level netlist and output constraints file(.sdc)
v) Generate the reports of timing power and area?

Conter RTL :

```
module counter(clk,reset,up_down,load,data,count);
 //define input and output ports
 input clk,reset,load,up_down;
 input [3:0] data;
 output reg [3:0] count;
 //always block will be executed at each and every positive edge of the clock
 always@(posedge clk)
 begin
  if(reset)    //Set Counter to Zero
    count <= 0;
  else if(load)    //load the counter with data value
    count <= data;
  else if(up_down)       //count up
    count <= count + 1;
  else          //count down
    count <= count - 1;
 end
endmodule :counter
```

For design with the clock frequency of 50 MHZ
Period taken is 20ns

Synopsis Design Constraints:

Create_ clock -period 20 -name clk [get_ports PCLK]
set_input_delay -clock clk 8 [all_inputs]
set_input_delay -clock clk 8 [all_inputs]
set_output_delay -clock clk 8 [all_outputs]
set_output_delay -clock clk 8 [all_outputs]

Outputs:

Steps

i) Synthesize the design by invoking the genus tool
ii) Write out the gate level netlist and output constraints file(.sdc)
iii) Generate the reports of timing power and area?

Remarks:

Experiment No: 8 Date: _____

Booth Multiplier

Aim: To write a Verilog Code for booth multiplier

Verilog

```
module BoothMulti(X, Y, Z);
     input signed [3:0] X, Y;
      output reg signed[7:0] Z;
      reg [1:0] temp;
      integer i;
      reg E1;
      reg [3:0] Y1;
      always @ (X, Y)
      begin
      Z = 8'd0;
      E1 = 1'd0;
      for (i = 0; i < 4; i = i + 1)
      begin
      temp = {X[i], E1};
      Y1 = - Y;
      case (temp)
      2'd2 : Z [7 : 4] = Z [7 : 4] + Y1;
      2'd1 : Z [7 : 4] = Z [7 : 4] + Y;
      default : begin end
      endcase
      Z = Z >> 1;
      Z[7] = Z[6];
      E1 = X[i];
         end
      if (Y == 4'd8)
         begin
            Z = - Z;
         end
      end
endmodule
```

```
module BoothTB;
```

```
    reg [3:0] X;
    reg [3:0] Y;
    wire [7:0] Z;
    BoothMulti uut (.X(X),.Y(Y),.Z(Z));
    initial begin
       X=4'b0110;Y=4'b0101;#10
  X=4'b0010;Y=4'b1001;#10
  X=4'b0110;Y=4'b0100;#10
  X=4'b1110;Y=4'b0110;#10
  X=4'b1100;Y=4'b1110;#10
$finish;
    end
endmodule
```

Application:

- Hardware implementations of multiplication operations
- Digital signal processors
- Microprocessors
- FPGAs

Remarks

Signature of Staff Incharge with date

Experiment No: 9 Date: _____

Asynchronous FIFO design

Aim: To write a Verilog Code for Asynchronous FIFO design

Verilog code

```
module async_fifo(
input wclk,rclk,reset,
input w_enable,r_enable,
input [7:0]wdata,
output reg[7:0]rdata,
output full_flag,empty_flag);
parameter FIFO_DEPTH=16, POINTER=4;
reg[7:0]fifo[0:FIFO_DEPTH-1];
reg[POINTER:0]wpointer,rpointer,wpointer_sync,rpointer_sync,Qr,Qw;

assign full_flag = (({~wpointer[POINTER],wpointer[POINTER-
1:0]}==rpointer_sync[POINTER:0]))?1:0;
assign empty_flag = (wpointer_sync == rpointer)?1:0;

//write
always@(posedge wclk,posedge reset)
begin
        if(reset==1)
        begin
                wpointer <= 0;
        end
        else
        begin
        if(w_enable==1 && full_flag != 1)
        begin
                fifo[wpointer[POINTER-1:0]] <= wdata;
                wpointer <= wpointer+1;
        end
        end
end

//read

always@(posedge rclk,posedge reset)
begin
```

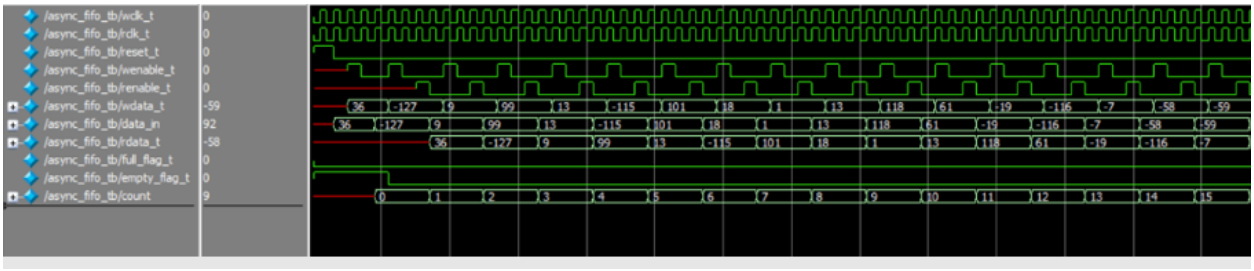
```
        if(reset==1)
        begin
                rpointer <= 0;
        end
        else
        begin
        if(r_enable == 1 && empty_flag != 1)
        begin
                rdata <= fifo[rpointer[POINTER-1:0]];
                rpointer <= rpointer+1;
        end
        end
end

//wr_ptr Synchronization for read side

always@(posedge rclk,posedge reset)
begin
        if(reset==1)
        begin
                Qw <= 0;
                wpointer_sync <= 0;
        end
        else
        begin
                Qw <= wpointer;
                wpointer_sync <= Qw;
        end
end

//rd_ptr Synchronization


always@(posedge wclk,posedge reset)
begin
        if(reset==1)
        begin
                Qr <= 0;
                rpointer_sync <= 0;
        end
        else
        begin
                Qr <= rpointer;
                rpointer_sync <= Qr;
        end
        end
```

```
endmodule


///Testbench///
module async_fifo_tb;
parameter FIFO_DEPTH=16,POINTER=4;
reg wclk_t,rclk_t,reset_t;
reg wenable_t,renable_t;
reg[7:0]wdata_t,data_in;
wire[7:0]rdata_t;
wire full_flag_t,empty_flag_t;
integer count;

async_fifo
#(FIFO_DEPTH,POINTER)async_inst(wclk_t,rclk_t,reset_t,wenable_t,renable_t,wdata_t,rdata_
t,full_flag_t,empty_flag_t);


//write task
task write_task;
input[7:0]data_in;
begin
        @(posedge wclk_t);
        if(!full_flag_t)
        begin
        wenable_t = 1;
        wdata_t = data_in;
        @(posedge wclk_t);
        wenable_t = 0;
        end
end
endtask

//read task

task read_task;
begin
        @(posedge rclk_t);
        if(!empty_flag_t)
        begin
        renable_t = 1;
        @(posedge rclk_t);
        renable_t = 0;
        end
end
endtask
```

```
//single_wr_rd_test

task single_wr_rd_test();
begin

data_in = $random;
write_task(data_in);
read_task();
end
endtask

//all_wr_rd_test
task all_wr_rd_test();
for(count=0;count < FIFO_DEPTH; count=count+1)
begin
        data_in = $random;
        write_task(data_in);
        read_task();
end
endtask

//fifo full and empty case
task fifo_full_empty_test();
begin
for(count =0;count < FIFO_DEPTH+1;count=count+1)
begin
        data_in = $random;
        write_task(data_in);
end
for(count=0; count < FIFO_DEPTH+1;count=count+1)
begin
        read_task();
end
end
endtask

//wclk clock gen
always
begin
#5 wclk_t = ~wclk_t;

end


//rclk clock gen
```

```
always
begin
#5 rclk_t = ~rclk_t;
end

//reset and calling the task
initial
begin
        wclk_t = 0;
        rclk_t = 0;
        reset_t = 1;
        #15
        reset_t = 0;
        @(posedge wclk_t);
        single_wr_rd_test();
        all_wr_rd_test();
        fifo_full_empty_test();
        #100 $finish;
end
endmodule
```

Waveforms:



Applications

To synchronize data flow between two systems working on different clocks

Remarks:

Experiment No: 9 Date: _____

Serial Peripheral Interface

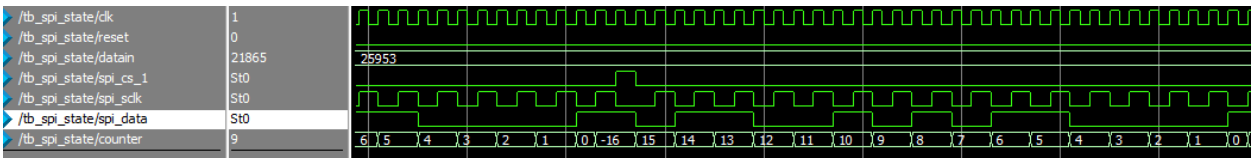
Aim: To write a Verilog Code for SPI Protocol

```
module spi_state(

  input wire clk,
  input wire reset,

  input wire [15:0] datain,
  output wire spi_cs_1,
  output wire spi_sclk,
  output wire spi_data,
  output [4:0] counter
 );

  reg [15:0] MOSI;
  reg [4:0] count;
  reg cs_1;
  reg sclk;
  reg [2:0] state;

  always@(posedge clk or posedge reset)
    if (reset) begin
       MOSI <=16'b0;
       count <=5'd16;
       cs_1 <=1'b1;
       sclk <=1'b0;
    end

  else begin
  case (state)

  0:begin
  sclk <=1'b0;
    cs_1 <=1'b1;
    state<=1;
  end


    1:begin
      sclk <=1'b0;
      cs_1 <=1'b0;
```

```
     MOSI<=datain[count-1];
     count <=count-1;
     state<=2;
    end

    2:begin
      sclk <= 1'b1;
      if (count > 0)
        state<=1;
        else begin
          count<=16;
         state<=0;
        end
     end


  default:state<=0;
        endcase
 end

  assign spi_cs_1 = cs_1;
  assign spi_sclk = sclk;
  assign spi_data = MOSI;
  assign counter=count;
endmodule

Test bench;
module tb_spi_state;
  reg clk;
  reg reset;
  reg [15:0]datain;


  wire spi_cs_1;
  wire spi_sclk;
  wire spi_data;
  wire [4:0]counter;

  spi_state dut (
    .clk(clk),
    .reset(reset),
    .counter(counter),
    .datain(datain),
    .spi_cs_1(spi_cs_1),
    .spi_sclk(spi_sclk),
    .spi_data(spi_data)
```

```
  );

  initial begin
    clk=0;
    reset =1;
    datain=0;
  end
  always #5 clk=~clk;

  initial begin
    #10 reset=1'b0;

    #10 datain=16'h5569;
    #335 datain=16'h2563;
    #335 datain=16'h9863;
    #335 datain=16'h6561;

  end
endmodule
```

Waveforms



Applications:

Remarks:

Experiment No: 11 Date: _____

Open Ended Experiment

Aim: Design RTL for APB protocol based memory (1kx32) and develop test bench to verify?

ADD LABORATORY (21EC62)

PROBABLE/SUGGESTED QUESTION BANK

1. Expand VHDL.What is the difference between VHDL and Verilog?
2. What is the difference between (i) signal and variable (ii) generic & Parameter (iii) function & procedure (iv) task & function (v) always & initial (vi) register & variable (vii) signal & wire.
3. What are the different styles of models in VHDL and Verilog?
4. What are the operators in VHDL & Verilog?
5. Which operator is having most priority?
6. What is meant by sensitivity list?
7. Give the Following syntax in HDL (i) if, for, function, procedure (ii) while, case.
8. What is the operating frequency of your FPGA?
9. Expand FPGA & ASIC.
10. What are the data types in VHDL?
11. What are the data types in Verilog? What is delta time?
12. What is the difference between (0 to 3) & (3 downto 0)?
13. Write the truth table & Excitation table for D filp flop, SR , T, JK
14. What are the file operations in Verilog?
15. What is meant by synthesis?
16. Write the flow chart for synthesis process?
17. What is the difference between combination circuit & sequential circuit?
18. What is the difference between latch & Flip flop?Write a Verilog code to swap contents of two registers with and without a temporary register?
19. In a pure combinational circuit is it necessary to mention all the inputs in sensitivity list? If yes, why?What is the difference between wire and reg?
20. Give only two xor gates one must function as buffer and another as not gate?
21. Build a 4:1 mux using only 2:1 mux? What are shift operators in HDL?
22. What are the logical operators in VHDL & Verilog?
23. What is the gate density of your FPGA?
24. What is data flow model, structural model, behavioral model? How you invoke from VHDL to Verilog and vice versa?

25. What is the difference between SR flip flop & JK flip flop?
26. What is the difference between synchronous reset & Asynchronous reset?
27. What is the difference between stepper motor & DC motor?
28. What is the step size of stepper motor?
29. What is mux and demux?
30. What is encoder and decoder?
31. What is the difference between encoder & priority encoder?
32. What is binding?
33. What is the difference between "bit" and "std_logic"?
34. What are the std_logic values?
35. What are the different types of buffers are in Verilog HDL?
36. What is the difference between dc motor and stepper motor?
37. What are the applications of dc motor and stepper motor?
38. Write the syntax for casex and casez. What is screen time?
39. What are the VHDL file processing?
40. Draw the simulation waveform for D-latch using signal assignment and variable assignment statements inside the process.
41. Give the syntax for arrays in VHDL and Verilog.

References:

1. http://ece.niu.edu.tw/~chu/download/fpga/verilog.pdf
2. http://www.ics.uci.edu/~alexv/154/VHDL-Cookbook.pdf
3. http://access.ee.ntu.edu.tw/course/dsd_99second/2011_lecture/W2_HDL_Fundamentals_2011-03-02.pdf
4. Stephen Brown and Zvonko Vranesic, "Fundamentals of Digital Logic Design with VHDL", Second Edition, The McGraw-Hill, 2009.
5. Nazeih M.Botros, "HDL Programming (VHDL and Verilog)", John Wiley India Pvt. Ltd. 2008.
6. Samir Palnitkar, "Verilog HDL-A guide to digital design and synthesis", Sunsoft Press, 1996
7. Charles H Roth Jr., "Digital System Design using VHDL", PWS Publishing Company.