**Department of Electronics and Telecommunication Engineering**

**VLSI Design**

**Lab Manual**

**Vision, Mission and PEOs of E&TC Department**

**Department Vision:**

To be a leading source of Electronics and Telecommunication of globally competent engineers that meets the needs of evolving industry and society.

**Department Mission:**

* Facilitate learning of cutting-edge technologies in association with industry and contribute towards the latest knowledge generation for better employability and sustainability.
* Promote innovation, research and development enabling higher education, entrepreneurship, and lifelong learning.
* Provide a platform for harnessing leadership qualities imbibed with social and ethical values.

**Department Program Educational Objectives:**

* **PEO1:** Graduates will have a strong understanding of the basics of science and engineering as well as analytical skills to solve real world problems.
* **PEO2:** Graduates will gain technical proficiency in Electronics and Telecommunication field and scale new heights in profession through lifelong learning.
* **PEO3:** Graduates will embrace professional and ethical manners at all the levels and constantly evolve in multidisciplinary approach leading towards sustainability.
* **PEO4:** Graduates will utilize engineering acumen with communication skills, leadership qualities and spirit of team work to exhibit qualities for imparting services to society.

**List of Experiment**

|  |  |
| --- | --- |
| **S. No.** | **Name of Experiment** |
| 1. | Implementation of all gates using Gate flow modelling in Verilog. |
| 2. | Implementation of half and full adder using Data flow and Gate level modelling in Verilog. |
| 3. | Implementation of Mux using Data flow, Gate level and Behavioral modelling styles in Verilog. |
| 4. | Implementation of Decoder in Verilog. |
| 5. | Implementation of full adder using instances of half adder in Verilog. |
| 6. | Implementation of Up down counter and Mod 30 counter using Test bench in Verilog. |
| 7. | Implementation of D and T flip flops using Test bench in Verilog. |
| 8. | Implementation of JK and SR flip flops using Test bench in Verilog. |
| 9. | Implementation of shift registers using Test bench in Verilog. |
| 10. | Implementation of 4 bit Ripple Carry Counter using instances of T & D Flip Flops. |
| 11. | Implementation of Moore machine using Test bench in Verilog. |
| 12. | Implementation of Mealy machine using Test bench in Verilog. |
| 13. | Implementation of ALU using Test bench in Verilog. |

**Experiment No. 1**

**Aim:** Implementation of all gates using Gate flow modelling in Verilog.

**Software Tool:** Vivado Design Suite

**Truth Table:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **AND** | **NAND** | **OR** | **NOR** | **XOR** | **XNOR** |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

module allgates(

input a, b,

output nd, nnd, r, nr, xr, nxr, nt

);

and (nd, a, b);

nand (nnd, a, b);

or (r, a, b);

nor (nr, a, b);

xor (xr, a, b);

xnor (nxr, a, b);

not (nt, a);

endmodule

Result

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematic.

Conclusion:

**Questions for Reflection:**

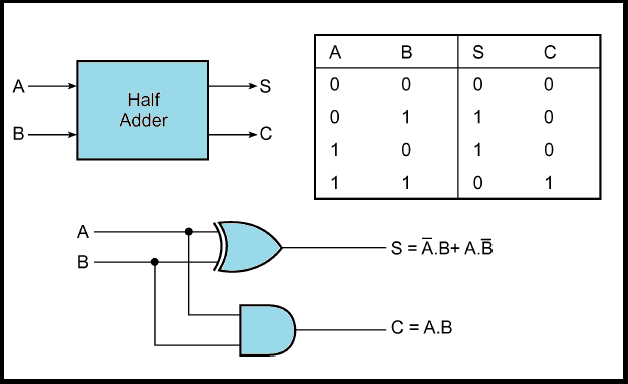
1. What do you mean by HDL?
2. What are different modelling styles in Verilog?
3. What is the syntax of gate level modelling?
4. What is the structure of Verilog coding?
5. What do you mean by concurrent and sequential programming?
6. Can you write only y output for all gates? Comment on the error if output is denoted by y for all gates.
7. Is there any effect on resource utilization if we change FPGA device for Synthesis?

**Experiment No. 2**

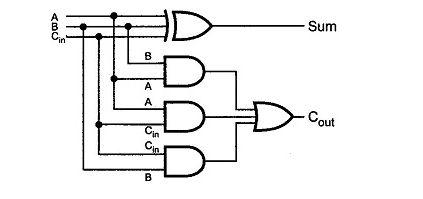
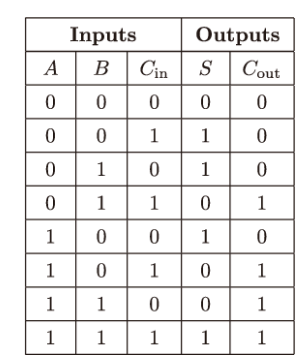
**Aim:** **Implementation of half and full adder using gate level and data flow modeling in Verilog**

**Apparatus:** Xilinx ISE Design Suite

**Theory:** Truth table of half adder and full adder



Full Adder

**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

## HALF ADDER

|  |  |
| --- | --- |
| Module ha\_glm (input a, input b, output sum, output c\_out );  xor x1(sum,a,b); and n1(c\_out,a,b);  endmodule | module ha\_dflm(input a,  input b, output s, output c);  assign s=a^b; assign c=a&b;  endmodule |

Gate level modeling Dataflow modeling

## FULL ADDER

|  |  |
| --- | --- |
| module fa\_glm ( input x,  input y, input cin, output A, output cout );  wire p,r,s; xor(p,x,y);  xor (A,p,cin);  and(r,p,cin);  and(s,x,y);  or(cout,r,s);  endmodule | module fsd\_dlm ( input x,  input y, input cin, output A, output cout);  assign A=(x^y)^cin;  assign cout =((x^y)&cin)|(x&y); endmodule |

Gate level modeling Dataflow modeling

**Result**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematic.

**Conclusion:**

**Questions for Reflection:**

1. What is the difference between data flow and gate level modelling?
2. What do you mean reg, wire and net?
3. What is the requirement for gate level modeling and data flow modeling?
4. What do you mean by assign statement in data flow modeling?
5. What are different data flow operators?
6. Can you assign FPGA pins to wire signal?
7. Is there any difference between simulation result of data flow and gate level modeling of Half and full adder?
8. Is there any difference between synthesis result of data flow and gate level modeling of Half and full adder?

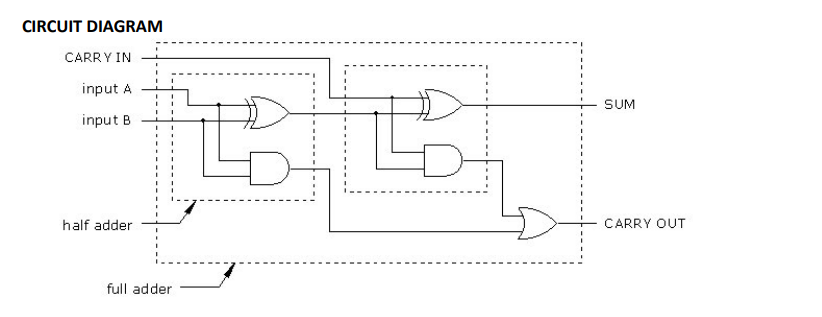
**Experiment 3**

**Aim: Implementation of full adder using 2 instances of half adder.**

**Apparatus:** Xilinx Vivado Design Suite

**Theory:** Truth table of full adder

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **Y** | **Z** | **SUM** | **CARRY** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |



1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

**Using Positional Mapping**

module FA\_using\_HA(

input x, y, z,

output s, c

);

wire s1, c1, c2;

HA A1(x, y, s1, c1);

HA A2(s1, z, s, c2);

or (c, c1, c2);

endmodule

module HA(

input a, b,

output s, c

);

xor (s, a, b);

and (c, a , b);

endmodule

**Using name mapping**

module FA\_using\_HA(

input x, y, z,

output s, c

);

wire s1, c1, c2;

HA\_dot A1(.a(x), .b(y), .s(s1), .c(c1));

HA\_dot A2(.a(s1), .b(z), .s(s), .c(c2));

or (c, c1, c2);

endmodule

module HA\_dot(

input a, b,

output s, c

);

xor (s, a, b);

and (c, a , b);

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematic of the design.

**Conclusion:**

**Questions for Reflection:**

1. What do you understand by instance mapping?
2. What are different types of instance mapping?
3. What are the rules for instance mapping?
4. How the mapping of the variables will take place in positional mapping?
5. How the mapping of the variables will take place in name mapping?
6. What is the use of dot operator in name mapping?
7. Is there any difference in synthesis design if we use instance mapping (Compare your result with earlier synthesis and schematic result.

**Experiment 4**

**Aim: Implementation of** 8**:1 mux using:**

**a) Dataflow modeling**

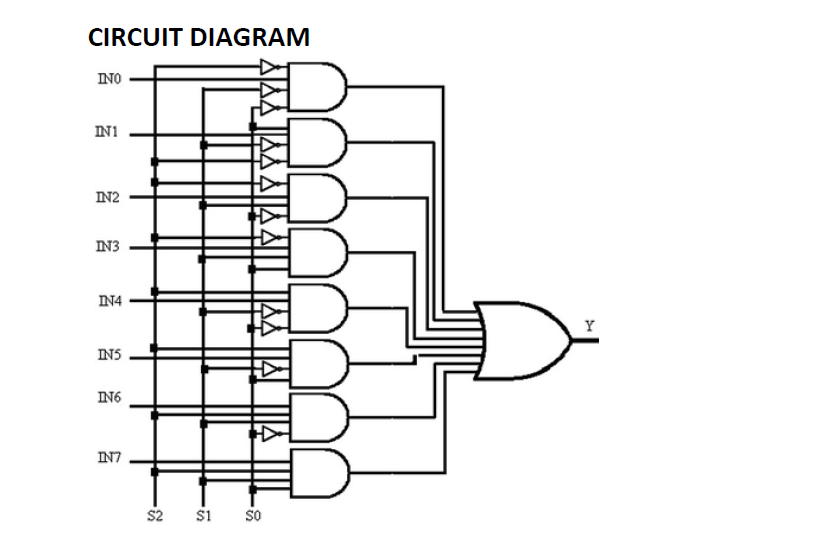
**b) Gate level modeling**

**c) Behavioral Modelling**

**Apparatus:** Xilinx Vivado Design Suite

**Theory:** Truth table of 8:1 Mux

|  |  |  |  |
| --- | --- | --- | --- |
| **S2** | **S1** | **S0** | **Y** |
| 0 | 0 | 0 | I0 |
| 0 | 0 | 1 | I1 |
| 0 | 1 | 0 | I2 |
| 0 | 1 | 1 | I3 |
| 1 | 0 | 0 | I4 |
| 1 | 0 | 1 | I5 |
| 1 | 1 | 0 | I6 |
| 1 | 1 | 1 | I7 |



**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

**Gate Level Modelling**

module mux81\_gate (

input [7:0] I,

input S0,S1,S2,

output Y

);

wire [11:1] T;

not(T[1], S0);

not(T[2], S1);

not(T[3], S2);

and(T[4], I[0], T[1], T[2], T[3]), (T[5], I[1], S0, T[2], T[3]);

and(T[6], I[2], T[1], S1, T[3]), (T[7], I[3], S0, S1, T[3]);

and(T[8], I[4], T[1], T[2], S2), (T[9], I[5], S0, T[2], S2);

and(T[10], I[6], T[1], S1, S2), (T[11], I[7], S0, S1, S2);

or(Y, T[4], T[5], T[6], T[7], T[8], T[9], T[10], T[11]);

endmodule

**Dataflow Modelling**

module mux81\_dataflow(

input i0,i1,i2,i3,i4,i5,i6,i7,

input s0,s1,s2,

output out

);

assign out = s2?(s1?(s0?i7:i6):(s0?i5:i4)):(s1?(s0?i3:i2):(s0?i1:i0));

endmodule

module mux8x1(

input [7:0] i,

input [2:0] s,

output Y

);

wire [7:0] y;

assign y[0] = ~s[2] & ~s[1] & ~s[0] & i[0];

assign y[1] = ~s[2] & ~s[1] & s[0] & i[1];

assign y[2] = ~s[2] & s[1] & ~s[0] & i[2];

assign y[3] = ~s[2] & s[1] & s[0] & i[3];

assign y[4] = s[2] & ~s[1] & ~s[0] & i[4];

assign y[5] = s[2] & ~s[1] & s[0] & i[5];

assign y[6] = s[2] & s[1] & ~s[0] & i[6];

assign y[7] = s[2] & s[1] & s[0] & i[7];

assign Y = y[0] | y[1] | y[2] | y[3] | y[4] | y[5] | y[6] | y[7];

endmodule

**Behavioral Modelling:**

module mux8x1(

input [7:0] i,

input [2:0] s,

output y

);

reg y;

always@(s or i)

case(s)

3'b000: y = i[0];

3'b001: y = i[1];

3'b010: y = i[2];

3'b011: y = i[3];

3'b100: y = i[4];

3'b101: y = i[5];

3'b110: y = i[6];

3'b111: y = i[7];

endcase

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematics of the design.

**Conclusion:**

**Questions for Reflection:**

1. What are the constraints for data flow modelling style?
2. What are the constraints for gate level modeling style?
3. What is conditional data operator?
4. What do you mean by [7:0]i?
5. How execution of assign statement will take place?
6. Whether all assign statements execute sequentially or concurrently?
7. Why behavioral modelling is called highest abstraction level?
8. What do you mean by always @
9. Is there any difference in RTL and technology schematics of the various modelling styles of Verilog?
10. Is there any difference in synthesis design for different codes used in the program (Compare your result with 3 codes written in the program)

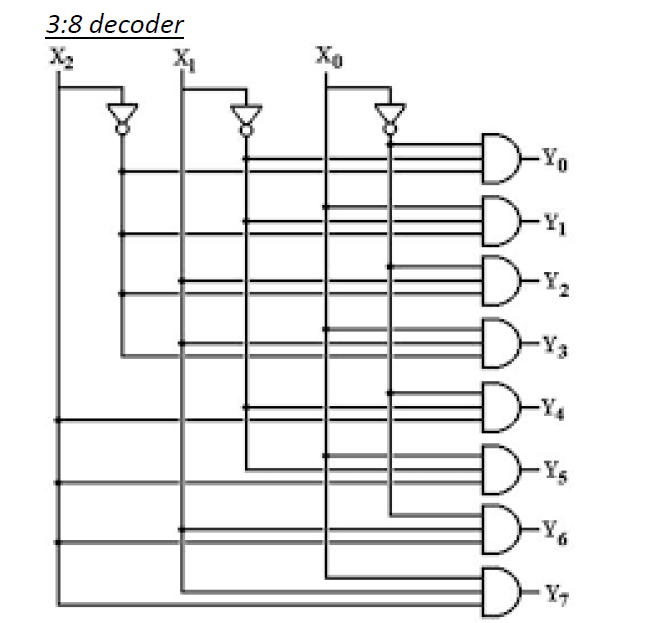
**Experiment 5**

**Aim: Implementation of 3:8 decoder using Behavioral Modelling in Verilog.**

**Software:** Xilinx Vivado Design Suite

**Theory:** Truth table of 8:1 Mux

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **IN2** | **IN1** | **IN0** | **OUT7** | **OUT6** | **OUT5** | **OUT4** | **OUT3** | **OUT2** | **OUT1** | **OUT0** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

****

**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

module decoder3to8\_ (

input [2:0] in,

output reg [7:0] out

);

always @(in)

case (in)

3'b000 : out = 8'b00000001;

3'b001 : out = 8'b00000010;

3'b010 : out = 8'b00000100;

3'b011 : out = 8'b00001000;

3'b100 : out = 8'b00010000;

3'b101 : out = 8'b00100000;

3'b110 : out = 8'b01000000;

3'b111 : out = 8'b10000000;

default : out = 8'b00000000;

endcase

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematics of the design.

**Conclusion:**

**Questions for Reflection:**

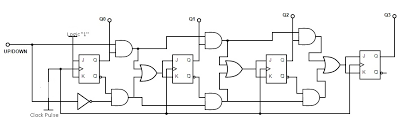
1. Why reg is required for output variable in Behavioral modelling?
2. What is the meaning of always@(\*)?
3. How case statement is executed?
4. Change the code using if- else condition.

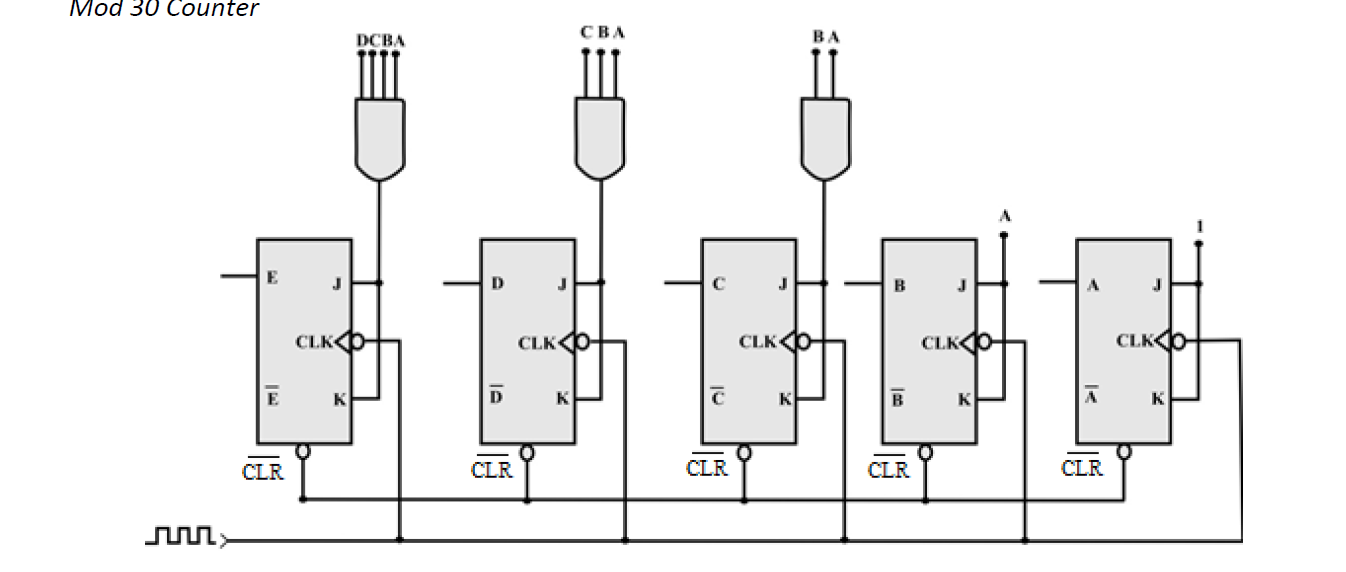
**Experiment 6**

**Aim: Implementation of 4 bit up- down counter and Mod 30 using Behavioral Modelling in Verilog.**

**Software:** Xilinx Vivado Design Suite

**Circuit Diagram:**



****

**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

module updowncounter (clk, reset, UpOrDown, Count);

input clk, reset, M;

output [3 : 0] Count;

reg[3 : 0] Count =0;

always @(posedge clk)

begin

if(reset == 1)

Count <= 0;

else

if(M == 1)

if(Count == 15)

Count <=0;

else

Count <= Count+1;

else

if(Count==0)

Count <= 15;

else

Count <= Count -1;

end

endmodule

**Test Bench**

module updowncounter

reg clk;

reg reset;

reg UpOrDown;

// Outputs

wire [3:0] Count;

// Instantiate the Unit Under Test (UUT)

updowncounter\_sid\_113 uut (

.clk(clk),

.reset(reset),

.UpOrDown(UpOrDown),

.Count(Count)

);

initial clk = 0;

always #5 clk = ~clk;

initial begin

// Apply Inputs

reset = 1;

UpOrDown = 0;

#300;

reset = 0;

UpOrDown = 1;

#100;

UpOrDown = 0;

end

initial

$monitor("reset=%0d UpOrDown=%0d Count=%0d simtime=%g clk-%b",reset ,UpOrDown ,Count ,$time , clk);

endmodule

**Mod 30 Counter code:**

module m30 (

input clk, reset,

output reg [4:0] Q

);

always @ (posedge clk)

if (reset)

begin Q <= 5'b00000; end

else

begin

if (Q == 5'b11110)

begin Q <= 5'b0; end

else

begin Q <= Q+1; end

end

endmodule

**Test Bench:**

module testmod30;

// Inputs

reg clk;

reg reset;

// Outputs

wire [4:0] Q;

// Instantiate the Unit Under Test (UUT)

m30 uut (

.clk(clk),

.reset(reset),

.Q(Q)

);

initial clk = 0;

always #5 clk = ~clk;

initial begin

reset = 1;

#50;

reset = 0;

end

initial

$monitor("reset=%0d simtime=%g clk-%b",reset ,$time , clk);

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematics of the design.

**Conclusion:**

**Questions for Reflection:**

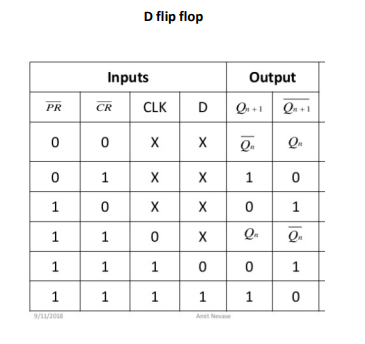
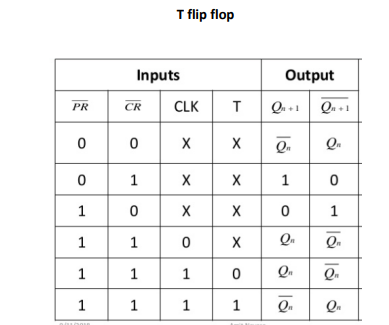
1. What is the need of test bench?
2. How input and output is defined in the test bench?
3. What is the use of dot operator in test bench?
4. What are the rules to define module and variables in the test bench?
5. What is the function of initial block in the test bench?
6. Is there any restriction for number of initial blocks in the test bench?
7. How the execution of various initial blocks will take place in the Verilog modelling?
8. How the execution of statements written inside the initial block will take place?
9. What is the difference between blocking and non-blocking statements?
10. What is the use of $monitor in the test bench?

**Experiment 7**

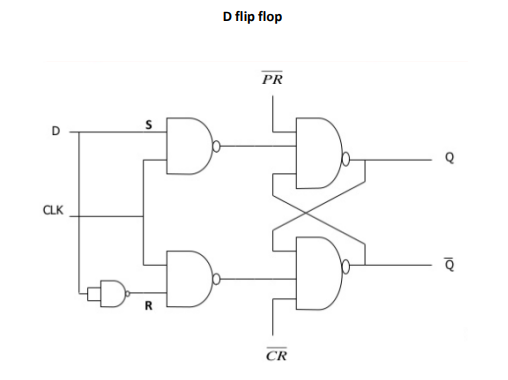
**Aim: Implementation of D and T flip flops in Verilog.**

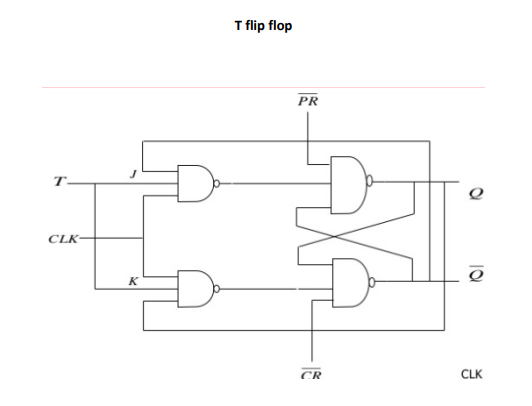
**Software:** Xilinx Vivado Design Suite

**Truth Table:**

**Circuit Diagram:**





**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

**D flip flop**

module d\_ff (

input clk, d, set, reset,

output reg q

);

always @ (posedge clk or posedge set or posedge reset)

begin

if (set)

q <= 1′b1;

else if (reset)

q <= 1′b0;

else

q <= d;

end

endmodule

**Test bench:**

D flip flop

Code

module dff\_test\_siddharth\_113;

// Inputs

reg clk;

reg d;

reg set;

reg reset;

// Outputs

wire q;

// Instantiate the Unit Under Test (UUT)

d\_ff uut (

.clk(clk),

.d(d),

.set(set),

.reset(reset),

.q(q)

);

initial begin

clk=0;

forever #10 clk = ~clk;

end

initial begin

reset=1; d <= 0;

#100; set=1; d <= 1;

#100; d <= 0;

#100; d <= 1;

$monitor("clk = %b, d = %b,reset = %b,set = %b, q = %b",clk, d, reset, set, q);

end

endmodule

**T flip flop**

module t\_ff(

input clk, t, set, reset,

output reg q

);

always @ (posedge clk or posedge set or posedge reset)

begin

if (set)

q <= 1'b1;

Siddharth Sameer 113

4

else if (reset)

q <= 1'b0;

else

q <= ~t;

end

endmodule

Test Bench:

T flip flop

Code

module tff\_test;

// Inputs

reg clk;

reg t;

reg set;

reg reset;

// Outputs

wire q;

// Instantiate the Unit Under Test (UUT)

t\_ff uut (

.clk(clk),

.t(t),

.set(set),

.reset(reset),

.q(q)

);

initial begin

clk=0;

forever #10 clk = ~clk;

end

initial begin

reset=1; t <= 0;

#100; set=1; t <= 1;

#100; t <= 0;

#100; t <= 1;

$monitor("clk = %b, t = %b,reset = %b,set = %b, q = %b",clk, t, reset, set, q);

end

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematics of the design.

**Conclusion:**

**Questions for Reflection:**

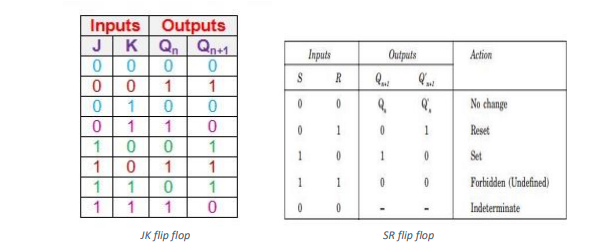
1. What is the need of test bench?
2. How input and output is defined in the test bench?
3. What is the use of dot operator in test bench?
4. What do you mean by #100 in the test bench code?
5. What is the alternative for forever?
6. What are the rules to define module and variables in the test bench?
7. What is the function of initial block in the test bench?
8. Is there any restriction for number of initial blocks in the test bench?
9. How the execution of various initial blocks will take place in the Verilog modelling?
10. How the execution of statements written inside the initial block will take place?
11. What is the difference between blocking and non-blocking statements?
12. Is test bench code synthesizable?

**Experiment 8**

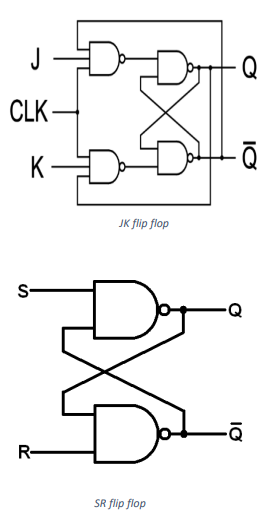
**Aim: Implementation of SR and JK flip flops in Verilog.**

**Software:** Xilinx Vivado Design Suite

**Truth Table:**



**Circuit Diagram**:



**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**JK flip flop Code:**

module jkff (input j,k,clk,

output reg q );

always @ (posedge clk)

case ({j,k})

2'b00 : q <= q;

2'b01 : q <= 0;

2'b10 : q <= 1;

2'b11 : q <= ~q;

endcase

endmodule

module jkff\_test;

// Inputs

reg j;

reg k;

reg clk;

// Outputs

wire q;

always #5 clk = ~clk;

// Instantiate the Unit Under Test (UUT)

Jkff uut (

.j(j),

.k(k),

.clk(clk),

.q(q)

);

initial begin

j <= 0;

k <= 0;

#5 j <= 0;

k <= 1;

#20 j <= 1;

k <= 0;

#20 j <= 1;

k <= 1;

#20 $finish;

end

initial

$monitor ("j=%0d k=%0d q=%0d", j, k, q);

Endmodule

**SR FF Code:**

module srff (

s,r,clk, q

);

input s,r,clk;

output reg q;

always@(posedge clk)

begin

if(s == 1)

begin

q <= 1;

end

else if(r == 1)

begin

q <= 0;

end

else if(s == 0 & r == 0)

begin

q <= q;

end

end

endmodule

module srff\_test;

// Inputs

reg s;

reg r;

reg clk;

// Outputs

wire q;

// Instantiate the Unit Under Test (UUT)

srff\_ uut (

.s(s),

.r(r),

.clk(clk),

.q(q)

);

initial

$monitor("s = %b, r = %b, q = %b",s, r, q);

initial begin

clk=0;

forever #10 clk = ~clk;

end

initial begin

s= 1; r= 0;

#100; s= 0; r= 1;

#100; s= 0; r= 0;

#100; s= 1; r=1;

end

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematics of the design.

**Conclusion:**

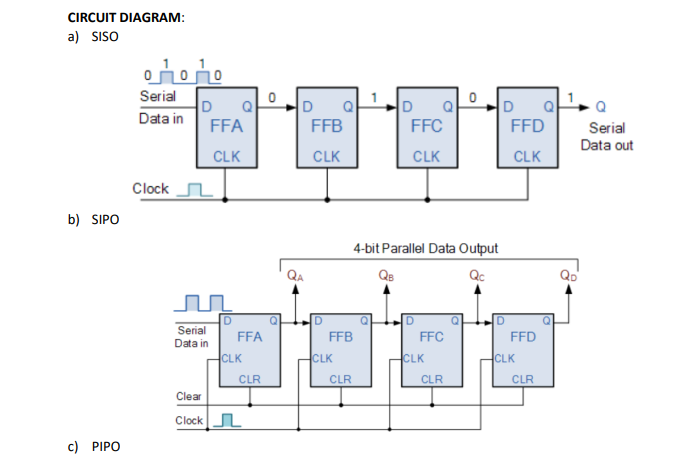
**Questions for Reflection:**

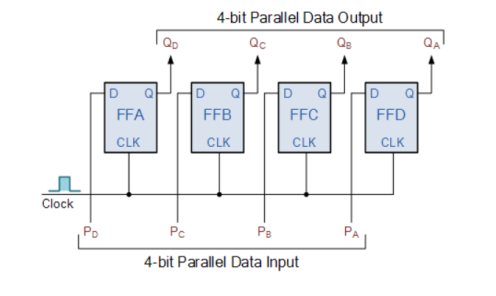
1. What is the need of test bench?
2. How input and output is defined in the test bench?
3. What is the use of dot operator in test bench?
4. What do you mean by #100 in the test bench code?
5. What is the alternative for forever?
6. What are the rules to define module and variables in the test bench?
7. What is the function of initial block in the test bench?
8. Is there any restriction for number of initial blocks in the test bench?
9. How the execution of various initial blocks will take place in the Verilog modelling?
10. How the execution of statements written inside the initial block will take place?
11. What is the difference between blocking and non-blocking statements?
12. Is test bench code synthesizable?

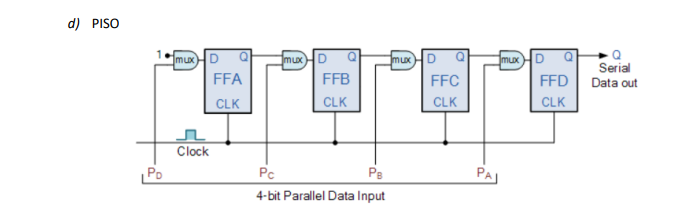
**Experiment 9**

**Aim: Implementation of shift registers in Verilog.**

**Software:** Xilinx Vivado Design Suite







**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Codes:**

**a) SISO**

module siso(clk, SI, SO);

input clk,SI;

output SO;

reg [7:0] tmp;

always @(posedge clk)

begin

tmp = tmp << 1;

tmp[0] = SI;

end

assign SO = tmp[7];

endmodule

**b) SIPO**

module sipo(clk, SI, PO);

input clk,SI;

output [7:0] PO;

reg [7:0] tmp;

always @(posedge clk)

begin

tmp = {tmp[6:0], SI};

end

assign PO = tmp;

endmodule

**c) PIPO**

module pipo(din,clk,rst,dout);

input [3:0] din;

input clk,rst;

output [3:0] dout;

wire [3:0] din;

wire clk,rst;

reg [3:0] dout;

always @(posedge clk or negedge rst)

begin

if(!rst)

begin

dout <= 4'b0;

end

else

begin

dout <= din;

end

end

endmodule

Testbench for PIPO

module testbench\_pipo;

// Inputs

reg [3:0] din;

reg clk;

reg rst;

// Outputs

wire [3:0] dout;

// Instantiate the Unit Under Test (UUT)

pipo uut (

.din(din),

.clk(clk),

.rst(rst),

.dout(dout)

);

initial begin

// Initialize Inputs

clk = 1'b0;

rst = 1'b0;

end

always

begin

#20 clk = ~clk;

end

initial

begin

#25 rst = 1'b1;

#10 din = 4'b1010;

#90 din = 4'b0110;

#500 $finish;

end

endmodule

**d) PISO**

module piso(din,clk,rst,dout);

input [3:0] din;

input clk,rst;

output [3:0] dout;

wire [3:0] din;

wire clk,rst;

reg [3:0] dout;

always @(posedge clk or negedge rst)

begin

if(!rst)

begin

dout <= 4'b0;

end

else

begin

dout <= din;

end

end

endmodule

Testbench for PISO

module testbench\_piso;

// Inputs

reg clk;

reg [3:0] PI;

reg load;

// Outputs

wire SO;

// Instantiate the Unit Under Test (UUT)

piso uut (

.clk(clk),

.PI(PI),

.load(load),

.SO(SO)

);

initial begin

// Initialize Inputs

clk = 0;

end

always

#10 clk = ~clk;

initial

begin

#25 load = 1;

#10 PI = 4'b1010;

#20 load = 0;

#100 load = 1;

#20 PI = 4'b1110;

#20 load = 0;

#300 $finish;

end

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematics of the design.

**Conclusion:**

**Questions for Reflection:**

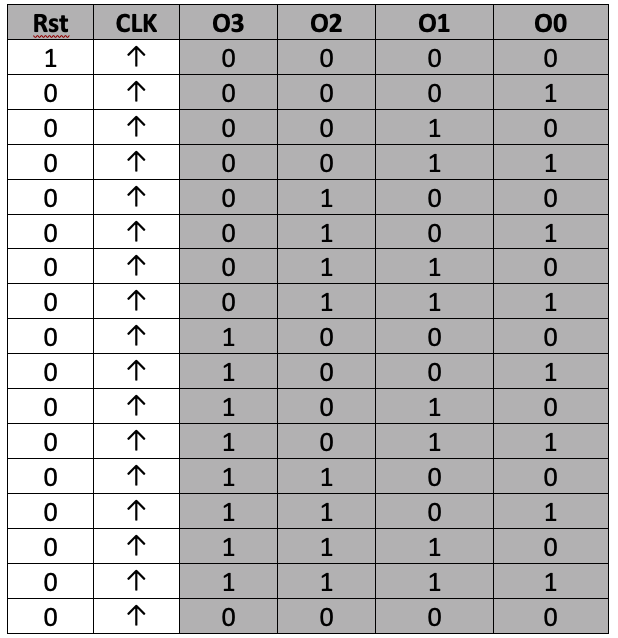
1. What is the need of test bench?
2. How input and output is defined in the test bench?
3. What is the use of dot operator in test bench?
4. What do you mean by #100 in the test bench code?
5. What is the alternative for forever?
6. What are the rules to define module and variables in the test bench?
7. What is the function of initial block in the test bench?
8. Is there any restriction for number of initial blocks in the test bench?
9. How the execution of various initial blocks will take place in the Verilog modelling?
10. How the execution of statements written inside the initial block will take place?
11. What is the difference between blocking and non-blocking statements?
12. Is test bench code synthesizable?

**Experiment 10**

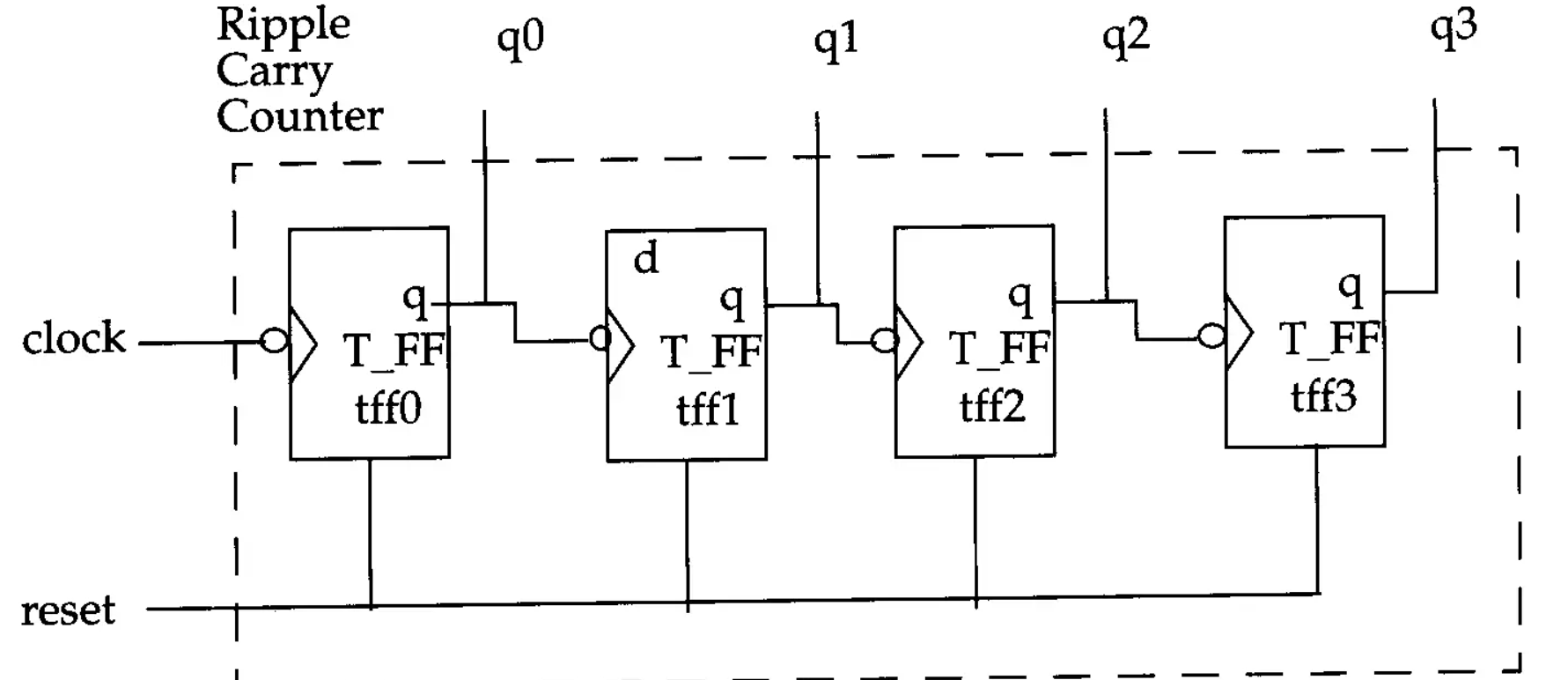
**Aim: Implementation of 4 bit Ripple counter using instances of T and D Flip flops**

**Software:** Xilinx Vivado Design Suite

**Truth Table:**



**Circuit Diagram:**



**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

module ripple\_carry\_counter (q, clk, reset);

output [3:0] q;

input clk, reset;

T\_FF tff0(q[0],clk, reset);

T\_FF tff1(q[1],q[0], reset);

T\_FF tff2(q[2],q[1], reset);

T\_FF tff3(q[3],q[2], reset);

endmodule

module T\_FF(q, clk, reset);

output q;

input clk, reset;

wire d;

D\_FF dff0(q, d, clk, reset); // Instantiate D\_FF. Call it dff0.

not n1(d, q);

endmodule

module D\_FF(q, d, clk,reset);

output q;

input d, clk, reset;

reg q;

always @(posedge reset or negedge clk)

if (reset)

q <= 1'b0;

else

q <= d;

endmodule

**Testbench:**

module ripple\_carry\_counter\_tb;

reg clk, reset;

wire [3:0] q;

ripple\_carry\_counter ripple\_carry\_counter\_inst(

.q(q), .clk(clk), .reset(reset)

);

initial clk = 0;

always #10 clk = ~clk;

initial

begin

reset = 1'b1;

#20 reset = 1'b0;

end

always @(posedge clk)

begin

$display("q = %b", q);

end

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematics of the design.

**Conclusion:**

**Questions for Reflection:**

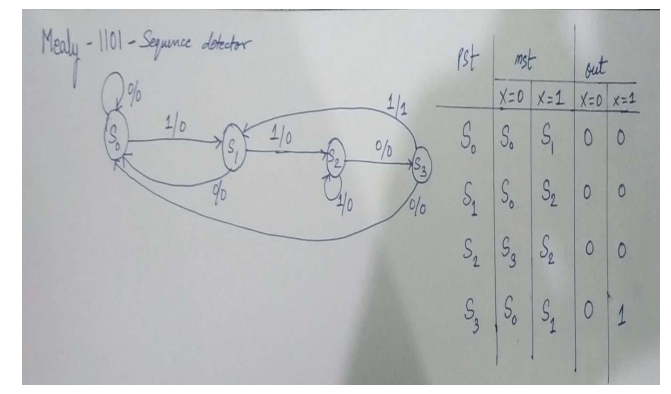
1. Explain hierarchy of the design?
2. What do you mean by top down and bottom-up approach of the design?
3. Which approach have you used in your design?
4. Have you created 3 different codes in the design and how these codes are linked with each other?
5. What are the rules to be followed for positional mapping?
6. How can you create instance using name mapping?

**Experiment 11**

**Aim: Implementation of Mealy FSM in Verilog.**

**Software:** Xilinx Vivado Design Suite

**State Diagram and State Table**



**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

module mealy1101(clk,reset,in,out);

input clk,reset,in;

output out;

reg out;

reg[2:0]pst,nst;

parameter

s0=3'b000,

s1=3'b001,

s2=3'b010,

s3=3'b011;

always @(posedge clk or posedge reset)

begin

if(reset==1)

pst<=s0;

else

pst<=nst;

end

always @(pst or in)

begin

case(pst)

s0:

if(in==1)

begin

nst<=s1;

out<=0;

end

else

nst<=pst;

s1:

if(in==1)

begin

nst<=s2;

out<=0;

end

else

begin

nst<=s0;

out<=0;

end

s2:

if(in==0)

begin

nst<=s3;

out<=0;

end

else

begin

nst<=s2;

out<=0;

end

s3:

if(in==1)

begin

nst<=s1;

out<=1;

end

else

begin

nst<=s0;

out<=0;

end

default

nst<=s0;

endcase

end

endmodule

**Test bench:**

module testbench\_mealy1101;

// Inputs

reg clk;

reg reset;

reg in;

// Outputs

wire out;

// Instantiate the Unit Under Test (UUT)

mealy1101 uut (

.clk(clk),

.reset(reset),

.in(in),

.out(out)

);

initial begin

// Initialize Inputs

clk = 0;

forever #5 clk=~clk;

end

initial begin

reset = 1;

in = 0;

#10 reset=0;

#20 in=0;

#20 in=1;

#10 in=1;

#10 in=0;

#10 in=1;

#10 in=0;

#20 in=0;

#20 in=1;

#10 in=1;

#10 in=0;

#10 in=1;

#100$finish;

end

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematics of the design.

**Conclusion:**

**Questions for Reflection:**

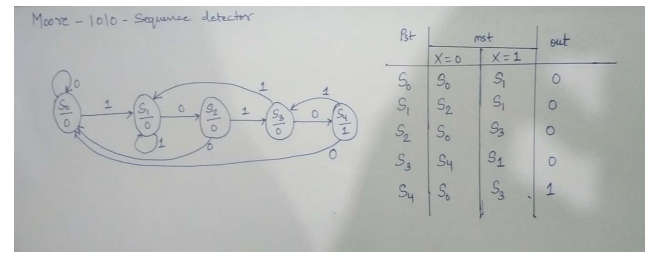
1. What is the difference between Mealy and Moore FSM?
2. Which FSM is more stable and why?
3. Draw FSM diagram for sequence detector 1100.
4. What do you mean by overlapping and non overlapping FSM?

**Experiment 12**

**Aim: Implementation of Moore FSM in Verilog.**

**Software:** Xilinx Vivado Design Suite

**State Diagram and State Table**



**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

module moore1010(clk,reset,in,out);

input clk,reset,in;

output out;

reg out;

reg[2:0]pst,nst;

parameter

s0=3'b000,

s1=3'b001,

s2=3'b010,

s3=3'b011,

s4=3'b100;

always @(posedge clk or posedge reset)

begin

if(reset==1)

pst<=s0;

else

pst<=nst;

end

always @(pst or in)

begin

case(pst)

s0:

if(in==1)

nst<=s1;

else

nst<=s0;

s1:

if(in==0)

nst<=s2;

else

nst<=s1;

s2:

if(in==1)

nst<=s3;

else

nst<=s0;

s3:

if(in==0)

nst<=s4;

else

nst<=s1;

s4:

if(in==1)

nst<=s3;

else

nst<=s0;

default

nst<=s0;

endcase

end

always@(pst)// moore logic

begin

if(pst==s4)

out<=1;

else

out<=0;

end

endmodule

**Test bench:**

module tt\_moore\_1010;

// Inputs

reg clk;

reg reset;

reg in;

// Outputs

wire out;

// Instantiate the Unit Under Test (UUT)

moore1010 uut (

.clk(clk),

.reset(reset),

.in(in),

.out(out)

);

initial

begin

reset=0;clk=0;in=0;

#5 in=0;

#5 in=0;

#5 in=1;

#10 in=0;

#10 in=1;

#10 in=0;

#10 in=1;

#10 in=0;

#10 in=0;

#5 in=0;

#5 in=1;

end

always

#5 clk=~clk;

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematics of the design.

**Conclusion:**

**Questions for Reflection:**

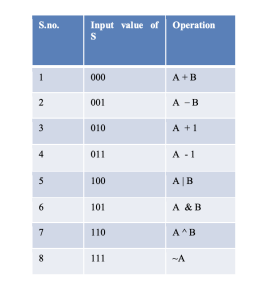
1. What is the difference between Mealy and Moore FSM?
2. Which FSM is more stable and why?
3. Draw FSM diagram for sequence detector 1100.
4. What do you mean by overlapping and non overlapping FSM?

**Experiment 13**

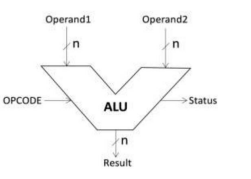
**Aim: Implementation of 8 bit ALU in Verilog.**

**Software:** Xilinx Vivado Design Suite

Truth Table:



Circuit Diagram:



**Procedure:**

1. Open the Xilinx Vivado Design Suite
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name of the FPGA Device, parameter setting and HDL is verilog click next and click finish.
5. Click new source.
6. Select Verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally, the report is shown click finish.
9. Type the program save and check syntax error.
10. To see the output waveform select ISim simulator
11. Give values to the input variables using force clock or force constant and then click run
12. In wave window, click run icon and you can see corresponding output.
13. For synthesis of the design, open XST synthesis tool and run the design for synthesis.
14. Open RTL schematic and Technology schematic and understand implemented design on FPGA
15. Open synthesis result to know resource utilization of the design.

**Code:**

Code of the program:

`timescale 1ns / 1ps

module alu (a, b, s, yout, cf);

input [7:0] a;

wire [7:0] a;

input [7:0] b;

wire [7:0] b;

input [2:0] s;

wire [2:0] s;

output [7:0] yout;

reg [7:0] yout;

output cf;

reg cf;

reg [8:0] temp;

always @ ( a,b,s )

begin

if ( s == 3'b000)

begin

temp = {0,a} + {0,b};

yout = temp [7:0];

cf = temp [8];

end

else if ( s == 3'b001)

begin

temp = {0,a}

- {0,b};

yout = temp [7:0];

cf = temp [8];

end

else if ( s == 3'b010)

begin

temp = {0,a} +5'b00001;

yout = temp [7:0];

cf = temp [8];

end

else if ( s == 3'b011)

begin

temp = {0,a}

- 5'b00001;

yout = temp [7:0];

cf = temp [8];

end

else if ( s == 3'b100)

begin

temp = a | b;

yout = temp [7:0];

cf = temp [8];

end

else if ( s == 3'b101)

begin

temp = a & b;

yout = temp [7:0];

cf = temp [8];

end

else if ( s == 3'b110)

begin

temp = a ^ b;

yout = temp [7:0];

cf = temp [8];

end

else if (s == 3'b100)

begin

temp = ~a;

yout = temp [7:0];

cf = temp [8];

end

end

endmodule

**Testbench**

`timescale 1ns / 1ps

Module test;

// Inputs

reg [7:0] a;

reg [7:0] b;

reg [2:0] s;

// Outputs

wire [7:0] yout;

wire cf;

// Instantiate the Unit Under Test (UUT)

Alu uut (

.a(a),

.b(b),

.s(s),

.yout(yout),

.cf(cf)

);

initial begin

$display("Inputs and Outputs are:");

$monitor("time= %t a=%b b=%b s=%b yout=%b cf=%b",$time,a,b,s,yout,cf);

end

initial begin

// Initialize Inputs

a = 0;

b = 0;

s = 0;

// Wait 100 ns for global reset to finish

#100;

a = 1;

b = 2;

s = 0;

#6

a = 1;

b = 2;

s = 2;

#6

a = 2;

b = 3;

s = 3;

#6

a = 6;

b = 3;

s = 4;

#6

a = 1;

b = 2;

s = 5;

#6

a = 2;

b = 4;

s = 6;

#6

a = 1;

b = 2;

s = 7;

#6

a = 5;

b = 2;

s = 1;

#6 $finish;

end

endmodule

**Result:**

1. Simulate the design using the simulation tool to verify its functionality.
2. Synthesis the design using synthesis tool to find out the resource utilization.
3. Draw RTL and Technology schematics of the design.

**Conclusion:**

**Questions for Reflection:**

1. What is the difference between $monitor and $ display?
2. What is the difference between $finish and $ stop?
3. What is default size of int variable?