**Digital System Design Lab**

**CEN-422**

# Lab Journal: 04



Name: M.OMAR

Class: BCE – 06(A)

## Enrollment No: 01-132182-024

**Lab # 04**

**Data Flow modeling**

**Objective:**

Implementing Data Flow Modeling.

**Introduction:**

**Verilog** is a hardware description language used to describe electronic circuits and digital systems. In practice it is generally used for simulating, testing, and programming PLD’s (programmable logic devices) or other similar hardware**.**

### Data Flow Modelling

Dataflow modeling provides the means of describing combinational circuits by their function rather than by their gate structure. Dataflow modeling uses a number of operators that act on operands to produce the desired results.

Dataflow modeling uses continuous assignments and the keyword assign. A continuous assignment is a statement that assigns a value to a net. The datatype net is used in Verilog HDL to represent a physical connection between circuit elements. The value assigned to the net is specified by an expression that uses operands and operators. As an example, assuming that the variables were declared, a 2-to-1 multilexer with data inputs A nad B, select input S, and output Y is described with continuous assignment assign

Y= (A & S) | (B & S)

**Task 01:**

**Write a Verilog code for the calculation of i’th power of two at dataflow modeling where i is given by user**

Code: module TaskOne(A, Y); input [3:0]A; output [3:0]Y; integer i=2;

assign Y=A<<i; endmodule

**#Testbench** module TaskOne\_Testbench; reg [3:0] a; wire [3:0] y;

TaskOne taskone(.A(a), .Y(y));

initial

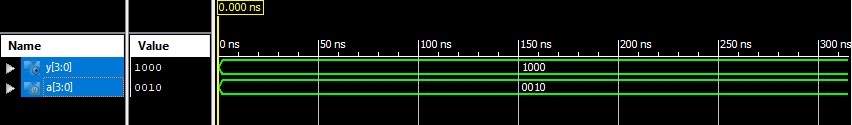
begin

$monitor("a=%b,y=%b",a,y); a = 4'b0010;

#100;

end endmodule

Output:



**Task 02:**

**Take 2’s complement of an input number. Write its verilog code at dataflow modeling.**

• **assign out = !(in) + 1;**

Code:

module TaskTwo(in, Y); input [3:0]in; output [3:0]Y;

assign Y = ~(in)+1;

endmodule

**#Testbench** module TaskTwo\_Testbench; reg [3:0] in; wire [3:0] y;

TaskTwo tasktwo (.in(in), .Y(y));

initial

begin

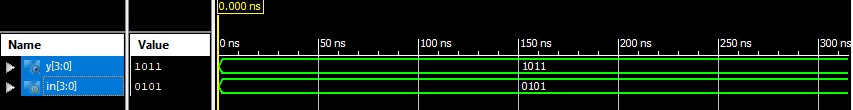
$monitor("in=%b , y=%b",in,y); in = 0101;

#100;

end

endmodule

Output:



**Task 03:**

**Split 8bit number into four 2bit numbers**

* **assign num1 = in[1:0];**
* **assign num2 = in[3:2];**
* **assign num3 = in[5:4];**
* **assign num4 = in[7:6];**

**Combine again all four 2bit numbers to form a single 9bit number by appending a single bit equal to 1 at LSB**

* **assign number = {num1, num2, num3, num4, 1’b1 };**

Code:

module TaskThree(in,n1,n2,n3,n4,num); input [7:0]in; output [1:0]n1,n2,n3,n4; output [8:0]num; assign n1 = in[1:0]; assign n2 = in[3:2]; assign n3 = in[5:4]; assign n4 = in[7:6]; assign num = {n1, n2, n3, n4, 1'b1 };

endmodule

**#Testbench** module TaskThree\_Testbench; reg [7:0] in; wire [1:0] n1; wire [1:0] n2; wire [1:0] n3; wire [1:0] n4; wire [8:0] num;

TaskThree taskthree(.in(in), .n1(n1), .n2(n2), .n3(n3), .n4(n4), .num(num));

initial

begin

$monitor("n1=%b,n2=%b,n3=%b,n4=%b ,

num=%b",n1[1:0],n2[1:0],n3[1:0],n4[1:0],num);

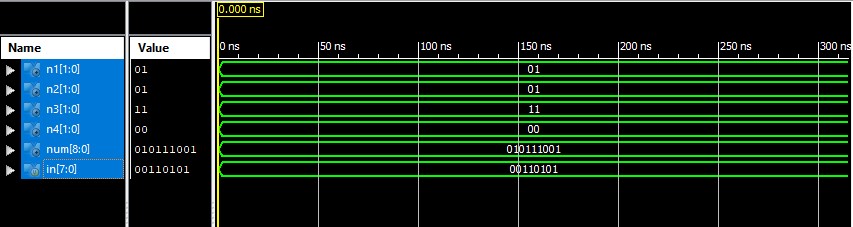
in = 8'b00110101;

#100;

end

endmodule

Output:



**Task 04:**

**If A is greater than B then out1 should be assigned ‘1’ else it should be zero.**

**Also If A is equal to B then out2 should be assigned ‘1’ else it should be zero.**

* **assign out2 = (A>B) ? 1: 0;**
* **assign out2 = (A==B)? 1: 0;**

Code:

module TaskFour(a,b,out1,out2); input [3:0]a,b; output out1,out2; assign out1 = (a>b)?1:0; assign out2 = (a==b)?1:0;

endmodule

**#Testbench**

module TaskFour\_Testbench; reg [3:0] a; reg [3:0] b; wire out1; wire out2;

TaskFour taskfour (.a(a), .b(b), .out1(out1), .out2(out2));

initial

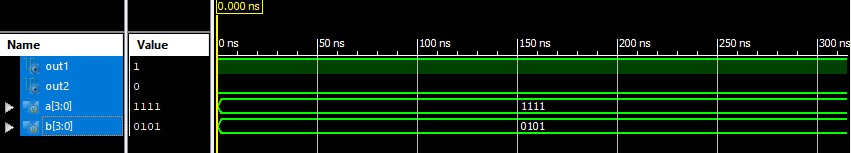
begin

$monitor("a=%b , b=%b , out1=%b , out2=%b",a,b,out1,out2);

a=4'b1111; b=4'b0101; #100; end

endmodule

Output:



**Task 05:**

**Take modulus of 4 bit number by 4. Write Verilog code at dataflow level.**

Code:

module TaskFive(A, Y); input [3:0]A; output [3:0]Y; assign Y = A%4;

endmodule

**#Testbench** module TaskFive\_Testbench; reg [3:0] a; wire [3:0] y;

TaskFive taskfive(.A(a), .Y(y));

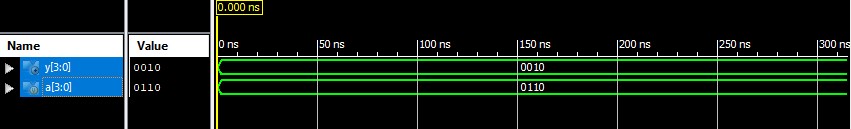
initial

begin

$monitor("a=%b , y=%b",a,y); a = 4'b0110; end

endmodule

Output:



**Task 06:**

**Implement 2 to 1 mux and 4 to 1 mux using conditional operator in Verilog.**

**2 to 1 Mux:**

Code:

module TaskSix(A,B,S,Y);

input A,B,S; output Y; assign Y = S?A:B; endmodule

**#Testbench**

module TaskSix\_Testbench;

reg a; reg b; reg s; wire y;

TaskSix tasksix(.A(a), .B(b), .S(s), .Y(y));

initial

begin

$monitor("s=%b , a=%b , b=%b , y=%b",s,a,b,y);

a = 0; b = 1; s = 0; #100;

1. = 0;
2. = 1; s = 1; #100;

end

endmodule

Output:



**4 to 1 Mux:**

Code:

module TaskSeven(i0,i1,i2,i3,s0,s1,out); input i0,i1,i2,i3,s0,s1;

output out;

assign out = s1?(s0?i3:i2):(s0?i1:i0);

endmodule

**#Testbench** module TaskSeven\_Testbench;

reg i0; reg i1; reg i2; reg i3; reg s0; reg s1; wire out;

TaskSeven taskseven (.i0(i0), .i1(i1), .i2(i2), .i3(i3), .s0(s0), .s1(s1), .out(out));

initial

begin

$monitor("s0=%b , s1=%b , i0=%b , i1=%b , i2=%b , i3=%b , out=%b",s0,s1,i0,i1,i2,i3,out);

i0 = 0; i1 = 1; i2 = 1; i3 = 1; s0 = 0; s1 = 0;

#100;

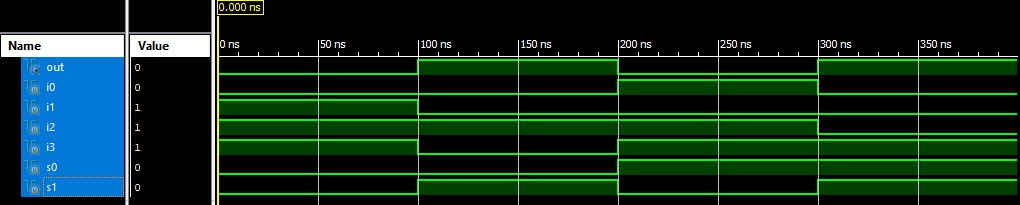
i0 = 0;

i1 = 0; i2 = 1; i3 = 0; s0 = 0; s1 = 1; #100;

i0 = 1; i1 = 0; i2 = 1; i3 = 1; s0 = 1; s1 = 0; #100;

i0 = 0; i1 = 0; i2 = 0; i3 = 1; s0 = 1; s1 = 1; #100;

end endmodule Output:



**Conclusion:**

In this lab we LEARNED ABOUT Data Flow Modeling.