DIGITAL SYSTEM DESIGN LAB CSE-308L

SEMESTER:6TH

LAB REPORT # 2

Submitted By: *Zainab Khalid*
Registration No:*19PWCSE1743*
Section: A
Submitted to: *Mam Madeeha Sher*

DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING
UNIVERSITY OF ENGINEERING AND TECHNOLOGY PESHAWAR

LAB No 2

INTRODUCTION TO MODELSIM AND GATE LEVEL MODELING

Objectives:

This lab will enable students to:

- Learn top down and bottom up design methodologies
- Data flow level modeling
- Data gate level modeling

Data Gate Level Modeling.

Designing circuits using basic logic gates is known as gate-level modeling. A digital circuit is implemented using logic gates and interconnections between these gates.

Data Flow Level Modeling.

Dataflow modeling uses a number of operators that act on operands to produce the desired results. Verilog HDL provides about 30 operator types like &, |, ^, ==, >, <,? etc. Dataflow modeling uses continuous assignments and the keyword ***assign***. It is called continuous assignment because it remains active all the time.

assign out = a&b;

The left hand side variable must be of type wire. At the right hand side, there can be a single variable or expression and of type reg or wire.

Operators used:

Arithmetic: +,-,*,/

Logical: **&&, ||**

Bitwise: **&, |, ~, ^**

Reduction: **~|, ~&, ~^**

Shift: <<, >>

Equality: ***==, !=, (case equality: ===, !==)***

Conditional: ? , :

Concatenation: ***{}***

Example: a=01;b=110; c={a,b};

Replication: ***{{}}***

Example: A=111111000000

x={6{1}}; y={6{0}};

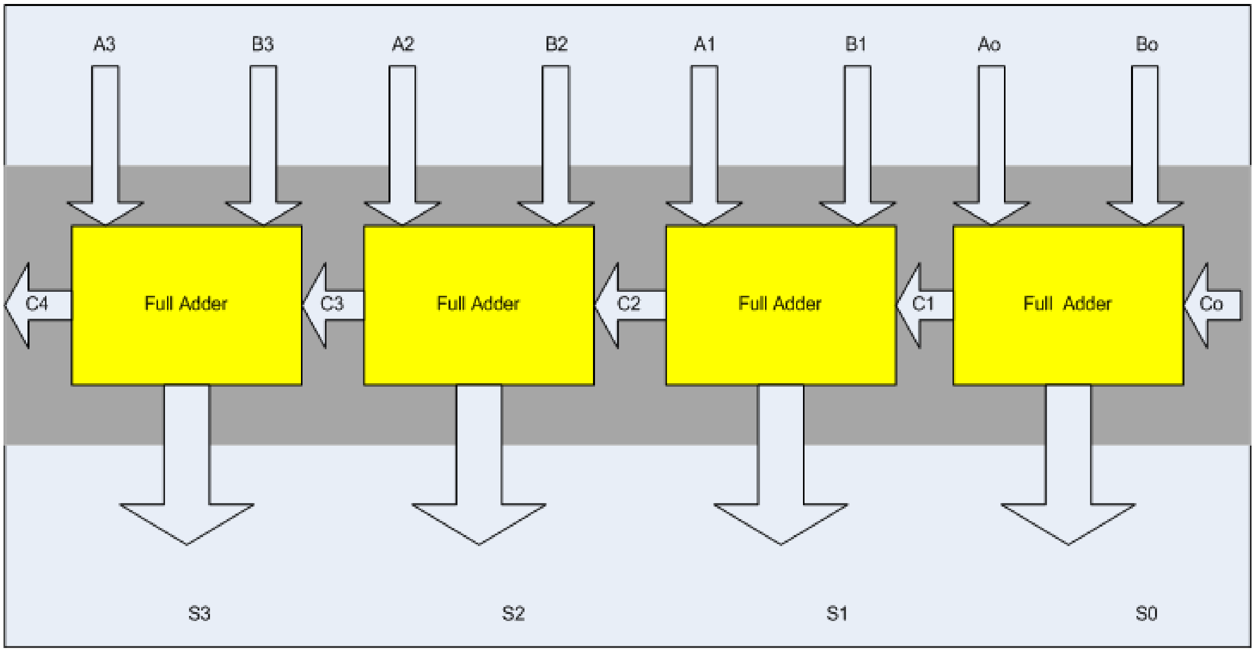
A={x,y} or A={6{1},6{0}};

TASK 1:

4 bit Ripple Carry Adder

Block Diagram:

Following is the block diagram of a 4 bit Ripple Carry Adder.



I/O Connection:

Ground “Co” Permanently and connect S0-S3 with four LEDs also connect C4 to another LED.

Steps for task 1a:

The following steps should be performed while designing a 4 bit RCA adder

ModelSim:

1- First implement a Full adder using data gate level modeling.

Data Gate Level Modeling.

Code (Full Adder):

```verilog
//Structural Model : Half Adder
module half_adder(output S,C,input A,B);
 xor(S,A,B);
 and(C,A,B);
endmodule

//Structural Model : Full Adder
module full_adder(S,Cout,A,B,Cin);
 input A,B,Cin;
 output S,Cout;
 wire s1,c1,c2;
 half_adder HA1(s1,c1,A,B);
 half_adder HA2(S,c2,s1,Cin);
 or o1(Cout,c1,c2);
endmodule
```

2- Simulate the Full adder with a test bench.

<u>Code (Test Bench For Full Adder):</u>

```verilog
module test_FA(); //test bench to test FA (Full Adder)
        // Inputs are regs here
        reg A, B, Cin;
        // Outputs are wires here
        wire S, Cout;
        // Instantiate the Unit/Design Under Test (U/DUT)
        // FA dut (S, Cout, A, B, Cin);  // Positional association
        full_adder  fa(.A(A), .B(B), .Cin(Cin), .S(S), .Cout(Cout));  // Explicit association

        initial begin
                // Initialize Inputs
                A = 0; B = 0; Cin = 0;
                #10
                Cin = 1;
                #10
                B = 1;
                #10
                A = 1;
                #10
                A = 0;
                #10
                Cin = 0;
        end

        initial begin
                $monitor("A=%b,B=%b,Cin=%b--->Sum=%b,Cout=%b  \n",A,B,Cin,S,Cout);
        end

endmodule
```

<u>Output (Full Adder):</u>

```
# Compile of fulladder.v was successful.
vsim work.test_FA
# vsim work.test_FA
# Loading work.test_FA
# Loading work.full_adder
# Loading work.half_adder
run
# A=0,B=0,Cin=0--->Sum=0,Cout=0
#
# A=0,B=0,Cin=1--->Sum=1,Cout=0
#
# A=0,B=1,Cin=1--->Sum=0,Cout=1
#
# A=1,B=1,Cin=1--->Sum=1,Cout=1
#
# A=0,B=1,Cin=1--->Sum=0,Cout=1
#
# A=0,B=1,Cin=0--->Sum=1,Cout=0
#
```

3- Instantiate the Full adder four times and connect the circuit as shown.

Code (Ripple Adder):

```
//Structural Model : 4 Bit Ripple Carry Adder
module ripple_adder_4bit(Cout,Sum,A,B,Cin);
 output [3:0] Sum;
 output Cout;
 input [3:0] A,B;
 input Cin;
wire c1,c2,c3;
 //full_adder fa(S,Cout,A,B,Cin);
 full_adder FA1(Sum[0],c1,A[0],B[0],Cin),
    FA2(Sum[1],c2,A[1],B[1],c1),
    FA3(Sum[2],c3,A[2],B[2],c2),
    FA4(Sum[3],Cout,A[3],B[3],c3);
endmodule
```

4- Now again write a test bench and simulate the 4 bit RCA.

Code (Test Bench For Ripple Adder):

```
//Test Bench : 4 Bit Ripple Carry Adder
module test_ripple_adder_4bit();
 // Inputs
 reg [3:0] A;
 reg [3:0] B;
 reg Cin;
 // Outputs
 wire [3:0] Sum;
 wire Cout;
 // Instantiate the Unit Under Test (UUT)
 ripple_adder_4bit rpa(.Sum(Sum), .Cout(Cout), .A(A), .B(B),.Cin(Cin));
 initial
        begin

        $monitor("A=%d   B=%d   Cin=%d  Sum=%d          Cout=%d",A,B,Cin,Sum,Cout);
        A=4'b0000;B=4'b0000;Cin=1'b0;
        #10
        A=4'b0010;B=4'b0010;Cin=1'b0;
        #10
        A=4'b0010;B=4'b0100;Cin=1'b0;
        #10
        A=4'b0110;B=4'b1010; Cin=1'b0;
        #10
        A=4'b1111;B=4'b1111;Cin=1'b0;
         end
endmodule
```

Output (Ripple Adder):

```
# Compile of example.v was successful.
vsim work.test_ripple_adder_4bit
# vsim work.test_ripple_adder_4bit
# Loading work.test_ripple_adder_4bit
# Loading work.ripple_adder_4bit
# Loading work.full_adder
# Loading work.half_adder
run
# A= 0   B= 0 Cin=0 Sum= 0  Cout=0
# A= 2   B= 2 Cin=0 Sum= 4  Cout=0
# A= 2   B= 4 Cin=0 Sum= 6  Cout=0
# A= 6   B=10 Cin=0 Sum= 0  Cout=1
# A=15   B=15 Cin=0 Sum=14  Cout=1
```

Xilinx:

1. Make new project in Xilinx and add the files that you simulated in ModelSim.
2. Add User Constraint File inputs should be locked with the switches, C0 should be permanently "0" while S0-S4 and C4 with LEDS

Task 1b:

Design the 4 bit full adder using data flow level modeling.

Data Flow Level Modeling:

Code (Ripple Adder):

```
module SUM (S, A, B, Cin);
        output S;
        input A, B, Cin;
        // Behavioral Code
        assign S = A^B^Cin;

endmodule

module CARRY (Cout, A, B, Cin);
        output Cout;
        input A, B, Cin;
        // Behavioral Code
        assign Cout = (A&B) | (B&Cin) | (A&Cin);
endmodule

module FA1 (Sum, Cout, A, B, Cin);
        output Sum, Cout;
        input A, B, Cin;
        // Structural Code
        SUM s1 (.S(Sum), .A(A), .B(B), .Cin(Cin));
        CARRY c1 (.Cout(Cout), .A(A), .B(B), .Cin(Cin));
endmodule
module RCA (Cout, S, A, B,Cin);
        output Cout;
        output [3:0] S;
        input [3:0] A, B;
        input Cin;
        wire [2:0] C;   //Intermediate/Internal Carries
        // FA1 ;   //FA1's Interface (I/O Pins)
        FA1 fa0 (S[0], C[0], A[0], B[0], Cin);
        FA1 fa1 (S[1], C[1], A[1], B[1], C[0]);
        FA1 fa2 (S[2], C[2], A[2], B[2], C[1]);
        FA1 fa3 (S[3], Cout, A[3], B[3], C[2]);
endmodule

```

Code (Test Bench):

```
module stim_RCA(); //test bench
reg [3:0]A,B; //4 bits inputs
reg Cin;
wire Cout; //1 bits output
wire [3:0]S; //4 bits outputs
RCA r(Cout, S, A, B,Cin);
initial
        begin
        $monitor("A=%d   B=%d   Cin=%d  Sum=%d          Cout=%d",A,B,Cin,S,Cout);
        A=4'b0000;B=4'b0000;Cin=1'b0;
        #10
        A=4'b0010;B=4'b0010;Cin=1'b0;
        #10
        A=4'b0010;B=4'b0100;Cin=1'b0;
        #10
        A=4'b0110;B=4'b1010; Cin=1'b0;
        #10
        A=4'b1111;B=4'b1111;Cin=1'b0;
        end
endmodule
```

Output:

```
# Compile of rippleadder.v was successful.
vsim work.stim_RCA
# vsim work.stim_RCA
# Loading work.stim_RCA
# Loading work.RCA
# Loading work.FA1
# Loading work.SUM
# Loading work.CARRY
run
# A= 0   B= 0 Cin=0 Sum= 0  Cout=0
# A= 2   B= 2 Cin=0 Sum= 4  Cout=0
# A= 2   B= 4 Cin=0 Sum= 6  Cout=0
# A= 6   B=10 Cin=0 Sum= 0  Cout=1
# A=15   B=15 Cin=0 Sum=14  Cout=1
```

2nd Code:

```
module rippleadder(cout,sum,a,b,cin);
input cin; //1 bits input
input [3:0]a,b;//4 bits inputs
output cout; ////1 bits output
output [3:0]sum; //4 bits outputs
assign {cout,sum}=a+b+cin; //behavioral modeling
endmodule

module stim_ripple(); //test bench
reg cin; //1 bits input
reg [3:0]a,b; //4 bits inputs
wire cout; //1 bits output
wire [3:0]sum; //4 bits outputs
rippleadder ra(cout,sum,a,b,cin);
initial
        begin
```

```
        begin
        $display("A     B       Cin             Sum             Cout");
        a=4;b=6;cin=1'b0;
        #10
        $display("%d %d  %d                     %d                      %d",a,b,cin,sum,cout);
        a=11;b=5;cin=1'b0;
        #10
        $display("%d %d  %d                     %d                      %d",a,b,cin,sum,cout);
        a=9;b=3;cin=1'b0;
        #10
        $display("%d %d %d                      %d                      %d",a,b,cin,sum,cout);
        a=2;b=1;cin=1'b0;
        #10
        $display("%d %d  %d                     %d                      %d",a,b,cin,sum,cout);
        a=5;b=3;cin=1'b0;
        #10
        $display("%d %d  %d                     %d                      %d",a,b,cin,sum,cout);
        end

//initial
        //begin
        //$monitor("A=%b  B=%b          Cin=%b          Sum=%b          Cout=%b",a,b,cin,sum,cout);
        //end
endmodule
```

Output:

```
# Compile of 4bitrippleadder.v was successful.
vsim work.stim_ripple
# vsim work.stim_ripple
# Loading work.stim_ripple
# Loading work.rippleadder
run
# A  B Cin  Sum  Cout
#  4  6  0   10   0
# 11  5  0    0   1
#  9  3  0   12   0
#  2  1  0    3   0
#  5  3  0    8   0
```