

LAB NO 12



Fall 2024

CSE-304L Computer Organization and Architecture Lab

Submitted by:

Name : **Hassan Zaib Jadoon**

Reg no. : **22PWCSE2144**

Class Section : **A**

Signature: _____

Submitted to:

Dr. Amaad Khalil

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

ADDER AND ADDER SUBTRACTOR

TASK:1

Write a Verilog code for Full Adder using Dataflow Level modeling.

CODE:

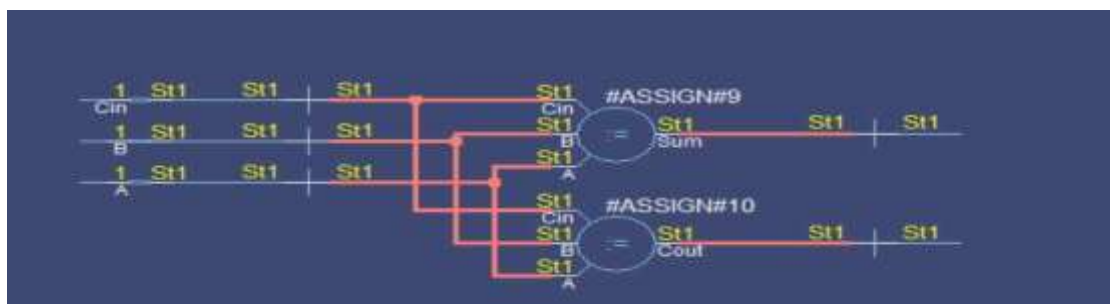
```
1 module FullAdder
2
3   input A; // First input
4   input B; // Second input
5   input Cin; // Carry input
6   output Sum; // Sum output
7   output Cout; // Carry output
8
9
10  // Dataflow modeling
11  assign Sum = A ^ B ^ Cin; // Sum is XOR of inputs
12  assign Cout = (A & B) | (B & Cin) | (A & Cin); // Carry out is derived from the majority function
13
14 endmodule
```

```
1 module stimFA();
2   reg A, B, Cin; // Inputs: A, B, and Carry-in
3   wire Sum, Cout; // Outputs: Sum and Carry-out
4
5   // Instantiate the full adder module
6   full_adder FA(A, B, Cin, Sum, Cout);
7
8   initial
9   begin
10    $display("A B Cin | Sum Cout");
11
12    // Test all input combinations
13    A = 0; B = 0; Cin = 0; #10
14    $display(" %b %b %b | %b %b", A, B, Cin, Sum, Cout);
15
16    A = 0; B = 0; Cin = 1; #10
17    $display(" %b %b %b | %b %b", A, B, Cin, Sum, Cout);
18
19    A = 0; B = 1; Cin = 0; #10
20    $display(" %b %b %b | %b %b", A, B, Cin, Sum, Cout);
21
22    A = 0; B = 1; Cin = 1; #10
23    $display(" %b %b %b | %b %b", A, B, Cin, Sum, Cout);
24
25    A = 1; B = 0; Cin = 0; #10
26    $display(" %b %b %b | %b %b", A, B, Cin, Sum, Cout);
27
28    A = 1; B = 0; Cin = 1; #10
29    $display(" %b %b %b | %b %b", A, B, Cin, Sum, Cout);
30
31    A = 1; B = 1; Cin = 0; #10
32    $display(" %b %b %b | %b %b", A, B, Cin, Sum, Cout);
33
34    A = 1; B = 1; Cin = 1; #10
```

OUTPUT:

Table output:

```
run -all
#A B Cin | Sum Cout
# 0 0 0 | 0 0
# 0 0 1 | 0 0
# 0 1 0 | 0 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
V$IM 25>
```



REMARKS:

The Verilog code for the Full Adder using Dataflow Level modeling is concise and clear, utilizing assign statements to directly express the relationships between inputs and outputs. It accurately implements the Full Adder logic by calculating the Sum as the XOR of inputs and the Cout using the OR of multiple AND operations. The code is compact, easy to understand, and correctly models the Full Adder's behavior. However, it can be improved in terms of adding comments for further clarification, especially for beginners, and ensuring the code adheres to specific naming conventions if required by a larger project.

TASK:2

Write a Verilog code for Half Adder using Dataflow Level modeling.

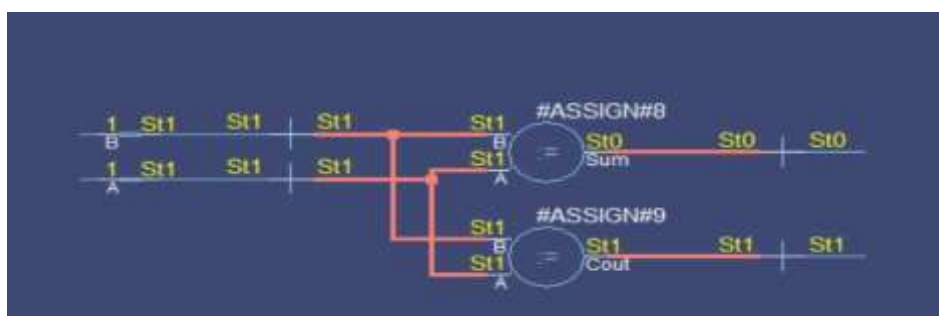
CODE:

```
1 module HalfAdder (
2     input A, // First input
3     input B, // Second input
4     output Sum, // Sum output
5     output Cout // Carry output
6 );
7
8 // Dataflow modeling
9 assign Sum = A ^ B; // Sum is XOR of inputs
10 assign Cout = A & B; // Carry is AND of inputs
11
12 endmodule
```

```
1 module stimHA();
2     reg A, B; // Inputs: A and B
3     wire Sum, Cout; // Outputs: Sum and Carry-out
4
5 // Instantiate the Half Adder module
6 half_adder HA(A, B, Sum, Cout);
7
8 initial
9 begin
10     $display("A B | Sum Cout");
11
12 // Test all possible input combinations
13 A = 0; B = 0; #10
14 $display("%b %b | %b %b", A, B, Sum, Cout);
15
16 A = 0; B = 1; #10
17 $display("%b %b | %b %b", A, B, Sum, Cout);
18
19 A = 1; B = 0; #10
20 $display("%b %b | %b %b", A, B, Sum, Cout);
21
22 A = 1; B = 1; #10
23 $display("%b %b | %b %b", A, B, Sum, Cout);
24 end
25 endmodule
```

OUTPUT:

```
run -all
#A B | Sum Cout
#0 0 | 0 0
#0 1 | 1 0
#1 0 | 1 0
#1 1 | 0 1
VSIM 30>
```



REMARKS:

The Verilog code for the Half Adder using Dataflow Level modeling is simple, clear, and correct. It utilizes assign statements to express the relationships between inputs and outputs, with the Sum calculated using the XOR operation ($A \oplus B$) and the Cout using the AND operation ($A \& B$). The code accurately models the behavior of a Half Adder, making it easy to understand and verify. However, for readability and completeness, it could include additional comments, especially for beginners, and adhere to consistent naming conventions if used in a larger design project.

TASK 3:

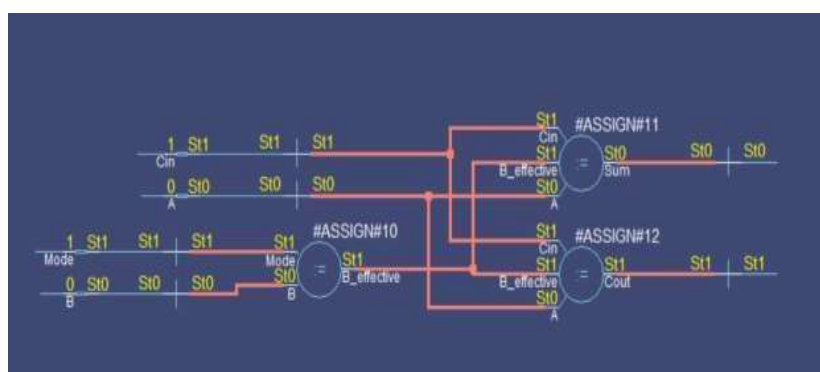
Write a Verilog code for Adder and Subtractor using Dataflow Level modeling.

Code:

```
1 module stick3_1 {
2     reg A, B, Mode, Cin; // Inputs: A, B, Mode, Carry-in
3     wire Sum, Cout; // Outputs: Sum/Difference, Carry-out
4
5     // Instantiate the Adder/Subtractor module
6     adder_subtractor A1(A, B, Mode, Cin, Sum, Cout);
7
8     initial
9     begin
10        $display("A B Mode Cin | Sum Cout");
11
12        // Addition Mode (Mode = 0)
13        A = 1; B = 0; Mode = 0; Cin = 0; $0;
14        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
15
16        A = 1; B = 0; Mode = 0; Cin = 0; $0;
17        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
18
19        A = 1; B = 0; Mode = 0; Cin = 0; $0;
20        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
21
22        // Subtraction Mode (Mode = 1)
23        A = 1; B = 0; Mode = 1; Cin = 0; $0;
24        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
25
26        A = 1; B = 0; Mode = 1; Cin = 0; $0;
27        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
28
29        A = 1; B = 0; Mode = 1; Cin = 0; $0;
30        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
31
32        A = 1; B = 0; Mode = 1; Cin = 0; $0;
33        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
34    end
35}
36
37 module stick3_2 {
38     reg A, B, Mode, Cin; // Inputs: A, B, Mode, Carry-in
39     wire Sum, Cout; // Outputs: Sum/Difference, Carry-out
40
41     // Instantiate the Adder/Subtractor module
42     adder_subtractor A1(A, B, Mode, Cin, Sum, Cout);
43
44     initial
45     begin
46        $display("A B Mode Cin | Sum Cout");
47
48        // Addition Mode (Mode = 0)
49        A = 1; B = 0; Mode = 0; Cin = 0; $0;
50        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
51
52        A = 1; B = 0; Mode = 0; Cin = 0; $0;
53        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
54
55        A = 1; B = 0; Mode = 0; Cin = 0; $0;
56        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
57
58        // Subtraction Mode (Mode = 1)
59        A = 1; B = 0; Mode = 1; Cin = 0; $0;
60        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
61
62        A = 1; B = 0; Mode = 1; Cin = 0; $0;
63        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
64
65        A = 1; B = 0; Mode = 1; Cin = 0; $0;
66        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
67
68        A = 1; B = 0; Mode = 1; Cin = 0; $0;
69        $display("A B Mode Cin | Sum Cout", A, B, Mode, Cin, Sum, Cout);
70    end
71}
72
73 test_adder_subtractor;
74
75 test_adder_subtractor;
76
```

Output:

Table:



```
run -all
# A B Mode Cin | Sum Cout
# 0 0 0 0 | 0 0
# 1 0 0 0 | 1 0
# 1 1 0 1 | 1 1
# 1 0 1 0 | 1 1
# 1 1 1 0 | 1 0
# 0 1 1 1 | 1 0
# 0 0 1 1 | 0 1
VSIM 20> |
```