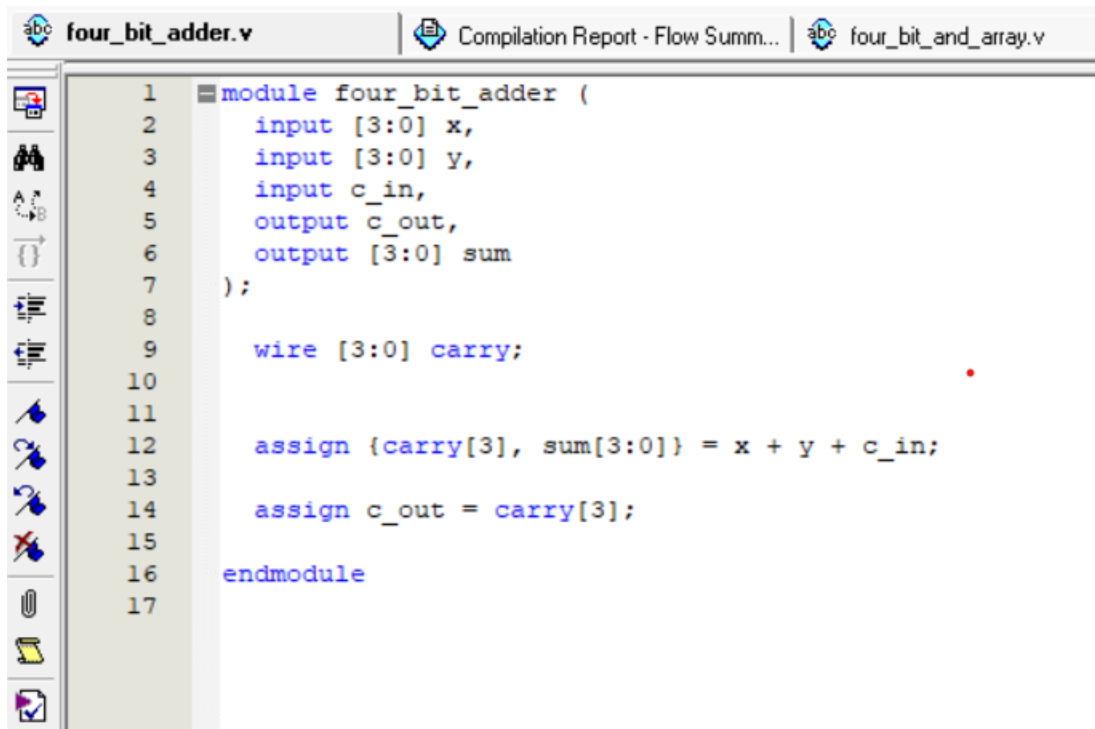


Name: Rasha Mansour

ID:1210773

8.7 Post Lab

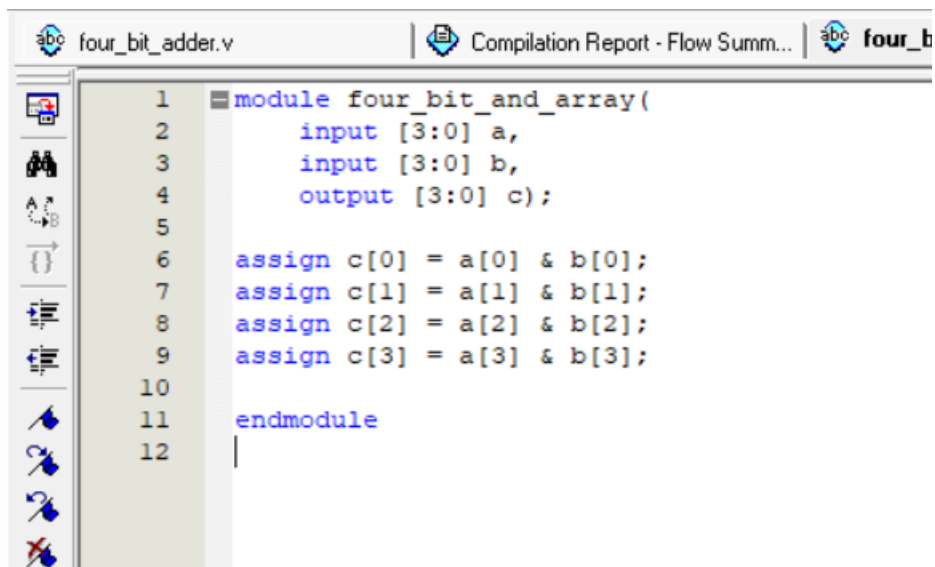
1-4-bit-adder



The screenshot shows a Verilog code editor with the file name 'four_bit_adder.v'. The code defines a module 'four_bit_adder' with three inputs: a 4-bit vector 'x', a 4-bit vector 'y', and a single-bit input 'c_in'. It has two outputs: a single-bit output 'c_out' and a 4-bit vector output 'sum'. The code includes a wire declaration for a 4-bit 'carry', an assignment for 'sum' and 'carry[3]' based on the addition of 'x', 'y', and 'c_in', and an assignment for 'c_out' based on 'carry[3]'. The module is closed with 'endmodule'.

```
1 module four_bit_adder (  
2     input [3:0] x,  
3     input [3:0] y,  
4     input c_in,  
5     output c_out,  
6     output [3:0] sum  
7 );  
8  
9     wire [3:0] carry;  
10  
11  
12     assign {carry[3], sum[3:0]} = x + y + c_in;  
13  
14     assign c_out = carry[3];  
15  
16 endmodule  
17
```

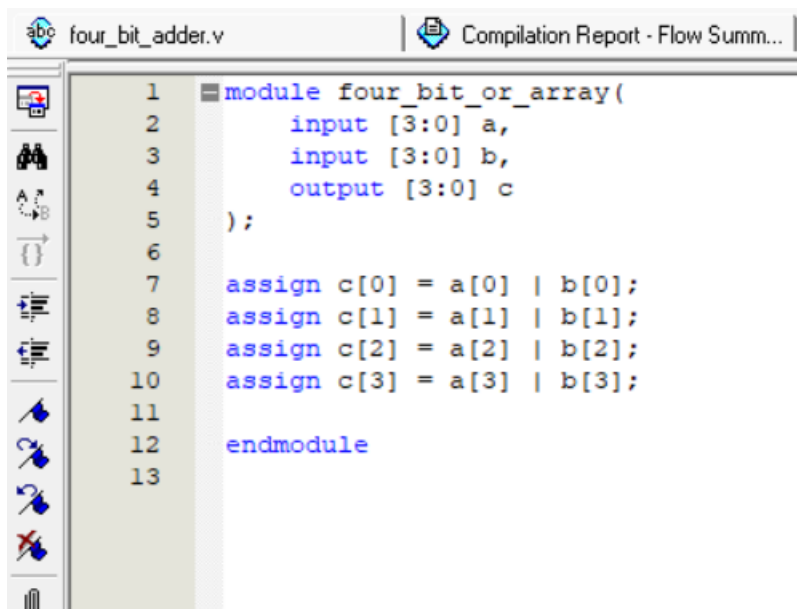
2-4-bit-and-array:



The screenshot shows a Verilog code editor with the file name 'four_bit_and_array.v'. The code defines a module 'four_bit_and_array' with two 4-bit inputs 'a' and 'b', and a 4-bit output 'c'. The code uses four 'assign' statements to perform a bitwise AND operation on each corresponding bit of 'a' and 'b' to produce the output 'c'. The module is closed with 'endmodule'.

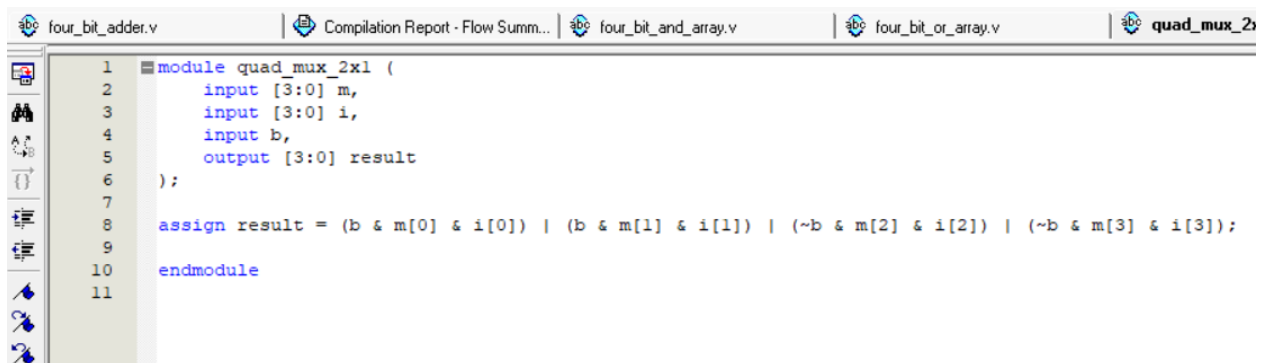
```
1 module four_bit_and_array(  
2     input [3:0] a,  
3     input [3:0] b,  
4     output [3:0] c);  
5  
6     assign c[0] = a[0] & b[0];  
7     assign c[1] = a[1] & b[1];  
8     assign c[2] = a[2] & b[2];  
9     assign c[3] = a[3] & b[3];  
10  
11 endmodule  
12
```

3-4-bit-or array:



```
1 module four_bit_or_array(  
2     input [3:0] a,  
3     input [3:0] b,  
4     output [3:0] c  
5 );  
6  
7 assign c[0] = a[0] | b[0];  
8 assign c[1] = a[1] | b[1];  
9 assign c[2] = a[2] | b[2];  
10 assign c[3] = a[3] | b[3];  
11  
12 endmodule  
13
```

4-quad-mux2x1:



```
1 module quad_mux_2x1 (  
2     input [3:0] m,  
3     input [3:0] i,  
4     input b,  
5     output [3:0] result  
6 );  
7  
8 assign result = (b & m[0] & i[0]) | (b & m[1] & i[1]) | (~b & m[2] & i[2]) | (~b & m[3] & i[3]);  
9  
10 endmodule  
11
```

The circuit:

