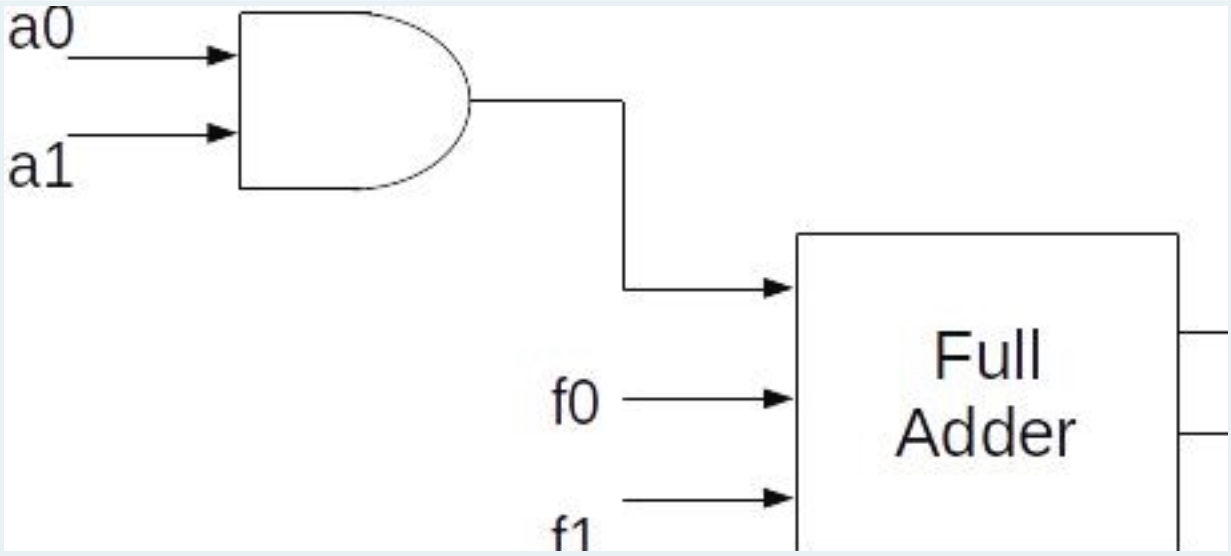
CS224_2021_quiz1_B

Points: 30/20

1

For the circuit diagram given below, write structural Verilog code. Write separate modules for the required basic gates. Full adder is also a separate module, you can write behavioural code for it.
Use these modules to build your complete design.
Add a constant delay of 2 units in the basic gate and 2 units for both sum and carry-out in full-adder.
Synthesise the design and check if RTL matches the given diagram.



Yes RTL Matches

2

Write the code of basic gate.
(2/2 Points)

```
module andgate(
   input a0,
   input a1,
   output out
   );
assign #2 out = a0&a1;

endmodule
```

3

Write the code for full adder.
(4/4 Points)

```
module full_adder(
   input a0,
   input a1,
   input a2,
   output sum,
   output car
   );

assign #2 sum = a0^a1^a2;
assign #2 car = a0&a1 | a1&a2 | a2&a0;
endmodule
```

4

Write the code for the circuit diagram showing instantiations of modules
(4/4 Points)

```
module final(
    input a0,
    input a1,
    input f0,
    input f1,
    output sum,
    output cout
    );
wire g;
andgate and1(a0, a1, g);
full_adder fa1(g, f0, f1, sum, cout);

endmodule
```

5

Write test-bench for the design to identify the delay taken by the circuit to give the final output sum and carry-out. It should demonstrate the output coming after 2 and 4 time units. Use monitor commands to print the output. Give appropriate test cases and paste the test-bench code below.
Your test-bench will have all signals 0 at time=0.
At time=10ns change the inputs to new values.
(2/2 Points)

```
module final_test_bench;

	// Inputs
	reg a0;
	reg a1;
	reg f0;
	reg f1;

	// Outputs
	wire sum;
	wire cout;

	// Instantiate the Unit Under Test (UUT)
	final uut (
		.a0(a0),
		.a1(a1),
		.f0(f0),
		.f1(f1),
		.sum(sum),
		.cout(cout)
	);
	always@*
	$monitor("time : ", $time , " a0 = %b , a1 = %b, f0 = %b , f1 = %b , sum = %b, cout = %b "
,a0, a1, f0, f1, sum, cout) ;

	initial begin
		// Initialize Inputs
		a0 = 0;
		a1 = 0;
		f0 = 0;
		f1 = 0;
		//2 unit delay
		#10;
      a0 = 0;
		a1 = 1;
		f0 = 0;
		f1 = 1;

		//4 unit delay
		#10;
      a0 = 1;
		a1 = 1;
		f0 = 0;
		f1 = 1;

	end

endmodule
```

6

Generate the waveform and output of monitor.
Paste the output of monitor command showing delay of 2 units to generate final output.
(4/4 Points)

```
time :                0 a0 = 0 , a1 = 0, f0 = 0 , f1 = 0 , sum = x, cout = x
time :                2 a0 = 0 , a1 = 0, f0 = 0 , f1 = 0 , sum = x, cout = 0
time :                4 a0 = 0 , a1 = 0, f0 = 0 , f1 = 0 , sum = 0, cout = 0
time :               10 a0 = 0 , a1 = 1, f0 = 0 , f1 = 1 , sum = 0, cout = 0
time :               12 a0 = 0 , a1 = 1, f0 = 0 , f1 = 1 , sum = 1, cout = 0
time :               20 a0 = 1 , a1 = 1, f0 = 0 , f1 = 1 , sum = 1, cout = 0
time :               24 a0 = 1 , a1 = 1, f0 = 0 , f1 = 1 , sum = 0, cout = 1


lines 4 and 5 show a 2s delay
```

7

Generate the waveform and output of monitor.
Paste the output of monitor command showing delay of 4 units to generate final output.
(4/4 Points)

```
time :                0 a0 = 0 , a1 = 0, f0 = 0 , f1 = 0 , sum = x, cout = x
time :                2 a0 = 0 , a1 = 0, f0 = 0 , f1 = 0 , sum = x, cout = 0
time :                4 a0 = 0 , a1 = 0, f0 = 0 , f1 = 0 , sum = 0, cout = 0
time :               10 a0 = 0 , a1 = 1, f0 = 0 , f1 = 1 , sum = 0, cout = 0
time :               12 a0 = 0 , a1 = 1, f0 = 0 , f1 = 1 , sum = 1, cout = 0
time :               20 a0 = 1 , a1 = 1, f0 = 0 , f1 = 1 , sum = 1, cout = 0
time :               24 a0 = 1 , a1 = 1, f0 = 0 , f1 = 1 , sum = 0, cout = 1

LAST 2 lines show a 4s delay
```