



San Francisco Bay University EE461L - Verilog HDL Lab Assignment 1

Week#1 Introduction to Verilog and Combinational & Sequential Logic Design

Example “Hello World” & Simulation

- Directly run the module `helloWorld` in the online compiler

The screenshot displays the EDA Playground interface. The top navigation bar includes 'New', 'Run', 'Save', and 'Copy' buttons, along with a 'KnowHow WEBINARS' banner and a 'Deep Learning with FPGAs' promotion. The left sidebar contains a 'Languages & Libraries' section with 'Testbench + Design' and 'Tools & Simulators' tabs. The main workspace is divided into two panels: 'testbench.v' and 'design.v'. The 'testbench.v' panel contains the following Verilog code:

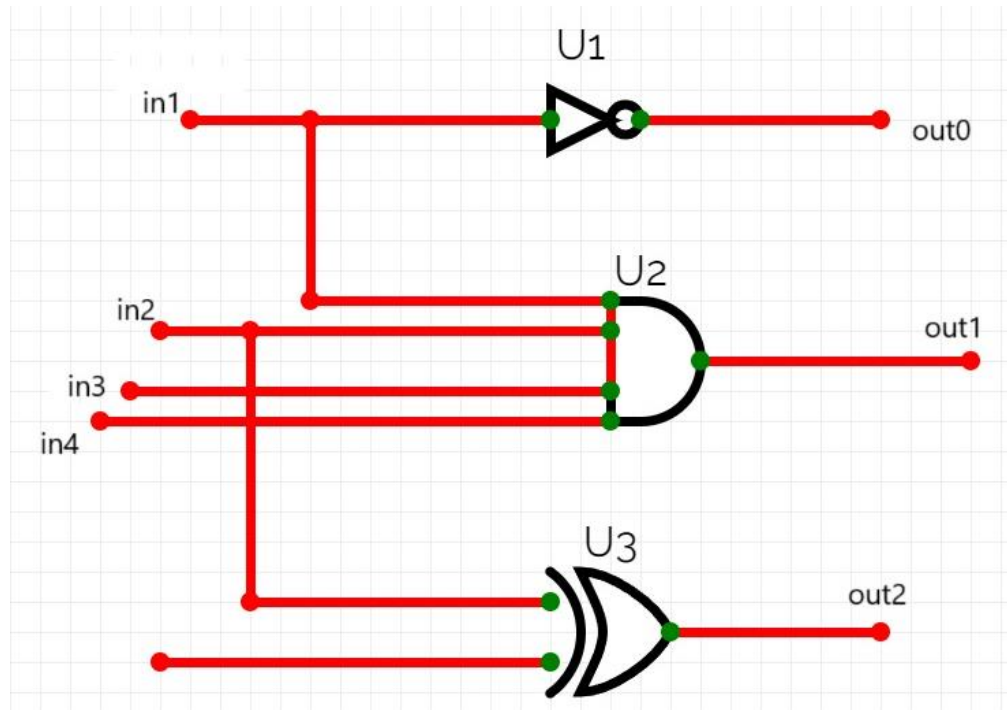
```
1 // Anika Haque, 20283
2 // EE 461L, Lab Assignment 1
3 //
4 // This is my first Verilog Program
5 // Design Name : hello_world
6 // File Name : hello_world.v
7 // Function : This program will print 'hello world'
8 //
9 module helloworld;
10     initial begin
11         $display ("Hello world!!!");
12         #10 $finish;
13     end
14 endmodule // End of Module helloworld
15
```

The 'design.v' panel is currently empty, showing a placeholder for code. Below the code panels, the 'Log' tab is active, displaying the simulation output:

```
[2024-09-26 06:47:12 UTC] iverilog '-wall' '-g2012' design.v testbench.v && unbuffer vvp a.out
Hello world!!!
testbench.v:12: $finish called at 10 (1s)
Done
```

Based on the module `gates`, draw the digital circuit schematic.

```
module gates();  
  wire out0;  
  wire out1;  
  wire out2;  
  reg  in1,in2,in3,in4;  
  
  not U1(out0,in1);  
  and U2(out1,in1,in2,in3,in4);  
  xor U3(out2,in1,in2,in3);  
endmodule
```



Combinational & Sequential Logic in RTL

- Create the testbench first, and then simulate the design modules *oneBitFA1*, *oneBitFA2*, *DFFSynch*, and *DFFAsynch*. *Note: Ignore creating the testbenches for module *DFFSynch*, and *DFFAsynch*

Combinational Logic in RTL

1. “continuous assignment” in oneBitFA1.v

```
module oneBitFA1(
    input wire a_i,
    input wire b_i,
    input wire ci_i,
    output wire sum_o,
    output wire co_o
);
    assign {co_o,sum_o} = a_i + b_i + ci_i;
endmodule
```

The screenshot displays the EDA Playground interface. The left sidebar contains project settings for 'testbench.v' and 'design.v'. The central code editor shows the testbench code for 'tb_oneBitFA1()' and the design code for 'oneBitFA1' module. The bottom log window shows the simulation results, including the initial values of inputs and outputs, and the final state of the design after simulation.

Testbench Code (testbench.v):

```
1 // Anika Haque, 20283
2 // EE 461L, Lab Assignment 1, Q3
3 // =====
4 module tb_oneBitFA1(); // Testbench, no inputs or outputs
5 // Declare testbench variables (registers)
6 reg a_i;
7 reg b_i;
8 reg ci_i;
9
10 // Declare wires for design output
11 wire sum_o;
12 wire co_o;
13
14 // Instantiate DUT
15 oneBitFA1 dut (
16     .a_i(a_i),
17     .b_i(b_i),
18     .ci_i(ci_i),
19     .sum_o(sum_o),
20     .co_o(co_o)
21 );
22
23 // Testbench initial block to apply inputs
```

Design Code (design.v):

```
1 // Code your design here
2 module oneBitFA1(
3     input wire a_i,
4     input wire b_i,
5     input wire ci_i,
6     output wire sum_o,
7     output wire co_o
8 );
9     assign {co_o, sum_o} = a_i + b_i + ci_i; // Continuous assignment for full adder
10 endmodule
```

Simulation Log:

```
[2024-09-26 07:40:24 UTC] iverilog design.v testbench.v && unbuffer vvp a.out
a_i b_i ci_i | sum_o co_o
0 0 0 | 0 0
0 0 1 | 1 0
0 1 0 | 1 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
testbench.v:39: $finish called at 80 (1s)
```

2. “always block” in oneBitFA2.v

```

module oneBitFA2(
    input wire a_i,
    input wire b_i,
    input wire ci_i,
    output reg sum_o,
    output reg co_o
);

always @(a_i, b_i, ci_i)begin
    {co_o, sum_o} = a_i + b_i + ci_i;
end
endmodule

```

The screenshot shows the EDA Playground interface with the following components:

- Left Sidebar:**
 - Languages & Libraries:** SystemVerilog/Verilog, UVM / OVM, Other Libraries (None, CIVL, SVUnit).
 - Tools & Simulators:** Icarus Verilog 12.0.
 - Compile Options:** Compile Options.
 - Run Options:** Run Options.
 - Examples:** using EDA Playground, VHDL, Verilog.
- Main Editor:**
 - testbench.v:**

```

1 module tb_oneBitFA2(); // Testbench module has no inputs or outputs
2 // Declare testbench variables - registers for inputs
3 reg a_i;
4 reg b_i;
5 reg ci_i;
6
7 // Declare wires for outputs
8 wire sum_o;
9 wire co_o;
10
11 // Instantiate DUT
12 oneBitFA2 dut (
13     .a_i(a_i),
14     .b_i(b_i),
15     .ci_i(ci_i),
16     .sum_o(sum_o),
17     .co_o(co_o)
18 );
19
20 // Testbench initial block to apply inputs
21 initial begin
22     // Display headers for output
23     $display("a_i b_i ci_i | sum_o co_o");
24     $monitor("%b %b %b | %b %b", a_i, b_i, ci_i, sum_o, co_o);
25
26     // Apply test vectors
27     a_i = 0; b_i = 0; ci_i = 0; #10; // Wait 10 time units
28     a_i = 0; b_i = 0; ci_i = 1; #10;

```
 - design.v:**

```

1 module oneBitFA2(
2     input wire a_i,
3     input wire b_i,
4     input wire ci_i,
5     output reg sum_o,
6     output reg co_o
7 );
8
9 always @(a_i, b_i, ci_i)begin
10     {co_o, sum_o} = a_i + b_i + ci_i;
11 end
12 endmodule
13
14

```
- Output Console:**

```

[2024-09-26 07:54:49 UTC] {verilog design.v testbench.v && unbuffer vvp a.out
a_i b_i ci_i | sum_o co_o
0 0 0 | 0 0
0 0 1 | 1 0
0 1 0 | 1 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
testbench.v:39: $finish called at 80 (1s)

```

Sequential Logic in RTL

1. Synchronous DFF:

```

module DFFSynch(
    d_i,
    rst_i,
    clk_i,
    q_o
);
    input d_i,rst_i,clk_i;
    output q_o;

    reg q_o;

    always @(posedge clk_i)begin
        if(rst_i) q_o <= 0;
        else q_o <= d_i;
    end
endmodule

```

The screenshot shows the EDA Playground interface. On the left, there's a sidebar with 'Languages & Libraries' and 'Tools & Simulators'. The main area displays a Verilog testbench for the DFF module. The testbench includes a stimulus for the clock, reset, and data inputs, and instantiates the DFF module. The simulation results are shown in a table below the code.

Testbench Code:

```

0      always @(posedge clk_i) begin
9          if (rst_i)
10             q_o <= 0;
11         else
12             q_o <= d_i;
13     end
14 endmodule
15
16 // Include stimulus directly inside the file
17 module test_DFFSynch();
18     reg d_i, rst_i, clk_i;
19     wire q_o;
20
21     // Instantiate DFFSynch module
22     DFFSynch uut (
23         .d_i(d_i),
24         .rst_i(rst_i),
25         .clk_i(clk_i),
26         q_o(q_o)
27     );
28 endmodule

```

Simulation Results:

Time	clk_i	rst_i	d_i	q_o
0	0	1	0	x
5	1	1	0	0
10	0	0	0	0
15	1	0	0	0
20	0	0	1	0
25	1	0	1	1
30	0	0	0	1
35	1	0	0	0
40	0	0	1	0
45	1	0	1	1
50	0	1	1	1
55	1	1	1	0
60	0	0	1	0
65	1	0	1	1
70	0	0	1	1
75	1	0	1	1
80	0	0	1	1
85	1	0	1	1
90	0	0	1	1
95	1	0	1	1
100	0	0	1	1
105	1	0	1	1
110	0	0	1	1

2. Asynchronous DFF:

```

module DFFAsynch(
    d_i,
    rst_i,
    clk_i,
    q_o
);
    input d_i,rst_i,clk_i;
    output q_o;

    reg q_o;

    always @(posedge clk_i or posedge rst_i)begin
        if(rst_i) q_o <= 0;
        else q_o <= d_i;
    end
endmodule

```

The screenshot shows the EDA Playground interface. On the left, there are navigation tabs for Languages & Libraries, Tools & Simulators, and Examples. The main area displays a Verilog testbench for the DFFAsynch module. The testbench includes a clock generation, stimulus initialization, asynchronous reset, and test stimulus application. The simulation results show the output q_o following the input d_i and reset rst_i signals.

Testbench Code:

```

30 always #5 clk_i = ~clk_i; // Generate a clock with a period of 10 time units
31
32 // Stimulus
33 initial begin
34     // Initialize signals
35     clk_i = 0;
36     rst_i = 1;
37     d_i = 0;
38
39     // Asynchronous reset
40     #10 rst_i = 0; // Release reset after 10 time units
41
42     // Apply test stimulus
43     #10 d_i = 1; // Set d_i to 1
44     #10 d_i = 0; // Set d_i back to 0
45     #10 d_i = 1; // Set d_i to 1
46     #10 rst_i = 1; // Apply asynchronous reset, should reset q_o immediately
47     #10 rst_i = 0; // Release reset again
48
49     // Finish simulation
50     #50 $finish;

```

Simulation Results:

```

[2024-09-26 08:20:14 UTC] iverilog '-wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out
Time: 0 | clk_i: 0 | rst_i: 1 | d_i: 0 | q_o: 0
Time: 5 | clk_i: 1 | rst_i: 1 | d_i: 0 | q_o: 0
Time: 10 | clk_i: 0 | rst_i: 0 | d_i: 0 | q_o: 0
Time: 15 | clk_i: 1 | rst_i: 0 | d_i: 0 | q_o: 0
Time: 20 | clk_i: 0 | rst_i: 0 | d_i: 1 | q_o: 0
Time: 25 | clk_i: 1 | rst_i: 0 | d_i: 1 | q_o: 1
Time: 30 | clk_i: 0 | rst_i: 0 | d_i: 0 | q_o: 1
Time: 35 | clk_i: 1 | rst_i: 0 | d_i: 0 | q_o: 0
Time: 40 | clk_i: 0 | rst_i: 0 | d_i: 1 | q_o: 0
Time: 45 | clk_i: 1 | rst_i: 0 | d_i: 1 | q_o: 1
Time: 50 | clk_i: 0 | rst_i: 1 | d_i: 1 | q_o: 0
Time: 55 | clk_i: 1 | rst_i: 1 | d_i: 1 | q_o: 0
Time: 60 | clk_i: 0 | rst_i: 0 | d_i: 1 | q_o: 0
Time: 65 | clk_i: 1 | rst_i: 0 | d_i: 1 | q_o: 1
Time: 70 | clk_i: 0 | rst_i: 0 | d_i: 1 | q_o: 1
Time: 75 | clk_i: 1 | rst_i: 0 | d_i: 1 | q_o: 1
Time: 80 | clk_i: 0 | rst_i: 0 | d_i: 1 | q_o: 1
Time: 85 | clk_i: 1 | rst_i: 0 | d_i: 1 | q_o: 1
Time: 90 | clk_i: 0 | rst_i: 0 | d_i: 1 | q_o: 1
Time: 95 | clk_i: 1 | rst_i: 0 | d_i: 1 | q_o: 1
Time: 100 | clk_i: 0 | rst_i: 0 | d_i: 1 | q_o: 1
Time: 105 | clk_i: 1 | rst_i: 0 | d_i: 1 | q_o: 1
testbench.sv:50: $finish called at 110 (1s)
Time: 110 | clk_i: 0 | rst_i: 0 | d_i: 1 | q_o: 1

```