**Logo

Description automatically generated San Francisco Bay University**

**EE461 Digital Design and HDL**

**Week#5 Sequential Logic**

**1. Lab Outlines:**

1. DFF & Latch modeling in Verilog
2. Sequential logic feedback
3. Not glue logic in the top level
4. Race condition again
5. Exercises

**2. Lab Procedures**

**I. DFF & Latch modeling in Verilog**

*`timescale 1ns/100ps*

*module xor3(input D\_i, clock\_i, output Q\_o);*

*reg Q\_o;*

*always@(clock\_i or D\_i)*

*if (clock\_i) Q \_o<=D\_i;*

*endmodule*

*`timescale 1ns/100ps*

*module xor3 (input a\_i, b\_i, c\_i, clock\_i, output Q\_o);*

*reg Q\_o;*

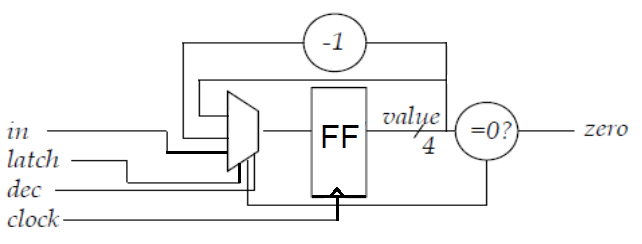
*always@(posedge clock\_i)*

*if (a\_i) Q\_o< = c\_i;*

*else Q\_o<= b\_i;*

*endmodule*

**II. Sequential logic feedback**



*/\* If latch is high, value will get 'in'; if dec is high and zero isn't 0, value is to decrease 1; if value is 0, zero is set to 1.*

*\*/*

*module counter (clock, in, latch, dec, zero);*

*input clock; /\* clock \*/*

*input [3:0] in; /\* starting count \*/*

*input latch; /\* latch `in’ when high \*/*

*input dec; /\* decrement count when dec high \*/*

*output zero; /\* high when count down to zero \*/*

*reg [3:0] value; /\* current count value \*/*

*always@(posedge clock) begin*

*if (latch) value <= in;*

*else if (dec && !zero) value <= value - 1’b1;*

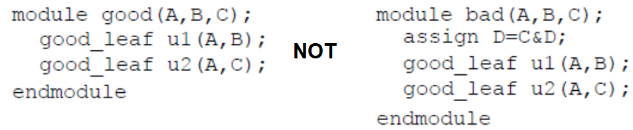
*end*

*assign zero = (value == 4’b0);*

*endmodule /\* counter \*/*

1. **Not glue logic in the top level**

* Generally, it is a good idea to only implement logic in the leaf cells of a hierarchical design, and not at a higher level.



**IV. Race condition Again**

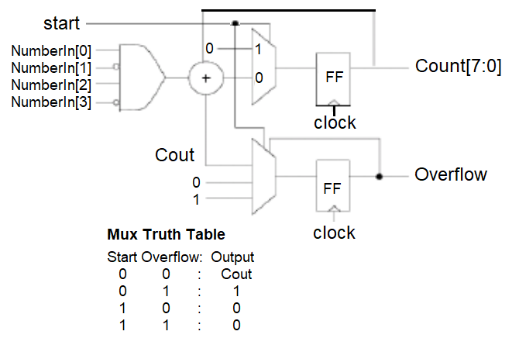
* Guideline 1: When modeling sequential logic, use nonblocking assignments.
* Guideline 2: When modeling combinational logic with an "always" block, use

blocking assignments.

* Guideline 3: Do not mix blocking and nonblocking assignments in the same "always" block.
* Guideline 4: Do not make assignments to the same variable from more than one "always" block.
* Guideline 5: Do not make assignments using #0 delays.

**V Exercises**

* Convert the following circuit into Verilog module and write the testbench to verify the design.



**Answer:**

**Design :**

module mux\_ff (input clk,input load,input [7:0] inp,output [7:0] out );

reg [7:0] mux\_ff;

always @(posedge clk)

begin

if (load)

begin

mux\_ff <= inp;

end else begin

mux\_ff <= {mux\_ff[6:0], mux\_ff[7]};

end

end

assign out = mux\_ff;

endmodule

**Testbench :**

module tb;

reg clk;

reg load;

reg [7:0] inp;

wire [7:0] out;

mux\_ff prob\_1 (.clk(clk),.load(load),.inp(inp),.out(out)

);

initial begin

clk = 0;

load = 0;

inp = 0;

#5

load = 1;

#5

inp = 8'hFF;

#5

load = 0;

#5

inp = 8'h00;

#5

inp = 8'h01;

#5

inp= 8'h02;

#5

$finish;

end

always #5 clk = ~clk;

initial begin

$dumpfile("tb.vcd");

$dumpvars();

#10;

$display("Time out");

repeat (50) begin

#10

$display("%d %b", $time, out);

end

end

endmodule

* What does logic statement specify for the hardware in the module?

*`timescale 1ns/100ps*

*module xor3 (input B\_i, D\_i, sel\_i, clock, output E\_o);*

*reg E\_o;*

*always@(posedge clock) begin*

*case(sel\_i)*

*0: E\_o <= D\_i + B\_i;*

*1: E\_o <= B\_i;*

*endcase*

*end*

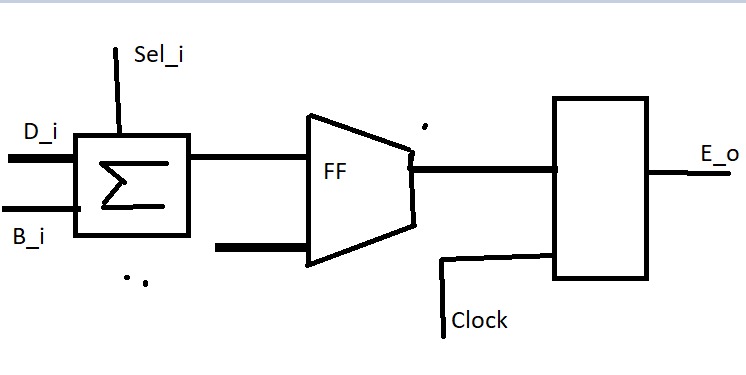
*endmodule*

**Answer:**

Here, timescale used for to sets the simulation time units to 1 nanosecond and time precision to 100 picoseconds. It’s behave like a XOR 3 gate. There 3 input *B\_i, D\_i, sel\_i and* Output is E\_o.

In this condition, E\_o difine as register, when sel\_i = 0, E\_o = B\_i + D\_i . again when sel\_i = 1,

E\_o = B\_i. The always block with the posedge clock trigger defines that the output E\_o is updated on every positive edge of the clock signal.



* Modify design in **Lab Procedures II** so it is decremented by *two,* and then stops, setting zero flag high, when it reaches 0000 or 0001. After that, verify the design in the testbench.

**Answer:**

**Design:**

module counter (clock, inp, latch, dec, zero);

input clock;

input [3:0] inp;

input latch;

input dec;

output zero;

reg [3:0] value;

always@(posedge clock) begin

if (latch) value <= inp;

else if (dec && !zero) value <= value - 2;

end

assign zero = ((value == 4'b0) | (value == 4'b0001));

endmodule

**TestBench:**

module tb;

reg clock, latch, dec;

reg [3:0] inp;

wire zero;

counter prob\_3 (.clock(clock),.inp(inp),.latch(latch),.dec(dec),.zero(zero));

always

#2

clock = ~clock;

initial begin

$dumpfile("tb.vcd");

$dumpvars;

clock = 0;

inp = 4'b1111;

latch = 1'b0;

dec = 1'b0;

#8;

latch = 1'b1;

#40;

inp = 4'b0101;

latch = 1'b0;

#10;

dec = 1'b1;

#20;

dec = 1'b0;

end

always @(posedge clock) begin

$display("Count: %d, Zero: %b", prob\_3.value, zero);

#10

$finish();

end

endmodule