**Logo

Description automatically generated San Francisco Bay University**

**EE461 Verilog-HDL**

**Homework #3**

**Due day: 3/7/2023**

**Name: Simon Gezae ID: 19830**

**Instruction:**

1. **Push answer sheets/source code to Github**
2. **Please follow the code style rule like programs on handout.**
3. **Overdue homework submission could not be accepted.**
4. **Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**
5. Design a static logic gate f =in Verilog modeling by PMOS/NMOS devices based on the logic schematic as follows. And then write the testbench to verify the design to cover all input combinations and some x/z.

Diagram, schematic

Description automatically generated

*module testbench;*

*reg a\_r;*

*reg b\_r;*

*reg c\_r;*

*wire f\_w;*

*designModule u(*

*.a\_i (a\_r),*

*.ab\_i (b\_r),*

*.c\_i (c\_r),*

*.f\_o (f\_w)*

*);*

*initial begin*

*a\_r = 1’b0; b\_r = 1’b0; c\_r = 1’b0;*

*#5 a\_r = ... ... ; b\_r = ... ... ; c\_r = ... ... ;*

*#5 ... ...*

*#5 $monitor(... ...);*

*#5 $finish;*

*end*

*endmodule*

*Ans:*

*Graphical user interface, table

Description automatically generated*

*//Design module:*

*module question1(input a\_r, b\_r, c\_r,*

*output f\_w);*

*supply1 vdd;*

*supply0 gnd;*

*wire abp\_w;*

*wire abn\_w;*

*pmos p1(abp\_w, vdd, a\_r);*

*pmos p2(f\_w, abp\_w, b\_r);*

*pmos p3(f\_w, vdd, c\_r);*

*nmos n1(abn\_w, gnd, a\_r);*

*nmos n2(abn\_w, gnd, b\_r);*

*nmos n3(f\_w, abn\_w, c\_r);*

*endmodule*

*//testbench*

*module tb;*

*reg a\_r;*

*reg b\_r;*

*reg c\_r;*

*wire f\_w;*

*integer i;*

*question1 u1(*

*.a\_r(a\_r),*

*.b\_r(b\_r),*

*.c\_r(c\_r),*

*.f\_w(f\_w)*

*);*

*initial begin*

*a\_r = 1'b0; b\_r = 1'b0; c\_r = 1'b0;*

*#2 $display("A\tB\tC\t!((A+B)C)");*

*#2 $monitor("%d\t%d\t%d\t %d", a\_r, b\_r, c\_r, f\_w);*

*for (i=0; i<8; i=i+1)begin*

*{c\_r, b\_r, a\_r} = i;*

*#2;*

*end*

*#2 a\_r = 1'bx; b\_r = 1'b0; c\_r = 1'b1;*

*#2 a\_r = 1'b1; b\_r = 1'b0; c\_r = 1'bx;*

*#2 a\_r = 1'b0; b\_r = 1'b1; c\_r = 1'bx;*

*#2 a\_r = 1'bz; b\_r = 1'b0; c\_r = 1'b1;*

*#2 a\_r = 1'b0; b\_r = 1'b1; c\_r = 1'bz;*

*#5 $finish;*

*end*

*endmodule*

*Eda link:* [*https://edaplayground.com/x/WfKr*](https://edaplayground.com/x/WfKr)

1. Design a 2-to-1 mux UDP, and write testbench to assign selection bit to 0, 1 and X/Z to verify you design.

A picture containing text, antenna, gauge

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

primitive question2(f\_o, a, b, sel);

output f\_o;

input a, b, sel;

table

//A B sel : f\_o

0 ? 0 : 0;

1 ? 0 : 1;

? 0 1 : 0;

? 1 1 : 1;

endtable

endprimitive

//testbench

module tb;

reg a,b,sel;

wire f\_o;

question2 u1 (f\_o, a, b, sel);

initial begin

a = 0; b=0; sel=0;

$monitor("A = %b B = %b Sel = %b out = %b", a,b,sel,f\_o);

#2 a = 1;

#2 sel= 1;

#2 b=1; a=0;

#2 sel=1'bx;

#2 a=1; b=0;

#2 sel=1'bz;

#2 b=1; a=0;

#2 sel= 1;

#2 b=1'bx; a=0;

#2 $finish;

end

endmodule

EDA link: <https://edaplayground.com/x/tFJV>

1. Create top module to instantiate mux UDPs from the above primitive and system gates with delay time as below and simulate it by testbench with only several inputs values. What is the delay time of longest path from inputs to output in hand calculation if we don't consider the delay from the selection bit to output in two muxes? If *D* is either *1* or *0*, what is the possible delay time of longest path from the inputs to output?

Diagram, schematic

Description automatically generated

Ans:

Delay time from A and B to out: 5+8+5+8 = 26

Delay time from C to out: 3+8+5+8 = 24

The longest path from the input to output is 26

primitive udpMux(f\_o, a, b, sel);

output f\_o;

input a, b, sel;

table

//A B sel : f\_o

0 ? 0 : 0;

1 ? 0 : 1;

? 0 1 : 0;

? 1 1 : 1;

endtable

endprimitive

primitive udpOR(out, a, b);

output out;

input a, b;

table

//A B : out

1 ? : 1;

? 1 : 1;

0 0 : 0;

endtable

endprimitive

primitive udpnot(out, a);

output out;

input a;

table

//A : out

1 : 0;

0 : 1;

endtable

endprimitive

module tb(out, a, b, c, d);

output out;

input a, b, d, c;

integer i;

wire or1, n1, muxo, muxor, muxon, dn;

#5 udpOR u1 (or1, a, b);

#3 udpnot u2(n1, c);

#8 udpMux u3 (muxo, or1, n1, d);

#5 udpOR u4 (muxor, muxo, muxo);

#3 udpnot u5 (muxon, muxo);

#3 udpnot u6 (dn, d);

#8 udpMux u7 (out, muxor, muxon, dn);

initial begin

$monitor("A=%b, B=%b, C=%b, D=%b, out=%b", a, b, c, d)

#2 for(i=0; i<16; i=i+1)begin

{d,c,b,a} = i;

#2

end

end

endmodule

1. Fix the bugs in the following example primitive, and explain why.

*#include “basicPrimitive.v”*

*primitive example(a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r);*

*output[1:0] a, r;*

*input b, d, e, f, g, h, i, j, k, l, m, n, o, p, q;*

*inout c;*

*reg[1:0] a, r;*

*table*

*//a: c: r: b d e f g h i j k l m n o p q*

*00: 1: 01: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0*

*1: 1: 11: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 ?*

*00: 0: 01: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0*

*... ...*

*endtable*

*basicPrimitive u0(a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r);*

*endprimitive*

*Ans:*

*According to the lecture note all UDP ports are scalar so output a and r should not be vector. And UDP should have one output and up to ten input in this program we have two outputs. Bidirectional ports are not permitted so port c should be reg or wire instead of inout. # is used to indicate delay if we want to include anything we need to use `.*

*primitive example(a, b, c, d, e, f, g, h, i, j, r);*

*output a;*

*input b, c, d, e, f, g, h, i, j, r;*

*reg a;*

*table*

*b c d e f g h i j r : a*

*0 0 0 1 0 1 0 1 0 1 : 0*

*0 0 0 1 0 1 0 1 0 1 : 1*

*0 0 0 1 0 1 0 1 0 1 : 0*

*... ...*

*Endtable*

*endprimitive*

1. Generate UDP of D-Flip Flop with reset, and then build 3-bits counter in top module by instantiating UDP of DFF and NOT/NAND/XOR gates in terms of following schematic. After that, write testbench to observe output results.

Diagram, engineering drawing

Description automatically generated

Ans:

primitive dffudp(

output reg q,

input d,

input clk,

input rst

);

table

// d clk rst: q: q+;

? ? 0 : ?: 0;

? ? (01): ?: -;

0 (01) 1: ?: 0;

1 (01) 1: ?: 1;

? (x1) 1: ?: -;

? (10) 1: ?: -;

? (1x) 1: ?: -;

(??) ? 1: ?: -;

endtable

endprimitive

primitive udpNand(f, a, b);

output f;

input a;

input b;

table

// a b : f;

1 1 : 0;

0 ? : 1;

? 0 : 1;

endtable

endprimitive

primitive udpNot(f, a);

output f\_o;

input a\_i;

table

// a : f;

1 : 0;

0 : 1;

endtable

endprimitive

primitive udpXor(f, a, b);

output f;

input a;

input b;

table

// a b : f;

1 1 : 0;

0 1 : 1;

1 0 : 1;

0 0 : 0;

endtable

endprimitive

module q5top(

output[2:0] out,

input clk,

input rst

);

wire d\_1, d\_2, nand\_w, rst\_b;

udpXor xor1(d\_1, out[1], out[0]);

dffudp dff1(.q(out[1]), .d(d\_1), .clk(clk), .rst\_b(rst\_b));

udpNand nand1(f(nand\_w), .a(out[1]), .b(out[0]));

udpXor xor\_2(

.f(d\_2),

.a(!out[2]),

.b(nand\_w)

);

dffudp dff0(

.q(out[0]),

.d(!out[0]),

.clk(clk),

.rst\_b(rst\_b)

);

udpNot not1(

.a(rst),

.f(rst\_b)

);

dffudp dff2(.q(out[2]), .d(d\_2), .clk(clk), .rst\_b(rst\_b));

endmodule

`include "q5top.sv"

module q5\_tb;

reg clk, rst;

wire[2:0] out;

q5\_top u1(

.clk(clk),

.rst(rst),

.out(out)

);

always #2 clk = ~clk;

initial begin

$dumpfile("wave.vcd");

$dumpvars;

clk = 1; rst = 1;

#3 rst = 0;

#20 rst = 1;

#4 rst = 0;

#20 $finish;

end

endmodule