**Logo

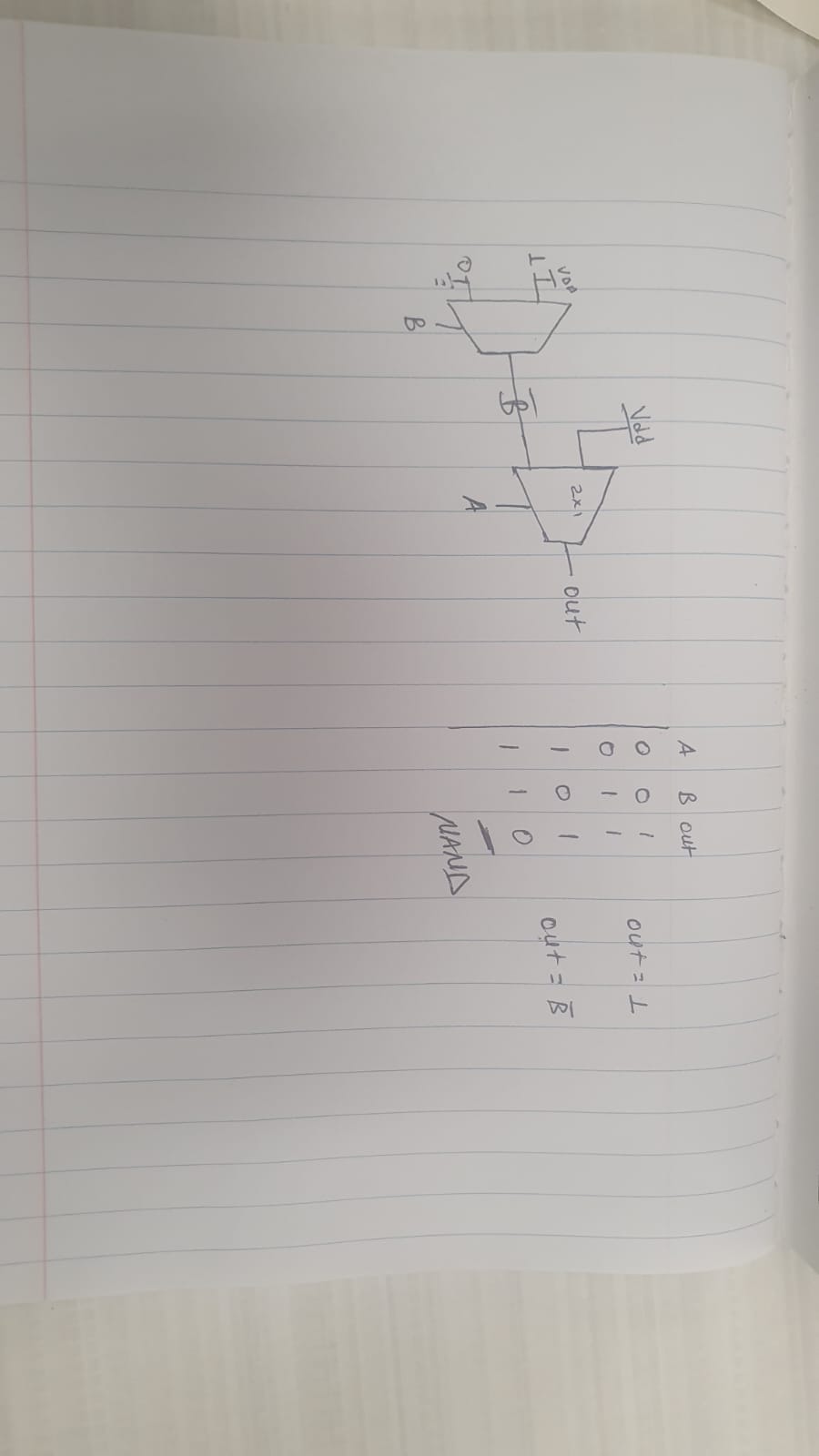
Description automatically generated San Francisco Bay University**

**EE461L - Digital Design and HDL Lab**

**2023 Spring Midterm Exam**

**Student Name: Simon Student ID:19830**

1. Draw logic schematic to implement NAND gate function ONLY using 2:1 MUXes.



1. Generate a Verilog RTL module to swap contents of two registers without a temporary register.

Ans:

module question2 ();

reg reg1;

reg reg2;

reg clk;

always@(posedge clk)begin

reg1 <= reg2;

reg2 <= reg1;

end

always #5 clk = ~clk;

initial begin

clk <= 0;

reg1 <= 1;

reg2 <= 0;

#2 $display("Time = %g, reg1 = %b reg2 = %b", $time, reg1, reg2);

#6 $display("Time = %g, reg1 = %b reg2 = %b", $time, reg1, reg2);

#2 $finish();

end

endmodule

Graphical user interface, text, application

Description automatically generated

Code: <https://edaplayground.com/x/pwZR>

1. Design RTL level module to implement T-FF.

Ans:

module t\_ff (

input t,

input rst,

input clk,

output reg q

);

always @(posedge clk) begin

if (!rst)

q<=0;

else

if (t)

q <= ~q;

else

q <= q;

end

endmodule

Testbench:

module tfftb;

reg rst;

reg clk;

reg t;

// reg q;

t\_ff u1(.t(t),

.rst(rst),

.clk(clk),

.q(q));

always #5 clk = ~clk;

initial begin

rst<=0; clk<=0; t<=0;

$monitor("rst= %b t= %b q= %b", rst, t, q);

#6 rst <= 1;

#7 t <= 1;

#6 t<= 0;

#20 $finish;

end

endmodule

Result:

Background pattern

Description automatically generated with low confidence

EDA link: <https://edaplayground.com/x/GcQ3>

1. Design a comparator to compare two 1-bit inputs in gate level.

module compare(a,b,eq,gr,ls);

input a, b;

output eq, gr, ls;

assign eq = a ~^ b;

assign gr = a & (~b);

assign ls = (~a) &b;

endmodule

#Testbench:

module tb;

reg a, b;

wire eq, gr, ls;

integer i;

compare u1 (.a(a),

.b(b),

.eq(eq),

.gr(gr),

.ls(ls));

initial begin

$monitor("A = %b B = %b equal=%b less=%b greater=%b", a, b, eq, ls, gr);

for (i=0; i<4; i=i+1)begin

{a,b} = i;

#2;

end

#2 $finish;

end

endmodule

Graphical user interface, text, application

Description automatically generated

Code Eda: <https://edaplayground.com/x/6Qzg>

1. Design a DFF in gate level module by instantiating built-in gates

Diagram

Description automatically generated

*module dff(*

*input d,*

*input clk,*

*output q*

*);*

*wire d\_bar;*

*wire clk\_bar;*

*not #(1) dbar(d, d\_bar);*

*not #(1) clkbar(clk, clk\_bar);*

*and #(1) and1(d\_bar, clk, q);*

*and #(1) and2(q, clk\_bar, q);*

*endmodule*

*testbench*

*module dff\_tb;*

*reg d;*

*reg clk;*

*wire q;*

*dff u1 (*

*.d(d),*

*.clk(clk),*

*.q(q)*

*);*

*always #5 clk = ~clk;*

*initial begin*

*d = 1;*

*#10 $display("d = %b q = %b", d, q);*

*$finish;*

*end*

*endmodule*

*eda:* [*https://edaplayground.com/x/ZxhN*](https://edaplayground.com/x/ZxhN)

1. Create UDPs and then design top module with UDP instances for the following circuit.

Diagram

Description automatically generated

Ans:

primitive udpAnd(f\_o, a\_i, b\_i);

output f\_o;

input a\_i;

input b\_i;

table

1 1 : 1;

0 ? : 0;

? 0 : 0;

endtable

endprimitive

primitive udpOR(f\_o, a\_i, b\_i);

output f\_o;

input a\_i;

input b\_i;

table

1 1 : 1;

1 ? : 1;

? 1 : 1;

0 0 : 0;

endtable

endprimitive

primitive udpXOR(f\_o, a\_i, b\_i);

output f\_o;

input a\_i;

input b\_i;

table

1 1 : 0;

1 0 : 1;

0 1 : 1;

0 0 : 0;

endtable

endprimitive

`include "udpXOR.v"

`include "udpAND.v"

`include "udpOR.v"

module udpfinal(sum, cout, x, y, cin);

output sum, cout;

output f\_oa;

input x, y, cin;

wire s1, c1, c2;

udpXOR (s1, x, y);

udpAND (c1, x, y);

udpXOR (sum, s1, cin);

udpAND (c2, s1, cin);

udpOR (cout, c1, c2);

endmodule