Embedded Systems

CS429 Lab2

Name: Dipean Dasgupta **ID:202151188**

Task1 Solution:

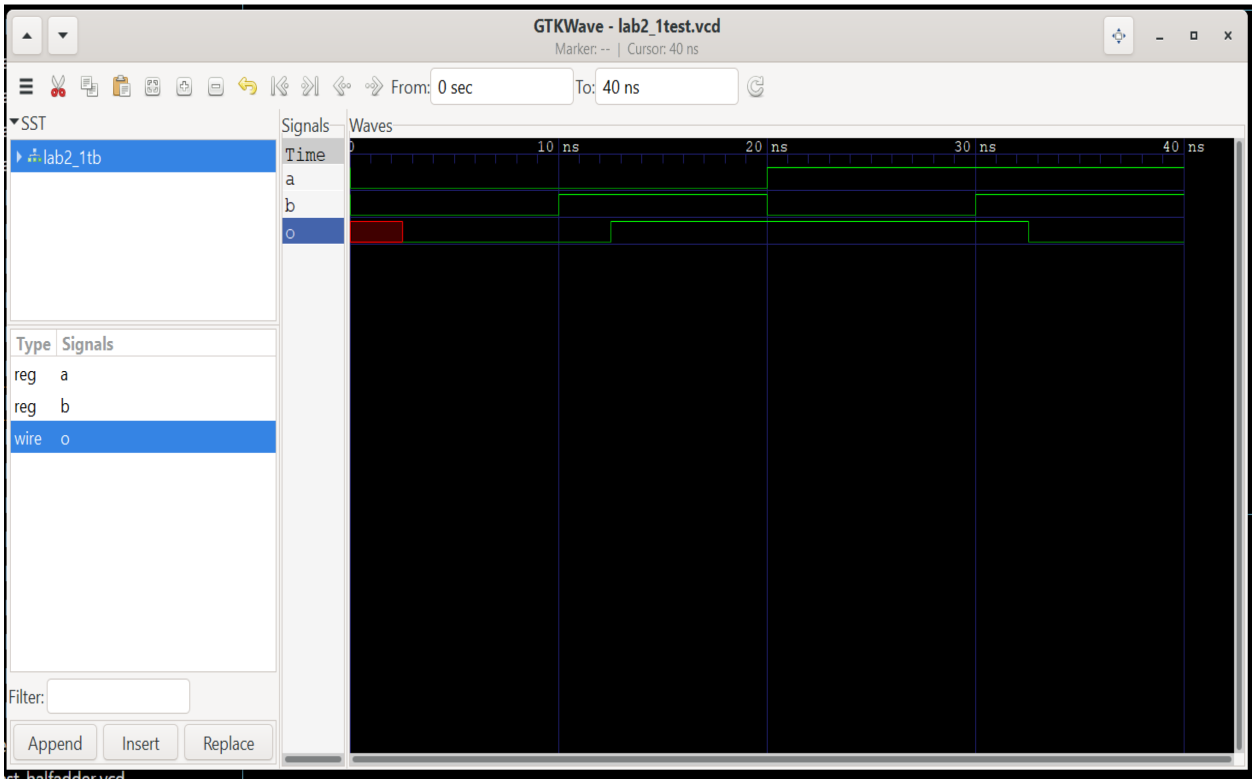
Module code:

```
`timescale 1ns/1ps
module xor_gate(
input a,b,
output out
);
wire d,e,f,g;
not_gate n1(a,d);
not_gate n2(b,e);
and_gate a1(a,e,f);
and_gate a2(b,d,g);
or_gate o1(f,g,out);
endmodule

module not_gate(
  input a,
  output o
  );
 assign #0.03 o = ~a;
endmodule

module and_gate(
  input a,
  input b,
  output o1
  );
 assign #1 o1 = a & b;
endmodule

module or_gate(
  input a,
  input b,
  output o2
  );
 assign #1.5 o2 = a | b;
endmodule
```

TestBench Code:

```
`include "lab2_1.v"
module lab2_1tb();
reg a;
reg b;
wire o;

xor_gate x1(.a(a),.b(b),.out(o));

initial begin
$dumpfile("lab2_1test.vcd");
$dumpvars(0, lab2_1tb);

  #0  a = 0;  b = 0;
  #10 a = 0;  b = 1;
  #10 a = 1;  b = 0;
  #10 a = 1;  b = 1;
  #10 $finish;
end

endmodule
```

OUTPUT:



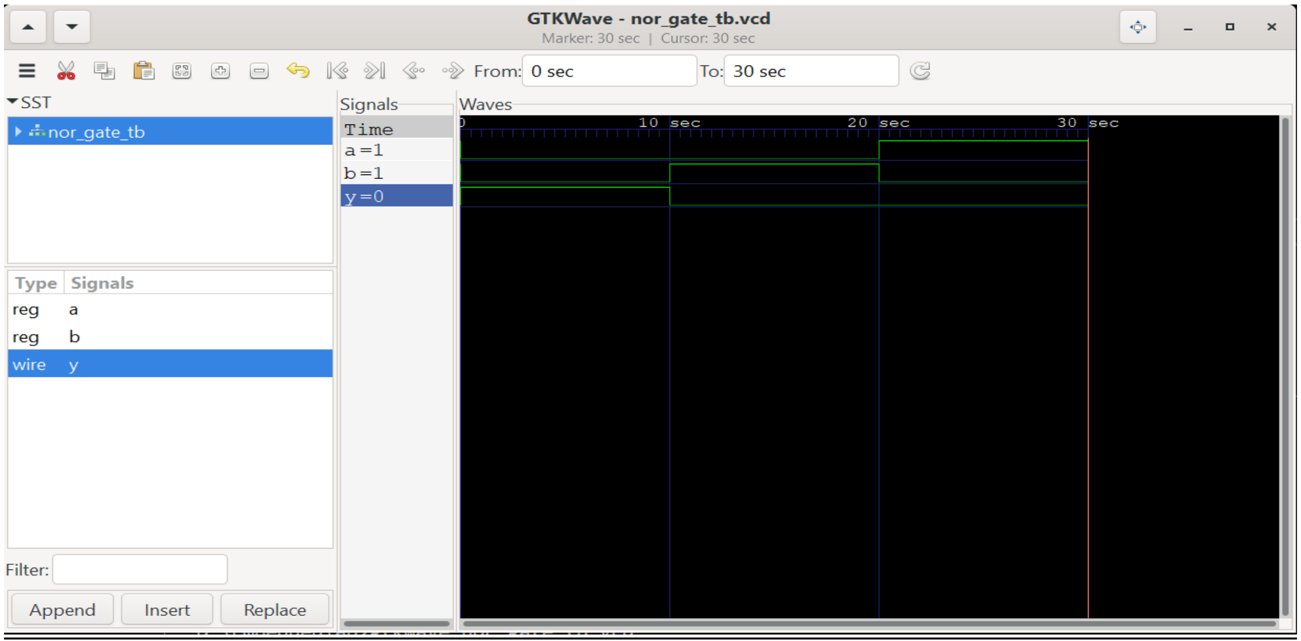
Task2a Solution:

Gate primitive

Module and Testbench Code:

```
module nor_gate(
    input a,
    input b,
    output y
    );
    assign y = ~(a | b);
endmodule

module nor_gate_tb;
reg a;
reg b;
wire y;
nor_gate uut(
    .a(a),
    .b(b),
    .y(y)
    );

initial begin
    $dumpfile("nor_gate_tb.vcd");
    $dumpvars(0, nor_gate_tb);
    //$monitor($time, " a=%b b=%b y=%b", a, b, y);

    #0  a=0; b=0;
    #10 a=0; b=1;
    #10 a=1; b=0;
    #10 a=1; b=1;
    $finish;
end

endmodule
```

OUTPUT:



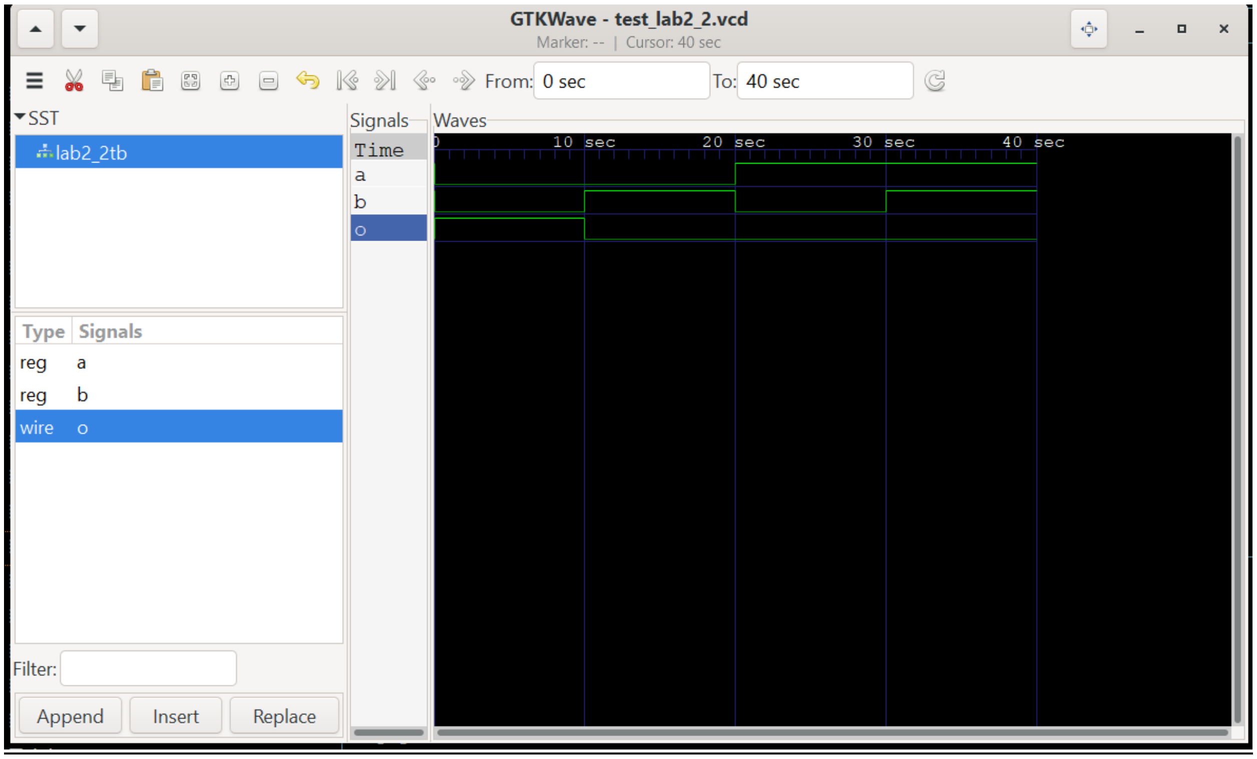
User Defined Primitive

Module and Testbench Code:

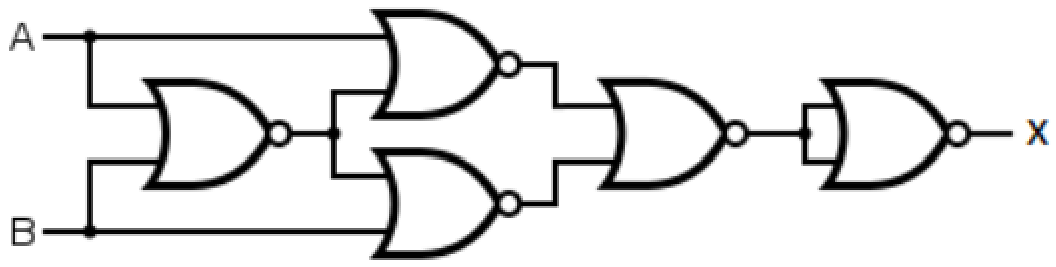
```
primitive lab2_2(
output o,
input a,
input b
);
    table
        0 0: 1;
        0 1: 0;
        1 0: 0;
        1 1: 0;
    endtable
endprimitive

module lab2_2tb;
wire o;
reg a;
reg b;
lab2_2 u0_DUT(
.o(o),
.a(a),
.b(b)
);
```

```
initial begin
$dumpfile("test_lab2_2.vcd");
$dumpvars(0,lab2_2tb);
a=0; b=0; #10;
a=0; b=1; #10;
a=1; b=0; #10;
a=1; b=1; #10;
end
endmodule
```

OUTPUT:



Task2b Solution ()

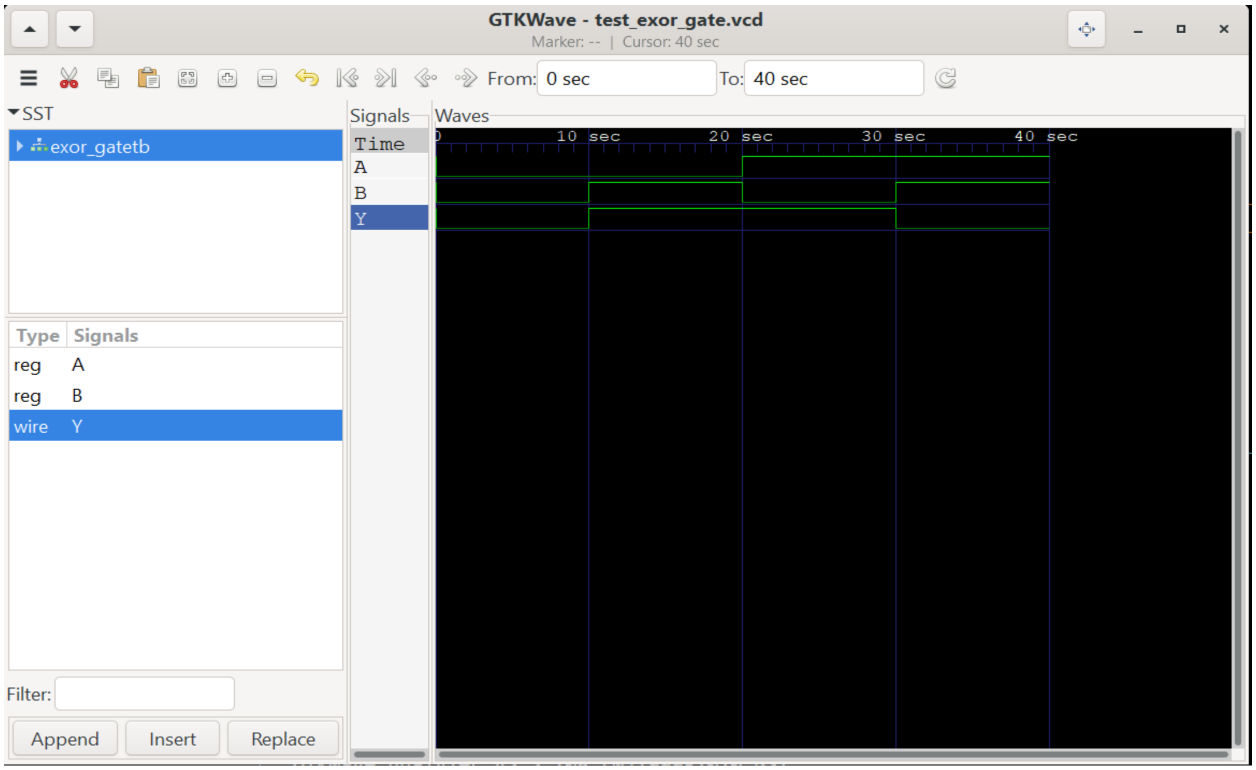


XOR using NOR gates

Module and Testbench Code:

```
primitive nor_gate (
output Y,
input A,
input B);
    table
        0 0 :  1;
        0 1 :  0;
        1 0 :  0;
        1 1 :  0;
    endtable
endprimitive

// UDP definition for XOR gate using NOR gates
module xor_gate (output Y, input A, input B);
  wire n1, n2, n3,n4;
    nor_gate u1 (.Y(n1), .A(A), .B(B));
    nor_gate u2 (.Y(n2), .A(A), .B(n1));
    nor_gate u3 (.Y(n3), .A(B), .B(n1));
    nor_gate u4 (.Y(n4), .A(n2), .B(n3));
    nor_gate u5 (.Y(Y), .A(n4), .B(n4));
endmodule

module exor_gatetb;
  reg A, B;
  wire Y;
  // Instantiate the XOR gate
  xor_gate uut (
    .Y(Y),
    .A(A),
    .B(B)
  );
  initial begin
    $dumpfile("test_exor_gate.vcd");
    $dumpvars(0, exor_gatetb);
    A = 0; B = 0; #10;
    A = 0; B = 1; #10;
    A = 1; B = 0; #10;
    A = 1; B = 1; #10;
    $finish;
  end
endmodule
```

OUTPUT:



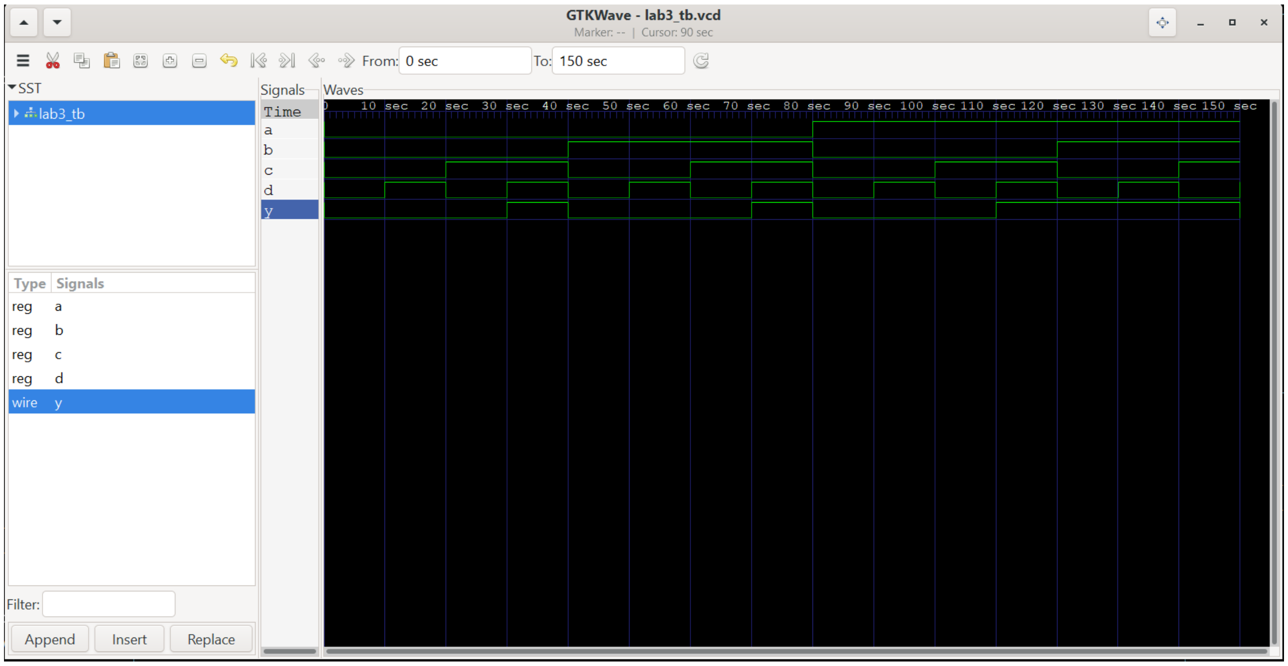
Task3 Solution:

RTL Level:

Module and Testbench Code:

```
module lab3(
    input a,
    input b,
    input c,
    input d,
    output y
    );
    assign y = (~(a & b) | (c & d)) ^ ((a & b) | ~(c & d));
endmodule

module lab3_tb;
reg a;
reg b;
reg c;
reg d;
wire y;

lab3 uut(.a(a),
      .b(b),
      .c(c),
      .d(d),
      .y(y));

initial begin
    $dumpfile("lab3_tb.vcd");
    $dumpvars(0, lab3_tb);
    #0  a=0; b=0; c=0; d=0;
    #10 a=0; b=0; c=0; d=1;
    #10 a=0; b=0; c=1; d=0;
    #10 a=0; b=0; c=1; d=1;
    #10 a=0; b=1; c=0; d=0;
    #10 a=0; b=1; c=0; d=1;
    #10 a=0; b=1; c=1; d=0;
    #10 a=0; b=1; c=1; d=1;
    #10 a=1; b=0; c=0; d=0;
    #10 a=1; b=0; c=0; d=1;
    #10 a=1; b=0; c=1; d=0;
    #10 a=1; b=0; c=1; d=1;
    #10 a=1; b=1; c=0; d=0;
    #10 a=1; b=1; c=0; d=1;
    #10 a=1; b=1; c=1; d=0;
    #10 a=1; b=1; c=1; d=1;
    $finish;
end

endmodule
```

OUTPUT:



Type	Signals
reg	a
reg	b
reg	c
reg	d
wire	y

Structural level:

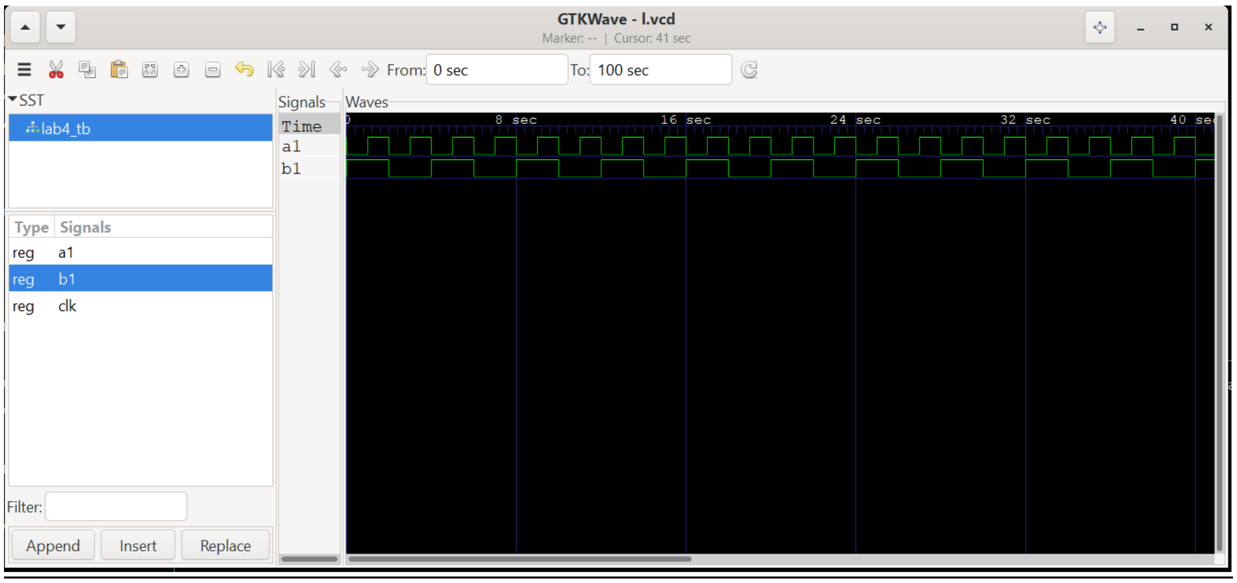
```
module logic_function (output, a, b, c, d);
input a, b, c, d;
output y
wire w1, w2, w3, w4, w5, w6;

nand g1(w1, a, b);
and g2(w2, c, d);
or g3(w3, w1, w2);
nand g4(w4, c, d);
and g5(w5, a, b);
or g6(w6, w4, w5);
xor g7(y, w3, w6);

endmodule
```

Task4 Solution:

Verilog RTL code:

```
module lab2_4;
reg a1=0;
reg b1=1;


always begin
  #1 a1 = ~a1;
end
always begin
 #2 b1 = ~b1;
end

initial begin
    $dumpfile("l.vcd");
    $dumpvars;
    #100 $finish;


end
endmodule
```

OUTPUT:



GTKWave - l.vcd
Marker: -- | Cursor: 41 sec
From: 0 sec
To: 100 sec
SST
lab4_tb
Type Signals
reg a1
reg b1
reg clk
Filter:
Append
Insert
Replace
Signals
Waves
Time
a1
b1
8 sec
16 sec
24 sec
32 sec
40 sec