**Experiment 2. Half adder and full adder**

Date: 04/06/21

**AIM**

b) Development of Verilog modules for full adder(dataflow level)

**VERILOG CODE**

**Data flow level**

module fulladder(

input wire a,b,cin,

output wire s,cout);

assign s = a^b^cin;

assign cout = (cin&(a^b))|(a&b);

endmodule

**Testbench**

// Code your testbench here

// or browse Examples

`timescale 1ns/1ps

module test\_full;

reg t\_a,t\_b,t\_cin;

wire t\_s,t\_cout;

fulladder uut(.a(t\_a),.b(t\_b),.cin(t\_cin),.s(t\_s),.cout(t\_cout));

initial

begin

$dumpvars(1,test\_full);

t\_a=1'b0;

t\_b=1'b0;

t\_cin=1'b0;

#10;

t\_a=1'b0;

t\_b=1'b0;

t\_cin=1'b1;

#10;

t\_a=1'b0;

t\_b=1'b1;

t\_cin=1'b0;

#10;

t\_a=1'b0;

t\_b=1'b1;

t\_cin=1'b1;

#10;

t\_a=1'b1;

t\_b=1'b0;

t\_cin=1'b0;

#10;

t\_a=1'b1;

t\_b=1'b0;

t\_cin=1'b1;

#10;

t\_a=1'b1;

t\_b=1'b1;

t\_cin=1'b0;

#10;

t\_a=1'b1;

t\_b=1'b1;

t\_cin=1'b1;

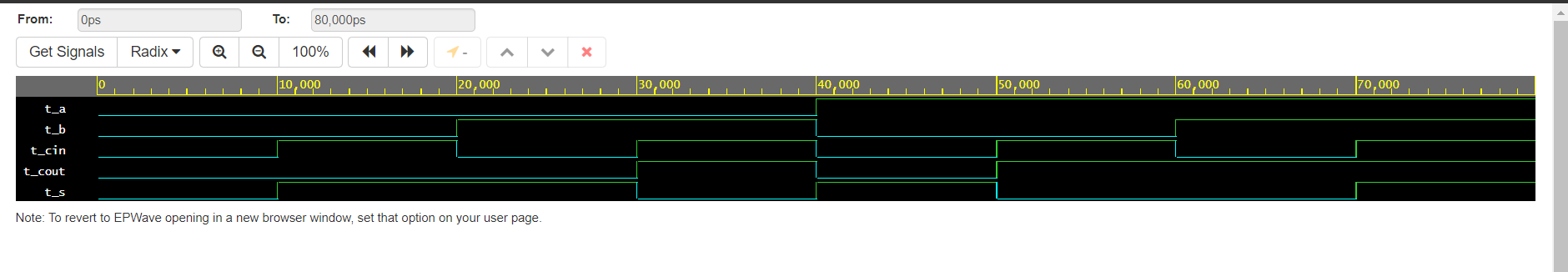
#10;

$stop;

end

endmodule

**Output**



**Experiment 3: Design of code converters**

**AIM**

Design and simulate the HDL code for(1/06/2021)  
(a) 4- bit binary to gray code converter

**VERILOG CODE**

**Gate level**

module bintogray(

input [3:0]b,

output [3:0]g);

xor x1(g[0],b[0],b[1]);

xor x2(g[1],b[1],b[2]);

xor x3(g[2],b[2],b[3]);

buf b1(g[3],b[3]);

endmodule

**Testbench**

// Code your testbench here

// or browse Examples

`timescale 1ns/1ps

module test\_bintogray;

reg [3:0] t\_b;

wire [3:0] t\_g;

bintogray uut(.b(t\_b),.g(t\_g));

initial

begin

$dumpvars(1,test\_bintogray);

t\_b = 4'b1100;

#10;

t\_b = 4'b1010;

#10;

t\_b = 4'b0011;

#10;

t\_b = 4'b1111;

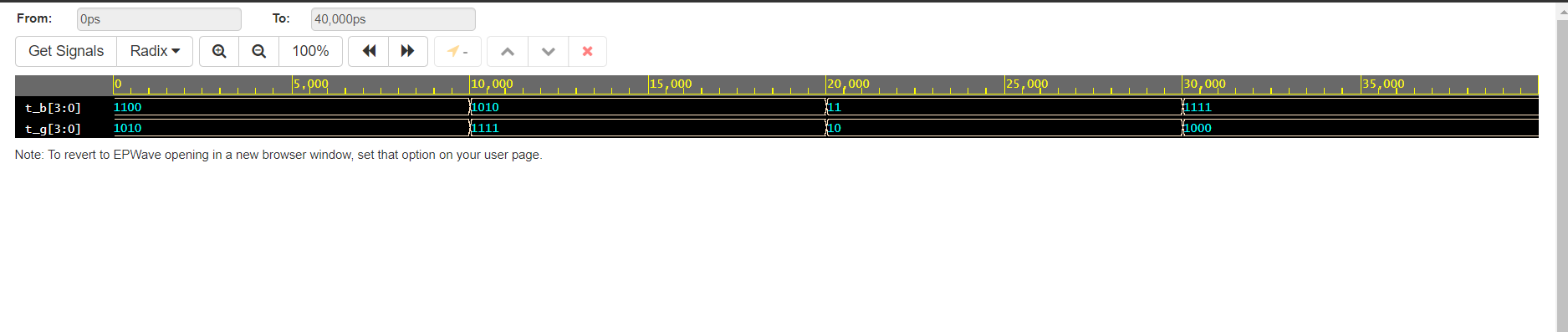
#10;

$stop;

end

endmodule

**Output**



(b) 4- bit gray to binary code converter

**VERILOG CODE**

**Gate level**

// Code your design here

module graytobin(

input [3:0]g,

output [3:0]b);

buf b1(b[3],g[3]);

xor x1(b[2],b[3],g[2]);

xor x2(b[1],b[2],g[1]);

xor x3(b[0],b[1],g[0]);

endmodule

**Testbench**

// Code your testbench here

// or browse Examples

`timescale 1ns/1ps

module test\_graytobin;

reg [3:0] t\_g;

wire [3:0] t\_b;

graytobin uut(.g(t\_g),.b(t\_b));

initial

begin

$dumpvars(1,test\_graytobin);

t\_g = 4'b1100;

#10;

t\_g = 4'b1001;

#10;

t\_g = 4'b0101;

#10;

t\_g = 4'b1101;

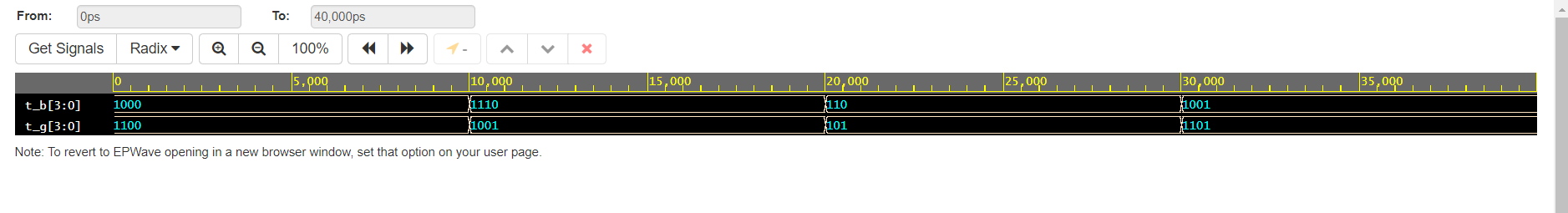
#10;

$stop;

end

endmodule

**Output**

****