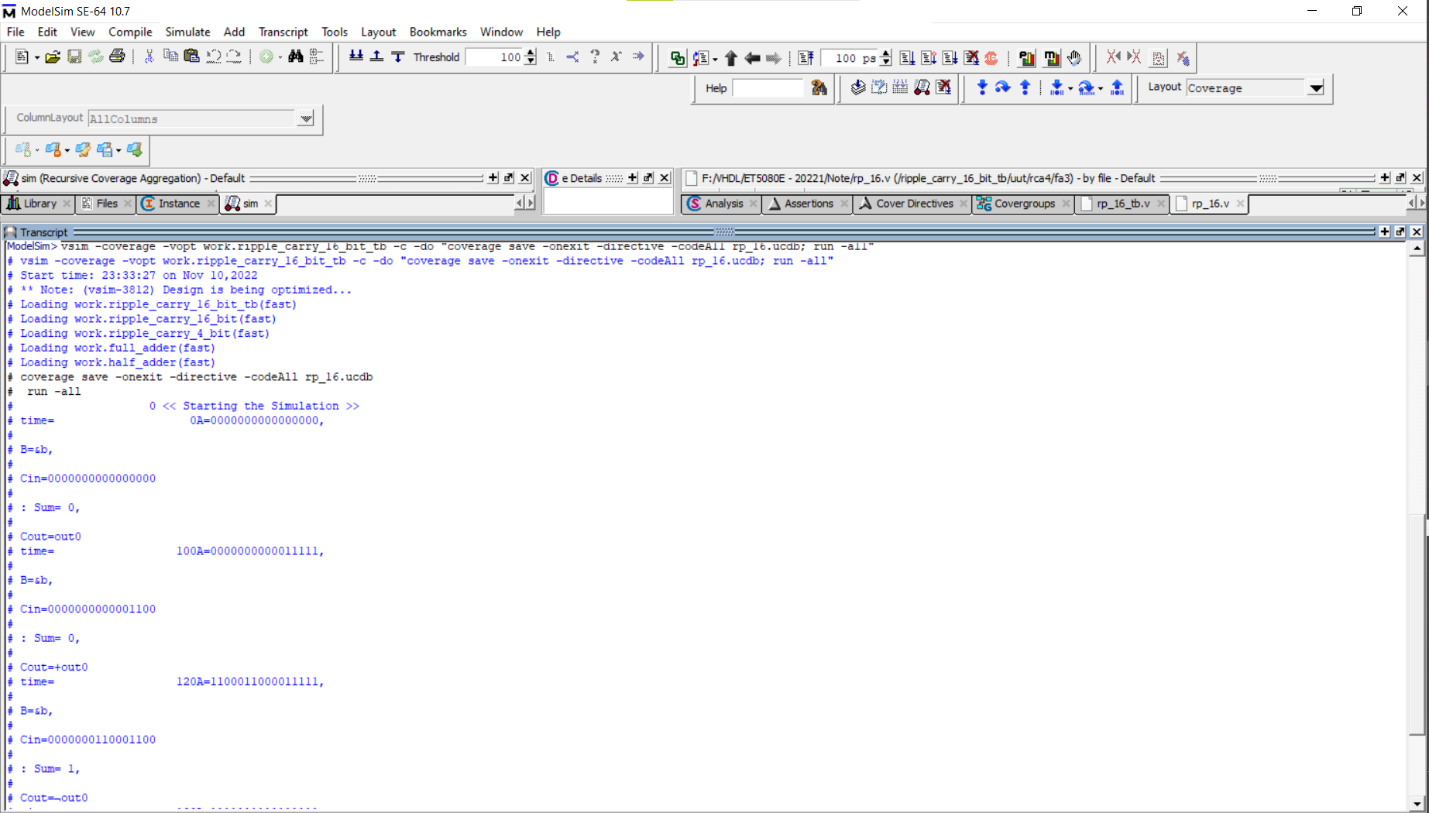
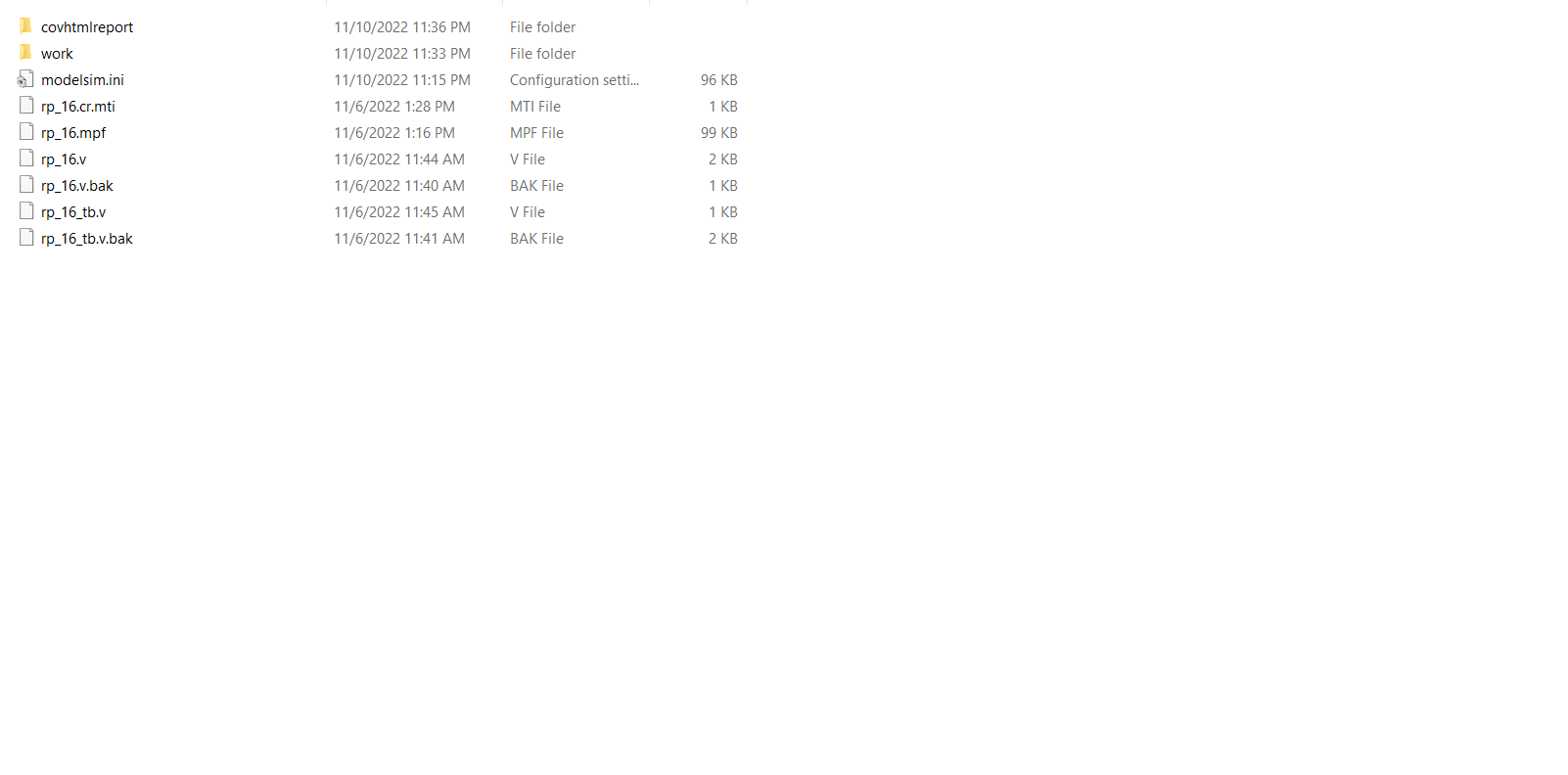
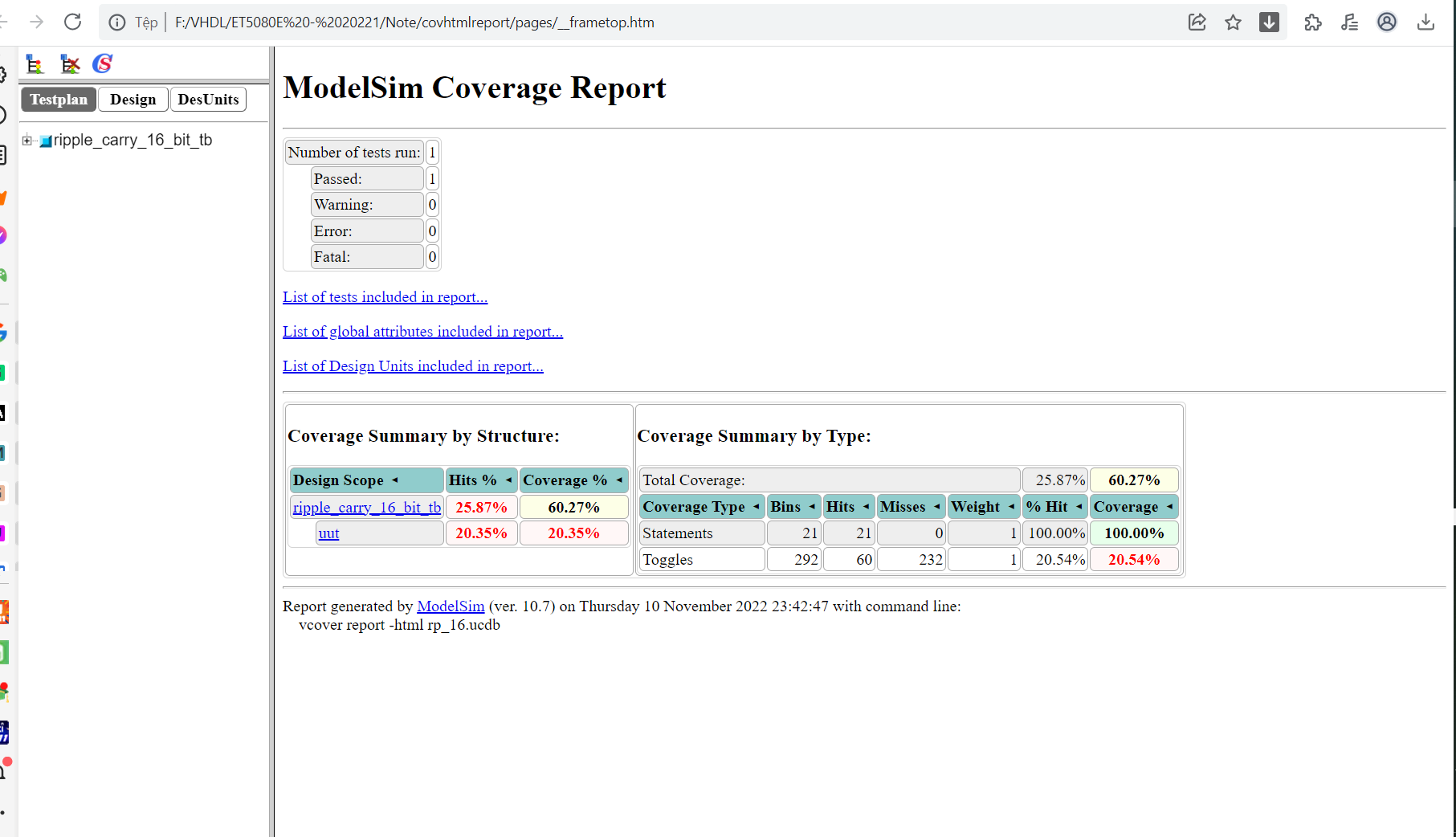
**Prob 1**

****

****

****

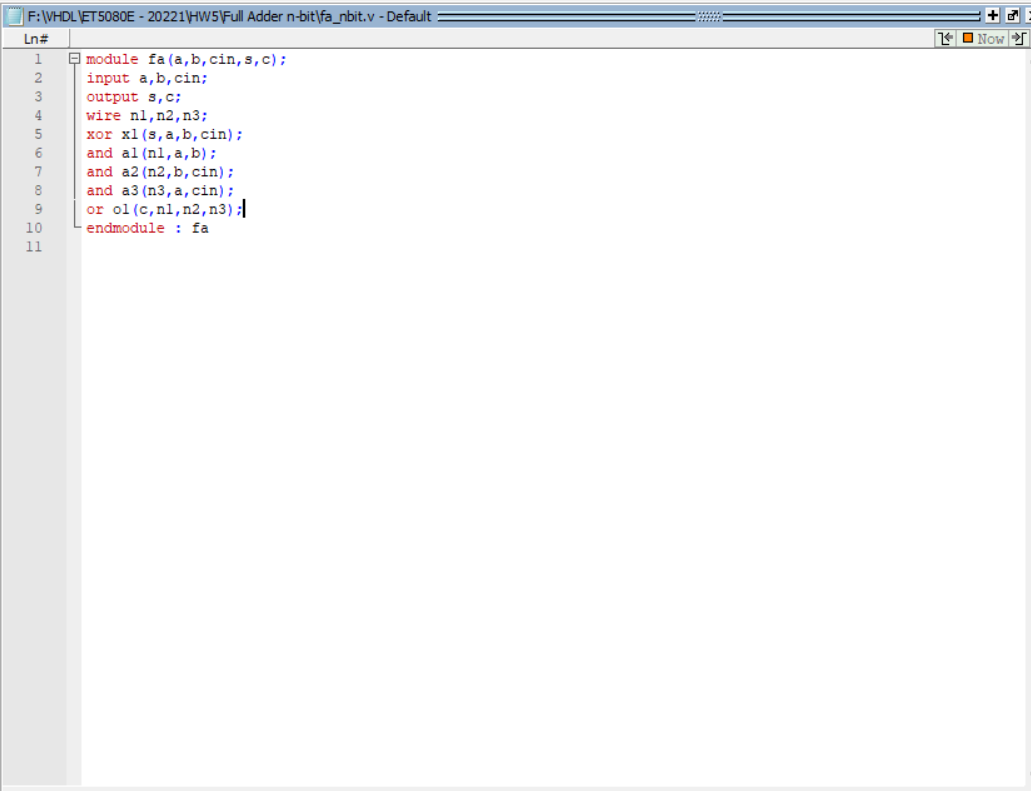
****

****

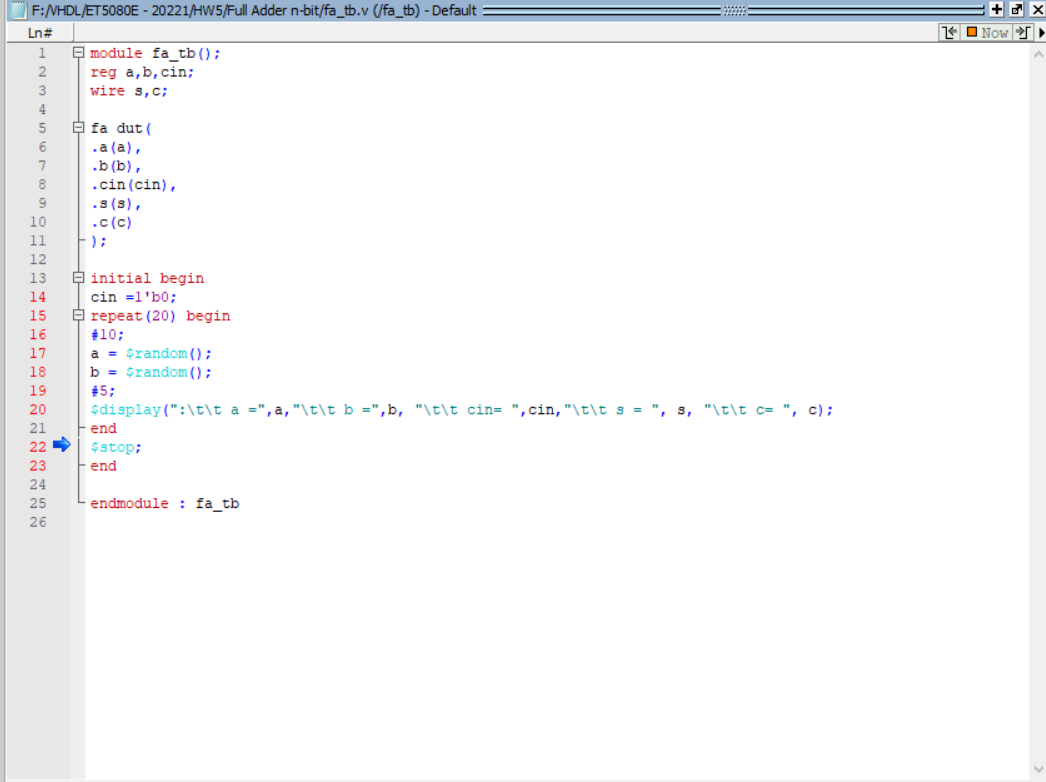
**Prob 2**

**Full Adder**

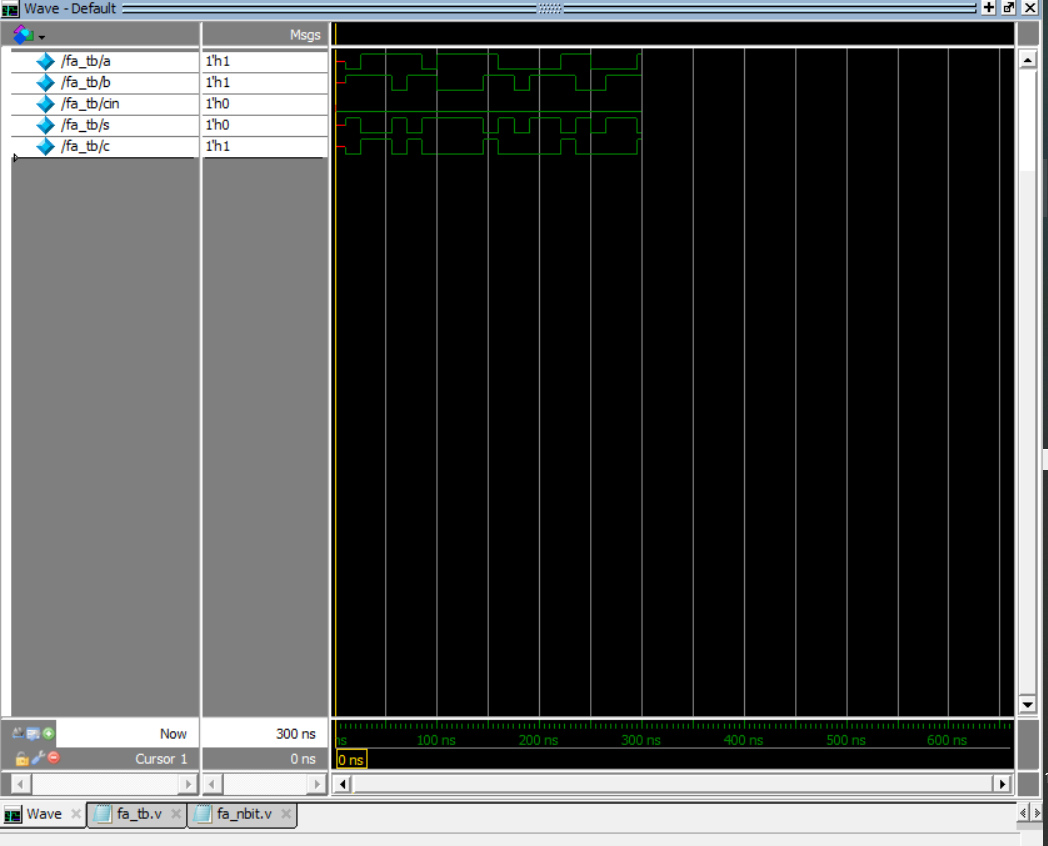
Verilog



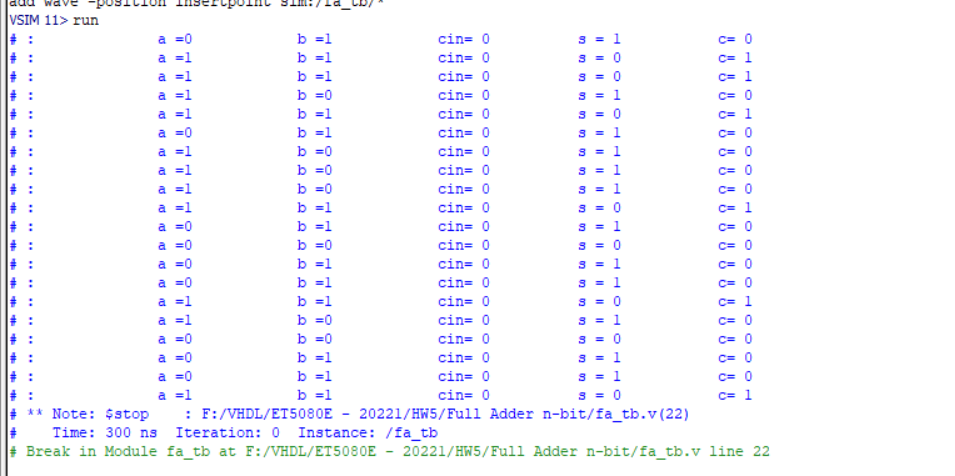
Testbench



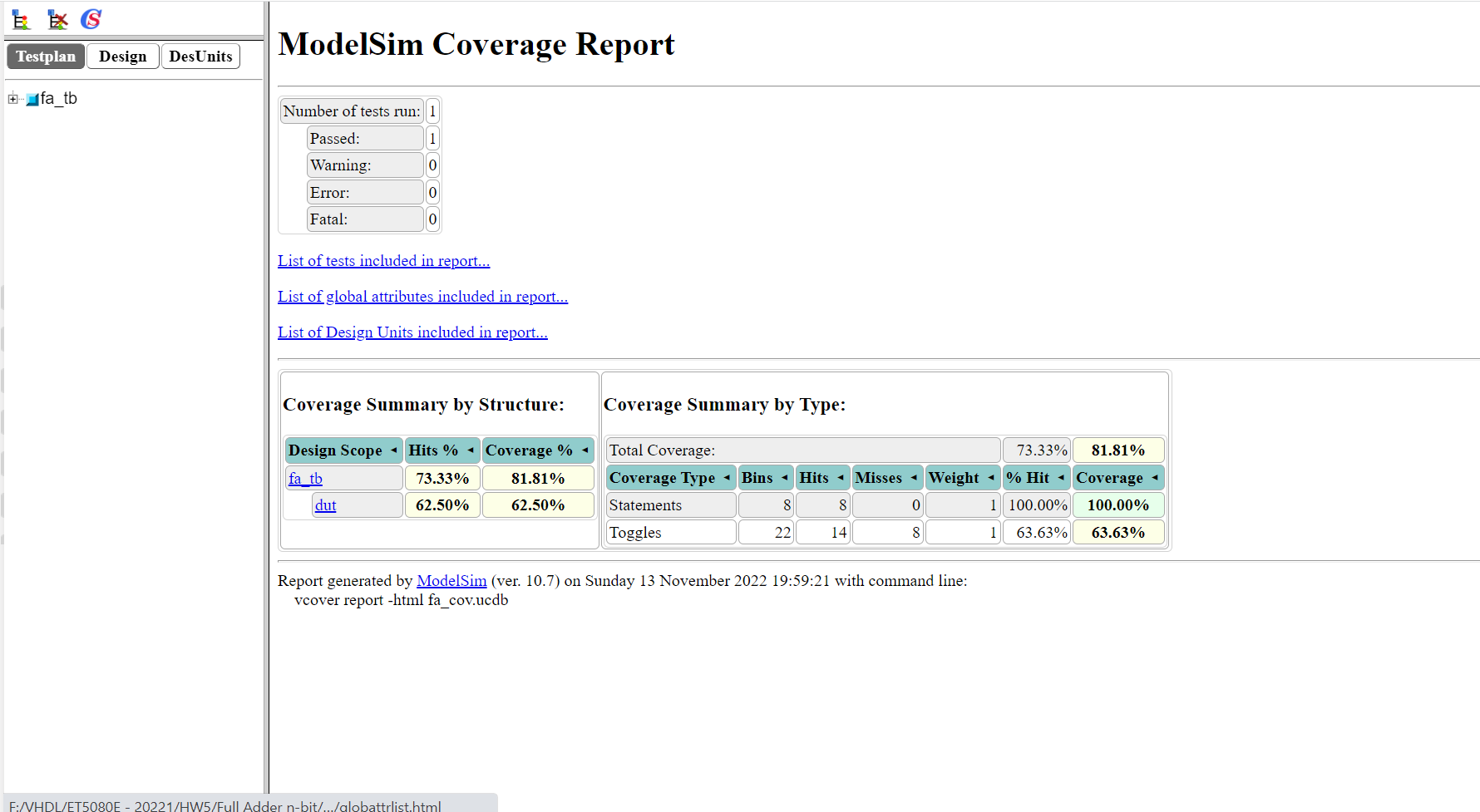
Waveform



Transcript



Report



**Carry Look Ahead Adder**

Verilog

CLA Verilog code

`timescale 1ns / 1ps

module full\_adder\_gp(

input wire A,

input wire B,

input wire Ci,

output wire S,

output wire Co

);

assign S = A ^ B ^ Ci;

endmodule

`timescale 1ns / 1ps

module GP(

input wire A,

input wire B,

input wire Ci,

output reg G,

output reg P,

output reg C

);

always @(\*)

begin

G = A & B;

P = A | B; // A xor B can also be used

C = G | P & Ci;

end

endmodule

`timescale 1ns / 1ps

module CLA (A,B,Ci,S,Co,PG,GG,CG);

parameter N = 4;

input wire [N-1:0] A;

input wire [N-1:0] B;

input wire Ci;

output wire [N-1:0] S;

output wire Co;

output reg PG,GG,CG;

wire [N-1:0] P;

wire [N-1:0] G;

wire [N:0] C;

assign C[0] = Ci;

genvar i;

generate

for (i=0; i < N; i=i+1)

begin : Build

full\_adder\_gp COMP1(.A(A[i]), .B(B[i]), .Ci(C[i]), .S(S[i]));

GP COMP2(.A(A[i]), .B(B[i]), .Ci(C[i]), .G(G[i]), .P(P[i]), .C(C[i+1]));

end

endgenerate

assign Co = C[N];

always@(\*)

begin

PG = &P;

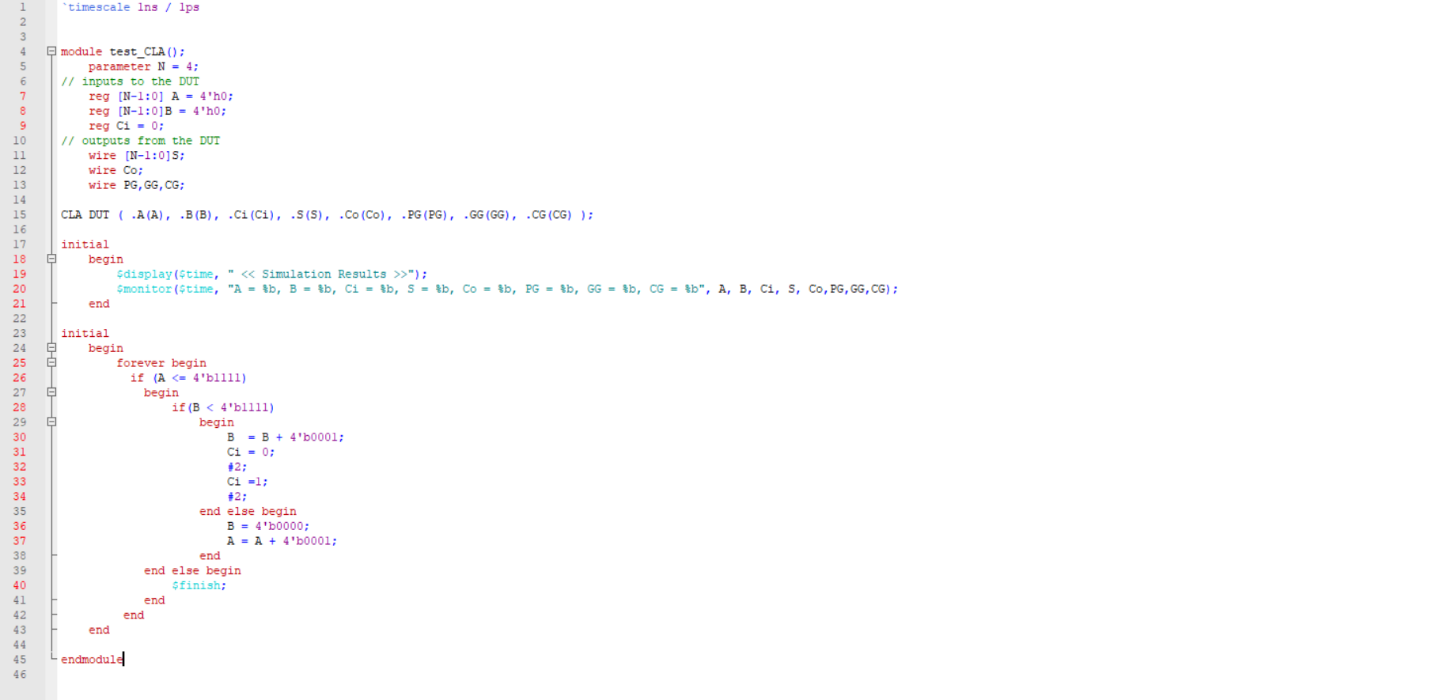
GG = C[N];

CG = GG | PG & Ci;

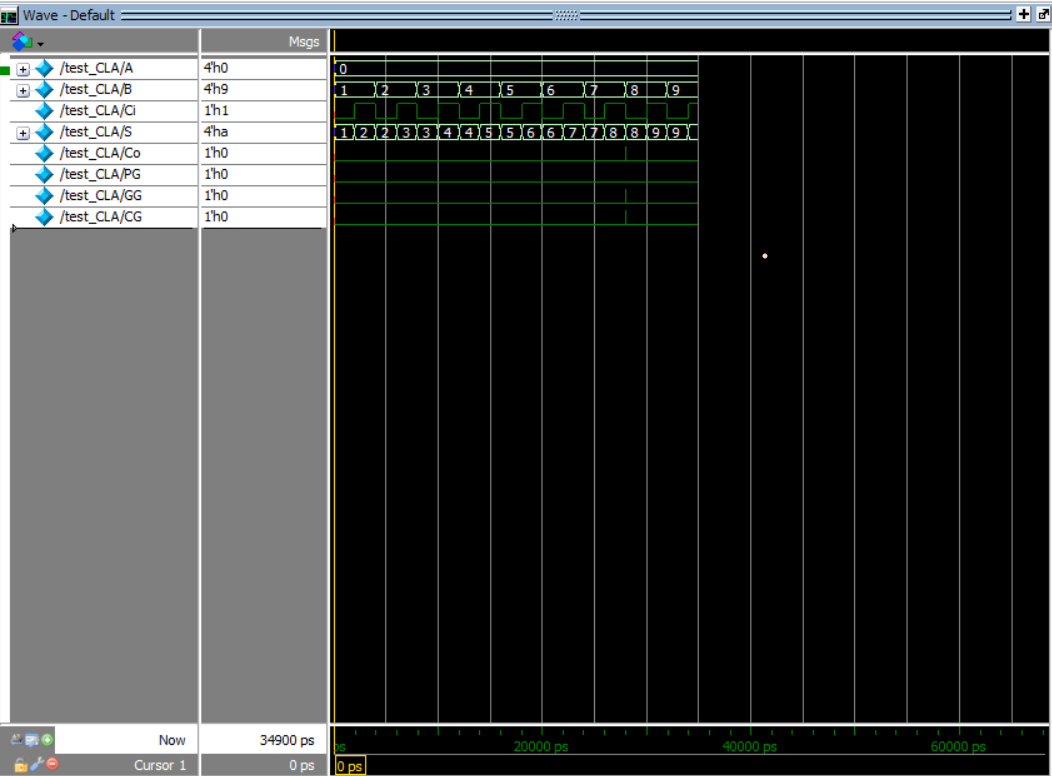
end

endmodule

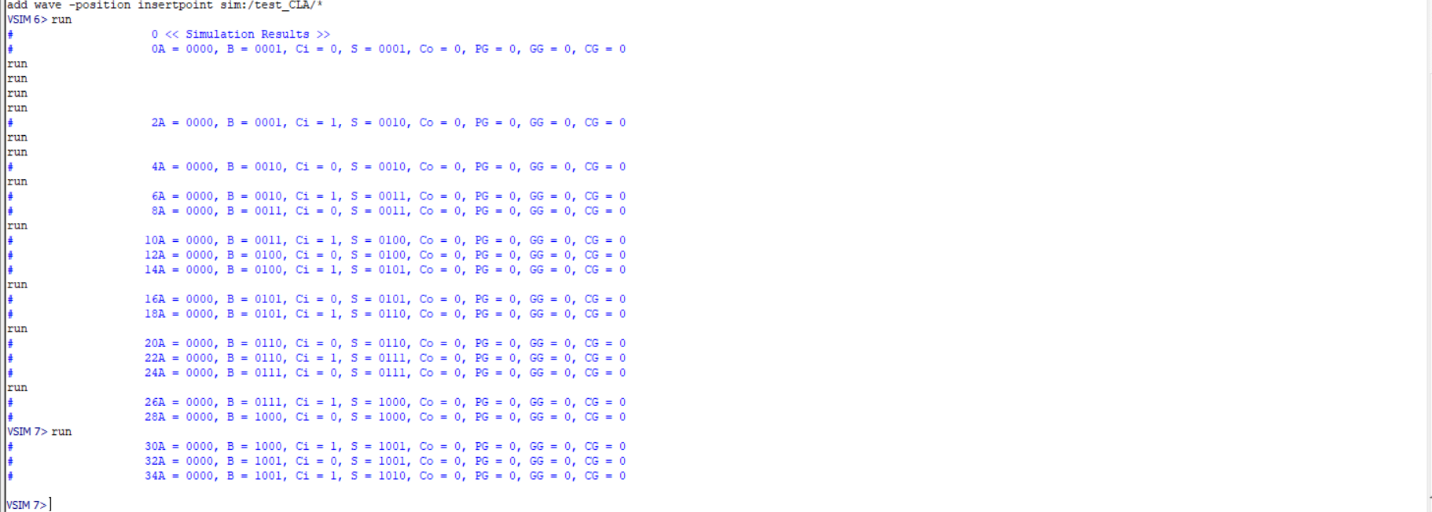
Testbench



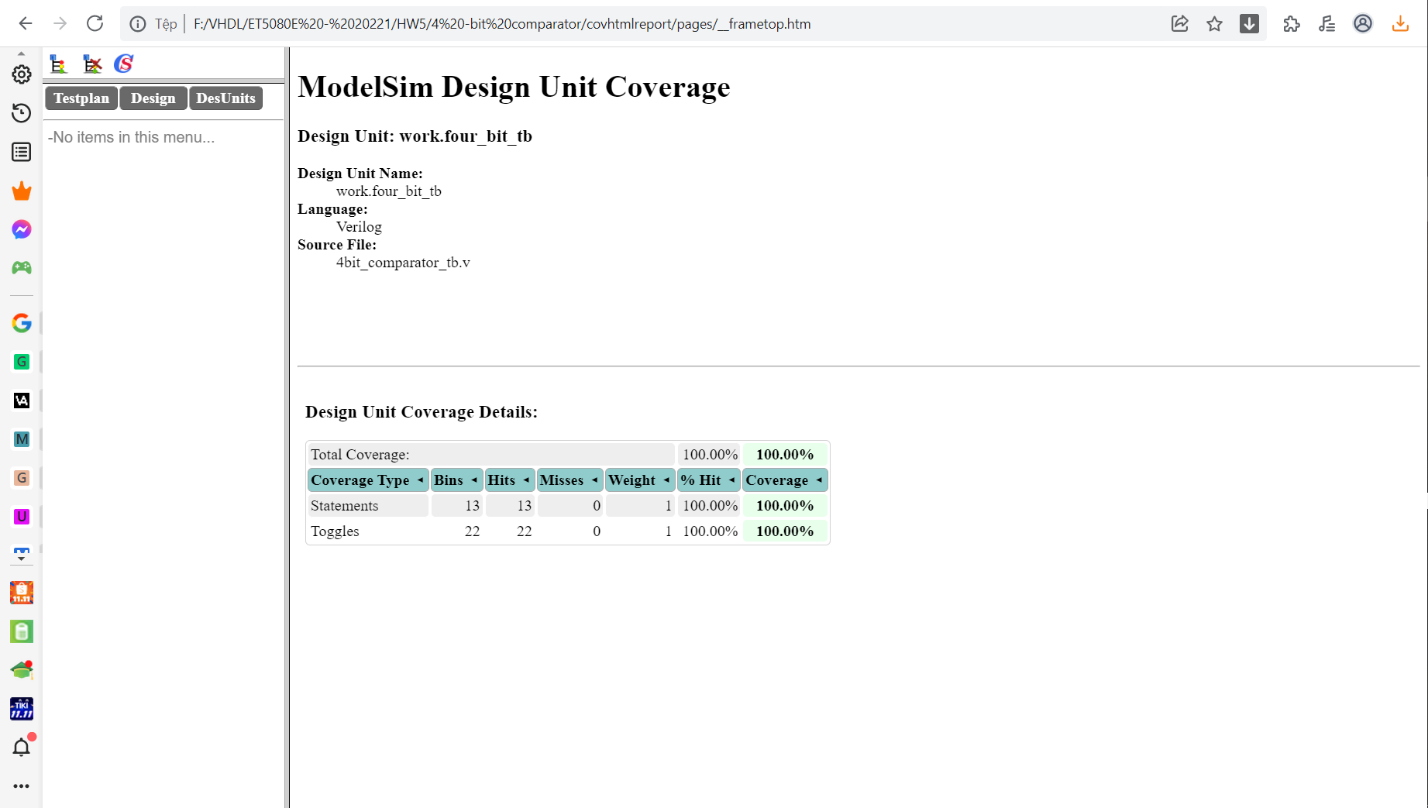
Wave



Transcipt

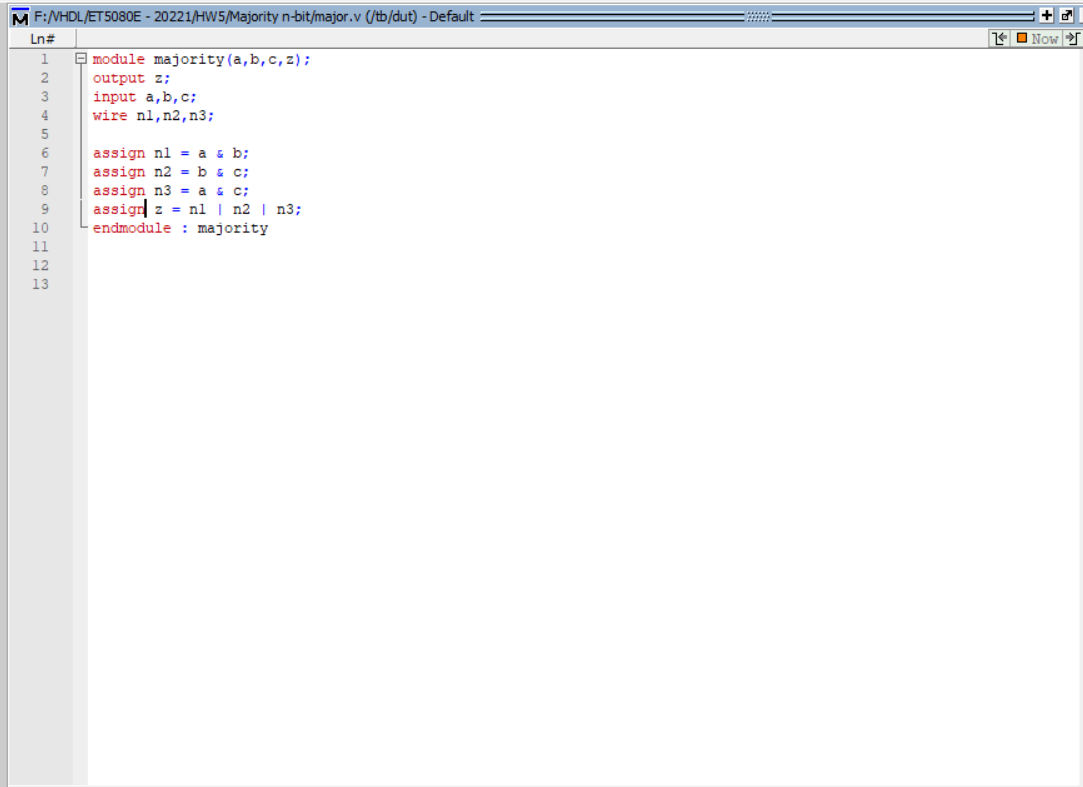


Report

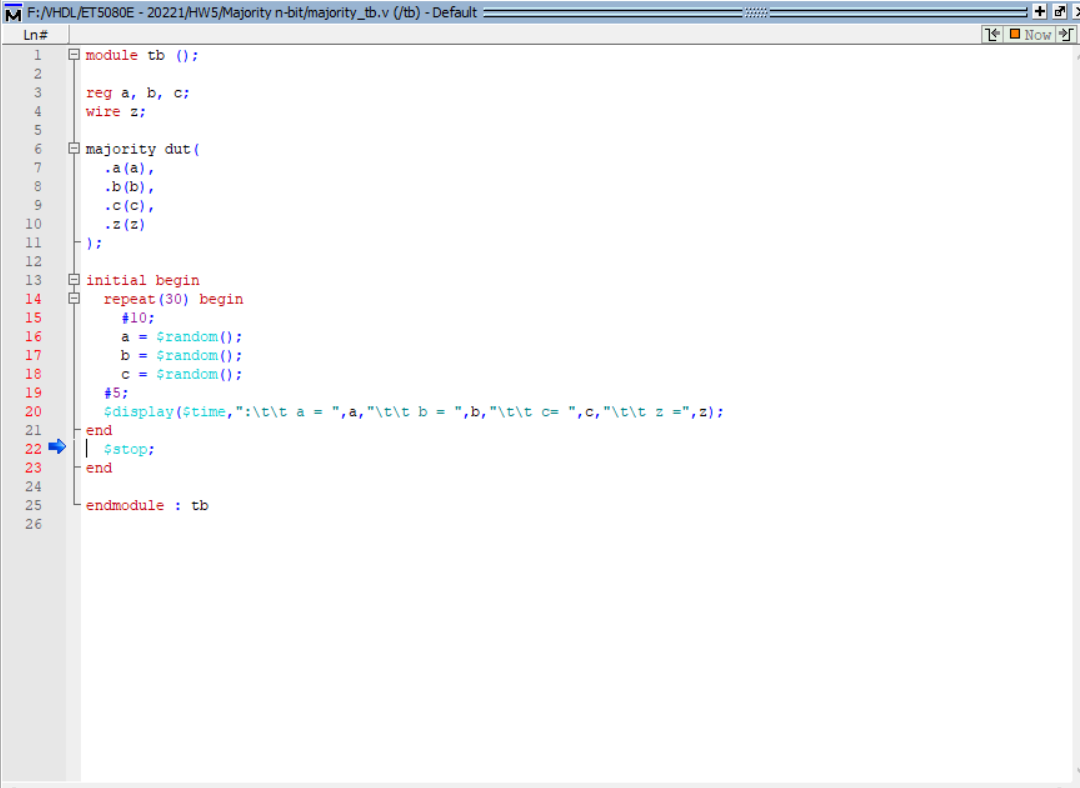


**Majority**

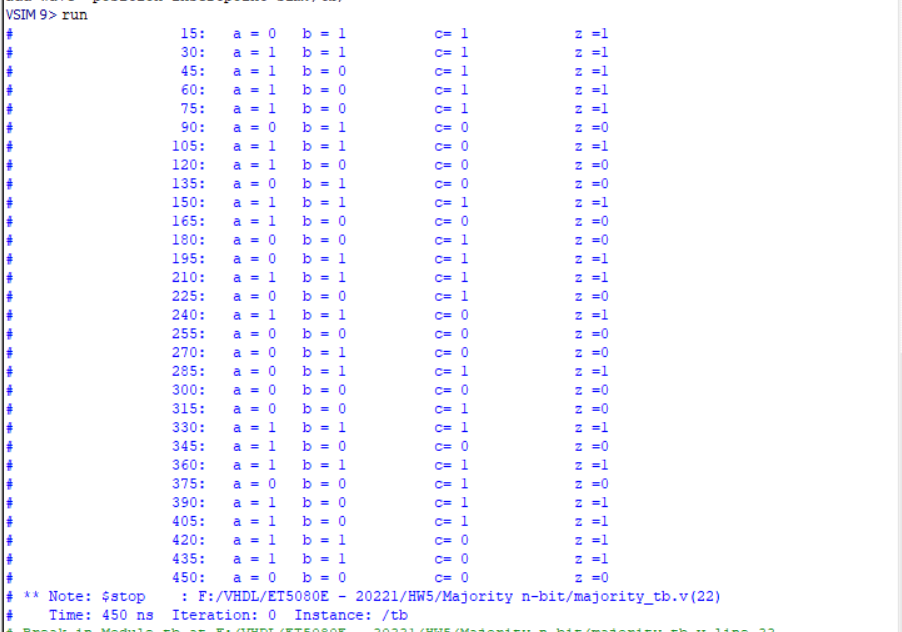
Verilog



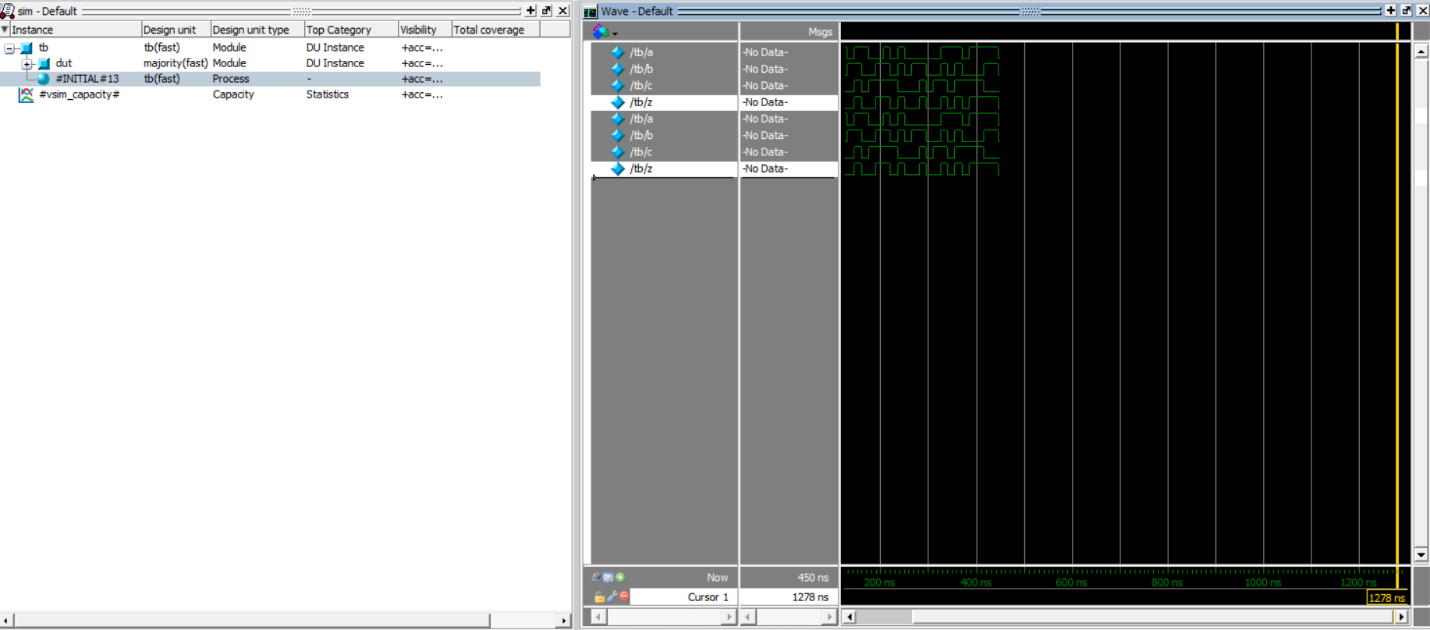
Testbench



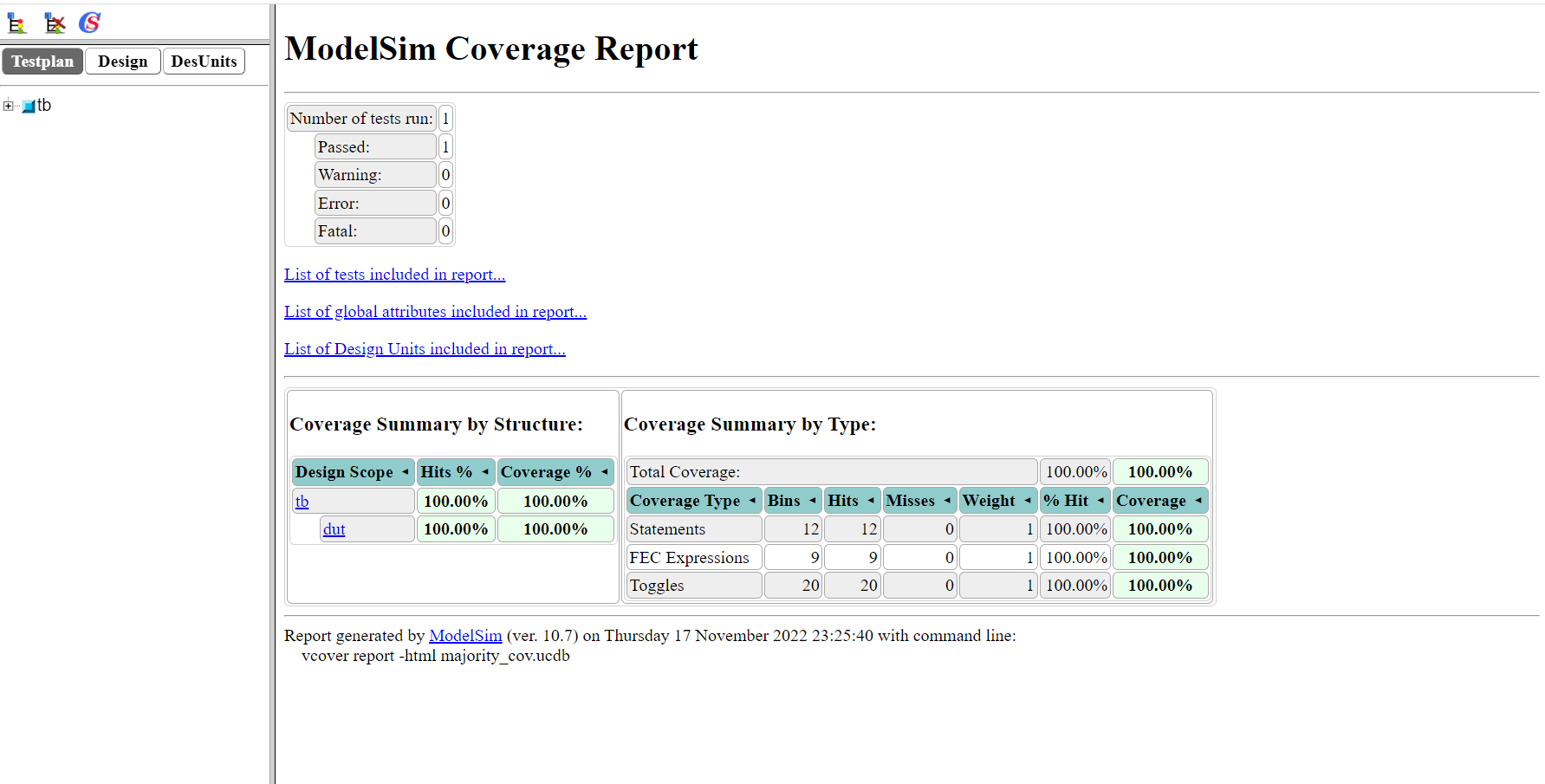
Script



Wave

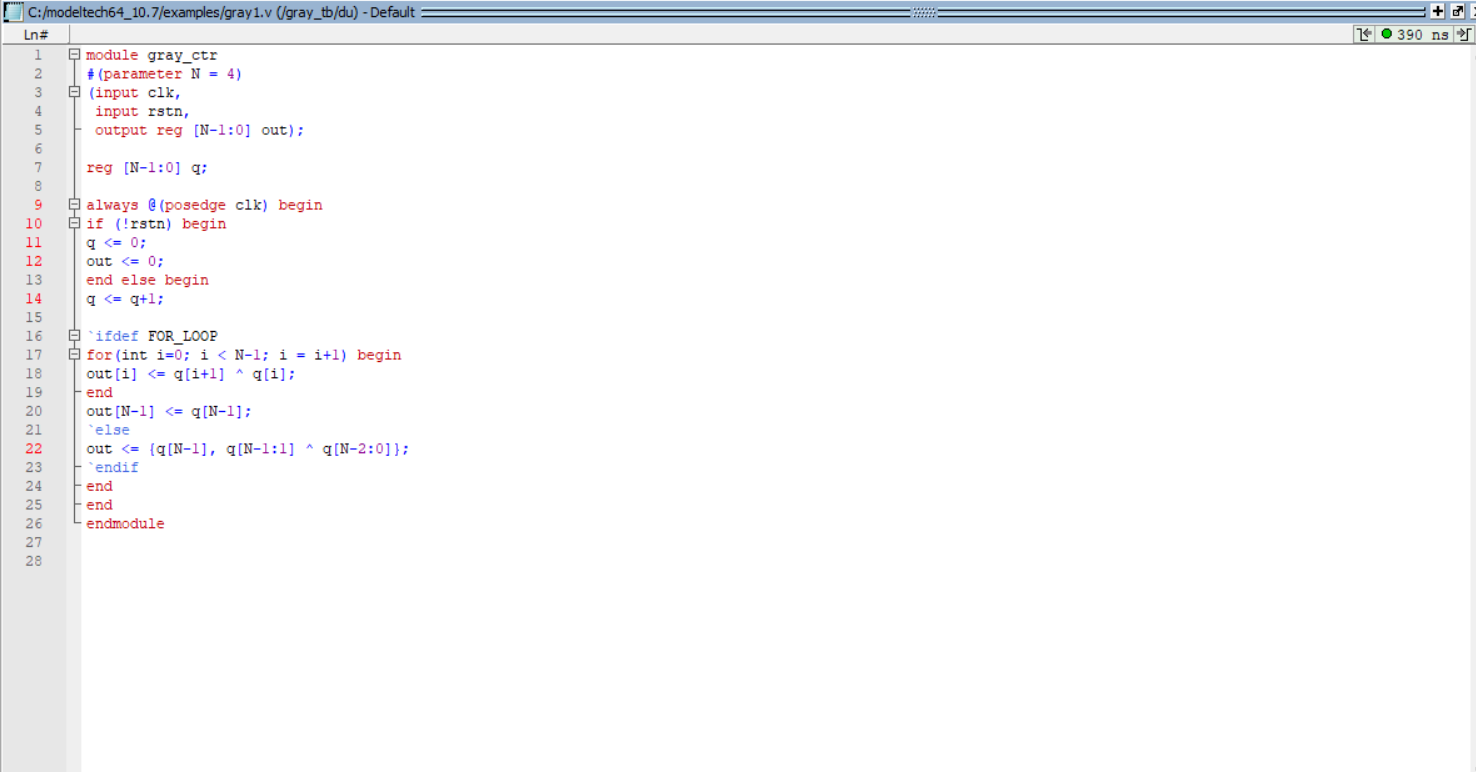


Report

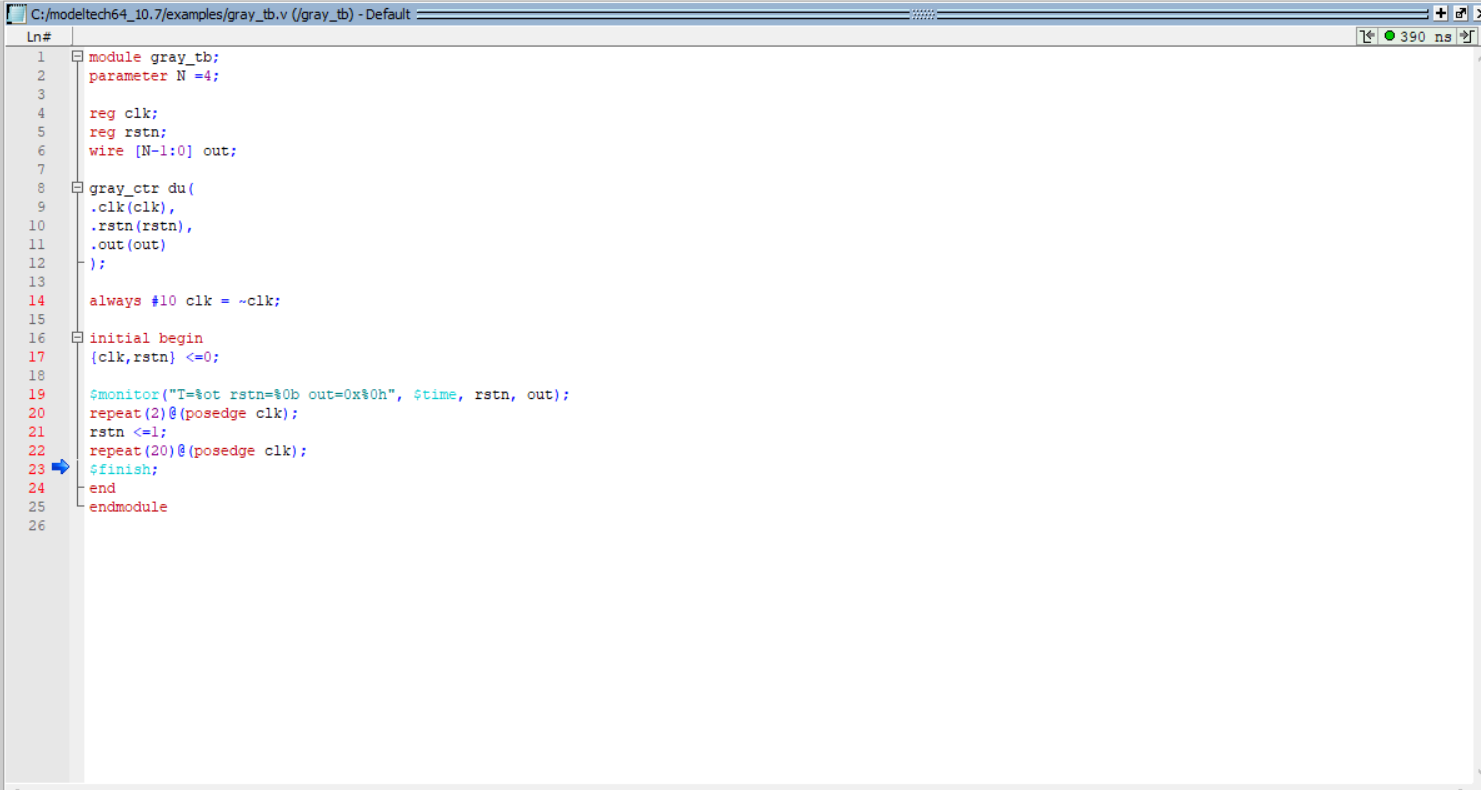


**Gray counter**

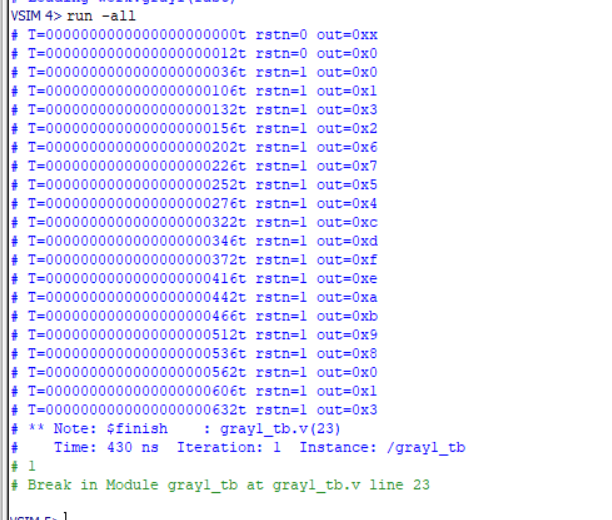
Verilog



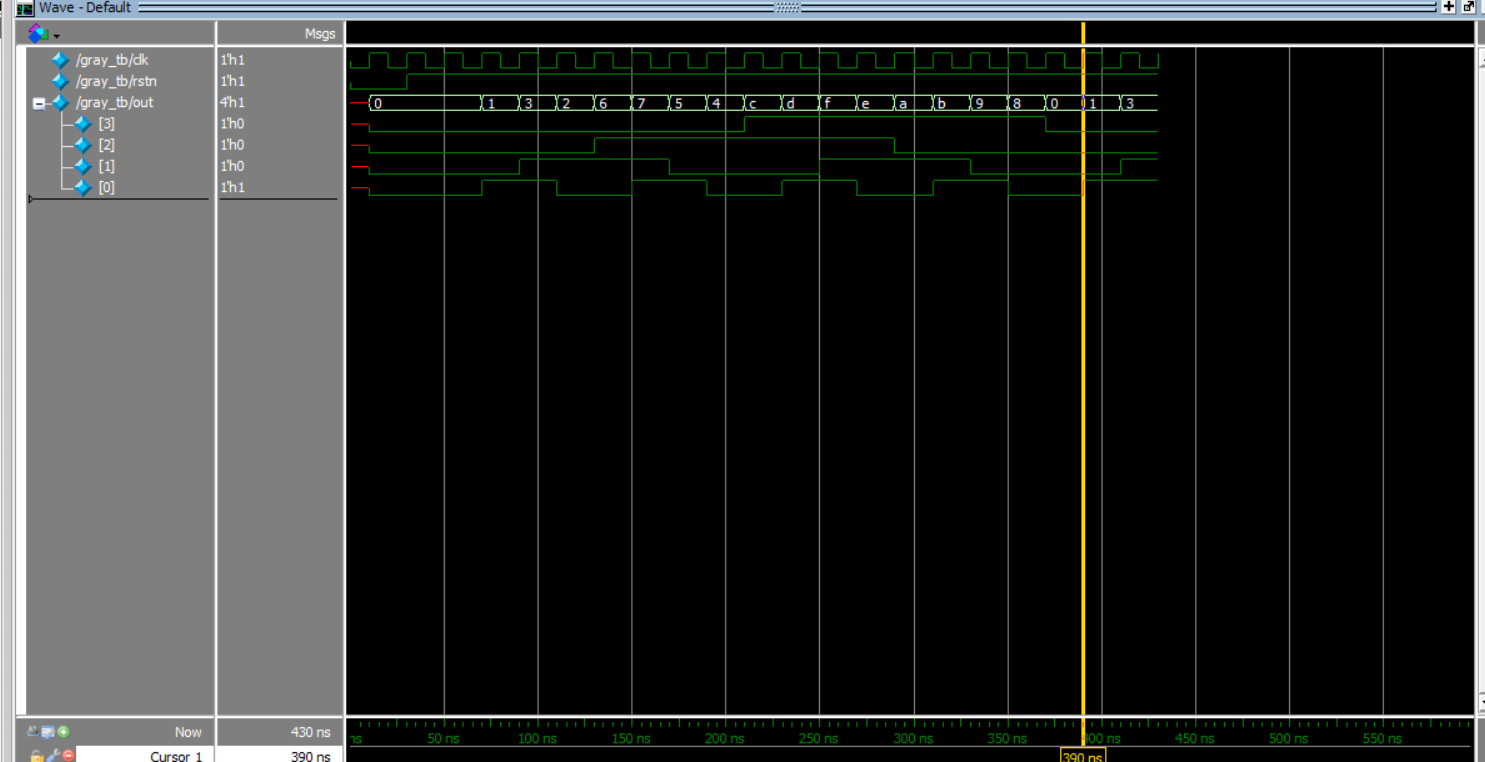
Testbench



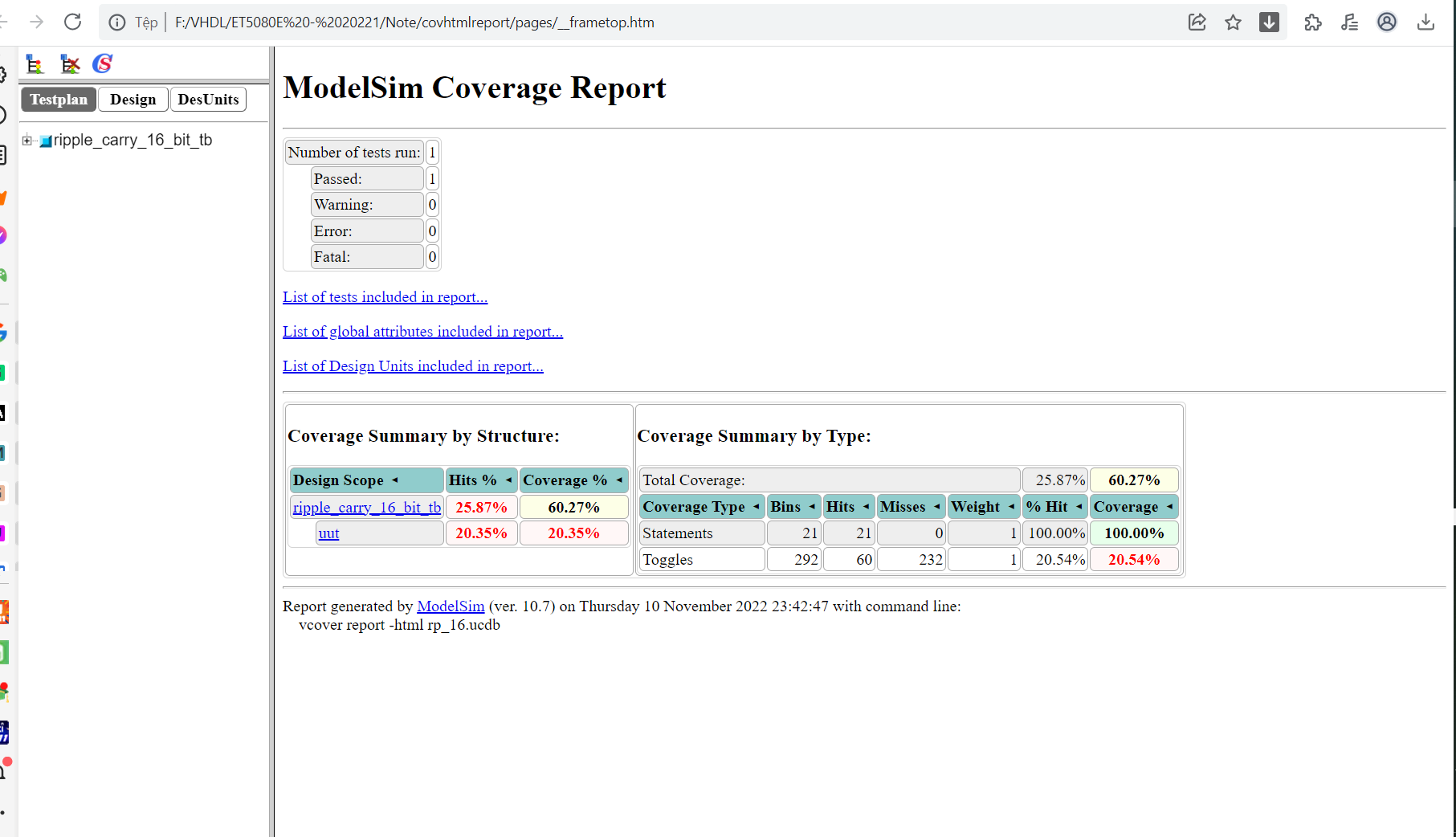
Script



Wave

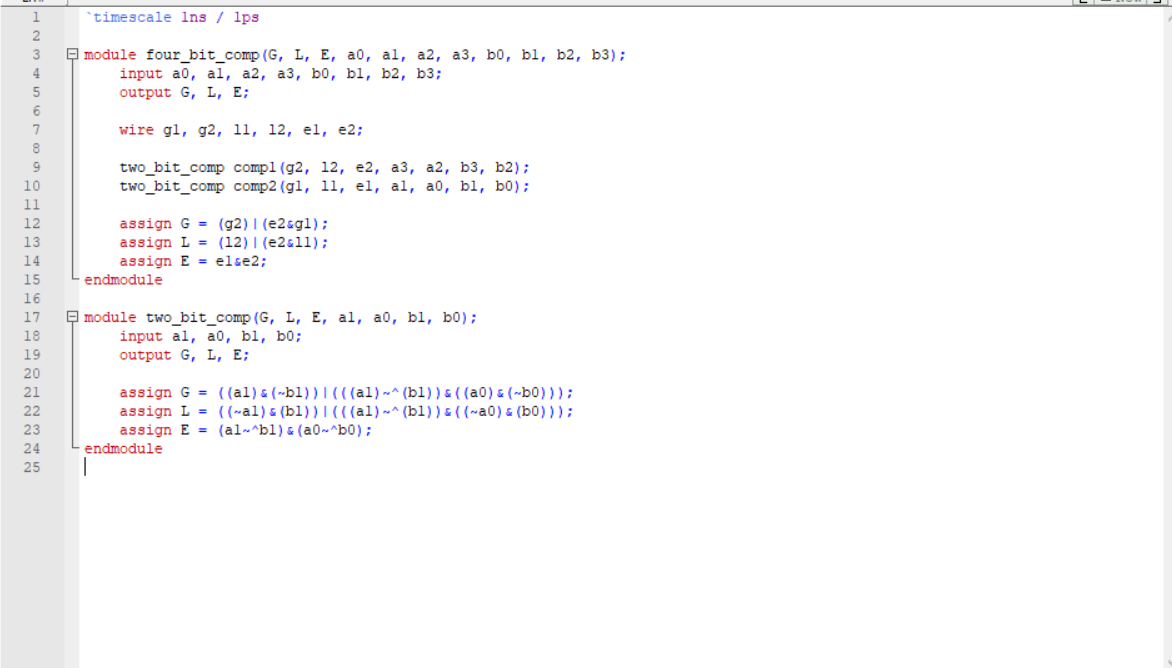


Report

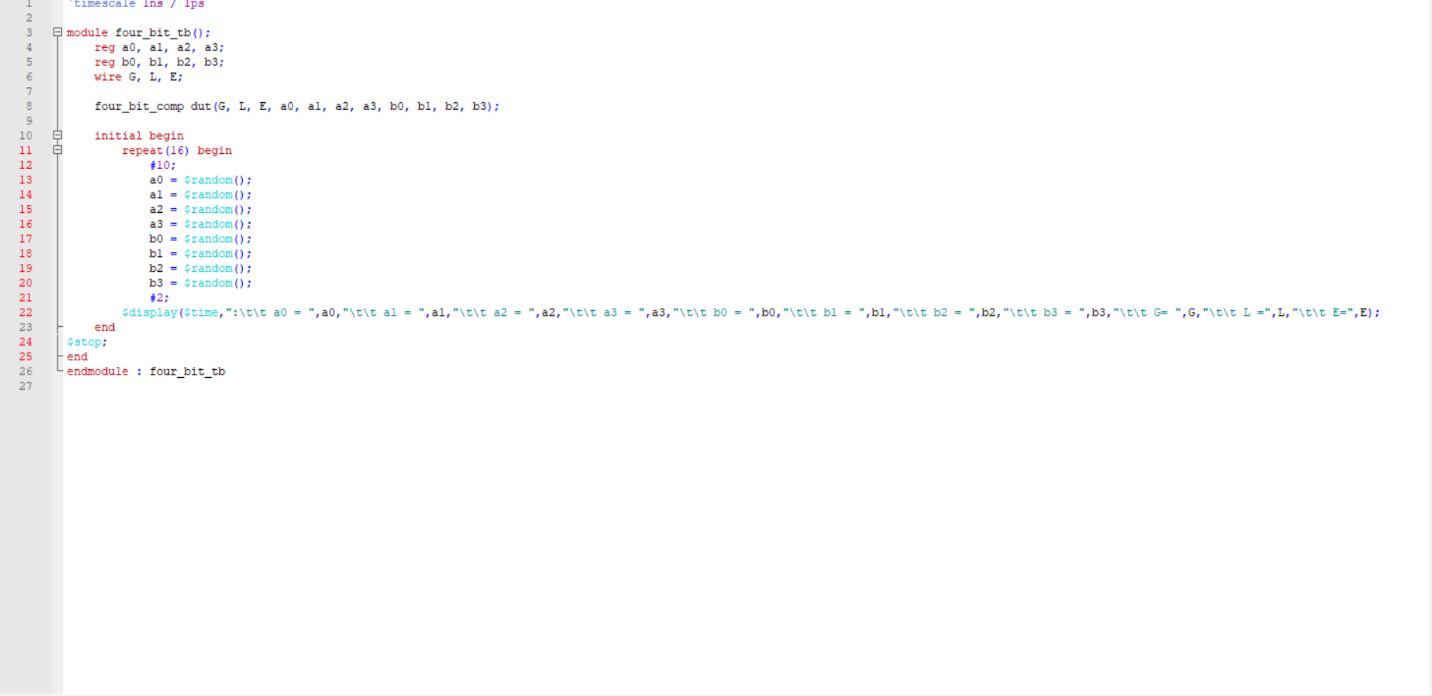
****

**4 -bit comparator**

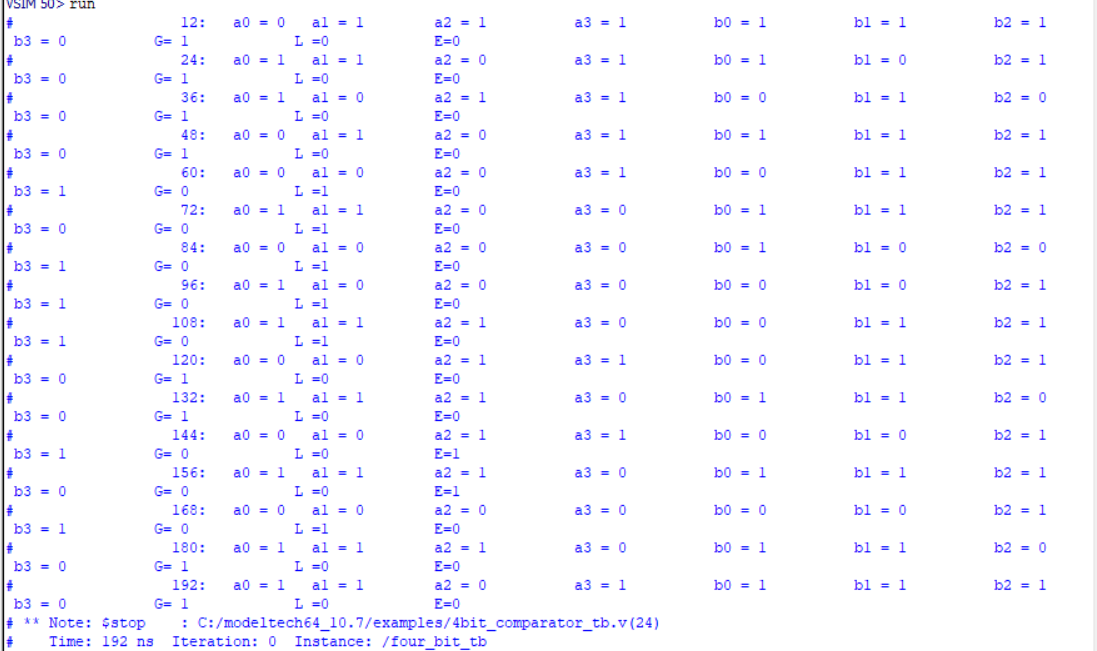
Verilog



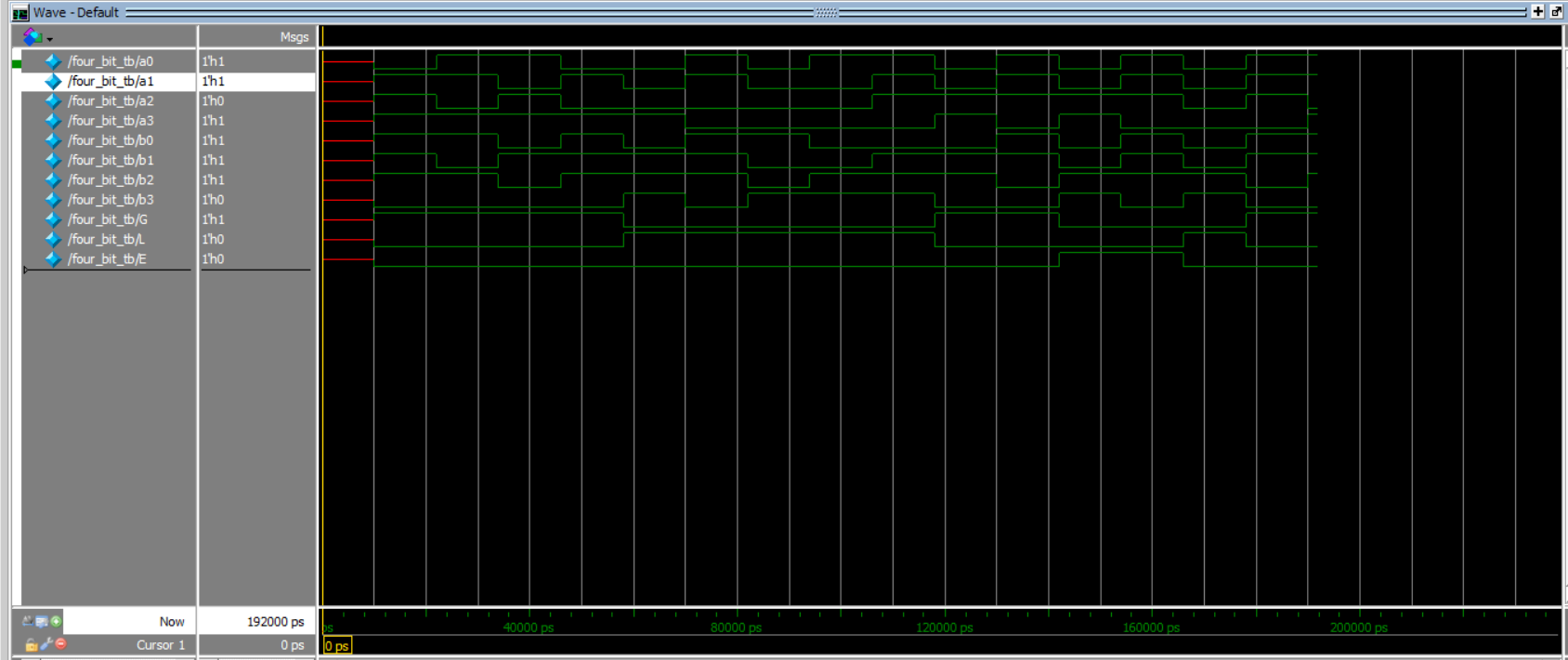
Testbench



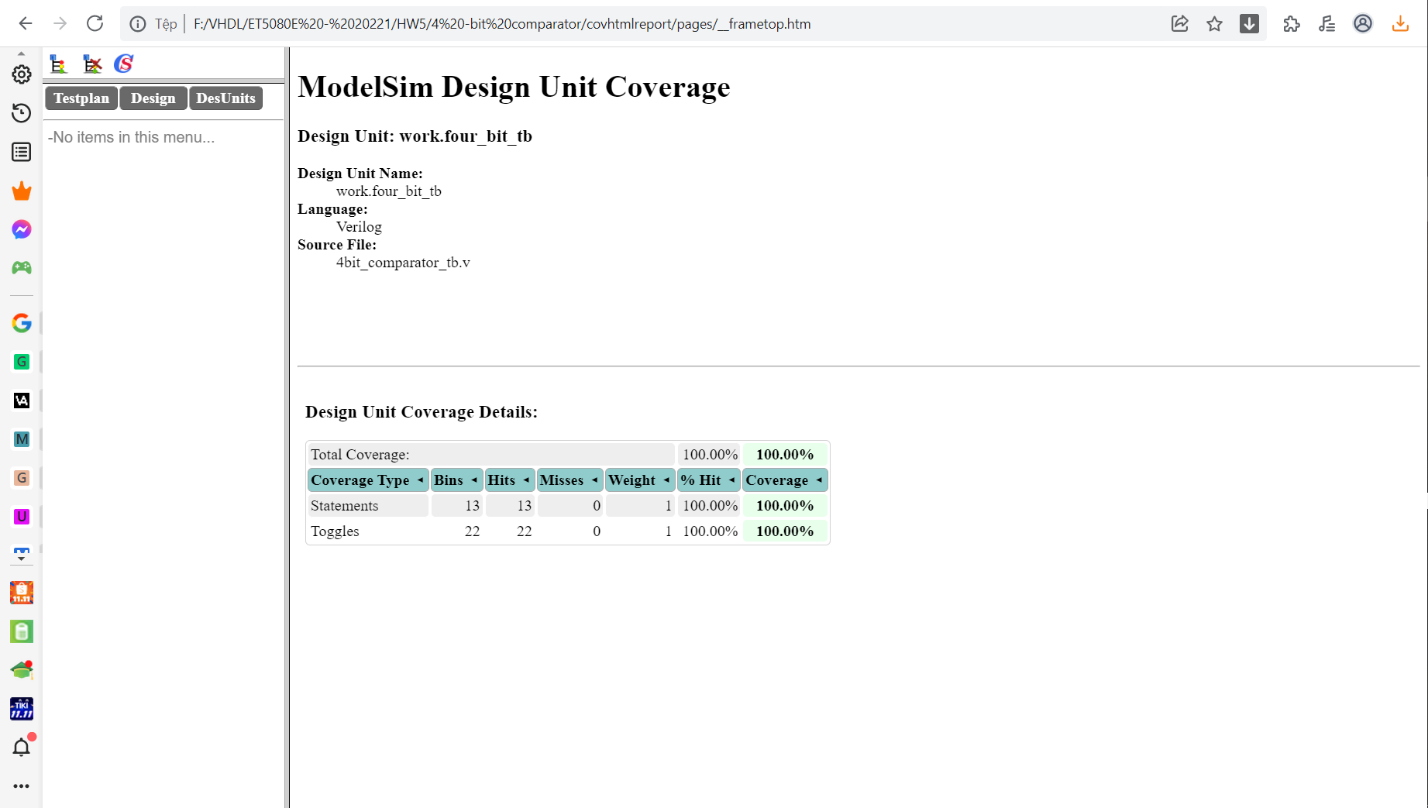
Script



Wave



Report

****