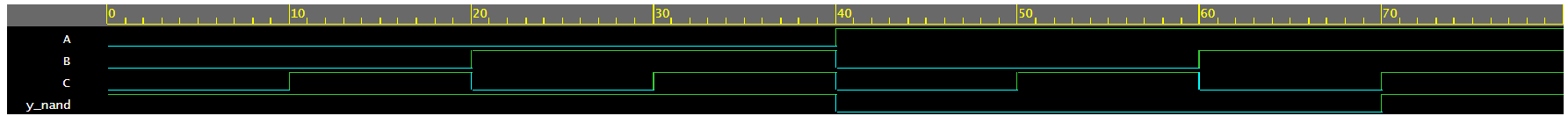
**Experiment – 3**

**Verilog code for designing the boolean expression, F = A' + BC using NAND gate.**

**design.sv** module nand\_implementation(A, B, C, y);  
 input A, B, C;  
 output y;  
 wire bc;  
  
 // Original Function, F = A' + BC  
 // Double negated SOP Function, F = (A(BC'))'  
 nand(bc, B, C);  
 nand(y, A, bc);  
endmodule

**testbench.sv** module nand\_test();  
 reg A, B, C;  
 wire y\_nand;  
  
 nand\_implementation nand\_dut(A, B, C, y\_nand);  
   
 initial begin  
 A = 0; B = 0; C = 0; #10;  
 A = 0; B = 0; C = 1; #10;  
 A = 0; B = 1; C = 0; #10;  
 A = 0; B = 1; C = 1; #10;  
 A = 1; B = 0; C = 0; #10;  
 A = 1; B = 0; C = 1; #10;  
 A = 1; B = 1; C = 0; #10;  
 A = 1; B = 1; C = 1; #10;  
 $finish;  
 end  
  
 initial begin  
 $dumpfile("test.vcd");  
 $dumpvars(1, nand\_test);  
 end  
endmodule

**Output Waveform**



**Verilog code for designing the boolean expression, F = A' + BC using NOR gate.**

**design.sv** module nor\_implementation(A, B, C, y);  
 input A, B, C;  
 output y;  
 wire not\_a, not\_a\_nor\_b, not\_a\_nor\_c;  
  
 // Original Function, F = A' + BC  
 // Double negated POS Function, F = ((A' + B)' + (A' + C)')'  
 nor(not\_a, A, A);  
 nor(not\_a\_nor\_b, not\_a, B);  
 nor(not\_a\_nor\_c, not\_a, C);  
 nor(y, not\_a\_nor\_b, not\_a\_nor\_c);  
endmodule

**testbench.sv** module nor\_test();  
 reg A, B, C;  
 wire y\_nor;  
  
 nor\_implementation nor\_dut(A, B, C, y\_nor);  
   
 initial begin  
 A = 0; B = 0; C = 0; #10;  
 A = 0; B = 0; C = 1; #10;  
 A = 0; B = 1; C = 0; #10;  
 A = 0; B = 1; C = 1; #10;  
 A = 1; B = 0; C = 0; #10;  
 A = 1; B = 0; C = 1; #10;  
 A = 1; B = 1; C = 0; #10;  
 A = 1; B = 1; C = 1; #10;  
 $finish;  
 end  
  
 initial begin  
 $dumpfile("test.vcd");  
 $dumpvars(1, nor\_test);  
 end  
endmodule

**Output Waveform**

