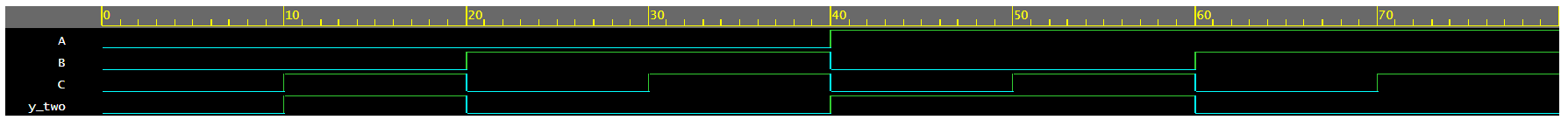
**Experiment – 2**

**Verilog code for designing level two minimization of F = AB' + B'C + AB'C.**

**design.sv**module level\_two\_minimization(A, B, C, y);  
 input A, B, C;  
 output y;  
 wire ac, not\_b;  
  
 // Original Function, F = AB' + B'C + AB'C  
 // Minimized Function, F = B'(A + C)  
 not(not\_b, B);  
 or(ac, A, C);  
 and(y, not\_b, ac);  
endmodule

**testbench.sv**module level\_two\_test();  
 reg A, B, C;  
 wire y\_two;  
  
 level\_two\_minimization two\_dut(A, B, C, y\_two);  
   
 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, level\_two\_test);  
 end  
endmodule

**Output Waveform**



**Verilog code for designing level three minimization of F = A'BC + A'B'C + A'BC'.**

**design.sv**  module level\_three\_minimization(A, B, C, y);  
 input A, B, C;  
 output y;  
 wire bc, not\_a, b\_xor\_c, not\_a\_b\_xor\_c;  
  
 // Original Function, F = A'BC + A'B'C + A'BC'  
 // Minimized Function, F = BC + A'(B ^ C)  
 and(bc, B, C);  
 not(not\_a, A);  
 xor(b\_xor\_c, B, C);  
 and(not\_a\_b\_xor\_c, not\_a, b\_xor\_c);  
 or(y, bc, not\_a\_b\_xor\_c);  
endmodule

**testbench.sv** module level\_three\_test();  
 reg A, B, C;  
 wire y\_three;  
  
 level\_three\_minimization three\_dut(A, B, C, y\_three);  
   
 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, level\_three\_test);  
 end  
endmodule

**Output Waveform**

