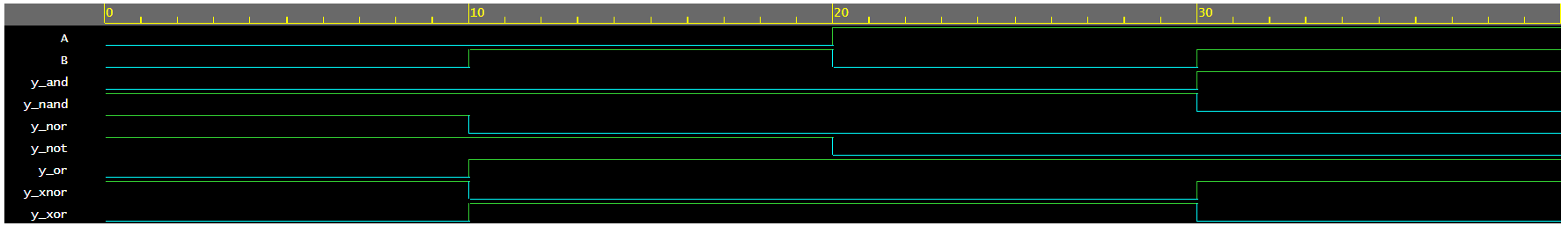
**Experiment – 1**

**Verilog code for designing basic gates by using basic gates.**

**design.sv**module basic\_gates(A, B, y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor);  
 input A, B;  
 output y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor;  
  
 not(y\_not, A);  
 and(y\_and, A, B);  
 or(y\_or, A, B);  
 nand(y\_nand, A, B);  
 nor(y\_nor, A, B);  
 xor(y\_xor, A, B);  
 xnor(y\_xnor, A, B);  
endmodule

**testbench.sv**module basic\_test();  
 reg A, B;  
 wire y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor;  
  
 basic\_gates basic\_dut(A, B, y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor);  
   
 initial begin  
 A = 0; B = 0; #10;  
 A = 0; B = 1; #10;  
 A = 1; B = 0; #10;  
 A = 1; B = 1; #10;  
 $finish;  
 end  
  
 initial begin  
 $dumpfile("test.vcd");  
 $dumpvars(1, basic\_test);  
 end  
endmodule

**Output Waveform**



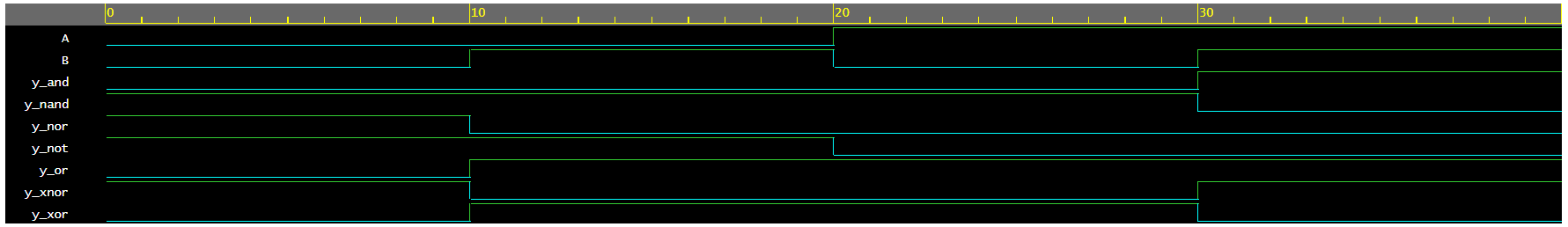
**Verilog code for designing basic gates by using NAND gates.**

**design.sv**module nand\_basic\_gates(A, B, y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor);  
 input A, B;  
 output y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor;  
 wire and\_ab, or\_a\_not, or\_b\_not, xor\_ab, xor\_a\_ab, xor\_b\_ab;  
  
 // NOT  
 nand(y\_not, A, A);  
  
 // AND  
 nand(and\_ab, A, B);  
 nand(y\_and, and\_ab, and\_ab);  
  
 // OR  
 nand(or\_a\_not, A, A);  
 nand(or\_b\_not, B, B);  
 nand(y\_or, or\_a\_not, or\_b\_not);  
  
 // NAND  
 nand(y\_nand, A, B);  
  
 // NOR  
 nand(y\_nor, y\_or, y\_or);  
  
 // XOR  
 nand(xor\_ab, A, B);  
 nand(xor\_a\_ab, A, xor\_ab);  
 nand(xor\_b\_ab, B, xor\_ab);  
 nand(y\_xor, xor\_a\_ab, xor\_b\_ab);  
  
 // XNOR  
 nand(y\_xnor, y\_xor, y\_xor);  
endmodule

**testbench.sv**module nand\_basic\_test();  
 reg A, B;  
 wire y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor;  
  
 nand\_basic\_gates nand\_dut(A, B, y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor);  
   
 initial begin  
 A = 0; B = 0; #10;  
 A = 0; B = 1; #10;  
 A = 1; B = 0; #10;  
 A = 1; B = 1; #10;  
 $finish;  
 end

initial begin  
 $dumpfile("test.vcd");  
 $dumpvars(1, nand\_basic\_test);  
 end  
endmodule

**Output Waveform**



**Verilog code for designing basic gates by using NOR gates.**

**design.sv**module nor\_basic\_gates(A, B, y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor);  
 input A, B;  
 output y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor;  
 wire and\_a\_not, and\_b\_not, or\_ab, xnor\_ab, xnor\_a\_ab, xnor\_b\_ab;  
  
 // NOT  
 nor(y\_not, A, A);  
  
 // AND  
 nor(and\_a\_not, A, A);  
 nor(and\_b\_not, B, B);  
 nor(y\_and, and\_a\_not, and\_b\_not);  
  
 // OR  
 nor(or\_ab, A, B);  
 nor(y\_or, or\_ab, or\_ab);  
  
 // NAND  
 nor(y\_nand, y\_and, y\_and);  
  
 // NOR  
 nor(y\_nor, A, B);  
  
 // XOR  
 nor(y\_xor, y\_xnor, y\_xnor);

// XNOR  
 nor(xnor\_ab, A, B);  
 nor(xnor\_a\_ab, A, xnor\_ab);  
 nor(xnor\_b\_ab, B, xnor\_ab);  
 nor(y\_xnor, xnor\_a\_ab, xnor\_b\_ab);  
endmodule

**testbench.sv**module nor\_basic\_test();  
 reg A, B;  
 wire y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor;  
  
 nor\_basic\_gates nor\_dut(A, B, y\_not, y\_and, y\_or, y\_nand, y\_nor, y\_xor, y\_xnor);  
   
 initial begin  
 A = 0; B = 0; #10;  
 A = 0; B = 1; #10;  
 A = 1; B = 0; #10;  
 A = 1; B = 1; #10;  
 $finish;  
 end  
  
 initial begin  
 $dumpfile("test.vcd");  
 $dumpvars(1, nor\_basic\_test);  
 end  
endmodule

**Output Waveform**

