KVLSI PG DIPLOMA COURSE

NAME: BHAVAN KUMAR

ROLL NO: KVLSI2501110

VERILOG HDL

Introduction

I enrolled in the KVLSI course to build industry-ready skills in digital design and VLSI system development through a structured, hands-on learning approach. In the first two phases of the course, I developed a solid foundation in digital system design and RTL implementation. **Phase 1: Design of Digital Systems** focused on understanding the fundamentals of digital logic, including Boolean algebra, combinational and sequential circuits, and the design of Finite State Machines (FSMs) using both Mealy and Moore models. I learned to optimize state machines, design basic data path components like adders and multiplexers, and conceptualize complete digital subsystems from specifications.

Building on that, **Phase 2: RTL Implementation** introduced me to Verilog HDL for writing synthesizable Register Transfer Level (RTL) code. I gained hands-on experience in modeling digital designs using Verilog constructs, structuring hierarchical modules, and implementing FSMs in code. I also learned to write simple testbenches for simulation and waveform analysis, and understood the importance of writing clean, synthesizable code using good design practices such as proper clocking and reset handling. These two phases together have equipped me with the essential skills to model and implement digital logic systems and prepared me for the next phase in functional verification.

Here I had done the document of some codes that I had written and implemented during my course.

1) Design and implement 2:1 MUX using Gate level modeling and Data flow modeling.

Expression for 2:1 MUX is Y= S'I0+SI

Data flow modeling design code

design.sv

```
// Code your design here
module mux_21(i0,i1,s,y);
  input i0,i1,s;
  output y;
  assign y=((~s)&i0)|(s&i1);
endmodule
```

Test bench

testbench.sv

```
// Code your testbench here
// or browse Examples
module mux_21_test;
  reg i0,i1,s;
  wire y;
  mux_21 dut(.i0(i0),.i1(i1),.s(s),.y(y));
  initial begin
    s=1'b0;i0=1'b0;i1=1'b0;
 #5 s=1'b0;i0=1'b0;i1=1'b1;
 #5 s=1'b0;i0=1'b1;i1=1'b0;
 #5 s=1'b0;i0=1'b1;i1=1'b1;
 #5 s=1'b1;i0=1'b0;i1=1'b0;
 #5 s=1'b1;i0=1'b0;i1=1'b1;
 #5 s=1'b1;i0=1'b1;i1=1'b0;
 #5 s=1'b1;i0=1'b1;i1=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,i0=%b,i1=%b,s=%b,y=%b",$time,i0,i1,s,y);
    $dumpfile("dump.vcd");
    $dumpvars(0,i0,i1,s,y);
  end
endmodule
```

Gate level modeling design code

design.sv

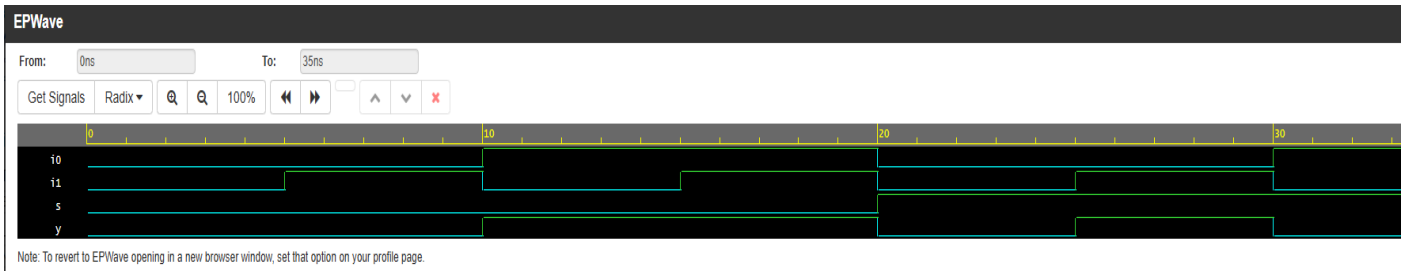
```
// Code your design here
module mux_21(i0,i1,s,y);
  input i0,i1,s;
  output y;
  wire w1,w2,w3;
  not n1(w1,s);
  and a1(w2,i0,w1);
  and a2(w3,i1,s);
  or o1(y,w2,w3);
endmodule
```

Output:

Log | Share

```
[2025-04-09 11:48:07 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 09, 2025 at 07:48:07 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                mux_21_test
Loading snapshot worklib.mux_21_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,i0=0,i1=0,s=0,y=0
sim time=5,i0=0,i1=1,s=0,y=0
sim time=10,i0=1,i1=0,s=0,y=1
sim time=15,i0=1,i1=1,s=0,y=1
sim time=20,i0=0,i1=0,s=1,y=0
sim time=25,i0=0,i1=1,s=1,y=1
sim time=30,i0=1,i1=0,s=1,y=0
sim time=35,i0=1,i1=1,s=1,y=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 09, 2025 at 07:48:08 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-09 11:48:09 UTC] Opening EPWave...
Done
```

Output waveform



2) Design Full adder using Gate level modeling and Data flow modeling.

Expression of a Full adder for Sum is S= A⊕B⊕C

Carry is Cout=AB+BC+CA

Data flow modeling design code

design.sv

```
module full_adder(a,b,c,s,cout);
  input a,b,c;
  output s,cout;
  assign s=a^b^c;
  assign cout=(a&b)|(b&c)|(c&a);
endmodule
```

Test bench

testbench.sv

```
module full_adder_test;
  reg a,b,c;
  wire s,cout;
  full_adder dut(a,b,c,s,cout);
  initial begin
    a=1'b0;b=1'b0;c=1'b0;
#5  a=1'b0;b=1'b0;c=1'b1;
#5  a=1'b0;b=1'b1;c=1'b0;
#5  a=1'b0;b=1'b1;c=1'b1;
#5  a=1'b1;b=1'b0;c=1'b0;
#5  a=1'b1;b=1'b0;c=1'b1;
#5  a=1'b1;b=1'b1;c=1'b0;
#5  a=1'b1;b=1'b1;c=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,c=%b,s=%b,cout=%b",$time,a,b,c,s,cout);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,c,s,cout);
  end
endmodule
```

Gate level modeling design code

design.sv

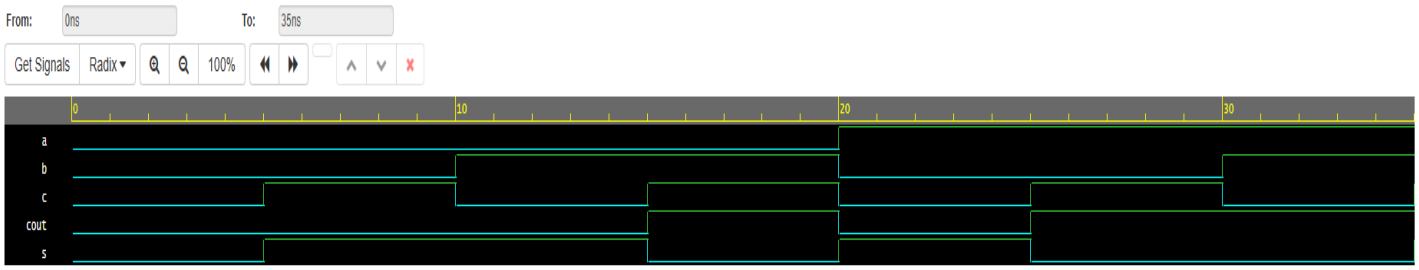
```
module full_adder(a,b,c,s,cout);
  input a,b,c;
  output s,cout;
  wire w1,w2,w3;
  xor x1(s,a,b,c);
  and a1(w1,a,b);
  and a2(w2,b,c);
  and a3(w3,c,a);
  or o1(cout,w1,w2,w3);
endmodule
```

Output:

Log | Share

```
[2025-04-09 12:31:58 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 09, 2025 at 08:31:58 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                full_adder_test
Loading snapshot worklib.full_adder_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,c=0,s=0,cout=0
sim time=5,a=0,b=0,c=1,s=1,cout=0
sim time=10,a=0,b=1,c=0,s=1,cout=0
sim time=15,a=0,b=1,c=1,s=0,cout=1
sim time=20,a=1,b=0,c=0,s=1,cout=0
sim time=25,a=1,b=0,c=1,s=0,cout=1
sim time=30,a=1,b=1,c=0,s=0,cout=1
sim time=35,a=1,b=1,c=1,s=1,cout=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 09, 2025 at 08:31:59 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-09 12:31:59 UTC] Opening EPWave...
Done
```

Output waveform



3) Design Half adder using Gate level modeling and Data flow modeling.

Expression of Half adder for Sum is S=A⊕B

Carry is Cout=AB

Data flow modeling design code

design.sv

```
module half_adder(a,b,s,cout);
  input a,b;
  output s,cout;
  assign s=a^b;
  assign cout=a&b;
endmodule
```

Test bench

testbench.sv

```
module half_adder_test;
  reg a,b;
  wire s,cout;
  half_adder dut(a,b,s,cout);
  initial begin
    a=1'b0;b=1'b0;
#5  a=1'b0;b=1'b1;
#5  a=1'b1;b=1'b0;
#5  a=1'b1;b=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,s=%b,cout=%b",$time,a,b,s,cout);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,s,cout);
  end
endmodule
```

Gate level modeling design code

design.sv

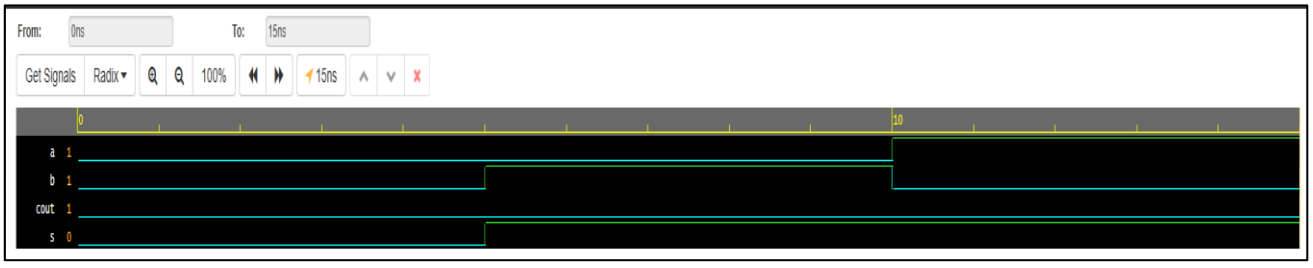
```
// Code your design here
module half_adder(a,b,s,cout);
  input a,b;
  output s,cout;
  wire w1,w2;
  xor x1(s,a,b);
  and a1(w1,a,b);
  and a2(w2,a,b);
  or o1(cout,w1,w2);
endmodule
```

Output:

Log | Share

```
[2025-04-09 13:02:58 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 09, 2025 at 09:02:58 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
  xor x1(s,a,b);;
                |
xmvlog: *W,UEXPSC (design.sv,6|16): Ignored unexpected semicolon following SystemVerilog description keyword (b).
        Top level design units:
                half_adder_test
Loading snapshot worklib.half_adder_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,s=0,cout=0
sim time=5,a=0,b=1,s=1,cout=0
sim time=10,a=1,b=0,s=1,cout=0
sim time=15,a=1,b=1,s=0,cout=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 09, 2025 at 09:02:59 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-09 13:03:00 UTC] Opening EPWave...
Done
```

Output waveform



4) Design Half subtractor using Gate level modeling and Data flow modeling.

Expression of Half subtractor for Difference is Diff= A⊕B

Borrow is Bor= A'B

Data flow modeling design code

design.sv

```
module half_subtractor(a,b,diff,bor);
  input a,b;
  output diff,bor;
  assign diff=a^b;
  assign bor=(~a)&b;
endmodule
```

Test bench

testbench.sv

```
module hallf_subtractor;
  reg a,b;
  wire diff,borr;
  half_subtractor dut(a,b,diff,bor);
  initial begin
    a=1'b0;b=1'b0;
#5  a=1'b0;b=1'b1;
#5  a=1'b1;b=1'b0;
#5  a=1'b1;b=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,diff=%b,bor=%b",$time,a,b,diff,bor);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,diff,bor);
  end
endmodule
```

Gate level modeling design code

design.sv

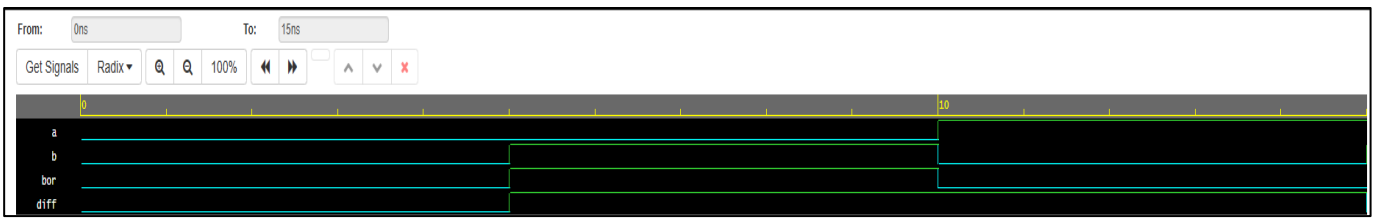
```
module half_subtractor(a,b,diff,bor);
  input a,b;
  output diff,bor;
  wire w1;
  not n1(w1,a);
  xor x1(diff,a,b);
  and a1(bor,w1,b);
endmodule
```

Output:

Log | Share

```
[2025-04-09 13:40:50 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 09, 2025 at 09:40:51 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                hallf_subtractor
Loading snapshot worklib.hallf_subtractor:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,diff=0,bor=0
sim time=5,a=0,b=1,diff=1,bor=1
sim time=10,a=1,b=0,diff=1,bor=0
sim time=15,a=1,b=1,diff=0,bor=0
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 09, 2025 at 09:40:52 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-09 13:40:52 UTC] Opening EPWave...
Done
```

Output waveform



5) Design Full subtractor using Gate level modeling and Data flow modeling.

Expression of Full subtractor for Difference is Diff= A⊕B⊕C
Borrow is Bor= A'B+ BC+ A'C

Data flow modeling design code

design.sv

```
module full_subtractor(a,b,c,diff,bor);
  input a,b,c;
  output diff,bor;
  assign diff=a^b^c;
  assign bor=((~a)&b)|(b&c)|(c&(~a));
endmodule

```

Test bench

testbench.sv

```
module full_subtractor_test;
  reg a,b,c;
  wire diff,bor;
  full_subtractor dut(a,b,c,diff,bor);
  initial begin
    a=1'b0;b=1'b0;c=1'b0;
#5  a=1'b0;b=1'b0;c=1'b1;
#5  a=1'b0;b=1'b1;c=1'b0;
#5  a=1'b0;b=1'b1;c=1'b1;
#5  a=1'b1;b=1'b0;c=1'b0;
#5  a=1'b1;b=1'b0;c=1'b1;
#5  a=1'b1;b=1'b1;c=1'b0;
#5  a=1'b1;b=1'b1;c=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,c=%b,diff=%b,bor=%b",$time,a,b,c,diff,bor);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,c,diff,bor);
  end
endmodule
```

Gate level modeling design code

design.sv

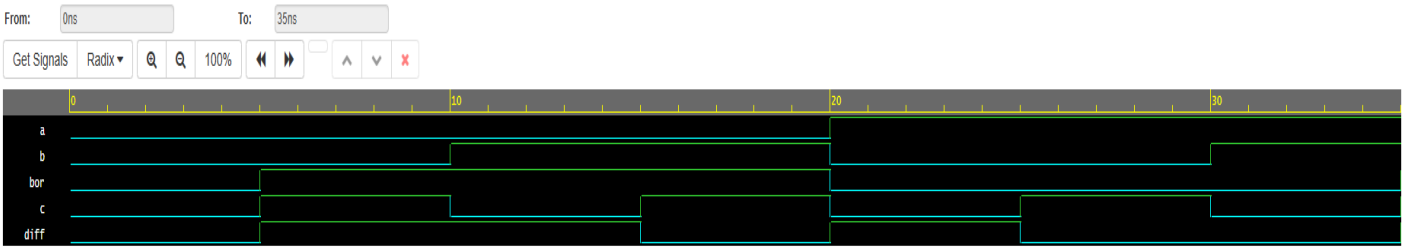
```
module full_subtractor(a,b,c,diff,bor);
  input a,b,c;
  output diff,bor;
  wire w1,w2,w3,w4;
  not n1(w1,a);
  xor x1(diff,a,b,c);
  and a1(w2,w1,b);
  and a2(w3,b,c);
  and a3(w4,c,w1);
  or o1(bor,w2,w3,w4);
endmodule
```

Output:

Log | Share

```
[2025-04-09 14:25:48 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 09, 2025 at 10:25:48 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                full_subtractor_test
Loading snapshot worklib.full_subtractor_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,c=0,diff=0,bor=0
sim time=5,a=0,b=0,c=1,diff=1,bor=1
sim time=10,a=0,b=1,c=0,diff=1,bor=1
sim time=15,a=0,b=1,c=1,diff=0,bor=1
sim time=20,a=1,b=0,c=0,diff=1,bor=0
sim time=25,a=1,b=0,c=1,diff=0,bor=0
sim time=30,a=1,b=1,c=0,diff=0,bor=0
sim time=35,a=1,b=1,c=1,diff=1,bor=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 09, 2025 at 10:25:50 EDT  (total: 00:00:02)
Finding VCD file...
./dump.vcd
[2025-04-09 14:25:50 UTC] Opening EPWave...
Done
```

Output waveform



6) Design 2bit Comparator using Gate level modeling and Data flow modeling.

Expression of 2bit Comparator for Equal is E= (A1⊙B1)(A0⊙B0)
Greater than is G= A1B1'+ A0B1'B0'+ A1A0B0'
Less than is L=A1'B1+A0'B1B0+A1'A0'B0

Data flow modeling design code

design.sv

```
module comparator_2(a1,a0,b1,b0,e,g,l);
  input a1,a0,b1,b0;
  output e,g,l;
  assign e=(~(a1^b1))&(~(a0^b0));
  assign g=(a1&(~b1))|(a1&(~b1)&(~b0))|(a1&a0&(~b0));
  assign l=((~a1)&b1)|((~a0)&b1&b0)|((~a1)&(~a0)&b0);
endmodule
```

Gate level modeling design code

design.sv

```
module comparator_2(a1,a0,b1,b0,e,g,l);
  input a1,a0,b1,b0;
  output e,g,l;
  wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12;
  not N1(w1,a1);
  not N2(w2,a0);
  not N3(w3,b1);
  not N4(w4,b0);
  xnor X1(w5,a1,b1);
  xnor X2(w6,a0,b0);
  and A1(e,w5,w6);
  and A2(w7,a1,w3);
  and A3(w8,a0,w3,w4);
  and A4(w9,a1,a0,w4);
  or O1(g,w7,w8,w9);
  and A5(w10,w1,b1);
  and A6(w11,w2,b1,b0);
  and A7(w12,w1,w2,b0);
  or O2(l,w10,w11,w12);
endmodule
```

Test bench

testbench.sv

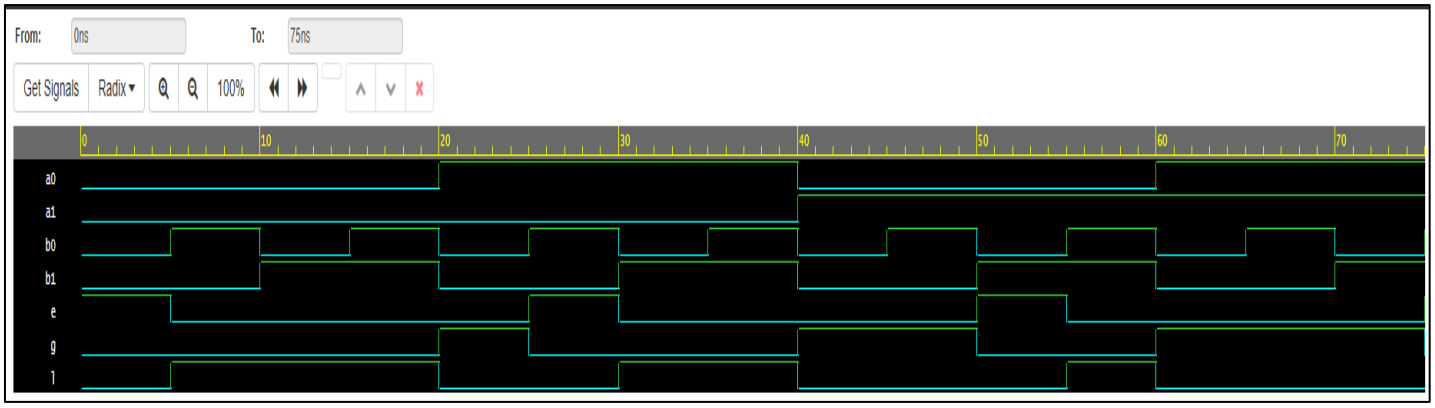
```
module comparator_2_test;
  reg a1,a0,b1,b0;
  wire e,g,l;
  comparator_2 dut(a1,a0,b1,b0,e,g,l);
  initial begin
    a1=1'b0;a0=1'b0;b1=1'b0;b0=1'b0;
#5  a1=1'b0;a0=1'b0;b1=1'b0;b0=1'b1;
#5  a1=1'b0;a0=1'b0;b1=1'b1;b0=1'b0;
#5  a1=1'b0;a0=1'b0;b1=1'b1;b0=1'b1;
#5  a1=1'b0;a0=1'b1;b1=1'b0;b0=1'b0;
#5  a1=1'b0;a0=1'b1;b1=1'b0;b0=1'b1;
#5  a1=1'b0;a0=1'b1;b1=1'b1;b0=1'b0;
#5  a1=1'b0;a0=1'b1;b1=1'b1;b0=1'b1;
#5  a1=1'b1;a0=1'b0;b1=1'b0;b0=1'b0;
#5  a1=1'b1;a0=1'b0;b1=1'b0;b0=1'b1;
#5  a1=1'b1;a0=1'b0;b1=1'b1;b0=1'b0;
#5  a1=1'b1;a0=1'b0;b1=1'b1;b0=1'b1;
#5  a1=1'b1;a0=1'b1;b1=1'b0;b0=1'b0;
#5  a1=1'b1;a0=1'b1;b1=1'b0;b0=1'b1;
#5  a1=1'b1;a0=1'b1;b1=1'b1;b0=1'b0;
#5  a1=1'b1;a0=1'b1;b1=1'b1;b0=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a1=%b,a0=%b,b1=%b,b0=%b,e=%b,g=%b,l=%b",$time,a1,a0,b1,b0,e,g,l);
    $dumpfile("dump.vcd");
    $dumpvars(0,comparator_2_test);
  end
endmodule

```

Output

Log | Share

```
[2025-04-09 16:03:51 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 09, 2025 at 12:03:51 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                comparator_2_test
Loading snapshot worklib.comparator_2_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a1=0,a0=0,b1=0,b0=0,e=1,g=0,l=0
sim time=5,a1=0,a0=0,b1=0,b0=1,e=0,g=0,l=1
sim time=10,a1=0,a0=0,b1=1,b0=0,e=0,g=0,l=1
sim time=15,a1=0,a0=0,b1=1,b0=1,e=0,g=0,l=1
sim time=20,a1=0,a0=1,b1=0,b0=0,e=0,g=0,l=0
sim time=25,a1=0,a0=1,b1=0,b0=1,e=1,g=0,l=0
sim time=30,a1=0,a0=1,b1=1,b0=0,e=0,g=0,l=1
sim time=35,a1=0,a0=1,b1=1,b0=1,e=0,g=0,l=1
sim time=40,a1=1,a0=0,b1=0,b0=0,e=0,g=1,l=0
sim time=45,a1=1,a0=0,b1=0,b0=1,e=0,g=1,l=0
sim time=50,a1=1,a0=0,b1=1,b0=0,e=1,g=0,l=0
sim time=55,a1=1,a0=0,b1=1,b0=1,e=0,g=0,l=1
sim time=60,a1=1,a0=1,b1=0,b0=0,e=0,g=1,l=0
sim time=65,a1=1,a0=1,b1=0,b0=1,e=0,g=1,l=0
sim time=70,a1=1,a0=1,b1=1,b0=0,e=0,g=1,l=0
sim time=75,a1=1,a0=1,b1=1,b0=1,e=1,g=0,l=0
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 09, 2025 at 12:03:53 EDT  (total: 00:00:02)
Finding VCD file...
./dump.vcd
[2025-04-09 16:03:53 UTC] Opening EPWave...
Done
```

Output waveform



7) Design 2 input XOR using NAND gate using Gate level modeling and Data flow modeling.

Expression of 2 input XOR is Y=A'B+AB'

Data flow modeling design code

design.sv

```
module XOR_nand_2(y,a,b);
  input a,b;
  output y;
  assign y=((~a)&b)|(a&(~b));
endmodule
```

Test bench

testbench.sv

```
module XOR_nand_2_test;
  reg a,b;
  wire y;
  XOR_nand_2 dut(y,a,b);
  initial begin
    a=1'b0;b=1'b0;
#5  a=1'b0;b=1'b1;
#5  a=1'b1;b=1'b0;
#5  a=1'b1;b=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,y=%b",$time,a,b,y);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,y);
  end
endmodule
```

Gate level modeling design code

design.sv

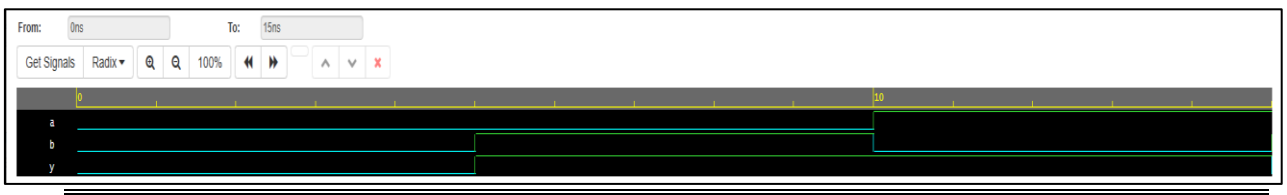
```
module XOR_nand_2(y,a,b);
  input a,b;
  output y;
  wire w1,w2,w3;
  nand N1(w1,a,b);
  nand N2(w2,w1,a);
  nand N3(w3,w1,b);
  nand N4(y,w2,w3);
endmodule
```

Output:

Log | Share

```
[2025-04-09 17:43:12 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 09, 2025 at 13:43:13 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                XOR_nand_2_test
Loading snapshot worklib.XOR_nand_2_test:sv ..... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,y=0
sim time=5,a=0,b=1,y=1
sim time=10,a=1,b=0,y=1
sim time=15,a=1,b=1,y=0
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 09, 2025 at 13:43:14 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-09 17:43:15 UTC] Opening EPWave...
Done
```

Output waveform



8) Design and implement 2:1 MUX using
   a. NAND gate using Gate level modeling and Data flow modeling.
   b. Tristate buffer using Gate level modeling and Data flow modeling

Expression for 2:1 MUX is Y= S'I0+SI

Data flow modeling design code

design.sv

```
module mux2_1nand(y,i0,i1,s);
  input i0,i1,s;
  output y;
  assign y=((~s)&i0)|(s&i1);
endmodule
```

a. Gate level modeling design code using NAND gate

design.sv

```
module mux2_1nand(y,i0,i1,s);
  input i0,i1,s;
  output y;
  wire w1,w2,w3;
  nand n1(w1,s);
  nand n2(w2,i0,w1);
  nand n3(w3,i1,s);
  nand n4(y,w2,w3);
endmodule
```

Test bench

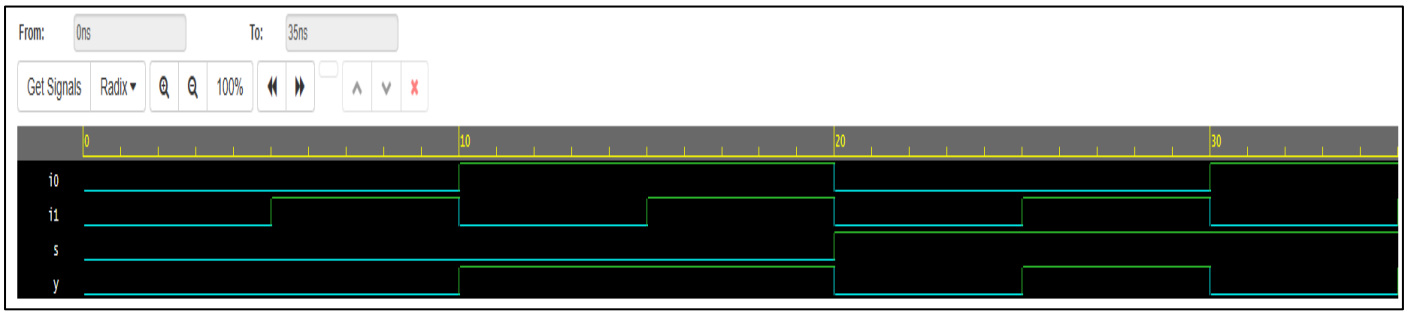
testbench.sv

```
module mux2_1nand_test;
  reg i0,i1,s;
  wire y;
  mux2_1nand dut(y,i0,i1,s);
  initial begin
    s=1'b0;i0=1'b0;i1=1'b0;
 #5 s=1'b0;i0=1'b0;i1=1'b1;
 #5 s=1'b0;i0=1'b1;i1=1'b0;
 #5 s=1'b0;i0=1'b1;i1=1'b1;
 #5 s=1'b1;i0=1'b0;i1=1'b0;
 #5 s=1'b1;i0=1'b0;i1=1'b1;
 #5 s=1'b1;i0=1'b1;i1=1'b0;
 #5 s=1'b1;i0=1'b1;i1=1'b1;
  end
    initial begin
    $monitor("sim time=%0t,i0=%b,i1=%b,s=%b,y=%b",$time,i0,i1,s,y);
    $dumpfile("dump.vcd");
    $dumpvars(0,i0,i1,s,y);
  end
endmodule
```

Output:

```
[2025-04-09 18:47:06 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 09, 2025 at 14:47:07 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                mux2_1nand_test
Loading snapshot worklib.mux2_1nand_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,i0=0,i1=0,s=0,y=0
sim time=5,i0=0,i1=1,s=0,y=0
sim time=10,i0=1,i1=0,s=0,y=1
sim time=15,i0=1,i1=1,s=0,y=1
sim time=20,i0=0,i1=0,s=1,y=0
sim time=25,i0=0,i1=1,s=1,y=1
sim time=30,i0=1,i1=0,s=1,y=0
sim time=35,i0=1,i1=1,s=1,y=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 09, 2025 at 14:47:08 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-09 18:47:08 UTC] Opening EPWave...
Done
```

Output waveform



b. Gate level modeling design code using tristate buffer

design.sv

```
// Code your design here
module mux_tristate(y,i0,i1,s);
  input i0,i1,s;
  output y;
  bufif0 b1(y,i0,s);
  bufif1 b2(y,i1,s);
endmodule
```

Test bench

testbench.sv

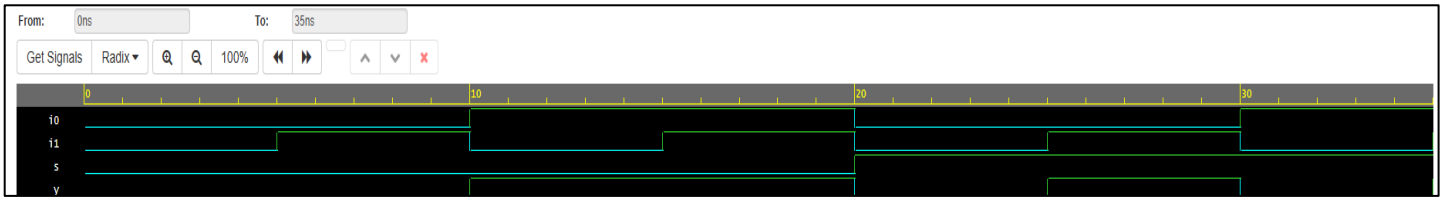
```
// Code your testbench here
// or browse Examples
module mux_tristate_test;
  reg i0,i1,s;
  wire y;
  mux_tristate dut(y,i0,i1,s);
  initial begin
   /*s=1'b0;i0=1'b0;i1=1'b0;*/{s,i0,i1}=0;
#5 /*s=1'b0;i0=1'b0;i1=1'b1;*/{s,i0,i1}=1;
#5 /*s=1'b0;i0=1'b1;i1=1'b0;*/{s,i0,i1}=2;
#5 /*s=1'b0;i0=1'b1;i1=1'b1;*/{s,i0,i1}=3;
#5 /*s=1'b1;i0=1'b0;i1=1'b0;*/{s,i0,i1}=4;
#5 /*s=1'b1;i0=1'b0;i1=1'b1;*/{s,i0,i1}=5;
#5 /*s=1'b1;i0=1'b1;i1=1'b0;*/{s,i0,i1}=6;
#5 /*s=1'b1;i0=1'b1;i1=1'b1;*/{s,i0,i1}=7;
  end
    initial begin
    $monitor("sim time=%0t,i0=%b,i1=%b,s=%b,y=%b",$time,i0,i1,s,y);
    $dumpfile("dump.vcd");
      $dumpvars(1,mux_tristate_test);
  end
endmodule
```

Output

Log | Share

```
[2025-04-14 11:40:48 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 14, 2025 at 07:40:48 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                mux_tristate_test
Loading snapshot worklib.mux_tristate_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,i0=0,i1=0,s=0,y=0
sim time=5,i0=0,i1=1,s=0,y=0
sim time=10,i0=1,i1=0,s=0,y=1
sim time=15,i0=1,i1=1,s=0,y=1
sim time=20,i0=0,i1=0,s=1,y=0
sim time=25,i0=0,i1=1,s=1,y=1
sim time=30,i0=1,i1=0,s=1,y=0
sim time=35,i0=1,i1=1,s=1,y=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 14, 2025 at 07:40:50 EDT  (total: 00:00:02)
Finding VCD file...
./dump.vcd
[2025-04-14 11:40:50 UTC] Opening EPWave...
Done
```

Output waveform



9) Design and implement Even and Odd Parity generator using Gate level and Data flow modeling.
Expression for Even parity generator is = A⊕B⊕C
Expression for Odd parity generator is = A⊙B⊙C

Data flow modeling design code

design.sv

```
// Code your design here
module parity_generator(podd,peven,a,b,c);
  input a,b,c;
  output podd,peven;
  assign peven=a^b^c;
  assign podd=~(a^b^c);
endmodule
```

Test bench

testbench.sv

```
// Code your testbench here
// or browse Examples
module parity_generator_test;
  reg a,b,c;
  wire podd,peven;
  parity_generator dut(podd,peven,a,b,c);
  initial begin
    a=1'b0;b=1'b0;c=1'b0;
#5  a=1'b0;b=1'b0;c=1'b1;
#5  a=1'b0;b=1'b1;c=1'b0;
#5  a=1'b0;b=1'b1;c=1'b1;
#5  a=1'b1;b=1'b0;c=1'b0;
#5  a=1'b1;b=1'b0;c=1'b1;
#5  a=1'b1;b=1'b1;c=1'b0;
#5  a=1'b1;b=1'b1;c=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,c=%b,podd=%b,peven=%b",$time,a,b,c,podd,peven);
    $dumpfile("dump.vcd");
    $dumpvars(0,podd,peven,a,b,c);
  end
endmodule
```

Gate level modeling design code

design.sv

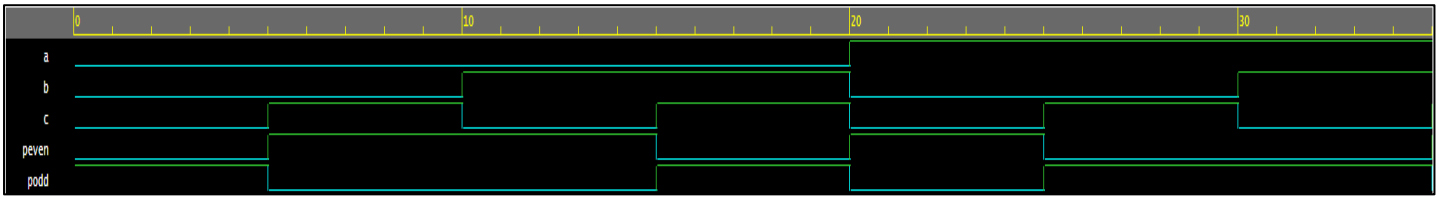
```
// Code your design here
module parity_generator(podd,peven,a,b,c);
  input a,b,c;
  output podd,peven;
  xor x1(peven,a,b,c);
  xnor x2(podd,a,b,c);
endmodule
```

Output:

Log | Share

```
[2025-04-10 14:56:52 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 10, 2025 at 10:56:52 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                parity_generator_test
Loading snapshot worklib.parity_generator_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,c=0,podd=1,peven=0
sim time=5,a=0,b=0,c=1,podd=0,peven=1
sim time=10,a=0,b=1,c=0,podd=0,peven=1
sim time=15,a=0,b=1,c=1,podd=1,peven=0
sim time=20,a=1,b=0,c=0,podd=0,peven=1
sim time=25,a=1,b=0,c=1,podd=1,peven=0
sim time=30,a=1,b=1,c=0,podd=1,peven=0
sim time=35,a=1,b=1,c=1,podd=0,peven=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 10, 2025 at 10:56:53 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-10 14:56:54 UTC] Opening EPWave...
Done
```

Output waveform



10) Design and implement Even and Odd Parity checker using Gate level modeling and Data flow modeling.
Expression for Even parity checker is = A⊕B⊕C⊕P
Expression for Odd parity checker is = A⊙B⊙C⊙P

Data flow modeling design cod

design.sv

```
// Code your design here
module parity_checker(podd,peven,a,b,c,p);
  input a,b,c,p;
  output podd,peven;
  assign peven=a^b^c^p;
  assign podd=~(a^b^c^p);
endmodule
```

Test bench

testbench.sv

```
// Code your testbench here
// or browse Examples
module parity_checker_test;
  reg a,b,c,p;
  wire podd,peven;
  parity_checker dut(podd,peven,a,b,c,p);
  initial begin
    a=1'b0;b=1'b0;c=1'b0;p=1'b0;
#5  a=1'b0;b=1'b0;c=1'b0;p=1'b1;
#5  a=1'b0;b=1'b0;c=1'b1;p=1'b0;
#5  a=1'b0;b=1'b0;c=1'b1;p=1'b1;
#5  a=1'b0;b=1'b1;c=1'b0;p=1'b0;
#5  a=1'b0;b=1'b1;c=1'b0;p=1'b1;
#5  a=1'b0;b=1'b1;c=1'b1;p=1'b0;
#5  a=1'b0;b=1'b1;c=1'b1;p=1'b1;
#5  a=1'b1;b=1'b0;c=1'b0;p=1'b0;
#5  a=1'b1;b=1'b0;c=1'b0;p=1'b1;
#5  a=1'b1;b=1'b0;c=1'b1;p=1'b0;
#5  a=1'b1;b=1'b0;c=1'b1;p=1'b1;
#5  a=1'b1;b=1'b1;c=1'b0;p=1'b0;
#5  a=1'b1;b=1'b1;c=1'b0;p=1'b1;
#5  a=1'b1;b=1'b1;c=1'b1;p=1'b0;
#5  a=1'b1;b=1'b1;c=1'b1;p=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,c=%b,p=%b,podd=%b,peven=%b",$time,a,b,c,p,podd,peven);
    $dumpfile("dump.vcd");
    $dumpvars(0,podd,peven,a,b,c,p);
  end
endmodule
```

Gate level modeling design code

design.sv

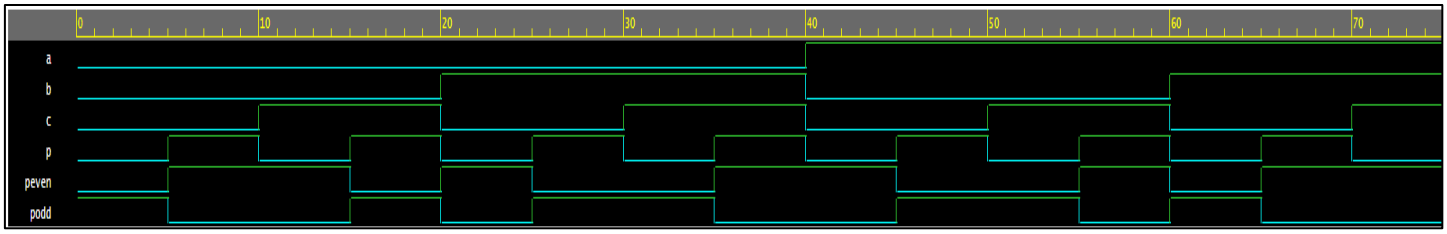
```
// Code your design here
module parity_checker(podd,peven,a,b,c,p);
  input a,b,c,p;
  output podd,peven;
  assign peven=a^b^c^p;
  assign podd=~(a^b^c^p);
endmodule
```

Output:

Log | Share

```
[2025-04-10 15:30:41 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 10, 2025 at 11:30:41 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                parity_checker_test
Loading snapshot worklib.parity_checker_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,c=0,p=0,podd=1,peven=0
sim time=5,a=0,b=0,c=0,p=1,podd=0,peven=1
sim time=10,a=0,b=0,c=1,p=0,podd=0,peven=1
sim time=15,a=0,b=0,c=1,p=1,podd=1,peven=0
sim time=20,a=0,b=1,c=0,p=0,podd=0,peven=1
sim time=25,a=0,b=1,c=0,p=1,podd=1,peven=0
sim time=30,a=0,b=1,c=1,p=0,podd=1,peven=0
sim time=35,a=0,b=1,c=1,p=1,podd=0,peven=1
sim time=40,a=1,b=0,c=0,p=0,podd=0,peven=1
sim time=45,a=1,b=0,c=0,p=1,podd=1,peven=0
sim time=50,a=1,b=0,c=1,p=0,podd=1,peven=0
sim time=55,a=1,b=0,c=1,p=1,podd=0,peven=1
sim time=60,a=1,b=1,c=0,p=0,podd=1,peven=0
sim time=65,a=1,b=1,c=0,p=1,podd=0,peven=1
sim time=70,a=1,b=1,c=1,p=0,podd=0,peven=1
sim time=75,a=1,b=1,c=1,p=1,podd=1,peven=0
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 10, 2025 at 11:30:42 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-10 15:30:42 UTC] Opening EPWave...
Done
```

Output waveform



11) Design and implement 4:1 mux using Data flow modeling in vector form.

Expression for 4:1 mux is = S1'S0'I0+ S1'S0I1+ S1S0'I2+ S1S0I3

Design code

design.sv

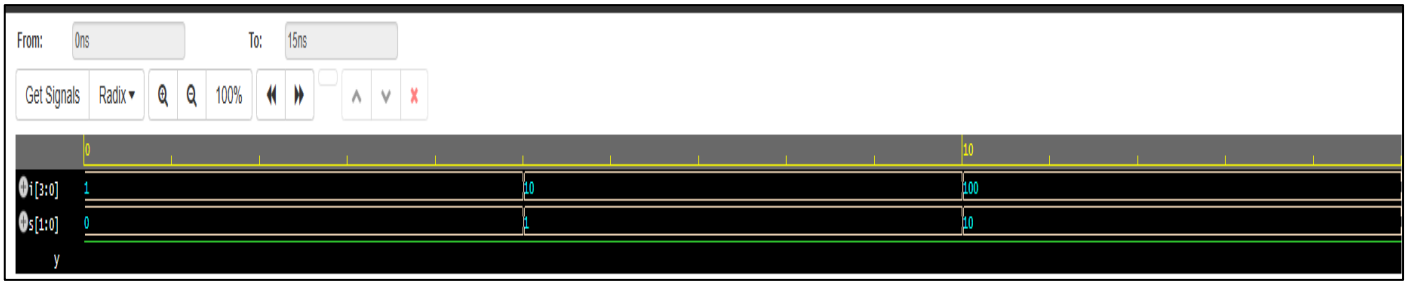
```
// Code your design here
module mux4_1(y,i,s);
  input [3:0]i;
  input [1:0]s;
  output y;
  assign y=(~(s[1])&(~(s[0]))&i[0])|(~(s[1])&s[0]&i[1])|(s[1]&(~(s[0]))&i[2])|(s[1]&s[0]&i[3]);
endmodule
```

Test bench

testbench.sv

```
// Code your testbench here
// or browse Examples
module mux4_1_test;
  reg [3:0]i;
  reg [1:0]s;
  wire y;
  mux4_1 dut(y,i,s);
  initial begin
    {s}=0;{i}=1;
#5  {s}=1;{i}=2;
#5  {s}=2;{i}=4;
#5  {s}=3;{i}=8;
   end
 initial begin
    $monitor("sim time=%0t,i=%b,s=%b,y=%b",$time,i,s,y);
    $dumpfile("dump.vcd");
    $dumpvars(1,mux4_1_test);
  end
endmodule
```

Output waveform



Output

Log | Share

```
[2025-04-15 15:23:06 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:23:06 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                mux4_1_test
Loading snapshot worklib.mux4_1_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,i=0001,s=00,y=1
sim time=5,i=0010,s=01,y=1
sim time=10,i=0100,s=10,y=1
sim time=15,i=1000,s=11,y=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:23:07 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-15 15:23:07 UTC] Opening EPWave...
Done
```

12) Design and implement 4bit comparator using Gate level modeling in vector form.

A = A (3) A (2) A (1) A (0)
B = B (3) B (2) B (1) B (0)

x (i) = A (i). B (i) + A (i)'. B (i)'

Expression is
A>B = A (3). B (3)' + x (3). A (2). B (2)' + x (3). x (2). A (1). B (1)' + x (3). x (2). x (1). A (0). B (0)'
A<B = A (3)'. B (3) + x (3). A (2)'. B (2) + x (3). x (2). A (1)'. B (1) + x (3). x (2). x (1). A (0)'. B (0)
A=B = x (3).x (2).x (1).x (0)

Design code

testbench.sv

```
// Code your testbench here
// or browse Examples
module magnitude_comparator_test;
  reg [3:0]a;
  reg [3:0]b;
  wire gt,le,e;
  magnitude_comparator dut(gt,le,e,a,b);
  initial begin
    {a,b}={4'd0,4'd0};
#5  {a,b}={4'd0,4'd1};
#5  {a,b}={4'd1,4'd0};
#5  {a,b}={4'd1,4'd1};
#5  {a,b}={4'd1,4'd2};
#5  {a,b}={4'd2,4'd1};
#5  {a,b}={4'd2,4'd2};
    end
    initial begin
      $monitor("sim time=%0t,\ta[3]=%b,\ta[2]=%b,\ta[1]=%b,\ta[0]=%b,\tb[3]=%b,\tb[2]=%b,\tb[1]=%b,\tb[0]=%b,\tgt=%b,\tle=%b,\te=%b",$time,a[3],a[2],a[1],a[0],b[3],b[2],b[1],b[0],gt,le,e);
      $dumpfile("dump.vcd");
      $dumpvars(1,magnitude_comparator_test);
    end
endmodule
```

Test bench

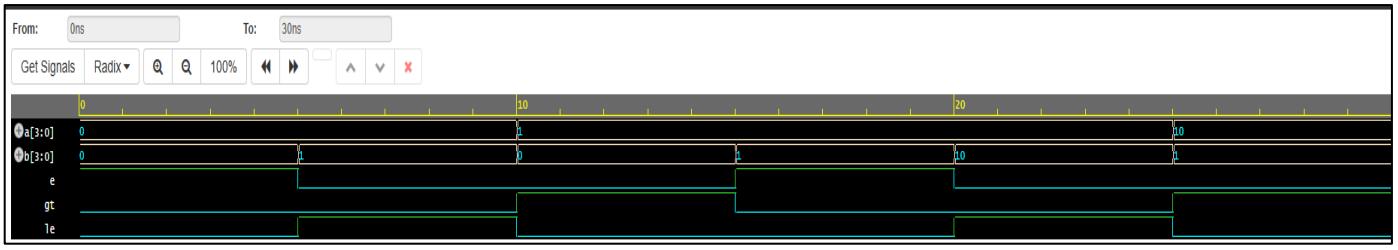
design.sv

```
// Code your design here
module magnitude_comparator(gt,le,e,a,b);//gt=greater than,le=lessthan,e=eual,a=A,b=B
input [3:0]a;
input [3:0]b;
output gt,le,e;//gt=A_gt_B,le=A_lt_B,e=A_eq_B
  wire w[7:0],x[3:0],y[8:0];
  not n1(w[7],a[3]);//a3'
  not n2(w[6],a[2]);//a2'
  not n3(w[5],a[1]);//a1'
  not n4(w[4],a[0]);//a0'
  not n5(w[3],b[3]);//b3'
  not n6(w[2],b[2]);//b2'
  not n7(w[1],b[1]);//b1'
  not n8(w[0],b[0]);//b0'
  xnor X1(x[3],a[3],b[3]);
  xnor X2(x[2],a[2],b[2]);
  xnor X3(x[1],a[1],b[1]);
  xnor X4(x[0],a[0],b[0]);
  and A1(y[8],a[3],w[3]);
  and A2(y[7],a[2],w[2],x[3]);
  and A3(y[6],a[1],w[1],x[3],x[2]);
  and A4(y[5],a[0],w[0],x[3],x[2],x[1]);
  or O1(gt,y[8],y[7],y[6],y[5]);//greater than
  and A5(y[4],w[7],b[3]);
  and A6(y[3],w[6],b[2],x[3]);
  and A7(y[2],w[5],b[1],x[3],x[2]);
  and A8(y[1],w[4],b[0],x[3],x[2],x[1]);
  or O2(le,y[4],y[3],y[2],y[1]);//less than
and A9(e,x[3],x[2],x[1],x[0]);//equal
endmodule
```

Output

```
Log  Share
[2025-04-15 15:36:27 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:36:27 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                magnitude_comparator_test
Loading snapshot worklib.magnitude_comparator_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,     a[3]=0, a[2]=0, a[1]=0, a[0]=0, b[3]=0, b[2]=0, b[1]=0, b[0]=0, gt=0,   le=0,   e=1
sim time=5,     a[3]=0, a[2]=0, a[1]=0, a[0]=0, b[3]=0, b[2]=0, b[1]=0, b[0]=1, gt=0,   le=1,   e=0
sim time=10,    a[3]=0, a[2]=0, a[1]=0, a[0]=1, b[3]=0, b[2]=0, b[1]=0, b[0]=0, gt=1,   le=0,   e=0
sim time=15,    a[3]=0, a[2]=0, a[1]=0, a[0]=1, b[3]=0, b[2]=0, b[1]=0, b[0]=1, gt=0,   le=0,   e=1
sim time=20,    a[3]=0, a[2]=0, a[1]=0, a[0]=1, b[3]=0, b[2]=0, b[1]=1, b[0]=0, gt=0,   le=1,   e=0
sim time=25,    a[3]=0, a[2]=0, a[1]=1, a[0]=0, b[3]=0, b[2]=0, b[1]=0, b[0]=1, gt=1,   le=0,   e=0
sim time=30,    a[3]=0, a[2]=0, a[1]=1, a[0]=0, b[3]=0, b[2]=0, b[1]=1, b[0]=0, gt=0,   le=0,   e=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:36:29 EDT  (total: 00:00:02)
Finding VCD file...
./dump.vcd
[2025-04-15 15:36:29 UTC] Opening EPWave...
Done
```

Output waveform



13) Design and implement 4bit carry look ahead adder using Gate level modeling in vector form.

Expressions are

G0=A0B0	P0= A0⊕B0	C0=G0+P0Cin	S0=P0⊕Cin
G1=A1B1	P1= A1⊕B1	C1=G1+P1C0	S1=P1⊕C0
G2=A2B2	P2= A2⊕B2	C2=G2+P2C1	S2=P2⊕C1
G3=A3B3	P3= A3⊕B3	C3=G3+P3C2	S3=P3⊕C2

S0, S1, S2, S3 = SUM

C3 = CARRY

Design code

design.sv

```
// Code your design here
module carry_4_adder(sum,carry,a,b,cin);
  input [3:0]a;
  input [3:0]b;
  input cin;
  output [3:0]sum;
  output carry;
  wire [3:0]c;
  wire [3:0]p;
  wire [3:0]g;
  assign g=a&b;
  assign p=a^b;
  assign c[0]=g[0]|(p[0]&cin);
  assign c[1]=g[1]|(p[1]&c[0]);
  assign c[2]=g[2]|(p[2]&c[1]);
  assign c[3]=g[3]|(p[3]&c[2]);
  assign sum[0]=p[0]^cin;
  assign sum[1]=p[1]^c[0];
  assign sum[2]=p[2]^c[1];
  assign sum[3]=p[3]^c[2];
  assign carry=c[3];
endmodule

```

Test bench

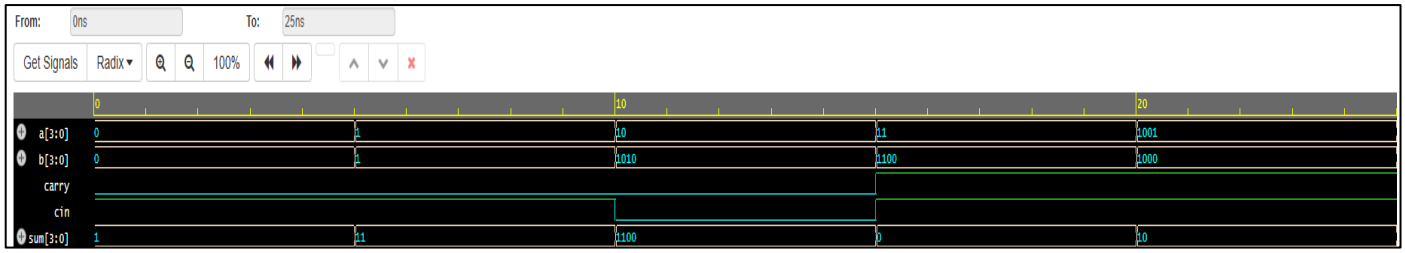
testbench.sv

```
// Code your testbench here
// or browse Examples
module carry_4_adder_test;
  reg [3:0]a;
  reg [3:0]b;
  reg cin;
  wire [3:0]sum;
  wire carry;
  carry_4_adder dut(sum,carry,a,b,cin);
  initial begin
    {a,b,cin}={4'b0,4'b0,1'b1};
#5  {a,b,cin}={4'b1,4'b1,1'b1};
#5  {a,b,cin}={4'b0010,4'b1010,1'b0};
#5  {a,b,cin}={4'b0011,4'b1100,1'b1};
#5  {a,b,cin}={4'd9,4'd8,1'b1};
#5  {a,b,cin}={4'ha,4'hb,1'b1};
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,cin=%b,sum=%b,carry=%b",$time,a,b,cin,sum,carry);
    $dumpfile("dump.vcd");
      $dumpvars(1,carry_4_adder_test);
  end
endmodule

```

Output

```
Log  Share
[2025-04-15 15:30:23 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:30:23 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                carry_4_adder_test
Loading snapshot worklib.carry_4_adder_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0000,b=0000,cin=1,sum=0001,carry=0
sim time=5,a=0001,b=0001,cin=1,sum=0011,carry=0
sim time=10,a=0010,b=1010,cin=0,sum=1100,carry=0
sim time=15,a=0011,b=1100,cin=1,sum=0000,carry=1
sim time=20,a=1001,b=1000,cin=1,sum=0010,carry=1
sim time=25,a=1010,b=1011,cin=1,sum=0110,carry=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:30:24 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-15 15:30:24 UTC] Opening EPWave...
Done
```

Output waveform



14) Design and implement Decoder 2:4 using Data flow modeling in vector form.

Expression is
Y0=I1'I0'
Y1=I1'I0
Y2=I1I0'
Y3=I1I0

Design code

design.sv

```
// Code your design here
module decoder2_4(y,i);
  input [1:0]i;
  output [3:0]y;
  assign y[0]=(~i[1])&(~i[0]);
  assign y[1]=(~i[1])&(i[0]);
  assign y[2]=(i[1])&(~i[0]);
  assign y[3]=(i[1])&(i[0]);
endmodule
```

Test bench

testbench.sv

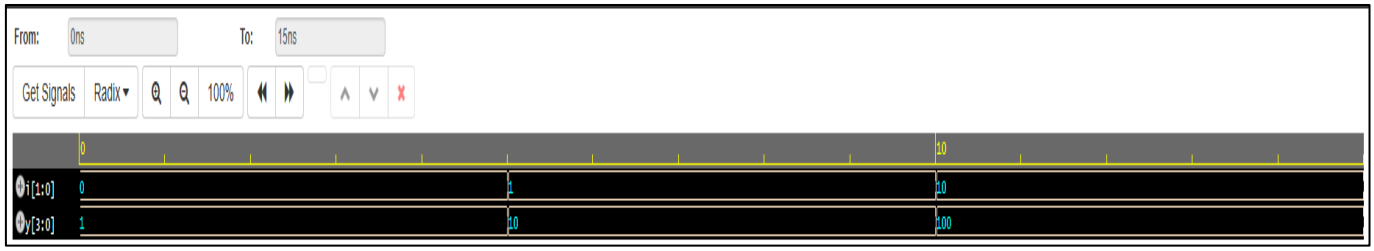
```
// Code your testbench here
// or browse Examples
module decoder2_4_test;
  reg [1:0]i;
  wire [3:0]y;
  decoder2_4 dut(y,i);
  initial begin
    {i}=0;
#5  {i}=1;
#5  {i}=2;
#5  {i}=3;
  end
  initial begin
    $monitor("sim time=%0t,i=%b,y=%b",$time,i,y);
    $dumpfile("dump.vcd");
    $dumpvars(1,decoder2_4_test);
  end
endmodule
```

Output

Log | Share

```
[2025-04-15 15:27:19 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:27:19 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                decoder2_4_test
Loading snapshot worklib.decoder2_4_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,i=00,y=0001
sim time=5,i=01,y=0010
sim time=10,i=10,y=0100
sim time=15,i=11,y=1000
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:27:20 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-15 15:27:20 UTC] Opening EPWave...
Done
```

Output waveform



15) Design and implement Gray to binary converter using Data flow modeling in vector form.

Expression is
B3 = G3
B2 = B3⊕G2
B1 = B2⊕G1
B0 = B1⊕G0

Design code

design.sv

```
// Code your design here
module gray4_binary(b,g);
  input [3:0]g;
  output [3:0]b;
  assign b[3]=g[3];
  assign b[2]=b[3]^g[2];
  assign b[1]=b[2]^g[1];
  assign b[0]=b[1]^g[0];
endmodule
```

Test bench

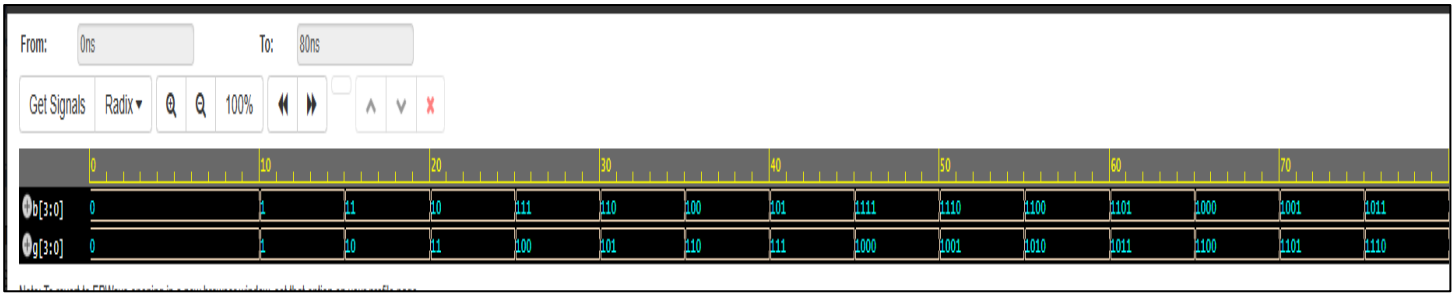
testbench.sv

```
// Code your testbench here
// or browse Examples
module gray4_binary_test;
  reg [3:0]g;
  wire [3:0]b;
  gray4_binary dut(b,g);
  initial begin
    {g}={4'b0000};
    #5  {g}={4'b0000};
    #5  {g}={4'b0001};
    #5  {g}={4'b0010};
    #5  {g}={4'b0011};
    #5  {g}={4'b0100};
    #5  {g}={4'b0101};
    #5  {g}={4'b0110};
    #5  {g}={4'b0111};
    #5  {g}={4'b1000};
    #5  {g}={4'b1001};
    #5  {g}={4'b1010};
    #5  {g}={4'b1011};
    #5  {g}={4'b1100};
    #5  {g}={4'b1101};
    #5  {g}={4'b1110};
    #5  {g}={4'b1111};
  end
  initial begin
    $monitor("sim time+%0t,g=%b,b=%b",$time,g,b);
    $dumpfile("dump.vcd");
    $dumpvars(1,gray4_binary_test);
  end
endmodule
```

Output

```
Log  Share
[2025-04-15 15:34:47 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:34:47 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                gray4_binary_test
Loading snapshot worklib.gray4_binary_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time+0,g=0000,b=0000
sim time+10,g=0001,b=0001
sim time+15,g=0010,b=0011
sim time+20,g=0011,b=0010
sim time+25,g=0100,b=0111
sim time+30,g=0101,b=0110
sim time+35,g=0110,b=0100
sim time+40,g=0111,b=0101
sim time+45,g=1000,b=1111
sim time+50,g=1001,b=1110
sim time+55,g=1010,b=1100
sim time+60,g=1011,b=1101
sim time+65,g=1100,b=1000
sim time+70,g=1101,b=1001
sim time+75,g=1110,b=1011
sim time+80,g=1111,b=1010
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:34:48 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-15 15:34:49 UTC] Opening EPWave...
Done
```

Output waveform



16) Design and implement Binary to Gray converter using Data flow modeling in vector form.

Expression is
G3 = B3
G2 = B3⊕B2
G1 = B2⊕B1
G0 = B1⊕B0

Design code

design.sv

```
// Code your design here
module binary4_gray(g,b);
  input [3:0]b;
  output [3:0]g;
  assign g[3]=b[3];
  assign g[2]=b[3]^b[2];
  assign g[1]=b[2]^b[1];
  assign g[0]=b[1]^b[0];
endmodule
```

Test bench

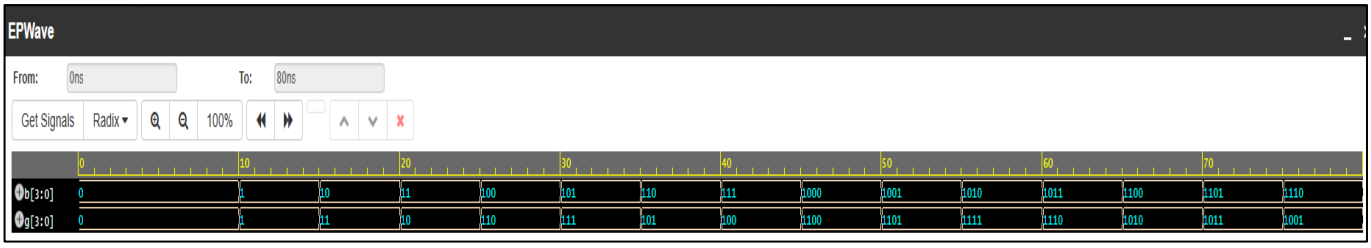
testbench.sv

```
// Code your testbench here
// or browse Examples
module binary4_gray_test;
  reg [3:0]b;
  wire [3:0]g;
  binary4_gray dut(g,b);
  initial begin
    {b}={4'b0000};
#5  {b}={4'b0000};
#5  {b}={4'b0001};
#5  {b}={4'b0010};
#5  {b}={4'b0011};
#5  {b}={4'b0100};
#5  {b}={4'b0101};
#5  {b}={4'b0110};
#5  {b}={4'b0111};
#5  {b}={4'b1000};
#5  {b}={4'b1001};
#5  {b}={4'b1010};
#5  {b}={4'b1011};
#5  {b}={4'b1100};
#5  {b}={4'b1101};
#5  {b}={4'b1110};
#5  {b}={4'b1111};
  end
  initial begin
    $monitor("sim time+%0t,b=%b,g=%b",$time,b,g);
    $dumpfile("dump.vcd");
    $dumpvars(1,binary4_gray_test);
  end
endmodule
```

Output

```
Log  Share
[2025-04-15 15:32:05 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:32:06 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                binary4_gray_test
Loading snapshot worklib.binary4_gray_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time+0,b=0000,g=0000
sim time+10,b=0001,g=0001
sim time+15,b=0010,g=0011
sim time+20,b=0011,g=0010
sim time+25,b=0100,g=0110
sim time+30,b=0101,g=0111
sim time+35,b=0110,g=0101
sim time+40,b=0111,g=0100
sim time+45,b=1000,g=1100
sim time+50,b=1001,g=1101
sim time+55,b=1010,g=1111
sim time+60,b=1011,g=1110
sim time+65,b=1100,g=1010
sim time+70,b=1101,g=1011
sim time+75,b=1110,g=1001
sim time+80,b=1111,g=1000
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:32:07 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-15 15:32:07 UTC] Opening EPWave...
Done
```

Output waveform



17) Design and implement 2's compliment using Data flow modeling in vector form.
Expression is
F2=AB'C'+A'B+A'C
F1=B⊕C
F0=C

Design code

design.sv

```
// Code your design here
module compliment2_3bit(f,a);
  input [2:0]a;
  output [2:0]f;
  assign f[2]=(a[2]&(~a[1])&(~a[0]))|((~a[2])&a[1])|((~a[2])&a[0]);
  assign f[1]=a[1]^a[0];
  assign f[0]=a[0];
endmodule
```

Test bench

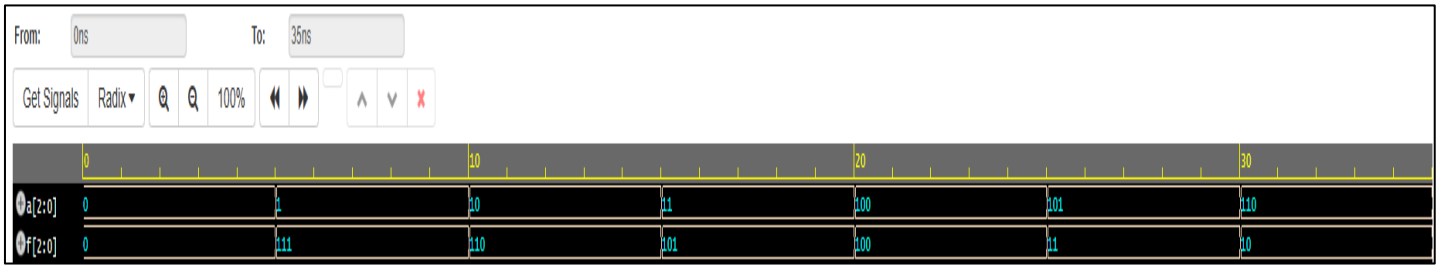
testbench.sv

```
// Code your testbench here
// or browse Examples
module compliment2_3bit_test;
  reg [2:0]a;
  wire [2:0]f;
  compliment2_3bit dut(f,a);
  initial begin
    {a}={3'd0};
    #5  {a}={3'd1};
    #5  {a}={3'd2};
    #5  {a}={3'd3};
    #5  {a}={3'd4};
    #5  {a}={3'd5};
    #5  {a}={3'd6};
    #5  {a}={3'd7};
  end
  initial begin
    $monitor("sim time=%0t,a=%b,f=%b",$time,a,f);
    $dumpfile("dump.vcd");
    $dumpvars(1,compliment2_3bit_test);
  end
endmodule
```

Output

```
[2025-04-15 15:39:55 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:39:55 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                compliment2_3bit_test
Loading snapshot worklib.compliment2_3bit_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=000,f=000
sim time=5,a=001,f=111
sim time=10,a=010,f=110
sim time=15,a=011,f=101
sim time=20,a=100,f=100
sim time=25,a=101,f=011
sim time=30,a=110,f=010
sim time=35,a=111,f=001
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:39:56 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-15 15:39:57 UTC] Opening EPWave...
Done
```

Output waveform



18) Design and implement 2:1 mux using Ternary operator.

Design code

design.sv

```
// Code your design here
module mux2_1ternary(y,i,s);
  input [1:0]i;
  input s;
  output y;
  assign y=(s)?i[1]:i[0];
endmodule

```

Test bench

testbench.sv

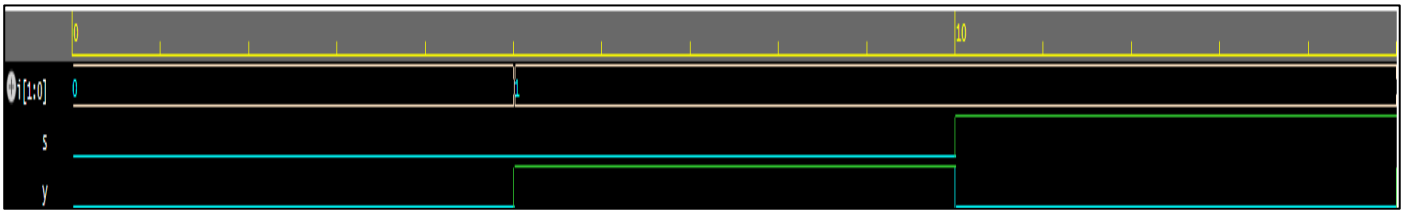
```
// Code your testbench here
// or browse Examples
module mux2_1ternary_test;
  reg [1:0]i;
  reg s;
  wire y;
  mux2_1ternary dut(y,i,s);
  initial begin
    {s,i}={1'b0,2'b00};
#5  {s,i}={1'b0,2'b01};
#5  {s,i}={1'b1,2'b01};
#5  {s,i}={1'b1,2'b11};
  end
  initial begin
    $monitor("sim time=%0t,s=%b,i=%b,y=%b",$time,s,i,y);
    $dumpfile("dump.vcd");
    $dumpvars(0,mux2_1ternary_test);
  end
endmodule
```

Output

Log | Share

```
[2025-04-15 15:49:13 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:49:14 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                mux2_1ternary_test
Loading snapshot worklib.mux2_1ternary_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,s=0,i=00,y=0
sim time=5,s=0,i=01,y=1
sim time=10,s=1,i=01,y=0
sim time=15,s=1,i=11,y=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:49:15 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-15 15:49:15 UTC] Opening EPWave...
Done
```

Output waveform



19) Design and implement 4:1 mux using Ternary operator.

Design code

design.sv

```
// Code your design here
module mux4_1ternary(y,i,s);
  input [3:0]i;
  input [1:0]s;
  output y;
  assign y=(s[1])?((s[0])?i[3]:i[2]):((s[0])?i[1]:i[0]);
endmodule
```

Test bench

testbench.sv

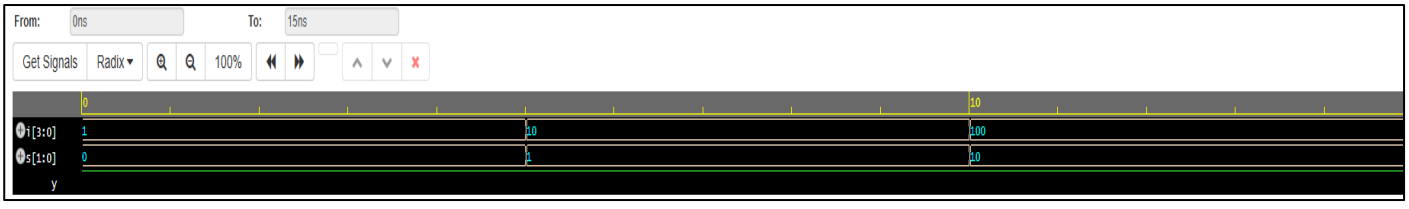
```
// Code your testbench here
// or browse Examples
module mux4_1ternary_test;
  reg [3:0]i;
  reg [1:0]s;
  wire y;
  mux4_1ternary dut(y,i,s);
  initial begin
    {s,i}={2'b00,4'b0001};
#5  {s,i}={2'b01,4'b0010};
#5  {s,i}={2'b10,4'b0100};
#5  {s,i}={2'b11,4'b1000};
  end
  initial begin
    $monitor("sim time=%0t,s=%b,i=%b,y=%b",$time,s,i,y);
    $dumpfile("dump.vcd");
    $dumpvars(0,mux4_1ternary_test);
  end
endmodule
```

Output

Log | Share

```
[2025-04-15 15:51:07 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:51:07 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                mux4_1ternary_test
Loading snapshot worklib.mux4_1ternary_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,s=00,i=0001,y=1
sim time=5,s=01,i=0010,y=1
sim time=10,s=10,i=0100,y=1
sim time=15,s=11,i=1000,y=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:51:08 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-15 15:51:08 UTC] Opening EPWave...
Done
```

Output waveform



20) Design and implement 2bit comparator using Ternary operator.

Design code

design.sv

```
// Code your design here
module comparator_2(a,b,y);
  input [1:0]a;
  input  [1:0]b;
  output [2:0]y;//y[2]=equal,y[1]=greater,y[0]=less
  assign y[2]=(a==b)?1'b1:1'b0;
  assign y[1]=(a>b)?1'b1:1'b0;
  assign y[0]=(a<b)?1'b1:1'b0;
endmodule
```

Test bench

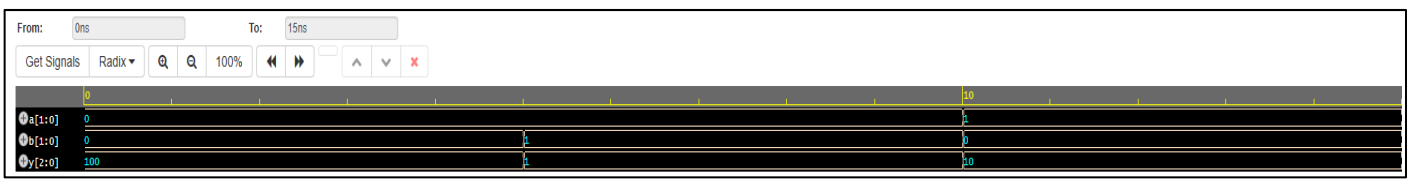
testbench.sv

```
// Code your testbench here
// or browse Examples
module comparator_2_test;
  reg [1:0]a;
  reg  [1:0]b;
  wire [2:0]y;
  comparator_2 dut(a,b,y);
  initial begin
    {a,b}={2'b00,2'b00};
#5  {a,b}={2'b00,2'b01};
#5  {a,b}={2'b01,2'b00};
#5  {a,b}={2'b11,2'b11};
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,y=%b",$time,a,b,y);
    $dumpfile("dump.vcd");
    $dumpvars(1,comparator_2_test);
  end
endmodule
```

Output

```
Log  Share
[2025-04-15 15:54:03 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:54:03 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                comparator_2_test
Loading snapshot worklib.comparator_2_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=00,b=00,y=100
sim time=5,a=00,b=01,y=001
sim time=10,a=01,b=00,y=010
sim time=15,a=11,b=11,y=100
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:54:04 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-15 15:54:05 UTC] Opening EPWave...
Done
```

Output waveform



21) Design and implement 3bit Palindrome detector using Ternary operator.

Logic is the MSB and LSB of bit should be same i.e., for A B C=A and C should be same.

Design code

design.sv

```
// Code your design here
module palindrome_3bit(a,y);
  input [2:0]a;
  output y;
  assign y=(a[2]===a[0])?1'b1:1'b0;/*=== is used because if x and z is given then it should compare and give as input 1 and 0 but if we give == it will give x value */
endmodule

```

Test bench

testbench.sv

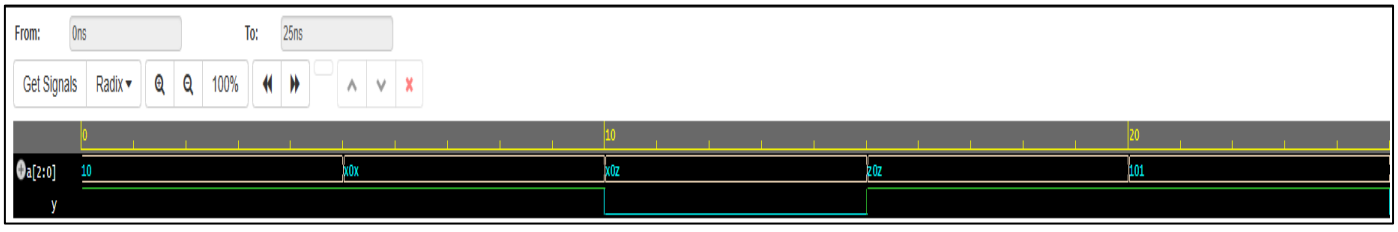
```
// Code your testbench here
// or browse Examples
module palindrome_3bit_test;
  reg [2:0]a;
  wire y;
  palindrome_3bit dut(a,y);
  initial begin
   a=3'b010;
#5 a=3'bx0x;
#5 a=3'bx0z;
#5 a=3'bz0z;
#5 a=3'b101;
#5 a=3'b100;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,y=%b",$time,a,y);
    $dumpfile("dump.vcd");
    $dumpvars(1,palindrome_3bit_test);
  end
endmodule
```

Output

Log | Share

```
[2025-04-15 15:45:37 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 15, 2025 at 11:45:37 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                palindrome_3bit_test
Loading snapshot worklib.palindrome_3bit_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=010,y=1
sim time=5,a=x0x,y=1
sim time=10,a=x0z,y=0
sim time=15,a=z0z,y=1
sim time=20,a=101,y=1
sim time=25,a=100,y=0
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 15, 2025 at 11:45:38 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-15 15:45:39 UTC] Opening EPWave...
Done
```

Output waveform



22) Design and implement 16bit Even and odd detector using Ternary operator.

Logic for even is observing LSB of bit, when there is 0 the bit is even.
1 the bit is odd.

Design code

design.sv

```
// Code your design here
module even_odd(a,even,odd);
  input [15:0]a;
  output even,odd;
  assign even = (!a[0])?1'b1:1'b0;
  assign odd = (a[0])?1'b1:1'b0;
endmodule
```

Test bench

testbench.sv

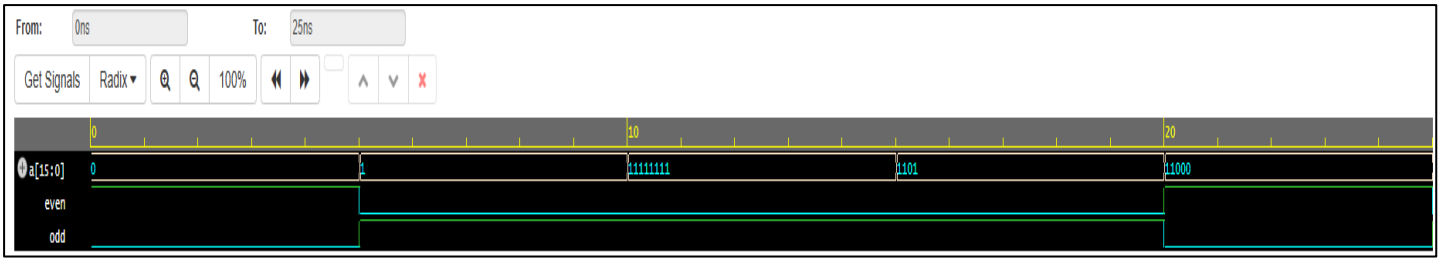
```
// Code your testbench here
// or browse Examples
module even_odd_test;
  reg [15:0]a;
  wire even,odd;
  even_odd dut(a,even,odd);
  initial begin
   a=15'b0;
#5 a=15'b1;
#5 a=15'hff;
#5 a=15'b1101;
#5 a=15'd24;
#5 a=15'd11;
  end
  initial begin
    $monitor("sim tim=%0t,a=%b,even=%b,odd=%b",$time,a,even,odd);
    $dumpfile("dump.vcd");
    $dumpvars(1,even_odd_test);
  end
endmodule
```

Output

Log | Share

```
[2025-04-17 12:09:49 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 17, 2025 at 08:09:50 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                even_odd_test
Loading snapshot worklib.even_odd_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim tim=0,a=0000000000000000,even=1,odd=0
sim tim=5,a=0000000000000001,even=0,odd=1
sim tim=10,a=0000000011111111,even=0,odd=1
sim tim=15,a=0000000000001101,even=0,odd=1
sim tim=20,a=0000000000011000,even=1,odd=0
sim tim=25,a=0000000000001011,even=0,odd=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 17, 2025 at 08:09:51 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-17 12:09:52 UTC] Opening EPWave...
Done
```

Output waveform



23) Write a Verilog code which accepts a 4-bit number and implement 8*n.

By left shifting 3 times or by appending three zeros at the LSB side of the 4-bit number i.e., n we can get the output of 8*n.

Design code

design.sv

```
// Code your design here
/*write a verilog code which accepts a 4bit number and implement 8*n*/

module bit4_8mulN(n,y);
  input [3:0]n;
  output [6:0]y;
  assign y={n,3'b000};
endmodule
/*
module bit4_8mulN(n,y);
  input [3:0]n;
  output [6:0]y;
  assign y=n<<3;
endmodule

module bit4_8mulN(n,y);
  input [3:0]n;
  output [6:0]y;
  assign y=8*n;
endmodule
*/

```

Test bench

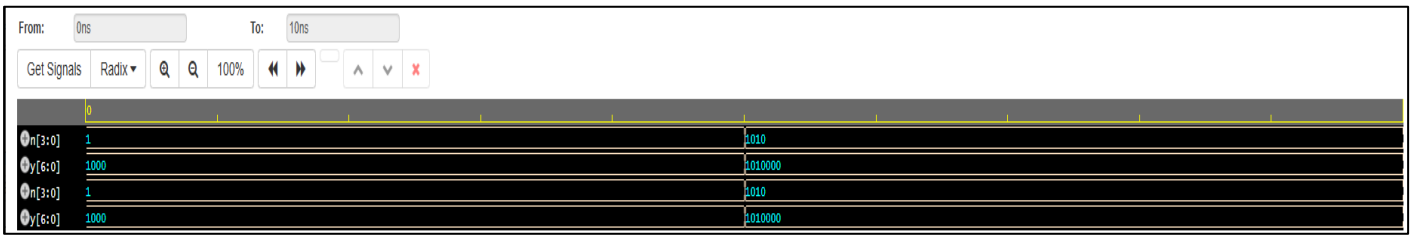
testbench.sv

```
// Code your testbench here
// or browse Examples
module bit4_8mulN_test;
  reg [3:0]n;
  wire [6:0]y;
  bit4_8mulN dut(n,y);
  initial begin
    n=4'b0001;
    #5 n=4'b1010;
    #5 n=4'b0011;
  end
  initial begin
    $monitor("sim time+%0t,n=%b,y=%b",$time,n,y);
    $dumpfile("dump.vcd");
    $dumpvars(0,bit4_8mulN_test);
  end
endmodule
```

Output

```
Log   Share
[2025-04-17 12:30:29 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 17, 2025 at 08:30:29 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                bit4_8mulN_test
Loading snapshot worklib.bit4_8mulN_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time+0,n=0001,y=0001000
sim time+5,n=1010,y=1010000
sim time+10,n=0011,y=0011000
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 17, 2025 at 08:30:30 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-17 12:30:31 UTC] Opening EPWave...
Done
```

Output waveform



24) Write a Verilog code for Full adder using Half adder in structural modeling.

Design code of lower-level module half adder

design.sv HA.v

```
module Half_adder(a,b,s,c);
  input a,b;
  output s,c;
  assign s=a^b;
  assign c=a&b;
endmodule
```

Design code

design.sv HA.v

```
// Code your design here
`include "HA.v"
module Full_adder(a,b,cin,sum,cout);
  input a,b,cin;
  output sum,cout;
  wire w1,w2,w3;
  Half_adder h1(a,b,w1,w2);
  Half_adder h2(w1,cin,sum,w3);
  assign cout=w2|w3;
endmodule
```

Test bench

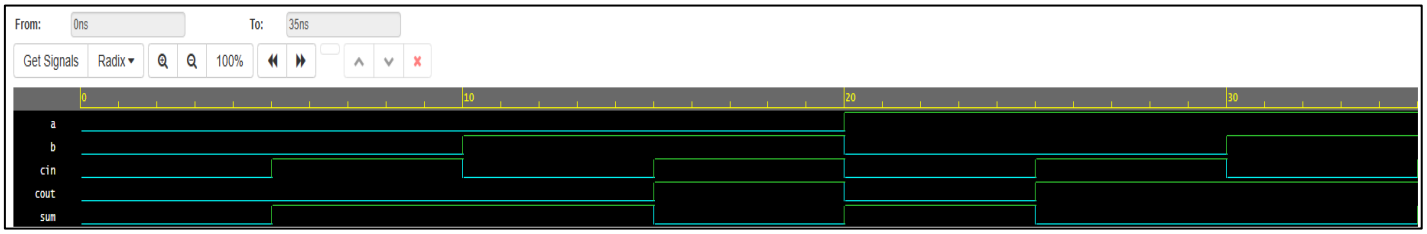
testbench.sv

```
// Code your testbench here
// or browse Examples
module Full_adder_test;
  reg a,b,cin;
  wire sum,cout;
  Full_adder dut(a,b,cin,sum,cout);
  initial begin
    a=1'b0;b=1'b0;cin=1'b0;
#5  a=1'b0;b=1'b0;cin=1'b1;
#5  a=1'b0;b=1'b1;cin=1'b0;
#5  a=1'b0;b=1'b1;cin=1'b1;
#5  a=1'b1;b=1'b0;cin=1'b0;
#5  a=1'b1;b=1'b0;cin=1'b1;
#5  a=1'b1;b=1'b1;cin=1'b0;
#5  a=1'b1;b=1'b1;cin=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,cin=%b,sum=%b,cout=%b",$time,a,b,cin,sum,cout);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,cin,sum,cout);
  end
endmodule
```

Output

```
Log  Share
[2025-04-17 12:43:25 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 17, 2025 at 08:43:25 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                Full_adder_test
Loading snapshot worklib.Full_adder_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,cin=0,sum=0,cout=0
sim time=5,a=0,b=0,cin=1,sum=1,cout=0
sim time=10,a=0,b=1,cin=0,sum=1,cout=0
sim time=15,a=0,b=1,cin=1,sum=0,cout=1
sim time=20,a=1,b=0,cin=0,sum=1,cout=0
sim time=25,a=1,b=0,cin=1,sum=0,cout=1
sim time=30,a=1,b=1,cin=0,sum=0,cout=1
sim time=35,a=1,b=1,cin=1,sum=1,cout=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 17, 2025 at 08:43:26 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-17 12:43:27 UTC] Opening EPWave...
Done
```

Output waveform



25) Write a Verilog code for 4-bit adder using Full adder in structural modeling.

Design code for lower-level module full adder

design.sv FA.v

```
module full_adder(a,b,cin,s,cout);
  input a,b,cin;
  output s,cout;
  assign s=a^b^cin;
  assign cout=(a&b)|(b&cin)|(cin&a);
endmodule
```

Design code

design.sv FA.v

```
// Code your design here
`include "FA.v"
module bit_4adder(a,b,cin,s,cout);
  input [3:0]a,b;
  input cin;
  output [3:0]s;
  output cout;
  wire [2:0]c;
  full_adder f1(a[0],b[0],cin,s[0],c[0]);
  full_adder f2(a[1],b[1],c[0],s[1],c[1]);
  full_adder f3(a[2],b[2],c[1],s[2],c[2]);
  full_adder f4(a[3],b[3],c[2],s[3],cout);
endmodule

```

Test bench

testbench.sv

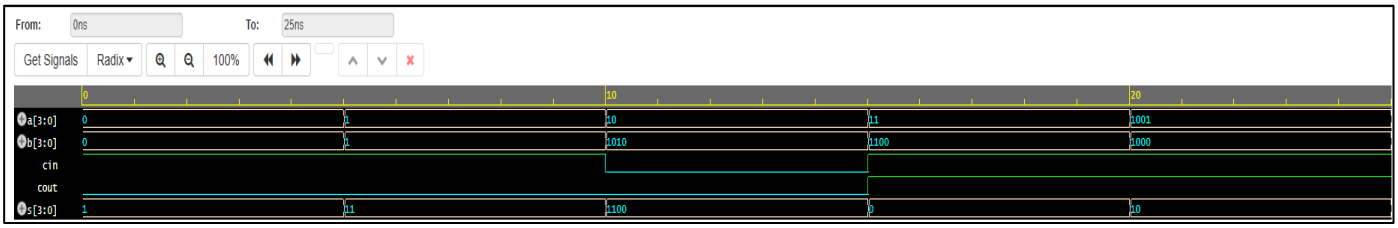
```
// Code your testbench here
// or browse Examples
module bit_4adder_test;
  reg [3:0]a,b;
  reg cin;
  wire [3:0]s;
  wire cout;
  bit_4adder dut(a,b,cin,s,cout);
  initial begin
    {a,b,cin}={4'b0,4'b0,1'b1};
#5  {a,b,cin}={4'b1,4'b1,1'b1};
#5  {a,b,cin}={4'b0010,4'b1010,1'b0};
#5  {a,b,cin}={4'b0011,4'b1100,1'b1};
#5  {a,b,cin}={4'd9,4'd8,1'b1};
#5  {a,b,cin}={4'ha,4'hb,1'b1};
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,cin=%b,s=%b,cout=%b",$time,a,b,cin,s,cout);
    $dumpfile("dump.vcd");
    $dumpvars(1,bit_4adder_test);
  end
endmodule
```

Output waveform

Log | Share

```
[2025-04-17 12:55:39 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 17, 2025 at 08:55:39 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                bit_4adder_test
Loading snapshot worklib.bit_4adder_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0000,b=0000,cin=1,s=0001,cout=0
sim time=5,a=0001,b=0001,cin=1,s=0011,cout=0
sim time=10,a=0010,b=1010,cin=0,s=1100,cout=0
sim time=15,a=0011,b=1100,cin=1,s=0000,cout=1
sim time=20,a=1001,b=1000,cin=1,s=0010,cout=1
sim time=25,a=1010,b=1011,cin=1,s=0110,cout=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 17, 2025 at 08:55:40 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-17 12:55:41 UTC] Opening EPWave...
Done
```

Output Waveform



26) Write a Verilog code for 4:1 mux using 2:1 mux in structural modeling.

Design code for lower-level module 2:1 mux

design.sv MUX.v

```
module mux2_1(i,s,y);
  input [1:0]i;
  input s;
  output y;
  assign y=((~s)&i[0])|(s&i[1]);
endmodule
```

Design code

design.sv MUX.v

```
// Code your design here
`include "MUX.v"
module mux4_1(i,s,y);
  input [3:0]i;
  input [1:0]s;
  output y;
  wire [1:0]w;
  mux2_1 M1(i[1:0],s[0],w[0]);
  mux2_1 M2(i[3:2],s[0],w[1]);
  mux2_1 M3(w,s[1],y);
endmodule

```

Test bench

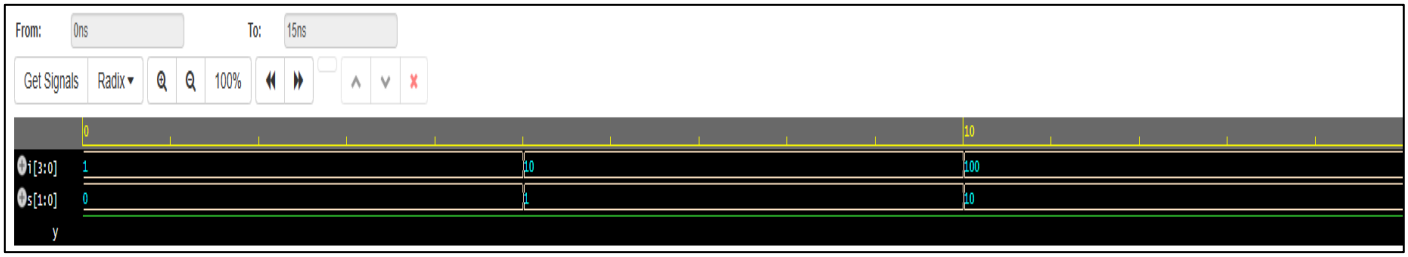
testbench.sv

```
// Code your testbench here
// or browse Examples
module mux4_1_test;
  reg [3:0]i;
  reg [1:0]s;
  wire y;
  mux4_1 dut(i,s,y);
  initial begin
    {s,i}={2'b00,4'b0001};
#5  {s,i}={2'b01,4'b0010};
#5  {s,i}={2'b10,4'b0100};
#5  {s,i}={2'b11,4'b1000};
  end
  initial begin
    $monitor("sim time=%0t,s=%b,i=%b,y=%b",$time,s,i,y);
    $dumpfile("dump.vcd");
    $dumpvars(1,mux4_1_test);
  end
endmodule
```

Output

```
Log  Share
[2025-04-17 13:05:25 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 17, 2025 at 09:05:25 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                mux4_1_test
Loading snapshot worklib.mux4_1_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,s=00,i=0001,y=1
sim time=5,s=01,i=0010,y=1
sim time=10,s=10,i=0100,y=1
sim time=15,s=11,i=1000,y=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 17, 2025 at 09:05:27 EDT  (total: 00:00:02)
Finding VCD file...
./dump.vcd
[2025-04-17 13:05:27 UTC] Opening EPWave...
Done
```

Output waveform



27) Write a Verilog code for 8:1 mux using 4:1 and 2:1 mux in structural modeling.

Design code for lower-level module

2:1 mux

design.sv | MUX4_1 | MUX2_1

```
1  module mux2_1(i,s,y);
2    input [1:0]i;
3    input s;
4    output y;
5    assign y=(s)?i[1]:i[0];
6  endmodule
7
```

4:1 mux

design.sv | MUX4_1 | MUX2_1

```
1  module mux4_1(i,s,y);
2    input [3:0]i;
3    input [1:0]s;
4    output y;
5    assign y=(s[1])?((s[0])?i[3]:i[2]):((s[0])?i[1]:i[0]);
6  endmodule
```

Design code

design.sv | MUX4_1 | MUX2_1

```
1  // Code your design here
2  `include"MUX2_1"
3  `include"MUX4_1"
4  module mux8_1(i,s,y);
5    input [7:0]i;
6    input [2:0]s;
7    output y;
8    wire [1:0]w;
9    mux4_1 m1(i[3:0],s[1:0],w[0]);
10   mux4_1 m2(i[7:4],s[1:0],w[1]);
11   mux2_1 m3(w,s[2],y);
12 endmodule
```

Test bench

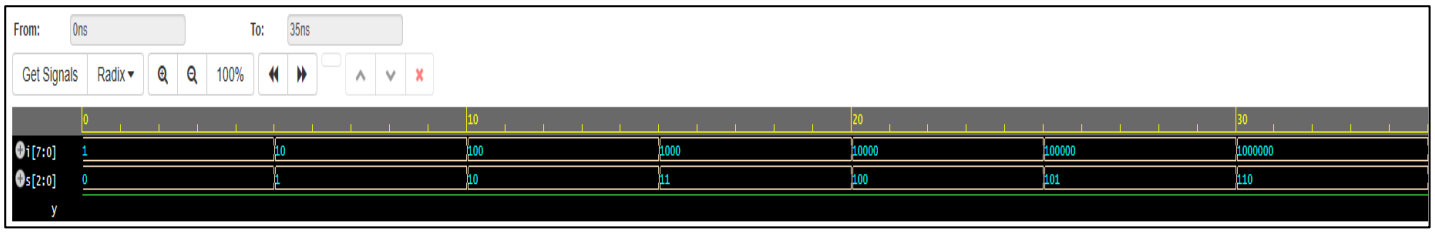
testbench.sv

```
1  // Code your testbench here
2  // or browse Examples
3  module mux8_1_test;
4    reg [7:0]i;
5    reg [2:0]s;
6    wire y;
7    mux8_1 dut(i,s,y);
8    initial begin
9         {s,i}={3'd0,8'b1};
10     #5 {s,i}={3'd1,8'b10};
11     #5 {s,i}={3'd2,8'b100};
12     #5 {s,i}={3'd3,8'b1000};
13     #5 {s,i}={3'd4,8'b10000};
14     #5 {s,i}={3'd5,8'b100000};
15     #5 {s,i}={3'd6,8'b1000000};
16     #5 {s,i}={3'd7,8'b10000000};
17   end
18   initial begin
19     $monitor("sim time=%0t,i=%b,s=%b,y=%b",$time,i,s,y);
20     $dumpfile("dump.vcd");
21     $dumpvars(1,mux8_1_test);
22   end
23 endmodule
```

Output

```
Log  Share
[2025-04-17 13:11:48 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 17, 2025 at 09:11:49 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                mux8_1_test
Loading snapshot worklib.mux8_1_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,i=00000001,s=000,y=1
sim time=5,i=00000010,s=001,y=1
sim time=10,i=00000100,s=010,y=1
sim time=15,i=00001000,s=011,y=1
sim time=20,i=00010000,s=100,y=1
sim time=25,i=00100000,s=101,y=1
sim time=30,i=01000000,s=110,y=1
sim time=35,i=10000000,s=111,y=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 17, 2025 at 09:11:50 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-17 13:11:50 UTC] Opening EPWave...
Done
```

Output



28) Write a Verilog code for 3:8 decoder using 2:4 and 1:2 decoder in structural modeling.

Design code for lower-level module

1:2 decoder

design.sv | DEC2_4.v | DEC1_2.v

```
module decoder1_2(y,i);
  input i;
  output [1:0]y;
  assign y[0]=(~i);
  assign y[1]=i;
endmodule
```

2:4 decoder

design.sv | DEC2_4.v | DEC1_2.v

```
module decoder2_4(y,i,w);
  input [1:0]i;
  output [3:0]y;
  input w;
  assign y[0]=((~i[1])&(~i[0])&(w));
  assign y[1]=((~i[1])&(i[0])&(w));
  assign y[2]=((i[1])&(~i[0])&(w));
  assign y[3]=((i[1])&(i[0])&(w));
endmodule
```

Design code

design.sv | DEC2_4.v | DEC1_2.v

```
// Code your design here
`include"DEC1_2.v"
`include"DEC2_4.v"
module decoder3_8(y,i);
  input [2:0]i;
  output [7:0]y;
  wire [1:0]w;
  decoder1_2 d1(w,i[2]);
  decoder2_4 d2(y[3:0],i[1:0],w[0]);
  decoder2_4 d3(y[7:4],i[1:0],w[1]);
endmodule
```

Test bench

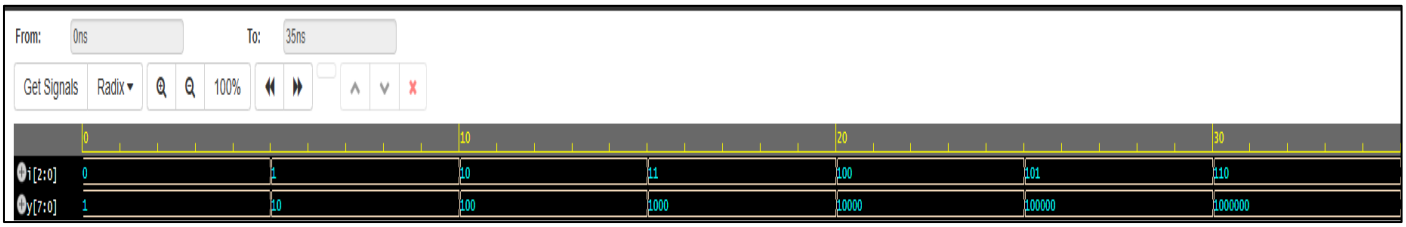
testbench.sv

```
// Code your testbench here
// or browse Examples
module decoder3_8_test;
  reg [2:0]i;
  wire [7:0]y;
  decoder3_8 dut(y,i);
  initial begin
        i=3'd0;
    #5  i=3'd1;
    #5  i=3'd2;
    #5  i=3'd3;
    #5  i=3'd4;
    #5  i=3'd5;
    #5  i=3'd6;
    #5  i=3'd7;
  end
  initial begin
    $monitor("sim time=%0t,i=%b,y=%b",$time,i,y);
    $dumpfile("dump.vcd");
    $dumpvars(1,decoder3_8_test);
  end
endmodule
```

Output

```
Log  Share
[2025-04-17 13:19:11 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 17, 2025 at 09:19:12 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                decoder3_8_test
Loading snapshot worklib.decoder3_8_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,i=000,y=00000001
sim time=5,i=001,y=00000010
sim time=10,i=010,y=00000100
sim time=15,i=011,y=00001000
sim time=20,i=100,y=00010000
sim time=25,i=101,y=00100000
sim time=30,i=110,y=01000000
sim time=35,i=111,y=10000000
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 17, 2025 at 09:19:13 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-17 13:19:13 UTC] Opening EPWave...
Done
```

Output waveform



29) Write a Verilog code for full adder using 4:1 mux in structural modeling.

Design code for lower-level module 4:1 mux

design.sv mux.v

```
module mux4_1(i,s,y);
  input [3:0]i;
  input [1:0]s;
  output y;
  assign y=(s[1])?((s[0])?i[3]:i[2]):((s[0])?i[1]:i[0]);
endmodule
```

Design code

design.sv mux.v

```
// Code your design here
`include "mux.v"
module Full_adder(a,b,sum,cin,cout);
  input a,b,cin;
  output sum,cout;
  mux4_1 m1({cin,(~cin),(~cin),cin},{a,b},sum);
  mux4_1 m2({1'b1,cin,cin,1'b0},{a,b},cout);
endmodule
```

Test bench

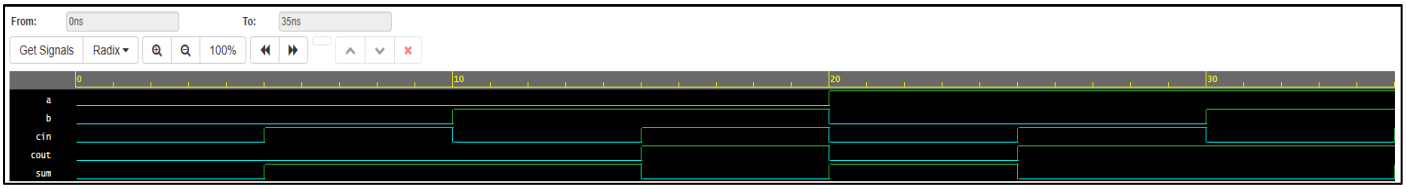
testbench.sv

```
// or browse Examples
module Full_adder_test;
  reg a,b,cin;
  wire sum,cout;
  Full_adder dut(a,b,sum,cin,cout);
  initial begin
    a=1'b0;b=1'b0;cin=1'b0;
#5  a=1'b0;b=1'b0;cin=1'b1;
#5  a=1'b0;b=1'b1;cin=1'b0;
#5  a=1'b0;b=1'b1;cin=1'b1;
#5  a=1'b1;b=1'b0;cin=1'b0;
#5  a=1'b1;b=1'b0;cin=1'b1;
#5  a=1'b1;b=1'b1;cin=1'b0;
#5  a=1'b1;b=1'b1;cin=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,cin=%b,sum=%b,cout=%b",$time,a,b,cin,sum,cout);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,cin,sum,cout);
  end
endmodule
```

Output

```
Log  Share
[2025-04-17 14:10:12 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 17, 2025 at 10:10:13 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                Full_adder_test
Loading snapshot worklib.Full_adder_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,cin=0,sum=0,cout=0
sim time=5,a=0,b=0,cin=1,sum=1,cout=0
sim time=10,a=0,b=1,cin=0,sum=1,cout=0
sim time=15,a=0,b=1,cin=1,sum=0,cout=1
sim time=20,a=1,b=0,cin=0,sum=1,cout=0
sim time=25,a=1,b=0,cin=1,sum=0,cout=1
sim time=30,a=1,b=1,cin=0,sum=0,cout=1
sim time=35,a=1,b=1,cin=1,sum=1,cout=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 17, 2025 at 10:10:14 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-17 14:10:14 UTC] Opening EPWave...
Done
```

Output waveform



30) Write a Verilog code for the 3 input xnor gate using only 2:1 mux.

Design code for lower-level module 2:1 mux

design.sv mux.v

```
module mux2_1(i,s,y);
  input [1:0]i;
  input s;
  output y;
  assign y=((~s)&i[0])|(s&i[1]);
endmodule
```

Design code

design.sv mux.v

```
// Code your design here
`include"mux.v"
module xnor3(a,b,c,y);
  input a,b,c;
  output y;
  wire [1:0]w;
  mux2_1 m1({(c),(~c)},b,w[0]);
  mux2_1 m2({(~c),(c)},b,w[1]);
  mux2_1 m3({w[1],w[0]},a,y);
endmodule
```

Test bench

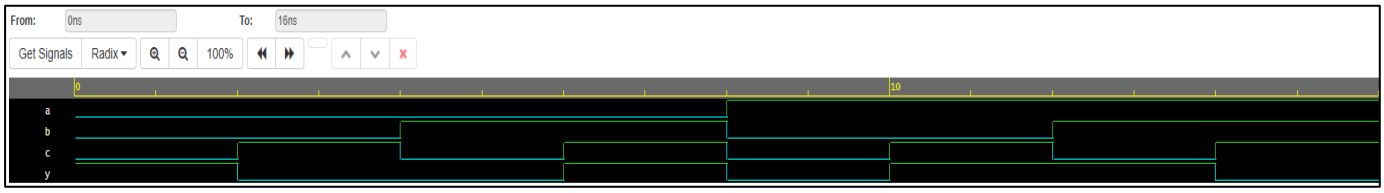
testbench.sv

```
// Code your testbench here
// or browse Examples
module xnor3_test;
  reg a,b,c;
  wire y;
  integer i;
  xnor3 dut(a,b,c,y);
  initial begin
    for(integer i=0; i<8;i++)
      begin
        {a,b,c}=i;
      #2;
      end
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,c=%b,y=%b",$time,a,b,c,y);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,c,y);
  end
endmodule
```

Output

```
Log   Share
[2025-04-17 14:56:29 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 17, 2025 at 10:56:29 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                xnor3_test
Loading snapshot worklib.xnor3_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,c=0,y=1
sim time=2,a=0,b=0,c=1,y=0
sim time=4,a=0,b=1,c=0,y=0
sim time=6,a=0,b=1,c=1,y=1
sim time=8,a=1,b=0,c=0,y=0
sim time=10,a=1,b=0,c=1,y=1
sim time=12,a=1,b=1,c=0,y=1
sim time=14,a=1,b=1,c=1,y=0
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 17, 2025 at 10:56:31 EDT  (total: 00:00:02)
Finding VCD file...
./dump.vcd
[2025-04-17 14:56:31 UTC] Opening EPWave...
Done
```

Output waveform



31) Write a Verilog code for Generic adder.

Design code

design.sv

```
// Code your design here
module generic_adder #(parameter N=4)(a,b,s);
  input [N-1:0]a,b;
  output [N:0]s;
  assign s=a+b;
endmodule
```

Test bench

testbench.sv

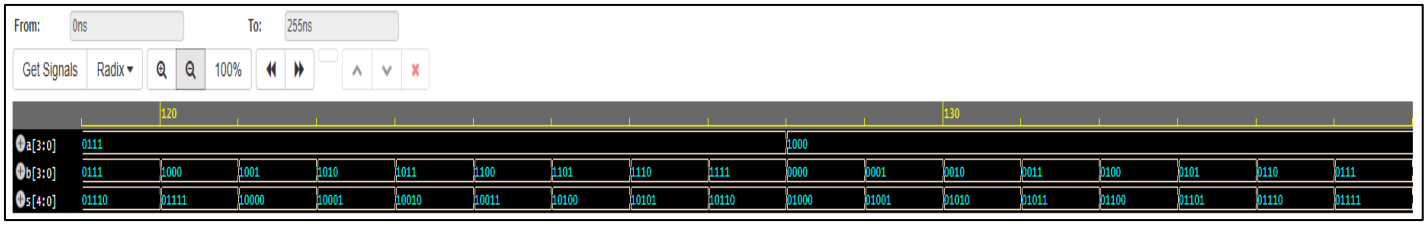
```
// Code your testbench here
// or browse Examples
module generic_adder_test;
  reg [dut.N-1:0]a,b;
  wire [dut.N:0]s;
  integer i;
  generic_adder dut (a,b,s);
  initial begin
    for(integer i=0;i<2**(2*dut.N);i=i++)
      begin
        {a,b}=i;
        #1;
      end
  end
  initial begin
    $monitor("sim time=%ot,a=%b,b=%b,s=%b",$time,a,b,s);
    $dumpfile("dump.vcd");
    $dumpvars(1,a,b,s);
  end
endmodule
```

Output

```
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64;  Apr 17 09:35 2025
sim time=00000000000000000000t,a=0000,b=0000,s=00000
sim time=00000000000000000001t,a=0000,b=0001,s=00001
sim time=00000000000000000002t,a=0000,b=0010,s=00010
sim time=00000000000000000003t,a=0000,b=0011,s=00011
sim time=00000000000000000004t,a=0000,b=0100,s=00100
sim time=00000000000000000005t,a=0000,b=0101,s=00101
```

```
sim time=00000000000000000365t,a=1111,b=0101,s=10100
sim time=00000000000000000366t,a=1111,b=0110,s=10101
sim time=00000000000000000367t,a=1111,b=0111,s=10110
sim time=00000000000000000370t,a=1111,b=1000,s=10111
sim time=00000000000000000371t,a=1111,b=1001,s=11000
sim time=00000000000000000372t,a=1111,b=1010,s=11001
sim time=00000000000000000373t,a=1111,b=1011,s=11010
sim time=00000000000000000374t,a=1111,b=1100,s=11011
sim time=00000000000000000375t,a=1111,b=1101,s=11100
sim time=00000000000000000376t,a=1111,b=1110,s=11101
sim time=00000000000000000377t,a=1111,b=1111,s=11110
```

Output waveform



32) Write a Verilog code for D latch using 2:1 mux.

Design code of lower-level module 2:1 mux

design.sv mux2_1

```
module mux2_1(i,s,y);
  input [1:0]i;
  input s;
  output y;
  assign y=(~s)?i[0]:i[1];
endmodule
```

Design code

design.sv mux2_1

```
// Code your design here
`include "mux2_1"
module dlatch(d,clk,q);
  input d,clk;
  output q;
  wire w;
  mux2_1 m1({d,w},clk,w);
  assign q=w;
endmodule
```

Test bench

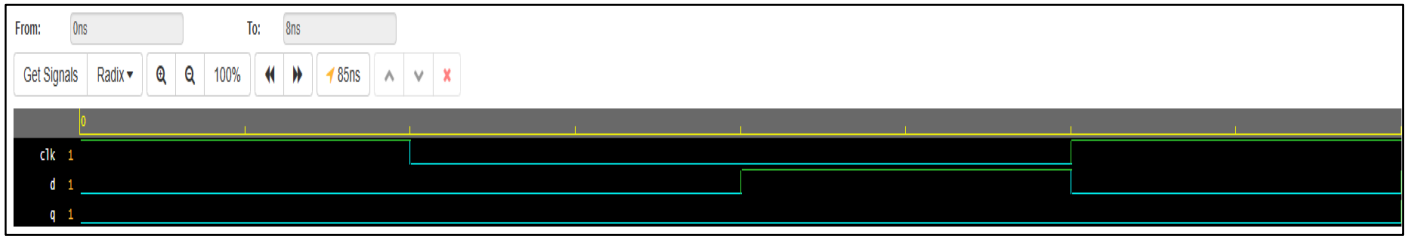
testbench.sv

```
// Code your testbench here
// or browse Examples
module dlatch_test;
  reg d,clk;
  wire q;
  dlatch dut(d,clk,q);
  initial begin
        {clk,d}={1'b1,1'b0};
    #2  {clk,d}={1'b0,1'b0};
    #2  {clk,d}={1'b0,1'b1};
    #2  {clk,d}={1'b1,1'b0};
    #2  {clk,d}={1'b1,1'b1};
  end
  initial begin
    $monitor("sim time=%0t,clk=%b,d=%b,q=%b",$time,clk,d,q);
    $dumpfile("dump.vcd");
    $dumpvars(1,dlatch_test);
  end
endmodule
```

Output

```
Log  Share
[2025-04-24 13:46:23 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 24, 2025 at 09:46:24 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                dlatch_test
Loading snapshot worklib.dlatch_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,clk=1,d=0,q=0
sim time=2,clk=0,d=0,q=0
sim time=4,clk=0,d=1,q=0
sim time=6,clk=1,d=0,q=0
sim time=8,clk=1,d=1,q=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 24, 2025 at 09:46:25 EDT  (total: 00:00:01)
Done
```

Output waveform



33) Write a Verilog code for D Flip flop using 2:1 mux.

Design code for 2:1 mux

design.sv mux2_1

```
module mux2_1(i,s,y);
  input [1:0]i;
  input s;
  output y;
  assign y=(~s)?i[0]:i[1];
endmodule
```

Design code

design.sv mux2_1

```
// Code your design here
`include"mux2_1"
module dff(d,clk,q);
  input d,clk;
  output q;
  wire [1:0]w;
  mux2_1 m1({d,w[1]},clk,w[1]);
  mux2_1 m2({w[1],w[0]},clk,w[0]);
  assign q=w[0];
endmodule

```

Test bench

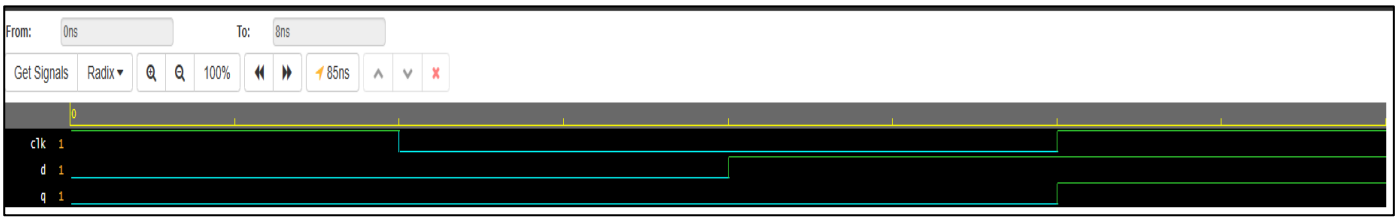
testbench.sv

```
// Code your testbench here
// or browse Examples
module dff_test;
  reg d,clk;
  wire q;
  dff dut(d,clk,q);
  initial begin
        {clk,d}={1'b1,1'b0};
    #2  {clk,d}={1'b0,1'b0};
    #2  {clk,d}={1'b0,1'b1};
    #2  {clk,d}={1'b1,1'b1};
    #2  {clk,d}={1'b1,1'b1};
  end
  initial begin
    $monitor("sim time=%0t,clk=%b,d=%b,q=%b,",$time,clk,d,q);
    $dumpfile("dump.vcd");
    $dumpvars(1,dff_test);
  end
endmodule
```

Output

```
Log  Share
[2025-04-24 13:57:28 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 24, 2025 at 09:57:28 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                dff_test
Loading snapshot worklib.dff_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,clk=1,d=0,q=0,
sim time=2,clk=0,d=0,q=0,
sim time=4,clk=0,d=1,q=0,
sim time=6,clk=1,d=1,q=1,
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 24, 2025 at 09:57:29 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-24 13:57:30 UTC] Opening EPWave...
Done
```

Output waveform



34) Write a Verilog code for Full adder in Behavioural modeling.

Design code

design.sv

```
// Code your design here
module Full_adder(a,b,cin,sum,cout);
  input a,b,cin;
  output reg sum,cout;
  always @(a,b,cin)
    begin
      sum=a^b^cin;
      cout=(a&b)|(b&cin)|(cin&a);
    end
endmodule
```

Test bench

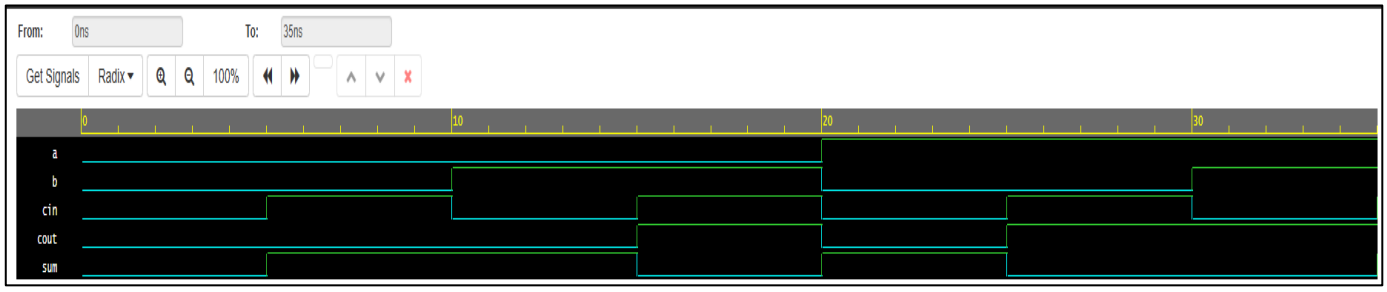
testbench.sv

```
// Code your testbench here
// or browse Examples
module Full_adder_test;
  reg a,b,cin;
  wire sum,cout;
  Full_adder dut(a,b,cin,sum,cout);
  initial begin
    a=1'b0;b=1'b0;cin=1'b0;
#5  a=1'b0;b=1'b0;cin=1'b1;
#5  a=1'b0;b=1'b1;cin=1'b0;
#5  a=1'b0;b=1'b1;cin=1'b1;
#5  a=1'b1;b=1'b0;cin=1'b0;
#5  a=1'b1;b=1'b0;cin=1'b1;
#5  a=1'b1;b=1'b1;cin=1'b0;
#5  a=1'b1;b=1'b1;cin=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,cin=%b,sum=%b,cout=%b",$time,a,b,cin,sum,cout);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,cin,sum,cout);
  end
endmodule
```

Output

```
Log  Share
[2025-04-24 15:54:58 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 24, 2025 at 11:54:59 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                Full_adder_test
Loading snapshot worklib.Full_adder_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,cin=0,sum=0,cout=0
sim time=5,a=0,b=0,cin=1,sum=1,cout=0
sim time=10,a=0,b=1,cin=0,sum=1,cout=0
sim time=15,a=0,b=1,cin=1,sum=0,cout=1
sim time=20,a=1,b=0,cin=0,sum=1,cout=0
sim time=25,a=1,b=0,cin=1,sum=0,cout=1
sim time=30,a=1,b=1,cin=0,sum=0,cout=1
sim time=35,a=1,b=1,cin=1,sum=1,cout=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 24, 2025 at 11:55:00 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-24 15:55:00 UTC] Opening EPWave...
Done
```

Output waveform



35) Write a Verilog code for Half adder in Behavioural modeling.

Design code

design.sv

```
// Code your design here
module half_adder (a,b,s,cout);
  input a,b;
  output reg s,cout;
  always @(a,b)
  begin
    {cout,s}=a+b;//s=a^b;
                 //cout=a&b;
  end
endmodule
```

Test bench

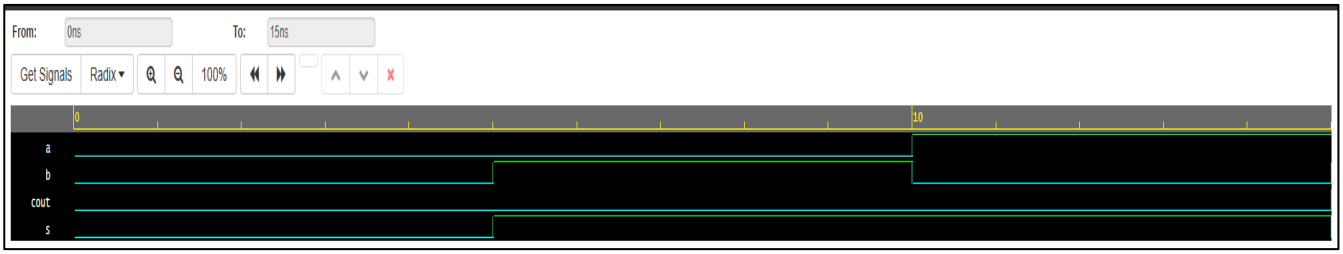
testbench.sv

```
// Code your testbench here
// or browse Examples
module half_adder_test;
  reg a,b;
  wire s,cout;
  half_adder dut(a,b,s,cout);
  initial begin
    a=1'b0;b=1'b0;
#5  a=1'b0;b=1'b1;
#5  a=1'b1;b=1'b0;
#5  a=1'b1;b=1'b1;
  end
  initial begin
    $monitor("sim time=%0t,a=%b,b=%b,s=%b,cout=%b",$time,a,b,s,cout);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,s,cout);
  end
endmodule

```

Output

```
Log  Share
[2025-04-24 16:07:55 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 24, 2025 at 12:07:56 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                half_adder_test
Loading snapshot worklib.half_adder_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,a=0,b=0,s=0,cout=0
sim time=5,a=0,b=1,s=1,cout=0
sim time=10,a=1,b=0,s=1,cout=0
sim time=15,a=1,b=1,s=0,cout=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 24, 2025 at 12:07:57 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-24 16:07:57 UTC] Opening EPWave...
Done
```

Output waveform



36) Write a Verilog code for 2:1 mux in Behavioural modeling.

Design code

design.sv

```
// Code your design here
module mux2_1(s,i,y);
  input s;
  input [1:0]i;
  output reg y;
  always @(s,i)
    begin
      y=(s)?i[1]:i[0];
    end
endmodule
```

Test bench

testbench.sv

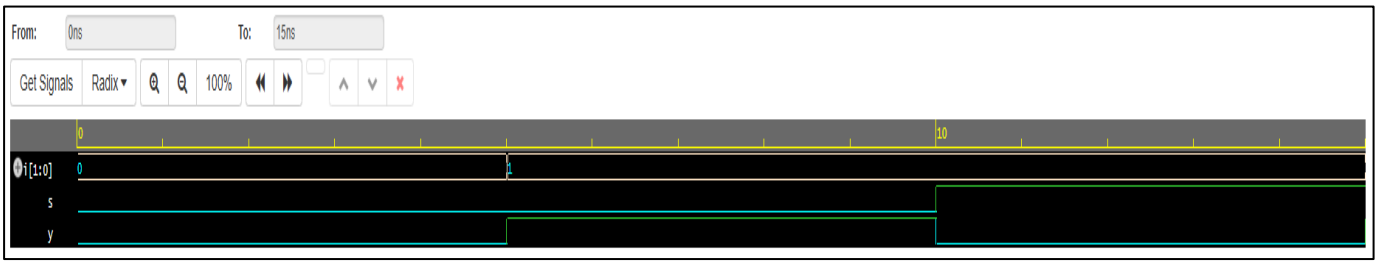
```
// Code your testbench here
// or browse Examples
module mux2_1_test;
  reg [1:0]i;
  reg s;
  wire y;
  mux2_1 dut(s,i,y);
  initial begin
    {s,i}={1'b0,2'b00};
#5  {s,i}={1'b0,2'b01};
#5  {s,i}={1'b1,2'b01};
#5  {s,i}={1'b1,2'b11};
  end
  initial begin
    $monitor("sim time=%0t,s=%b,i=%b,y=%b",$time,s,i,y);
    $dumpfile("dump.vcd");
    $dumpvars(1,mux2_1_test);
  end
endmodule
```

Output

Log | Share

```
[2025-04-24 16:19:32 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 24, 2025 at 12:19:32 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                mux2_1_test
Loading snapshot worklib.mux2_1_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,s=0,i=00,y=0
sim time=5,s=0,i=01,y=1
sim time=10,s=1,i=01,y=0
sim time=15,s=1,i=11,y=1
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 24, 2025 at 12:19:34 EDT  (total: 00:00:02)
Finding VCD file...
./dump.vcd
[2025-04-24 16:19:34 UTC] Opening EPWave...
Done
```

Output waveform



37) Write a Verilog code for 2:4 decoder in Behavioural modeling.

Design code

design.sv

```
// Code your design here
module decoder2_4(y,i);
  input [1:0]i;
  output reg [3:0]y;
  always @(i)
    begin
    y[0]=(~i[1])&(~i[0]);
    y[1]=(~i[1])&(i[0]);
    y[2]=(i[1])&(~i[0]);
    y[3]=(i[1])&(i[0]);
    end
endmodule
```

Test bench

testbench.sv

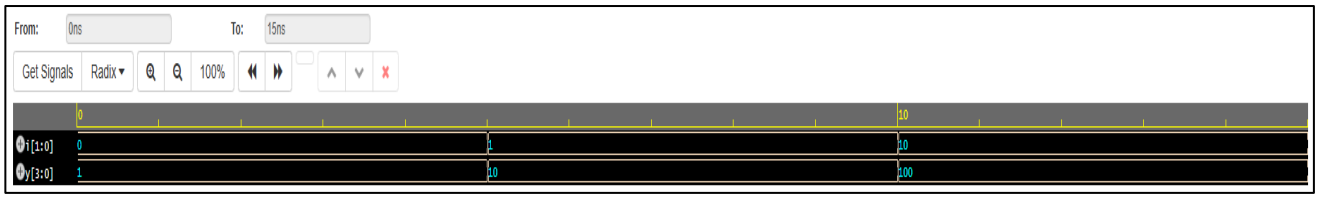
```
// Code your testbench here
// or browse Examples
module decoder2_4_test;
  reg [1:0]i;
  wire [3:0]y;
  decoder2_4 dut(y,i);
  initial begin
    {i}=0;
#5  {i}=1;
#5  {i}=2;
#5  {i}=3;
  end
  initial begin
    $monitor("sim time=%0t,i=%b,y=%b",$time,i,y);
    $dumpfile("dump.vcd");
    $dumpvars(1,decoder2_4_test);
  end
endmodule
```

Output

Log | Share

```
[2025-04-24 16:32:07 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 24, 2025 at 12:32:07 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                decoder2_4_test
Loading snapshot worklib.decoder2_4_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time=0,i=00,y=0001
sim time=5,i=01,y=0010
sim time=10,i=10,y=0100
sim time=15,i=11,y=1000
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 24, 2025 at 12:32:08 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-24 16:32:09 UTC] Opening EPWave...
Done
```

Output waveform



38) Write a Verilog code for Binary to Gray converter in Behavioural modeling.

Design code

design.sv

```
// Code your design here
module binary4_gray(g,b);
  input [2:0]b;
  output reg [2:0]g;
  always @(b)
    begin
    g[2]=b[2];
    g[1]=b[2]^b[1];
    g[0]=b[1]^b[0];
    end
endmodule
```

Test bench

testbench.sv

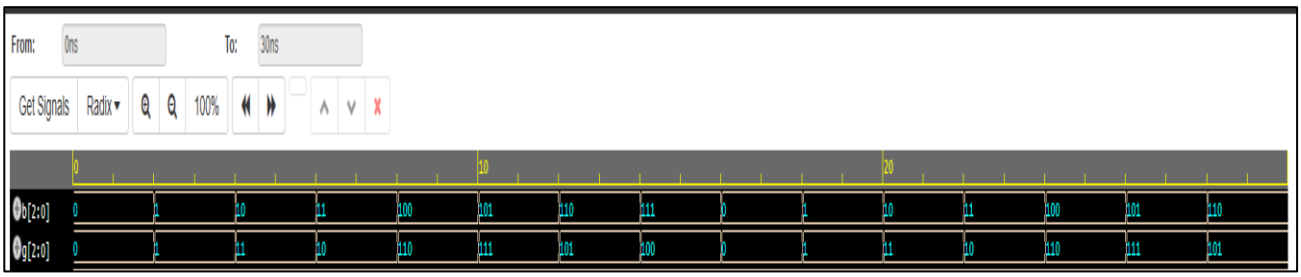
```
// Code your testbench here
// or browse Examples
module binary4_gray_test;
  reg [2:0]b;
  wire [2:0]g;
  integer i;
  binary4_gray dut(g,b);
  initial begin
    for (integer i=0;i<15;i++)
      begin
        b=i;
      #2;
      end
  end
  initial begin
    $monitor("sim time+%0t,b=%b,g=%b",$time,b,g);
    $dumpfile("dump.vcd");
    $dumpvars(1,binary4_gray_test);
  end
endmodule
```

Output

Log | Share

```
[2025-04-24 16:42:56 UTC] xrun -Q -unbuffered '-timescale' '1ns/1ns' '-sysv' '-access' '+rw' design.sv testbench.sv
TOOL:   xrun    23.09-s001: Started on Apr 24, 2025 at 12:42:57 EDT
xrun: 23.09-s001: (c) Copyright 1995-2023 Cadence Design Systems, Inc.
        Top level design units:
                binary4_gray_test
Loading snapshot worklib.binary4_gray_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
sim time+0,b=000,g=000
sim time+2,b=001,g=001
sim time+4,b=010,g=011
sim time+6,b=011,g=010
sim time+8,b=100,g=110
sim time+10,b=101,g=111
sim time+12,b=110,g=101
sim time+14,b=111,g=100
sim time+16,b=000,g=000
sim time+18,b=001,g=001
sim time+20,b=010,g=011
sim time+22,b=011,g=010
sim time+24,b=100,g=110
sim time+26,b=101,g=111
sim time+28,b=110,g=101
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on Apr 24, 2025 at 12:42:58 EDT  (total: 00:00:01)
Finding VCD file...
./dump.vcd
[2025-04-24 16:42:58 UTC] Opening EPWave...
Done
```

Output waveform



39) Write a Verilog code for D Latch using 2:1 mux.

Design code

design.sv | mux2_1

```
// Code your design here
`include "mux2_1"
module dlatch(d,clk,q);
  input d,clk;
  output q;
  wire w;
  mux2_1 m1({d,w},clk,w);
  assign q=w;
endmodule
```

Design code of lower-level module

design.sv | mux2_1

```
module mux2_1(i,s,y);
  input [1:0]i;
  input s;
  output y;
  assign y=(~s)?i[0]:i[1];
endmodule
```

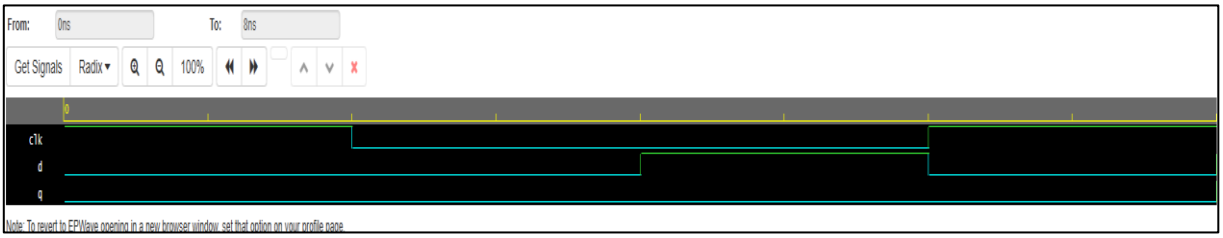
Test bench

```
// Code your testbench here
// or browse Examples
module dlatch_test;
  reg d,clk;
  wire q;
  dlatch dut(d,clk,q);
  initial begin
        {clk,d}={1'b1,1'b0};
    #2  {clk,d}={1'b0,1'b0};
    #2  {clk,d}={1'b0,1'b1};
    #2  {clk,d}={1'b1,1'b0};
    #2  {clk,d}={1'b1,1'b1};
  end
  initial begin
    $monitor("sim time=%0t,clk=%b,d=%b,q=%b",$time,clk,d,q);
    $dumpfile("dump.vcd");
    $dumpvars(1,dlatch_test);
  end
endmodule
```

Output

```
sim time=0,clk=1,d=0,q=0
sim time=2,clk=0,d=0,q=0
sim time=4,clk=0,d=1,q=0
sim time=6,clk=1,d=0,q=0
sim time=8,clk=1,d=1,q=1
xmsim: *W,RNQUIE: Simulation is complete.
```

Output Waveform



40) Write a Verilog code for D latch Asynchronous reset.

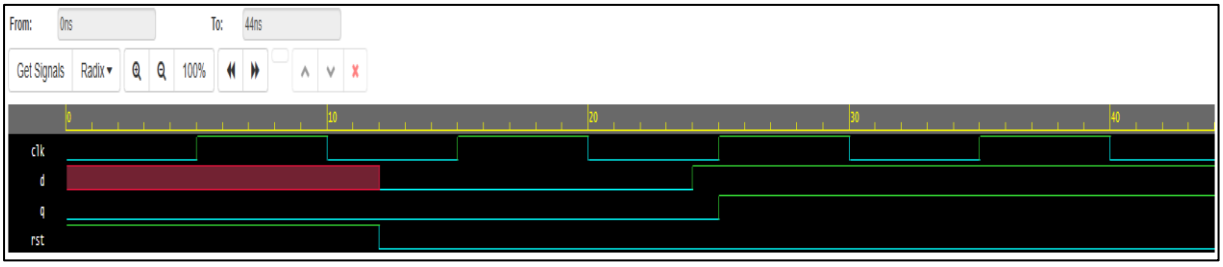
Design code

```
// Code your design here
module d_latch(clk,rst,d,q);
  input clk,rst,d;
  output reg q;
  always @(clk,rst,d)
    begin
      if (rst)
        q<=0;
      else if (clk)
        q<=d;
      else
        q<=q;
    end
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module d_latch_test;
  reg clk,rst,d;
  wire q;
  d_latch dut(clk,rst,d,q);
  initial begin
    clk=0;rst=1;
    #12 rst=0;d=0;
    #12 d=1;
    #20 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0,clk,rst,d,q);
  end
endmodule
```

Output waveform



41) Write a Verilog code for JK latch Asynchronous active low reset.

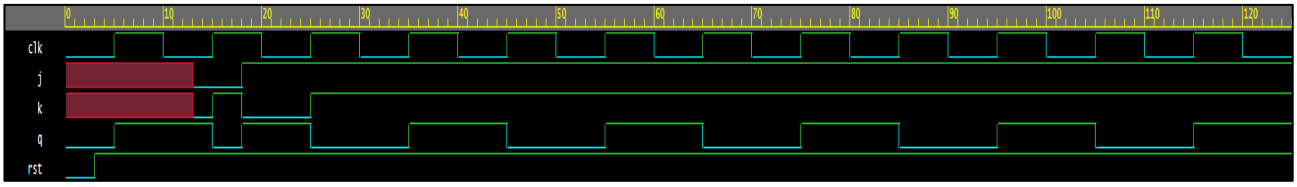
Design code

```
// Code your design here
module jk_latch(j,k,clk,rst,q);
  input clk,rst,j,k;
  output q;
  reg temp;
  assign q=temp;
  always @ (clk,rst,j,k)
    begin
      if (!rst)
        temp<=0;
      else if (clk)
        begin
          if (j==0 && k==0)
            temp<=temp;
          else if (j==0 && k==0)
            temp<=0;
          else if (j==1 && k==0)
            temp<=1;
          else
            temp<=~temp;
        end
      else
        temp<=temp;
    end
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module jk_latch_test;
  reg clk,rst,j,k;
  wire temp;
  jk_latch dut(j,k,clk,rst,q);
  initial begin
    clk=0;rst=0;
    #3 rst=1;
    #10 {j,k}=0;
    #2 {j,k}=1;
    #3 {j,k}=2;
    #7 {j,k}=3;
    #100 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $dumpfile ("dump.vcd");
    $dumpvars (0,clk,rst,j,k,q);
  end
endmodule
```

Output waveform



42) Write a Verilog code for T FF Asynchronous reset.

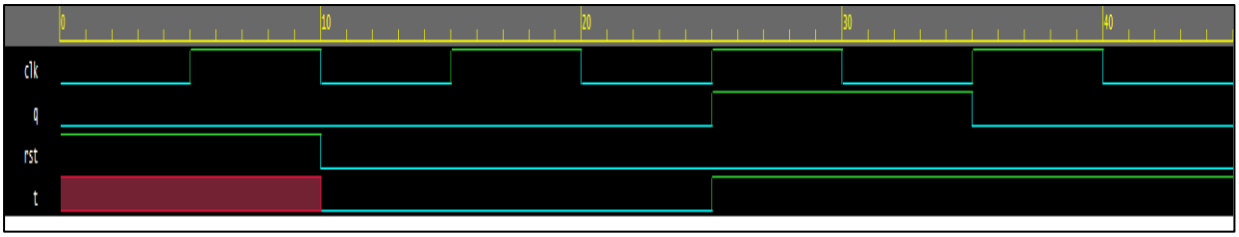
Design code

```
// Code your design here
module t_ff (t,clk,rst,q);
  input t,clk,rst;
  output q;
  reg temp;
  assign q=temp;
  always @(posedge clk or posedge rst)
    begin
      if(rst)
        temp<=0;
      else if(t)
        temp<=(~temp);
      else
        temp<=temp;
    end
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module t_ff_test;
  reg t,clk,rst;
  wire q;
  t_ff dut (t,clk,rst,q);
  always
    #5 clk=~clk;
  initial begin
    clk=0; rst=1;
    #10 rst=0;t=0;
    #15 t=1;
    #20 $finish;
  end
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(1,t_ff_test);
  end
endmodule
```

Output waveform



43) Write a Verilog code for SR FF Synchronous reset.

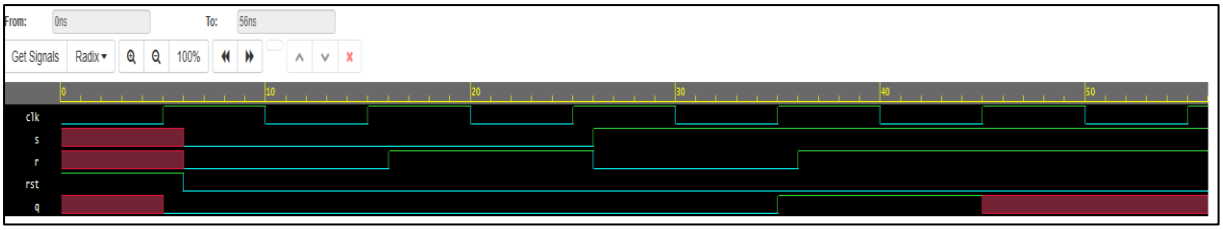
Design code

```
// Code your design here
module sr_ff(clk,rst,s,r,q);
  input clk,rst,s,r;
  output q;
  reg temp;
  assign q=temp;
  always @(posedge clk)
    begin
      if(rst)
        temp<=0;
      else if(s==0&&r==0)
        temp<=temp;
      else if (s==0&&r==1)
        temp<=0;
      else if (s==1&&r==0)
        temp<=1;
      else if (s==1&&r==1)
        temp<=1'bx;
      else
        temp<=temp;
    end
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module sr_ff_test;
  reg clk,rst,s,r;
  wire q;
  sr_ff dut (clk,rst,s,r,q);
  initial begin
    clk=0;rst=1;
    #6 rst=0;{s,r}=0;
    #10 {s,r}=1;
    #10 {s,r}=2;
    #10 {s,r}=3;
    #20 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0,clk,rst,s,r,q);
  end
endmodule
```

Output waveform

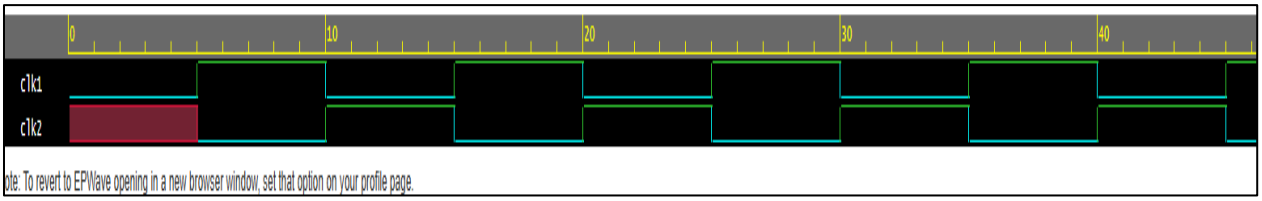


44) Generate 2 clock each of 10ns time period and 50 percent duty. Clk2 is delayed version of clock1 by 5ns

Code

```
module test;
  reg clk1,clk2;
  real tp=10;
  real duty=50;
  real ton;
  initial begin
    clk1=0;
    #5 clk2=0;
    ton=(tp*duty/100);
    #50 $finish;
  end
 always
   #5 clk1=(~clk1);
  always
     #(tp-ton) clk2=(~clk2);

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0,clk1,clk2);
  end
endmodule
```

Output waveform



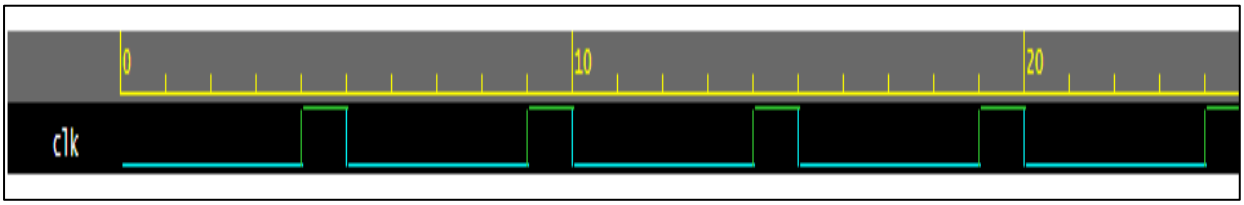
45) Generate clock for user defined time period and frequency.

This works for MHz frequency.

Code

```
// Code your testbench here
// or browse Examples
module test;
  reg clk;
  real freq= 200;
  real tp, ton;
  real duty= 20;
  initial begin
    clk=0;
    tp=(1000/freq);
    ton=(tp*duty)/100;
    #50 $finish;
  end
  always
    begin
      #(tp-ton) clk=1;
      #ton clk=0;
    end
  initial begin
    $dumpfile ("dump.vcd");
    $dumpvars(1,clk);
  end
endmodule
```

Output waveform



46) Write a Verilog code for the ALU using if-else.

Design code

```
// Code your design here
module alu(cntr,a,b,y);
  input  [3:0]cntr;
  input  [3:0]a;
  input  [3:0]b;
  output reg [7:0]y;

  always @(cntr,a,b)
  begin
    if (cntr == 4'b0000)
        y = a + b;
    else if (cntr == 4'b0001)
        y = a - b;
    else if (cntr == 4'b0010)
        y = a * b;
    else if (cntr == 4'b0011)
        y = a / b;
    else if (cntr == 4'b0100)
        y = a & b;
    else if (cntr == 4'b0101)
        y = a | b;
    else if (cntr == 4'b0110)
        y = a ^ b;
    else if (cntr == 4'b0111)
        y = ~a;
    else if (cntr == 4'b1000)
        y = a << 1;
    else if (cntr == 4'b1001)
        y = a >> 1;
    else if (cntr == 4'b1010)
        y = (a > b);
    else if (cntr == 4'b1011)
        y = (a < b);
    else if (cntr == 4'b1100)
        y = (a === b);
    else
        y = 8'b0;
end

endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module alu_test;
  reg [3:0] cntr;
  reg [3:0] a;
  reg [3:0]b;
  wire [7:0]y;
  integer i;
  alu dut(cntr,a,b,y);
    initial begin
        a = 8'd15; b = 8'd10;
      for (integer i=0; i<13;i++)
        begin
         #5 cntr=i;
        end
    end
  initial begin
    $monitor ("time=%0t,a=%b,b=%b,cntr=%b,y=%d",$time,a,b,cntr,y);
    $dumpfile("dump.vcd");
    $dumpvars(0,a,b,cntr,y);
  end
endmodule

```

Output

```
time=0,a=1111,b=1010,cntr=xxxx,y=  0
time=5,a=1111,b=1010,cntr=0000,y= 25
time=10,a=1111,b=1010,cntr=0001,y=  5
time=15,a=1111,b=1010,cntr=0010,y=150
time=20,a=1111,b=1010,cntr=0011,y=  1
time=25,a=1111,b=1010,cntr=0100,y= 10
time=30,a=1111,b=1010,cntr=0101,y= 15
time=35,a=1111,b=1010,cntr=0110,y=  5
time=40,a=1111,b=1010,cntr=0111,y=240
time=45,a=1111,b=1010,cntr=1000,y= 30
time=50,a=1111,b=1010,cntr=1001,y=  7
time=55,a=1111,b=1010,cntr=1010,y=  1
time=60,a=1111,b=1010,cntr=1011,y=  0
time=65,a=1111,b=1010,cntr=1100,y=  0
xmsim: *W,RNQUIE: Simulation is complete.
```

47) Write a Verilog code for the MOD 10 counter.

Design code

```
// or browse Examples
module counter_test;
 reg clk,rst;
  wire [3:0]count;
 integer i;
 counter dut(clk,rst,count);
 initial begin
   clk=0; rst=1;
   #6 rst=0;
   #100 $finish;
 end
   always
     #5 clk=~clk;
  initial begin
    $monitor ("time=%0t,count=%d",$time,count);
    $dumpfile("dump.vcd");
    $dumpvars(0,count);
  end
endmodule
```

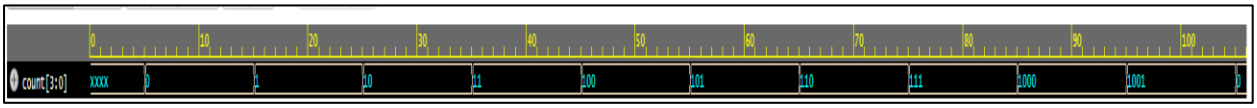
Test bench

```
// Code your design here
module counter(clk,rst,count);
  input clk,rst;
  output [3:0]count;
  reg [3:0]temp;
  always @(posedge clk)
    begin
      if (rst)
        temp<=0;
      else if (temp==4'd9)
        temp<=0;
      else
        temp<=temp+1;
    end
  assign count=temp;
endmodule
```

Output

```
time=0,count= x
time=5,count= 0
time=15,count= 1
time=25,count= 2
time=35,count= 3
time=45,count= 4
time=55,count= 5
time=65,count= 6
time=75,count= 7
time=85,count= 8
time=95,count= 9
time=105,count= 0
```

Output Waveform



48) Write a Verilog code for the 3 bit up asynchronous up counter.

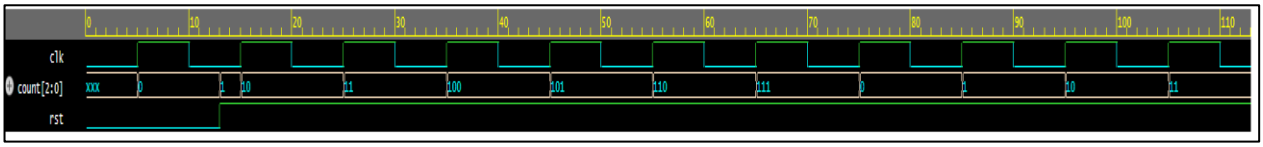
Design code

```
// Code your design here
module up3counter(count,clk,rst);
  input clk,rst;
  output [2:0]count;
  reg [2:0]temp;
  always @(posedge clk or posedge rst)
    begin
      if (!rst)
        temp<=0;
      else
        temp<=temp+1;
    end
  assign count=temp;
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
 module up3counter_test;
  reg clk,rst;
  wire [2:0] count;
  up3counter dut(count,clk,rst);
  initial begin
    clk= 0;
    rst= 0;
    #13 rst= 1;
    #100 $finish;
  end
    always #5 clk=~clk;
  initial begin
    $monitor("sim time=%0t,clk=%b,rst=%b,count=%b",$time,clk,rst,count);
    $dumpfile("dump.vcd");
    $dumpvars(1,up3counter_test);
  end
endmodule
```

Out put waveform



49) Write a Verilog code for the Parameterized up/down counter.

Design code

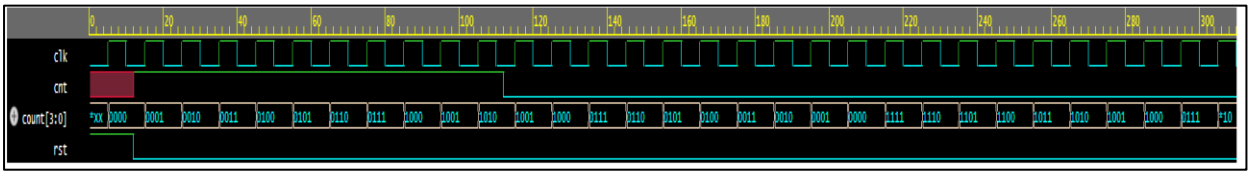
```
// Code your design here
module counterup_down #(parameter n=4)(clk,rst,cnt,count);
  input clk,rst,cnt;
  output [n-1:0]count;
  reg [n-1:0]temp;
  always @(posedge clk)
    begin
      if(rst)
        temp<=0;
      else if (cnt==1)
        temp= temp+1;
      else
        temp= temp-1;
    end
  assign count=temp;
endmodule
```

Test bench

testbench.sv

```
// Code your testbench here
// or browse Examples
module counterup_down_test;
  reg clk,rst,cnt;
  wire [dut.n-1:0]count;
  counterup_down dut(clk,rst,cnt,count);
  initial begin
    clk=0;rst=1;
    #12 rst=0;cnt=1;
    #100 cnt=0;
    #200 $finish;
  end
  always #5 clk=~clk;
  initial begin
    $monitor ("sim time=%0t,clk=%b,rst=%b,cnt=%b,count=%b",$time,clk,rst,cnt
              ,count);
    $dumpfile("dump.vcd");
    $dumpvars(0,clk,rst,cnt,count);
  end
endmodule
```

Output waveform



50) Write a Verilog code for Priority encoder using case z.

Design code

```
// Code your design here
module priority_encoder(y,i);
  input [7:0]i;
  output reg [2:0]y;
  always @(i)
    begin
    casez(i)
      8'b0000_0001:y=3'd0;
      8'b0000_001z:y=3'd1;
      8'b0000_01zz:y=3'd2;
      8'b0000_1zzz:y=3'd3;
      8'b0001_zzzz:y=3'd4;
      8'b001z_zzzz:y=3'd5;
      8'b01zz_zzzz:y=3'd6;
      8'b1zzz_zzzz:y=3'd7;
      default:y=3'dx;
    endcase
    end
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module priority_encoder_test;
  reg [7:0]i;
  wire [2:0]y;
  integer a;
  priority_encoder dut(y,i);
  initial begin
    i=8'b00z1_x000;
    #5   i=8'b000?_000x;
    #5   i=8'b000x_0?00;
    #5;
    for(integer a=0;a<256;a=a+1)
      begin
        i=a;
        #5;
      end
  end
  initial begin
    $monitor("sim time=%0t,i=%b,y=%d",$time,i,y);
  end
endmodule
```

Output

```
sim time=0,i=00z1x000,y=4
sim time=5,i=000z000x,y=4
sim time=10,i=000x0z00,y=x
sim time=15,i=00000000,y=x
sim time=20,i=00000001,y=0
sim time=25,i=00000010,y=1
sim time=30,i=00000011,y=1
sim time=35,i=00000100,y=2
sim time=40,i=00000101,y=2
sim time=45,i=00000110,y=2
sim time=50,i=00000111,y=2
sim time=55,i=00001000,y=3
sim time=60,i=00001001,y=3
sim time=65,i=00001010,y=3
```

51) Write a Verilog code for Digital clock.

Design code

```
// Code your design here
module digital_clk (
    input clk,
    input rst,
    output reg [3:0] hour,
    output reg [5:0] minute,
    output reg [5:0] second
);

    always @(posedge clk)
      begin
        if (rst)
          begin
            hour   <= 0;
            minute <= 0;
            second <= 0;
        end
      else
        begin
            if (second == 6'd59)
              begin
                second <= 0;
                if (minute == 6'd59)
                  begin
                    minute <= 0;
                    if (hour == 4'd11)
                        hour <= 0;
                  else
                        hour <= hour + 1;
                end
                else
                    minute <= minute + 1;
            end
          else
                second <= second + 1;
        end
    end
endmodule

```

Test bench

```
// Code your testbench here
// or browse Examples
module digital_clk_test;
  reg clk,rst;
  wire [3:0]hour;
  wire [5:0]minute;
  wire [5:0]second;
  digital_clk dut(clk,rst,hour,minute,second);
  initial begin
    clk=0;rst=1;
    #10 rst=0;
    #432000 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $display("Time H:M:S");
    $monitor("%0t %0d:%0d:%0d", $time, hour, minute, second);

  end
endmodule
```

Output

```
0 x:x:x
5 0:0:0
15 0:0:1
25 0:0:2
35 0:0:3
45 0:0:4
55 0:0:5
65 0:0:6
75 0:0:7
85 0:0:8
95 0:0:9
105 0:0:10
115 0:0:11
125 0:0:12
135 0:0:13
145 0:0:14
155 0:0:15
165 0:0:16
175 0:0:17
```

52) Write a Verilog code for Even odd checker using function.

Design code

```
// Code your design here
module even(N, y);
  input [31:0] N;
  output y;
  assign y = eve(N);
  function eve (input [31:0]N);
    begin
      if (N%2== 0)
        eve = 1;
      else
        eve = 0;
    end
  endfunction
endmodule
```

Testbench

```
// Code your testbench here
// or browse Examples
module even_test;
  reg [31:0] N;
  wire y;
  even dut (N, y);
  initial begin
    #10 N=2;
    #10 N = 3;
    #10 N = 100;
    #10 N = 255;
    #10 N = 0;
    #10 N = 1;
  end
  initial
  forever @(N)
    begin
     #1;
    if (y)
      $display("The number %0d is even", N);
    else
      $display("The number %0d is odd", N);
  end
endmodule

```

Output

```
Loading snapshot worklib.even_test:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
The number 2 is even
The number 3 is odd
The number 100 is even
The number 255 is odd
The number 0 is even
The number 1 is odd
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL:   xrun    23.09-s001: Exiting on May 25, 2025 at 10:15:07 EDT  (total: 00:00:01)
Done
```

53) Write a Verilog code for Prime number detector using function.

Design code

```
// Code your testbench here
// or browse Examples
module prime_detector_test;
  reg [31:0] a;
  prime_detector dut (a);
  initial begin
    a = 2;
  end
endmodule
```

Test bench

```
// Code your design here
module prime_detector(input [31:0] a);
 reg prime;

  function  prime_fun (input [31:0] a);
    integer i;
    begin
      if (a<2)
        prime_fun = 0;
      else begin
        prime_fun = 1;
        for (i = 2; i <= a/2; i = i + 1) begin
          if (a % i == 0)
            prime_fun = 0;
        end
      end
    end
  endfunction

  always @(a) begin
    prime = prime_fun(a);
     if (prime)
      $display("%0d is a prime", a);
    else
      $display("%0d is not a prime", a);
  end
endmodule
```

Output

```
xcelium> source /xcelium23
xcelium> run
2 is a prime
```

54) Write a Verilog code for Fibonacci series generator using function.

Code

```
module fibonacci_generator #(parameter N = 15);
  integer i;
  integer y;
  function  integer fib(input integer n);
    integer a, b, temp, j;
    begin
      a = 0;
      b = 1;
      if (n == 0)
        fib = 0;
      else if (n == 1)
        fib = 1;
      else
        begin
          for (j = 2; j <= n; j = j + 1)
            begin
              temp=a+b;
              a=b;
              b=temp;
            end
          fib=b;
        end
    end
  endfunction
  initial begin
    for (i=0;i<N;i=i+1)
      begin
        y = fib(i);
        $display("Fibonacci[%0d]=%0d",i,y);
      end
  end
endmodule
```

Output

```
Fibonacci[0]=0
Fibonacci[1]=1
Fibonacci[2]=1
Fibonacci[3]=2
Fibonacci[4]=3
Fibonacci[5]=5
Fibonacci[6]=8
Fibonacci[7]=13
Fibonacci[8]=21
Fibonacci[9]=34
Fibonacci[10]=55
Fibonacci[11]=89
Fibonacci[12]=144
Fibonacci[13]=233
Fibonacci[14]=377
```

55) Write a Verilog code generate pattern 0102030405.

Code

```
// Code your testbench here
// or browse Examples
/*module pattern;
  integer i,N=5;
  initial begin
    for (i=0;i<=9;i=i+1)
      if (i%2==0)
        $write("%0d",0);
        else
          $write("%0d",(i+1)/2);
      end
endmodule*/
module pattern();
  initial
    begin
      for (integer i = 1; i<=5; i++)
        begin
        $write("0%0d",i);
      end
    end
endmodule
```

Output

```
                pattern
Loading snapshot worklib.pattern:sv ..................... Done
xcelium> source /xcelium23.09/tools/xcelium/files/xmsimrc
xcelium> run
0102030405xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
```

56) Write a Verilog code generate pattern 122333444455555.

Code

```
// Code your testbench here
// or browse Examples
module pattern;
  integer i,j,N=5;
  initial begin
    for (i=1;i<=N;i=i+1)
      begin
        for (j=1;j<=i;j=j+1)
        begin
          $write("%0d",i);
      end
    end
  end
endmodule
```

Output

```
xcelium> run
122333444455555xmsim: *W,RNQUIE: Simulation is complete.
```

57) Write a Verilog code generate clock using task.

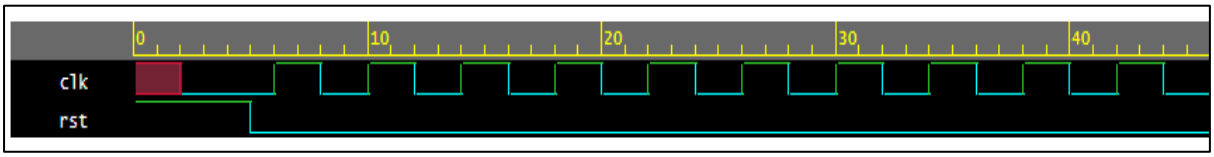
Design code

```
// Code your design here
module clock_generator (rst,clk);
  input rst;
  output reg clk;
  reg int_clk;
  assign clk=int_clk;
  task cg();
    begin
      if (rst)
        int_clk=0;
      else
        int_clk=~int_clk;
      #2;
    end
    endtask
  always
    begin
    cg();
    end
endmodule
```

Testbench

```
// Code your testbench here
// or browse Examples
module clock_generator_test;
  reg rst;
  wire clk;
  clock_generator dut(rst,clk);
  initial begin
    rst=1;
    #5 rst=0;
   #100 $finish;
  end
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0,rst,clk);
  end
endmodule
```

Output Waveform



58) Write a Verilog code Frequency divider using task.

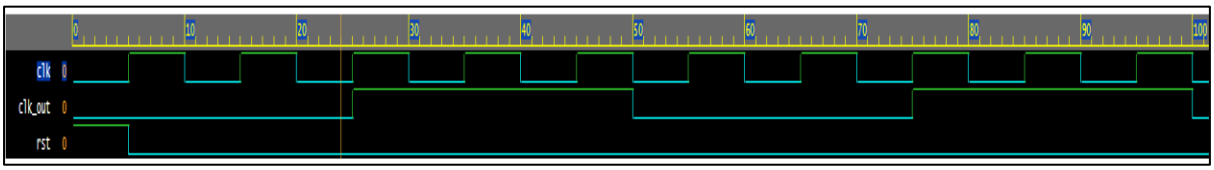
Design code

```
// Code your design here
module frequency_devider#(parameter N=5)(clk,rst,clk_out);
  input clk,rst;
  output clk_out;
  reg int_clk;
  integer count;
  assign clk_out = int_clk;
  task freq_devi();
    begin
      count=count+1;
      if (count%N==0)
        begin
          int_clk=~int_clk;
          count=0;
        end
    end
  endtask
  always @(posedge clk or negedge clk)
    begin
      if (rst)
        begin
          int_clk=0;
          count=0;
        end
      else
        begin
          freq_devi();
        end
      end
endmodule
```

Testbench

```
// Code your testbench here
// or browse Examples
module frequency_devider_test;
  reg clk,rst;
  wire clk_out;
  frequency_devider dut(clk,rst,clk_out);
  initial begin
    clk=0;rst=1;
    #5 rst=0;
    #100 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0,clk,clk_out,rst);
  end
endmodule
```

Output waveform



59) Write a Verilog code factorial of a number.

Design code

```
// Code your design here
module factorial(N,y);
  input [31:0]N;
  output [63:0]y;
  assign y=fact(N);
  function automatic[63:0]fact (input [31:0]n);
    begin
      if (n>=1)
        fact = n*fact(n-1);
      else
        fact = 1;
    end
  endfunction
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module factorial_test;
  reg [31:0]N;
  wire [63:0]y;
  factorial dut(N,y);
  initial begin
    N=0;
    #5 N=4;
    #5 N=5;
    #6 N=6;
    #5 N=1;
  end
  initial begin
    $monitor ("%0d factorial is = %0d",N,y);
  end
endmodule
```

Output

```
0 factorial is = 1
4 factorial is = 24
5 factorial is = 120
6 factorial is = 720
1 factorial is = 1
```

60) Write a Verilog code SISO.

Design code

```
// Code your design here
module siso(clk,rst,sin,sout);
  input clk,rst,sin;
  output sout;
  reg [3:0] temp;
  assign sout=temp[0];
  always @(posedge clk)
    begin
      if (rst)
        temp<=0;
      else
        temp<={sin,temp[3:1]};
    end
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module siiso_test;
  reg clk,rst,sin;
  wire sout;
  siso dut(clk,rst,sin,sout);
  initial begin
    clk=0;rst=1;
    #6 rst=0; sin=1;
    #9 sin=0;
    #10 sin=1;
    #10 sin=0;
    #10 sin=1;
    #10 sin=1;
    #100 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $monitor("sim time=%0t,rst=%b,sin=%b,sout=%b",$time,rst,sin,sout);
    $dumpfile("dump.vcd");
    $dumpvars(0,clk,rst,sin,sout);
  end
endmodule
```

Output

```
sim time=0,rst=1,sin=x,sout=x
sim time=5,rst=1,sin=x,sout=0
sim time=6,rst=0,sin=1,sout=0
sim time=15,rst=0,sin=0,sout=0
sim time=25,rst=0,sin=1,sout=0
sim time=35,rst=0,sin=0,sout=0
sim time=45,rst=0,sin=1,sout=0
sim time=55,rst=0,sin=1,sout=1
sim time=65,rst=0,sin=1,sout=0
sim time=75,rst=0,sin=1,sout=1
Simulation complete via $finish(1) at time 155 NS + 0
./testbench.sv:15      #100 $finish;
```

61) Write a Verilog code SIPO.

Design code

```
// Code your design here
module sipo(clk,rst,si,po);
  input clk,rst,si;
  output reg [3:0] po;
  reg [3:0] temp;
  always @(posedge clk or posedge rst)
    begin
      if(rst)
        temp<=0;
      else
          temp<={si,temp[3:1]};
      po<=temp;
    end
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module sipo_test;
  reg clk,rst,si;
  wire [3:0] po;
  sipo dut(clk,rst,si,po);
  initial begin
   clk=0;rst=1;
    #12 rst=0;si=1;
    #12 si=1;
    #12 si=1;
    #12 si=1;
    #12 si=1;
    #200 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $monitor("sim time=%0t,rst=%b,si=%b,po=%b",$time,rst,si,po);
    $dumpfile("dump.vcd");
    $dumpvars(0,clk,rst,si,po);
  end
endmodule
```

Output

```
sim time=0,rst=1,si=x,po=xxxx
sim time=5,rst=1,si=x,po=0000
sim time=12,rst=0,si=1,po=0000
sim time=25,rst=0,si=1,po=1000
sim time=35,rst=0,si=1,po=1100
sim time=45,rst=0,si=1,po=1110
sim time=55,rst=0,si=1,po=1111
Simulation complete via $finish(1) at ti
./testbench.sv:14      #200 $finish;
```

62) Write a Verilog code PISO.

Design code

```
// Code your design here
module piso(clk,rst,cnt,pi,so);
  input clk,rst,cnt;
  input [3:0]pi;
  output reg so;
  reg [3:0]temp;
  always @ (posedge clk)
    begin
      if (rst)
        temp<=0;
      else
        begin
          if(cnt)
            temp<=pi;
          else
            temp<={1'b0,temp[3:1]};
        end
      so<=temp[0];
    end
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module piso_test;
  reg clk,rst,cnt;
  reg [3:0]pi;
  wire so;
  piso dut(clk,rst,cnt,pi,so);
  initial begin
    clk=0;rst=1;
    #12 rst=0; cnt=1; pi=4'b1010;
    #10 cnt=0;
    #200 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $monitor ("sim time=%0t,rst=%b,cnt=%b,pi=%b,so=%b",$time,rst,cnt,pi,so);
    $dumpfile("dump.vcd");
    $dumpvars(0,clk,rst,cnt,pi,so);
  end
endmodule
```

Output

```
sim time=0,rst=1,cnt=x,pi=xxxx,so=x
sim time=12,rst=0,cnt=1,pi=1010,so=x
sim time=15,rst=0,cnt=1,pi=1010,so=0
sim time=22,rst=0,cnt=0,pi=1010,so=0
sim time=35,rst=0,cnt=0,pi=1010,so=1
sim time=45,rst=0,cnt=0,pi=1010,so=0
sim time=55,rst=0,cnt=0,pi=1010,so=1
sim time=65,rst=0,cnt=0,pi=1010,so=0
Simulation complete via $finish(1) at time 222 NS + 0
```

63) Write a Verilog code 3bit up-down counter using FSM.

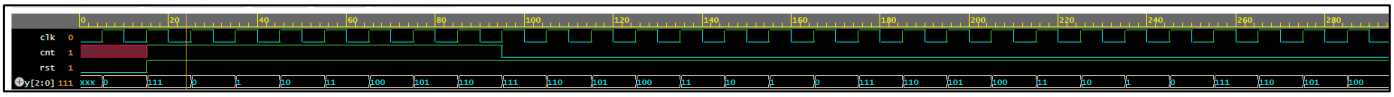
Design code

```
// Code your design here
module up_down3bit(clk,rst,cnt,y);
  input clk,rst,cnt;
  output reg [2:0] y;
  reg [2:0]ps,ns;
  parameter s0=3'd0;
  parameter s1=3'd1;
  parameter s2=3'd2;
  parameter s3=3'd3;
  parameter s4=3'd4;
  parameter s5=3'd5;
  parameter s6=3'd6;
  parameter s7=3'd7;
  always @(posedge clk)
    begin
      if(!rst)
        ps<=s0;
      else
        ps<=ns;
    end
  always @(ps,cnt)
    begin
      case(ps)
        s0:if(cnt)
          begin
            ns<=s1;
            y<=3'd0;
          end
        else
          begin
            ns<=s7;
            y<=3'd0;
          end
         s1:if(cnt)
          begin
            ns<=s2;
            y<=3'd1;
          end
        else
          begin
            ns<=s0;
            y<=3'd1;
          end
         s2:if(cnt)
          begin
            ns<=s3;
            y<=3'd2;
          end
        else
          begin
            ns<=s1;
            y<=3'd2;
          end
         s3:if(cnt)
          begin
            ns<=s4;
            y<=3'd3;
          end
        else
          begin
            ns<=s2;
            y<=3'd3;
          end
          end
         s4:if(cnt)
          begin
            ns<=s5;
            y<=3'd4;
          end
        else
          begin
            ns<=s3;
            y<=3'd4;
          end
         s5:if(cnt)
          begin
            ns<=s6;
            y<=3'd5;
          end
        else
          begin
            ns<=s4;
            y<=3'd5;
          end
         s6:if(cnt)
          begin
            ns<=s7;
            y<=3'd6;
          end
        else
          begin
            ns<=s5;
            y<=3'd6;
          end
         s7:if(cnt)
          begin
            ns<=s0;
            y<=3'd7;
          end
        else
          begin
            ns<=s6;
            y<=3'd7;
          end
        default: begin
          ns<=s0;
          y<=3'd0;
        end
      endcase
    end
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module up_down3bit_test;
  reg clk,rst,cnt;
  wire [2:0] y;
  up_down3bit dut(clk,rst,cnt,y);
  initial begin
    clk=0;rst=0;
    #15 rst=1; cnt=1;
    #80 cnt=0;
    #200 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $monitor("sim time=%0t,clk=%b,rst=%b,cnt=%b,y=%b",$time,clk,rst,cnt,y);
    $dumpfile("dump.vcd");
    $dumpvars(0,clk,rst,cnt,y);
  end
endmodule
```

Output waveform



Output

```
sim time=0,clk=0,rst=0,cnt=x,y=xxx
sim time=5,clk=1,rst=0,cnt=x,y=000
sim time=10,clk=0,rst=0,cnt=x,y=000
sim time=15,clk=1,rst=1,cnt=1,y=111
sim time=20,clk=0,rst=1,cnt=1,y=111
sim time=25,clk=1,rst=1,cnt=1,y=000
sim time=30,clk=0,rst=1,cnt=1,y=000
sim time=35,clk=1,rst=1,cnt=1,y=001
sim time=40,clk=0,rst=1,cnt=1,y=001
sim time=45,clk=1,rst=1,cnt=1,y=010
sim time=50,clk=0,rst=1,cnt=1,y=010
sim time=55,clk=1,rst=1,cnt=1,y=011
sim time=60,clk=0,rst=1,cnt=1,y=011
sim time=65,clk=1,rst=1,cnt=1,y=100
sim time=70,clk=0,rst=1,cnt=1,y=100
sim time=75,clk=1,rst=1,cnt=1,y=101
sim time=80,clk=0,rst=1,cnt=1,y=101
sim time=85,clk=1,rst=1,cnt=1,y=110
sim time=90,clk=0,rst=1,cnt=1,y=110
sim time=95,clk=1,rst=1,cnt=0,y=111
sim time=100,clk=0,rst=1,cnt=0,y=111
sim time=105,clk=1,rst=1,cnt=0,y=110
sim time=110,clk=0,rst=1,cnt=0,y=110
sim time=115,clk=1,rst=1,cnt=0,y=101
sim time=120,clk=0,rst=1,cnt=0,y=101
sim time=125,clk=1,rst=1,cnt=0,y=100
sim time=130,clk=0,rst=1,cnt=0,y=100
sim time=135,clk=1,rst=1,cnt=0,y=011
sim time=140,clk=0,rst=1,cnt=0,y=011
sim time=145,clk=1,rst=1,cnt=0,y=010
sim time=150,clk=0,rst=1,cnt=0,y=010
sim time=155,clk=1,rst=1,cnt=0,y=001
sim time=160,clk=0,rst=1,cnt=0,y=001
sim time=165,clk=1,rst=1,cnt=0,y=000
```

64) Write a Verilog code for 0-2-5-7 using FSM.

Design code

```
// Code your design here
module  fsm0257(clk,rst,y);
  input clk,rst;
  output reg [2:0]y;
  reg [1:0]ps,ns;
  parameter s0=3'd0;
  parameter s2=3'd1;
  parameter s5=3'd2;
  parameter s7=3'd3;
  always @(posedge clk)
    begin
      if(rst)
        ps<=0;
      else
        ps<=ns;
    end
  always @(ps)
    begin
      case(ps)
        s0:begin
          ns<=s2;
          y<=3'd0;
        end
        s2:begin
          ns<=s5;
          y<=3'd2;
        end
        s5:begin
          ns<=s7;
          y<=3'd5;
        end
        s7:begin
          ns<=s0;
          y<=3'd7;
        end
        default:begin
          ns<=s0;
          y<=3'd0;
        end
      endcase
    end
endmodule

```

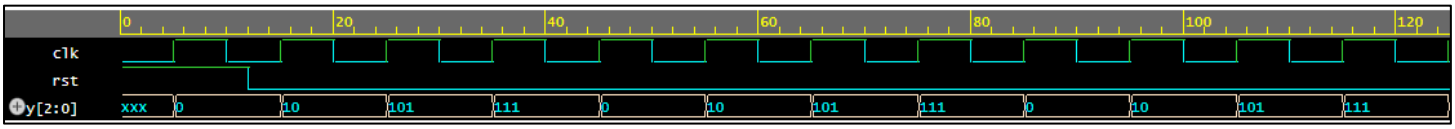
Test bench

```
// Code your testbench here
// or browse Examples
module fsm0257_test;
  reg clk,rst;
  wire [2:0]y;
   fsm0257 dut(clk,rst,y);
  initial begin
    clk=0;rst=1;
    #12 rst=0;
    #200 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $monitor ("sim time=%0t,clk=%b,rst=%b,y=%b",$time,clk,rst,y);
    $dumpfile("dump.vcd");
    $dumpvars(0,clk,rst,y);
  end
endmodule
```

Output

```
sim time=0,clk=0,rst=1,y=xxx
sim time=5,clk=1,rst=1,y=000
sim time=10,clk=0,rst=1,y=000
sim time=12,clk=0,rst=0,y=000
sim time=15,clk=1,rst=0,y=010
sim time=20,clk=0,rst=0,y=010
sim time=25,clk=1,rst=0,y=101
sim time=30,clk=0,rst=0,y=101
sim time=35,clk=1,rst=0,y=111
sim time=40,clk=0,rst=0,y=111
sim time=45,clk=1,rst=0,y=000
sim time=50,clk=0,rst=0,y=000
sim time=55,clk=1,rst=0,y=010
sim time=60,clk=0,rst=0,y=010
sim time=65,clk=1,rst=0,y=101
sim time=70,clk=0,rst=0,y=101
sim time=75,clk=1,rst=0,y=111
sim time=80,clk=0,rst=0,y=111
sim time=85,clk=1,rst=0,y=000
sim time=90,clk=0,rst=0,y=000
sim time=95,clk=1,rst=0,y=010
sim time=100,clk=0,rst=0,y=010
sim time=105,clk=1,rst=0,y=101
sim time=110,clk=0,rst=0,y=101
sim time=115,clk=1,rst=0,y=111
sim time=120,clk=0,rst=0,y=111
sim time=125,clk=1,rst=0,y=000
sim time=130,clk=0,rst=0,y=000
sim time=135,clk=1,rst=0,y=010
sim time=140,clk=0,rst=0,y=010
sim time=145,clk=1,rst=0,y=101
sim time=150,clk=0,rst=0,y=101
sim time=155,clk=1,rst=0,y=111
```

Output Waveform



65) Write a Verilog code for XOR using UDP.

Design code

```
// Code your design here

module xnor_gate_w(output y, input a, input b);
  xnor_gate my_xnor(y, a, b);
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
primitive xnor_gate (out, a, b);
  output out;
  input a, b;
  table
     1 1 : 1;
     0 0 : 1;
     1 0 : 0;
     1 0 : 0;
  endtable
endprimitive
module tb;
  reg a, b;
  wire y;
xnor_gate_w dut(y, a, b);
initial begin
      a = 0; b = 1;
  #10 a = 0; b =1'bx;
  #10 a = 1'bx; b = 0;
  #10 a = 1'bx; b = 1;
  #10 a = 1; b = 1;
  #20  $finish;
  end
  initial begin
  $monitor("a = %b, b = %b, y = %b", a, b, y);
  end
endmodule
```

Output

```
a = 0, b = 1, y = x
a = 0, b = x, y = x
a = x, b = 0, y = x
a = x, b = 1, y = x
a = 1, b = 1, y = 1
```

66) Write a Verilog code for JK FF using FSM.

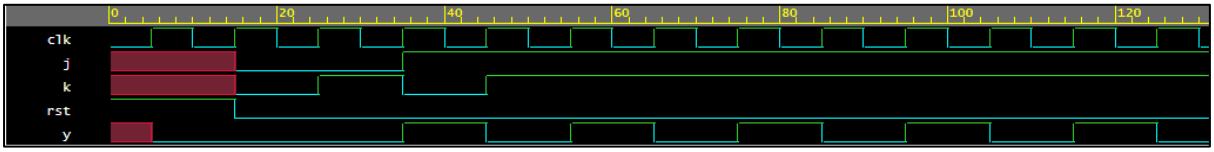
Design code

```
module JK_fsm(clk,rst,j,k,y);
  input clk,rst,j,k;
  output reg y;
  reg ps,ns;
  parameter s0=1'b0;
  parameter s1=1'b1;
  always @(posedge clk)
    begin
      if (rst)
        ps<=s0;
      else
        ps<=ns;
    end
  always @(ps,j,k)
    begin
      case(ps)
        s0:if((j==1&&k==0)||(j==1&&k==1))
          begin
            ns<=s1;
            y<=1;
          end
        else
          begin
            ns<=s0;
            y<=0;
          end
        s1:if((j==0&&k==1)||(j==1&&k==1))
         begin
            ns<=s0;
            y<=0;
          end
        else
          begin
            ns<=s1;
            y<=1;
          end
        default:begin
          ns<=s0;
          y<=0;
        end
      endcase
    end
endmodule
```

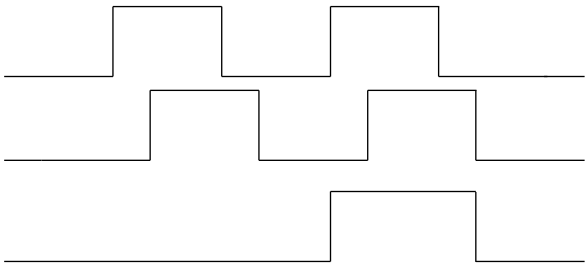
Test bench

```
// Code your testbench here
// or browse Examples
module JK_fsm_test;
  reg clk,rst,j,k;
  wire y;
  JK_fsm dut(clk,rst,j,k,y);
  initial begin
    clk=0;rst=1;
    #15 rst=0;j=0;k=0;
    #10 j=0;k=1;
    #10 j=1;k=0;
    #10 j=1;k=1;
    #200 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0,clk,rst,j,k,y);
  end
endmodule
```

Output waveform



67) Write a Verilog code for



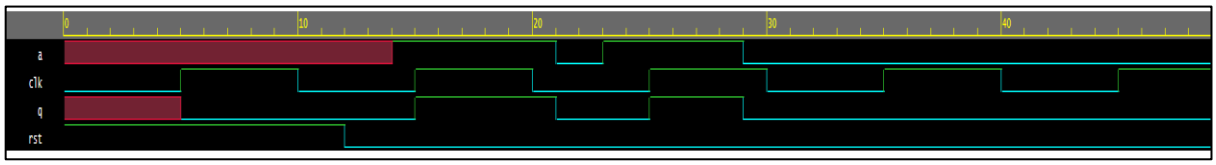
Design code

```
// Code your design here
module clk_gen(clk,rst,a,q);
  input clk,rst,a;
  output reg q;
  always @(posedge clk,posedge a,negedge a)
    begin
      if (rst)
        q<=0;
      else if(a)
        begin
          if (clk)
          q<=1;
        end
      else if (!a)
        q<=0;
      else
        q<=q;
    end
endmodule
```

Test bench

```
// Code your testbench here
// or browse Examples
module clk_gen_test;
  reg clk,rst,a;
  wire q;
  clk_gen dut(clk,rst,a,q);
  initial begin
    clk=0;rst=1;
    #12 rst=0;
    #2 a=1;
    #7 a=0;
    #2 a=1;
    #6 a=0;
    #20 $finish;
  end
  always
    #5 clk=~clk;
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(1,clk_gen_test);
  end
endmodule
```

Output Waveform



68) Write a Verilog code to generate clock of 0.5ns,5ns,3ns time period.

Design code

```
// Code your testbench here
// or browse Examples
`timescale 1ns/1ps;
//****************************************3ns time period**************
/*module test;
  reg clk;
  initial begin
    clk=0;
    #20 $finish;
  end
  always
    #1.5 clk=~clk;
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0,clk);
  end
endmodule*/
//****************************************5ns time period**************
/*module test;
  reg clk;
  initial begin
    clk=0;
    #20 $finish;
  end
  always
    #2.5 clk=~clk;
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0,clk);
  end
endmodule */
//****************************************0.5ns or 500ps time period**
module test;
  reg clk;
  initial begin
    clk=0;
    #20 $finish;
  end
  always
    #0.25 clk=~clk;
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0,clk);
  end
endmodule
```

Output Wave form for 0.5 ns