1)FULL ADDER(Dataflow modelling)(1 bit)

```
libraryieee;
use ieee.std_logic_1164.all;

entity adder is
port (a,b,c : in std_logic;
s,carry :out std_logic);
end adder;
architectureabc of adder is
begin
s <= a xor b xor c;
carry<= (a and b) or (b and c) or (c and a);
endabc;
```

TESTBENCH FOR FULLADDER

```
entitytestbencha is
endtestbencha;
architectureabc of testbencha is
signalx,y,z,d,e: bit;
component adder
port(a,b,c:in bit; s,carry:out bit);
end component;
begin
A1:adder port map(x,y,z,d,e);

Process
begin

x<='0'; y<='0'; z<='1';
wait for 2 ns;
x<='0'; y<='1';z<='0';
wait for 2 ns;
x<='0'; y<='1';z<='1';
wait for 2 ns;
x<='1'; y<='0';z<='0';
wait for 2 ns;
x<='1'; y<='0';z<='1';
wait for 2 ns;
x<='1'; y<='1';z<='0';
wait for 2 ns;
x<='1'; y<='1';z<='1';
wait for 2 ns;
end process ;
endabc ;
```

2)4 BIT Ripple carry adder

```
libraryieee;
use ieee.std_logic_1164.all;
entity adder4 is
port (a,b : in std_logic_vector(3 downto 0) ;
s :out std_logic_vector(3 downto 0);
c0 :instd_logic_vector  ;
cout: out std_logic);
end adder4;
architectureabc of adder4 is
signal c1,c2,c3:std_logic;

component adder
port (a,b,c : in std_logic;
s,carry :out std_logic);
end component;
begin
F1: adder port map (a(0),b(0),c0,s(0),c1);
F2: adder port map (a(1),b(1),c1,s(1),c2);
F3: adder port map (a(2),b(2),c2,s(2),c3);
F4: adder port map (a(3),b(3),c3,s(3),cout);
endabc;
```

TEST BENCH FOR 4 BIT ADDER

```
entitytestbencha is
endtestbencha;
architectureabc of testbencha is
signalx,y,z,d,e: bit;
component adder
port(a,b,c:in bit; s,carry:out bit);
end component;
begin
A1:adder port map(x,y,z,d,e);

Process
begin

x<='0'; y<='0'; z<='1';
wait for 2 ns;
x<='0'; y<='1';z<='0';
wait for 2 ns;
x<='0'; y<='1';z<='1';
wait for 2 ns;
x<='1'; y<='0';z<='0';
wait for 2 ns;
x<='1'; y<='0';z<='1';
```

```
wait for 2 ns;
x<='1'; y<='1';z<='0';
wait for 2 ns;
x<='1'; y<='1';z<='1';
wait for 2 ns;
end process ;
endabc ;
```

3)FULL SUBTRACTOR(Dataflow)

```
libraryieee;
use ieee.std_logic_1164.all;
entitySubr is
port (a,b,c : in bit;
d,e :out bit);
endsubr ;
architectureabc of
subr is
begin
d <= a xor b xor c;
e <= (a and b) or (not b and c) or (not c and b);
endabc;
```

TESTBENCH FOR FULL SUBTRACTOR

```
entitytestbench is
endtestbench;
architectureabc of testbench is
signalx,y,z,m,n: bit;
componentsubr
port(a,b,c:in bit; d,e:out bit);
end component;
begin
A1:subr port map(x,y,z,m,n);

Process
begin

x<='0'; y<='0'; z<='1';
wait for 2 ns;
x<='0'; y<='1';z<='0';
wait for 2 ns;
x<='0'; y<='1';z<='1';
wait for 2 ns;
x<='1'; y<='0';z<='0';
wait for 2 ns;
x<='1'; y<='0';z<='1';
wait for 2 ns;
x<='1'; y<='1';z<='0';
```

```
wait for 2 ns;
x<='1'; y<='1';z<='1';
wait for 2 ns;
end process ;
endabc ;
```

Aim:Design ,Analysis And Circuit Simulation Of 4X1 MULTIPLEXER and1x4 DEMULTIPLEXER.

1)4X1 MULTIPLEXER

```
libraryieee;
use ieee.std_logic_1164.all;
entity MUX is
port(D0,D1,D2,D3:in std_logic;
S:instd_logic_vector(1 downto 0); Z:out std_logic);
end MUX;
architectureabc of MUX is
begin
process(en)
begin
caseS is
when"00"=>Z<=D0;
when"01"=>Z<=D1;
when"10"=>Z<=D2;
when"11"=>Z<=D3;
when others=>null;
end case;
end process;
endabc;
```

TESTBENCH FOR MULTIPLEXER IS:

```
libraryieee;
use ieee.std_logic_1164.all;
entitytestmux is
endtestmux;
architectureabc of testmux is

component MUX
port(D0,D1,D2,D3:in std_logic;
S:instd_logic_vector(1 downto 0);
Z:outstd_logic);
end component;
signala,b,c,d:std_logic;
signal e:std_logic_vector(1 downto 0);
signal g:std_logic;
begin
M1:MUX port map(a,b,c,d,e,g);
```

```
process
begin
a<='1';b<='1';c<='0';d<='1';e<="00";
wait for 2 ns;
a<='0';b<='0';c<='1';d<='1';e<="01";
wait for 2 ns;
a<='0';b<='1';c<='1';d<='0';e<="10";
wait for 2 ns;
a<='1';b<='0';c<='0';d<='0';e<="11";
wait for 2 ns;
end process;
endabc;
```

2)DEMULTIPLEXER

```
libraryieee;
use ieee.std_logic_1164.all;
entitydemux is
port(A:instd_logic;
S:instd_logic_vector(1 downto 0);
D:outstd_logic_vector(3 downto 0));
enddemux;

architectureabc of demux is
begin
process(en)
begin

case en is
when"00"=>D(0)<=A;D(1)<='0';D(2)<='0';D(3)<='0';

when"01"=>D(0)<='0';D(1)<=A;D(2)<='0';D(3)<='0';

when"10"=>D(0)<='0';D(1)<='0';D(2)<=A;D(3)<='0';

when"11"=>D(0)<='0';D(1)<='0';D(2)<='0';D(3)<=A;
when others=> null;
end case;
end process;
endabc;
```

TESTBENCH FOR DEMULTIPLEXER

```
libraryieee;
use ieee.std_logic_1164.all;
entitytbdemux is
endtbdemux;
architecture xyz of tbdemux is
componentdemux
port(A:instd_logic;en:instd_logic_vector(1 downto 0);D:outstd_logic_vector(3 downto 0));
end component;
signal A:std_logic;
signalen:std_logic_vector(1 downto 0);
signal D:std_logic_vector(3 downto 0);
begin
M1:demux port map(A,en,D);
process
begin
A<='0';en<="00";
wait for 2 ns;
```

```
A<='1';en<="10";
wait for 2 ns;
A<='0';en<="01";
wait for 2 ns;
A<='1';en<="11";
wait for 2 ns;
end process;
end xyz;
```

AI<:Design ,Analysis And Circuit Simulation Of 8x3 Encoder And 3x8 Decoder.

1)3X8 ENCODER

```
libraryieee;
use ieee.std_logic_1164.all;
entity enc1 is
port(D:instd_logic_vector(7 downto 0);en:instd_logic; A:out std_logic_vector(2 downto 0));
end enc1;
architectureabc of enc1 is
begin
process(en,D)
begin

if (en='1')then
case D is
        when"00000001"=> A<="000";
        when"00000010"=>  A<="001";
        when"00000100"=>  A<="010";
        when"00001000"=> A<="011";
        when"00010000"=>  A<="100";
        when"00100000"=>  A<="101";
        when"01000000"=> A<="110";
        when"10000000"=> A<="111";
        when others=> null;

        end case;
        else A<="000";
        end if;
        end process;
        endabc;
```

TESTBENCH FOR 3X8 ENCODER

```
libraryieee;
use ieee.std_logic_1164.all;
entitytbenc is
```

```
endtbenc;
architecture xyz of tbenc is
component enc1
            port(D:instd_logic_vector(7 downto 0);en:instd_logic; A:out std_logic_vector(2
downto 0));
end component;
signal e:std_logic;
signal D:std_logic_vector(7 downto 0);
signal A:std_logic_vector(2 downto 0);
begin
 ENCOD: enc1 port map(D,e,A);
process
begin

D<="00000001"; e<='1';
wait for 2 ns;
D<="00000010";
wait for 2 ns;
D<="00000100";
wait for 2 ns;
D<="00001000";
wait for 2 ns;
D<="00010000";
wait for 2 ns;
D<="00100000";e<='0';
wait for 2 ns;
D<="01000000";
wait for 2 ns;
D<="10000000";
wait for 2 ns;
end process;
end xyz;
```

2)8X3 DECODER

```
libraryieee;
use ieee.std_logic_1164.all;
entity dec3to8 is
port(dout:outstd_logic_vector(7 downto 0);
din:instd_logic_vector (2 downto 0));
end dec3to8;
architecture enc of dec3to8 is
begin
process(din)
begin
case din is
when "000"=>dout<="00000001";
when "001"=>dout<="00000010";
when "010"=>dout<="00000100";
when "011"=>dout<="00001000";
when "100"=>dout<="00010000";
when "101"=>dout<="00100000";
when "110"=>dout<="01000000";
when "111"=>dout<="10000000";
when others=> null;
end case;
end process;
end enc;
```

TESTBENCH FOR 8X3 DECODER

```
libraryieee;
use ieee.std_logic_1164.all;

entity dec3to8tb is
end dec3to8tb;
architectureabc of dec3to8tb is
signal y:std_logic_vector(7 downto 0);
signal z:std_logic_vector(2 downto 0);
component dec3to8
port(dout:outstd_logic_vector(7 downto 0);
din:instd_logic_vector (2 downto 0));
end component;
begin
A1:dec3to8 port map(y,z);
process
begin
z<="001";
wait for 2 ns;
```

```
z<="111";
wait for 2 ns;

end process;
endabc;
```

1)D FLIP FLOP

```
libraryieee;
use ieee.std_logic_1164.all;

entitydff is
port(d,clk:in bit; q,q1:out bit);
enddff;

architecturearch_dff of dff is
signal temp : bit:='0';
begin
process(d,clk)
begin
ifclk='1' and clk 'event
then
temp<=d;
end if;
end process;
q<=temp;
q1<= not temp;
endarch_dff;
```

TESTBENCH FOR D FLIP FLOP

```
libraryieee;
use ieee.std_logic_1164.all;
entitydff_tb is
enddff_tb;

architecturearch_dfftb of dff_tb is
componentdff
port(d,clk: in bit; q,q1:out bit);
end component;

signal d,clk,q,q1: bit:='0';
begin
a1:dff port map (d,clk,q,q1);
clk<=not clk after 25 ns;
d<= not d after 100 ns;
endarch_dfftb;
```

2)T FLIP FLOP

```
libraryieee;
use ieee.std_logic_1164.all;
entitytff is
port(CLK,reset:instd_logic;T:instd_logic;Q:outstd_logic);
endtff;
architectureabc of tff is
signalqin:std_logic:='0';
begin
process (CLK,reset)
begin
if(reset='0')then
qin<='0';
elsif(clk' event and CLK='1')then
case T is
when '0'=>qin<=qin;
when '1'=>qin<= not qin;
when others=>null;
end case;
end if;
end process;
 Q<=qin;
endabc;
```

TESTBENCH FOR T FLIP FLOP

```
libraryieee;
use ieee.std_logic_1164.all;

entitytbtff is
endtbtff;

architecture xyz of tbtff is
componenttff
port(CLK,reset:instd_logic;T:instd_logic;Q:outstd_logic);
end component;
signalCLK:std_logic:='0';
signalreset:std_logic:='1';
signal t:std_logic;
signal q:std_logic;
begin
t1:tff port map(clk,reset,t,q);
clk<=not clk after 5 ns;
reset<=not reset after 40 ns;
```

```
process
begin
 t<='0';
wait for 10 ns;
 t<='1';
wait for 10 ns;
end process;
end xyz;
```

3)SR FLIP FLOP

```
libraryieee;
use ieee.std_logic_1164.all;

entitySRff is
port(SR:instd_logic_vector(1 downto 0);CLK,reset:instd_logic;Q:inoutstd_logic);
endSRff;

architectureabc of SRff is
signalqin:std_logic:='1';
begin

process(CLK,reset)
begin


if(reset='0')then
qin<='0';
elsif(clk' event and CLK='1') then
case SR is
when"00"=>qin<=qin;
when"01"=>qin<='0';
when"10"=>qin<='1';
when"11"=>qin<='X';
when others=>null;
end case;
end if;
end process;
  Q<=qin;

endabc;
```

TESTBENCH FOR SR FLIP FLOP

```
libraryieee;
use ieee.std_logic_1164.all;

entity TBSR is
end TBSR;

architecture xyz of TBSR is
componentSRff
port(SR:std_logic_vector(1 downto 0);CLK,reset:instd_logic;Q:inoutstd_logic);
end component;
signalSR:std_logic_vector(1 downto 0);
signalclk:std_logic:='0';
signalreset:std_logic:='1';
signal q:std_logic;



begin
d1:SRff port map(SR,clk,reset,q);
clk<= not clk after 5 ns;
reset<= not reset after 40 ns;
process
begin
SR<="00";
wait for 10 ns;
SR<="01";
wait for 10 ns;
SR<="10";
wait for 10 ns;
SR<="11";
wait for 10 ns;
end process;
end xyz;
```

4)JK FLIP FLOP

```
libraryieee;
use ieee.std_logic_1164.all;
entity JK is
port(J,K,CLK:instd_logic;Q:inoutstd_logic);
end JK;
architectureabc of JK is
signaltemp:std_logic_vector(1 downto 0);
begin

process(CLK,temp)
begin
temp<=J&K;
if(CLK='1') then
case temp is
when"00"=> Q<=Q;
when"01"=>Q<='0';
when"10"=>Q<='1';
when "11"=>Q<=not Q;
when others=>Q<='X';

end case;
else
  Q<=Q;
end if;
end process;
endabc;
```

TESTBENCH FOR JK FLIP FLOP

```
libraryieee;
use ieee.std_logic_1164.all;
entity TBJK1 is
end TBJK1;
architecture xyz of TBJK1 is
component JK1
port(JK:instd_logic_vector(1 downto 0);CLOCK,reset:instd_logic;Q:inoutstd_logic);
end component;
signaljk:std_logic_vector(1 downto 0);
signalclock:std_logic:='0';
signalreset:std_logic:='1';
signal q:std_logic;
begin
d1:JK1 port map(jk,clock,reset,q);
clock<=not clock after 5 ns;
```

```
reset<=not reset after 40 ns;
process
begin
jk<="00";
wait for 10 ns;
jk<="01";
wait for 10 ns;
jk<="10";
wait for 10 ns;
jk<="11";
wait for 10 ns;
end process;
end xyz;
```

Aim:Design ,Analysis And Circuit Simulation OF 4BIT UPDOWN COUNTER.

UPDOWN COUNTER:

```
libraryieee;
use ieee.std_logic_1164.all;
useieee.std_logic_arith.all;
useieee.std_logic_unsigned.all;

entityupdown is
   Port ( clk : in  std_logic;
rst : in  std_logic;
UD : in  std_logic;
Q : out  std_logic_vector (3 downto 0));
endupdown;

architectureBehavioral of updown is
signaltmp:std_logic_vector (3 downto 0);
begin
process(clk,rst)
begin
if(rst='1') then
tmp<="0000";
elsif(clk'event and clk='1') then
if(UD='1') then
tmp<=tmp+1;
else
tmp<=tmp-1;
end if;
end if;
end process;
Q <= tmp;
endBehavioral;
```

TESTBENCH FOR UPDOWN COUNTER

```
libraryieee;
use ieee.std_logic_1164.all;

entitytestupdown is
endtestupdown;

architecturetestupdownarch of testupdown is
```

```
// always assign to 0
signal x:std_logic:='0';
signaly,z: std_logic;
signal w:std_logic_vector(3 downto 0);

componentupdown
port(
            clk : in  std_logic;
rst : in  std_logic;
UD : in  std_logic;
Q : out  std_logic_vector (3 downto 0));
end component;

begin
a1:updown port map(x,y,z,w);

process
begin
x<=not x;
wait for 5 ns;
end process;


process
begin

y<='1';
wait for 10 ns;

y<= '0';


 z<= '1';
wait for 30 ns;
z<= '0';

wait for 50 ns;
z<= '1';
wait for 30 ns;


end process;

endtestupdownarch;
```

Aim:Design ,Analysis And Circuit Simulation OF 4 BIT Shift Registers(SISO,SIPO,PISO,PIPO).

Serial In Serial Out

```
libraryieee;
use ieee.std_logic_1164.all;
entitysiso is
port(input:instd_logic;clk:instd_logic;output:outstd_logic);
endsiso;
architecturesiso_arch of siso is
signaltemp:std_logic_vector(2 downto 0):="000";
begin
process(input,clk)
begin
if(clk='1' and clk'event) then
output<=temp(2);
temp(2)<=temp(1);
temp(1)<=temp(0);
temp(0)<=input;
end if;
end process;
endsiso_arch;
```

Testbench:

```
libraryieee;
use ieee.std_logic_1164.all;
entitysiso_tb is
endsiso_tb;
architecturesisotb_arch of siso_tb is
componentsiso
port(input:instd_logic;clk:instd_logic;output:outstd_logic);
end component;
signalinput,clk,output:std_logic:='0';
begin
a1:siso port map(input,clk,output);
clk<=not clk after 25 ns;
process
begin
input<='1';
wait for 50 ns;
input<='0';
wait for 50 ns;
input<='1';
wait for 50 ns;
end process;
endsisotb_arch;
```

2)Serial In Parallel Out(SIPO)

```
libraryieee;
use ieee.std_logic_1164.all;
entitysipo is
port(input:instd_logic;clk:instd_logic;output:outstd_logic_vector(3 downto 0));
endsipo;
architecturesipo_arch of sipo is
signaltemp:std_logic_vector(3 downto 0):="0000";
begin
process(input,clk)
begin

if(clk='1' and clk'event) then
temp(3)<=temp(2);
temp(2)<=temp(1);
temp(1)<=temp(0);
temp(0)<=input;
end if;
output<=temp;
end process;
endsipo_arch;
```

Testbench:

```
libraryieee;
use ieee.std_logic_1164.all;
entitysipo_tb is
endsipo_tb;
architecturesipotb_arch of sipo_tb is
componentsipo
port(input:instd_logic;clk:instd_logic;output:outstd_logic_vector(3 downto 0));
end component;
signaloutput:std_logic_vector(3 downto 0):="0000";
signalinput,clk:std_logic:='0';
begin
a1:sipo port map(input,clk,output);
clk<=not clk after 100 ns;
process
begin
input<='1';
wait for 100 ns;
input<='0';
wait for 100 ns;
input<='1';
wait for 100 ns;
end process;
```

```
endsipotb_arch;
```

3)Parallel In Serial Out(PISO)

```
libraryieee;
use ieee.std_logic_1164.all;
entitypiso is
port(input:instd_logic_vector(3 downto 0);load,clk:instd_logic;output:outstd_logic);
endpiso;
architecturepiso_arch of piso is
signaltemp:std_logic_vector(3 downto 0):="0000";
begin
process(input,clk)
begin
if load='1' then
temp<=input;
else
if(clk='1' and clk'event) then
output<=temp(3);
temp(3)<=temp(2);
temp(2)<=temp(1);
temp(1)<=temp(0);
end if;
end if;
end process;
endpiso_arch;
```

Testbench:

```
libraryieee;
use ieee.std_logic_1164.all;
entitypiso_tb is
endpiso_tb;
architecturepisotb_arch of piso_tb is
componentpiso
port(input:instd_logic_vector(3 downto 0);load,clk:instd_logic;output:outstd_logic);
end component;
signalinput:std_logic_vector(3 downto 0):="0000";
signalload,clk,output:std_logic:='0';
begin
a1:piso port map(input,load,clk,output);
clk<=not clk after 100 ns;
process
begin
input<="1010";
load<='1';
wait for 10 ns;
load<='0';
wait for 1000 ns;
input<="1111";
load<='1';
wait for 10 ns;
load<='0';
wait for 1000 ns;
end process;
endpisotb_arch;
```

4)Parallel In Parallel Out

```
libraryieee;
use ieee.std_logic_1164.all;
entitypipo is
port(input:instd_logic_vector(3 downto 0);load,clk:instd_logic;output:outstd_logic_vector(3
downto 0));
endpipo;
architecturepipo_arch of pipo is
signaltemp:std_logic_vector(3 downto 0):="0000";
begin
process(input,clk)
begin
if load='1' then
temp<=input;
else
if(clk='1' and clk'event) then
temp(3)<=temp(2);
temp(2)<=temp(1);
temp(1)<=temp(0);
end if;
output<=temp;
end if;
end process;
endpipo_arch;
```

Testbench:

```
libraryieee;
use ieee.std_logic_1164.all;
entitypipo_tb is
endpipo_tb;
architecturepipotb_arch of pipo_tb is
componentpipo
```

```
port(input:instd_logic_vector(3 downto 0);load,clk:instd_logic;output:outstd_logic_vector(3
downto 0));
end component;
signalinput,output:std_logic_vector(3 downto 0):="0000";
signalload,clk:std_logic:='0';
begin
a1:pipo port map(input,load,clk,output);
clk<=not clk after 100 ns;
process
begin
input<="1011";
wait for 90 ns;
load<='1';
wait for 20 ns;
load<='0';
wait for 40 ns;
input<="1100";
wait for 100 ns;
end process;
endpipotb_arch;
```