COUNTER

--incremment by 1, decrement by 1, lead of the input

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity Count is

port(clk, reset, inc, dec,load : in std\_logic;

input : in std\_logic\_vector(3 downto 0);

output : out std\_logic\_vector(3 downto 0));

end;

architecture beh of Count is

signal update : std\_logic\_vector(3 downto 0);

begin

process(clk, reset)

begin

if(reset = '1') then

update<= "0000";--reset

elsif(clk'event and clk='1') then

if(inc = '1') then update<= input + "0001";--increment by 1

elsif(dec = '1') then update<= input - "0001";--decrement by 1

elsif(load ='1') then update<= input;--loading the input in

end if;

end if;

end process;

output<= update;

end beh;

**Shifter**

--this code does a shift right, rotate right, rotate left

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity shift is

port(clk, clear, sr, rr,rl: in std\_logic;--clk, clear, shift right, rotate right, rotate left

input : in std\_logic\_vector(3 downto 0);

output : out std\_logic\_vector(3 downto 0));

end;

architecture beh of shift is

signal update : std\_logic\_vector(3 downto 0);

begin

process(clk, clear, sr, rr,rl)

begin

if(clear = '1') then update <= "0000";--Clear

elsif (clk'event and clk = '1') then

if(sr = '1') then

--shift right

update<= '0' & input(3 downto 0);

elsif (rr = '1') then

update(3)<=input(0);

update(2)<=input(3);

update(1)<=input(2);

update(0)<=input(1);

elsif (rl='1') then

update(0)<=input(3);

update(3)<=input(2);

update(2)<=input(1);

update(1)<=input(0);

end if;

end if;

end process;

output <= update;

end beh;

**FSM**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity fsmhw is

Port(reset,clk: in std\_logic;

rc,inc,dec,rl,rr: out std\_logic);

end;

architecture bev of fsmhw is

signal cs,ns: std\_logic\_vector(2 downto 0);

begin

process(reset,clk)

begin

if(reset='1') then

cs<="000";

elsif(clk='1' and clk'event) then

cs<=ns;

end if;

end process;

process(cs)

begin

case cs is

when "000"=>--state 0

rc<='1';--reset counter and clear the bits

rl<='0';

rr<='0';

inc<='0';

dec<='0';

ns<="001";

when"001"=>--state 1

rl<='0';

rr<='1';--rotate right

inc<='0';

dec<='0';

ns<="010";

when"010"=>--state 2

rl<='0';

rr<='0';

inc<='1';--increment by 1

dec<='0';

ns<="011";

when"011"=>--state 3

rl<='1';--rotate left

rr<='0';

inc<='0';

dec<='0';

ns<="100";

when"100"=>--state 4

rl<='0';

rr<='0';

inc<='0';

dec<='1';--decrement by 1

ns<="000";

When others=>null;

end case;

end process;

end;

**TopLevel**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity TopLevel is

port(ResetClear, clk : in std\_logic;

I : in std\_logic\_vector(3 downto 0);--Input

OShift, OCount : out std\_logic\_vector(3 downto 0));--outputs for counter and shift register

end;

architecture beh of TopLevel is

signal rc,inc,dec,sr,rr,load,rl : std\_logic;

--Counter Component Declaration

component Count is

port(clk, reset, inc, dec,load : in std\_logic;

input : in std\_logic\_vector(3 downto 0);

output : out std\_logic\_vector(3 downto 0));

end component;

--Shift Register Component Declaration

component shift is

port(clk, clear, sr, rr,rl: in std\_logic;

input : in std\_logic\_vector(3 downto 0);

output : out std\_logic\_vector(3 downto 0));

end component;

--FSM Component Declaration

component fsmhw is--With Feedback

Port(reset, clk: in std\_logic;

rc, inc, dec, rl, rr : out std\_logic);

end component;

begin

--Component Instantiation

Counter: Count port map(clk, ResetClear, inc, dec,load, I, OCount);

ShiftRegister : shift port map(clk, ResetClear, sr, rr,rl, I, OShift);

FSMan : fsmhw port map(ResetClear, clk, rr, inc, rl,dec);

end beh;

**Testbench**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity TB\_hw7 is

end;

architecture beh of TB\_hw7 is

--input signals

signal reset, clk : std\_logic;

signal Input : std\_logic\_vector(3 downto 0);

--output signals

signal ShiftOutput, CountOutput : std\_logic\_vector(3 downto 0);

--TopLevel Component Declaration

component TopLevel is

port(ResetClear, CLK : in std\_logic;

I : in std\_logic\_vector(3 downto 0);--Input

OShift, OCount : out std\_logic\_vector(3 downto 0));--outputs for counter and shift register

end component;

begin

--Component Instantiation

DUT : TopLevel port map(reset, clk, Input, ShiftOutput, CountOutput);

--Clock Process

process

begin

wait for 10 ns;

Input <= "1001";

clk <= '0';

reset<= '1';

wait for 10 ns;

clk <= '1';

reset<='0';

wait for 10 ns;

clk <= '0';

reset<='0';

wait for 10 ns;

clk <= '1';

reset<='0';

wait for 10 ns;

clk <= '0';

reset<='0';

wait for 10 ns;

clk <= '1';

reset<='0';

wait for 10 ns;

clk <= '0';

reset<='0';

wait for 10 ns;

clk <= '1';

reset<='0';

wait for 10 ns;

clk <= '0';

reset<='0';

wait for 10 ns;

clk <= '1';

reset<='0';

wait for 10 ns;

clk <= '0';

reset<='0';

wait for 10 ns;

clk <= '1';

reset<='0';

wait for 10 ns;

clk <= '0';

reset<='0';

wait for 10 ns;

end process;

end beh;

