COUNTER

--4 Bit Counter that can Asynchronous reset, increment by 1 and decrement by 2

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity Counter\_HW7 is

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

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

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

end;

architecture beh of Counter\_HW7 is

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

begin

process(clk, reset)

begin

if(reset = '1') then

tmp<= "0000";--reset

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

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

elsif(dec = '1') then tmp<= INPUT - "0010";--decrement by 2

end if;

end if;

end process;

OUTPUT<= tmp;

end beh;

SHIFT REGISTER

--4 Bit Shift Register that can clear, shift to right with the lsb being filled by 1 bit, and rotate its contents to the right

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity ShiftRegister\_HW7 is

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

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

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

end;

architecture beh of ShiftRegister\_HW7 is

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

begin

process(clk, clear, sr, rr)

begin

if(clear = '1') then tmp <= "0000";--clear the system

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

if(sr = '1') then

--shift right

tmp <= '0' & INPUT(3 downto 1);

elsif (rr = '1') then

--rotate right (idk if correct)

tmp<= INPUT(0) & INPUT(3 downto 1);

--tmp(3)<=INPUT(0);

--tmp(2)<=INPUT(3);

--tmp(1)<=INPUT(2);

--tmp(0)<=INPUT(1);

end if;

end if;

end process;

OUTPUT <= tmp;

end beh;

FSM

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

Entity FSM\_HW7 is--With Feedback

Port(RESET, CLK: in std\_logic;

rc, inc, dec, sr, rr : out std\_logic);--outputs for counter and shift

End;

Architecture Behavioral of FSM\_HW7 is

signal CS,NS: std\_logic\_vector(2 downto 0); --Current and Next State

Begin

Process(RESET,CLK)

Begin

if (RESET = '1')then-- When RESET is active

CS<= "000";--Reset State Machine

elsif(CLK='1' and CLK'EVENT) then

CS<=NS;--When Clock is active, move to next state

end if;

end process;

process (CS)

begin

case CS is

when "000"=>--State 0

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

sr<= '0';

rr<='0';

inc<='0';

dec<='0';

NS<="001";

When "001"=>--State 1

sr <= '1';--shift the input right

rr<='0';

inc<='0';

dec<='0';

NS<= "010";

When "010"=>--State 2

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

sr<= '0';

rr<='0';

dec<='0';

NS<="011";

When "011"=>--State 3

rr <= '1' ;--rotate the input right

sr<= '0';

inc<='0';

dec<='0';

NS <= "100";

When "100"=>--State 4

dec <= '1';--decrement the input by 2

sr<= '0';

rr<='0';

inc<='0';

NS<= "000";

When others => NULL;

end case;

end process;

end Behavioral;

TOP LEVEL

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity TopLevel\_HW7 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\_HW7 is

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

--Counter Component Declaration

component Counter\_HW7 is

port(clk, reset, inc, dec : 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 ShiftRegister\_HW7 is

port(clk, clear, sr, rr: 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 FSM\_HW7 is--With Feedback

Port(RESET, CLK: in std\_logic;

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

end component;

begin

--Component Instantiation

Counter: Counter\_HW7 port map(CLK, ResetClear, inc, dec, I, OCount);

ShiftRegister : ShiftRegister\_HW7 port map(CLK, ResetClear, sr, rr, I, OShift);

FSM : FSM\_HW7 port map(ResetClear, CLK, rc, inc, dec, sr, rr);

end beh;

TEST BENCH

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\_HW7 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

UUT : TopLevel\_HW7 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';

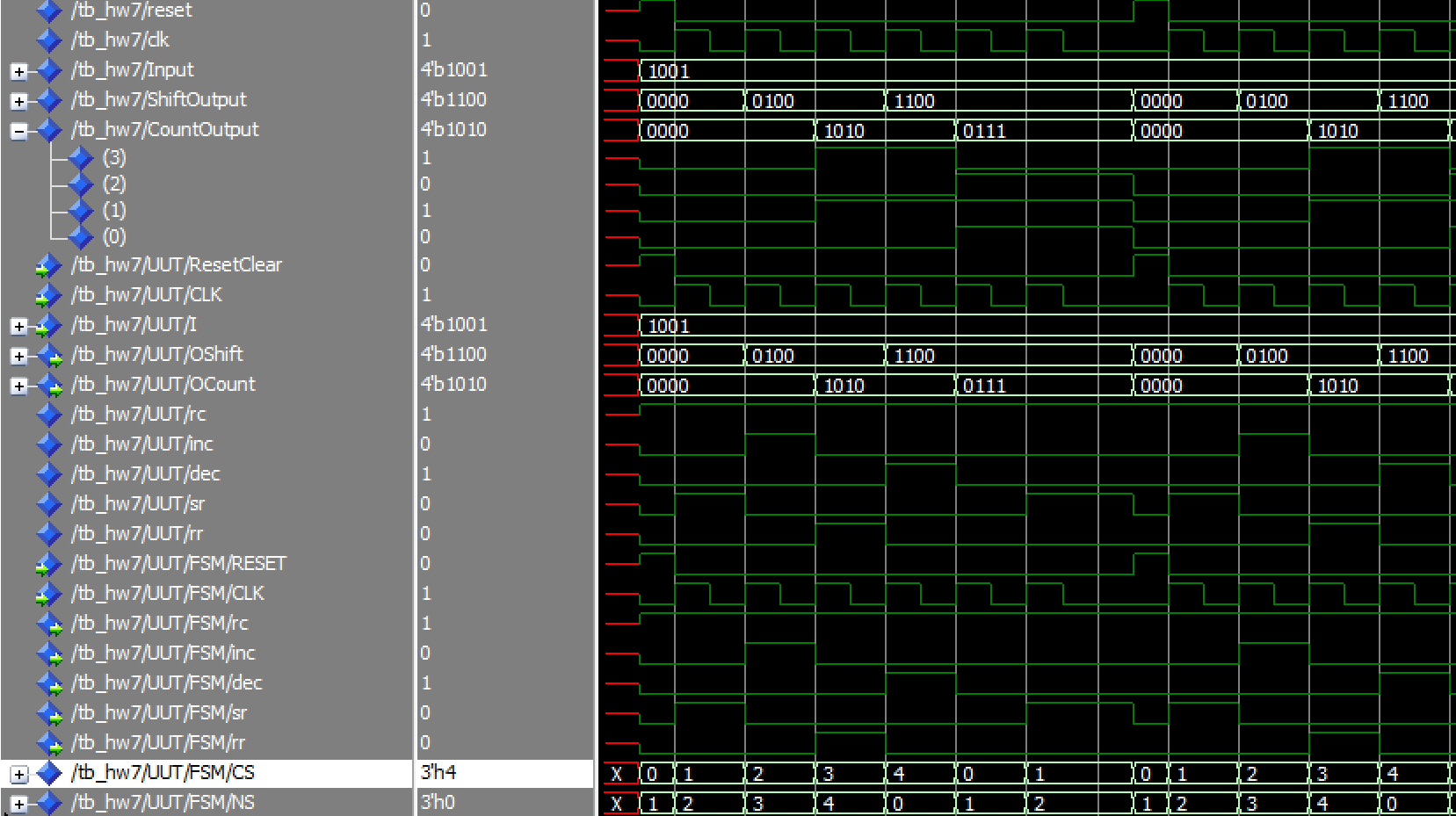
reset<='0';

wait for 10 ns;

end process;

end beh;

Simulation



**\*NOTE\*** For this system notice that the FSM repeats after state 1 during every other iteration. Though functionally it does carry out all the required commands.

YOUTUBE: https://www.youtube.com/watch?v=wPADbX2MqfQ