**UNIVERSALSHIFT CODE**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity USR31 is

Port ( Din : in STD\_LOGIC\_VECTOR (3 downto 0);

clk : in STD\_LOGIC;

rst : in STD\_LOGIC;

S : in STD\_LOGIC\_VECTOR (1 downto 0);

Dout :inout STD\_LOGIC\_VECTOR (3 downto 0));

end USR31;

architecture Behavioral of USR31 is

signal msbin, lsbin: STD\_LOGIC;

begin

process(clk,rst)

begin

if(rst='1') then

dout<="0000";

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

msbin<=din(3);

lsbin<=din(0);

case s is

when "00"=>dout<=dout;--hold

when "01"=>dout<=msbin&dout(3 downto 1);--right shift

when "10"=>dout<=dout(2 downto 0) &lsbin;--left shift

when "11"=>dout<=din;--parallel

when others=>dout<="XXXX";

end case;

end if;

end process;

end Behavioral;

**TEST BENCH**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY USR\_test IS

END USR\_test;

ARCHITECTURE behavior OF USR\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT USR

PORT(

din : IN std\_logic\_vector(3 downto 0);

clk : IN std\_logic;

rst : IN std\_logic;

s : IN std\_logic\_vector(1 downto 0);

dout : INOUT std\_logic\_vector(3 downto 0)

);

END COMPONENT;

--Inputs

signal din :std\_logic\_vector(3 downto 0) := (others => '0');

signal clk :std\_logic := '0';

signal rst :std\_logic := '0';

signal s :std\_logic\_vector(1 downto 0) := (others => '0');

--BiDirs

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

--Clock period definitions

constant clk\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: USR PORT MAP (

din => din,

clk =>clk,

rst =>rst,

s => s,

dout =>dout

);

--Clock process definitions

clk\_process :process

begin

clk<= '0';

wait for clk\_period/2;

clk<= '1';

wait for clk\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 100 ns;

rst<='1';

Assert dout<= "0000" report "error";

wait for clk\_period\*10;

din <= "1000"; rst<='0'; S<="01";

wait for clk\_period/2;

Assert dout<="1100" report "error";

wait for clk\_period/2;

Assert dout<="1110" report "error";

wait for clk\_period/2;

Assert dout<="1111" report "error";

wait for clk\_period/2;

-- wait for clk\_period\*10;

-- insert stimulus here

wait;

end process;

END;