EXPERIMENT NO. 4

UNIVERSAL SHIFT REGISTER

AIM: To write VHDL code, simulate with test bench, synthesis, implement on PLD for Universal Shift Register.

SOFTWARE TOOL: Xilinx, FPGA Kit.

THEORY:

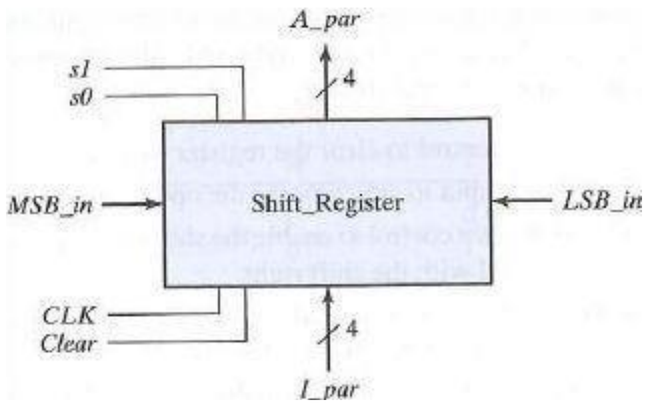
A unidirectional shift register is a register that can capable of transferring data in only one direction. Whereas the register that is capable of transferring data in both left and right direction is called a 'bidirectional shift register.' Now let we have a register which can capable to transfer data in both the shift-right and shift-left, along with the necessary input and output terminals for parallel transfer, then it is called a *shift register* with *parallel load* or 'universal shift register.'

Now what are the parameters should have in a Universal Shift Register. The list of those parameters discussed below

. A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift-right.

1. A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift-left.
2. A parallel-load control to enable a parallel transfer and the input lines associated with the parallel transfer.
3. Parallel output lines.
4. A *RST* control to clear the register to 0.
5. A *CLK* input for clock pulses to synchronize all operations.
6. A control state that leaves the information in the register unchanged even though clock pulses are continuously applied

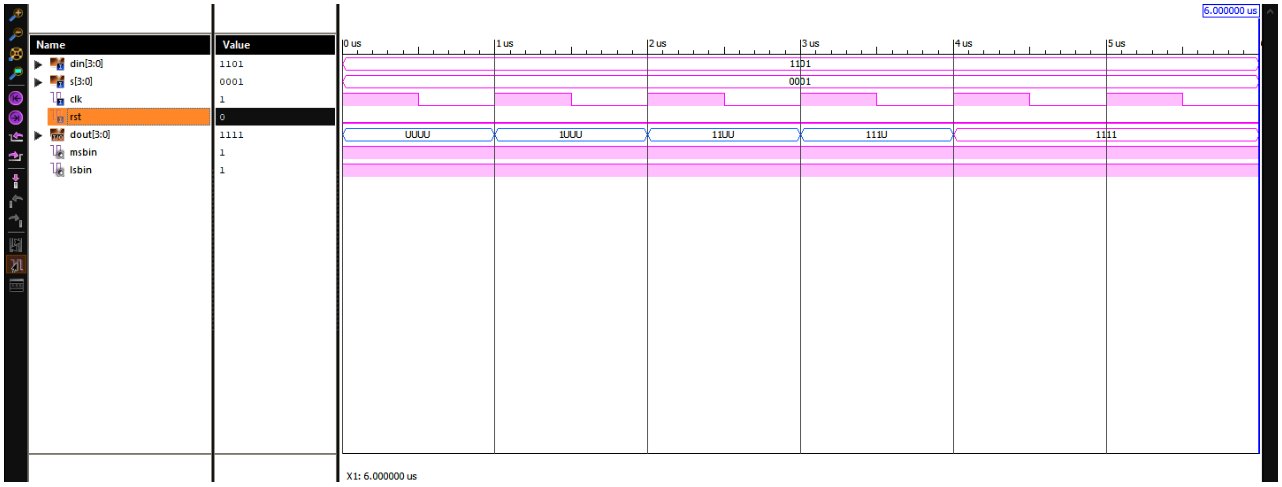
DIAGRAM OF SHIFT REGISTER:



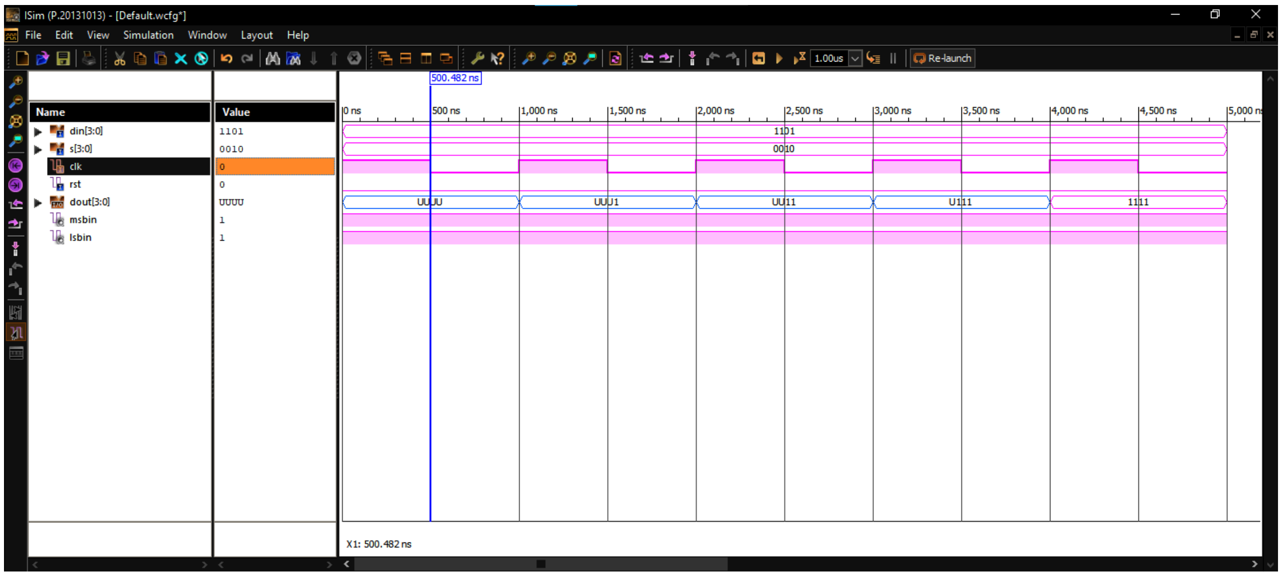
VHDL CODE:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity shiftrg is

   Port ( din : in  STD_LOGIC;

          clk : in  STD_LOGIC_VECTOR (3 downto 0);

          rst : in  STD_LOGIC;

          s : in  STD_LOGIC_VECTOR (1 downto 0);

          dout : inout  STD_LOGIC_VECTOR (3 downto 0));

end shiftrg;

architecture Behavioral of shiftrg 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

When "01"=>dout<= msbin & dout(3 downto 1);

When "10"=>dout<= lsbin & dout(2 downto 0);

When "00"=>dout<=din

When others =>dout<= "XXXX";

end case;

end if;

end Process;

end Behavioral;
```



Name	Value
din[3:0]	1101
s[3:0]	0001
clk	1
rst	0
dout[3:0]	1111
msbin	1
lsbin	1



Name	Value
din[3:0]	1101
s[3:0]	0010
clk	0
rst	0
dout[3:0]	UUUU
msbin	1
lsbin	1

CONCLUSION: