EXPERIMENT NO:1

Half Adder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity adder is
 Port ( a : in STD_LOGIC;
 b : in STD_LOGIC;
 s : out STD_LOGIC;
 c : out STD_LOGIC);
end adder;
architecture Behavioral of adder is
begin
s<=a xor b;
c<=a and b;
end Behavioral;
```

TESTBENCH

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;
ENTITY hadder IS
```

```
END hadder;
ARCHITECTURE behavior OF hadder IS
 -- Component Declaration for the Unit Under Test (UUT)
 COMPONENT adder
 PORT(
 a : IN std_logic;
 b : IN std_logic;
 s : OUT std_logic;
 c : OUT std_logic
 );
 END COMPONENT;

 --Inputs
 signal a : std_logic := '0';
 signal b : std_logic := '0';
--Outputs
 signal s : std_logic;
 signal c : std_logic;
 -- No clocks detected in port list. Replace <clock> below with
 -- appropriate port name
 --constant <clock>_period : time := 10 ns;
BEGIN
-- Instantiate the Unit Under Test (UUT)
 uut: adder PORT MAP (
 a => a,
 b => b,
 s => s,
 c => c
 );
 -- Clock process definitions
-- <clock>_process :process
-- begin
-- <clock> <= '0';
-- wait for <clock>_period/2;
-- <clock> <= '1';
-- wait for <clock>_period/2;
-- end process;
--
 -- Stimulus process
 stim_proc: process
 begin
a<='0';
b<='0';
 -- hold reset state for 100 ns.
 wait for 100 ns;
a<='0';
b<='1';
 -- hold reset state for 100 ns.
 wait for 100 ns;
a<='1';
b<='0';
 -- hold reset state for 100 ns.
 wait for 100 ns;
a<='1';
b<='1';
 -- hold reset state for 100 ns.
 wait for 100 ns;
 --wait for <clock>_period*10;
 -- insert stimulus here
 wait;
 end process;
END;
```

Full Adder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity fulladder is
 Port ( x : in STD_LOGIC;
 y : in STD_LOGIC;
 z : in STD_LOGIC;
 s : out STD_LOGIC;
 c : out STD_LOGIC);
end fulladder;
architecture Behavioral of fulladder is
begin
s<=(x xor y)xor z;
c<=((x xor y) and z) or (x and z);
end Behavioral;
```

testbench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;
ENTITY fadder IS
END fadder;
ARCHITECTURE behavior OF fadder IS
 -- Component Declaration for the Unit Under Test (UUT)
 COMPONENT fulladder
 PORT(
 x : IN std_logic;
 y : IN std_logic;
 z : IN std_logic;
 s : OUT std_logic;
 c : OUT std_logic
 );
 END COMPONENT;

 --Inputs
 signal x : std_logic := '0';
 signal y : std_logic := '0';
 signal z : std_logic := '0';
--Outputs
 signal s : std_logic;
 signal c : std_logic;
 -- No clocks detected in port list. Replace <clock> below with
 -- appropriate port name
 --constant <clock>_period : time := 10 ns;
BEGIN
-- Instantiate the Unit Under Test (UUT)
 uut: fulladder PORT MAP (
 x => x,
 y => y,
 z => z,
 s => s,
 c => c
 );
 -- Clock process definitions
-- <clock>_process :process
-- begin
-- <clock> <= '0';
-- wait for <clock>_period/2;
-- <clock> <= '1';
-- wait for <clock>_period/2;
-- end process;
--
 -- Stimulus process
 stim_proc: process
 begin
x<='0';
y<='0';
z<='0';
 -- hold reset state for 100 ns.
 wait for 100 ns;
```

```
x<='0';
y<='1';
z<='0';
 -- hold reset state for 100 ns.
 wait for 100 ns;
x<='0';
y<='0';
z<='1';
 -- hold reset state for 100 ns.
 wait for 100 ns;
x<='0';
y<='1';
z<='1';
 -- hold reset state for 100 ns.
 wait for 100 ns;
x<='1';
y<='0';
z<='0';
 -- hold reset state for 100 ns.
 wait for 100 ns;
x<='1';
y<='0';
z<='1';
 -- hold reset state for 100 ns.
 wait for 100 ns;
x<='1';
y<='1';
z<='0';
 -- hold reset state for 100 ns.
 wait for 100 ns;
x<='1';
y<='1';
z<='1';
 -- hold reset state for 100 ns.
 wait for 100 ns;
 --wait for <clock>_period*10;
 -- insert stimulus here
 wait;
 end process;
END;
```

EXPERIMENT NO 2

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity vlsi is

 Port ( a : in STD_LOGIC_VECTOR (3 downto 0);

 b : in STD_LOGIC_VECTOR (3 downto 0);

 sel : in STD_LOGIC_VECTOR (2 downto 0);

 y : out STD_LOGIC_VECTOR (3 downto 0));

end vlsi;

architecture Behavioral of vlsi is

begin

process(a,b,sel)

begin

case sel is

when"000"=>y<=a+b;

when"001"=>y<=a-b;

when"010"=>y<=a and b;

when"011"=>y<=a or b;
```

```
);
when"100"=>y<=a nand b;
when"101"=>y<=a nor b;
when"110"=>y<=not a;
when"111"=>y<=a;
when others=>null;
end case;
end process;
end Behavioral;
```

TESTBENCH

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;
ENTITY ssss IS
END ssss;
ARCHITECTURE behavior OF ssss IS
 -- Component Declaration for the Unit Under Test (UUT)
 COMPONENT vlsi
 PORT(
 a : IN std_logic_vector(3 downto 0);
 b : IN std_logic_vector(3 downto 0);
 sel : IN std_logic_vector(2 downto 0);
 y : OUT std_logic_vector(3 downto 0)
 );
 END COMPONENT;
 --Inputs
 signal a : std_logic_vector(3 downto 0) := (others => '0');
 signal b : std_logic_vector(3 downto 0) := (others => '0');
 signal sel : std_logic_vector(2 downto 0) := (others => '0');
--Outputs
 signal y : std_logic_vector(3 downto 0);
 -- No clocks detected in port list. Replace <clock> below with
 -- appropriate port name
-- constant <clock>_period : time := 10 ns;
BEGIN
-- Instantiate the Unit Under Test (UUT)
 uut: vlsi PORT MAP (
 a => a,
 b => b,
 sel => sel,
 y => y
 );
 -- Clock process definitions
-- <clock>_process :process
-- begin
-- <clock> <= '0';
-- wait for <clock>_period/2;
-- <clock> <= '1';
-- wait for <clock>_period/2;
-- end process;
```

```
--
 -- Stimulus process
 stim_proc: process
 begin
a<="0101";
b<="0100";
sel<="000";
-- hold reset state for 100 ns.
 wait for 100 ns;
a<="0101";
b<="0100";
sel<="001";
 -- hold reset state for 100 ns.
 wait for 100 ns;
a<="0101";
b<="0100";
sel<="010";
 -- hold reset state for 100 ns.
 wait for 100 ns;
a<="0101";
b<="0100";
sel<="011";
 -- hold reset state for 100 ns.
 wait for 100 ns;
a<="0101";
b<="0100";
```

```
sel<="100";
 -- hold reset state for 100 ns.
 wait for 100 ns;
a<="0101";
b<="0100";
sel<="101";
 -- hold reset state for 100 ns.
 wait for 100 ns;
a<="0101";
b<="0100";
sel<="110";
 -- hold reset state for 100 ns.
 wait for 100 ns;
a<="0101";
b<="0100";
sel<="111";
 -- hold reset state for 100 ns.
 wait for 100 ns;
-- wait for <clock>_period*10;
 -- insert stimulus here
 wait;
 end process;
END;
```

EXP 3 – SHIFT RESISTER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity shiftreg is
 Port ( si : in STD_LOGIC;
 clk : in STD_LOGIC;
 so : out STD_LOGIC;
 pin : in STD_LOGIC_VECTOR (3 downto 0);
 po : out STD_LOGIC_VECTOR (3 downto 0);
 sel : in STD_LOGIC_VECTOR (1 downto 0));
end shiftreg;
architecture Behavioral of shiftreg is
signal temp:STD_LOGIC_VECTOR( 3 downto 0);
begin
process(clk)
begin
if(clk'event and clk='1')then
case sel is
when"00"=> temp<= si&temp(3 downto 1);
```

```
so <=temp(3);
when"01"=> temp<= si&temp(3 downto 1);
po<=temp;
when others=>null;
end case;
end if;
end process;
end Behavioral;
```

TEST BENCH

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;
ENTITY shifttest IS
END shifttest;
ARCHITECTURE behavior OF shifttest IS
 -- Component Declaration for the Unit Under Test (UUT)
 COMPONENT shiftreg
 PORT(
 si : IN std_logic;
 clk : IN std_logic;
 so : OUT std_logic;
 pin : IN std_logic_vector(3 downto 0);
 po : OUT std_logic_vector(3 downto 0);
 sel : IN std_logic_vector(1 downto 0)
 );
 END COMPONENT;
 --Inputs
 signal si : std_logic := '0';
 signal clk : std_logic := '0';
 signal pin : std_logic_vector(3 downto 0) := (others => '0');
 signal sel : std_logic_vector(1 downto 0) := (others => '0');
--Outputs
 signal so : std_logic;
 signal po : std_logic_vector(3 downto 0);
 -- Clock period definitions
 constant clk_period : time := 10 ns;
BEGIN
-- Instantiate the Unit Under Test (UUT)
 uut: shiftreg PORT MAP (
 si => si,
 clk => clk,
 so => so,
 pin => pin,
 po => po,
 sel => sel
 );
 -- 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

 sel<="00";

 si<='1';

 -- hold reset state for 100 ns.

 wait for 100 ns;

sel<="01";

si<='1';

wait for 100 ns;

sel<="10";

pin<="1100";

wait for 100 ns;

sel<="11";

pin<="1100";

 --=wait for clk_period*10;

 -- insert stimulus here

 wait;

 end process;

END;
```

EXP 4 – MOD –N COUNTER

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity mod5ex is

 Port ( clk : in STD_LOGIC;

 clr : in STD_LOGIC;

 q : inout STD_LOGIC_VECTOR (2 downto 0));

end mod5ex;

architecture Behavioral of mod5ex is

signal count: std_logic_vector(2 downto 0);

begin

process(clk)

begin

if (clr='1') then count <= "000";

elsif (rising_edge (clk)) then

if (count="100")

then count <= "000";
```

```
else
count<=count+ 1;
end if;
end if;
end process;
q<=count;
end Behavioral;
```

TESTBENCH

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;
ENTITY mod5test IS
END mod5test;
ARCHITECTURE behavior OF mod5test IS
 -- Component Declaration for the Unit Under Test (UUT)
 COMPONENT mod5ex
 PORT(
 clk : IN std_logic;
 clr : IN std_logic;
 q : INOUT std_logic_vector(2 downto 0)
 );
 END COMPONENT;
 --Inputs
 signal clk : std_logic := '0';
 signal clr : std_logic := '0';

--BiDirs

 signal q : std_logic_vector(2 downto 0);

 -- Clock period definitions

 constant clk_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

 uut: mod5ex PORT MAP (

 clk => clk,

 clr => clr,

 q => q

 );

 -- Clock process definitions

 clk_process :process

 begin

clk <= '0';

wait for 10 ns;

clk <= '1';

wait for 10 ns;

 end process;

 -- Stimulus process

 stim_proc: process

 begin

clr<='1';

 -- hold reset state for 100 ns.

 wait for 20 ns;
```

```
clr<='0';

 -- hold reset state for 100 ns.

 wait for 20 ns;

 -- hold reset state for 100 ns.

-- wait for 100 ns;

-- wait for clk_period*10

 -- insert stimulus here

 wait;

 end process;
```

EXP 5 – FIFO

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.numeric_std.all;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity fifo_vhdl is
 Port ( clk,rst : in STD_LOGIC;
 enr : in STD_LOGIC; --enable read,should be '0' when not in use.
 enw : in STD_LOGIC; --enable write,should be '0' when not in use.
 dataout : out STD_LOGIC_VECTOR (3 downto 0); --output data
 datain : in STD_LOGIC_VECTOR (3 downto 0); --input data
 FF_empty,clk_div : out STD_LOGIC; --set as '1' when the queue is empty
 FF_full : out STD_LOGIC); ---set as '1' when the queue is full
end fifo_vhdl;
architecture Behavioral of fifo_vhdl is
type memory_type is array (0 to 7) of std_logic_vector(3 downto 0);
signal memory: memory_type :=(others=>(others=> '0'));
signal readptr,writeptr: std_logic_vector(2 downto 0) :="000";
signal count : std_logic_vector(2 downto 0):="000";

signal newclk : std_logic;

signal count1 : std_logic_vector(25 downto 0);

begin

---------------------------------------------------------------

clk_div1: process(clk,rst)

begin

if rst='1' then

count1<=(others =>'0');

elsif clk'event and clk='1' then

count1<=count1+'1';

end if;

end process;

---------------------------------------------------------------

newclk<=count1(25);

clk_div<=count1(25);

---------------------------------------------------------------

fifo_emty_full: process(readptr,writeptr,count)

begin

if(count="000") then

FF_empty<='1';

FF_full<='0';

elsif(count="111")then

FF_empty<='0';

FF_full<='1';

end if;
end process;

-----------------------------------------------------------------

count1_reptr_wdptr: process(newclk,rst,enr,enw,readptr,writeptr)

begin

if rst='1' then

count<="000";

readptr<=(others=>'0');

writeptr<=(others=>'0');

else if newclk'event and newclk='1' then

if enw='1' and enr='0' then

count<=count+'1';

if count="111" then

count<=count;

end if;

readptr<=readptr;

writeptr<=writeptr+1;

elsif enw='0' and enr='1' then

count<=count-'1';

if count="000" then

count<=count;

end if;

readptr<=readptr+1;

writeptr<=writeptr;

else

null;

end if;
end if;

end if;

end process;

-----------------------------------------------------------------

mem_read_write:process(newclk,count,enw,enr)

begin

if(newclk'event and newclk='1') then

if enw='1' and enr='0' then

if count /="111" then

memory(conv_integer(writeptr))<=datain;

end if;

elsif enw='0' and enr='1' then

if count /="000" then

dataout<=memory(conv_integer(readptr));

end if;

end if;

end if;

end process;

end Behavioral;
```

TEST BENCH

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--USE ieee.numeric_std.ALL;

ENTITY fifo_test IS
```

```
END fifo_test;

ARCHITECTURE behavior OF fifo_test IS

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

 COMPONENT fifo_vhdl

 PORT(

 clk : IN std_logic;

 rst : IN std_logic;

 enr : IN std_logic;

 enw : IN std_logic;

 dataout : OUT std_logic_vector(3 downto 0);

 datain : IN std_logic_vector(3 downto 0);

 FF_empty : OUT std_logic;

 clk_div : OUT std_logic;

 FF_full : OUT std_logic

 );

 END COMPONENT;


 --Inputs

 signal clk : std_logic := '0';

 signal rst : std_logic := '0';

 signal enr : std_logic := '0';

 signal enw : std_logic := '0';

 signal datain : std_logic_vector(3 downto 0) := (others => '0');

--Outputs

 signal dataout : std_logic_vector(3 downto 0);

 signal FF_empty : std_logic;
 signal clk_div : std_logic;
 signal FF_full : std_logic;
 -- Clock period definitions
 constant clk_period : time := 10 ns;
 constant clk_div_period : time := 10 ns;
BEGIN
-- Instantiate the Unit Under Test (UUT)
 uut: fifo_vhdl PORT MAP (
 clk => clk,
 rst => rst,
 enr => enr,
 enw => enw,
 dataout => dataout,
 datain => datain,
 FF_empty => FF_empty,
 clk_div => clk_div,
 FF_full => FF_full
 );
 -- Clock process definitions
 clk_process :process
 begin
clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;
 end process;
 clk_div_process :process
 begin
clk_div <= '0';
wait for clk_div_period/2;
clk_div <= '1';
wait for clk_div_period/2;
 end process;
-- Stimulus process
 stim_proc: process
 begin
 -- hold reset state for 100 ns.
rst<='1';
enr<='0';
enw<='1';
dataout<="0110";
wait for 100 ns;
rst<='0';
enr<='0';
enw<='1';
dataout<="1100";
wait for 100 ns;
rst<='0';
enr<='1';
enw<='0';
dataout<="1100";
wait for 100 ns;
```

```
 wait for clk_period*10;

 -- insert stimulus here

 wait;

 end process;

END;
```

EXP 6 – LCD

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity LCD_SIMPLE_Practice is

 Port ( rst,clk : in STD_LOGIC;

 lcd_RS : out STD_LOGIC;

 -- din : in std_logic_vector(7 downto 0);

 --LED : OUT std_logic_vector(7 downto 0);

 lcd_EN : out STD_LOGIC;

 data_out : out STD_LOGIC_VECTOR (7 downto 0));

 end LCD_SIMPLE_Practice;

architecture Behavioral of LCD_SIMPLE_Practice is

signal div_count : std_logic_vector(20 downto 0);

signal clk_new : std_logic;

-----------------------------------------------------------

type state
is(reset,fuction,mode,cur,clear,d0,d1,d2,fuction1,mode1,cur1,clear1,d01,d11,d21);

signal pss,nx : state;

begin

--LED<=din;

--------------------------------------------------------------

clk_DIV: process(clk,rst)

begin

if rst= '1' then

div_count<= (others=>'0');
```

```
elsif clk'event and clk='1' then

div_count<= div_count + '1';

end process;

------------------------------------------------------------

clk_new<= div_count(20);

--------------------------------------------------------------

p_state_transactin: process (clk_new,rst)

begin

if rst='1' then

pss<=reset;

elsif clk_new'event and clk_new= '1' then

pss<= nx;

end if;

end process;

----------------------------------------------------------------

LCD_working: process(pss)

begin

case pss is

when reset =>

lcd_RS<='0';

lcd_EN<='1';

nx<= fuction;

data_out<="00111100"; --3Ch

when fuction =>

lcd_RS<='0';

lcd_EN<='1';
```

```
data_out<="00111100"; --3Ch

nx<= fuction

when fuction1 =>

lcd_RS<='0';

lcd_EN<='0';

data_out<="00111100"; --3Ch

nx<= mode;

when mode =>

lcd_RS<='0';

lcd_EN<='1';

data_out<="00000110"; --06h

nx<= mode1;

when mode1 =>

lcd_RS<='0';

lcd_EN<='0';

data_out<="00000110"; --06h

nx<= cur;

when cur =>

lcd_RS<='0';

lcd_EN<='1';

data_out<="00001100"; --0Ch

nx<= cur1;

when cur1 =>

lcd_RS<='0';

lcd_EN<='0';

data_out<="00001100"; --0Ch
```

```
nx<= clear;

when clear =>

lcd_RS<='0';

lcd_EN<='1';

data_out<="00000001"; --01h

nx<= clear1;

when clear1 =>

lcd_RS<='0';

lcd_EN<='0';

data_out<="00000001"; --01h

nx<= d0;

when d0 =>

lcd_RS<='1';

lcd_EN<='1';

data_out<="01010011"; -----din; --S

nx<= d01;

when d01 =>

lcd_RS<='1';

lcd_EN<='0';

data_out<="01010011";-----din; --S

nx<= d1;

when d1 =>

lcd_RS<='1';

lcd_EN<='1';

data_out<="01001011"; --K

nx<= d11;
```

```
when d11 =>

lcd_RS<='1';

lcd_EN<='0';

data_out<="01001011"; --K

nx<= d2;

when d2 =>

lcd_RS<='1';

lcd_EN<='1';

data_out<="01001110"; --N

nx<= d21;

when d21 =>

lcd_RS<='1';

lcd_EN<='0';

data_out<="01001110"; --N

nx<= d21;

when others=>

null;

end case;

end process;

end Behavioral;
```

TEST BENCH

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--USE ieee.numeric_std.ALL;
```

```
ENTITY LCDTEST IS
END LCDTEST;
ARCHITECTURE behavior OF LCDTEST IS
 -- Component Declaration for the Unit Under Test (UUT)
 COMPONENT LCD_SIMPLE_Practice
 PORT(
 rst : IN std_logic;
 clk : IN std_logic;
 lcd_RS : OUT std_logic;
 lcd_EN : OUT std_logic;
 data_out : OUT std_logic_vector(7 downto 0)
 );
 END COMPONENT;
 --Inputs
 signal rst : std_logic := '0';
 signal clk : std_logic := '0';
--Outputs
 signal lcd_RS : std_logic;
 signal lcd_EN : std_logic;
 signal data_out : std_logic_vector(7 downto 0);
 -- Clock period definitions
 constant clk_period : time := 10 ns;
BEGIN
-- Instantiate the Unit Under Test (UUT)
 uut: LCD_SIMPLE_Practice PORT MAP (
 rst => rst,
 clk => clk,
 lcd_RS => lcd_RS,
 lcd_EN => lcd_EN,
 data_out => data_out
 );
 -- 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
rst<='1';
 -- hold reset state for 100 ns.
 wait for 100 ns;
rst<='0';
--din<="00101010";
 wait for 100 ns;
 rst<='1';
 --din<="00101010";
 -- hold reset state for 100 ns.
rst<='0';
```

```
--din<="00101010";
 wait for 100 ns;
rst<='1';
--din<="00111111";
--hold reset state for 100 ns.
 wait for 100 ns;
 rst<='0';
 wait for 100 ns;
 -- insert stimulus here
 wait;
 end process;
END;
```