**ALU**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity alu2 is

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

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

S : in STD\_LOGIC\_VECTOR (2 downto 0);

Y : out STD\_LOGIC\_VECTOR (3 downto 0));

end alu2;

architecture Behavioral of alu2 is

begin

process(A,B,S)

begin

IF S="000" THEN

Y<= A AND B;

ELSIF S="001" THEN

Y<= A NAND B;

ELSIF S="010" THEN

Y<= A OR B;

ELSIF S="011" THEN

Y<= A NOR B;

ELSIF S="100" THEN

Y<= A XOR B;

ELSIF S="101" THEN

Y<= A XNOR B;

ELSIF S="110" THEN

Y<= NOT(A);

ELSIF S="111" THEN

Y<= NOT(B);

END IF ;

END PROCESS;

end Behavioral;

**ALU\_TB**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

ENTITY alu2\_tb IS

END alu2\_tb;

ARCHITECTURE behavior OF alu2\_tb IS

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

COMPONENT alu2

PORT(

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

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

S : 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 S : std\_logic\_vector(2 downto 0) := (others => '0');

--Outputs

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

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: alu2 PORT MAP (

A => A,

B => B,

S => S,

Y => Y

);

-- Stimulus process

stim\_proc: process

begin

s<= s+1;

wait for 100 ns;

end process;

END;

**USR**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity usr is

Port ( clk : in STD\_LOGIC;

rst : in STD\_LOGIC;

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

sin : in STD\_LOGIC;

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

sout : out STD\_LOGIC;

pout : out STD\_LOGIC\_VECTOR (3 downto 0));

end usr;

architecture Behavioral of usr is

signal temp:std\_logic\_vector(3 downto 0):="0000";

begin

process(clk,rst)

begin

if rst='1' then

sout<='0';

pout<="0000";

elsif rising\_edge(clk) then

case mode is

when "00" =>

for i in 3 downto 0 loop

temp(i)<=sin;

sout<= temp(i);

end loop;

when "01" =>

for i in 3 downto 0 loop

temp(i)<=sin;

pout<= temp;

end loop;

when "10" =>

for i in 3 downto 0 loop

temp(i)<=pin(i);

sout<= temp(i);

end loop;

when others =>

pout<=pin;

end case;

end if;

end process;

end Behavioral;

**USR\_TB**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY usr\_tb IS

END usr\_tb;

ARCHITECTURE behavior OF usr\_tb IS

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

COMPONENT usr

PORT(

clk : IN std\_logic;

rst : IN std\_logic;

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

sin : IN std\_logic;

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

sout : OUT std\_logic;

pout : OUT std\_logic\_vector(3 downto 0)

);

END COMPONENT;

--Inputs

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

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

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

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

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

--Outputs

signal sout : std\_logic;

signal pout : 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 (

clk => clk,

rst => rst,

mode => mode,

sin => sin,

pin => pin,

sout => sout,

pout => pout

);

-- Clock process definitions

clk\_process :process

begin

clk <= '0';

wait for clk\_period/2;

clk <= '1';

wait for clk\_period/2;

end process;

rst\_process :process

begin

rst<='1';

wait for 10 ns;

rst<='0';

wait for 10 ns;

end process;

-- Stimulus process

stim\_proc\_mode: process

begin

mode<="00";

wait for 20 ns;

mode<="01";

wait for 20 ns;

mode<="10";

wait for 20 ns;

mode<="11";

wait for 20 ns;

end process;

stim\_proc\_sin: process

begin

wait for 20 ns;

sin<=not(sin);

wait for 20 ns;

end process;

stim\_proc\_pin: process

begin

pin<="1001";

wait ;

end process;

END;

/////////////////////////////////////////////////////////////////////

**FIFO**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity RAM is

Port ( CLK : in std\_logic;

RST : in std\_logic;

RD : in std\_logic;

WR : in std\_logic;

Data\_in : in std\_logic\_vector(7 downto 0);

Addr : in std\_logic\_vector(1 downto 0);

Data\_Out : out std\_logic\_vector(7 downto 0));

end RAM;

architecture Behavioral of RAM is

Type Mem is array (0 to 3) of std\_logic\_vector(7 downto 0);

signal Reg : Mem:=(OTHERS=>(OTHERS=>'0'));

begin

process(RST,CLK,RD,WR,Data\_in)

begin

if(RST = '1') then

Reg<=(OTHERS=>(OTHERS=>'0'));

elsif(rising\_edge(CLK))then

if(WR = '1' AND RD = '0')then

case Addr is

when "00" =>Reg(0) <= Data\_in;

when "01" =>Reg(1) <= Data\_in;

when "10" =>Reg(2) <= Data\_in;

when others =>Reg(3) <= Data\_in;

end case;

elsif(WR = '0' AND RD = '1')then

case Addr is

when "00" =>Data\_Out<= Reg(0);

when "01" =>Data\_Out<= Reg(1);

when "10" =>Data\_Out<= Reg(2);

when others =>Data\_Out<= Reg(3);

end case;

end if;

end if;

end process;

end Behavioral;

 FIFO\_TB

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.std\_logic\_unsigned.ALL;

ENTITY RAM\_8x8\_TB IS

END RAM\_8x8\_TB;

ARCHITECTURE behavior OF RAM\_8x8\_TB IS

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

COMPONENT RAM

PORT(

CLK : IN std\_logic;

RST : IN std\_logic;

RD : IN std\_logic;

WR : IN std\_logic;

Data\_in : IN std\_logic\_vector(7 downto 0);

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

Data\_Out : OUT std\_logic\_vector(7 downto 0)

);

END COMPONENT;

--Inputs

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

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

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

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

signal Data\_in :std\_logic\_vector(7 downto 0) := X"00";

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

--Outputs

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: RAM PORT MAP (

CLK => CLK,

RST => RST,

RD => RD,

WR => WR,

Data\_in =>Data\_in,

Addr =>Addr,

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

rst\_proc: process

begin

wait for 80 ns;

RST<='1';

wait for 20 ns;

RST<='0';

wait;

end process;

addr\_proc: process

begin

Addr<=Addr+1;

wait for 10 ns;

end process;

datain\_proc: process

begin

Data\_in<= X"41";

wait for 10 ns;

Data\_in<= X"42";

wait for 10 ns;

Data\_in<= X"43";

wait for 10 ns;

Data\_in<= X"44";

wait for 10 ns;

end process;

rd\_proc: process

begin

RD<='0';

wait for 40 ns;

RD<='1';

wait for 120 ns;

RD<='0';

wait;

end process;

wr\_proc: process

begin

WR<='1';

wait for 40 ns;

WR<='0';

wait;

end process;

END;

**LCD**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity LCD\_31 is

Port ( rst : in std\_logic; -- reset

clk\_12Mhz : in std\_logic; -- high freq. clock

lcd\_rs : out std\_logic; -- LCD RS control

lcd\_en : out std\_logic; -- LCD Enable

lcd\_data : out std\_logic\_vector(7 downto 0)); -- LCD Data port

end LCD\_31;

architecture Behavioral of LCD\_31 is

signal div :std\_logic\_vector(20 downto 0); --- delay timer 1

signal clk\_fsm,lcd\_rs\_s: std\_logic;

-- LCD controller FSM states

type state is (reset,func,mode,cur,clear,d0,d1,d2,d3,d4,d5,hold);

signal ps,nx : state;

signal dataout\_s :std\_logic\_vector(7 downto 0); --- internal data command multiplexer

begin

----- clk divider ---------------------------------

process(rst,clk\_12Mhz)

begin

if(rst = '1')then

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

elsif( clk\_12Mhz'event and clk\_12Mhz ='1')then

div <= div + 1;

end if;

end process;

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

clk\_fsm<= div(15);

----- Presetn state Register -----------------------

process(rst,clk\_fsm)

begin

if(rst = '1')then

ps <= reset;

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

ps <= nx;

end if;

end process;

----- state and output decoding process

process(ps)

begin

case(ps) is

when reset =>

nx <= func;

lcd\_rs\_s <= '0';

dataout\_s <= "00111000"; -- 38h

when func =>

nx <= mode;

lcd\_rs\_s <= '0';

dataout\_s <= "00111000"; -- 38h

when mode =>

nx <= cur;

lcd\_rs\_s <= '0';

dataout\_s <= "00000110"; -- 06h

when cur =>

nx <= clear;

lcd\_rs\_s <= '0';

dataout\_s <= "00001100"; -- 0Ch curser at starting point of line1

when clear=>

nx <= d0;

lcd\_rs\_s <= '0';

dataout\_s <= "00000001"; -- 01h

when d0 =>

lcd\_rs\_s <= '1';

dataout\_s <= "01010000"; -- p

nx <= d1;

when d1 =>

lcd\_rs\_s <= '1';

dataout\_s <= "01001111"; -- o

nx <= d2;

when d2 =>

lcd\_rs\_s <= '1';

dataout\_s <= "01001111"; -- o

nx <= d3;

when d3 =>

lcd\_rs\_s <= '1';

dataout\_s <= "01001010"; -- J

nx <= d4;

when d4 =>

lcd\_rs\_s <= '1';

dataout\_s <= "01000001"; -- A

nx <= d5;

when d5 =>

lcd\_rs\_s <= '1';

dataout\_s <= "00100000"; -- space

nx <= hold;

when hold =>

lcd\_rs\_s <= '0';

dataout\_s <= "00000000"; -- hold

nx <= hold;

when others=>

nx <= reset;

lcd\_rs\_s <= '0';

dataout\_s <= "00000001"; -- CLEAR

end case;

end process;

lcd\_en<= clk\_fsm;

lcd\_rs<= lcd\_rs\_s;

lcd\_data<= dataout\_s;

end Behavioral;

**LCD\_TB**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY LCD\_31\_TB IS

END LCD\_31\_TB;

ARCHITECTURE behavior OF LCD\_31\_TB IS

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

COMPONENT LCD\_31

PORT(

rst : IN std\_logic;

clk\_12Mhz : IN std\_logic;

lcd\_rs : OUT std\_logic;

lcd\_en : OUT std\_logic;

lcd\_data : OUT std\_logic\_vector(7 downto 0)

);

END COMPONENT;

--Inputs

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

signal clk\_12Mhz :std\_logic := '0';

--Outputs

signal lcd\_rs :std\_logic;

signal lcd\_en :std\_logic;

signal lcd\_data :std\_logic\_vector(7 downto 0);

-- Clock period definitions

constant clk\_12Mhz\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: LCD\_31 PORT MAP (

rst =>rst,

clk\_12Mhz => clk\_12Mhz,

lcd\_rs =>lcd\_rs,

lcd\_en =>lcd\_en,

lcd\_data =>lcd\_data

);

-- Clock process definitions

clk\_12Mhz\_process :process

begin

clk\_12Mhz <= '0';

wait for clk\_12Mhz\_period/2;

clk\_12Mhz <= '1';

wait for clk\_12Mhz\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

rst<= '1';

wait for 20 ns;

rst<= '0';

-- insert stimulus here

wait;

end process;

END;