ALU CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_arith.all;
use IEEE.STD_LOGIC_unsigned.all;
entity ALU is
    Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
           B : in  STD_LOGIC_VECTOR (3 downto 0);
           S : in  STD_LOGIC_VECTOR (3 downto 0);
           M  : in  STD_LOGIC;
           F : out  STD_LOGIC_VECTOR (3 downto 0));
end ALU;
architecture Behavioral of ALU is
begin
process(A,B,S,M)
  begin
  If (M='0') then
    case S is
  when"0000"=>F<=A+B;
  when"0001"=>F<=A-B;
  when"0010"=>F<=A+'1';
  when"0011"=>F<=A-'1';
             when others=>Null;
            end case;
else
    case S is
  when"0000" => F<=A AND B;
  when"0001" => F<=A OR B;
  when"0010" => F<=A NAND B;
  when"0011" => F<=NOT A;
  when others => Null;
  end case;
  end if;
  end process;
  end  Behavioral ;
```

TEST BENCH

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ALUTB IS
END ALUTB;
ARCHITECTURE behavior OF ALUTB IS
    COMPONENT ALU
    PORT(
         A : IN  std_logic_vector(3 downto 0);
         B : IN  std_logic_vector(3 downto 0);
         S : IN  std_logic_vector(3 downto 0);
         M : IN  std_logic;
         F : 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(3 downto 0) := (others => '0');
   signal M : std_logic := '0';
          --Outputs
   signal F : std_logic_vector(3 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name
```

```
BEGIN
 -- Instantiate the Unit Under Test (UUT)
   uut: ALU PORT MAP (
          A => A,
          B => B,
          S => S,
          M => M,
          F => F
        );
   stim_proc: process
   begin
      M <= '0' ; S <= "0000"; A <= "0001"; B <= "0010";
   Assert F <= "0011" report "error in code";
   wait for 100 ns;
   M <= '1' ; S <= "0011"; A <= "0011";
   Assert F <= "1100" report "error in code";
   wait for 100 ns;
   M <= '1' ; S <= "0000"; A <= "0001"; B <= "0010";
   Assert F <= "0011" report "error in code";
    wait for 100 ns;
      wait;
   end process;
END;
```

FIFO CODE

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;
entity STD_FIFO is
Generic (
constant DATA_WIDTH : positive := 8;
constant FIFO_DEPTH : positive := 8
);
```

```
Port(
CLK : in STD_LOGIC;
RST : in STD_LOGIC;
WriteEn : in STD_LOGIC;
Dataln : in STD_LOGIC_VECTOR (DATA_WIDTH - 1 downto 0);
ReadEn : in STD_LOGIC;
DataOut : out STD_LOGIC_VECTOR (DATA_WIDTH - 1 downto 0);
Empty : out STD_LOGIC;
Full : out STD_LOGIC);
end STD_FIFO;
architecture Behavioral of STD_FIFO is
begin
fifo_proc : process(CLK)
type FIFO_Memory is array (0 to FIFO_DEPTH -1) of
STD_LOGIC_VECTOR(DATA_WIDTH - 1 downto 0);
variable Memory : FIFO_Memory;
variable Head : natural range 0 to FIFO_DEPTH - 1;
variable Tail : natural range 0 to FIFO_DEPTH - 1;
variable Looped : boolean;
begin
if rising_edge(CLK) then
if RST = '1' then
Head := 0;
Tail := 0;
Looped := false;
Full <= '0';
Empty <= '1';
else
if (ReadEn = '1') then
if ((Looped = true) or (Head /= Tail)) then
DataOut <= Memory(Tail);
if(Tail = FIFO_DEPTH - 1) then
```

```
Tail := 0;
Looped := false;
else
Tail := Tail + 1;
end if;
end if;
end if;
if (WriteEn = '1') then
if ((Looped = false) or (Head /= Tail)) then
Memory(Head) := Dataln;
if (Head = FIFO_DEPTH - 1) then
Head := 0;
Looped := true;
else
Head := Head + 1;
end if;
end if;
end if;
if (Head = Tail) then
if Looped then
Full <= '1';
else
Empty <= '1';
end if;
else
Empty <= '0';
Full <= '0';
end if;
end if;
end if;
end process;
end Behavioral;
```

TEST BENCH

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;
entity tb_STD_FIFO is
end tb_STD_FIFO;
architecture Behavioral of tb_STD_FIFO is
    -- Constants for the test bench
    constant DATA_WIDTH : positive := 8;
    constant FIFO_DEPTH : positive := 8;
    -- Signals for the DUT
    signal CLK : STD_LOGIC := '0';
    signal RST : STD_LOGIC := '0';
    signal WriteEn : STD_LOGIC := '0';
    signal Dataln : STD_LOGIC_VECTOR (DATA_WIDTH - 1 downto 0) := (others => '0');
    signal ReadEn : STD_LOGIC := '0';
    signal DataOut : STD_LOGIC_VECTOR (DATA_WIDTH - 1 downto 0);
    signal Empty : STD_LOGIC;
    signal Full : STD_LOGIC;
    -- Instantiate the DUT
    component STD_FIFO
    Generic (
        DATA_WIDTH : positive;
        FIFO_DEPTH : positive
    );
    Port (
        CLK : in STD_LOGIC;
        RST : in STD_LOGIC;
        WriteEn : in STD_LOGIC;
        Dataln : in STD_LOGIC_VECTOR (DATA_WIDTH - 1 downto 0);
        ReadEn : in STD_LOGIC;
        DataOut : out STD_LOGIC_VECTOR (DATA_WIDTH - 1 downto 0);
        Empty : out STD_LOGIC;
        Full : out STD_LOGIC
    );
    end component;
begin
    -- Instantiate the FIFO
    DUT: STD_FIFO
        Generic map (
            DATA_WIDTH => DATA_WIDTH,
            FIFO_DEPTH => FIFO_DEPTH
        )
        Port map (
            CLK => CLK,
            RST => RST,
            WriteEn => WriteEn,
            Dataln => Dataln,
            ReadEn => ReadEn,
            DataOut => DataOut,
            Empty => Empty,
            Full => Full
        );
    -- Clock generation
    CLK_process: process
    begin
        while true loop
            CLK <= '0';
            wait for 10 ns;
            CLK <= '1';
            wait for 10 ns;
        end loop;
    end process;
```

```
-- Test process
stim_proc: process
begin
    -- Reset the FIFO
    RST <= '1';
    wait for 20 ns;
    RST <= '0';
    wait for 20 ns;

    -- Write data to FIFO
    for i in 0 to 7 loop
        WriteEn <= '1';
        Dataln <= std_logic_vector(to_unsigned(i, DATA_WIDTH));
        wait for 20 ns;
    end loop;
    WriteEn <= '0'; -- Stop writing
    wait for 20 ns;
    -- Read data from FIFO
    for i in 0 to 7 loop
        ReadEn <= '1';
        wait for 20 ns; -- Allow time for the read operation
        ReadEn <= '0';
        wait for 20 ns;
    end loop;
    -- Attempt to read from an empty FIFO
    ReadEn <= '1';
    wait for 20 ns; -- Allow time for the read operation
    ReadEn <= '0';
    wait for 20 ns;
    -- Write more data to check full condition
    for i in 8 to 15 loop
        WriteEn <= '1';
```

```
            Dataln <= std_logic_vector(to_unsigned(i, DATA_WIDTH));
            wait for 20 ns;
        end loop;
        WriteEn <= '0'; -- Stop writing
        wait for 20 ns;
        -- Read all data to check FIFO behavior
        for i in 0 to 7 loop
            ReadEn <= '1';
            wait for 20 ns; -- Allow time for the read operation
            ReadEn <= '0';
            wait for 20 ns;
        end loop;
        -- Reset the FIFO again to see if it clears correctly
        RST <= '1';
        wait for 20 ns;
        RST <= '0';
        wait for 20 ns;

        -- Finalize simulation
        wait;
    end process;

end Behavioral;
```

HALF ADDER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity HA is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           s : out  STD_LOGIC;  -- Sum output
           c : out  STD_LOGIC); -- Carry output
```

```
end HA;

architecture structural of HA is
    component XOR2 is
        port ( x, y: in std_logic;
               z: out std_logic);
    end component;

    component AND2 is
        port ( p, q: in std_logic;
               r: out std_logic);
    end component;
begin
    HA1: XOR2 port map(A, B, s);  -- Connect XOR for sum
    HA2: AND2 port map(A, B, c);   -- Connect AND for carry
end structural;
```

XOR

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity XOR2 is
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           z : out STD_LOGIC);
end XOR2;

architecture behavior of XOR2 is
begin
    z <= x XOR y;  -- XOR logic
end behavior;
```

AND

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity AND2 is
    Port ( p : in STD_LOGIC;
           q : in STD_LOGIC;
           r : out STD_LOGIC);
end AND2;
architecture behavior of AND2 is
begin
    r <= p AND q;  -- AND logic
end behavior;
```

TEST BENCH

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY HFTB1 IS
END HFTB1;
ARCHITECTURE behavior OF HFTB1 IS
 Declaration for the Unit Under Test (UUT)
    COMPONENT HA
    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;
BEGIN
   uut: HA PORT MAP (
          A => A,
          B => B,
          s => s,
          c => c
        );

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
                        A<='0';
                        B<='1';
      wait for 100 ns;
      Assert s<='1' report"error in logic";
                        assert c<= '0' report "error in logic";
                        wait for 100 ns;
                        A<='1';
                        B<='1';
                        assert s<='0'; report "error in logic";
                        assert c<='1'; report "error in logic";
             wait for 100 ns;
   end process;

END;
```

MOD N CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
```

```
entity mod6 is
Generic (N:integer:=6);
Port ( clk : in STD_LOGIC;
 rst : in STD_LOGIC;
 Q : out integer range 0 to N-1);
end mod6;
architecture Behavioral of mod6 is
signal temp: integer range 0 to N-1;
begin
process (clk,rst)
begin
if clk'event and clk='1' then
if (rst='1' OR temp>=N-1) then
temp <=0;
else
temp<=temp+1;
end if;
end if;
Q <=temp;
end process;
end Behavioral;
```

TEST BENCH

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY modntb IS
END modntb;

ARCHITECTURE behavior OF modntb IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT mod6
    PORT(
         clk : IN  std_logic;
         rst : IN  std_logic;
         Q : OUT  integer range 0 to 5
        );
    END COMPONENT;
   --Inputs
   signal clk : std_logic := '0';
   signal rst : std_logic := '0';

         --Outputs
   signal Q : integer range 0 to 5;

   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN

         -- Instantiate the Unit Under Test (UUT)
   uut: mod6 PORT MAP (
          clk => clk,
          rst => rst,
          Q => Q
        );

   -- 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

         wait for 100 ns;
         rst<='1';
         wait for 100 ns;
         rst<='0';
         wait for clk_period*10;
         wait;
         end process;

END;
```

MUX 8-1 CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity MUX is
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
           S : in  STD_LOGIC_VECTOR (2 downto 0);
           Y : out  STD_LOGIC);
end MUX;

architecture Behavioral of MUX is

begin
process(A,S)
begin
```

```
  case S is
      when "000"=> Y <=A(0);
                        when "001"=> Y <=A(1);
                        when "010"=> Y <=A(2);
                        when "011"=> Y <=A(3);
                        when "100"=> Y <=A(4);
                        when "101"=> Y <=A(5);
                        when "110"=> Y <=A(6);
                        when others => Y <=A(7);
                        end case;
                        end process;
end Behavioral;
```

TEST BENCH

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY MUXTB IS
END MUXTB;

ARCHITECTURE behavior OF MUXTB IS

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

    COMPONENT MUX
    PORT(
         A : IN  std_logic_vector(7 downto 0);
         S : IN  std_logic_vector(2 downto 0);
         Y : OUT  std_logic
        );
    END COMPONENT;

   --Inputs
   signal A : std_logic_vector(7 downto 0) := (others => '0');
   signal S : std_logic_vector(2 downto 0) := (others => '0');
--Outputs
   signal Y : 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: MUX PORT MAP (
          A => A,
          S => S,
          Y => Y
        );
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
		A(1)<='0'; S<="000";
      wait for 100 ns;
		assert Y='0'; report "error in logic";
      wait for 100 ns;
      -- insert stimulus here
		A(2)<='1'; S<="010";
		wait for 100 ns ;
		assert Y='1'; report "error in logic";
   end process;
END;
```

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;
```