4 bit ALU

```
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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;


entity alu4bit is

   Port ( n1,n2 : in STD_LOGIC_VECTOR (3 downto 0);

         op : in STD_LOGIC_VECTOR (2 downto 0);

         res : out STD_LOGIC_VECTOR (3 downto 0);

         flag : out STD_LOGIC);

end alu4bit;


architecture Behavioral of alu4bit is


begin

process(n1,n2,op)

variable temp: std_logic_vector(4 downto 0);

begin

flag<='0';

case op is

when "000"=>

temp:= ('0' & n1) + n2; --when op =000 then perform addition

res<= temp(3 downto 0);
```

```
flag<= temp(4);
when "001"=>  ---Subtraction
if(n1> n2)then
res<= n1- n2;
else
res<=n2- n1;
end if;
when "010"=>
res<= n1 and n2;
when "011"=>
res<= n1 nand n2;
when "100"=>
res<= n1 xor n2;
when "101"=>
res<= n1 nor n2;
when "110"=>
res<=  not n1;
when others =>
res<= n2;
end case;
end process;
end Behavioral;
```

4 bit ALU Testbench

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
entity alu_tb is
--  Port ( );
end alu_tb;
architecture Behavioral of alu_tb is
```

```
component alu4bit
   Port ( n1,n2 : in STD_LOGIC_VECTOR (3 downto 0);
          op : in STD_LOGIC_VECTOR (2 downto 0);
          res : out STD_LOGIC_VECTOR (3 downto 0);
          flag : out STD_LOGIC);
end component;
signal n1,n2:std_logic_vector(3 downto 0);
signal op:std_logic_vector(2 downto 0);
signal res:std_logic_vector(3 downto 0);
signal flag: std_logic;
begin
uut: alu4bit port map( n1=>n1, n2=>n2, op=>op, res=>res, flag=>flag);
process
begin
n1<= "0101";
n2<="1011";
op<="000";
wait for 50 ns;
op<="001";
wait for 50 ns;
op<="010";
wait for 50 ns;
op<="011";
wait for 50 ns;
op<="100";
wait for 50 ns;
op<="101";
wait for 50 ns;
op<="110";
wait for 50 ns;
op<="111";
```

```
wait for 50 ns;
wait;
end process;
end Behavioral;
```

FIFO

```
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 leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity FIFO is
  Port ( clk,rst, wren,rden : in STD_LOGIC;
        empty : out STD_LOGIC;
        full : out STD_LOGIC;
        dataout:out std_logic_vector(7 downto 0));

end FIFO;

architecture Behavioral of FIFO is
signal clkout:std_logic;
signal counter: integer range 0 to 49999999;
signal rdptr,wrptr:integer range 0 to 7;
```

```
type fifo1 is array (0 to 7) of std_logic_vector(7 downto 0);

signal memory:fifo1;

type datain1 is array (0 to 7) of std_logic_vector(7 downto 0);

signal
data:datain1:=("10101010","10111011","10010101","11110000","00001111","10100101","1111010
1","01100110");

begin


process(clk,rst)   --clock frequency division

begin

if(rst='1')then

clkout<='0';

counter<=0;

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

if(counter=49999999)then

clkout<=not clkout;

counter<=0;

else

counter<= counter + 1;

end if;

end if;

end process;


process(clk)   ----Actual FIFO process

begin

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

if(rst='1')then

for i in 0 to 7 loop

memory(i)<="00000000";

end loop;

empty<='1';

full<='0';
```

```
rdptr<=0;

wrptr<=0;


elsif(wren='1')then       ---Write in FIFO

rdptr<=0;

memory(wrptr)<=data(wrptr);

empty<='0';

if(wrptr=7)then

wrptr<=wrptr;

full<='1';

else

wrptr<=wrptr + 1;

end if;


elsif(rden='1')then    ---Read From FIFO

wrptr<=0;

dataout<=memory(rdptr);

full<='0';

if(rdptr=7)then

rdptr <= rdptr;

empty<='1';

else

rdptr<=rdptr + 1;

end if;


else

dataout<= "ZZZZZZZZ";

end if;

end if;

end process;

end Behavioral;
```

FIFO Testbench

```
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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;


entity FIFO_TB is

--  Port ( );

end FIFO_TB;


architecture Behavioral of FIFO_TB is

component FIFO

   Port ( clk,rst, wren,rden : in STD_LOGIC;

         empty : out STD_LOGIC;

         full : out STD_LOGIC;

         dataout:out std_logic_vector(7 downto 0));


end component;

signal clk,rst,wren,rden:std_logic;

signal empty:std_logic;

signal full:std_logic;

signal dataout:std_logic_vector(7 downto 0);

constant clk_period: time:= 10 ns;

begin
```

```
uut: fifo port map (
    clk=>clk,
    rst=>rst,
    wren=>wren,
    rden=>rden,
    empty=>empty,
    full=>full,
    dataout=>dataout);

process
   begin
   clk<='0';
   wait for clk_period/2;
   clk<='1';
    wait for clk_period/2;
   end process;

process
begin
rst<='1';
wait for 90 ns;
rst<='0';
wait for 10 ns;
wren<='1';
wait for 80 ns;
wren<='0';
wait for 10 ns;
rden<='1';
wait for 90 ns;
rden<='0';
wait for 10 ns;
```

```
wait;
end process;
end Behavioral;
```

Shift register

```
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 leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity shiftreg is
   Port ( clk,rst,load,si : in STD_LOGIC;
          pi : in STD_LOGIC_VECTOR (3 downto 0);
          po : out STD_LOGIC_VECTOR (3 downto 0);
          mode:in STD_LOGIC_VECTOR (1 downto 0);
          so : out STD_LOGIC);
end shiftreg;

architecture Behavioral of shiftreg is
signal clkout:std_logic;
signal temp:std_logic_vector(3 downto 0);
signal counter: integer range 0 to 49999999;                Counter = Fin/Fout = 100 MHz/ 1Hz = 100 X
10 ^ 6

begin

process(clk,rst)   -----------for clock frequency division      Clkout High for 50 X 10^ 6 Cycles

                                                                          Clkout low for 50 X 10^ 6 Cycles

begin

if(rst='1')then

clkout<='0';

counter<=0;

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

if(counter=49999999)then

clkout<=not clkout;

counter<=0;

else

counter<= counter + 1;

end if;

end if;

end process;


process(clk,rst) -----------Actual shift register operation

begin

if(rst='1') then

so<='0';

po<="0000";

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

case mode is

when "00" =>

po<=pi;     -------PIPO

so<='0';


when "01"=>  -------SISO
```

```
temp(0)<=si;

temp(1)<=temp(0);

temp(2)<=temp(1);

temp(3)<=temp(2);

so<=temp(3);


po<="0000";


when "10"=>

if (load='1')then  ------PISO

temp<=pi;

else

temp(2)<=temp(3);

temp(1)<=temp(2);

 temp(0)<=temp(1);

 so<=temp(0);

 end if;


 when others=>

 if (load='0')then   -----SIPO

 temp(0)<=si;

 temp(1)<=temp(0);

 temp(2)<=temp(1);

 temp(3)<=temp(2);

 else

  po<=temp;

 end if;

 end case;

 end if;

  end process;

    end Behavioral;
```

Shift register Testbench

```
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 leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity exp2_98_tb is
--  Port ( );
end exp2_98_tb;

architecture Behavioral of exp2_98_tb is
component shiftreg
Port ( clk,rst,load,si : in STD_LOGIC;
        pi : in STD_LOGIC_VECTOR (3 downto 0);
        po : out STD_LOGIC_VECTOR (3 downto 0);
        mode:in STD_LOGIC_VECTOR (1 downto 0);
        so : out STD_LOGIC);
end component shiftreg;

signal clk,rst,load,si : STD_LOGIC;
signal  pi :STD_LOGIC_VECTOR (3 downto 0);
signal po :STD_LOGIC_VECTOR (3 downto 0);
signal mode:STD_LOGIC_VECTOR (1 downto 0);
signal so :STD_LOGIC;

constant clk_period: time:= 10 ns;
begin
uut: shiftreg port map (
     pi=>pi,
     si=>si,
     clk=>clk,
     rst=>rst,
     mode=>mode,
     load=>load,
     po=>po,
     so=>so);

 process
    begin
    clk<='0';
    wait for clk_period/2;
    clk<='1';
     wait for clk_period/2;
    end process;

    process
    begin
    rst<='1';
    wait for 10 ns;
    rst<='0';
```

```
mode<="00";
pi<="1001";
wait for 10 ns;

mode<="01";
si<='1';
wait for 10 ns;
si<='0';
wait for 10 ns;
si<='1';
wait for 10 ns;
si<='0';
wait for 50 ns;

mode<="10";
load<='1';
pi<="1010";
wait for 20 ns;
load<='0';
wait for 50 ns;

mode<="11";
load<='0';
wait for 10 ns;
si<='1';
wait for 10 ns;
si<='0';
wait for 10 ns;
```

```
    si<='1';

    wait for 10 ns;

    si<='0';

    wait for 10 ns;

    load<='1';

    wait for 20 ns;

    end process;

end Behavioral;
```

PmodKYPD.vhd

```
----------------------------------------------------------------------------------

-- Company: Digilent Inc 2011

-- Engineer: Michelle Yu

-- Create Date: 17:05:39 08/23/2011

--

-- Module Name:           PmodKYPD - Behavioral

-- Project Name: PmodKYPD

-- Target Devices:
Nexys3   -- Tool
versions: Xilinx ISE
13.2  -- Description:

-- This file defines a project that outputs the key pressed on the PmodKYPD to the seven segment
display

--
-- Revision:

-- Revision 0.01 - File Created

---------------------------------------------------------------------------
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity PmodKYPD is

Port (

clk : in STD_LOGIC;

JA : inout STD_LOGIC_VECTOR (7 downto 0); -- PmodKYPD is designed to be connected to JA  an : out
STD_LOGIC_VECTOR (3 downto 0); -- Controls which position of the seven segment          display to
display

seg : out STD_LOGIC_VECTOR (6 downto 0)); -- digit to display on the seven segment display  end
PmodKYPD;


architecture Behavioral of PmodKYPD is


component Decoder is

Port (

clk : in  STD_LOGIC;

Row : in STD_LOGIC_VECTOR (3 downto 0);

Col : out STD_LOGIC_VECTOR (3 downto 0);

DecodeOut : out STD_LOGIC_VECTOR (3 downto 0));
end component;


component DisplayController is

Port (
DispVal : in        STD_LOGIC_VECTOR (3 downto 0);
anode: out std_logic_vector(3 downto 0);

segOut : out STD_LOGIC_VECTOR (6 downto 0));
end component;


signal Decode: STD_LOGIC_VECTOR (3 downto 0);
begin



C0: Decoder port map (clk=>clk, Row =>JA(7 downto 4), Col=>JA(3 downto 0), DecodeOut=>
Decode);

C1: DisplayController port map (DispVal=>Decode, anode=>an, segOut=>seg );
```

Decoder.vhd

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

entity Decoder is
Port (
clk : in  STD_LOGIC;
Row : in STD_LOGIC_VECTOR (3 downto 0);
Col : out STD_LOGIC_VECTOR (3 downto 0);
DecodeOut : out STD_LOGIC_VECTOR (3 downto 0));  end Decoder;

architecture Behavioral of Decoder is

signal sclk :STD_LOGIC_VECTOR(19 downto 0); begin        process(clk)  begin
if clk'event and clk = '1' then
-- 1ms
if sclk = "00011000011010100000" then
--C1 Col<= "0111";  sclk <= sclk+1;  -- check row pins
elsif sclk = "00011000011010101000" then
--R1
if Row = "0111" then  DecodeOut <= "0001"; --1
--R2
elsif Row = "1011" then  DecodeOut <= "0100"; --4
--R3
elsif Row = "1101" then  DecodeOut <= "0111"; --7
--R4
elsif Row = "1110" then  DecodeOut <= "0000"; --0  end if;            sclk
<= sclk+1;
-- 2ms
elsif sclk = "00110000110101000000" then
--C2 Col<= "1011";  sclk <= sclk+1;  -- check row pins
elsif sclk = "00110000110101001000" then
```

```
--R1

if Row = "0111" then

DecodeOut <= "0010"; --2

--R2        elsif Row = "1011" then  DecodeOut <= "0101"; --5  --R3 elsif
Row = "1101" then DecodeOut <= "1000"; --8

--R4 elsif Row = "1110" then DecodeOut <= "1111"; --F end if; sclk
<= sclk+1; --3ms

elsif sclk = "01001001001111100000" then

--C3 Col<= "1101"; sclk <= sclk+1; -- check row pins

elsif sclk = "01001001001111101000" then

--R1 if Row = "0111" then DecodeOut <= "0011"; --3

--R2 elsif Row = "1011" then DecodeOut <= "0110"; --6

--R3 elsif Row = "1101" then DecodeOut <= "1001"; --9

--R4 elsif Row = "1110" then DecodeOut <= "1110"; --E end if; sclk
<= sclk+1; --4ms

elsif sclk = "01100001101010000000" then

--C4 Col<= "1110"; sclk <= sclk+1; -- check row pins

elsif sclk = "01100001101010001000" then

--R1 if Row = "0111" then DecodeOut <= "1010"; --A

--R2 elsif Row = "1011" then DecodeOut <= "1011"; --B

--R3 elsif Row = "1101" then DecodeOut <= "1100"; --C

--R4 elsif Row = "1110" then DecodeOut <= "1101"; --D end if;

sclk <= "00000000000000000000";

else sclk <= sclk+1; end if; end if; end process; end
Behavioral;
```

DisplayController.vhd

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


entity DisplayController is
```

```
Port (

--output from the Decoder

DispVal : in STD_LOGIC_VECTOR (3 downto 0);

--controls the display digits

anode: out std_logic_vector(3 downto 0);  --controls
which digit to display      segOut : out
STD_LOGIC_VECTOR (6 downto 0));  end
DisplayController;


architecture Behavioral of DisplayController is begin


-- only display the leftmost digit
anode<="1110";



with DispVal select segOut <=
"1000000" when "0000", --0

"1111001" when "0001", --1

"0100100" when "0010", --2

"0110000" when "0011", --3

"0011001" when "0100", --4

"0010010" when "0101", --5

"0000010" when "0110", --6

"1111000" when "0111", --7

"0000000" when "1000", --8

"0010000" when "1001", --9

"0001000" when "1010", --A

"0000011" when "1011", --B

"1000110" when "1100", --C

"0100001" when "1101", --D

"0000110" when "1110", --E

"0001110" when "1111", --F

"0111111" when others;
```

```
end Behavioral;
```

MOD N counter

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity ModnCounter is
 GENERIC (N: integer := 4);
 port (
Q:out std_logic_vector(N-1 downto 0);

 enable :in std_logic;

 clk : in std_logic;

 reset : in std_logic);

 end ModnCounter;

architecture countern_arch of ModnCounter is
 signal count:std_logic_vector (N-1 downto 0);
 signal clkout : std_logic ;
signal ctr : integer range 0 to 49999999 := 0;
 begin
 process(clk,reset)
 begin
if(reset = '1') then
```

```
clkout <= '0' ;
 ctr <= 0 ;
elsif(clk'event and clk = '1') then
 if(ctr = 49999999) then
 clkout <= not clkout ;
 ctr <= 0 ;
 else
ctr <= ctr + 1 ;

end if ;
end if ;
 end process ;
 process(clkout, reset)
 begin
 if (reset='1') then
 count <= (others =>'0');
 elsif (clkout = '1' and clkout'EVENT) then
 if (enable= '1') then
 count <= count + '1';
 end if;
 end if;
 Q <= count;
 end process;
end countern_arch;
```

MOD N counter testbench

```
library IEEE;


 use IEEE.STD_LOGIC_1164.ALL;
```

```
entity test_bench is
--  Port ( );
end test_bench;
architecture tb of test_bench is

 component ModnCounter port(

    Q : out std_logic_vector(3 downto 0);

enable :in std_logic;

     clk : in std_logic;

     reset : in std_logic);

     end component;

 signal clk : std_logic := '0';

 signal reset : std_logic := '0';

 signal enable : std_logic := '0';

 signal Q : std_logic_vector(3 downto 0);

constant clock_period : time:= 20 ns;

 Begin

uut: ModnCounter Port Map (
clk => clk ,
 reset => reset
, enable => enable,
 Q => Q);
 Process
 begin
clk <= '1';
 wait for clock_period/2;
 clk <= '0';
 wait for clock_period/2;
END PROCESS;
PROCESS  BEGIN
reset <= '1';
wait for 10ns;
reset <= not reset;
enable <= '1';
 wait for 100ns;
wait;
END PROCESS;
END;
```