VHDL Code Archive 2018

VHDL NIRVANA
Computer Architecture Lab : CS 493
15 / May / 2018

```
library ieee;
use ieee.std_logic_1164.all;
entity XORgate is
port
(
      A, B: in std_logic;
      S: out std_logic
);
end XORgate;

architecture behaviour of XORgate is

begin
      S <= A xor B;

end behaviour;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity upCounter is port ( CLR, CLK : in std_logic;
    Q : out std_logic_vector(2 downto 0));
end upCounter;

architecture behavior of upCounter is
begin
      process(CLR, CLK)
            variable tmp : std_logic_vector(2 downto 0) := "000";
      begin
            if CLK'event and CLK = '1' then
                  if tmp = "111" or CLR = '1' then
             tmp := "000";
                  else
             tmp := tmp + '1';
                  end if;
                  Q <= tmp;
            end if;
      end process;
end behavior;
```

Compiled by Oishik Mukhopadhyay.

```
library ieee;
use ieee.std_logic_1164.all;
entity Two_Bit_Comparator is
port  (
      A1, B1, A0, B0 : in bit;
      Agreater, Bgreater, ABequal : out bit
);
end Two_Bit_Comparator;
architecture behave of Two_Bit_Comparator is
begin
      process(A1, A0, B1, B0)
      begin
            Agreater <= '0';
            Bgreater <= '0';
            ABequal <= '0';
            if(A1 > B1) then
                  Agreater <= '1';
            elsif(A1 < B1) then
                  Bgreater <= '1';
            else
                  if(A0 > B0) then
                        Agreater <= '1';
                  elsif(A0 < B0) then
                        Bgreater <= '1';
                  else
                        ABequal <= '1';
                  end if;
            end if;
      end process;
end behave;


library ieee;
use ieee.std_logic_1164.all;

entity sRFlipFlop is
port(
      S           :     in std_logic;
      R           :     in std_logic;
      CLK   :     in std_logic;
      Q           :     out std_logic;
      Qb    :     out std_logic
);
end sRFlipFlop;
architecture behavior of sRFlipFlop is
begin
      process(S, R, CLK)
            variable temp : std_logic := '0';
      begin
            if(CLK'event and CLK='1')     then  -- if(rising_edge(CLK))
                  if(S='0' and R='0')     then
                        temp := temp;
                  elsif(S='0' and R='1')  then
                        temp := '0';
                  elsif(S='1' and R='0')  then
                        temp := '1';
                  else
```

```
                        temp := 'Z';
                  end if;
            end if;
            Q <= temp;
            Qb <= not temp;
      end process;
end behavior;




library ieee;
use ieee.std_logic_1164.all;

entity sisoreg is
      port(
            I, CLK, CLR : in bit;
            Q : out bit
      );
end sisoreg;

architecture behavior of sisoreg is
begin
      process (I, CLK)
            variable temp : bit_vector(3 downto 0) := "0000";
      begin
            if(CLR='1') then
                  temp := "0000";
            elsif(CLK'event and CLK='1') then
                  Q <= temp(0);
                  temp(2 downto 0) := temp(3 downto 1);
                  temp(3) := I;
            end if;
      end process;
end behavior;




library ieee;
use ieee.std_logic_1164.all;

entity siporeg is
      port(
            I, CLK, CLR : in bit;
            M : out bit;
            Q : out bit_vector(3 downto 0)
      );
end siporeg;

architecture behavior of siporeg is
begin
      process (I, CLK)
            variable temp : bit_vector(3 downto 0) := "0000";
      begin
            if(CLR='1') then
                  temp := "0000";
            elsif(CLK'event and CLK='1') then
```

```
                  temp(3 downto 1) := temp(2 downto 0);
                  temp(0) := I;
            end if;
            M <= I;
            Q <= temp;
      end process;
end behavior;



library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity ringCounter is port ( CLR, CLK : in std_logic;
    Q : out std_logic_vector(0 to 3));
end ringCounter;

architecture behavior of ringCounter is
begin
      process(CLR, CLK)
            variable tmp : std_logic_vector(0 to 3) := "1000";
            variable index : integer := 1;
      begin
            if CLK'event and CLK = '1' then
                  if CLR = '1' or index = 4 then
                        index := 1;
                        tmp := "1000";
                  else
                        tmp(index) := '1';
                        tmp(index - 1) := '0';
                        index := index + 1;
                  end if;
                  Q <= tmp;
            end if;
      end process;
end behavior;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity ringCounter is port ( CLR, CLK : in std_logic;
    Q : out std_logic_vector(0 to 3));
end ringCounter;

architecture behavior of ringCounter is
      component dFlipFlop port(D, DCLR, DCLK : in std_logic;
            Q, Qb : out std_logic);
      end component;
      signal S1, S2, S3, S4, S1b, S2b, S3b, S4b, INIT, ZERO : std_logic;
begin
      ZERO <= '0';
      INIT <= S4 or CLR;
      MAP0 : dFlipFlop port map(INIT, ZERO, CLK, S1, S1b);
      Q(0) <= S1;
      MAP1 : dFlipFlop port map(S1, CLR, CLK, S2, S2b);
      Q(1) <= S2;
      MAP2 : dFlipFlop port map(S2, CLR, CLK, S3, S3b);
      Q(2) <= S3;
      MAP3 : dFlipFlop port map(S3, CLR, CLK, S4, S4b);
      Q(3) <= S4;
end behavior;

library ieee;
use ieee.std_logic_1164.all;

entity dFlipFlop is
      port  (
            D, DCLR, DCLK     :     in std_logic;
            Q, Qb       :     out std_logic
      );
end dFlipFlop;

architecture behavior of dFlipFlop is
begin
      process(D, DCLR, DCLK)
      begin
            if(DCLK'event and DCLK = '1') then
                  if DCLR = '1' then
                        Q <= '0';
                        Qb <= '1';
                  else
                        Q <= D;
                        Qb <= not D;
                  end if;
            end if;
      end process;
end behavior;


library ieee;
use ieee.std_logic_1164.all;

entity pisoreg is
      port(
            CLK, CLR, S : in bit;   -- S is select line for I/P or O/P
            I : in bit_vector(3 downto 0);
            Q : out bit
      );
end pisoreg;

architecture behavior of pisoreg is
begin
      process (I, CLK)
            variable temp : bit_vector(3 downto 0) := "0000";
      begin
            if(CLR='1') then
```

```
                  temp := "0000";
            elsif(CLK'event and CLK='1') then
                  if(S = '1') then  -- High = O/P
                        Q <= temp(0);
                        temp(2 downto 0) := temp(3 downto 1);
                        temp(3) := '0';
                  else                        -- Low = I/P
                        temp := I;
                  end if;
            end if;
      end process;
end behavior;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity piporeg is
      port(
            CLK, CLR : in bit;
            I : in bit_vector(3 downto 0);
            Q : out bit_vector(3 downto 0)
      );
end piporeg;

architecture behavior of piporeg is
begin
      process (I, CLK)
            variable temp : bit_vector(3 downto 0) := "0000";
      begin
            if(CLR='1') then
                  temp := "0000";
            elsif(CLK'event and CLK='1') then
                  Q <= temp;
                  temp := I;
            end if;
      end process;
end behavior;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity mux8 is
port(
      I     : in bit_vector(7 downto 0);
      S     : in bit_vector(2 downto 0);
      En : in bit;
      Y     : out bit
);
end mux8;
```

```
architecture behaviour of mux8 is
      begin
      process(I, S, En)
            begin
                  if(En = '0') then       -- Low enable
                        if(S="000")       then  Y <= I(0);
                        elsif(S="001")    then  Y <= I(1);
                        elsif(S="010")    then  Y <= I(2);
                        elsif(S="011")    then  Y <= I(3);
                        elsif(S="100")    then  Y <= I(4);
                        elsif(S="101")    then  Y <= I(5);
                        elsif(S="110")    then  Y <= I(6);
                        else                                  Y <= I(7);
                        end if;
                  else  Y<='0';
                  end if;
      end process;
end behaviour;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity mux4 is
port(
      I3, I2, I1, I0, S1, S0  : in bit;
      F                                              : out bit
);
end mux4;

architecture behaviour of mux4 is
begin
      process(I3, I2, I1, I0, S1, S0)
      begin
            if(S1='0' and S0='0')         then
                  F <= I0;
            elsif(S1='0' and S0='1')      then
                  F <= I1;
            elsif(S1='1' and S0='0')      then
                  F <= I2;
            else
                  F <= I3;
            end if;
      end process;
end behaviour;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity johnsonCounter is port ( CLR, CLK : in std_logic;
    Q : out std_logic_vector(0 to 3));
end johnsonCounter;

architecture behavior of johnsonCounter is
      component dFlipFlop port(D, DCLR, DCLK : in std_logic;
            Q, Qb : out std_logic);
      end component;

      signal S1, S2, S3, S4, S1b, S2b, S3b, S4b : std_logic;
begin
      MAP0 : dFlipFlop port map(S4b, CLR, CLK, S1, S1b);
      Q(0) <= S1;
      MAP1 : dFlipFlop port map(S1, CLR, CLK, S2, S2b);
      Q(1) <= S2;
      MAP2 : dFlipFlop port map(S2, CLR, CLK, S3, S3b);
      Q(2) <= S3;
      MAP3 : dFlipFlop port map(S3, CLR, CLK, S4, S4b);
      Q(3) <= S4;

end behavior;

library ieee;
use ieee.std_logic_1164.all;

entity dFlipFlop is
      port  (
            D, DCLR, DCLK     :     in std_logic;
            Q, Qb       :     out std_logic
      );
end dFlipFlop;

architecture behavior of dFlipFlop is
begin
      process(D, DCLR, DCLK)
      begin
            if(DCLK'event and DCLK = '1') then
                  if DCLR = '1' then
                        Q <= '0';
                        Qb <= '1';
                  else
                        Q <= D;
                        Qb <= not D;
                  end if;
            end if;
      end process;
end behavior;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity jKMSFlipFlop is
port(
      J           :     in std_logic;
      K           :     in std_logic;
      CLK   :     in std_logic;
      Q           :     out std_logic;
      Qb    :     out std_logic
);
end jKMSFlipFlop;

architecture behavior of jKMSFlipFlop is
begin
      process(J, K, CLK)
            variable temp : std_logic := '0';
      begin
            if(CLK'event and CLK='1')     then
                  if(J='0' and K='0')     then
                        temp := temp;
                  elsif(J='0' and K='1')  then
                        temp := '0';
                  elsif(J='1' and K='0')  then
                        temp := '1';
                  else
                        temp := not temp;
                  end if;
            elsif(CLK'event and CLK='0')  then
                  Q <= temp;
                  Qb <= not(temp);
            end if;
      end process;
end behavior;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity jKFlipFlop is
port(
      J           :     in std_logic;
      K           :     in std_logic;
      CLK   :     in std_logic;
      Q           :     out std_logic;
      Qb    :     out std_logic
);
end jKFlipFlop;

architecture behavior of jKFlipFlop is
begin
      process(J, K, CLK)
            variable temp : std_logic := '0';
      begin
            if(CLK'event and CLK='1')     then
                  if(J='0' and K='0')     then
                        temp := temp;
                  elsif(J='0' and K='1')  then
                        temp := '0';
                  elsif(J='1' and K='0')  then
                        temp := '1';
                  else
                        temp := not temp;
                  end if;
            end if;
            Q <= temp;
            Qb <= not(temp);
      end process;
end behavior;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity HalfAdder is
port
(
      A, B : in std_logic;
      S, C : out std_logic
);
end HalfAdder;

architecture behaviour of HalfAdder is

begin
      S <= A xor B;
      C <= A and B;

end behaviour;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
entity Half_Subtractor_Behavorial is
port  (
      X, Y : in bit;
      D, B : out bit
);
end Half_Subtractor_Behavorial;

architecture behaviour of Half_Subtractor_Behavorial is
signal E : bit;
begin
      E <= not Y;
      process(X, Y)
            begin
                  if (X = '0') then
                        B <= Y;
                        D <= Y;
                  else
                        B <= '0';
                        D <= E;
                  end if;
            end process;

end behaviour;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
entity Half_Subtractor is
port  (
      X, Y : in std_logic;
      D, B : out std_logic
);
end Half_Subtractor;

architecture behaviour of Half_Subtractor is
signal e : std_logic;
      begin
            E <= not X;
            D <= X xor Y;
            B <= E and Y;

end behaviour;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity fullSubtractor is
port  (
      Bin, X, Y   : in bit;
      D, Bout           : out bit
);
end fullSubtractor;

architecture behaviour of fullSubtractor is
begin
      process(Bin, X, Y)
            begin
                  if(Bin = '0') then
                        D           <= X xor Y;
                        Bout  <= X or Y;
                  else
                        D           <= X xnor Y;
                        Bout  <= X and Y;
                  end if;
      end process;
end behaviour;




-- Full Adder by cascading two Half Adders
-- @auth Oishik Mukhopadhyay
-- 12 May 2018 :: 21:14
library ieee;
use ieee.std_logic_1164.all;

entity fullAdder is port (Cin, X, Y : in bit; Sum, Cout : out bit);
end fullAdder;

architecture behavior of fullAdder is
      component halfAdder port(A, B : in bit; S, C : out bit);
      end component;

      signal sigsum, sigcarryA, sigcarryB : bit;
begin
      MAP1 : halfAdder port map(X, Y, sigsum, sigcarryA);
      MAP2 : halfAdder port map(sigsum, Cin, Sum, sigcarryB);
      Cout <= sigcarryA or sigcarryB;
end behavior;

library ieee;
use ieee.std_logic_1164.all;

entity halfAdder is port(A, B : in bit; S, C : out bit);
end halfAdder;

architecture halfAdderWiki of halfAdder is
begin
      S <= A xor B;
      C <= A and B;
end halfAdderWiki;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity encoder8to3 is
port  (
      En : in bit;
      I : in bit_vector(7 downto 0);
      O : out bit_vector(2 downto 0)
);
end encoder8to3;

architecture behaviour of encoder8to3 is
begin
      process(En, I)
      begin
            if (En='1') then
                  O <= "000";
            else  --Low enable
                  if (I = "00000001") then
                        O <= "000";
                  elsif (I = "00000010") then
                        O <= "001";
                  elsif (I = "00000100") then
                        O <= "010";
                  elsif (I = "00001000") then
                        O <= "011";
                  elsif (I = "00010000") then
                        O <= "100";
                  elsif (I = "00100000") then
                        O <= "101";
                  elsif (I = "01000000") then
                        O <= "110";
                  elsif (I = "10000000") then
                        O <= "111";
                  else
                        O <= "000";
                  end if;
            end if;
      end process;
end behaviour;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity enc8to3 is
port  (
      En : in bit;
      I : in bit_vector(7 downto 0);
      O : out bit_vector(2 downto 0)
);
end enc8to3;

architecture behaviour of enc8to3 is
begin
      process(En, I)
      begin
            if (En='1') then
                  O <= "000";
            else  --Low enable
                  if (I = "00000001") then
                        O <= "000";
                  elsif (I = "00000010") then
                        O <= "001";
                  elsif (I = "00000100") then
                        O <= "010";
                  elsif (I = "00001000") then
                        O <= "011";
                  elsif (I = "00010000") then
                        O <= "100";
                  elsif (I = "00100000") then
                        O <= "101";
                  elsif (I = "01000000") then
                        O <= "110";
                  elsif (I = "10000000") then
                        O <= "111";
                  else
                        O <= "000";
                  end if;
            end if;
      end process;
end behaviour;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity dFlipFlop is
      port  (
            D, CLK      :     in std_logic;
            Q, Qb       :     out std_logic
      );
end dFlipFlop;

architecture behavior of dFlipFlop is
begin
      process(D, CLK)
      begin
            if(CLK'event and CLK = '1') then
                  Q <= D;
                  Qb <= not D;
            end if;
      end process;
end behavior;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity demux8 is
port(
      I : in bit;
      En : in bit;
      S : in bit_vector(2 downto 0);
      Y : out bit_vector(7 downto 0)
);
end demux8;

architecture modelling of demux8 is
begin
      Y <=  "00000000" when En = '1' else
                  "00000000" when I = '0' else
                  "00000001" when S = "000" else
                  "00000010" when S = "001" else
                  "00000100" when S = "010" else
                  "00001000" when S = "011" else
                  "00010000" when S = "100" else
                  "00100000" when S = "101" else
                  "01000000" when S = "110" else
                  "10000000";
end modelling;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity decoder3to8 is
port(
      En : in bit;
      I     : in bit_vector(2 downto 0);
      Y     : out bit_vector(7 downto 0)
);
end decoder3to8;

architecture behaviour of decoder3to8 is
begin
            Y <= "00000000" when En = '1' else
                    "00000001" when I = "000" else
                    "00000010" when I = "001" else
                    "00000100" when I = "010" else
                    "00001000" when I = "011" else
                    "00010000" when I = "100" else
                    "00100000" when I = "101" else
                    "01000000" when I = "110" else
                    "10000000";
end behaviour;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity binToGRAYwithLOOP is
      port (
            I : in bit_vector(7 downto 0);
            O : out bit_vector(7 downto 0)
      );
end binToGRAYwithLOOP;

architecture behavior of binToGRAYwithLOOP is
begin
      process(I)
            variable index : integer;
      begin
            O(7) <= I(7);
            for index in 6 downto 0 loop
                  O(index) <= I(index+1) xor I(index);
            end loop;
      end process;
end behavior;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity binToGray is
      port (
            I : in bit_vector(3 downto 0);
            O : out bit_vector(3 downto 0)
      );
end binToGray;

architecture behavior of binToGray is
begin
      process(I)
      begin
            O(3) <= I(3);
            O(2) <= I(3) xor I(2);
            O(1) <= I(2) xor I(1);
            O(0) <= I(1) xor I(0);
      end process;
end behavior;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity bCDtoSSD is
      port(
            I : in std_logic_vector (3 downto 0);
            H : out std_logic_vector(6 downto 0)
      );
end bCDtoSSD;

architecture behavior of bCDtoSSD is
begin
            H <=  "1000000" when I = "0000" else
                        "1111001" when I = "0001" else
                        "0100100" when I = "0010" else
                        "0110000" when I = "0011" else
                        "0011001" when I = "0100" else
                        "0010010" when I = "0101" else
                        "0000010" when I = "0110" else
                        "1111000" when I = "0111" else
                        "0000000" when I = "1000" else
                        "0010000" when I = "1001" else
                        "0001000" when I = "1010" else
                        "0000011" when I = "1011" else
                        "1000110" when I = "1100" else
                        "0100001" when I = "1101" else
                        "0000110" when I = "1110" else
                        "0001110";
end behavior;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity aluMux is
port(
      A, B : in std_logic;
      S : in std_logic_vector(1 downto 0);      -- For 4 selects : 00 01 10 11
      F : out std_logic
);
end aluMux;

architecture oishik_alu of aluMux is

      component mux4to1
            port  (
                  I : in std_logic_vector(3 downto 0);
                  Selects : in std_logic_vector(1 downto 0);
                  Output : out std_logic
            );
      end component;

signal M : std_logic_vector(3 downto 0) := "0000";

begin

      M(3) <= (A nand B);
      M(2) <= (A nor B);
      M(1) <= (not A);
      M(0) <= (not B);
      map1 : mux4to1 port map (M, S, F);

end oishik_alu;

-- Definition of 4 to 1 Multiplexer

library ieee;
use ieee.std_logic_1164.all;

entity mux4to1 is
port(
      I : in std_logic_vector(3 downto 0);
      Selects : in std_logic_vector(1 downto 0);
      Output : out std_logic
);
end mux4to1;

architecture oishik_mux of mux4to1 is
begin
      Output <= I(3) when Selects = "00" else
                        I(2) when Selects = "01" else
                        I(1) when Selects = "10" else
                        I(0);
end oishik_mux;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity alu74181 is
      port(
            A, B : in std_logic_vector(7 downto 0);
            CLK, M, Cn : in std_logic;
            S : in std_logic_vector(3 downto 0);
            F : out std_logic_vector(7 downto 0)
      );
end alu74181;

architecture behavior of alu74181 is
begin
      process(A, B, CLK, M, Cn, S)
      begin
            if(CLK'event and CLK = '1') then
                  if(M='1') then    -- Logic Functions      -- Cn is ignored.
                        case S is
                              when "0000" => F <= not (A);
                              when "0001" => F <= A nand B;
                              when "0010" => F <= (not A) or B;
                              when "0011" => F <= "00000001";
                              when "0100" => F <= A nor B;
                              when "0101" => F <= not B;
                              when "0110" => F <= A xnor B;
                              when "0111" => F <= A or (not(B));
                              when "1000" => F <= not (A) and B;
                              when "1001" => F <= A xor B;
                              when "1010" => F <= B;
                              when "1011" => F <= A or B;
                              when "1100" => F <= "00000000";
                              when "1101" => F <= A and (not(B));
                              when "1110" => F <= A and B;
                              when "1111" => F <= A;
                        end case;
                  else  -- M = '0'        --Arithmetic Functions.
                        if Cn = '0' then  -- SET 1
                              case S is
                                    when "0000" => F <= A - 1;
                                    when "0001" => F <= (A and B) - 1;
                                    when "0010" => F <= (A and not(B)) - 1;
                                    when "0011" => F <= -"00000001";
                                    when "0100" => F <= A + (A or not(B));
                                    when "0101" => F <= A and B + (A or not(B));
                                    when "0110" => F <= A - B - 1;
                                    when "0111" => F <= A or not(B);
                                    when "1000" => F <= A + (A or B);
                                    when "1001" => F <= A + B;
                                    when "1010" => F <= (A and not(B))+(A or B);
                                    when "1011" => F <= A or B;
                                    when "1100" => F <= A;
                                    when "1101" => F <= (A and B) + A;
                                    when "1110" => F <= (A and not(B)) + A;
                                    when "1111" => F <= A;
                              end case;
                        else  -- Cn = '1' --SET 2
                              case S is
                                    when "0000" => F <= A;
                                    when "0001" => F <= A and B;
                                    when "0010" => F <= A and not(B);
                                    when "0011" => F <= "00000000";
                                    when "0100" => F <= A + (A or not(B)) + 1;
                                    when "0101" => F <= A and B+(A or not(B))+1;
                                    when "0110" => F <= (A or not(B)) + 1;
                                    when "0111" => F <= A - B;
                                    when "1000" => F <= A + (A or B) + 1;
                                    when "1001" => F <= A + B + 1;
                                    when "1010" => F <= (A and not B)+(A or B)+1;
                                    when "1011" => F <= (A or B) + 1;
                                    when "1100" => F <= A + 1;
                                    when "1101" => F <= (A and B) + A + 1;
                                    when "1110" => F <= (A and not(B)) + A + 1;
                                    when "1111" => F <= A + 1;
                              end case;
                        end if;
                  end if;
            end if;
      end process;
end behavior;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity alu is
      port(
            A, B : in std_logic_vector(3 downto 0);
            CLK, M : in std_logic;
            S : in std_logic_vector(2 downto 0);
            F : out std_logic_vector(3 downto 0)
      );
end alu;

architecture behavior of alu is
begin
      process(A, B, CLK, M, S)
      begin
            if(CLK'event and CLK = '1') then
                  if(M='1') then
                        case S is
                              when "000" => F <= not (A);
                              when "001" => F <= A and (not(B));
                              when "010" => F <= A xnor B;
                              when "011" => F <= "0001";
                              when "100" => F <= A and B;
                              when "101" => F <= A or (not(B));
                              when "110" => F <= B;
                              when "111" => F <= "0000";
                        end case;
                  else
                        case S is
                              when "000" => F <= A - 1;
                              when "001" => F <= not(B) + 1;
                              when "010" => F <= A + B + 1;
                              when "011" => F <= -"0001";
                              when "100" => F <= A + B - 1;
                              when "101" => F <= A and B + 1;
                              when "110" => F <= B + 1;
                              when "111" => F <= A;
                        end case;
                  end if;
            end if;
      end process;
end behavior;
```