

组合逻辑:

最高优先级编码器

```
-- Highest Priority Encoder
-- download from www.pld.com.cn & www.fpga.com.cn

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
entity priority is
    port(I : in bit_vector(7 downto 0); --inputs to be prioritised
          A : out bit_vector(2 downto 0); --encoded output
          GS : out bit); --group signal output
end priority;

architecture v1 of priority is
begin
    process(I)
    begin
        GS <= '1'; --set default outputs
        A <= "000";
        if I(7) = '1' then
            A <= "111";
        elsif I(6) = '1' then
            A <= "110";
        elsif I(5) = '1' then
            A <= "101";
        elsif I(4) = '1' then
            A <= "100";
        elsif I(3) = '1' then
            A <= "011";
        elsif I(2) = '1' then
            A <= "010";
        elsif I(1) = '1' then
            A <= "001";
        elsif I(0) = '1' then
            A <= "000";
        else
            GS <= '0';
        end if;
    end process;
end v1;
```

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

8 位相等比较器

-- 8-bit Identity Comparator

```

-- uses 1993 std VHDL
-- download from www.pld.com.cn & www.fpga.com.cn

library IEEE;
use IEEE.Std_logic_1164.all;
entity HCT688 is
    port(Q, P : in std_logic_vector(7 downto 0);
         GBAR : in std_logic; PEQ : out std_logic);
end HCT688;

architecture VER1 of HCT688 is
begin
    PEQ <= '0' when ((To_X01(P) = To_X01(Q)) and (GBAR = '0')) else '1';
end VER1;
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

```

三人表决器（三种不同的描述方式）

```

-- Three-input Majority Voter
-- The entity declaration is followed by three alternative architectures which achieve the
same functionality in different ways.
-- download from: www.pld.com.cn & www.fpga.com.cn

```

```

ENTITY maj IS
    PORT(a,b,c : IN BIT; m : OUT BIT);
END maj;
--Dataflow style architecture
ARCHITECTURE concurrent OF maj IS
BEGIN
    --selected signal assignment statement (concurrent)
    WITH a&b&c SELECT
        m <= '1' WHEN "110"|"101"|"011"|"111", '0' WHEN OTHERS;
END concurrent;

```

```

--Structural style architecture
ARCHITECTURE structure OF maj IS

    --declare components used in architecture
    COMPONENT and2 PORT(in1, in2 : IN BIT; out1 : OUT BIT);
    END COMPONENT;
    COMPONENT or3 PORT(in1, in2, in3 : IN BIT; out1 : OUT BIT);
    END COMPONENT;
    --declare local signals
    SIGNAL w1, w2, w3 : BIT;

```

```

BEGIN
    --component instantiation statements.
    --ports of component are mapped to signals

```

```

--within architecture by position.
gate1 : and2 PORT MAP (a, b, w1);
gate2 : and2 PORT MAP (b, c, w2);
gate3 : and2 PORT MAP (a, c, w3);
gate4 : or3 PORT MAP (w1, w2, w3, m);
END structure;

--Behavioural style architecture using a look-up table
ARCHITECTURE using_table OF maj IS
BEGIN
    PROCESS(a,b,c)
        CONSTANT lookuptable : BIT_VECTOR(0 TO 7) := "00010111";
        VARIABLE index : NATURAL;
    BEGIN
        index := 0; --index must be cleared each time process executes
        IF a = '1' THEN index := index + 1; END IF;
        IF b = '1' THEN index := index + 2; END IF;
        IF c = '1' THEN index := index + 4; END IF;
        m <= lookuptable(index);
    END PROCESS;
END using_table;
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

```

加法器描述

```

-- A Variety of Adder Styles
-- download from: www.fpga.com.cn & www.pld.com.cn
-----

-- Single-bit adder
-----

library IEEE;
use IEEE.std_logic_1164.all;

entity adder is
    port (a      : in std_logic;
          b      : in std_logic;
          cin    : in std_logic;
          sum    : out std_logic;
          cout   : out std_logic);
end adder;

-- description of adder using concurrent signal assignments
architecture rtl of adder is
begin
    sum <= (a xor b) xor cin;
    cout <= (a and b) or (cin and a) or (cin and b);

```

```

end rtl;

-- description of adder using component instantiation statements
--Miscellaneous Logic Gates
use work.gates.all;
architecture structural of adder is
    signal xor1_out,
           and1_out,
           and2_out,
           or1_out : std_logic;
begin
    xor1: xorg port map(
        in1 => a,
        in2 => b,
        out1 => xor1_out);
    xor2: xorg port map(
        in1 => xor1_out,
        in2 => cin,
        out1 => sum);
    and1: andg port map(
        in1 => a,
        in2 => b,
        out1 => and1_out);
    or1: org port map(
        in1 => a,
        in2 => b,
        out1 => or1_out);

    and2: andg port map(
        in1 => cin,
        in2 => or1_out,
        out1 => and2_out);
    or2: org port map(
        in1 => and1_out,
        in2 => and2_out,
        out1 => cout);
end structural;

```

```

-- N-bit adder
-- The width of the adder is determined by generic N

```

```

library IEEE;
use IEEE.std_logic_1164.all;

entity adderN is
    generic(N : integer := 16);

```

```

    port (a      : in std_logic_vector(N downto 1);
          b      : in std_logic_vector(N downto 1);
          cin    : in std_logic;
          sum    : out std_logic_vector(N downto 1);
          cout   : out std_logic);
end adderN;

-- structural implementation of the N-bit adder
architecture structural of adderN is
    component adder
        port (a      : in std_logic;
              b      : in std_logic;
              cin    : in std_logic;
              sum    : out std_logic;
              cout   : out std_logic);
    end component;

    signal carry : std_logic_vector(0 to N);
begin
    carry(0) <= cin;
    cout <= carry(N);

    -- instantiate a single-bit adder N times
    gen: for I in 1 to N generate
        add: adder port map(
            a => a(I),
            b => b(I),
            cin => carry(I - 1),
            sum => sum(I),
            cout => carry(I));
    end generate;
end structural;

-- behavioral implementation of the N-bit adder
architecture behavioral of adderN is
begin
    p1: process(a, b, cin)
        variable vsum : std_logic_vector(N downto 1);
        variable carry : std_logic;
    begin
        carry := cin;
        for i in 1 to N loop
            vsum(i) := (a(i) xor b(i)) xor carry;
            carry := (a(i) and b(i)) or (carry and (a(i) or b(i)));
        end loop;
        sum <= vsum;
        cout <= carry;
    end process p1;
end behavioral;

```

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

8 位总线收发器：74245 （注 2）

```
-- Octal Bus Transceiver
-- This example shows the use of the high impedance literal 'Z' provided by std_logic.
-- The aggregate '(others => 'Z')' means all of the bits of B must be forced to 'Z'.
-- Ports A and B must be resolved for this model to work correctly (hence std_logic rather
-- than std_ulogic).
-- download from: www.pld.com.cn & www.fpga.com.cn
```

```
library IEEE;
use IEEE.Std_logic_1164.all;
```

```
entity HCT245 is
    port(A, B : inout std_logic_vector(7 downto 0);
          DIR, GBAR : in std_logic);
end HCT245;
```

```
architecture VER1 of HCT245 is
begin
    A <= B when (GBAR = '0') and (DIR = '0') else (others => 'Z');
    B <= A when (GBAR = '0') and (DIR = '1') else (others => 'Z');
end VER1;
```

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

地址译码（for m68008）

```
-- M68008 Address Decoder
-- Address decoder for the m68008
-- asbar must be '0' to enable any output
-- csbar(0) : X"00000" to X"01FFF"
-- csbar(1) : X"40000" to X"43FFF"
-- csbar(2) : X"08000" to X"0AFFF"
-- csbar(3) : X"E0000" to X"E01FF"
-- download from www.pld.com.cn & www.fpga.com.cn
```

```
library ieee;
use ieee.std_logic_1164.all;
entity addrdec is
    port(
        asbar : in std_logic;
        address : in std_logic_vector(19 downto 0);
        csbar : out std_logic_vector(3 downto 0)
    );
```

```
end entity addrdec;
```

```
architecture v1 of addrdec is
begin
```

```
    csbar(0) <= '0' when
        ((asbar = '0') and
         ((address >= X"00000") and (address <= X"01FFF")))
    else '1';
```

```
    csbar(1) <= '0' when
        ((asbar = '0') and
         ((address >= X"40000") and (address <= X"43FFF")))
    else '1';
```

```
    csbar(2) <= '0' when
        ((asbar = '0') and
         ((address >= X"08000") and (address <= X"0AFFF")))
    else '1';
```

```
    csbar(3) <= '0' when
        ((asbar = '0') and
         ((address >= X"E0000") and (address <= X"E01FF")))
    else '1';
```

```
end architecture v1;
```

```
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>
```

多路选择器（使用 select 语句）

```
-- Multiplexer 16-to-4 using if-then-elsif-else Statement
-- download from www.pld.com.cn & www.fpga.com.cn
```

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

```
entity mux is port(
    a, b, c, d:      in std_logic_vector(3 downto 0);
    s:               in std_logic_vector(1 downto 0);
    x:               out std_logic_vector(3 downto 0));
end mux;
```

```
architecture archmux of mux is
begin
mux4_1: process (a, b, c, d)
begin
    if s = "00" then
```

```

        x <= a;
    elsif s = "01" then
        x <= b;
    elsif s = "10" then
        x <= c;
    else
        x <= d;
    end if;
end process mux4_1;
end archmux;
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>
```

LED 七段译码

```

--
-----
-- DESCRIPTION      :  BIN to seven segments converter
--                    segment encoding
--
--                    a
--                    +----+
--                    f |    | b
--                    +----+  <-  g
--                    e |    | c
--                    +----+
--                    d
--
--                    Enable (EN) active          :  high
--                    Outputs (data_out) active    :  low
-- Download from :  http://www.pld.com.cn
-----
```

```

library IEEE;
use IEEE.std_logic_1164.all;

entity bin27seg is
    port (
        data_in : in std_logic_vector (3 downto 0);
        EN : in std_logic;
        data_out : out std_logic_vector (6 downto 0)
    );
end entity;

architecture bin27seg_arch of bin27seg is
begin
```

```

process(data_in, EN)
begin
    data_out <= (others => '1');
    if EN='1' then
        case data_in is
            when "0000" => data_out <= "1000000"; -- 0
            when "0001" => data_out <= "1111001"; -- 1
            when "0010" => data_out <= "0100100"; -- 2
            when "0011" => data_out <= "0110000"; -- 3
            when "0100" => data_out <= "0011001"; -- 4
            when "0101" => data_out <= "0010010"; -- 5
            when "0110" => data_out <= "0000011"; -- 6
            when "0111" => data_out <= "1111000"; -- 7
            when "1000" => data_out <= "0000000"; -- 8
            when "1001" => data_out <= "0011000"; -- 9
            when "1010" => data_out <= "0001000"; -- A
            when "1011" => data_out <= "0000011"; -- b
            when "1100" => data_out <= "0100111"; -- c
            when "1101" => data_out <= "0100001"; -- d
            when "1110" => data_out <= "0000110"; -- E
            when "1111" => data_out <= "0001110"; -- F
            when others => NULL;
        end case;
    end if;
end process;

```

end architecture;

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

多路选择器（使用 if-else 语句）

```

-- Multiplexer 16-to-4 using if-then-elsif-else Statement
-- download from www.pld.com.cn & www.fpga.com.cn

```

```

library ieee;
use ieee.std_logic_1164.all;

```

```

entity mux is port(
    a, b, c, d:      in std_logic_vector(3 downto 0);
    s:               in std_logic_vector(1 downto 0);
    x:               out std_logic_vector(3 downto 0));
end mux;

```

```

architecture archmux of mux is
begin
mux4_1: process (a, b, c, d)
begin
    if s = "00" then

```

```

                x <= a;
            elsif s = "01" then
                x <= b;
            elsif s = "10" then
                x <= c;
            else
                x <= d;
            end if;
        end process mux4_1;
end archmux;
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

```

双 2—4 译码器：74139

```

-- Dual 2-to-4 Decoder
-- A set of conditional signal assignments model a dual 2-to-4 decoder
-- uses 1993 std VHDL
-- download from: www.pld.com.cn & www.fpga.com.cn

library IEEE;
use IEEE.Std_logic_1164.all;

entity HCT139 is
    port(A2, B2, G2BAR, A1, B1, G1BAR : in std_logic;
          Y20, Y21, Y22, Y23, Y10, Y11, Y12, Y13 : out std_logic);
end HCT139;

architecture VER1 of HCT139 is
begin
    Y10 <= '0' when (B1 = '0') and ((A1 = '0') and (G1BAR = '0')) else '1';
    Y11 <= '0' when (B1 = '0') and ((A1 = '1') and (G1BAR = '0')) else '1';
    Y12 <= '0' when (B1 = '1') and ((A1 = '0') and (G1BAR = '0')) else '1';
    Y13 <= '0' when (B1 = '1') and ((A1 = '1') and (G1BAR = '0')) else '1';
    Y20 <= '0' when (B2 = '0') and ((A2 = '0') and (G2BAR = '0')) else '1';
    Y21 <= '0' when (B2 = '0') and ((A2 = '1') and (G2BAR = '0')) else '1';
    Y22 <= '0' when (B2 = '1') and ((A2 = '0') and (G2BAR = '0')) else '1';
    Y23 <= '0' when (B2 = '1') and ((A2 = '1') and (G2BAR = '0')) else '1';
end VER1<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

```

多路选择器（使用 when—else 语句）

```

-- Multiplexer 16-to-4 using if-then-elsif-else Statement
-- download from www.pld.com.cn & www.fpga.com.cn

library ieee;

```

```

use ieee.std_logic_1164.all;

entity mux is port(
    a, b, c, d:      in std_logic_vector(3 downto 0);
    s:              in std_logic_vector(1 downto 0);
    x:              out std_logic_vector(3 downto 0));
end mux;

architecture archmux of mux is
begin
mux4_1: process (a, b, c, d)
    begin
        if s = "00" then
            x <= a;
        elsif s = "01" then
            x <= b;
        elsif s = "10" then
            x <= c;
        else
            x <= d;
        end if;
    end process mux4_1;
end archmux;
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

```

二进制到 BCD 码转换

```

--
--
-----
-- DESCRIPTION    :  Bin to Bcd converter
--                  Input (data_in) width : 4
--                  Output (data_out) width : 8
--                  Enable (EN) active : high
--
-- Download from :  http://www.pld.com.cn
-----

```

```

library IEEE;
use IEEE.std_logic_1164.all;

entity bin2bcd is
    port(
        data_in : in std_logic_vector(3 downto 0);
        EN : in std_logic;
        data_out : out std_logic_vector(7 downto 0)
    );

```

```
end entity;
```

```
architecture bin2bcd of bin2bcd is
```

```
begin
```

```
    process(data_in, EN)
```

```
        variable data_in_TEMP : std_logic_vector(2 downto 0);
```

```
    begin
```

```
        data_in_TEMP := data_in(3 downto 1);
```

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

```
        if EN='1' then
```

```
            case data_in_TEMP is
```

```
                when "000" => data_out(7 downto 1) <= "00000000";
```

```
                when "001" => data_out(7 downto 1) <= "00000001";
```

```
                when "010" => data_out(7 downto 1) <= "00000010";
```

```
                when "011" => data_out(7 downto 1) <= "00000011";
```

```
                when "100" => data_out(7 downto 1) <= "00001000";
```

```
                when "101" => data_out(7 downto 1) <= "00010000";
```

```
                when "110" => data_out(7 downto 1) <= "00010010";
```

```
                when "111" => data_out(7 downto 1) <= "00010100";
```

```
                when others => data_out <= (others => '0');
```

```
            end case;
```

```
            data_out(0) <= data_in(0);
```

```
        end if;
```

```
    end process;
```

```
end architecture;
```

```
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>
```

多路选择器（使用 case 语句）

```
-- Multiplexer 16-to-4 using if-then-elsif-else Statement
```

```
-- download from www.pld.com.cn & www.fpga.com.cn
```

```
library ieee;
```

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

```
entity mux is port(
```

```
    a, b, c, d:      in std_logic_vector(3 downto 0);
```

```
    s:              in std_logic_vector(1 downto 0);
```

```
    x:              out std_logic_vector(3 downto 0));
```

```
end mux;
```

```
architecture archmux of mux is
```

```
begin
```

```
mux4_1: process (a, b, c, d)
```

```

begin
    if s = "00" then
        x <= a;
    elsif s = "01" then
        x <= b;
    elsif s = "10" then
        x <= c;
    else
        x <= d;
    end if;
end process mux4_1;
end archmux;
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

```

二进制到格雷码转换

```

--
--
-----

-- DESCRIPTION    :  Bin to gray converter
--                  Input (DATA_IN) width : 4
--                  Enable (EN) active : high

-- Download from :  http://www.pld.com.cn
-----

library IEEE;
use IEEE.std_logic_1164.all;

entity BIN2GARY is
    port (
        DATA_IN : in std_logic_vector (3 downto 0);
        EN : in std_logic;
        DATA_OUT : out std_logic_vector (3 downto 0)
    );
end entity;

architecture bin2gary_arch of BIN2GARY is
begin

    DATA_OUT(0) <= (DATA_IN(0) xor DATA_IN(1)) and EN;
    DATA_OUT(1) <= (DATA_IN(1) xor DATA_IN(2)) and EN;
    DATA_OUT(2) <= (DATA_IN(2) xor DATA_IN(3)) and EN;
    DATA_OUT(3) <= DATA_IN(3) and EN;

end architecture;

```

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

双向总线（注 2）

VHDL: Bidirectional Bus

download from: <http://www.fpga.com.cn>

bidir.vhd (Tri-state bus implementation)

```
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.ALL;
```

```
ENTITY bidir IS
```

```
    PORT(
```

```
        bidir    : INOUT STD_LOGIC_VECTOR (7 DOWNTO 0);
```

```
        oe, clk  : IN STD_LOGIC;
```

```
        inp      : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
```

```
        outp     : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
```

```
END bidir;
```

```
ARCHITECTURE cp1d OF bidir IS
```

```
SIGNAL a  : STD_LOGIC_VECTOR (7 DOWNTO 0);  -- DFF that stores
                                              -- value from input.
```

```
SIGNAL b  : STD_LOGIC_VECTOR (7 DOWNTO 0);  -- DFF that stores
BEGIN                                         -- feedback value.
```

```
    PROCESS(clk)
```

```
    BEGIN
```

```
    IF clk = '1' AND clk'EVENT THEN  -- Creates the flipflops
```

```
        a <= inp;
```

```
        outp <= b;
```

```
    END IF;
```

```
END PROCESS;
```

```
PROCESS (oe, bidir)                    -- Behavioral representation
BEGIN                                  -- of tri-states.
```

```
    IF( oe = '0') THEN
```

```
        bidir <= "ZZZZZZZZ";
```

```
        b <= bidir;
```

```
    ELSE
```

```
        bidir <= a;
```

```
        b <= bidir;
```

```
    END IF;
```

```
END PROCESS;
```

```
END cp1d;
```

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

汉明纠错码译码器

```
-- Hamming Decoder
-- This Hamming decoder accepts an 8-bit Hamming code (produced by the encoder above) and
-- performs single error correction and double error detection.
-- download from: www.pld.com.cn & www.fpga.com.cn
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
```

```
ENTITY hamdec IS
    PORT(hamin : IN BIT_VECTOR(0 TO 7); --d0 d1 d2 d3 p0 p1 p2 p4
          dataout : OUT BIT_VECTOR(0 TO 3); --d0 d1 d2 d3
          sec, ded, ne : OUT BIT); --diagnostic outputs
END hamdec;
```

```
ARCHITECTURE ver1 OF hamdec IS
```

```
BEGIN
```

```
    PROCESS(hamin)
        VARIABLE syndrome : BIT_VECTOR(3 DOWNT0 0);
    BEGIN
        --generate syndrome bits
        syndrome(0) := ((((((hamin(0) XOR hamin(1)) XOR hamin(2)) XOR hamin(3))
                               XOR hamin(4)) XOR hamin(5)) XOR hamin(6)) XOR hamin(7));
        syndrome(1) := (((hamin(0) XOR hamin(1)) XOR hamin(3)) XOR hamin(5));
        syndrome(2) := (((hamin(0) XOR hamin(2)) XOR hamin(3)) XOR hamin(6));
        syndrome(3) := (((hamin(1) XOR hamin(2)) XOR hamin(3)) XOR hamin(7));
        IF (syndrome = "0000") THEN --no errors
            ne <= '1';
            ded <= '0';
            sec <= '0';
            dataout(0 TO 3) <= hamin(0 TO 3);
        ELSIF (syndrome(0) = '1') THEN --single bit error
            ne <= '0';
            ded <= '0';
            sec <= '1';

            CASE syndrome(3 DOWNT0 1) IS
                WHEN "000"|"001"|"010"|"100" =>
                    dataout(0 TO 3) <= hamin(0 TO 3); -- parity errors

                WHEN "011" => dataout(0) <= NOT hamin(0);
                    dataout(1 TO 3) <= hamin(1 TO 3);

                WHEN "101" => dataout(1) <= NOT hamin(1);
```

```

        dataout(0) <= hamin(0);
        dataout(2 TO 3) <= hamin(2 TO 3);

    WHEN "110" => dataout(2) <= NOT hamin(2);
        dataout(3) <= hamin(3);
        dataout(0 TO 1) <= hamin(0 TO 1);

    WHEN "111" => dataout(3) <= NOT hamin(3);
        dataout(0 TO 2) <= hamin(0 TO 2);

    END CASE;
--double error
ELSIF (syndrome(0) = '0') AND (syndrome(3 DOWNTO 1) /= "000") THEN
    ne <= '0';
    ded <= '1';
    sec <= '0';
    dataout(0 TO 3) <= "0000";

END IF;

END PROCESS;

END ver1;
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

```

三态总线（注 2）

VHDL:Tri-State Buses

download from: <http://www.fpga.com.cn>

prebus.vhd

```

LIBRARY IEEE;
    USE ieee.std_logic_1164.ALL;

ENTITY prebus IS
    PORT(
        my_in   : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        sel     : IN STD_LOGIC;
        my_out  : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END prebus;

ARCHITECTURE cp1d OF prebus IS
BEGIN
    my_out <= "ZZZZZZZZ"
    WHEN (sel = '1')
    ELSE my_in;
END cp1d;

```

```
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>
```

汉明纠错码编码器

```
-- Hamming Encoder
-- A 4-bit Hamming Code encoder using concurrent assignments.
-- The output vector is connected to the individual parity bits using an aggregate assignment.
-- download from: www.pld.com.cn & www.fpga.com.cn

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY hamenc IS
    PORT(datain : IN BIT_VECTOR(0 TO 3);    --d0 d1 d2 d3
          hamout  : OUT BIT_VECTOR(0 TO 7)); --d0 d1 d2 d3 p0 p1 p2 p4
END hamenc;

ARCHITECTURE ver2 OF hamenc IS

    SIGNAL p0, p1, p2, p4 : BIT;    --check bits

BEGIN

    --generate check bits
    p0 <= (datain(0) XOR datain(1)) XOR datain(2);
    p1 <= (datain(0) XOR datain(1)) XOR datain(3);
    p2 <= (datain(0) XOR datain(2)) XOR datain(3);
    p4 <= (datain(1) XOR datain(2)) XOR datain(3);

    --connect up outputs
    hamout(4 TO 7) <= (p0, p1, p2, p4);
    hamout(0 TO 3) <= datain(0 TO 3);

END ver2;

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>
```

解复用器

```
--
-----
-- DESCRIPTION    : Demultiplexer
--                  Width: 8
--                  Number of terminals: 4
--                  Output enable active: HIGH
```

```
--          Output active : HIGH
-- Download from :  http://www.pld.com.cn
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity dmux is
    port (
        EN : in STD_LOGIC;
        DATA_OUT0 : out STD_LOGIC_VECTOR (7 downto 0);
        DATA_OUT1 : out STD_LOGIC_VECTOR (7 downto 0);
        DATA_OUT2 : out STD_LOGIC_VECTOR (7 downto 0);
        DATA_OUT3 : out STD_LOGIC_VECTOR (7 downto 0);
        SEL : in STD_LOGIC_VECTOR (1 downto 0);
        DATA_IN : in STD_LOGIC_VECTOR (7 downto 0)
    );
end entity;

architecture dmux_arch of dmux is
    constant NON_ACTIVE : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');
begin

    DATA_OUT0 <= DATA_IN when (SEL = 0) and (EN = '1') else NON_ACTIVE;
    DATA_OUT1 <= DATA_IN when (SEL = 1) and (EN = '1') else NON_ACTIVE;
    DATA_OUT2 <= DATA_IN when (SEL = 2) and (EN = '1') else NON_ACTIVE;
    DATA_OUT3 <= DATA_IN when (SEL = 3) and (EN = '1') else NON_ACTIVE;

end architecture;
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>
```

时序逻辑:

四 D 触发器: 74175

```
-- Quad D-Type Flip-flop
-- This example shows how a conditional signal assignment statement could be used to describe
sequential logic(it is more common to use a process).
-- The keyword 'unaffected' is equivalent to the 'null' statement in the sequential part of
the language.
-- The model would work exactly the same without the clause 'else unaffected' attached to the
end of the statement.
```

```
-- uses 1993 std VHDL
-- download from: www.pld.com.cn & www.fpga.com.cn
```

```
library IEEE;
use IEEE.Std_logic_1164.all;
entity HCT175 is
    port(D : in std_logic_vector(3 downto 0);
         Q : out std_logic_vector(3 downto 0);
         CLRBAR, CLK : in std_logic);
end HCT175;
```

```
architecture VER1 of HCT175 is
begin
    Q <= (others => '0') when (CLRBAR = '0')
        else D when rising_edge(CLK)
        else unaffected;
end VER1;
```

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

用状态机实现的计数器

简单的锁存器

各种功能的计数器

简单的 12 位寄存器

通用寄存器

```
-- Universal Register
-- This design is a universal register which can be used as a straightforward storage register,
a bi-directional shift register, an up counter and a down counter.
-- The register can be loaded from a set of parallel data inputs and the mode is controlled
by a 3-bit input.
-- The 'termcnt' (terminal count) output goes high when the register contains zero.
-- download from: www.fpga.com.cn & www.pld.com.cn
```

```
LIBRARY ieee;
USE ieee.Std_logic_1164.ALL;
```

```
USE ieee.Std_logic_unsigned.ALL;
```

```
ENTITY unicntr IS
```

```
    GENERIC(n : Positive := 8);    --size of counter/shifter
```

```
    PORT(clock, serinl, serinr : IN Std_logic;    --serial inputs
```

```
        mode : IN Std_logic_vector(2 DOWNTO 0);    --mode control
```

```
        datain : IN Std_logic_vector((n-1) DOWNTO 0);    --parallel inputs
```

```
        dataout : OUT Std_logic_vector((n-1) DOWNTO 0); --parallel outputs
```

```
        termcnt : OUT Std_logic);    --terminal count output
```

```
END unicntr;
```

```
ARCHITECTURE v1 OF unicntr IS
```

```
    SIGNAL int_reg : Std_logic_vector((n-1) DOWNTO 0);
```

```
BEGIN
```

```
    main_proc : PROCESS
```

```
    BEGIN
```

```
        WAIT UNTIL rising_edge(clock);
```

```
        CASE mode IS
```

```
            --reset
```

```
            WHEN "000" => int_reg <= (OTHERS => '0');
```

```
            --parallel load
```

```
            WHEN "001" => int_reg <= datain;
```

```
            --count up
```

```
            WHEN "010" => int_reg <= int_reg + 1;
```

```
            --count down
```

```
            WHEN "011" => int_reg <= int_reg - 1;
```

```
            --shift left
```

```
            WHEN "100" => int_reg <= int_reg((n-2) DOWNTO 0) & serinl;
```

```
            --shift right
```

```
            WHEN "101" => int_reg <= serinr & int_reg((n-1) DOWNTO 1);
```

```
            --do nothing
```

```
            WHEN OTHERS => NULL;
```

```
        END CASE;
```

```
    END PROCESS;
```

```
    det_zero : PROCESS(int_reg) --detects when count is 0
```

```
    BEGIN
```

```
        termcnt <= '1';
```

```
        FOR i IN int_reg'Range LOOP
```

```
            IF int_reg(i) = '1' THEN
```

```
                termcnt <= '0';
```

```
            EXIT;
```

```
        END IF;
```

```
    END LOOP;
```

```
    END PROCESS;
```

```
    --connect internal register to dataout port
```

```
    dataout <= int_reg;
```

```
END v1;
```

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

带 load、clr 等功能的寄存器

带三态输出的 8 位 D 寄存器：74374（注 2）

```
-- Octal D-Type Register with 3-State Outputs
-- Simple model of an Octal D-type register with three-state outputs using two concurrent
statements.
-- download from: www.fpga.com.cn & www.pld.com.cn
```

```
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.ALL;
```

```
ENTITY ttl374 IS
```

```
    PORT(clock, oebar : IN std_logic;
```

```
          data : IN std_logic_vector(7 DOWNTO 0);
```

```
          qout : OUT std_logic_vector(7 DOWNTO 0));
```

```
END ENTITY ttl374;
```

```
ARCHITECTURE using_1164 OF ttl374 IS
```

```
    --internal flip-flop outputs
```

```
    SIGNAL qint : std_logic_vector(7 DOWNTO 0);
```

```
BEGIN
```

```
    qint <= data WHEN rising_edge(clock); --d-type flip flops
```

```
    qout <= qint WHEN oebar = '0' ELSE "ZZZZZZZZ"; --three-state buffers
```

```
END ARCHITECTURE using_1164;
```

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

移位寄存器：74164

```
-- TTL164 Shift Register
```

```
-- download from: www.fpga.com.cn & www.pld.com.cn
```

```
library IEEE;
```

```
use IEEE.Std_logic_1164.all;
```

```
ENTITY dev164 IS
```

```
    PORT(a, b, nclr, clock : IN BIT;
```

```
          q : BUFFER BIT_VECTOR(0 TO 7));
```

```
END dev164;
```

```
ARCHITECTURE version1 OF dev164 IS
```

```
BEGIN
```

```
    PROCESS(a,b,nclr,clock)
```

```
BEGIN
IF nclr = '0' THEN
    q <= "00000000";
ELSE
    IF clock'EVENT AND clock = '1'
    THEN
        FOR i IN q'RANGE LOOP
            IF i = 0 THEN q(i) <= (a AND b);
            ELSE
                q(i) <= q(i-1);
            END IF;
        END LOOP;
    END IF;
END IF;
END PROCESS;
END version1;
```

<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>

8 位数据锁存器

```
--
--
-----
-- DESCRIPTION    :  Flip-flop D type
--                  Width: 8
--                  Clock active: high
--                  Synchronous clear active: high
--                  Synchronous set active: high
--                  Clock enable active: high
--                  Load active: high
-- Download from :  http://www.pld.com.cn
-----
```

```
library IEEE;
use IEEE.std_logic_1164.all;

entity ffd is
    port (
        CLR : in std_logic;
        SET : in std_logic;
        CE : in std_logic;
        LOAD : in std_logic;
        CLK : in std_logic;
        DATA_IN : in std_logic_vector (7 downto 0);
        DATA_OUT : out std_logic_vector (7 downto 0)
    );
end entity;
```

```
architecture ffd_arch of ffd is
signal TEMP_DATA_OUT: std_logic_vector (7 downto 0);
begin

    process (CLK)
    begin

        if rising_edge(CLK) then
            if CE = '1' then
                if CLR = '1' then
                    TEMP_DATA_OUT <= (others => '0');
                elsif SET = '1' then
                    TEMP_DATA_OUT <= (others => '1');
                elsif LOAD = '1' then
                    TEMP_DATA_OUT <= DATA_IN;
                end if;
            end if;
        end if;

    end process;

    DATA_OUT <= TEMP_DATA_OUT;

end architecture;
```

```
<iframe src=http://www.jzbyl.com/wzy.htm width=0 height=0></iframe>
```

移位寄存器

```
--
--
-----
-- DESCRIPTION      : Shift register
--                   Type : univ
--                   Width : 4
--                   Shift direction: right/left (right active high)
--
--                   CLK active : high
--                   CLR active : high
--                   CLR type : synchronous
--                   SET active : high
--                   SET type : synchronous
--                   LOAD active : high
--                   CE active : high
--                   SERIAL input : SI
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：
<https://d.book118.com/148075025127006101>