

Chapter A Hardware Design

VHDL

Decoder 3x8 (Ver 1)

```

library ieee;
use ieee.std_logic_1164.all;
entity Main is
port( Yout : out  std_logic_vector(7 downto 0);
      EN   : in   std_logic;
      Xin  : in   std_logic_vector(2 downto 0));
end Main;
-----
architecture Test of Main is
begin
process (EN, Xin)
begin
IF EN = '1' THEN
Yout <= "00000000";
ELSE
CASE Xin IS
WHEN "000" => Yout <= "00000001";
WHEN "001" => Yout <= "00000010";
WHEN "010" => Yout <= "00000100";
WHEN "011" => Yout <= "00001000";
WHEN "100" => Yout <= "00010000";
WHEN "101" => Yout <= "00100000";
WHEN "110" => Yout <= "01000000";
WHEN "111" => Yout <= "10000000";
WHEN others => Yout <= "-----";
END CASE;
END IF;
end process;
end Test;

```

Decoder 3x8 (Ver 2)

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Main is
port( Yout : out  std_logic_vector(7 downto 0);
      EN   : in   std_logic;
      Xin  : in   unsigned(2 downto 0));
end Main;
-----
architecture Test of Main is
begin
process (EN, Xin)
begin
Yout <= "00000000";
IF EN = '1' THEN
Yout <= "00000000";
ELSE
Yout (To_Integer(Xin)) <= '1';
END IF;
end process;
end Test;

```

Encoder 8x3 (Ver 1)

```

library ieee;
use ieee.std_logic_1164.all;
entity Main is
port( Yout : out  std_logic_vector(2 downto 0);
      Xin  : in   std_logic_vector(7 downto 0));
end Main;
-----
architecture Test of Main is
begin
Prol: process(Xin)
begin
CASE Xin IS
WHEN "00000001" => Yout <= "000";
WHEN "00000010" => Yout <= "001";
WHEN "00000100" => Yout <= "010";
WHEN "00001000" => Yout <= "011";
WHEN "00010000" => Yout <= "100";
WHEN "00100000" => Yout <= "101";
WHEN "01000000" => Yout <= "110";
WHEN "10000000" => Yout <= "111";
WHEN others => Yout <= "---"; -- Out of rang
END CASE;
end process;
end Test;

```

Priority Encoder 8x3 (Ver 2)

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Main is
port( Yout : out  unsigned(2 downto 0);
      Xin  : in   std_logic_vector(7 downto 0));
end Main;
-----
architecture Test of Main is
begin
Prol: process(Xin)
begin
FOR k IN 0 TO 7 LOOP
IF Xin(k) = '1' THEN
Yout <= To_Unsigned(k,3);
EXIT; -- Without exit is priority high
END IF;
END LOOP;
end process;
end Test;

```

Mux 4x1 (Ver 1)

```

library ieee;
use ieee.std_logic_1164.all;
entity Main is
port( Yout : out  std_logic;
      EN   : in   std_logic;
      Xin  : in   std_logic_vector(3 downto 0);
      SEL  : in   std_logic_vector(1 downto 0));
end Main;
-----
architecture Test of Main is
begin
Prol: process
begin
IF EN = '0' then
Yout <= 'Z';
ELSE
CASE SEL IS
WHEN "00" => Yout <= Xin(0);
WHEN "01" => Yout <= Xin(1);
WHEN "10" => Yout <= Xin(2);
WHEN "11" => Yout <= Xin(3);
WHEN others => Yout <= '-'; -- Out of rang
END CASE;
END IF;
end process;
end Test;

```

Mux 4x1 (Ver 2)

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Main is
  port( Yout  : out  std_logic;
        EN   : in   std_logic;
        Xin  : in   std_logic_vector(3 downto 0);
        SEL  : in   unsigned(1 downto 0));
end Main;
-----
architecture Test of Main is
begin
  Yout <= 'Z' WHEN EN = '0' ELSE Xin(to_integer(SEL));
end Test;

```

ALU (Entity)

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Main is
  port( X1   : in   signed(7 downto 0);
        X2   : in   signed(7 downto 0);
        Y    : out  signed(7 downto 0);
        Co   : out  std_logic;
        SEL  : in   std_logic_vector(2 downto 0);
        M    : in   std_logic);
end Main;
-----

```

ALU (Architecture)

```

architecture Test of Main is
begin
  process(X1, X2, M, SEL)
    variable Temp: signed(8 downto 0);
  begin
    case (M & SEL) is
      when "0000" => (Co, Y) <= ('0' & X1);
      when "0001" => (Co, Y) <= ('0' & X2);
      when "0010" => (Co, Y) <= ('0' & X1) + ('0' & X2);
      when "0011" => (Co, Y) <= ('0' & X1) - ('0' & X2);
      when "0100" => (Co, Y) <= ('0' & X1) + 1;
      when "0101" => (Co, Y) <= ('0' & X1) - 1;
      when "0110" => (Co, Y) <= '0' & (~X1);
      when "0111" => (Co, Y) <= '0' & (~X2);
      when "1000" => (Co, Y) <= ('0' & (X1 or X2));
      when "1001" => (Co, Y) <= ('0' & (X1 and X2));
      when "1010" => (Co, Y) <= ('0' & (X1 xor X2));
      when "1011" => (Co, Y) <= ('0' & (X1 nand X2));
      when "1100" => (Co, Y) <= ('0' & (X1 nor X2));
      when "1101" => (Co, Y) <= ('0' & (X1 xnor X2));
      when "1110" => (Co, Y) <= '0' & (not X1);
      when "1111" => (Co, Y) <= '0' & (not X2);
      when others =>
        end case;
    end process;
  end Test;

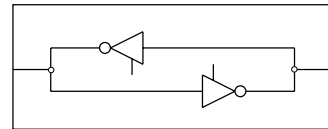
```

Bidirectional Bus

```

library ieee;
use ieee.std_logic_1164.all;
entity Main is
  port( X1, X2 : inout std_logic_vector(7 downto 0);
        Dir    : in   std_logic);
end Main;
-----
architecture Test of Main is
begin
  X1 <= not X2 when Dir = '1' else (OTHERS => 'Z');
  X2 <= not X1 when Dir = '0' else (OTHERS => 'Z');
end Test;

```



Positive Edge Triggered D Flip-flop

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY dff IS
  PORT (d, clk : IN std_logic;
        q : OUT std_logic);
END dff;

ARCHITECTURE behavior OF dff IS
BEGIN
  PROCESS (clk)
    -- sensitive ONLY to clk
  BEGIN
    -- rising clk edge
    IF (clk'EVENT AND clk = '1') THEN q <= d;
    ELSE q <= q; -- NOT needed (implied)
    END IF;
  END PROCESS;
END behavior;

```

Positive Edge JK Flip-flop

```

library ieee;
USE ieee.std_logic_1164.all;

ENTITY MYJKFF IS PORT (
  CLK, P, R, J, K : in bit;
  Q                : buffer bit);
END MYJKFF;

ARCHITECTURE Flaxer OF MYJKFF IS
BEGIN
  process(CLK, R, P)
  BEGIN
    IF R = '1' THEN
      Q <= '0';
    ELSIF P = '1' THEN
      Q <= '1';
    ELSIF (CLK'event and CLK = '1') THEN
      CASE J & K IS
        WHEN "00" => Q <= Q;
        WHEN "01" => Q <= '0';
        WHEN "10" => Q <= '1';
        WHEN "11" => Q <= not Q;
      END CASE;
    END IF;
  end process;
END Flaxer;

```

Register (Array of Flip-flops)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY Register8 IS
    PORT (d : IN std_logic_vector (0 TO 7); -- 8 bit register
          clk : IN std_logic;
          q : OUT std_logic_vector (0 TO 7));
END Register8;

ARCHITECTURE MyReg OF register8 IS
BEGIN
    PROCESS (clk)
    BEGIN
        IF rising_edge(clk) THEN q <= d;    -- using the function
        END IF;
    END PROCESS;
END MyReg ;
```

Reset and Preset for Flip-flops

```
PROCESS (clk, reset)    -- sensitive to clk or reset
BEGIN
    IF reset = '1' THEN
        q <= '0'; -- Async reset
    ELSIF (clk'EVENT AND clk = '1') THEN
        q <= d;
    END IF;
END PROCESS;

PROCESS (clk, preset)  -- sensitive to clk or preset
BEGIN
    IF preset = '1' THEN
        q <= '1'; -- Async preset
    ELSIF (clk'EVENT AND clk = '1') THEN
        q <= d;
    END IF;
END PROCESS;
```

Synchronous Reset and Preset

```
PROCESS (clk)    -- sensitive to clk
BEGIN
    IF (clk'EVENT AND clk = '1') THEN
        IF reset = '1' THEN
            q <= '0'; -- Sync reset
        ELSE
            q <= d;
        END IF;
    END IF;
END PROCESS;

PROCESS (clk)    -- sensitive to clk
BEGIN
    IF (clk'EVENT AND clk = '1') THEN
        IF preset = '1' THEN
            q <= '1'; -- Sync preset
        ELSE
            q <= d;
        END IF;
    END IF;
END PROCESS;
```

Binary Up Counters

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
ENTITY count8 IS
    PORT (clk : IN std_logic;
          cnt : BUFFER unsigned (7 DOWNTO 0));
END count8;

ARCHITECTURE Model1 OF count8 IS
BEGIN
    PROCESS (clk)
    BEGIN
        IF rising_edge(clk) THEN
            cnt <= cnt + 1;
        END IF;
    END PROCESS;
END Model1 ;
```

Smart Counters

```
ENTITY SmartCounter IS
    PORT (clk, reset, load, enable : IN std_logic;
          data : IN unsigned (7 DOWNTO 0);
          cnt : BUFFER unsigned (7 DOWNTO 0));
END SmartCounter;

ARCHITECTURE Model2 OF SmartCounter IS
BEGIN
    PROCESS (clk, reset)
    BEGIN
        IF reset='1' THEN cnt <= X"00"; -- or (OTHERS => '0')
        ELSIF rising_edge(clk) THEN
            IF load='1' THEN cnt <= data;
            ELSIF enable='1' THEN cnt <= cnt + 1;
            END IF;
        END IF;
    END PROCESS;
END Model2 ;
```

Bidirectional Bus Counter (Entity)

```
-- Counter with bidirectional data bus
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.std_arith.ALL; -- for std_logic arithmetic;

ENTITY BiDirCounter IS
    PORT (clk, reset, load, en : IN std_logic;
          count_io : INOUT std_logic_vector(7 DOWNTO 0));
END BiDirCounter;
```

Bidirectional Bus Counter (Architecture)

```

ARCHITECTURE Model3 OF BiDirCounter IS
    SIGNAL cnt: std_logic_vector (7 DOWNTO 0);
BEGIN
    PROCESS (clk, reset)
    BEGIN
        IF reset='1' THEN
            cnt <= (OTHERS => '0');
        ELSIF rising_edge(clk) THEN
            IF load='1' THEN
                cnt <= count_io;
            ELSIF enable='1' THEN
                cnt <= cnt + 1;
            END IF;
        END IF;
    END PROCESS;

    count_io <= cnt WHEN load='0' ELSE (OTHERS => 'Z');
END Model3;

```

Shift Register

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ShiftReg24 IS
    PORT (
        clk : IN std_logic;
        Pin  : IN std_logic;
        Pout : OUT std_logic);
END ShiftReg24 ;

ARCHITECTURE Model1 OF ShiftReg24 IS
    SIGNAL Reg : std_logic_vector(23 DOWNTO 0);
BEGIN
    PROCESS (clk)
    BEGIN
        IF rising_edge(clk) THEN
            Reg <= Reg(22 DOWNTO 0) & Pin;
        END IF;
    END PROCESS;
    Pout <= Reg(23);
END Model1 ;

```

FIFO

```

library ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY FIFO IS
    PORT (
        CLK : in std_logic;
        Rst  : in std_logic;
        R, W : in std_logic;
        Full : out std_logic;
        Empty: out std_logic;
        Data : inout std_logic_vector(7 downto 0));
END FIFO;

ARCHITECTURE Flaxer OF FIFO IS
    CONSTANT Len: integer := 7;
    TYPE FifoType IS ARRAY(0 TO Len) OF std_logic_vector(7 downto 0);
    SIGNAL FIFO: FifoType;
    SIGNAL Buff: std_logic_vector(7 downto 0);
    SIGNAL Head, Tail: integer RANGE 0 TO Len;

```

FIFO (Architecture)

```

BEGIN
    process(CLK, Rst)
    BEGIN
        IF Rst = '1' THEN
            FOR k in 0 TO Len LOOP
                FIFO(k) <= (OTHERS => '0');
            END LOOP;
            Head <= 0; Tail <= 0;
        ELSIF (CLK'event and CLK = '1') THEN
            IF W = '1' THEN
                FIFO(Head) <= Data;
                Head <= (Head + 1);
            ELSIF R = '1' THEN
                Buff <= FIFO(Tail);
                Tail <= Tail + 1;
            END IF;
        END process;
        Data <= Buff WHEN R = '1' ELSE (OTHERS => 'Z');
        Empty <= '1' WHEN Head = Tail ELSE '0';
        Full <= '1' WHEN (Head-Tail)= -1 ELSE '0';
    END Flaxer;

```

LIFO

```

library ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY LIFO IS
    PORT (
        CLK : in std_logic;
        Rst  : in std_logic;
        R, W : in std_logic;
        Full : out std_logic;
        Empty: out std_logic;
        Data : inout std_logic_vector(7 downto 0));
END LIFO;

ARCHITECTURE Flaxer OF LIFO IS
    CONSTANT Len: integer := 7;
    TYPE LifoType IS ARRAY(0 TO Len) OF std_logic_vector(7 downto 0);
    SIGNAL LIFO: LifoType;
    SIGNAL Buff: std_logic_vector(7 downto 0);
    SIGNAL SP: integer RANGE 0 TO Len;

```

LIFO (Architecture)

```

BEGIN
    process(CLK, Rst)
    VARIABLE TSP: integer RANGE 0 TO Len;
    BEGIN
        IF Rst = '1' THEN
            FOR k in 0 TO Len LOOP
                LIFO(k) <= (OTHERS => '0');
            END LOOP;
            TSP := Len;
        ELSIF (CLK'event and CLK = '1') THEN
            TSP := SP;
            IF W = '1' THEN
                LIFO(TSP) <= Data;
                TSP := (TSP - 1);
            ELSIF R = '1' THEN
                TSP := (TSP + 1);
                Buff <= LIFO(TSP);
            END IF;
        END process;
        SP <= TSP;
        Data <= Buff WHEN R = '1' ELSE (OTHERS => 'Z');
        Empty <= '1' WHEN SP = Len ELSE '0';
        Full <= '1' WHEN SP = 0 ELSE '0';
    END Flaxer;

```