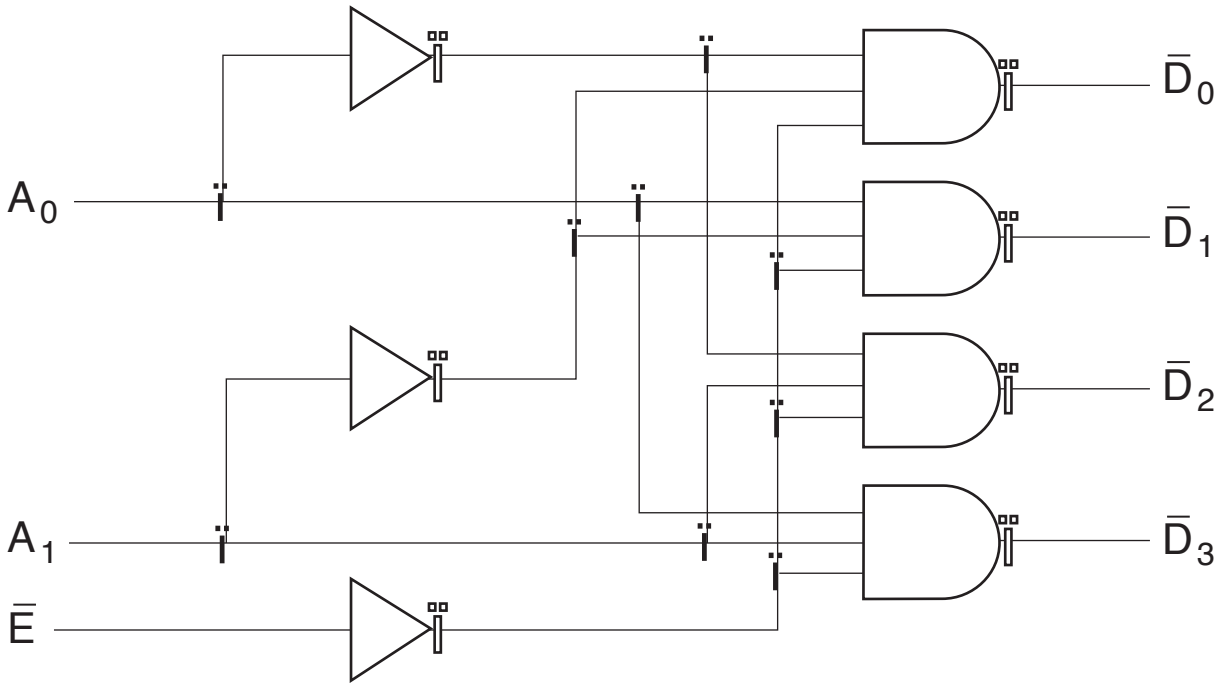
CSE-45208

Introduction to VHDL

Dr. Izadi

Figure 3-14 A 2–to–4-Line Decoder



(a) Logic diagram

$\bar{E}$	A_1	A_0	$\bar{D}_0$	$\bar{D}_1$	$\bar{D}_2$	$\bar{D}_3$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

(b) Truth table

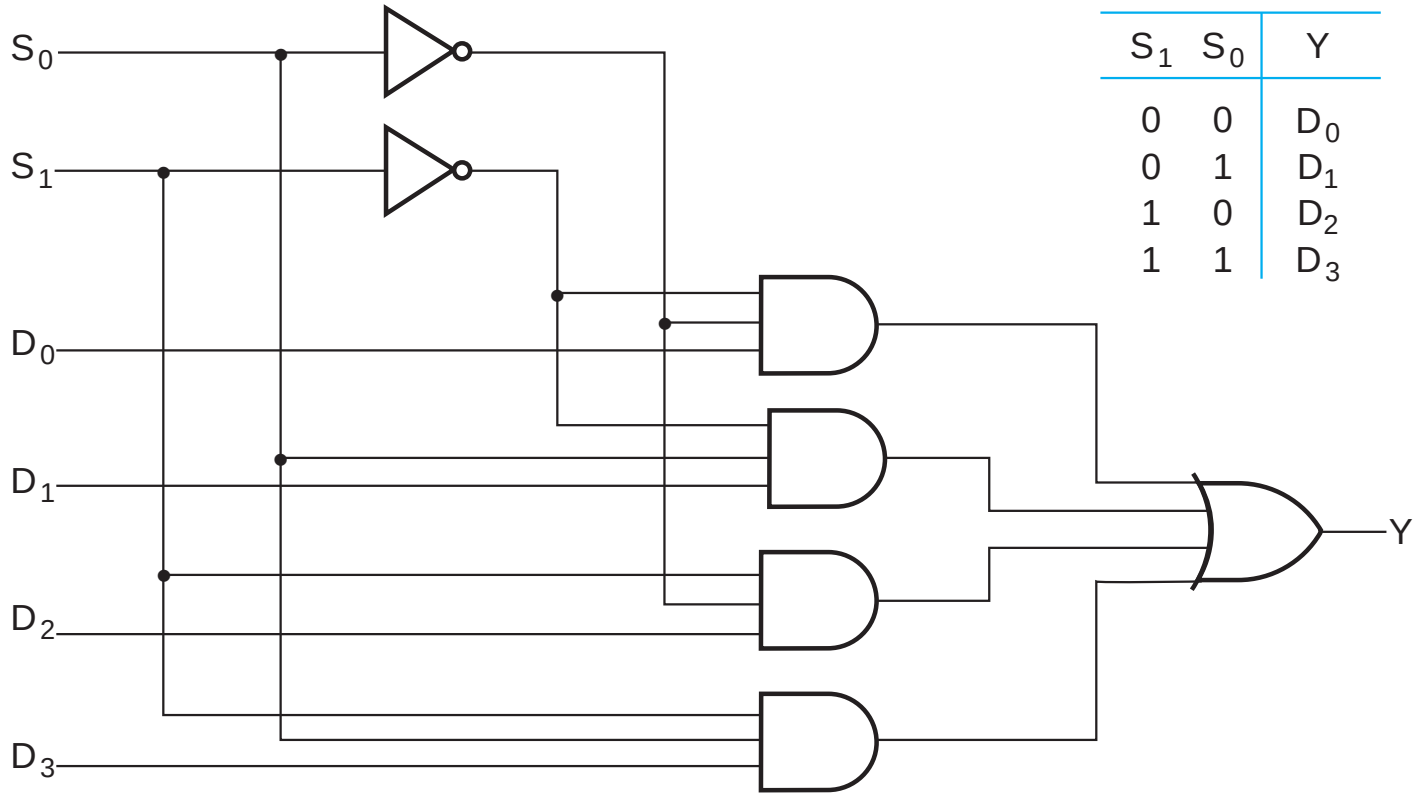
$$\bar{D}_0 = \overline{E\ \bar{A}_1\ \bar{A}_0}$$
$$\bar{D}_1 = \overline{E\ \bar{A}_1\ A_0}$$
$$\bar{D}_2 = \overline{E\ A_1\ \bar{A}_0}$$
$$\bar{D}_3 = \overline{E\ A_1\ A_0}$$

(c) Logic Equations

Figure 3-36 Structural VHDL Description of a 2-to-4 Line Decoder

```
-- 2-to-4 Line Decoder: Structural VHDL Description     -- 1
-- (See Figure 3-14 for logic diagram)                  -- 2
library ieee, lcdf_vhdl;                                -- 3
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;  -- 4
entity decoder_2_to_4 is                                -- 5
  port(E_n, A0, A1: in std_logic;                       -- 6
      D0_n, D1_n, D2_n, D3_n: out std_logic);           -- 7
end decoder_2_to_4;                                     -- 8
                                                        -- 9
architecture structural_1 of decoder_2_to_4 is          --10
  component NOT1                                        --11
    port(in1: in std_logic;                             --12
        out1: out std_logic);                           --13
  end component;                                        --14
  component NAND3                                       --15
    port(in1, in2, in3: in std_logic;                   --16
        out1: out std_logic);                           --17
  end component;                                        --18
  signal E, A0_n, A1_n: std_logic;                      --19
  begin                                                 --20
    g0: NOT1 port map (in1 => A0, out1 => A0_n);        --21
    g1: NOT1 port map (in1 => A1, out1 => A1_n);        --22
    g2: NOT1 port map (in1 => E_n, out1 => E);          --23
    g3: NAND3 port map (in1 => A0_n, in2 => A1_n,       --24
            in3 => E, out1 => D0_n);                    --25
    g4: NAND3 port map (in1 => A0, in2 => A1_n,         --26
            in3 => E, out1 => D1_n);                    --27
    g5: NAND3 port map (in1 => A0_n, in2 => A1,         --28
            in3 => E, out1 => D2_n);                    --29
    g6: NAND3 port map (in1 => A0, in2 => A1,           --30
            in3 => E, out1 => D3_n);                    --31
end structural_1;                                       --32
```

Figure 3-19 4–to–1-Line Multiplexer



Function table

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Figure 3-37 Structural VHDL Description of a 4-to-1 Line Multiplexer

```
-- 4-to-1 Line Multiplexer: Structural VHDL Description -- 1
-- (See Figure 3-19 for logic diagram)                  -- 2
library ieee, lcdf_vhdl;                                -- 3
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;  -- 4
entity multiplexer_4_to_1_st is                         -- 5
  port(S: in std_logic_vector(0 to 1);                  -- 6
       D: in std_logic_vector(0 to 3);                  -- 7
       Y: out std_logic);                               -- 8
end multiplexer_4_to_1_st;                              -- 9
                                                        --10
architecture structural_2 of multiplexer_4_to_1_st is   --11
  component NOT1                                        --12
    port(in1: in std_logic;                             --13
         out1: out std_logic);                          --14
  end component;                                        --15
  component AND3                                        --16
    port(in1, in2, in3: in std_logic;                   --17
         out1: out std_logic);                          --18
  end component;                                        --19
  component OR4                                         --20
    port(in1, in2, in3, in4: in std_logic;              --21
         out1: out std_logic);                          --22
  end component;                                        --23
  signal S_n: std_logic_vector(0 to 1);                 --24
  signal N: std_logic_vector(0 to 3);                   --25
  begin                                                 --26
    g0: NOT1 port map (S(0), S_n(0));                   --27
    g1: NOT1 port map (S(1), S_n(1));                   --28
    g2: AND3 port map (S_n(1), S_n(0), D0), N(0));      --29
    g3: AND3 port map (S_n(1), S(0), D(1), N(1));       --30
    g4: AND3 port map (S(1), S_n(0), D(2), N(2));       --31
    g5: AND3 port map (S(1), S(0), D(3), N(3));         --32
    g6: OR4 port map (N(0), N(1), N(2), N(3), Y);       --33
end structural_2;                                       --34
```

Figure 3-38 Dataflow VHDL Description of a 2-to-4 Line Decoder

```
-- 2-to-4 Line Decoder: Dataflow VHDL Description       -- 1
-- (See Figure 3-14 for logic diagram)                  -- 2
Use library, use, and entity entries from 2_to_4_decoder_st;
--3
                                                        -- 4
architecture dataflow_1 of decoder_2_to_4 is            -- 5
                                                        -- 6
signal A0_n, A1_n: std_logic;                           -- 7
begin                                                   -- 8
  A0_n <= not A0;                                       -- 9
  A1_n <= not A1;                                       --10
  E <= not E_n;                                         --11
  D0_n <= not ( A0_n and A1_n and E);                   --12
  D1_n <= not ( A0 and A1_n and E);                     --13
  D2_n <= not ( A0_n and A1 and E);                     --14
  D3_n <= not ( A0 and A1 and E);                       --15
```

Figure 3-39 Conditional Dataflow VHDL Description of a 4-to-1 Line Multiplexer Using When-Else

```
-- 4-to-1 Line Mux: Conditional Dataflow VHDL Description--1
-- Using When-Else (See Figure 3-19 for function table) -- 2
library ieee;                                             -- 3
use ieee.std_logic_1164.all;                              -- 4
entity multiplexer_4_to_1_we is                           -- 5
  port (S : in std_logic_vector(1 downto 0);              -- 6
        D : in std_logic_vector(0 to 3);                  -- 7
        Y : out std_logic);                               -- 8
end multiplexer_4_to_1_we;                                -- 9
                                                          --10
architecture function_table of multiplexer_4_to_1_we is --11
begin                                                     --12
  Y <=D(0) when S = "00" else                             --13
      D(1) when S = "01" else                             --14
      D(2) when S = "10" else                             --15
      D(3) when S = "11" else                             --16
      'X';                                                --17
end function_table;                                       --18
```

3-49 Figure 3-40 Conditional Dataflow VHDL Description of a 4-to-1 Line Multiplexer Using With-Select

```
--4-to-1 Line Mux: Conditional Dataflow VHDL Description-- 1
Using When-Else (See Figure 3-14 for logic equations)   -- 2
library ieee;                                           -- 3
use ieee.std_logic_1164.all;                            -- 4
entity multiplexer_4_to_1_ws is                         -- 5
  port (S : in std_logic_vector(1 downto 0);            -- 6
        D : in std_logic_vector(0 to 3);                -- 7
        Y : out std_logic);                             -- 8
end multiplexer_4_to_1_ws;                              -- 9
                                                        --10
architecture function_table_ws of multiplexer_4_to_1_ws is--
11
begin                                                   --12
  with S select                                         --13
    Y <=D(0) when "00",                                 --14
        D(1) when "01",                                 --15
        D(2) when "10",                                 --16
        D(3) when "11",                                 --17
        'X' when others;                                --18
```

Figure 3-41 Hierarchical Structural/Dataflow Description of a 4-Bit Adder

```
-- 4-bit Adder: Hierarchical Dataflow/Structural
-- (See Figures 3-27 and 3-28 for logic diagrams)
library ieee;
use ieee.std_logic_1164.all;
entity half_adder is
  port (x, y : in std_logic;
        s, c : out std_logic);
end half_adder;

architecture dataflow_3 of half_adder is
  begin
    s <= x xor y;
    c <= x and y;
end dataflow_3;

library ieee;
use ieee.std_logic_1164.all;
entity full_adder is
  port (x, y, z : in std_logic;
        s, c : out std_logic);
end full_adder;

architecture struc_dataflow_3 of full_adder is
  component half_adder
    port(x, y : in std_logic;
         s, c : out std_logic);
  end component;
  signal hs, hc, tc: std_logic;
  begin
    HA1: half_adder
      port map (x, y, hs, hc);
    HA2: half_adder
      port map (hs, z, s, tc);
    c <= tc or hc;
end struc_dataflow_3;

library ieee;
use ieee.std_logic_1164.all;
entity adder_4 is
  port(B, A : in std_logic_vector(3 downto 0);
       C0 : in std_logic;
       S : out std_logic_vector(3 downto 0);
       C4: out std_logic);
end adder_4;
```

Figure 3-42 Hierarchical Structural/Dataflow Description of a 4-Bit Adder (Continued)

```
architecture structural_4 of adder_4 is
  component full_adder
    port(x, y, z : in std_logic;
         s, c : out std_logic);
  end component;
  signal C: std_logic_vector(4 downto 0);
  begin
    Bit0: full_adder
      port map (B(0), A(0), C(0), S(0), C(1));
    Bit1: full_adder
      port map (B(1), A(1), C(1), S(1), C(2));
    Bit2: full_adder
      port map (B(2), A(2), C(2), S(2), C(3));
    Bit3: full_adder
      port map (B(3), A(3), C(3), S(3), C(4));
    C(0) <= C0;
    C4 <= C(4);
end structural_4;
```

Figure 3-43 Behavioral Description of a 4-Bit Adder

```
architecture structural_4 of adder_4 is
  component full_adder
    port(x, y, z : in std_logic;
         s, c : out std_logic);
  end component;
  signal C: std_logic_vector(4 downto 0);
  begin
    Bit0: full_adder
      port map (B(0), A(0), C(0), S(0), C(1));
    Bit1: full_adder
      port map (B(1), A(1), C(1), S(1), C(2));
    Bit2: full_adder
      port map (B(2), A(2), C(2), S(2), C(3));
    Bit3: full_adder
      port map (B(3), A(3), C(3), S(3), C(4));
    C(0) <= C0;
    C4 <= C(4);
end structural_4;
```