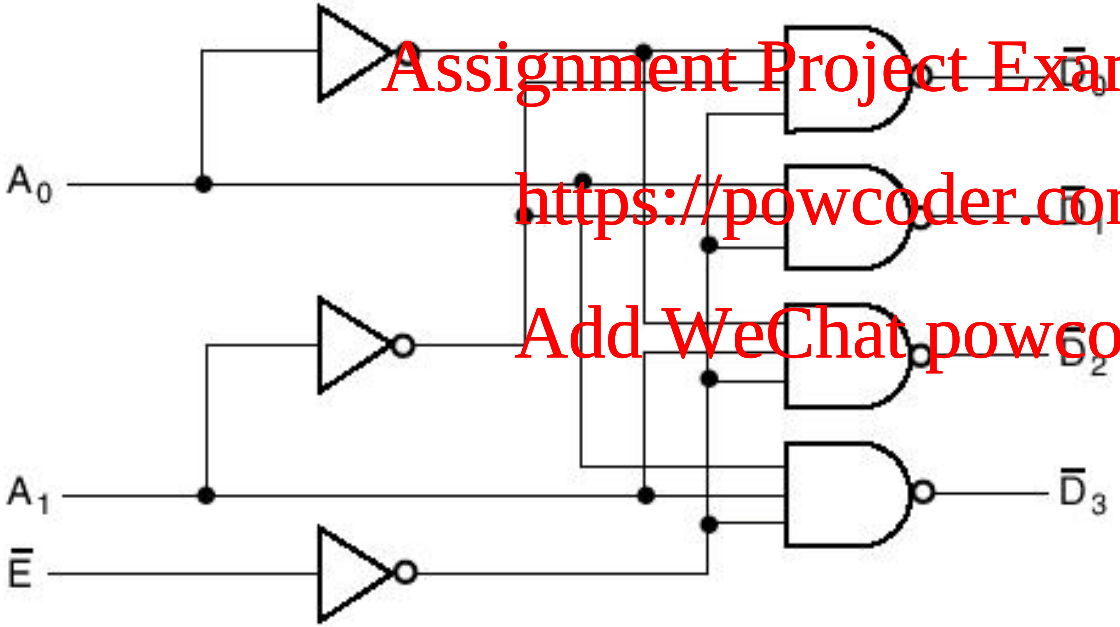
2 to 4 Line Decoder

Structural



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

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

2 to 4 Line Decoder (Page 1)

Structural

```
-- 2-to-4 Line Decoder: Structural VHDL Description
library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;
entity decoder_2_to_4 is
  port(E, A0, A1: in std_logic;
       D0, D1, D2, D3: out std_logic);
end decoder_2_to_4;

architecture structural_1 of decoder_2_to_4 is

component NOT1
  port(in1: in std_logic;
       out1: out std_logic);
end component;
```

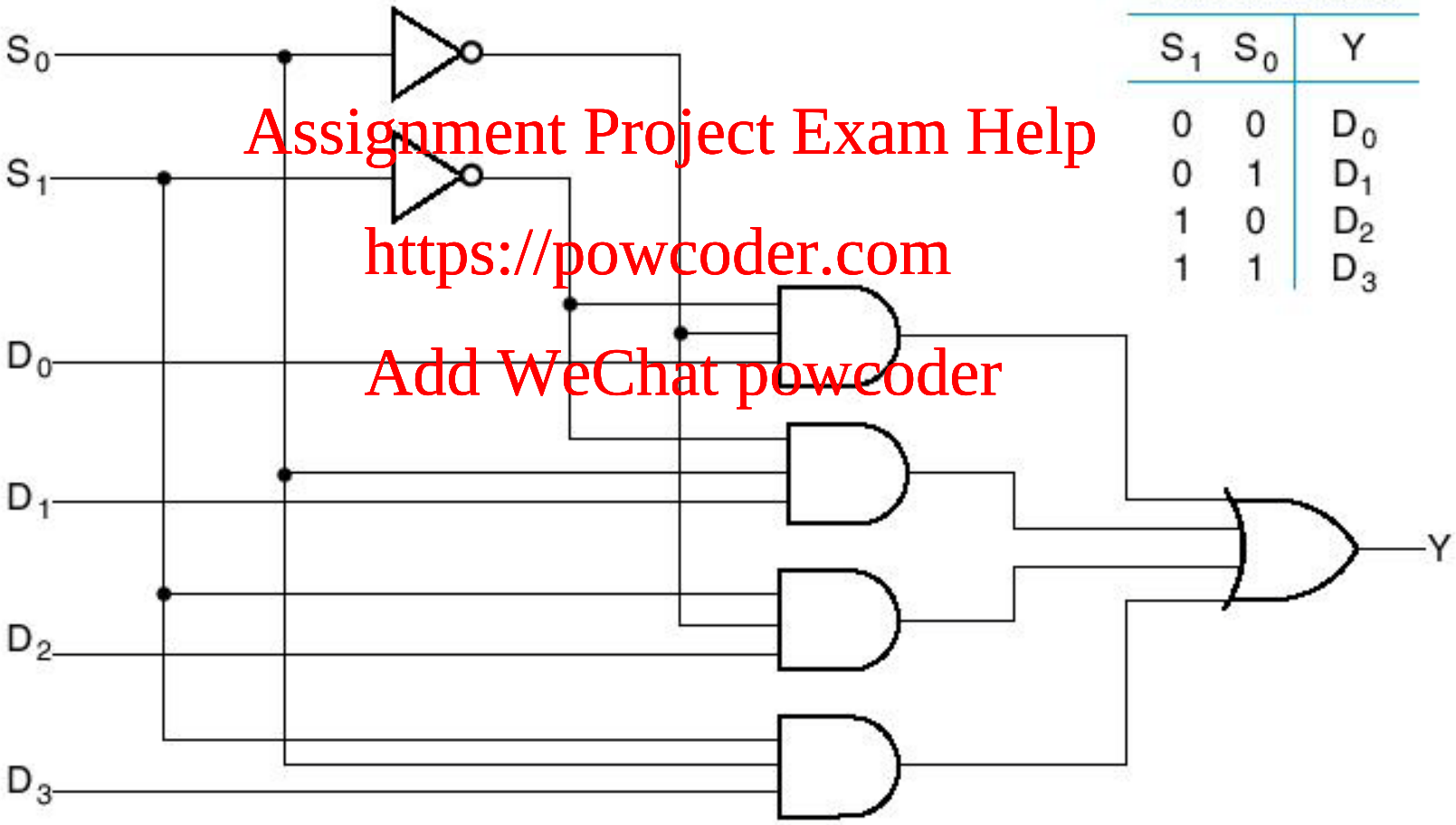
2 to 4 Line Decoder (Page 3)

Structural

```
component NAND3
   port(in1, in2, in3: in std_logic;
        out1: out std_logic);
end component;

signal not_A0, not_A1: std_logic;
begin
   g0: NOT1 port map (in1 => A0, out1 => not_A0);
   g1: NOT1 port map (in1 => A1, out1 => not_A1);
   g2: NAND3 port map (in1 => not_A0, in2 => not_A1,
                       in3 => E, out1 => D0);
   g3: NAND3 port map (in1 => A0, in2 => not_A1,
                       in3 => E, out1 => D1);
   g4: NAND3 port map (in1 => not_A0, in2 => A1,
                       in3 => E, out1 => D2);
   g5: NAND3 port map (in1 => A0, in2 => A1,
                       in3 => E, out1 => D3);
end structural_1;
```

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

4 to 1 Line Multiplexer (Page 1)

```
-- 4-to-1 Line Multiplexer: Structural VHDL Description
library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;
entity multiplexer_4_to_1_st is
  port(S: in std_logic_vector(0 to 1);
       D: in std_logic_vector(0 to 3);
       Y: out std_logic);
end multiplexer_4_to_1_st;

architecture structural_2 of multiplexer_4_to_1_st is
component NOT1
  port(in1: in std_logic;
       out1: out std_logic);
end component;
component AND3
  port(in1, in2, in3: in std_logic;
       out1: out std_logic);
end component;
```

4 to 1 Line Multiplexer (Page 2)

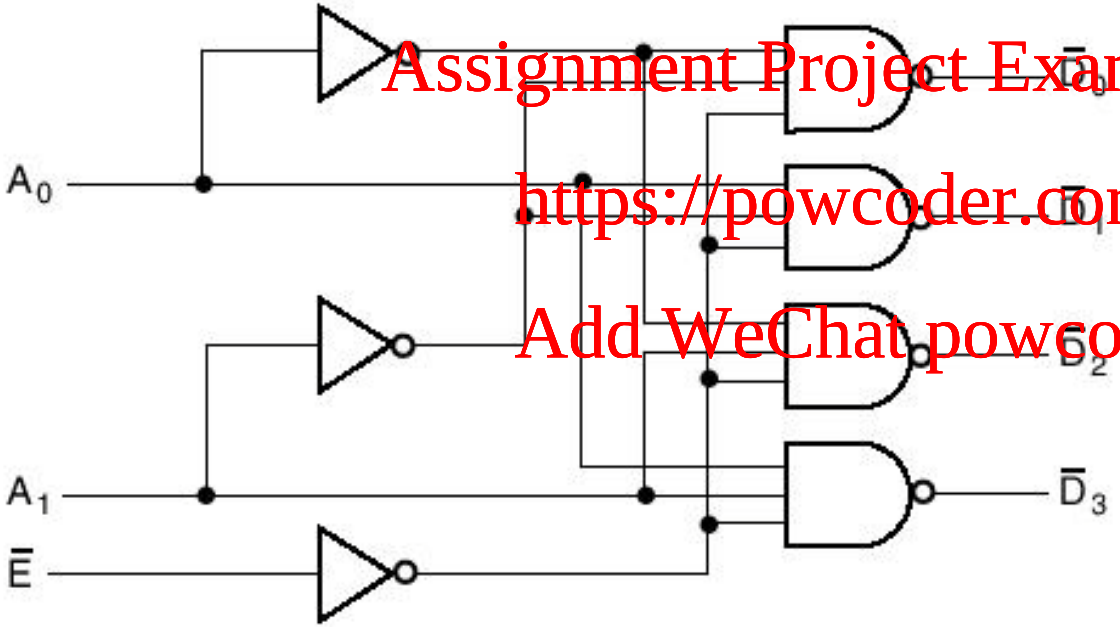
```
component OR4
   port(in1, in2, in3, in4: in std_logic;
        out1: out std_logic);
end component;

signal not_S: std_logic_vector(0 to 1);
signal N: std_logic_vector(0 to 3);
begin
   g0: NOT1 port map (S(0), not_S(0));
   g1: NOT1 port map (S(1), not_S(1));
   g2: AND3 port map (not_S(1), not_S(0), D(0), N(0));
   g3: AND3 port map (not_S(1), S(0), D(1), N(1));
   g4: AND3 port map (S(1), not_S(0), D(2), N(2));
   g5: AND3 port map (S(1), S(0), D(3), N(3));
   g6: OR4 port map (N(0), N(1), N(2), N(3), Y);
end structural_2;
```

2 to 4 Line Decoder

Dataflow

Same as slide 1



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

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

2 to 4 Line Decoder(Page 1)

Dataflow

```
-- 2-to-4 Line Decoder: Dataflow VHDL Description
library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;
entity decoder_2_to_4 is
  port(E, A0, A1: in std_logic;
       D0, D1, D2, D3: out std_logic);
end decoder_2_to_4;

architecture dataflow_1 of decoder_2_to_4 is

signal not_A0, not_A1: std_logic;
begin
  not_A0 <= not A0;
  not_A1 <= not A1;
  D0 <= not ( not_A0 and not_A1 and E);
  D1 <= not ( A0 and not_A1 and E);
  D2 <= not ( not_A0 and A1 and E);
  D3 <= not ( A0 and A1 and E);
end dataflow_1;
```

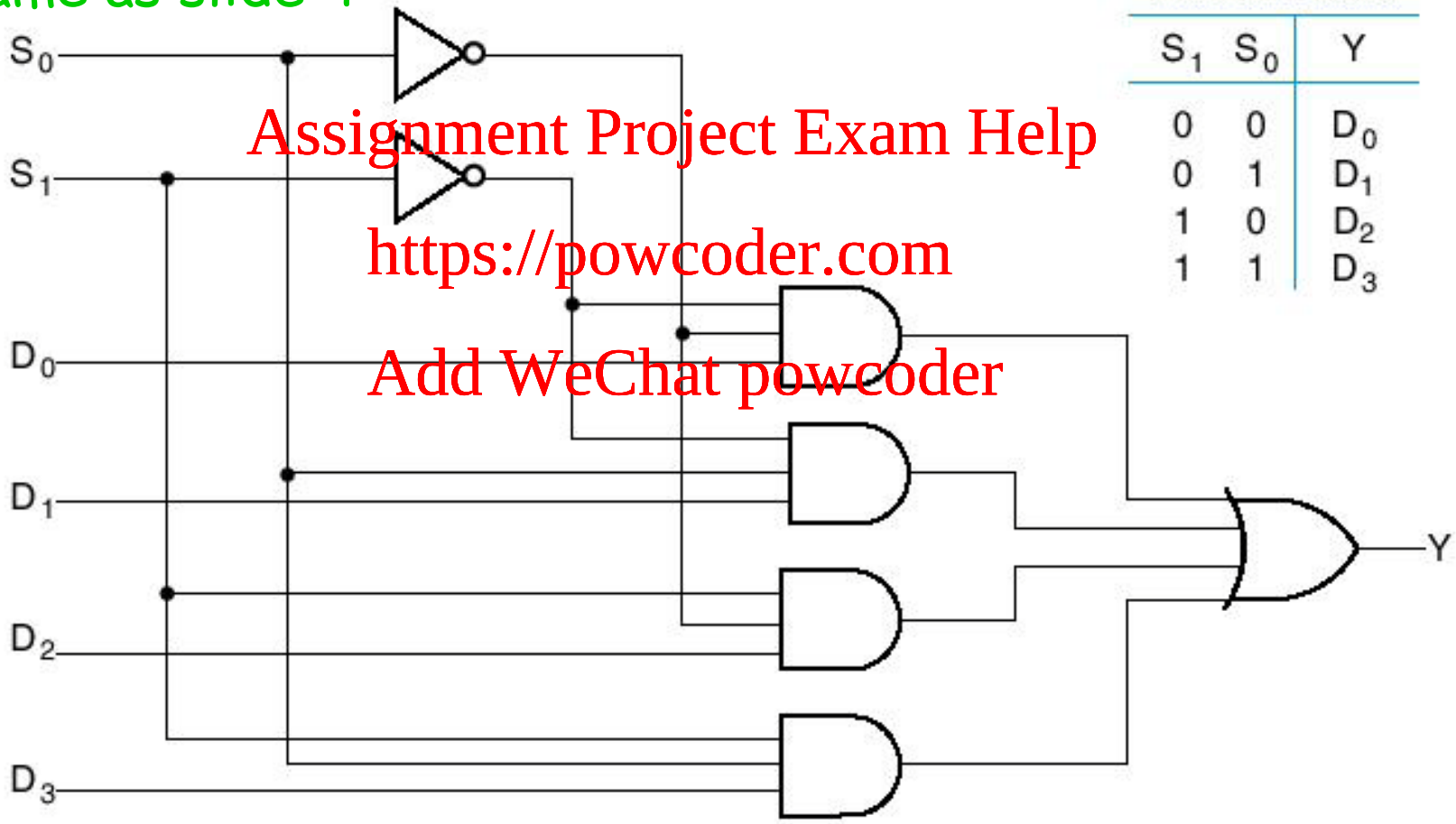
4 to 1 Line Multiplexer

Conditional Dataflow (When-Else)

Same as slide 4

Function table

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



4 to 1 Line Multiplexer (Page 1)

Conditional Dataflow (When-Else)

```
library ieee;
use ieee.std_logic_1164.all;
entity multiplexer_4_to_1_we is
    port (S : in std_logic_vector(1 downto 0);
          D : in std_logic_vector(3 downto 0);
          Y : out std_logic);
end multiplexer_4_to_1_we;

architecture function_table of multiplexer_4_to_1_we is
begin
    Y <= D(0) when S = "00" else
         D(1) when S = "01" else
         D(2) when S = "10" else
         D(3) when S = "11" else
         'X';
end function_table;
```

4 to 1 Line Multiplexer (Page 1)

Conditional Dataflow (With-Select)

```
library ieee;
use ieee.std_logic_1164.all;
entity multiplexer_4_to_1_ws is
    port (S : in std_logic_vector(1 downto 0);
          D : in std_logic_vector(3 downto 0);
          Y : out std_logic);
end multiplexer_4_to_1_ws;

architecture function_table_ws of multiplexer_4_to_1_ws is
begin
    with S select
    Y <= D(0) when "00",
         D(1) when "01",
         D(2) when "10",
         D(3) when "11",
         'X' when others;
end function_table_ws;
```