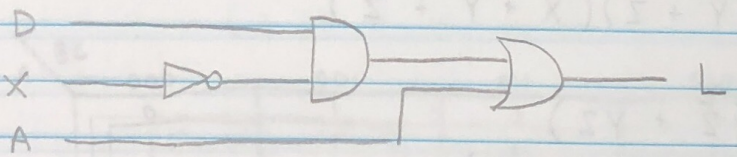
Stewart Dulaney
CS 42
Section 4105
SID: 1545566

Chapter 2 Problems

2-1 Boolean Function Example - Power Windows

$$L(D, X, A) = D\overline{X} + A$$

D	X	A	L
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



2-2 Complementing Functions

Find the complement of $F_1 = \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z$ and $F_2 = X(\bar{Y}\bar{Z} + YZ)$

$$\bar{F}_1 = \overline{\bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z}$$
$$= \overline{\bar{X}Y\bar{Z}} \cdot \overline{\bar{X}\bar{Y}Z}$$
$$= (X + \bar{Y} + Z)(X + Y + \bar{Z})$$

$$\bar{F}_2 = \overline{X(\bar{Y}\bar{Z} + YZ)}$$
$$= \bar{X} + \overline{(\bar{Y}\bar{Z} + YZ)}$$
$$= \bar{X} + (\overline{\bar{Y}\bar{Z}})(\overline{YZ})$$
$$= \bar{X} + (Y + Z)(\bar{Y} + \bar{Z})$$

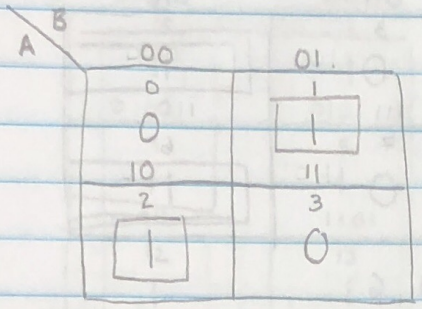
2-3 Complementing Functions by Using Duals

$$F_1 = \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z$$
$$= (\bar{X}Y\bar{Z}) + (\bar{X}\bar{Y}Z)$$
$$F_1 \text{ Dual} = (\bar{X} + Y + \bar{Z})(\bar{X} + \bar{Y} + Z)$$
$$\bar{F}_1 = (X + \bar{Y} + Z)(X + Y + \bar{Z})$$

$$F_2 = X(\bar{Y}\bar{Z} + YZ)$$
$$= X((\bar{Y}\bar{Z}) + (YZ))$$
$$F_2 \text{ Dual} = X + (\bar{Y} + \bar{Z})(Y + Z)$$
$$\bar{F}_2 = \bar{X} + (Y + Z)(\bar{Y} + \bar{Z})$$

2-4 Another 2-Variable Map Example

$G(A, B) = \Sigma m(1, 2)$



1 2

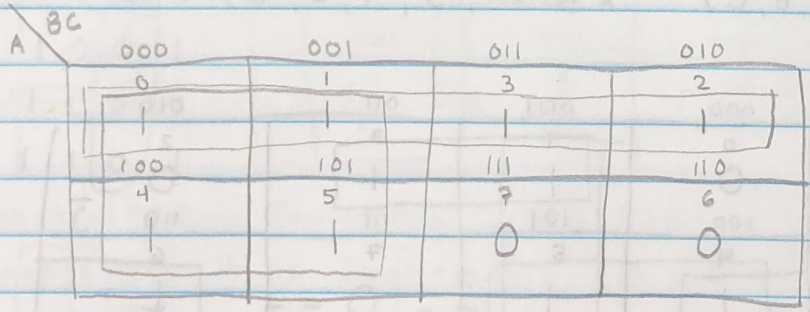
01 10

$\bar{A}B$ $A\bar{B}$

$G(A, B) = \bar{A}B + A\bar{B}$

2-5 Three-Variable Map Simplification 1

$F(A, B, C) = \Sigma m(0, 1, 2, 3, 4, 5)$



0, 1, 3, 2

0 0 0

0 0 1

0 1 1

0 1 0

$\bar{A}$

0, 1, 4, 5

0 0 0

0 0 1

1 0 0

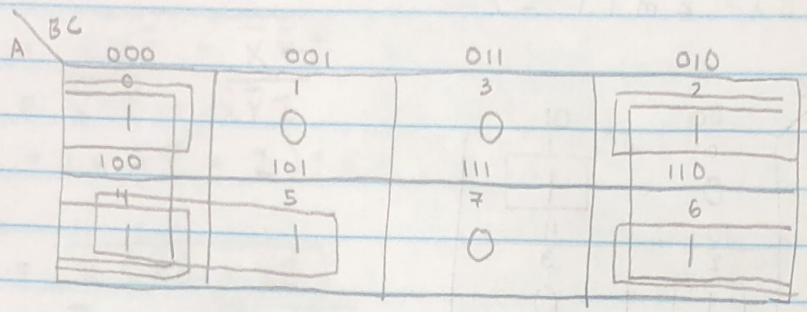
1 0 1

$\bar{B}$

$F = \bar{A} + \bar{B}$

2-6 Three-Variable Map Simplification 2

$G(A,B,C) = \Sigma m(0, 2, 4, 5, 6)$



4, 5

1 0 0
1 0 1

$A\bar{B}$

0, 2, 4, 6

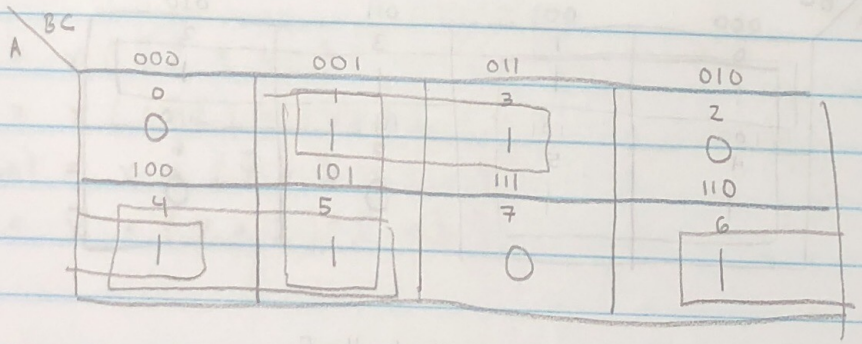
0 0 0
0 1 0
1 0 0
1 1 0

$\bar{C}$

$G(A,B,C) = A\bar{B} + \bar{C}$

2-7 Three-Variable Map Simplification 3

$H(A,B,C) = \Sigma m(1, 3, 4, 5, 6)$



1, 3

0 0 1
0 1 1

$\bar{A}C$

4, 5

1 0 0
1 0 1

$A\bar{B}$

4, 6

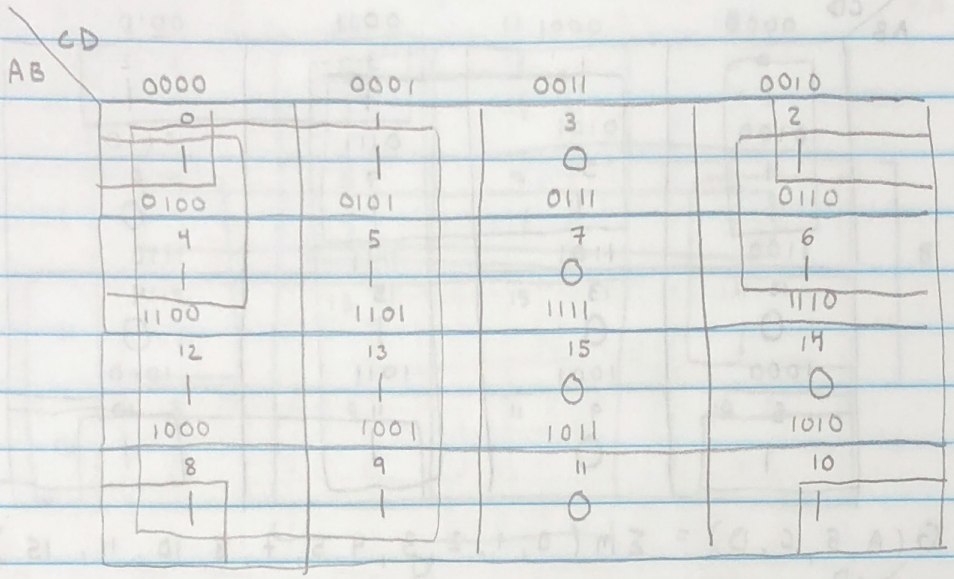
1 0 0
1 1 0

$A\bar{C}$

$H = \bar{A}C + A\bar{B} + A\bar{C}$

2-8 Four-Variable Map Simplification 1

$F(A,B,C,D) = \Sigma m(0,1,2,4,5,6,8,9,10,12,13)$



0, 1, 4, 5, 8, 9, 12, 13

0	0	0	0
0	0	0	1
0	1	0	0
0	1	0	1
1	0	0	0
1	0	0	1
1	1	0	0
1	1	0	1

$\bar{C}$

0, 2, 4, 6

0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0

$\bar{A}\bar{D}$

0, 2, 8, 10

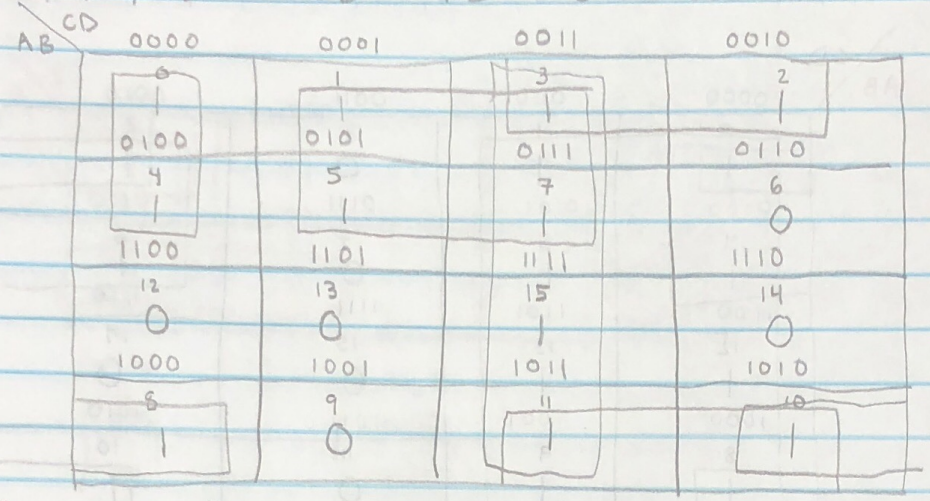
0	0	0	0
0	0	1	0
1	0	0	0
1	0	1	0

$\bar{B}\bar{D}$

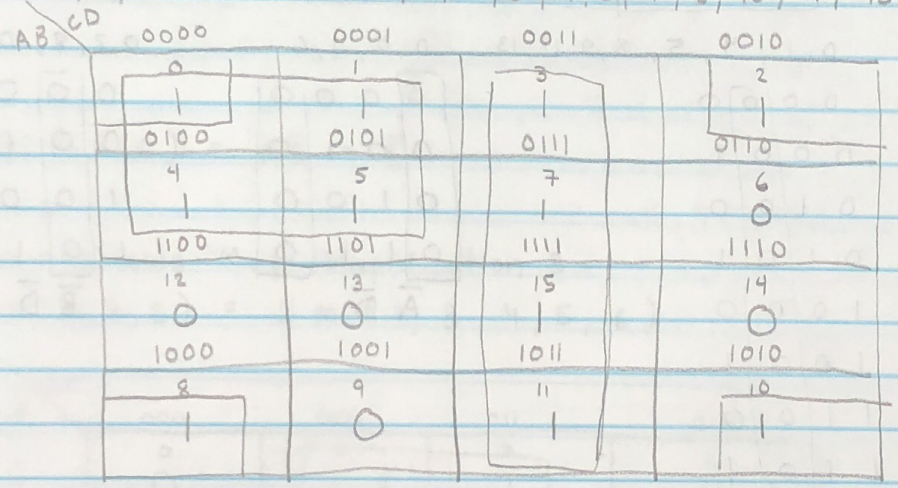
$$F = \bar{C} + \bar{A}\bar{D} + \bar{B}\bar{D}$$

2-9 Four-Variable Map Simplification 2

$G(A,B,C,D) = \bar{A}\bar{C}\bar{D} + \bar{A}D + \bar{B}C + CD + A\bar{B}\bar{D}$



$G(A,B,C,D) = \Sigma m(0,1,2,3,4,5,7,8,10,11,15)$



0, 2, 8, 10

0	0	0	0
0	0	1	0
1	0	0	0
1	0	1	0

$\bar{B}\bar{D}$

0, 1, 4, 5

0	0	0	0
0	0	0	1
0	1	0	0
0	1	0	1

$\bar{A}\bar{C}$

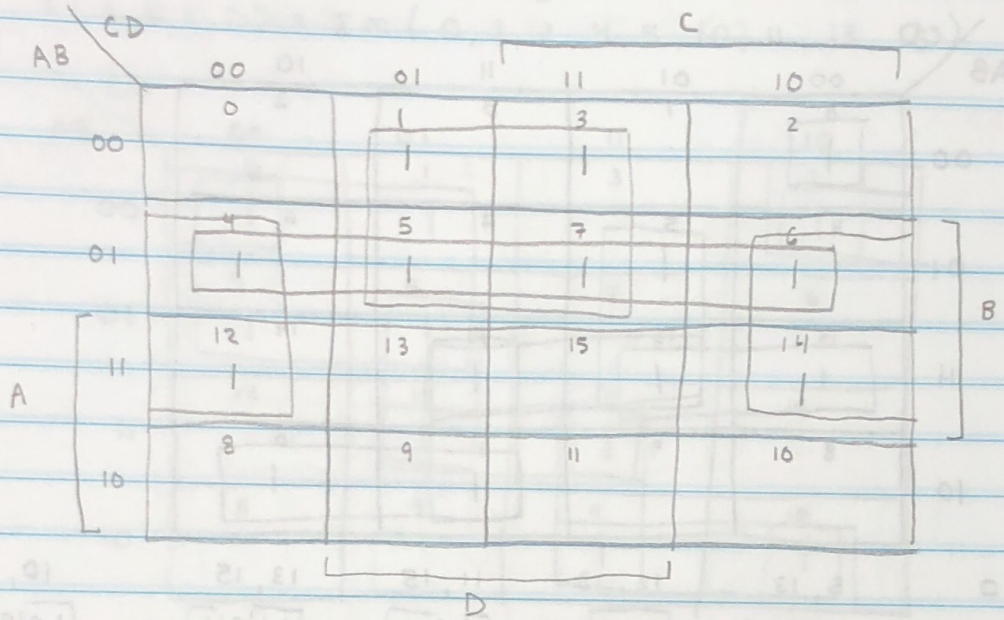
3, 7, 11, 15

0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	1

CD

$G = \bar{B}\bar{D} + \bar{A}\bar{C} + CD$

2-10 Simplification Using Prime Implicants



1, 3, 5, 7

0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1

$\bar{A}D$

4, 5, 6, 7

0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

$\bar{A}B$

4, 6, 12, 14

0	1	0	0
0	1	1	0
1	1	0	0
1	1	1	0

$B\bar{D}$

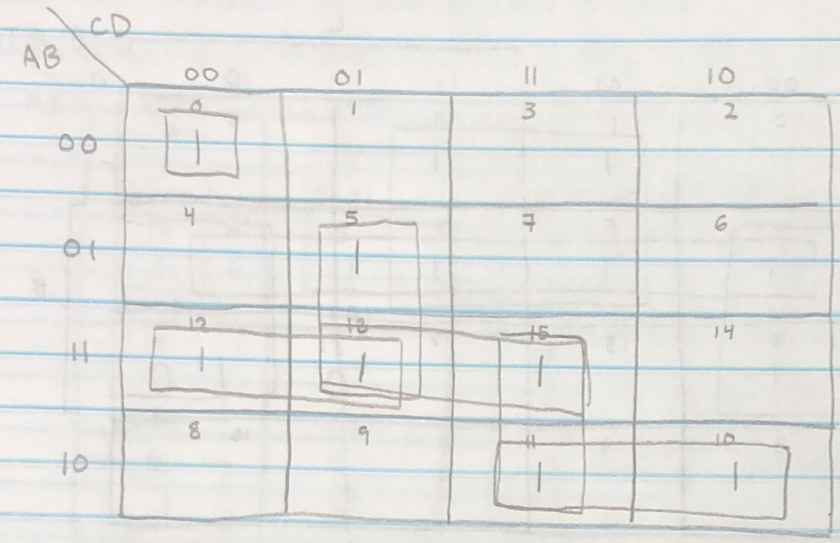
Prime Implicants : $\bar{A}D$, $\bar{A}B$, $B\bar{D}$

Essential Prime Implicants : $\bar{A}D$, $B\bar{D}$

Nonessential Prime Implicants : $\bar{A}B$

$F = \bar{A}D + B\bar{D}$

2-11 Simplification Via Essential and Nonessential Prime Implicants



0	5, 13	12, 13	11, 15	13, 15	10, 11
0000	0101	1100	1011	1101	1010
$\bar{A}\bar{B}\bar{C}\bar{D}$	1101	1101	1111	1111	1011
	$B\bar{C}D$	$AB\bar{C}$	ACD	ABD	$A\bar{B}C$

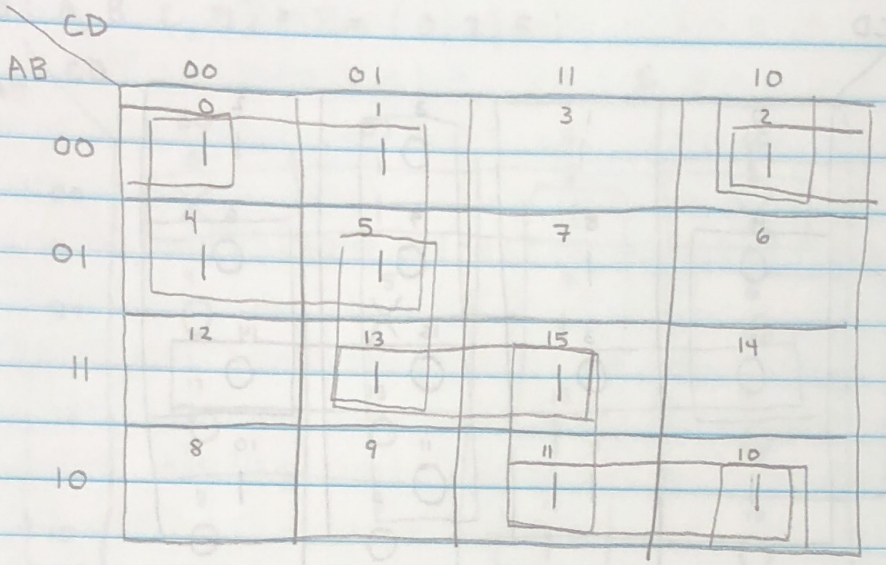
Prime Implicants: $\bar{A}\bar{B}\bar{C}\bar{D}$, $B\bar{C}D$, $AB\bar{C}$, $\underbrace{ACD, ABD}_{\text{nonessential}}$, $A\bar{B}C$

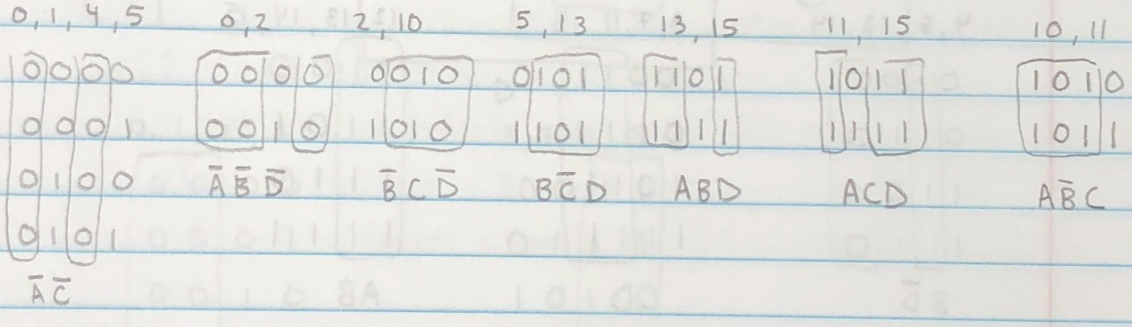
Essential Prime Implicants: $\bar{A}\bar{B}\bar{C}\bar{D}$, $B\bar{C}D$, $AB\bar{C}$, $A\bar{B}C$

$$F = \bar{A}\bar{B}\bar{C}\bar{D} + B\bar{C}D + AB\bar{C} + A\bar{B}C + \begin{cases} ACD \\ \text{or} \\ ABD \end{cases}$$

2-12 Simplifying a Function Using the Selection Rule

$F(A,B,C,D) = \Sigma m(0,1,2,4,5,10,11,13,15)$





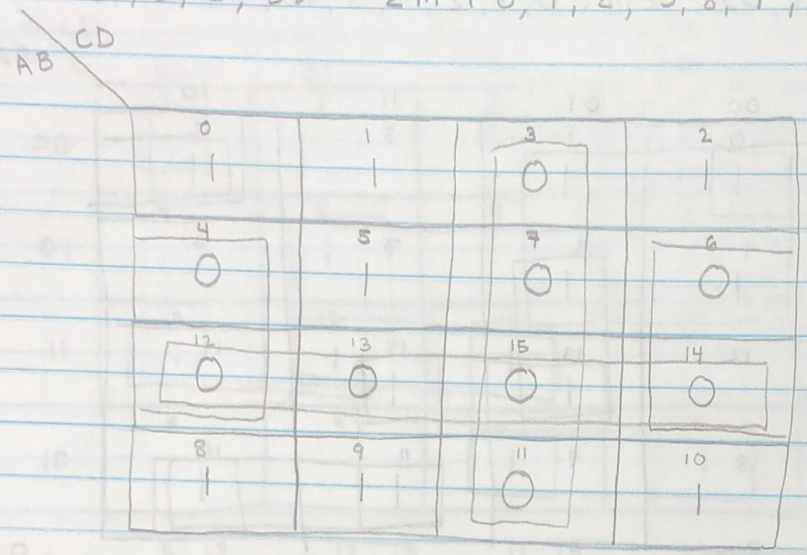
Prime Implicants: $\bar{A}\bar{C}$, $\bar{A}\bar{B}\bar{D}$, $\bar{B}C\bar{D}$, $B\bar{C}D$, ABD, ACD, $A\bar{B}C$

Essential Prime Implicants: $\bar{A}\bar{C}$

$F = \bar{A}\bar{C} + ABD + A\bar{B}C + \bar{A}\bar{B}\bar{D}$

2-13 Simplifying a Product-of-Sums Form

$F(A, B, C, D) = \Sigma m(0, 1, 2, 5, 8, 9, 10)$



4, 6, 12, 14

0	1	0	0
0	1	1	0
1	1	0	0
1	1	1	0

$B\bar{D}$

3, 7, 11, 15

0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	1

CD

12, 13, 14, 15

1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

AB

$\bar{F} = AB + CD + B\bar{D}$

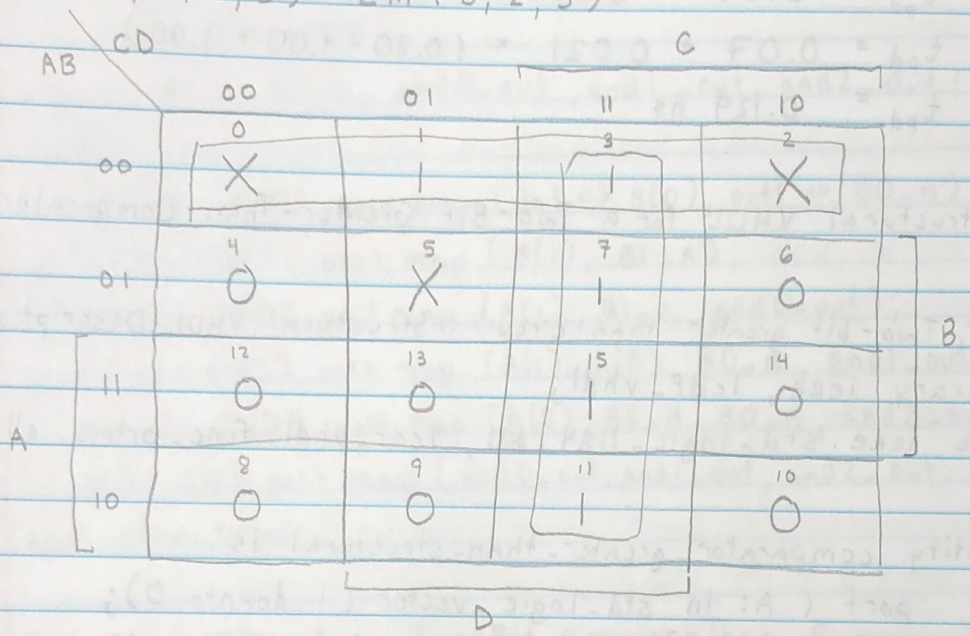
$\bar{F}_{Dual} = (A + B)(C + D)(B + \bar{D})$

$F = (\bar{A} + \bar{B})(\bar{C} + \bar{D})(\bar{B} + D)$

2-14 Simplification with Don't-Care Conditions

$F(A, B, C, D) = \Sigma m(1, 3, 7, 11, 15)$

$d(A, B, C, D) = \Sigma m(0, 2, 5)$



0, 1, 2, 3

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1

$\bar{A}\,\bar{B}$

3, 7, 11, 15

0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	1

CD

$F = CD + \bar{A}\,\bar{B}$

2-15 Calculation of Gate Delay Based on Fan-Out

$t_{pd} = 0.07 + 0.021 \times SL$ ns

$t_{pd} = 0.07 + 0.021 \times (0.80 + 1.00 + 1.00)$

$t_{pd} = 0.129$ ns

2-16 Structural VHDL for a Two-Bit Greater-Than Comparator Circuit

```
-- Two-bit greater-than circuit: Structural VHDL Description
library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;

entity comparator_greater_than_structural is
    port ( A: in std_logic_vector (1 downto 0);
           B: in std_logic_vector (1 downto 0);
           A_greater_than_B : out std_logic );
end comparator_greater_than_structural;

architecture structural of comparator_greater_than_structural is

    component NOT1
        port ( in1: in std_logic;
               out1 : out std_logic);
    end component;
    component AND2
        port ( in1, in2: in std_logic;
               out1: out std_logic);
    end component;
    component AND3
        port ( in1, in2, in3 : In std_logic;
               out1: out std_logic);
    end component;
```

```
    component OR3
        port ( in1, in2, in3 : in std_logic;
               out1 : out std_logic );
    end component;
signal B1_n, B0_n, and0_out, and1_out, and2_out : std_logic;
begin
    inv_0: NOT1 port map ( in1 => B(0), out1 => B0_n );
    inv_1: NOT1 port map ( B(1), B1_n );
    and_0: AND2 port map (A(1), B1_n, and0_out);
    and_1: AND3 port map (A(1), A(0), B0_n, and1_out);
    and_2: AND3 port map (A(0), B1_n, B0_n, and2_out);
    or0: OR3 port map ( and0_out, and1_out, and2_out, A_greater_than_B);
end structural;
```

2-17 Dataflow VHDL for a Two-Bit Greater-Than Comparator Circuit

```
-- Two-bit greater-than circuit: Dataflow VHDL Description
library ieee;
use ieee.std_logic_1164.all;

entity comparator_greater_than_dataflow is
    port ( A: in std_logic_vector (1 downto 0);
           B: in std_logic vector (1 downto 0);
           A_greater_than_B: out std_logic);
end comparator_greater_than_dataflow;

architecture dataflow of comparator_greater_than_dataflow is
signal B1_n, B0_n, and0_out, and1_out, and2_out: std_logic;
begin
    B1_n <= not B(1);
    B0_n <= not B(0);
    and0_out <= A(1) and B1_n;
```

```
    and1_out <= A(1) and A(0) and B0_n;
    and2_out <= A(0) and B1_n and B0_n;
    A_greater_than_B <= and0_out or and1_out or and2_out;
end dataflow;
```

2-18 VHDL for a Two-Bit Greater-Than Comparator Using When-Else

```
-- Two-bit greater-than circuit: Conditional VHDL Description
-- using when-else
library ieee;
use ieee.std_logic_1164.all;

entity comparator_greater_than_behavioral is
    port ( A: in std_logic_vector (1 downto 0);
           B: in std_logic_vector (1 downto 0);
      A_greater_than_B: out std_logic);
end comparator_greater_than_behavioral;

architecture when_else of comparator_greater_than_behavioral is
begin
    A_greater_than_B <= '1' when (A > B) else
                        '0';
end when_else;
```

2-19 VHDL For a Two-bit Greater-Than Comparator Using With-Select

```
-- Two-bit greater-than circuit: Conditional VHDL Description
-- using with-select
library ieee;
use ieee.std_logic_1164.all, ieee.std_logic_unsigned.all;

entity comparator_greater_than_behavioral2 is
   port ( A: in std_logic_vector(1 downto 0);
          B: in std_logic_vector(1 downto 0);
          A_greater_than_B: out std_logic);
end comparator_greater_than_behavioral2;

architecture with_select of comparator_greater_than_behavioral2 is
begin
   with A select
   A_greater_than_B <= '0' when "00",
                       B(0) nor B(1) when "01",
                       not B(1) when "10",
                       B(0) nand B(1) when "11",
                       'X' when others;
end with_select;
```

2-19 VHDL For a Two-bit Greater-Than Comparator Using With-Select

```
-- Two-bit greater-than circuit: Conditional VHDL Description
-- using with-select
library ieee;
use ieee.std_logic_1164.all, ieee.std_logic_unsigned.all;

entity comparator_greater_than_behavioral2 is
   port ( A: in std_logic_vector (1 downto 0);
          B: in std_logic_vector(1 downto 0);
          A_greater_than_B : out std_logic);
end comparator_greater_than_behavioral2;

architecture with_select of comparator_greater_than_behavioral2 is
begin
   with A select
   A_greater_than_B <= '0' when "00",
                       B(0) nor B(1) when "01",
                       not B(1) when "10",
                       B(0) nand B(1) when "11",
                       'X' when others;
end with_select;
```

2-20 Structural Verilog for a Two-Bit Greater-Than Circuit

```
// Two-bit greater-than circuit : Verilog structural model
module comparator_greater_than_structural (A,B, A_greater_than_B);
   input [1:0] A, B;
   output A_greater_than_B;
   wire B0_n, B1_n, and0_out, and1_out, and2_out;
     not
       inv0(B0_n, B[0]), inv1(B1_n, B[1]);
     and
       and0(and0_out, A[1], B1_n);
       and1(and1_out, A[0], B0_n);
       and2(and2_out, A[0], B1_n, B0_n);
   or
       or0(A_greater_than_B, and0_out, and1_out, and2_out);
endmodule
```

2-21 Dataflow Verilog for a Two-Bit Greater-Than Comparator

```
// Two-bit greater-than circuit : Dataflow model
module comparator_greater_than_dataflow (A, B, A_greater_than_B);
    input [1:0] A, B;
   output A_greater_than_B;
   wire B1_n, B0_n, and0_out, and1_out, and2out;
   assign B1_n = ~B[1];
   assign B0_n = ~B[0];
   assign and0_out = A[1] & B1_n;
   assign and1_out = A[1] & A[0] & B0_n;
   assign and2_out = A[0] & B1_n & B0_n;
    assign A_greater_than_B = and0_out | and1_out | and2_out;
endmodule
```

2-22 Verilog for a Two-Bit Greater-Than Comparator Using Conditional Operator

```
// Two-bit greater-than circuit : Conditional model
module comparator_greater_than_conditional2 (A,B, A_greater_than_B);
    input [1:0] A, B;
    output A_greater_than_B;
    assign A_greater_than_B = (A > B) ? 1'b1 :
                              1'b0;
endmodule
```

2-23 Verilog for a Two-Bit Greater-Than Circuit Using Behavioral Model

```
// Two-bit greater-than circuit: Conditional model
module comparator_greater_than_conditional (A,B, A_greater_than_B);
    input [1:0] A,B;
    output A_greater_than_B;
    assign A_greater_than_B = (A == 2'b00) ? 1'b0 :
                              (A == 2'b01) ? ~(B[1] | B[0]):
                              (A == 2'b10) ? ~B[1] :
                              (A == 2'b11) ? ~(B[1] & B[0]):
                              1'bx;
endmodule
```

2-24 Verilog for a Two-Bit Greater-Than Circuit Using a Behavioral Description

```
// Two-bit greater-than circuit: Behavioral model
module comparator_greater_than_behavioral (A,B, A_greater_than_B);
   input [1:0] A,B;
   output A_greater_than_B;
   assign A_greater_than_B = A > B;
endmodule
```