Digital Circuit And Logic Design I

Lecture 5

Outline

- Combinational Logic Design Examples
  1. 2-bit comparator circuit design using K-map
  2. 2-bit comparator circuit design using QM-procedure
  3. BCD to 2421 converter circuit design using K-map
  4. BCD to 2421 converter circuit design using QM-procedure
- Combinational logic design practices
  1. Documentation standards

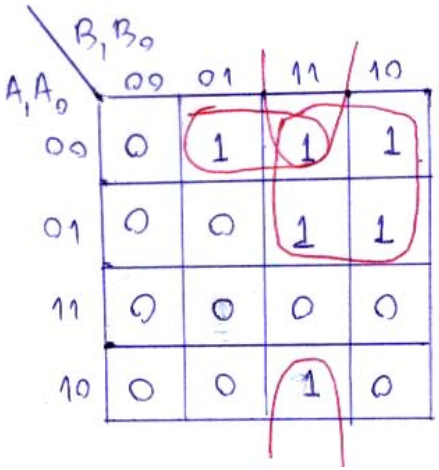
Combinational Logic Design Examples

1. 2-bit comparator circuit design using K-map

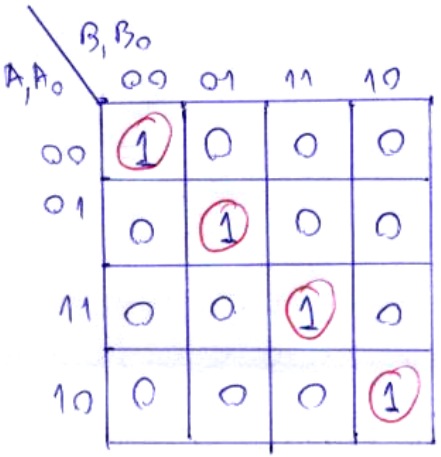
- A 2-bit comparator receives two 2-bit numbers, A = A_1A_0 and B = B_1B_0. Design a minimal sum of products circuit that produces a “001” output when A<B, “010” when A=B, and “100” when A>B
- From instruction, there are 3 bits output $F_2F_1F_0$
  - A<B ➔ $F_0 = \Sigma_{A1,A0,B1,B0}(1,2,3,6,7,11)$
  - A=B ➔ $F_1 = \Sigma_{A1,A0,B1,B0}(0,5,10,15)$
  - A>B ➔ $F_2 = \Sigma_{A1,A0,B1,B0}(4,8,9,12,13,14)$

1. 2-bit comparator circuit design using K-map (cont.)

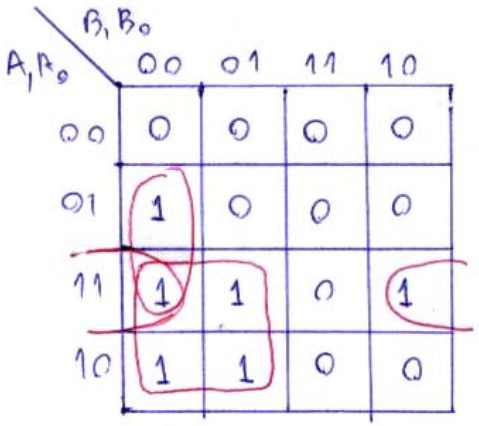
- Draw K-maps



$$F_0 = A_1' \cdot A_0' \cdot B_0 + A_0' \cdot B_1 \cdot B_0 + A_1' \cdot B_1$$



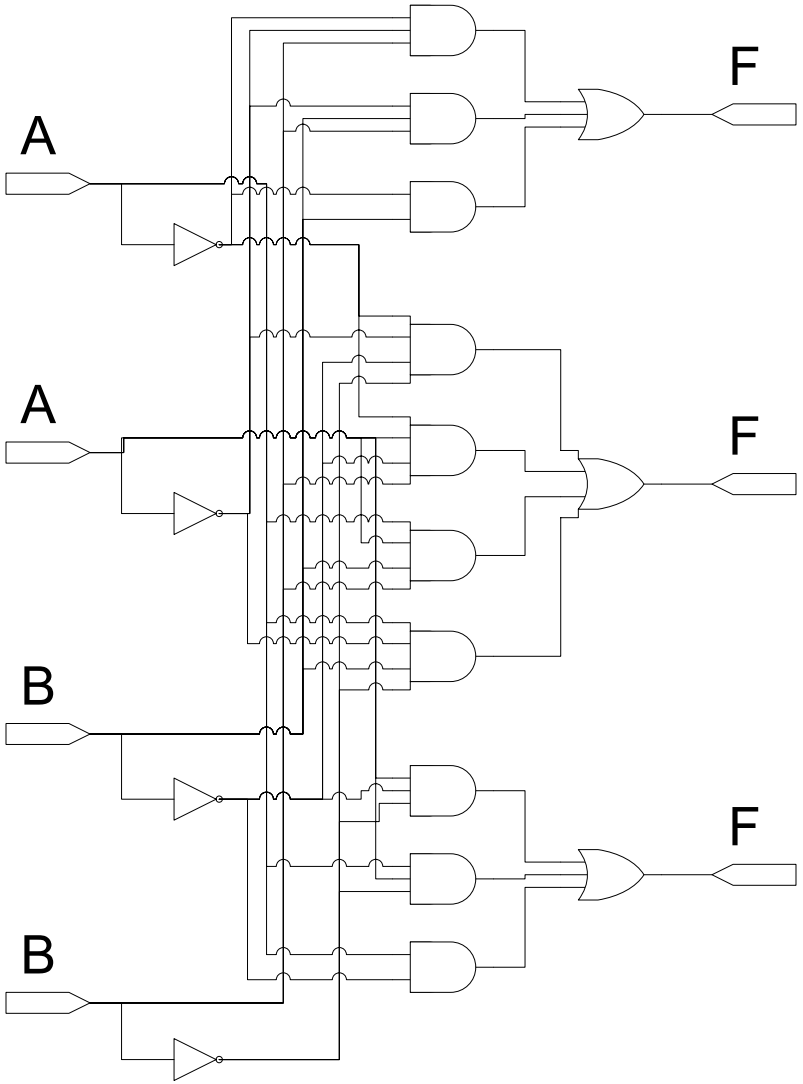
$$F_1 = A_1' \cdot A_0' \cdot B_1' \cdot B_0' + A_1' \cdot A_0 \cdot B_1' \cdot B_0 + A_1 \cdot A_0 \cdot B_1 \cdot B_0 + A_1 \cdot A_0' \cdot B_1 \cdot B_0'$$



$$F_2 = A_0 \cdot B_1' \cdot B_0' + A_1 \cdot A_0 \cdot B_0' + A_1 \cdot B_1'$$

1. 2-bit comparator circuit design using K-map (cont.)

- Draw circuit diagram



2. 2-bit comparator circuit design using QM-procedure

- From instruction, there are 3 bits output $F_2F_1F_0$
  - A<B ➔ F_0 = $\Sigma_{A1,A0,B1,B0}(1,2,3,6,7,11)$
  - A=B ➔ F_1 = $\Sigma_{A1,A0,B1,B0}(0,5,10,15)$
  - A>B ➔ F_2 = $\Sigma_{A1,A0,B1,B0}(4,8,9,12,13,14)$
- Finding minimal sum of F_0
  - Step 1 and 2

✓ 1 ✓ 2	0001 0010	Group1
✓ 3 ✓ 6	0011 0110	Group2
✓ 7 ✓ 11	0111 1011	Group3

2. 2-bit comparator circuit design using QM-procedure (cont.)

- Finding minimal sum of F_0 (cont.)
  - Step 3
  - Step 4

(1,3) ✓(2,3) ✓(2,6)	00-1 001- 0-10	Group 1 and 2
✓(3,7) (3,11) ✓(6,7)	0-11 -011 011-	Group 2 and 3

(1,3) (2,3,6,7) (3,11)	00-1 0-1- -011	PI_1 PI_2 PI_3

2. 2-bit comparator circuit design using QM-procedure (cont.)

- Finding minimal sum of F_0 (cont.)
  - Step 5,6, and 7 ⟶
  - Step 8
    - $F_0 = PI_1 + PI_2 + PI_3$
      $= 00\text{-}1 + 0\text{-}1\text{-} + \text{-}011$
  - Thus, $F_0 = A_1' \cdot A_0' \cdot B_0 + A_1' \cdot B_1 + A_0' \cdot B_1 \cdot B_0$

	✓	✓	✓	✓	✓	✓
	1	2	3	6	7	11
PI_1	×		×			
PI_2		×	×	×	×	
PI_3			×			×

2. 2-bit comparator circuit design using QM-procedure (cont.)

- Finding minimal sum of F_1
  - Step 1 and 2

0	0000	Group0
-		Group1
5 10	0101 1010	Group2
-		Group3
15	1111	Group4

2. 2-bit comparator circuit design using QM-procedure (cont.)

- Finding minimal sum of F_1 (cont.)
  - Step 3
    - Can't combine any term
  - Step 4

0	0000	PI_1
5	0101	PI_2
10	1010	PI_3
15	1111	PI_4

2. 2-bit comparator circuit design using QM-procedure (cont.)

- Finding minimal sum of F_1 (cont.)
  - Step 5,6, and 7 ⟶
  - Step 8
    - $F_0 = PI_1 + PI_2 + PI_3 + PI_4$
      $= 0000 + 0101 + 1010 + 1111$
  - Thus, $F_1 = A_1' \cdot A_0' \cdot B_1' \cdot B_0' + A_1' \cdot A_0 \cdot B_1' \cdot B_0' + A_1 \cdot A_0' \cdot B_1 \cdot B_0' + A_1 \cdot A_0 \cdot B_1 \cdot B_0$

	✓	✓	✓	✓
	0	5	10	15
PI_1	×			
PI_2		×		
PI_3			×	
PI_4				×

2. 2-bit comparator circuit design using QM-procedure (cont.)

- Finding minimal sum of F_2
  - Step 1 and 2

✓ 4 ✓ 8	0100 1000	Group1
✓ 9 ✓ 12	1001 1100	Group2
✓ 13 ✓ 14	1101 1110	Group3

2. 2-bit comparator circuit design using QM-procedure (cont.)

- Finding minimal sum of F_0 (cont.)
  - Step 3
  - Step 4

(4,12) ✓(8,9) ✓(8,12)	-100 100- 1-00	Group 1 and 2
✓(9,13) ✓(12,13) (12,14)	1-01 110- 11-0	Group 2 and 3

(4,12)	-100	PI_1
(8,9,12,13)	1-0-	PI_2
(12,14)	11-0	PI_3

2. 2-bit comparator circuit design using QM-procedure (cont.)

- Finding minimal sum of F_0 (cont.)
  - Step 5,6, and 7 ⟶
  - Step 8
    - $F_2 = PI_1 + PI_2 + PI_3$
      $=-100 + 1\text{-}0\text{-} + 11\text{-}0$
  - Thus, $F_2 = A_0 \cdot B_1' \cdot B_0' + A_1 \cdot B_1' + A_1 \cdot A_0 \cdot B_0'$

	✓	✓	✓	✓	✓	✓
	4	8	9	12	13	14
PI_1	×			×		
PI_2		×	×	×	×	
PI_3				×		×

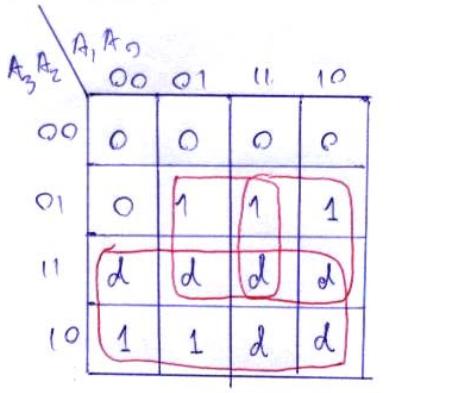
3. BCD to 2421 converter using K-map

- A BCD to 2421 converter receives 4-bit BCD and convert them to 4-bit 2421 code
- $F_3 = \Sigma_{A3,A2,A1,A0}(5,6,7,8,9) + d(10,11,12,13,14,15)$
- $F_2 = \Sigma_{A3,A2,A1,A0}(4,6,7,8,9) + d(10,11,12,13,14,15)$
- $F_1 = \Sigma_{A3,A2,A1,A0}(2,3,5,8,9) + d(10,11,12,13,14,15)$
- $F_0 = \Sigma_{A3,A2,A1,A0}(1,3,5,7,9) + d(10,11,12,13,14,15)$

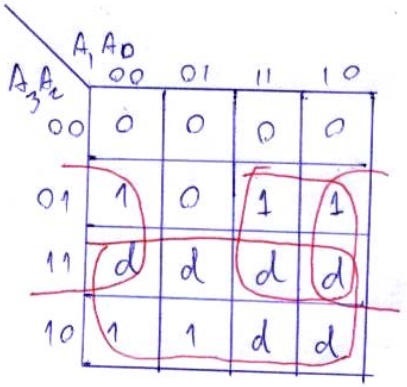
Value	BCD				2421			
	A_3	A_2	A_1	A_0	F_3	F_2	F_1	F_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	1	0	0
7	0	1	1	1	1	1	0	1
8	1	0	0	0	1	1	1	0
9	1	0	0	1	1	1	1	1

1. 2-bit comparator circuit design using K-map (cont.)

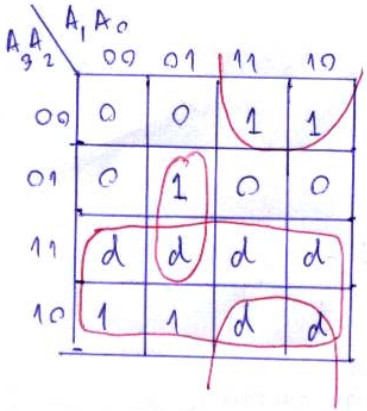
- Draw K-maps



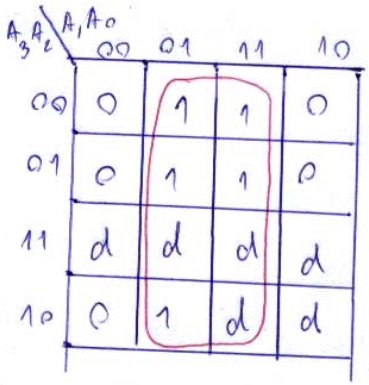
$F_3 = A_3 + A_2 \cdot A_0 + A_2 \cdot A_1$



$F_2 = A_3 + A_2 \cdot A_0' + A_2 \cdot A_1$



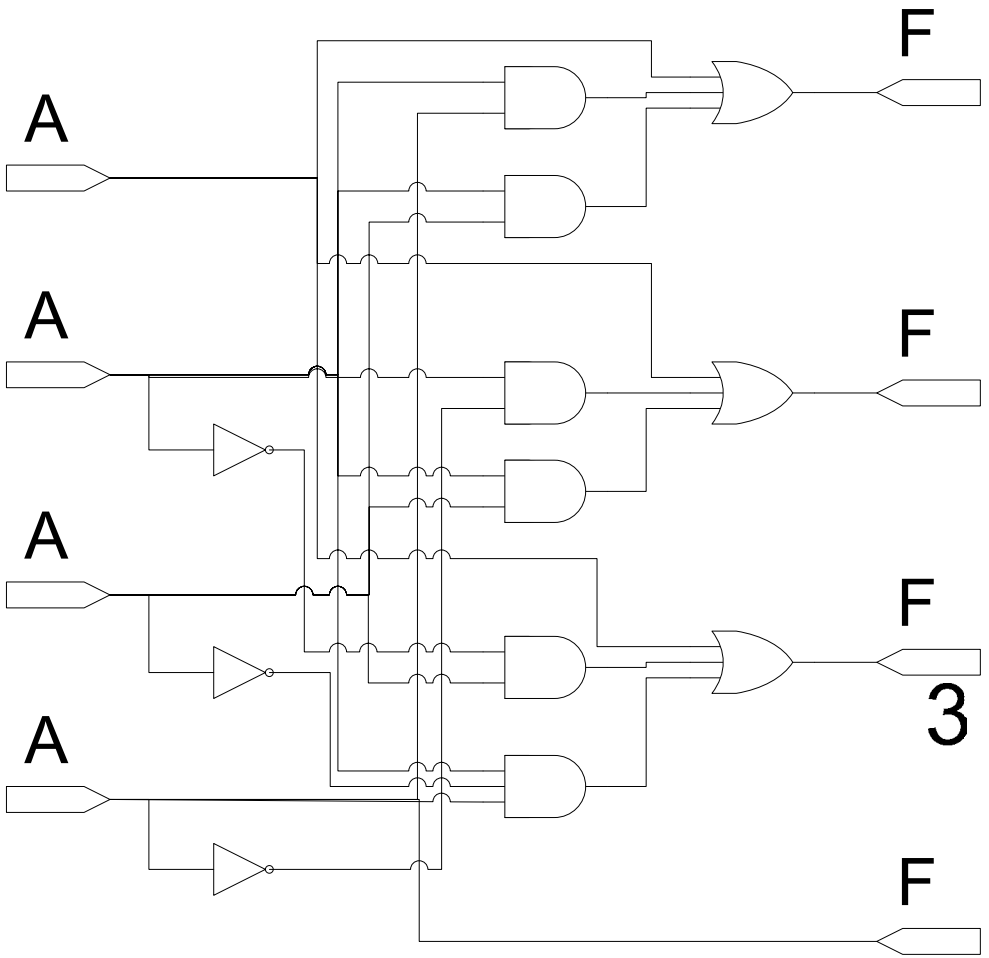
$F_1 = A_3 + A_2' \cdot A_1 + A_2 \cdot A_1' \cdot A_0$



$F_0 = A_0$

1. 2-bit comparator circuit design using K-map (cont.)

- Draw circuit diagram



4. BCD to 2421 converter using QM-procedure

- $F_3 = \Sigma_{A3,A2,A1,A0}(5,6,7,8,9) + d(10,11,12,13,14,15)$
- $F_2 = \Sigma_{A3,A2,A1,A0}(4,6,7,8,9) + d(10,11,12,13,14,15)$
- $F_1 = \Sigma_{A3,A2,A1,A0}(2,3,5,8,9) + d(10,11,12,13,14,15)$
- $F_0 = \Sigma_{A3,A2,A1,A0}(1,3,5,7,9) + d(10,11,12,13,14,15)$

4. BCD to 2421 converter using QM-procedure (cont.)

Step 1 and 2

F_3		F_2		F_1		F_0		
✓ 8	1000	✓ 4	0100	✓ 2	0010	✓ 1	0001	Group 1
		✓ 8	1000	✓ 8	1000			
✓ 5	0101	✓ 6	0110	✓ 3	0011	✓ 3	0011	Group 2
✓ 6	0110	✓ 9	1001	✓ 5	0101	✓ 5	0101	
✓ 9	1001	✓ 10	1010	✓ 9	1001	✓ 9	1001	
✓ 10	1010	✓ 12	1100	✓ 10	1010	✓ 10	1010	
✓ 12	1100			✓ 12	1100	✓ 12	1100	
✓ 7	0111	✓ 7	0111	✓ 11	1011	✓ 7	0111	Group 3
✓ 11	1011	✓ 11	1011	✓ 13	1101	✓ 11	1011	
✓ 13	1101	✓ 13	1101	✓ 14	1110	✓ 13	1101	
✓ 14	1110	✓ 14	1110			✓ 14	1110	
✓ 15	1111	✓ 15	1111	✓ 15	1111	✓ 15	1111	Group 4

4. BCD to 2421 converter using QM-procedure (cont.)

Step 3

F_3		F_2		F_1		F_0		
✓(8,9)	100-	✓(4,6)	01-0	✓(2,3)	001-	✓(1,3)	00-1	Group 1 and 2
✓(8,10)	10-0	✓(4,12)	-100	✓(2,10)	-010	✓(1,5)	0-01	
✓(8,12)	1-00	✓(8,9)	100-	✓(8,9)	100-	✓(1,9)	-001	
		✓(8,10)	10-0	✓(8,10)	10-0			
		✓(8,12)	1-00	✓(8,12)	1-00			
✓(5,7)	01-1	✓(6,7)	011-	✓(3,11)	-011	✓(3,7)	0-11	Group 2 and 3
✓(5,13)	-101	✓(6,14)	-110	(5,13)	-101	✓(3,11)	-011	
✓(6,7)	011-	✓(9,11)	10-1	✓(9,11)	10-1	✓(5,7)	01-1	
✓(6,14)	-110	✓(9,13)	1-01	✓(9,13)	1-01	✓(5,13)	-101	
✓(9,11)	10-1	✓(10,11)	101-	✓(10,11)	101-	✓(9,11)	10-1	
✓(9,13)	1-01	✓(10,14)	1-10	✓(10,14)	1-10	✓(9,13)	1-01	
✓(10,11)	101-	✓(12,13)	110-	✓(12,13)	110-	✓(10,11)	101-	
✓(10,14)	1-10	✓(12,14)	11-0	✓(12,14)	11-0	✓(10,14)	1-10	
✓(12,13)	110-					✓(12,13)	110-	
✓(12,14)	11-0					✓(12,14)	11-0	
✓(7,15)	-111	✓(7,15)	-111	✓(11,15)	1-11	✓(7,15)	-111	Group 3 and 4
✓(11,15)	1-11	✓(11,15)	1-11	✓(13,15)	11-1	✓(11,15)	1-11	
✓(13,15)	11-1	✓(13,15)	11-1	✓(14,15)	111-	✓(13,15)	11-1	
✓(14,15)	111-	✓(14,15)	111-			✓(14,15)	111-	

4. BCD to 2421 converter using QM-procedure (cont.)

Step 3

F_3		F_2		F_1		F_0		
✓(8,9,10,11)	10--	(4,6,12,14)	-1-0	(2,3,10,11)	-01-	✓(1,3,5,7)	0--1	Group 1,2, and 3
✓(8,9,12,13)	1-0-	✓(8,9,10,11)	10--	✓(8,9,10,11)	10--	✓(1,3,9,11)	-0-1	
✓(8,10,12,14)	1--0	✓(8,9,12,13)	1-0-	✓(8,9,12,13)	1-0-	✓(1,5,9,13)	--01	
		✓(8,10,12,14)	1--0	✓(8,10,12,14)	1--0			
(5,7,13,15)	-1-1	(6,7,14,15)	-11-	(5,13)	-101	✓(3,7,11,15)	--11	Group 2,3, and 4
(6,7,14,15)	-11-	✓(9,11,13,15)	1--1	✓(9,11,13,15)	1--1	✓(5,7,13,15)	-1-1	
✓(9,11,13,15)	1--1	✓(10,11,14,15)	1-1-	✓(10,11,14,15)	1-1-	✓(9,11,13,15)	1--1	
✓(10,11,14,15)	1-1-	✓(12,13,14,15)	11--	✓(12,13,14,15)	11--	(10,11,14,15)	1-1-	
✓(12,13,14,15)	11--					(12,13,14,15)	11--	

4. BCD to 2421 converter using QM-procedure (cont.)

Step 4

F_3		F_2		
(8,9,10,11,12,13,14,15)	1---	(4,6,12,14)	-1-0	PI_1
(5,7,13,15)	-1-1	(8,9,10,11,12,13,14,15)	1---	PI_2
(6,7,14,15)	-11-	(6,7,14,15)	-11-	PI_3

F_1		F_0		
(2,3,10,11)	-01-	(1,3,5,7,9,11,13,15)	---1	PI_1
(8,9,10,11,12,13,14,15)	1---	(10,11,14,15)	1-1-	PI_2
(5,13)	-101	(12,13,14,15)	11--	PI_3

4. BCD to 2421 converter using QM-procedure (cont.)

Step 5,6, and 7

		1	2	3	4	5	6	7	8	9
						✓	✓	✓	✓	✓
F_3	PI_1								×	×
	PI_2					×		×		
	PI_3						×	×		
					✓		✓	✓	✓	✓
F_2	PI_1				×		×			
	PI_2								×	×
	PI_3						×	×		
			✓	✓		✓			✓	✓
F_1	PI_1		×	×						
	PI_2								×	×
	PI_3					×				
		✓		✓		✓		✓		✓
F_0	PI_1	×		×		×		×		×
	PI_2									
	PI_3									

4. BCD to 2421 converter using QM-procedure (cont.)

- $F_3 = PI_1 + PI_2 + PI_3$
  - $F_3 = 1\text{---} + \text{-}1\text{-}1 + \text{-}11\text{-}$
  - $F_3 = A_3 + A_2 \cdot A_0 + A_2 \cdot A_1$
- $F_2 = PI_1 + PI_2 + PI_3$
  - $F_2 = \text{-}1\text{-}0 + 1\text{---} + \text{-}11\text{-}$
  - $F_2 = A_2 \cdot A_0' + A_3 + A_2 \cdot A_1$
- $F_1 = PI_1 + PI_2 + PI_3$
  - $F_1 = \text{-}01\text{-} + 1\text{---} + \text{-}101$
  - $F_1 = A_2' \cdot A_0 + A_3 + A_2 \cdot A_1' \cdot A_0$
- $F_0 = PI_1$
  - $F_0 = \text{---}1$
  - $F_0 = A_0$

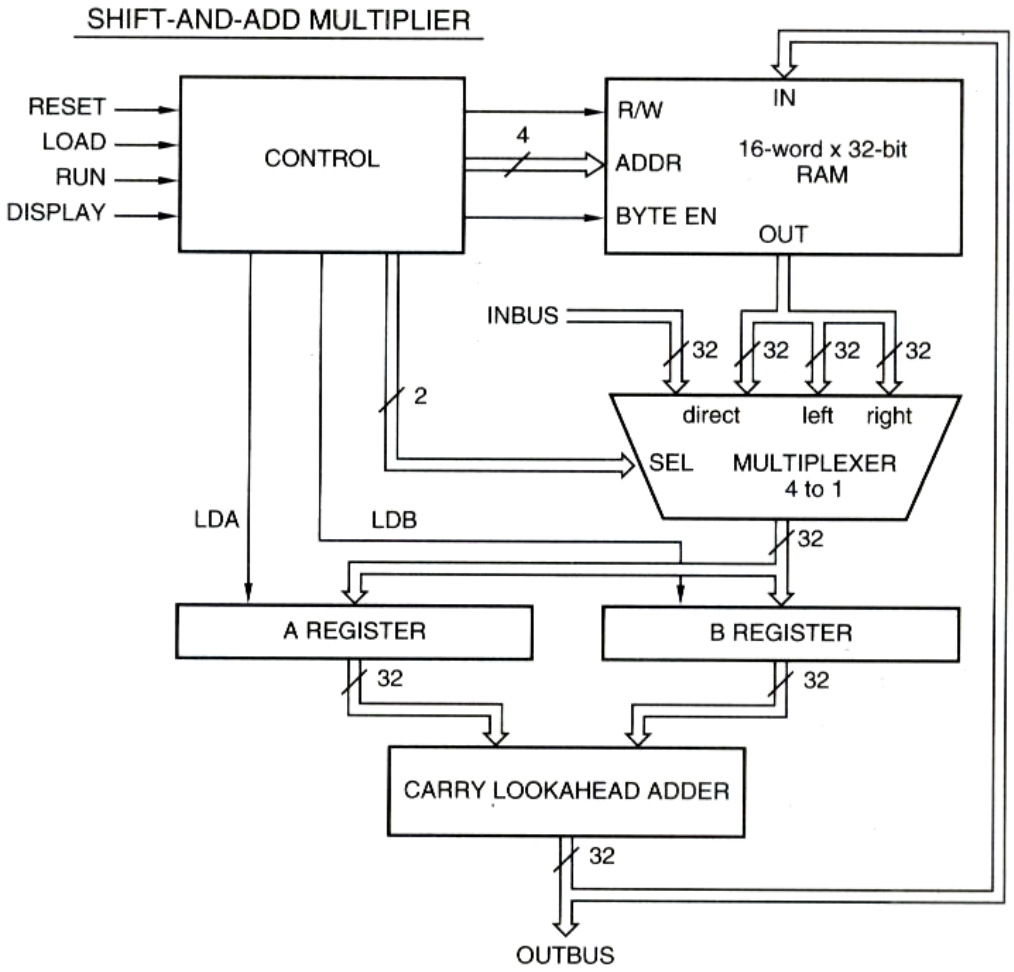
Combinational Logic Design Practices

1. Documentation Standards

- Good documentation is essential for correct design and efficient maintenance of digital systems.
- Although the type of documentation depends on system complexity and the engineering and manufacturing environments, a documentation package should generally contain at least the following six items:
  - Circuit specification
  - Block diagram
  - Schematic diagram
  - Timing diagram
  - Structured logic device description
  - Circuit description

1. Documentation Standards (cont.)

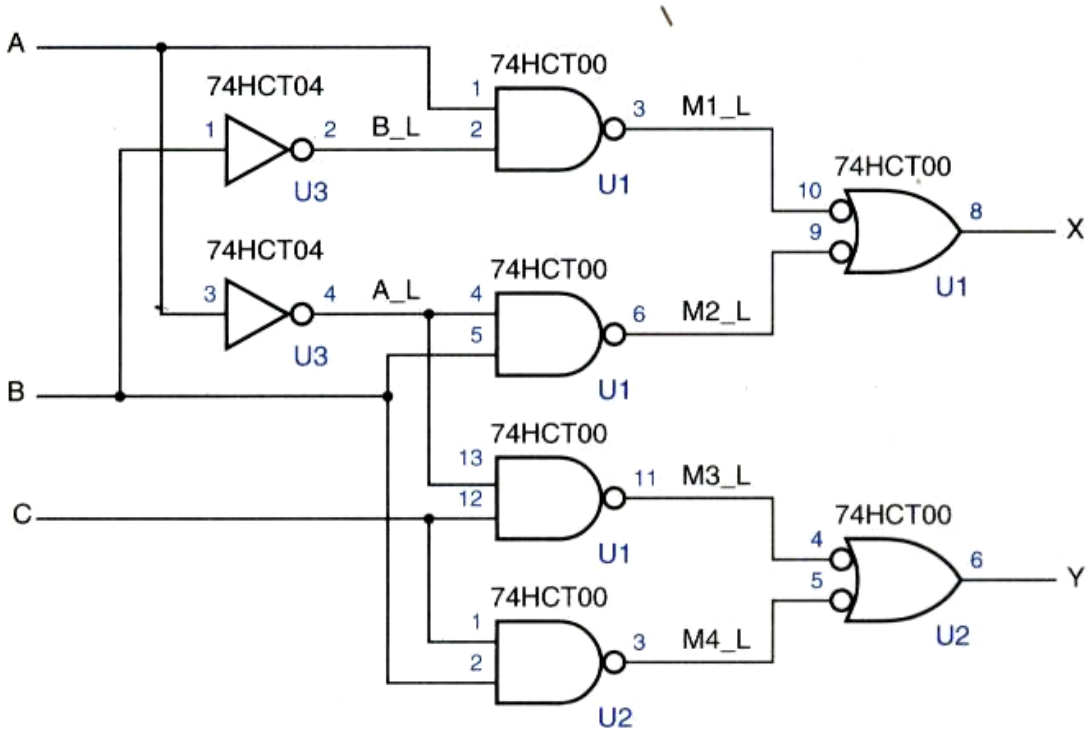
- Block diagram



Pictures from text book DDPP

1. Documentation Standards (cont.)

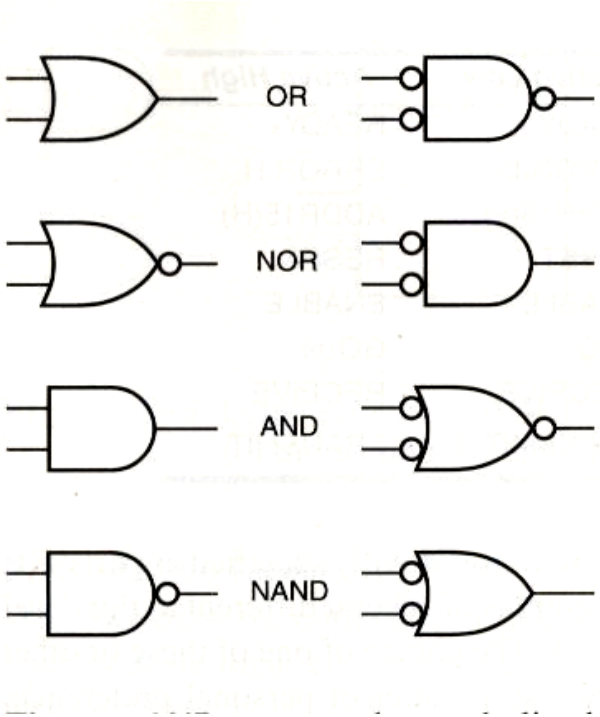
- Schematic diagram

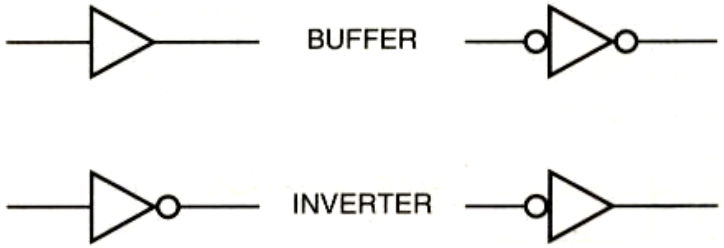


Pictures from text book DDPP

1. Documentation Standards (cont.)

- Gate symbols





Equivalent gate symbols

Pictures from text book DDPP

1. Documentation Standards (cont.)

- Signal names and active levels
  - Each input and output signal in a logic circuit should have a descriptive alphanumeric label, the signal's name.
  - A signal's name indicates an action that is controlled (GO, PAUSE), a condition that it detects (READY, ERROR), or data that it carries (INBUS[31:0]).
  - Each signal name should have an **active level** associated with it.
    - A signal is **active high** if it performs the named action or denotes the named condition when it is HIGH or 1.
    - A signal is **active low** if it performs the named action or denotes the named condition when it is LOW or 0.
    - A signal is said to be **asserted** when it is at its active level. A signal is said to be **negated** (or **deasserted**) when it is not at its active level.

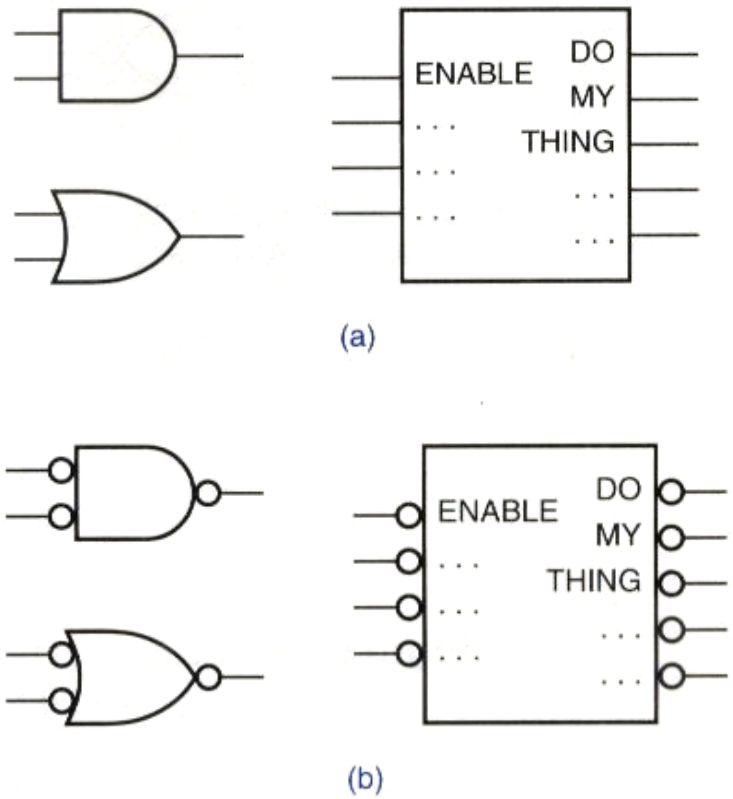
1. Documentation Standards (cont.)

- Signal names and active levels (cont.)
  - The active level of each signal in a circuit is normally specified as part of its name, according to some convention.

Active Low	Active High
READY-	READY+
ERROR.L	ERROR.H
ADDR15(L)	ADDR15(H)
RESET*	RESET
ENABLE~	ENABLE
~GO	GO
/RECEIVE	RECEIVE
TRANSMIT_L	TRANSMIT

1. Documentation Standards (cont.)

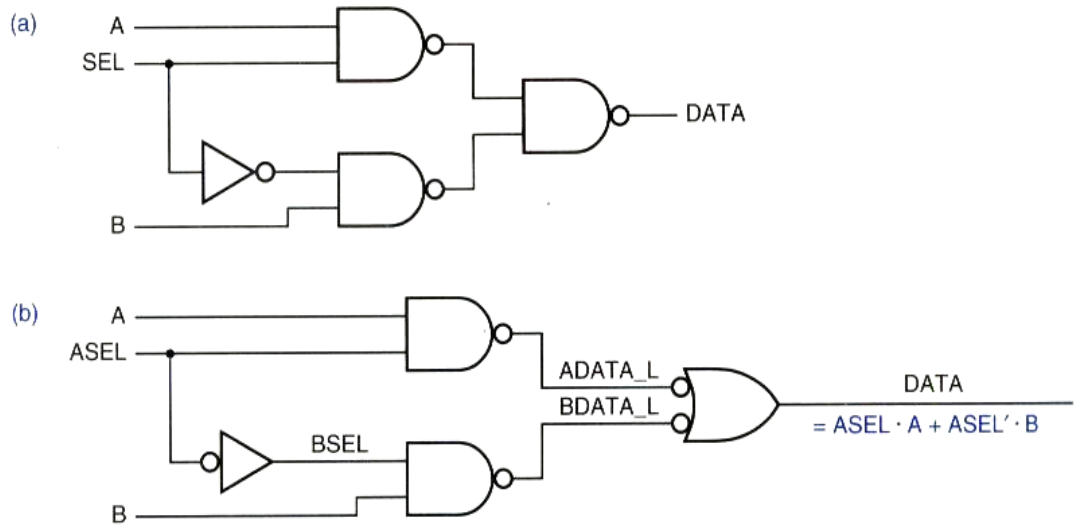
- Active levels for Pins
  - Active levels may be associated with the input and output pins of gates and larger-scale logic elements.
  - We use an inversion bubble to indicate an active-low pin and the absence of a bubble to indicate an active-high pin



Pictures from text book DDPP

1. Documentation Standards (cont.)

- Bubble-to-Bubble Logic Design
  - Bubble-to-bubble logic design is the practice of choosing logic symbols and signal names, including active-level designators, that make the function of a logic circuit easier to understand.
  - Usually, this means choosing signal names and gate types and symbols so the most of the inversion bubbles “cancel out” and the logic diagram can be analyzed as if all of the signals were active high.



1. Documentation Standards (cont.)

- Bubble-to-Bubble Logic Design (cont.)
  - Bubble-to-bubble logic design rules
    - The signal name on a device's output should have the same active level as the device's output pin
    - If the active level of the input signal is the same as that of the input pin to which it is connected, then the logic function inside the symbolic outline is activated when the signal is asserted. This is the most common case in a logic diagram.
    - If the active level of an input signal is the opposite of that of the input pin to which it is connected, then the logic function inside the symbolic outline is activated when the signal is negated. This case should be avoided whenever possible because it forces us to keep track mentally of a logical negation to understand the circuit.

1. Documentation Standards (cont.)