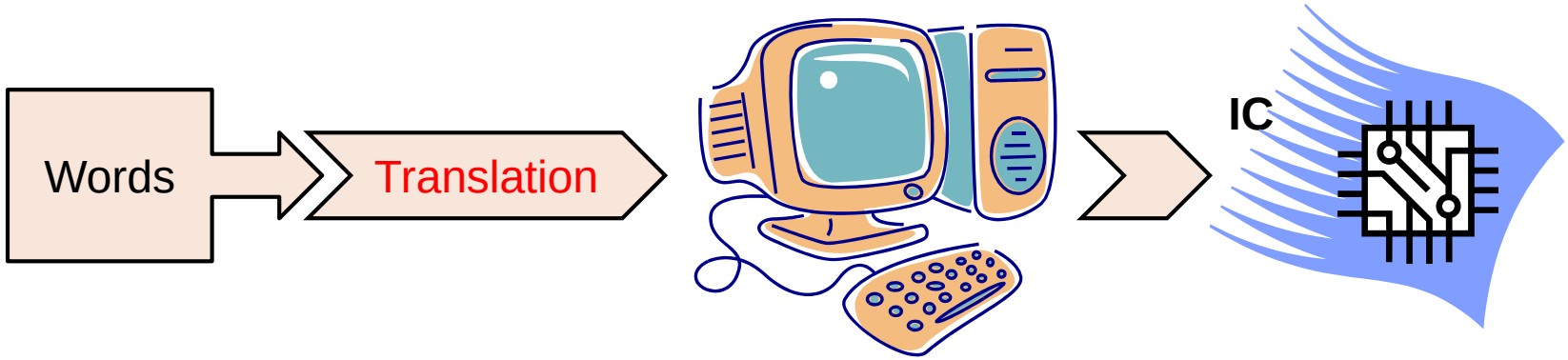
UNIT 4
MINTERM AND MAXTERM EXPANSIONS

Fall 2021

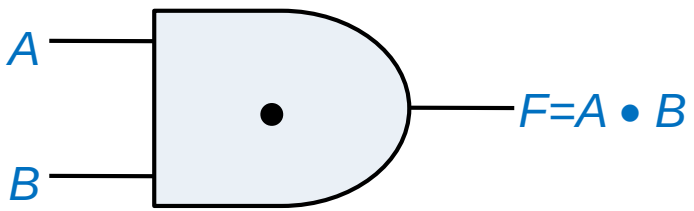
Minterm and Maxterm Expansions

- **Contents**
  - Conversion of English sentences to Boolean equations
  - Combinational logic design using a truth table
  - Minterm and maxterm expansions
  - Incompletely specified functions
  - Examples of truth table construction
  - Design of binary adders
- **Reading**
  - Unit 4

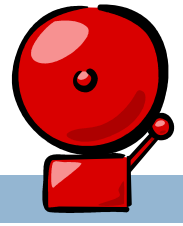


Objectives

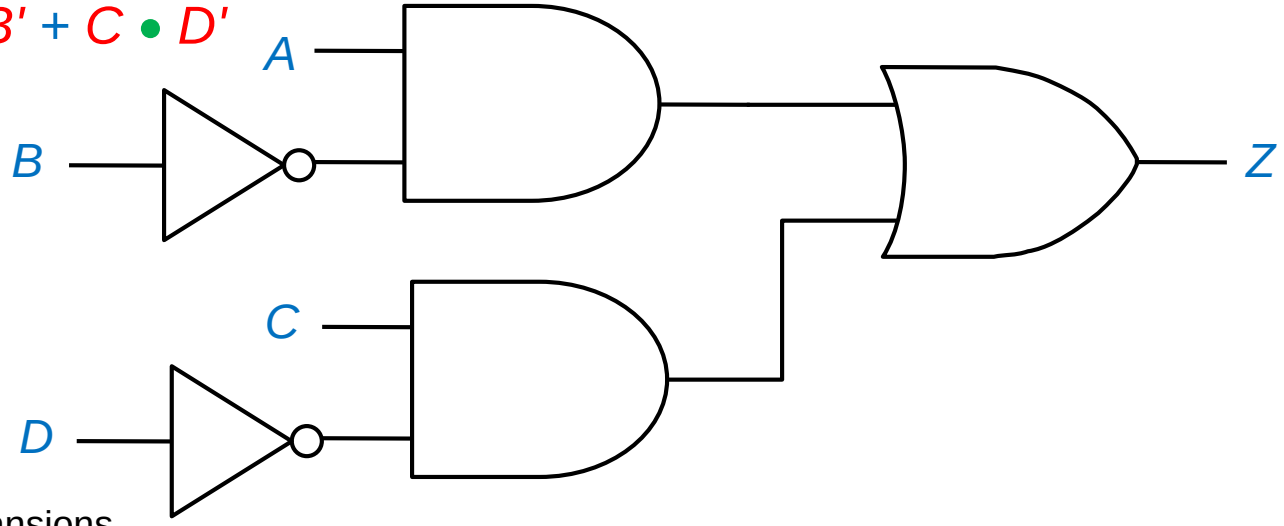
- **Design a combinational logic circuit starting with a word description of the desired circuit behavior**
- **Steps:**
  1. Translate the word description into a switching function
     - Boolean expression or truth table
  2. Simplify the function
  3. Realize it using available logic gates
- **e.g., Mary watches TV if it is Monday night and she has finished her homework**
  - ⇒ Translate…
  - Mary watches TV (*F*) if
  - It is Monday night (*A*)
  - And
  - She has finished her homework (*B*)
  - ⇒ $F = A \bullet B$



Example: Designing an Alarm



- **The alarm will ring iff the alarm switch is turned on and the door is not closed, or it is after 6 P.M. and the window is not closed**
  - ⇒ Translate…
  - The alarm will ring (*Z*) iff
  - The alarm switch is on (*A*) and the door is NOT closed (*B'*)
  - Or
  - It is after 6 P.M. (*C*) and the window is NOT closed (*D'*)
  - ⇒ $Z = A \bullet B' + C \bullet D'$



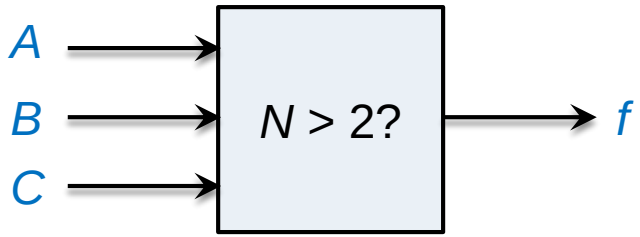
Logic Design using a Truth Table

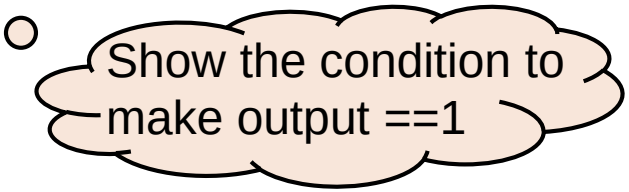
A truth table helps translation

Threshold Detector (1/2)

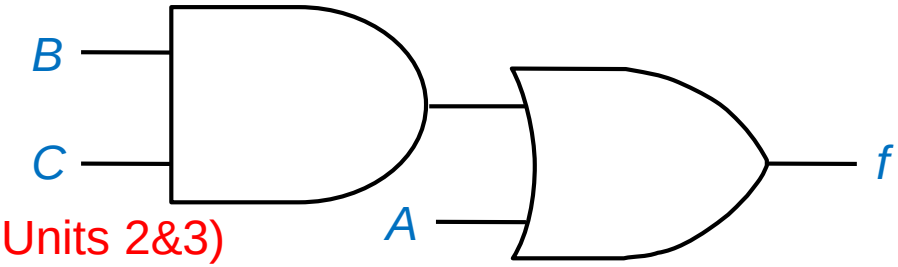
- **Design a detector that outputs 1 when input is greater than 2**
  - Inputs $(A, B, C)_2$ represent an unsigned binary integer N; if $N = (A, B, C)_2 \geq 3 = 011_2$, output $f = 1$; otherwise $f = 0$

	A	B	C	f	f′
0	0	0	0	0	1
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	1	0
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	0



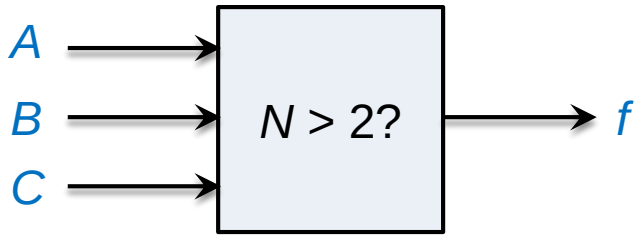


$$f = \overset{3}{A'BC} + \overset{4}{AB'C'} + \overset{5}{AB'C} + \overset{6}{ABC'} + \overset{7}{ABC} \text{ (SOP)}$$
$$= \underline{A'BC + ABC} + \underline{AB'C' + AB'C + ABC' + ABC}$$
$$= A + BC \text{ (Simplified using Boolean Algebra, Units 2\&3)}$$



Threshold Detector (2/2)

- **Counting 1's, we have SOP**
- **What if counting 0's?**



	A	B	C	f	f′
0	0	0	0	0	1
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	1	0
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	0

Show the condition to make output ==0

0 1 2

$f' = A'B'C' + A'B'C + A'BC'$

Taking the complement by DeMorgan's law,

$f = (A'B'C' + A'B'C + A'BC')'$

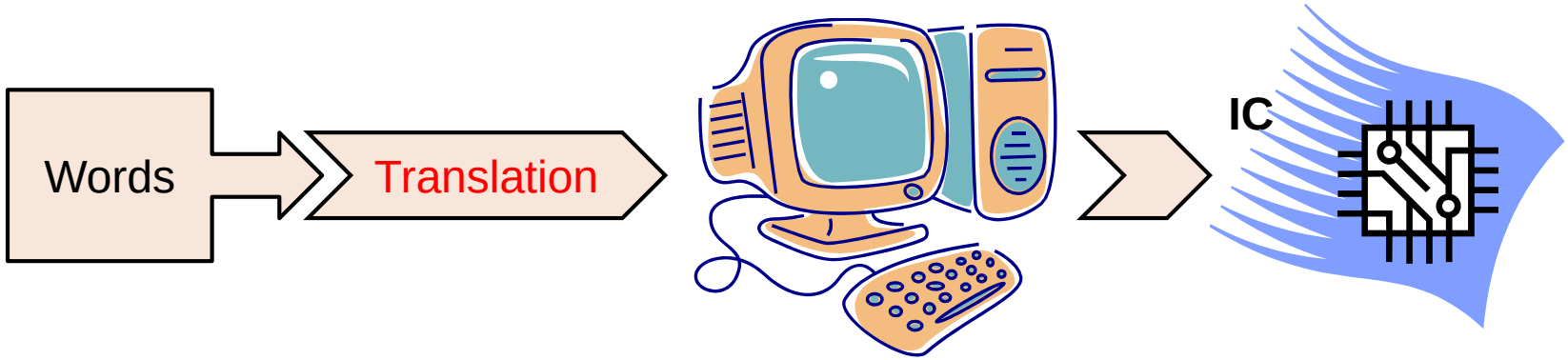
$= [(A' \bullet B' \bullet C') + (A' \bullet B' \bullet C) + (A' \bullet B \bullet C')]'$

$= (A + B + C)(A + B + C')(A + B' + C)$ (POS)

Minterm & maxterm expansions

Summary: Logic Design using a Truth Table

- **General steps:**
  1. Make a truth table according to the word description
  2. Generate a Boolean expression
     - SOP: check 1's
     - POS: check 0's
       - First of all, have *f'* in SOP, then derive *f* in POS
  3. Simplify the Boolean expression



Minterm & Maxterm Expansions

Minterm vs. Maxterm

- **Definition: A minterm/maxterm of *n* variables is a product/sum of *n* literals in which each variable appears exactly once in either true or complement form (but not both)**
  - A literal is a variable or its complement, e.g., *A*, *A'*
  - e.g., assume 3 variables
    - Minterm: *A'BC*, *AB'C'*
    - Maxterm: *A+B+C*, *A+B+C'*

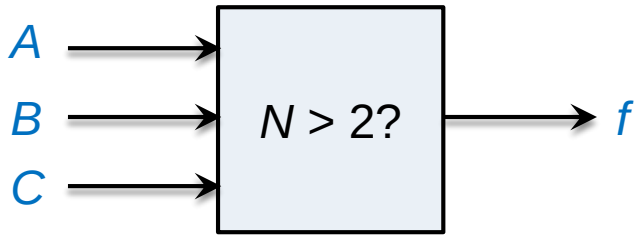
$(m_i)' = M_i$ DeMorgan

Each minterm corresponding one input condition

Row No.	*ABC*	Minterms m_i	Maxterms M_i
0	000	$m_0 = A'B'C'$	$M_0 = A + B + C$
1	001	$m_1 = A'B'C$	$M_1 = A + B + C'$
2	010	$m_2 = A'B\ C'$	$M_2 = A + B' + C$
3	011	$m_3 = A'B\ C$	$M_3 = A + B' + C'$
4	100	$m_4 = A\ B'C'$	$M_4 = A' + B + C$
5	101	$m_5 = A\ B'C$	$M_5 = A' + B + C'$
6	110	$m_6 = A\ B\ C'$	$M_6 = A' + B' + C$
7	111	$m_7 = A\ B\ C$	$M_7 = A' + B' + C'$

Minterm Expansion

- **A minterm expansion or a standard sum of products: a function is written as a sum of minterms**
- **⇒ i.e., counting 1's**
- **e.g.,**
  - $f = A'BC + AB'C' + AB'C + ABC' + ABC$
    $= m_3 + m_4 + m_5 + m_6 + m_7$ (*m*-notation)
    $= \Sigma\, m(3, 4, 5, 6, 7)$



Row No.	*ABC*	Minterms m_i	Maxterms M_i
0	000	$m_0 = A'B'C'$	$M_0 = A + B + C$
1	001	$m_1 = A'B'C$	$M_1 = A + B + C'$
2	010	$m_2 = A'B\ C'$	$M_2 = A + B' + C$
3	011	$m_3 = A'B\ C$	$M_3 = A + B' + C'$
4	100	$m_4 = A\ B'C'$	$M_4 = A' + B + C$
5	101	$m_5 = A\ B'C$	$M_5 = A' + B + C'$
6	110	$m_6 = A\ B\ C'$	$M_6 = A' + B' + C$
7	111	$m_7 = A\ B\ C$	$M_7 = A' + B' + C'$

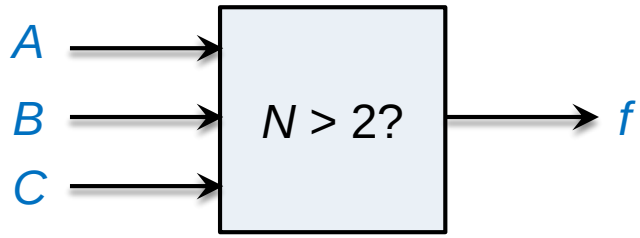
Minterm & maxterm expansions

Maxterm Expansion

- **A maxterm expansion or a standard product of sums: a function is written as a product of of maxterms**
- **⇒ i.e., counting 0's**
- **e.g.,**
  - $f' = A'B'C' + A'B'C + A'BC'$

$\Rightarrow f = (A + B + C)(A + B + C')(A + B' + C)$

$= M_0M_1M_2$ (M-notation) $= \Pi\, M(0, 1, 2)$



Row No.	ABC	Minterms m_i	Maxterms M_i
0	000	$m_0 = A'B'C'$	$M_0 = A + B + C$
1	001	$m_1 = A'B'C$	$M_1 = A + B + C'$
2	010	$m_2 = A'B\ C'$	$M_2 = A + B' + C$
3	011	$m_3 = A'B\ C$	$M_3 = A + B' + C'$
4	100	$m_4 = A\ B'C'$	$M_4 = A' + B + C$
5	101	$m_5 = A\ B'C$	$M_5 = A' + B + C'$
6	110	$m_6 = A\ B\ C'$	$M_6 = A' + B' + C$
7	111	$m_7 = A\ B\ C$	$M_7 = A' + B' + C'$

Complement using Minterms/Maxterms

- $(m_i)' = M_i$
- **Complement of *f*:**
  1. Count 0's in *f* (find *f'* directly)

     $f' = m_0 + m_1 + m_2 = \Sigma\, m(0, 1, 2)$

     $f' = M_3M_4M_5M_6M_7 = \Pi\, M(3, 4, 5, 6, 7)$
  2. Count 1's in *f* (find *f* and then complement it)

     $f' = (f)' = (m_3 + m_4 + m_5 + m_6 + m_7)'$

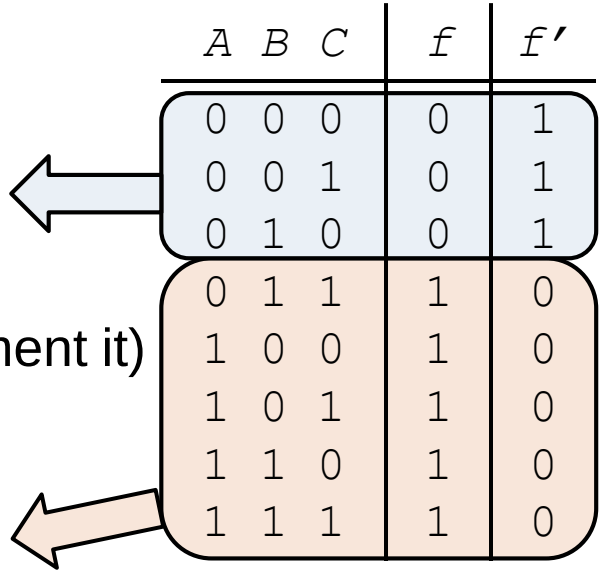
     $= m_3'm_4'm_5'm_6'm_7'$

     $= M_3M_4M_5M_6M_7 = \Pi\, M(3, 4, 5, 6, 7)$

     $f' = (f)' = (M_0M_1M_2)'$

     $= M_0' + M_1' + M_2'$

     $= m_0 + m_1 + m_2 = \Sigma\, m(0, 1, 2)$



A	B	C	f	f′
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Example: Minterm/Maxterm Expansion

- **e.g., $f(a, b, c, d) = a'(b' + d) + acd'$**
  - $f(a, b, c, d) = a'(b' + d) + acd'$
  - $= a'b' + a'd + acd'$
  - $= a'b'(c + c')(d + d') + a'd(b + b')(c + c') + acd'(b + b')$
  - $= a'b'c'd' + a'b'c'd + a'b'cd' + a'b'cd + a'bc'd + a'bcd + abcd' + ab'cd'$

    0000 0001 0010 0011 0101 0111 1110 1010
  - $= \Sigma\, m(0, 1, 2, 3, 5, 7, 10, 14)$ (Minterm Expansion)

    
  - $= \Pi\, M(4, 6, 8, 9, 11, 12, 13, 15)$ (Maxterm Expansion)

Summary (1/2)

- **Convert a Boolean expression to a minterm/maxterm expansion**
  - Use truth table
    - Sometimes there are too many terms
  - Use Boolean algebra
    - SOP $\rightarrow$ multiply out and use $(X + X')=1 \rightarrow$ minterm expansion
    - POS $\rightarrow$ factor and use $XX'=0 \rightarrow$ maxterm expansion

Summary (2/2)

- **Convert between a minterm and a maxterm expansion**
  - If $f = \Sigma\, m_i$, then $f = \Pi\, M_j$, where each m_j is not in f

GIVEN FORM \ DESIRED FORM	**Minterm Expansion of *f***	**Maxterm Expansion of *f***	**Minterm Expansion of *f'***	**Maxterm Expansion of *f'***
Minterm Expansion of *f*	----------	maxterm nos. are those nos. not on the minterm list for *f*	list minterms not present in *f*	maxterm nos. are the same as minterm nos. of *f*
Maxterm Expansion of *f*	minterm nos. are those nos. not on the maxterm list for *f*	----------	minterm nos. are the same as maxterm nos. of *f*	list maxterms not present in *f*

- **There is a 1-to-1 mapping between a truth table and the minterm/maxterm expansion**

AND of Two Minterm Expansions

- **e.g., Given $f_1 = \Sigma\, m(0, 2, 3, 5, 9, 11)$, $f_2 = \Sigma\, m(0, 3, 9, 11, 13, 14)$, find $f_1 f_2$ =?**
  - AND: take the numbers that appear in both expansions:

    $f_1 f_2 = \Sigma\, m(0, 3, 9, 11)$

- **Q: What if AND for two maxterm expansions?**

General Truth Table

- **Q: Given three Boolean variables *A*, *B*, *C*, how many different Boolean functions can you produce?**
- **A:**
  - Each a_i can be assigned with either 0 or 1
  - $\Rightarrow 2^8 = 256$

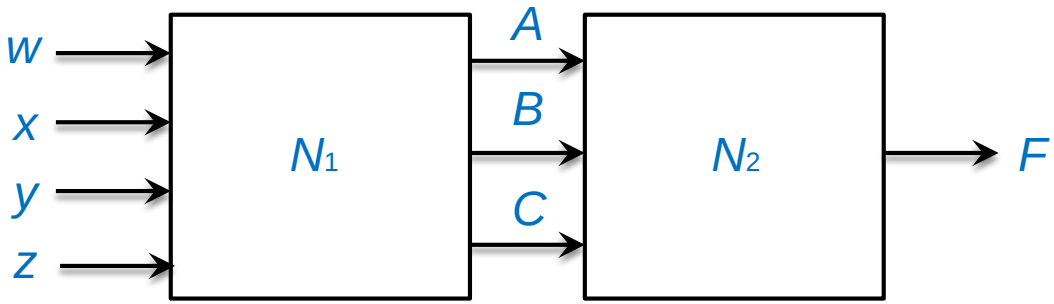
A	*B*	*C*	*f*
0	0	0	a_0
0	0	1	a_1
0	1	0	a_2
0	1	1	a_3
1	0	0	a_4
1	0	1	a_5
1	1	0	a_6
1	1	1	a_7

Incompletely Specified Functions

Incompletely Specified Functions (1/2)

- **A large digital system is usually divided into subcircuits**
  - Assume N_1 never generates ABC = 001/110 for any w, x, y, z

	A	B	C	F
0	0	0	0	1
1	0	0	1	x
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	x
7	1	1	1	1



"don't care" (DC) terms can be assigned with either "0"or "1"

  - F: Incompletely specified function
  - $A'B'C$, ABC': don't care terms

Incompletely Specified Functions (2/2)

- **Impact of don't care terms on Boolean simplification:**
  - Try exhaustive combinations of DCs to find the best expression (may be stupid but works for now)

	A	B	C	F
0	0	0	0	1
1	0	0	1	x
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	x
7	1	1	1	1

1. Assign "0" to both "X":

   $F = A'B'C' + A'BC + ABC = A'B'C' + BC$

2. Assign "1" to 1st "X", and "0" to 2nd "X":

   $F = A'B'C' + A'B'C + A'BC + ABC = A'B' + BC$

3. Assign "1" to both "X":

   $F = A'B'C' + A'B'C + A'BC + ABC' + ABC = A'B' + BC + AB$

4. …

  - 2. is the simplest solution
- **Notation:**
  - $F = \Sigma\, m(0, 3, 7) + \Sigma\, d(1, 6)$
  - $F = \Pi\, M(2, 4, 5) \bullet \Pi\, D(1, 6)$

don't care minterm and maxterm have the same numbers

Minterm & maxterm expansions

Truth Table Construction

Minterm & maxterm expansions

Error Detector for 6-3-1-1 Codes (1/2)

- **Design an error detector for 6-3-1-1 codes:**
  - The output $F = 1$ iff inputs (A, B, C, D) represent an invalid code combination

Decimal digit	6-3-1-1 code
0	0000
1	0001
2	0011
3	0100
4	0101
5	0111
6	1000
7	1001
8	1011
9	1100

1. Construct the truth table

ABCD	*F*
0000	0
0001	0
0010	1
0011	0
0100	0
0101	0
0110	1
0111	0
1000	0
1001	0
1010	1
1011	0
1100	0
1101	1
1110	1
1111	1

Error Detector for 6-3-1-1 Codes (2/2)

ABCD	*F*
0000	0
0001	0
0010	1
0011	0
0100	0
0101	0
0110	1
0111	0
1000	0
1001	0
1010	1
1011	0
1100	0
1101	1
1110	1
1111	1

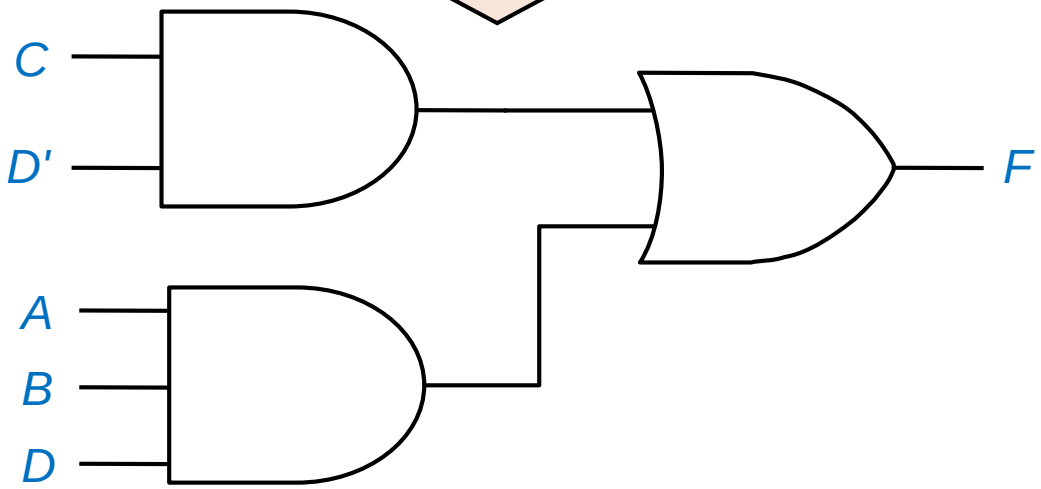
2. Simplify the function

$F(A, B, C, D)$
$= \Sigma\, m(2, 6, 10, 13, 14, 15)$
$= \underline{A'B'CD'} + \underline{A'BCD'} + \underline{AB'CD'} + \underline{ABCD'} + \underline{ABC'D} + \underline{ABCD}$

$= \underline{A'CD'} + \underline{ACD'} + ABD$

$= CD' + ABD$

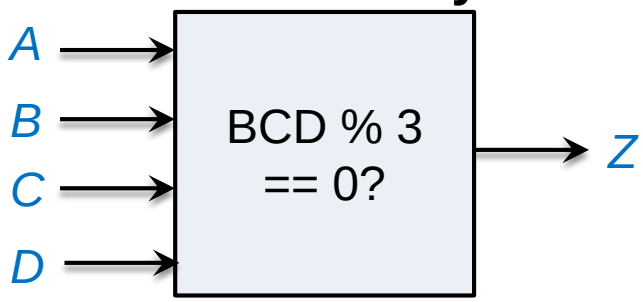
3. Realize it



Minterm & maxterm expansions

Example: Truth Table Construction

- **The output *Z* = 1 iff the 8-4-2-1 BCD number (*A*, *B*, *C*, *D*) is divisible by 3**



Decimal digit	8-4-2-1 code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

1. Construct the truth table

ABCD	*Z*
0000	1
0001	0
0010	0
0011	1
0100	0
0101	0
0110	1
0111	0
1000	0
1001	1
1010	X
1011	X
1100	X
1101	X
1110	X
1111	X

2. Simplify the function

$Z(A, B, C, D)$
$= \Sigma\, m(0, 3, 6, 9) +$
$\Sigma\, d(10, 11, 12, 13, 14, 15)$

$\Rightarrow$ Logic minimization using Karnaugh map (Unit 5)

3. Realize it

Minterm & maxterm expansions

Binary Adders and Subtractor

Divide-and-conquer

1-Bit Half Adder (HA)

- **Addition table:**

```
 0    0    1    1
+0   +1   +0   +1
--   --   --   --
00   01   01   10
```

carry sum

1. Construct the truth table

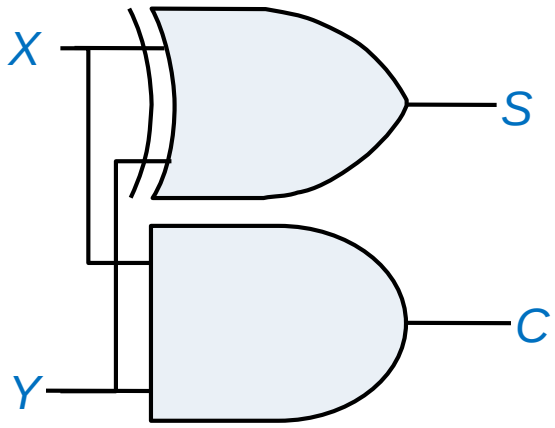
X	Y	Z		C	S
0	0	0	0	0	0
0	1	0	1	0	1
1	0	0	1	0	1
1	1	1	0	1	0

2. Simplify the function

$C = XY$

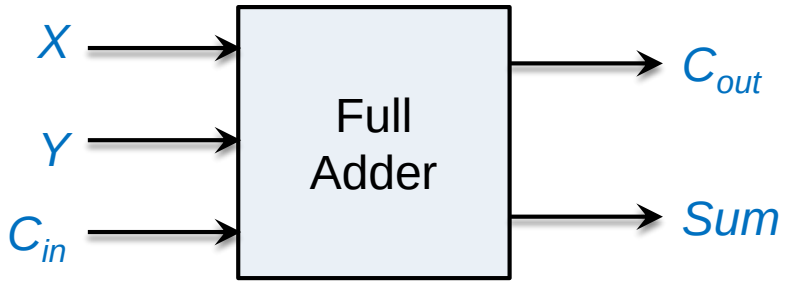
$S = X'Y + XY' = X \oplus Y$

3. Realize it



Minterm & maxterm expansions

1-Bit Full Adder (FA)



2. Simplify the function

$$C_{out} = X'YC_{in} + XY'C_{in} + XYC_{in}' + XYC_{in}$$
$$= XY + XC_{in} + YC_{in}$$
$$S = X'Y'C_{in} + X'YC_{in}' + XY'C_{in}' + XYC_{in}$$
$$= X \oplus Y \oplus C_{in}$$

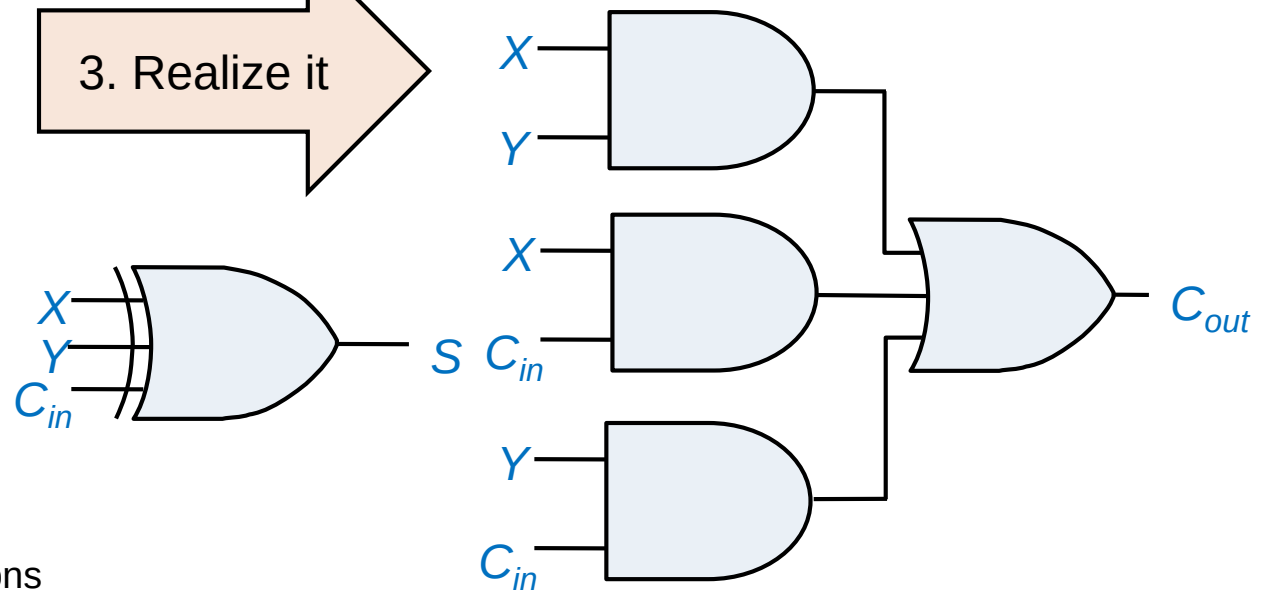
1. Construct the truth table

$$\begin{array}{r} X \\ Y \\ +\quad C_{in} \\ \hline C_{out}\ \ S \end{array}$$

X	Y	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Minterm & maxterm expansions

3. Realize it



Designing a 4-Bit Parallel Adder (1/3)

- $\boldsymbol{A} = (A_3A_2A_1A_0)$, $\boldsymbol{B} = (B_3B_2B_1B_0)$

$$\begin{array}{rcccc} & A_3 & A_2 & A_1 & A_0 \\ + & B_3 & B_2 & B_1 & B_0 \\ \hline C_4 & S_3 & S_2 & S_1 & S_0 \end{array} \quad C_0$$

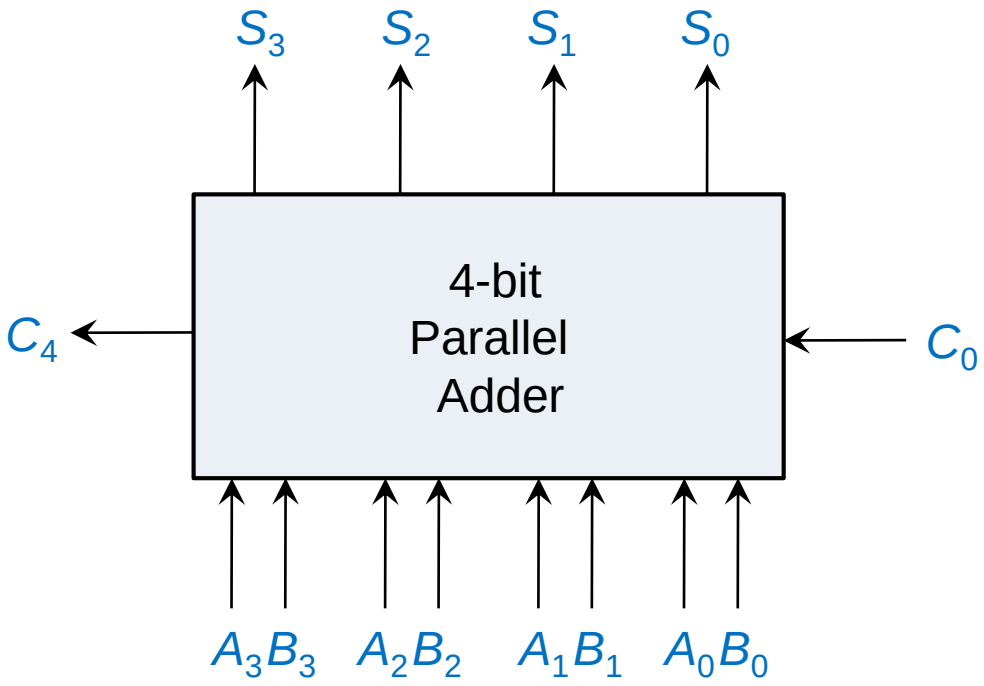
Carry out ← Carry in

$\Leftrightarrow$ $\boldsymbol{A + B}$

- **e.g.**,

$$\begin{array}{rcccc} 1 & 0 & 1 & 1 & 0 \\ & 1 & 0 & 1 & 1 \\ & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 1 & 0 \end{array} \quad \begin{array}{l} \leftarrow \text{Carries} \\ \\ \\ \leftarrow C_0 (=0) \end{array}$$

- **How?**
  - Use a single truth table?
  - $\Rightarrow$ Too big!



Minterm & maxterm expansions

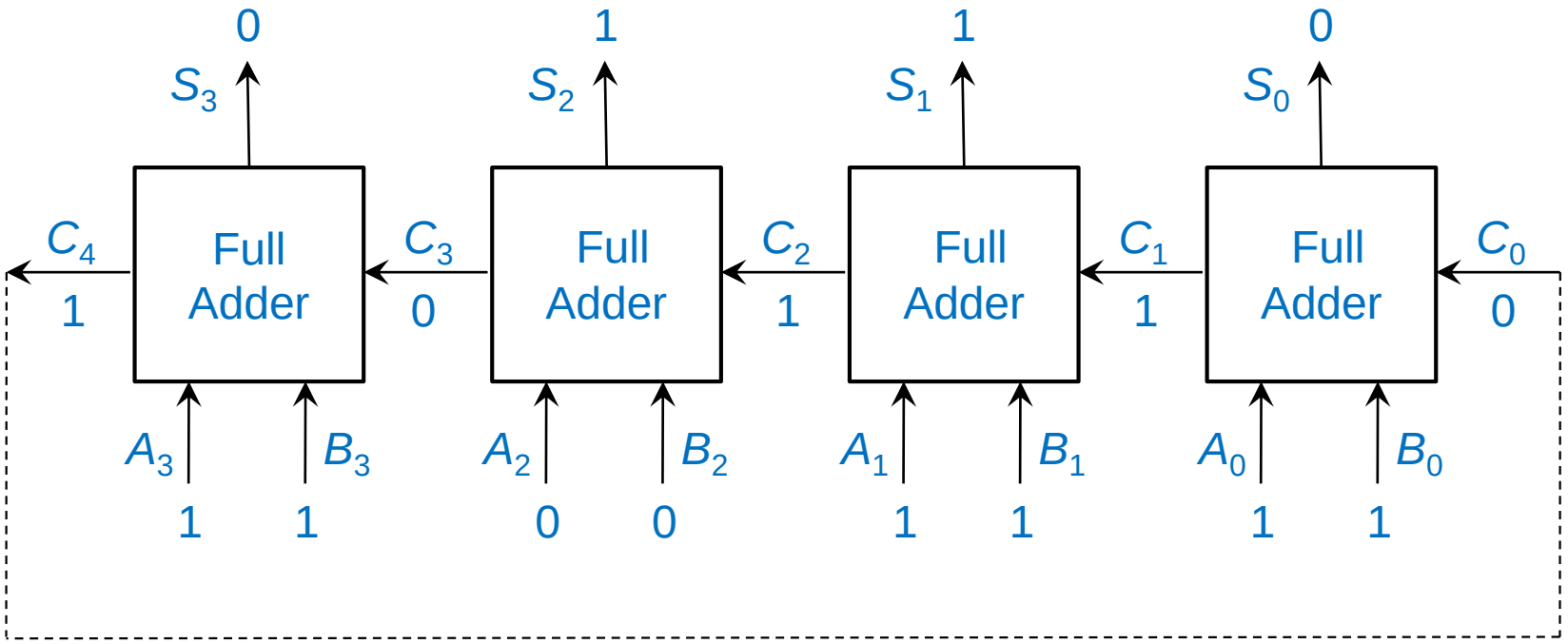
Designing a 4-Bit Parallel Adder (2/3)

- **Decompose the 4 bit adder into four modules**
  - Each module adds two bits and a carry ⇒ use full adder
- **Extend to negative numbers**
  - Consider one's complement (What if two's complement?)
    1. Add just as if all numbers were positive
    2. End-around carry: Add the carry out back to the rightmost bit
  - How to detect overflow?
    - Check sign!
    - (+) + (+) ⇒ (-) or (-) + (-) ⇒ (+)

Add(+*A*, +*B*) *A*+*B* < 2^{n-1}		Add(+*A*, -*B*) *B* > *A*		Add(-*A*, +*B*) *B* > *A*		Add(-*A*, -*B*) *A*+*B* ≤ 2^{n-1}	
+3	0011	+5	0101	−5	1010	−3	1100
+4	0100	−6	1001	+6	0110	−4	1011
+7	0111	−1	1110	+1	(1)0000	−7	(1)0111
					1		1
					0001		1000

Minterm & maxterm expansions

Designing a 4-Bit Parallel Adder (3/3)



- **Overflow detection?**

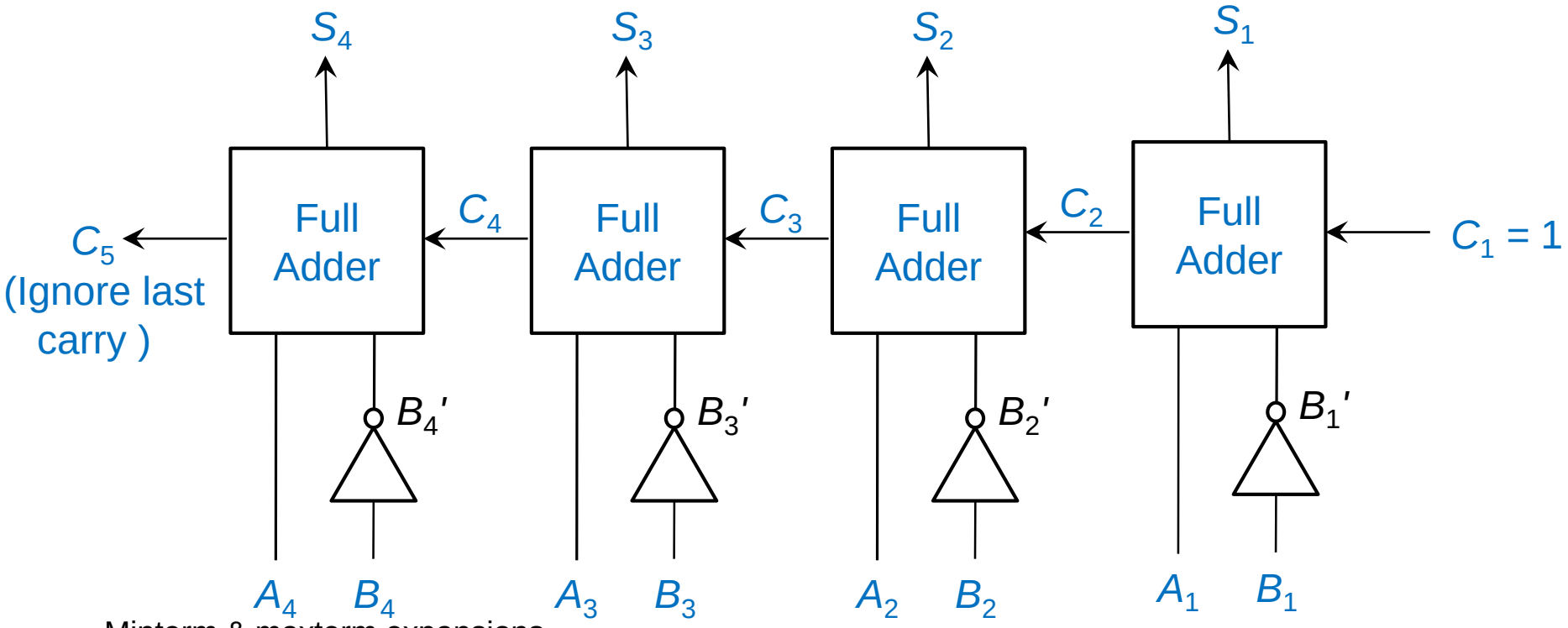
A	B	S	A_3	B_3	S_3	Overflow
+	+	−	0	0	1	1
−	−	+	1	1	0	1

$$V = A_3'B_3'S_3 + A_3B_3S_3'$$

Minterm & maxterm expansions

Designing a Binary Subtractor (1/2)

- **Consider *A* – *B* = *A* + (-*B*) in 2's complement**
  - $A - B = A + (-B) = A + B^* = A + B\_ + 1$
  - Convert *B* to 2's complement: inverse and then add 1
  - Discard the carry from the sign bit



Minterm & maxterm expansions

Designing a Binary Subtracter (2/2)

- **Or design a full subtracter**
  - $X - Y = D$ (difference), B: borrow
  - DIY!