CDA 3200 Digital Systems

Instructor: Dr. Janusz Zalewski
Developed by: Dr. Dahai Guo

Spring 2012

Outline

- Combinational Logic Design Using a Truth Table
- Minterm and Maxterm Expansions
- General Minterm and Maxterm Expansions
- Incompletely Specified Functions
- Examples of Truth Table Construction
- Design of Binary Adders and Subtracters

Combinational Logic Design Using a Truth Table (1/5)

- Sometimes, it is easier to first construct a truth table before developing the logic expression and design the logic circuit.
- The logic expression can be written in form of sum-of-products or product-of-sums, depending on how to interpret the truth table.

Combinational Logic Design Using a Truth Table (2/5)

- Any combination of 011, 100, 101, 110 or 111 can make f=1
  - ABC are 011 → A’BC=1
  - ABC are 100 →AB’C’=1
  - ABC are 101 →AB’C=1
  - ABC are 110 →ABC’=1
  - ABC are 111 →ABC=1

A B C	*dec*	f	f’
0 0 0	*0*	0	1
0 0 1	*1*	0	1
0 1 0	*2*	0	1
0 1 1	*3*	1	0
1 0 0	*4*	1	0
1 0 1	*5*	1	0
1 1 0	*6*	1	0
1 1 1	*7*	1	0

Combinational Logic Design Using a Truth Table (3/5)

- Therefore, the logic expression is
  - f=A'BC+AB'C'+AB'C+ABC'+ABC
  - =A'BC+AB'+AB
  - =A'BC+A
  - =(A'+A)(A+BC)
  - =A+BC

Combinational Logic Design Using a Truth Table (4/5)

- Any combination of 000, 001, or 010 can make f'=1
  - ABC are 000 → A'B'C'=1
  - ABC are 001 →A'B'C=1
  - ABC are 010 →A'BC'=1

A B C	*dec*	f	f'
0 0 0	*0*	0	1
0 0 1	*1*	0	1
0 1 0	*2*	0	1
0 1 1	*3*	1	0
1 0 0	*4*	1	0
1 0 1	*5*	1	0
1 1 0	*6*	1	0
1 1 1	*7*	1	0

Combinational Logic Design Using a Truth Table (5/5)

- Therefore, the logic expression for f’ is
  - f’=A’B’C’+A’B’C+A’BC’
  - (f’)’=(A’B’C’+A’B’C+A’BC’)’
  - f=(A’B’C’)’(A’B’C)’(A’BC’)’
  - =(A+B+C)(A+B+C’)(A+B’+C)

Minterm and Maxterm Expansions (1/10)

- A literal is a variable or its complement.
- A minterm of ***n*** variables is a product of ***n*** literals in which each variable appears once in either true or complemented form, but not both.
  - For a system with 3 variables
  - ABC is a minterm
  - AB'C' is a minterm
  - A'C' is NOT a minterm

Minterm and Maxterm Expansions (2/10)

- A minterm is designated $\boldsymbol{m_i}$, where ***i*** is the decimal value of the binary string of the variables.
  - ABC m_7
  - A'B'C' m_0
  - ABC' m_6

Minterm and Maxterm Expansions (3/10)

- The truth table of a logic function can be represented by a sum of minterms and in this case it is called a minterm expansion or a standard sum of products

Minterm and Maxterm Expansions (4/10)

- f=A’BC+AB’C’+AB’C+ABC’+ABC
- f=$m_3+m_4+m_5+m_6+m_7$
- f(A,B,C)= $\sum m(3,4,5,6,7)$

A B C	*dec*	f	f’
0 0 0	*0*	0	1
0 0 1	*1*	0	1
0 1 0	*2*	0	1
0 1 1	*3*	1	0
1 0 0	*4*	1	0
1 0 1	*5*	1	0
1 1 0	*6*	1	0
1 1 1	*7*	1	0

Minterm and Maxterm Expansions (5/10)

- Given a truth table, if the output of a certain row is 1, the corresponding minterm must be present in the logic expression.

Minterm and Maxterm Expansions (6/10)

- A maxterm of ***n*** variables is a sum of ***n*** literals.
  - In a system with three variables
  - (A+B+C) is a maxterm
  - (A’+B’+C) is a maxterm
  - (A’+B) is not maxterm
- A maxterm is designated M_i, where ***i*** is the decimal value of the <u>***complement***</u> of the binary string.

Minterm and Maxterm Expansions (7/10)

- f=(A+B+C)(A+B+C')(A+B'+C)
- f=$M_0M_1M_2$
- f= $\prod M(0,1,2)$

A B C	*dec*	f	f'
0 0 0	*0*	0	1
0 0 1	*1*	0	1
0 1 0	*2*	0	1
0 1 1	*3*	1	0
1 0 0	*4*	1	0
1 0 1	*5*	1	0
1 1 0	*6*	1	0
1 1 1	*7*	1	0

Minterm and Maxterm Expansions (8/10)

- Converting a general logic expression into a minterm expansion
  - Through repeatedly applying X+X'=1
  - f=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'bcd+a'bcd+ abcd'+ab'cd'
  - Note: a minterm expression is not necessary the simplest expression.

Minterm and Maxterm Expansions (9/10)

- Converting a general logic expression into a maxterm expression
  - Through repeatedly applying $XX'=0$

Minterm and Maxterm Expansions (10/10)

- When comparing two logic expressions, you can convert both into their minterm expressions and then compare.
  - Example:
  - a’c+b’c’+ab   and   a’b’+bc+ac’

General Minterm and Maxterm Expansion (1/3)

- A minterm expansion for an n variable function can be represented as a 2^n long vector
  - Example:
  - F(A,B,C)= $a_0 m_0 + a_1 m_1 ... a_6 m_6 + a_7 m_7 = \sum_{i=0}^{7} a_i m_i$
  - If a_i=1, m_i is present in the expression.

General Minterm and Maxterm Expansion (2/3)

- Similarly, a maxterm expansion for an n variable function can also be represented as a 2^n long vector
  - F(A,B,C)= $(a_0 + M_0)(a_1 + M_1)...(a_6 + M_6)(a_7 + M_7) = \prod_{i=0}^{7}(a_i + M_i)$
  - If a_i is 0, M_i is present in the expression. Why?

General Minterm and Maxterm Expansion (3/3)

- Given two different minterm expansions of n variables

$$f_1 = \sum_{i=0}^{2^n-1} a_i m_i$$

$$f_2 = \sum_{j=0}^{2^n-1} b_j m_j$$

$$f_1 f_2 = \left(\sum_{i=0}^{2^n-1} a_i m_i \right)\left(\sum_{j=0}^{2^n-1} b_j m_j \right) = \sum_{i=0}^{2^n-1} \sum_{j=0}^{2^n-1} a_i b_j m_i m_j = \sum_{i=0}^{2^n-1} a_i b_i m_i$$

Incompletely Specified Functions (1/5)

- Sometimes, not all the combinations of the inputs are considered in the circuit.
- Unconsidered combinations are referred to as “do not care” terms.
- In the truth table, the outputs for “do not care” terms are designated ‘X’

Incompletely Specified Functions (2/5)

- We could ignore the "do not care" terms, then the logic expression is
- F=A'B'C'+A'BC+ABC
- =A'B'C'+BC

A	B	C	F
0	0	0	1
0	0	1	X
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	1

Incompletely Specified Functions (3/5)

- It does not matter, if we assign 1/0 to Xs
  - F=A'B'C'+BC+A'B'C
  - =A'B'+BC   Simpler expression

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Incompletely Specified Functions (4/5)

- Sometimes, assigning 1 to X’s may contribute to simplifying the logic expression.

Incompletely Specified Functions (5/5)

- In a minterm expansion, the “do not care” terms are denoted ***d***

$$F = \sum m(0,3,7) + \sum d(1,6)$$

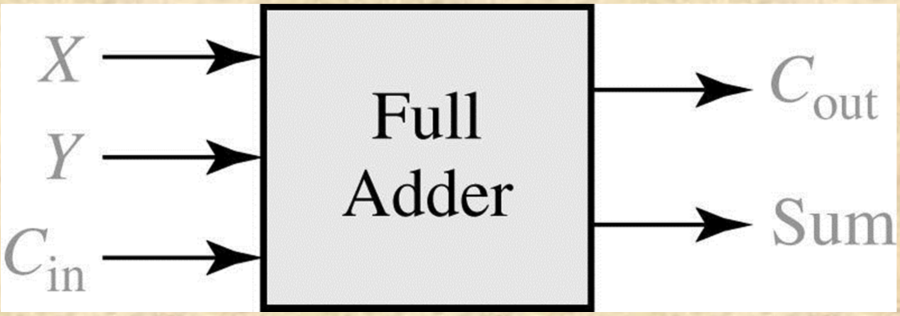
- In a maxterm expansion, the “do not care” terms are denoted ***D***

$$F = \prod M(2,4,5) + \prod D(1,6)$$

Design of Binary Adders and Subtracters (1/7)

- Full Adders

X	Y	C_{in}	Dec	C_{out}	Sum
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	2	0	1
0	1	1	3	1	0
1	0	0	4	0	1
1	0	1	5	1	0
1	1	0	6	1	0
1	1	1	7	1	1



$C_{out}=m_3+m_5+m_6+m_7$

$Sum=m_1+m_2+m_4+m_7$

Design of Binary Adders and Subtracters (2/7)

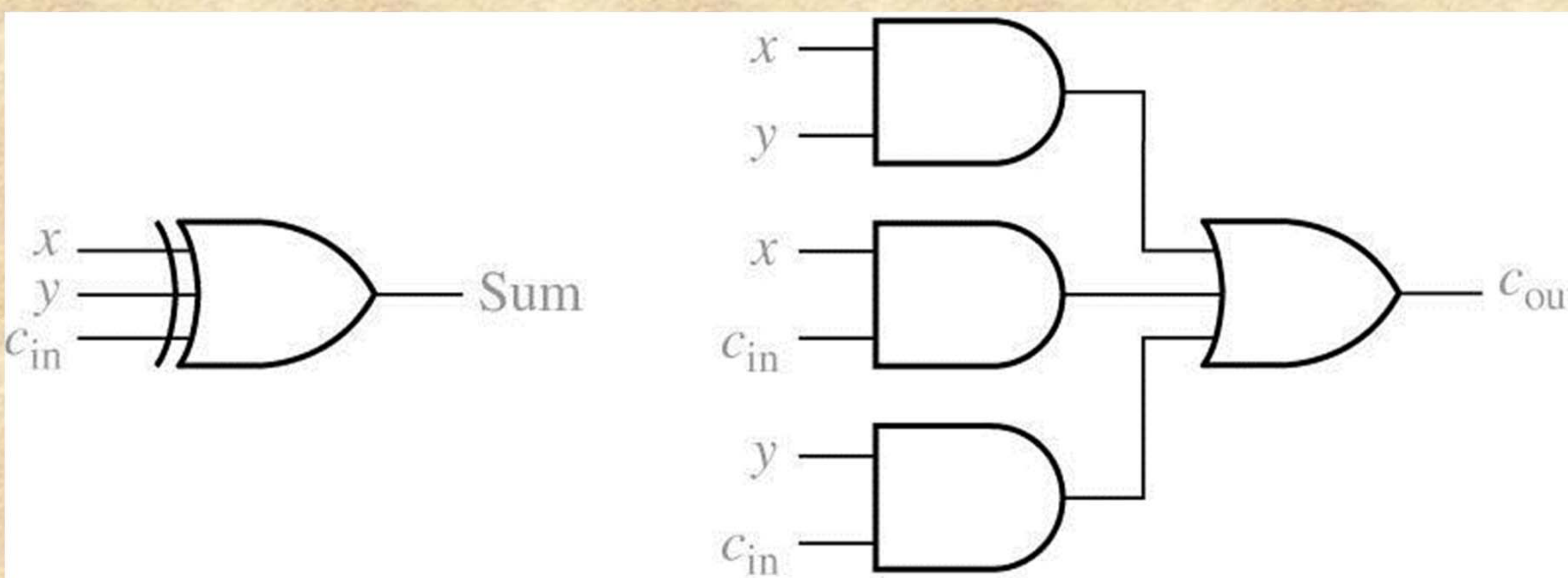
- Sum=m1+m2+m4+m7
  - $Sum=X'Y'C_{in}+X'YC_{in}'+XY'C_{in}'+XYC_{in}$
  - $=X'(Y'C_{in}+YC_{in}')+X(Y'C_{in}'+YC_{in})$
  - $=X'(Y \text{ xor } C_{in})+X(Y \text{ xor } C_{in})'$
  - $=X \text{ xor } (Y \text{ xor } C_{in})=X \text{ xor } Y \text{ xor } C_{in}$
- C_{out}=m3+m5+m6+m7
  - $C_{out}=X'YC_{in}+XY'C_{in}+XYC_{in}'+XYC_{in}$
  - $=YC_{in}+XC_{in}+XY$

Design of Binary Adders and Subtracters (3/7)

Full Adder

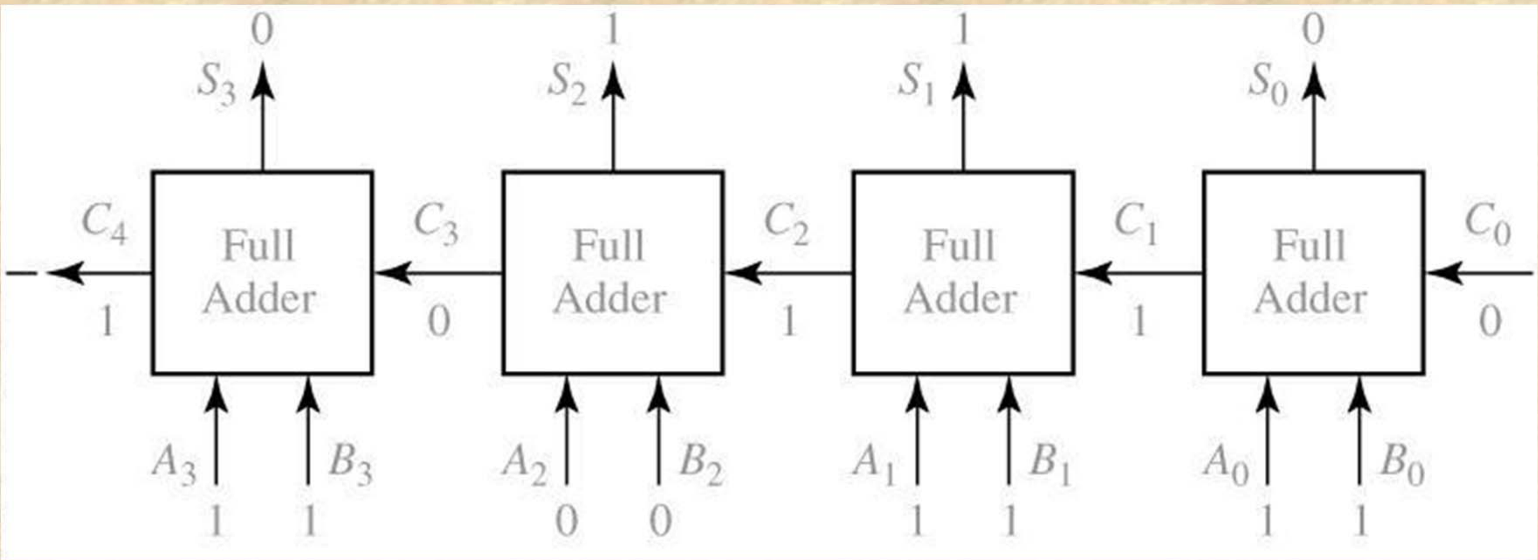
Sum=X xor Y xor C_{in}

C_{out}=YC_{in}+XC_{in}+XY



Design of Binary Adders and Subtracters (4/7)

- Four full adders can be used to make a 4-bit binary adder



Design of Binary Adders and Subtracters (5/7)

- When adding two signed number, overflow must be considered.
  - Adding two positive numbers gives a negative number: $A_3B_3S_3'$
  - Adding two negative numbers gives a positive number: $A_3'B_3'S_3$
  - $V=A_3'B_3'S_3+A_3B_3S_3'$ can be used to reflect if overflow occurs

Design of Binary Adders and Subtracters (6/7)

- Binary subtracter using full adders
  - Remember two's complement
    - Reverse all the bits
    - Add 1

Design of Binary Adders and Subtracters (7/7)