06-Applications
Text: Unit 4

ECEGR/ISSC 201
Digital Operations and Computations
Winter 2011

Overview

- Conversion of English Statements to Boolean Expressions
- Conversion of Truth Tables
- Binary Adders and Subtracters

Applications

- Interested in conversion of text statements to and from logical expressions
- Example
  The light is to turn on if button A and button B are pushed or if button A is pushed and button B is not pushed or if button C is pushed.

Conversion of Text

1. Identify the logical phrases
2. Assign variables to each phrase
3. Write the logical expression

Conversion of Text

- I will pass this course if I study hard and I do not have to work a full time job or if Professor Louie is an easy grader.

Conversion of Text

- I will pass this course if I study hard and I do not have to work a full time job or if Professor Louie is an easy grader.

Conversion of Text

- I will pass this course if I study hard and I do not have to work a full time job or if Professor Louie is an easy grader.
- Define variables
  - P = 1 if you pass the course
  - A = 1 if you study hard
  - B = 1 if you have to work full time
  - C = 1 if Prof. Louie is an easy grader

Conversion of Text

- I will pass this course if I study hard and I do not have to work a full time job or if Professor Louie is an easy grader.
- Define variables
  - P = 1 if you pass the course
  - A = 1 if you study hard
  - B = 1 if you have to work full time
  - C = 1 if Prof. Louie is an easy grader
- P = AB' + C

Conversion of Text

- Find the logical expression for:
  The alarm will sound if and only if the alarm switch is on and the door is not closed or it is after 6pm and the window is not closed.

Example

1. Identify logical phrases

   The alarm will sound if and only if the alarm switch is on and the door is not closed or it is after 6pm and the window is not closed.

Example

2. Assign variables

   The alarm will sound if and only if the alarm switch is on and the door is not closed or it is after 6pm and the window is not closed.

Z = alarm will sound
A = the alarm switch is on
B' = the door is not closed
C = it is after 6pm
D' = the window is not closed

Example

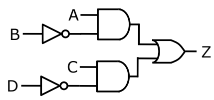
3. Write the logical expression

   The alarm will sound if and only if the alarm switch is on and the door is not closed or it is after 6pm and the window is not closed.

Z = alarm will sound
A = the alarm switch is on
B' = the door is not closed
C = it is after 6pm
D' = the window is not closed
Z = AB' + CD'

Conversion of Text

- The logic can be implemented as:
  Z = AB′ + CD′



Converting Truth Tables

- Assume you are given the following truth table to implement as a circuit:

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

Converting Truth Tables

- First, find all the combinations that result in F = 1

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

Converting Truth Tables

- F = 1 when A = 0 and B = 1 and C =1, this can be expressed as A′BC

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

Converting Truth Tables

- F = 1 when A = 1 and B = 0 and C =0, this can be expressed as AB′C′

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

Converting Truth Tables

- What can this row be expressed as?

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

Converting Truth Tables

- What can this row be expressed as? F = 1 when A = 1 and B = 0 and C =1, this can be expressed as AB'C

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

Converting Truth Tables

- What can this row be expressed as?

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

Converting Truth Tables

- What can this row be expressed as? F = 1 when A = 1 and B = 1 and C =0, this can be expressed as ABC'

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

Converting Truth Tables

- Finally: F = 1 when A = 1 and B = 1 and C =1, this can be expressed as ABC

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

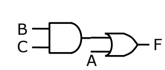
Converting Truth Tables

- F = 1 if A'BC or AB'C' or AB'C or ABC' or ABC
  => F = A'BC + AB'C' + AB'C + ABC' + ABC

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

Converting Truth Tables

- Next, simplify:
- F = A'BC + AB'C' + AB'C + ABC' + ABC
  = A'BC + AB'(C' + C) + AB(C' + C)
  = A'BC + AB' + AB
  = A'BC + A
  = A + BC



Converting Truth Tables

- We could also do the same procedure by
  - Writing the terms that result in F =0
  - Writing terms in which F′ = 1 or F′=0

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

Converting Truth Tables

- Terms that result in F = 0
  - F = 0 if A =0, B =0 and C =0
  - A + B + C = 0 iff A = B = C =0

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

Converting Truth Tables

- Terms that result in F = 0
  - F = 0 if A =0, B =0 and C =0
  - F = 0 if A = 0, B = 0 and C = 1
  - F = 0 if A = 0, B = 1 and C = 0
- Combining: (A+B+C)(A+B+C′)(A+B′+C) will equal 0 iff any of the above combinations occur
- All other combinations result in F = 1
- This must equal the previously derived expression:
- (A+B+C)(A+B+C′)(A+B′+C) = A + BC

Example

- Write a simplified expression for F in SoP form:

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

Example

- Write a simplified expression for F in SoP form:
- Identifying rows with F =1
  - A′B′C′
  - A′B′C
  - AB′C′
  - AB′C

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

Example

- Write a simplified expression for F in SoP form:
- F =A′B′C′ + A′B′C + AB′C′ + AB′C

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

Example

- Minimizing:
- F =A′B′(C′ + C) + AB′(C′ + C)
- F = A′B′ + AB′ = B′(A + A′) = B′

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

Example

- Write a simplified expression for F in PoS form

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

Example

- Write a simplified expression for F in PoS form

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

Example

- Write a simplified expression for F in PoS form
- A + B′ +C
- A + B′ + C′
- A′ + B′ + C
- A′ + B′ + C′

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

Example

F = (A+B′+C)(A +B′+C′)(A′+B′+C)(A′+B′+C′)

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

Example

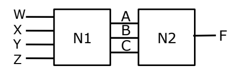
F = (A+B′+C)(A +B′+C′)(A′+B′+C)(A′+B′+C′)
F = (A+B′+C)(A +B′+C′)(A′+B′)
F = (A + B′)(A′+B′)
F = B′

Example

- The final answer in SoP or PoS form must give the same truth table
- Previous example is a special case in which the PoS and SoP forms are the same (B')

Incompletely Specified Functions

- Complex circuits are often composed of subcircuits
- Assume that not all combinations of A, B and C are output by N1 (i.e., there is no combination of W, X, Y, Z that result in certain A, B, C combinations occurring)



Incompletely Specified Functions

- Assume that the following ABC combinations do not occur:
  - 001
  - 110
- How do we design the circuit?

Incompletely Specified Functions

- Truth table with unspecified values represented as "X"
- These are "don't care" terms

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

- Don't cares need to specified as either a 1 or 0
- We should assign a value that reduces the complexity of the circuit
- Try each of the four combinations

Incompletely Specified Functions

- With 0 assigned to the don't cares:
  - F = A'B'C' + A'BC + ABC = A'B'C' + BC

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

- With 1 to the first, 0 to the second don't care
  - F = A'B'C' + A'B'C + A'BC + ABC = A'B' + BC

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

- With 0 to the first, 1 to the second don't care
  - F = A'B'C' + A'BC + ABC' + ABC= A'B'C'+ A'BC+AB

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

Incompletely Specified Functions

- With 1 to assigned to both don't cares
  - F = A'B'C' + A'B'C+A'BC + ABC' + ABC = A'B' + BC + AB

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	1
1	1	1	1

Incompletely Specified Functions

- Which is the simplest?
  F = A'B'C' + BC
  F = A'B' + BC
  F = A'B'C'+ A'BC+AB
  F = A'B' + BC + AB
- Second choice (first X =1, second X = 0)

Truth Table Examples

- Design a binary adder for two, 1-bit numbers
  - A two bit output is needed

Truth Table Example

- Design a binary adder for two, 1-bit numbers
- Let a, b be the numbers to be added
- A table for the input/output combinations is:

a	b	Sum
0	0	00
0	1	01
1	0	01
1	1	10

Truth Table Example

- This can easily be converted into a truth table:

a	b	Sum
0	0	00
0	1	01
1	0	01
1	1	10

A	B	X (first digit)	Y (second digit)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Truth Table Example

- Write the expressions for X and Y

A	B	X	Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Truth Table Example

- Write the expressions for X and Y
  - X = AB
  - Y = A'B + AB' = A⊕B
- This is simple, which is why we use 2's complement addition

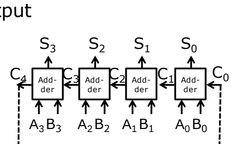
A	B	X	Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Binary Adders and Subtracters

- Goal: design an adder that adds two 4-bit unsigned binary numbers
- Output: 4 bit output and a carry
- One approach: develop a truth table (of what dimension?) and determine the expression
  - Complicated
- Our approach: modularize--design a module that adds two bits and a carry, connect four together

Binary Adders and Subtracters

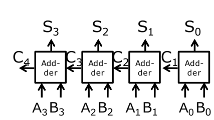
- A_x: bit of first number
- B_x: bit of second number
- C_x: carry output
- S_x: sum output



If adding 1's complement numbers, otherwise ignore C_0 and discard C4

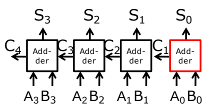
Binary Adders and Subtracters

- Example 2's complement addition
  - A = 1011 => A_0 = 1; A_1 = 1; A_2 = 0; A_3 = 1
  - B = 1110 => B_0 = 0; B_1 = 1; B_2 = 1; B_3 = 1



Binary Adders and Subtracters

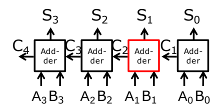
- Example
  - $A_0 + B_0 = 1 + 0 \Rightarrow S_0 = 1; C_1 = 0$



A = 1011; B = 1110

Binary Adders and Subtracters

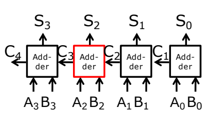
- Example
  - $A_0 + B_0 = 1 + 0 \Rightarrow S_0 = 1; C_1 = 0$
  - $C_1 + A_1 + B_1 = 0 + 1 + 1 \Rightarrow S_1 = 0; C_2 = 1$



A = 1011; B = 1110

Binary Adders and Subtracters

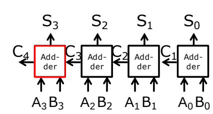
- Example
  - $A_0 + B_0 = 1 + 0 \Rightarrow S_0 = 1; C_1 = 0$
  - $C_1 + A_1 + B_1 = 0 + 1 + 1 \Rightarrow S_1 = 0; C_2 = 1$
  - $C_2 + A_2 + B_2 = 1 + 0 + 1 \Rightarrow S_2 = 0; C_3 = 1$



A = 1011; B = 1110

Binary Adders and Subtracters

- Example
  - $A_0 + B_0 = 1 + 0 \Rightarrow S_0 = 1; C_1 = 0$
  - $C_1 + A_1 + B_1 = 0 + 1 + 1 \Rightarrow S_1 = 0; C_2 = 1$
  - $C_2 + A_2 + B_2 = 1 + 0 + 1 \Rightarrow S_2 = 0; C_3 = 1$
  - $C_3 + A_3 + B_3 = 1 + 1 + 1 \Rightarrow S_3 = 1; C_4 = 1$



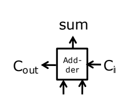
A = 1011; B = 1110

Binary Adders and Subtracters

- Did the counter operate properly?
- The decimal equivalent of the 2s complement numbers: $N^* = 2^n - N$
  - A = 1011 => 11 = 16 – N => N = 5 => A = -5
  - B = 1110 = -2
- A + B = -7 => 1001
- Our solution is (ignoring the final carry over)
  - $S_3S_2S_1S_0$ = 1001
- Success!

Binary Adders and Subtracters

- Need to determine the expressions for S_x and C_x
- We can build a truth table from the various cases of A_x, B_x, and C_x values
- Consider the generic adder

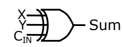


Binary Adders and Subtracters

X	Y	C_{in}	C_{out}	Sum
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

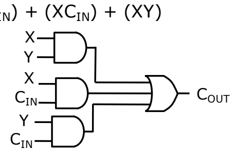
Binary Adders and Subtracters

- The resulting expression is:
  - Sum = $X'Y'C_{IN} + X'YC'_{IN} + XY'C'_{IN} + XYC_{IN}$
  - Sum = $X'(Y'C_{IN} + YC'_{IN}) + X(Y'C'_{IN} + YC_{IN})$
  - Sum = $X'(Y \oplus C_{IN}) + X(Y \oplus C_{IN})'$
  - Sum = $X \oplus Y \oplus C_{IN}$



Binary Adders and Subtracters

- The resulting expression is:
  - $C_{OUT} = X'YC_{IN} + XY'C_{IN} + XYC'_{IN} + XYC_{IN}$
  - $C_{OUT} = X'YC_{IN} + XY'C_{IN} + XYC'_{IN} + XYC_{IN}$ (add three more times to simplify)
  - $C_{OUT} = (X'YC_{IN} + XYC_{IN}) + (XY'C_{IN} + XYC_{IN}) + (XYC'_{IN} + XYC_{IN})$
  - $C_{OUT} = (YC_{IN}) + (XC_{IN}) + (XY)$

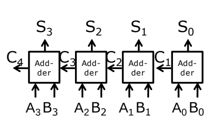


Binary Adders and Subtracters

- When working with numbers in 1's or 2's complement form, errors due to overflow can be detected by checking if
  - two positive numbers result in a negative number
  - two negative numbers result in a positive number
- Recall: an overflow error occurs when in an n-bit system, the correct display of the answer requires more than n bits

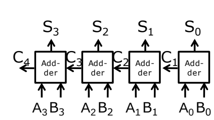
Binary Adders and Subtracters

- Find an expression for V, which is equal to 1 when an overflow error occurs



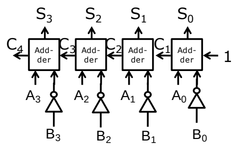
Binary Adders and Subtracters

- Find an expression for V, which is equal to 1 when an overflow error occurs
  - $V = A_3'B_3'S_3 + A_3B_3S_3'$



Binary Adders and Subtracters

- Subtracting numbers is easily done by adding the 1's or 2's complement of the number to be subtracted
- Example: A – B
  - Recall a 2's complement can be found by complementing each bit and adding 1



Binary Adders and Subtracters

- A subtracter circuit can also be designed
- Borrow and difference outputs for each column

X	Y	b	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1