Switching Circuits & Logic Design

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University

Fall 2013

§4 Applications of Boolean Algebra Minterm & Maxterm Expansions

Convex and Concave
M.C. Escher, 1955

Outline

- Conversion of English sentences to Boolean expressions
- 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

Conversion of English Sentences to Boolean Expressions

- Steps in designing a single-output combinational circuit:
  1. Find a switching function that specifies the desired behavior of the circuit
     - Translate English sentences into Boolean equations
       - Associate a Boolean variable with each phrase having a value of "true" or "false"
  2. Find a simplified algebraic expression for the function
  3. Realize the simplified function using available logic elements

Conversion of English Sentences to Boolean Expressions

Example

- Mary watches TV (F) if it is Monday night (A) and she has finished her homework (B)
  - F=1 iff "Mary watches TV" is true
  - A=1 iff "it is Monday night"
  - B=1 iff "she has finished her homework"

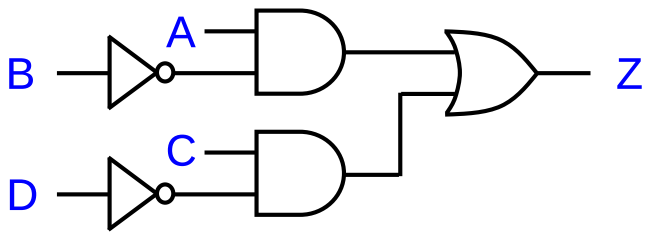
- F = A·B (equation)
  - More accurately A·B ⇒ F (formula)

Conversion of English Sentences to Boolean Expressions

Example

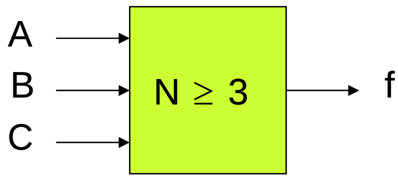
- The alarm will ring (Z) iff the alarm switch is turned on (A) and the door is not closed (B'), or it is after 6pm (C) and the window is not closed (D')

- Z = AB'+CD' (equation)
  - Z⇔(AB'+CD'), Z≡(AB'+CD') (formula)



Combinational Logic Design Using a Truth Table

- For a 3-input Boolean function f(A,B,C) with
  f=1 if $N \geq 3$ and
  f=0 if $N < 3$,
  where $N = A\times2^2+B\times2^1+C\times2^0$



ABC	f	f'
000	0	1
001	0	1
010	0	1
011	1	0
100	1	0
101	1	0
110	1	0
111	1	0

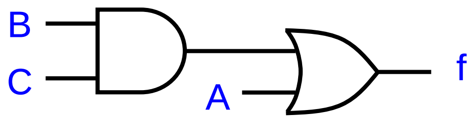
f = A'BC+AB'C'+AB'C+ABC'+ABC

(minterm expansion)

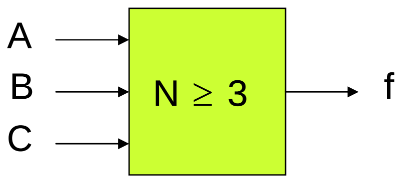
= A'BC+AB'+AB

= A'BC+A

= BC+A



Combinational Logic Design Using a Truth Table



A'B'C' ⇒ f=0
f=1 ⇒ A+B+C

ABC	f	f'
000	0	1
001	0	1
010	0	1
011	1	0
100	1	0
101	1	0
110	1	0
111	1	0

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

(maxterm expansion)

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

Alternative derivation by DeMorgan's law

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

- A **minterm** (**maxterm**) of n variables is a product (sum) of n literals in which each variable appears exactly once
  - Recall a **literal** is a variable or its complement

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

$m_i' = M_i$

Minterm and Maxterm Expansions

- Minterm expansion (or called **standard sum of products**)

$f = A'BC+AB'C'+AB'C+ABC'+ABC$

$= m_3 + m_4 + m_5 + m_6 + m_7$ (m-notation)

$= \sum m(3, 4, 5, 6, 7)$

- Maxterm expansion (or called **standard product of sums**)

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

$= M_0M_1M_2$ (M-notation)

$= \prod M(0, 1, 2)$

ABC	f	f'
000	0	1
001	0	1
010	0	1
011	1	0
100	1	0
101	1	0
110	1	0
111	1	0

Minterm and Maxterm Expansions Canonicity

- Minterm and maxterm expansions are **canonical** representations, that is, two functions are equivalent iff they have the same minterm and maxterm expansions
  - Recall truth tables are also a canonical representation of Boolean functions

Minterm and Maxterm Expansions Complementation

ABC	f	f'
000	0	1
001	0	1
010	0	1
011	1	0
100	1	0
101	1	0
110	1	0
111	1	0

$$f = m_3 + m_4 + m_5 + m_6 + m_7$$
$$= \sum m(3, 4, 5, 6, 7)$$

$$= M_0 M_1 M_2$$
$$= \prod M(0, 1, 2)$$

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

$$= M_3 M_4 M_5 M_6 M_7$$
$$= \prod M(3, 4, 5, 6, 7)$$

De Morgan's law

by $m_i' = M_i$

Minterm and Maxterm Expansions

Example

Find the minterm expansion of f = a'(b'+d)+acd'

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'b'c'd + a'b'cd + a'bc'd + a'bcd + abcd' + ab'cd'
= $\sum$m(0,1,2,3,5,7,10,14)
= $\prod$M(4,6,8,9,11,12,13,15)

Exercise: find the maxterm expansion directly (p.100)

Minterm and Maxterm Expansions

Example

Show that a'c+b'c'+ab = a'b'+bc+ac'

LHS:

a'c(b+b')+b'c'(a+a')+ab(c+c')
= a'bc + a'b'c+ ab'c'+ a'b'c'+abc+abc'
= $\sum$m(3,1,4,0,7,6)

RHS:

a'b'(c+c')+bc(a+a')+ac'(b+b')
= a'b'c+ a'b'c'+abc+ a'bc+abc'+ ab'c'
= $\sum$m(1,0,7,3,6,4)

LHS = RHS

General Minterm and Maxterm Expansions

- There are 2^{2^n} possible Boolean functions of n variables
  - There are 2^n minterms induced by n variables
  - For each minterm, function F can be 0 or 1

ABC	F
000	a_0
001	a_1
010	a_2
011	a_3
100	a_4
101	a_5
110	a_6
111	a_7

$$F = a_0m_0 + a_1m_1 + \cdots + a_7m_7 = \sum a_im_i$$

$$F = (a_0+M_0)(a_1+M_1) \cdots (a_7+M_7) = \prod(a_i+M_i)$$

$$F' = [\prod(a_i+M_i)]' = \sum a_i'm_i$$

$$F' = [\sum a_i'm_i]' = \prod(a_i'+M_i)$$

General Minterm and Maxterm Expansions

DESIRED FORM

GIVEN 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

General Minterm and Maxterm Expansions

DESIRED FORM

GIVEN FORM	Minterm Expansion of F	Maxterm Expansion of F	Minterm Expansion of F'	Maxterm Expansion of F'
F= $\sum m(3,4,5,6,7)$	______	$\prod M(0,1,2)$	$\sum m(0,1,2)$	$\prod M(3,4,5,6,7)$
F= $\prod M(0,1,2)$	$\sum m(3,4,5,6,7)$	______	$\sum m(0,1,2)$	$\prod M(3,4,5,6,7)$

Sets vs. Boolean Functions

- Representing and manipulating sets with Boolean algebra
  - Boolean functions can be used to represent sets
    - A Boolean function represents a set of minterms
    - Associate minterms with set elements
  - Boolean operations can be used to achieve set operations

Sets vs. Boolean Functions

Example

Let S = {a,b,c,d,e,f,g,h} be encoded with Boolean variables X,Y,Z as follows

S	X	Y	Z	F_A	F_B	F_C
a	0	0	0	1	0	0
b	0	0	1	1	1	1
c	0	1	0	0	0	0
d	0	1	1	0	1	1
e	1	0	0	1	1	0
f	1	0	1	1	1	0
g	1	1	0	0	1	0
h	1	1	1	0	1	0

The subsets A = {a,b,e,f}, B = {b,d,e,f,g,h}, C = {b,d} can be represented by Boolean functions $F_A=Y'$, $F_B=X+Z$, $F_C=X'Z$, respectively

The set $A \cap B$ can be represented by $F_A \cdot F_B$

The set $A \cup C'$ can be represented by F_A+F_C'

Sets vs. Boolean Functions

Isomorphism between sets and Boolean functions

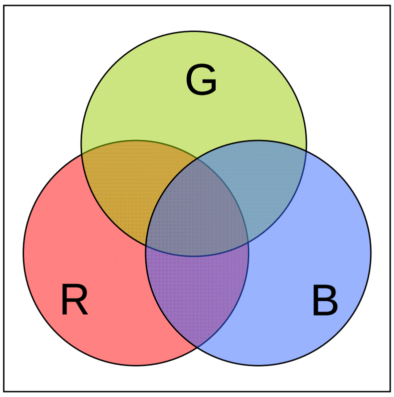
- Sets
  - A, B
  - $a_1 \in A$, $a_2 \notin A$
- Intersection
  - $A \cap B$
- Union
  - $A \cup B$
- Complement
  - A'

- Boolean functions
  - F_A, F_B
  - $F_A([a_1])=1$, $F_A([a_2])=0$

    Let $[a_i]$ be the binary codes of a_i
- AND
  - $F_A \cdot F_B$
- OR
  - $F_A + F_B$
- Complement
  - F_A'

Sets vs. Boolean Functions

Example



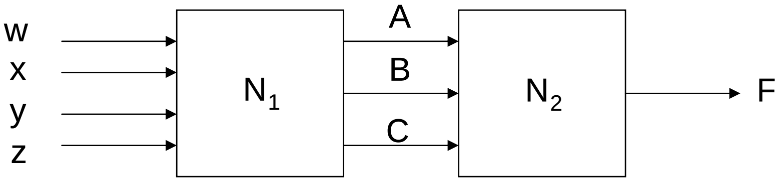
$$(R \cap G) \cup (R' \cap B) \cup (G \cap B) = (R \cap G) \cup (R' \cap B)$$

$$\Updownarrow$$

$$F_R F_G + F_R' F_B + F_G F_B = F_R F_G + F_R' F_B$$

(consensus theorem)

Incompletely Specified Functions



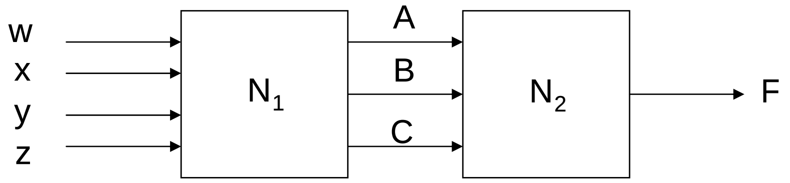
Assume (A,B,C) = (0,0,1) and (1,1,0) never occurs

ABC	F
000	1
001	x
010	0
011	1
100	0
101	0
110	x
111	1

don't cares
(F can be 0 or 1)

F is an **incompletely specified function** if it contains such don't care inputs

Incompletely Specified Functions



Assume (A,B,C) = (0,0,1) and (1,1,0) never occurs

ABC	F
000	1
001	x
010	0
011	1
100	0
101	0
110	x
111	1

don't cares

Don't cares can be exploited to minimize F:

Assign 0 to both x:

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

= A'B'C'+BC

Assign 1 to 1st x, 0 to 2nd x:

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

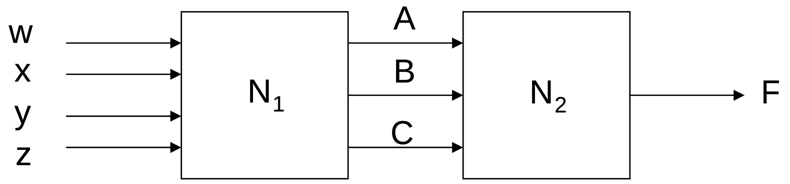
= A'B'+BC   Simpler!

Assign 1 to both x:

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

= A'B'+BC+AB

Incompletely Specified Functions



ABC	F
000	1
001	x
010	0
011	1
100	0
101	0
110	x
111	1

$F = \sum m(0,3,7) + \sum d(1,6)$ (don't care minterms)

$F = \prod M(2,4,5) \cdot \prod D(1,6)$ (don't care maxterms)

Examples of Truth Table Construction
Binary Codes

Decimal digit	8-4-2-1 code (BCD)	6-3-1-1 code	Excess-3 code (BCD+3)	2-out-of-5 code (good for error checking)	Gray code (good for low power and reliability)
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011
8	1000	1011	1011	10100	1001
9	1001	1100	1100	11000	1000

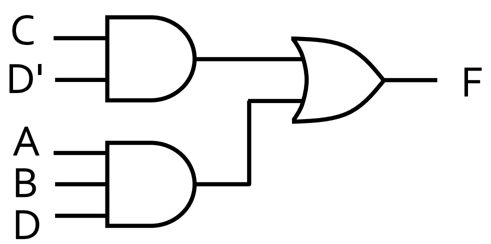
Examples of Truth Table Construction

- Error detector for 6-3-1-1 code

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

F(x)=1 indicates an error has occurred, i.e., x is an invalid 6-3-1-1 code

F = ∑m(2,6,10,13,14,15)
= A'B'CD'+A'BCD'+AB'CD'+ABCD'+ABC'D+ABCD
= A'CD'+ACD'+ABD
= CD'+ABD



Examples of Truth Table Construction

Multiples of 3 for 8-4-2-1 code

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

$$Z(a) = \begin{cases} X & \text{if a is not an 8-4-2-1 code} \\ 1 & \text{if a is a multiple of 3} \\ 0 & \text{if a is not a multiple of 3} \end{cases}$$

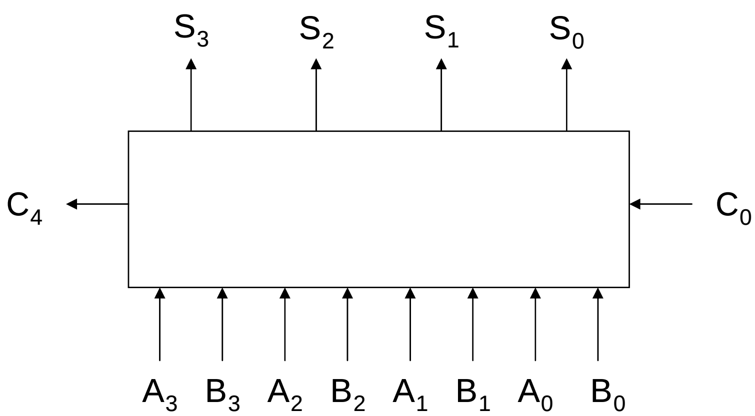
$$Z = \sum m(0,3,6,9) + \sum d(10,11,12,13,14,15)$$

We'll study how to minimize an incompletely specified function using Karnaugh maps in Unit 5

Design of Binary Adders

Adder design for 4-bit unsigned binary numbers

$$\begin{array}{lll} & A & A_3A_2A_1A_0 \\ & B & B_3B_2B_1B_0 \\ + & C_{in} & C_0 \\ \hline & & C_4S_3S_2S_1S_0 \end{array}$$



Method 1: Design from constructing a truth table of the whole system (difficult simplification and complex implementation)

Method 2: Design by constructing and composing local modules, i.e., full adders (simple and extendable to n-bit adder design; long circuit delay due to carry propagation)

Design of Binary Adders
By Construction from Truth Table

- A 2-bit adder example

A_1	A_0	B_1	B_0	C_0	C_2	S_1	S_0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	1	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	1	0	1	1
0	0	1	1	0	0	1	1
0	0	1	1	1	1	0	0
0	1	0	0	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1
0	1	1	0	1	1	0	0
0	1	1	1	0	1	0	0
0	1	1	1	1	1	0	1

A_1	A_0	B_1	B_0	C_0	C_2	S_1	S_0
1	0	0	0	0	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	0	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	0	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	0	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	0	0	1	1
1	1	0	0	1	1	0	0
1	1	0	1	0	1	0	0
1	1	0	1	1	1	0	1
1	1	1	0	0	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1

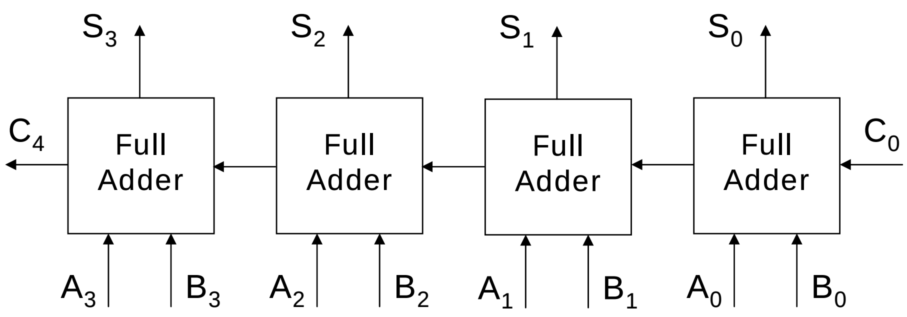
$$\begin{array}{rr} & A_1A_0 \\ & B_1B_0 \\ + & C_0 \\ \hline & C_2S_1S_0 \end{array}$$

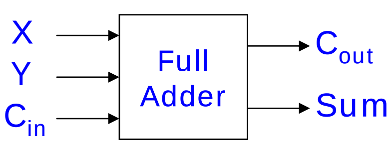
$S_0 = A_1'A_0'B_1'B_0'C_0 + \cdots$

$S_1 = A_1'A_0'B_1'B_0C_0 + \cdots$

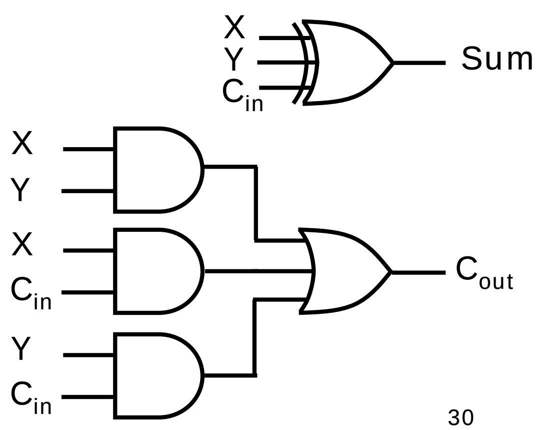
$C_2 = A_1'A_0'B_1B_0C_0 + \cdots$

Design of Binary Adders
By Composing Full Adders





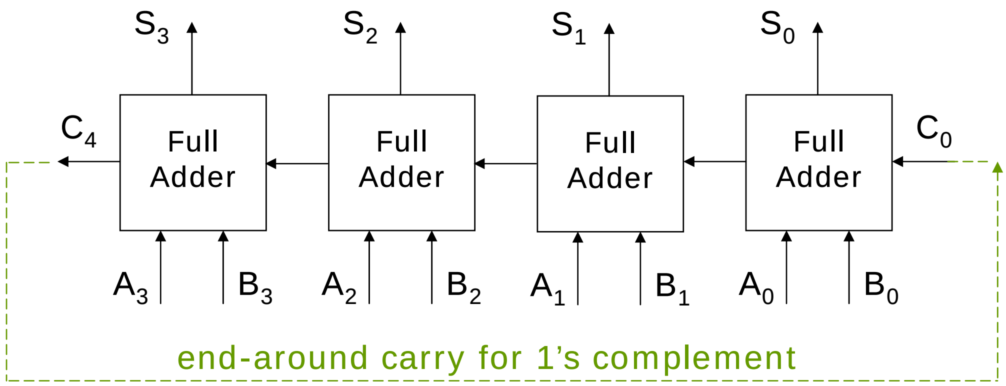
X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Design of Binary Adders

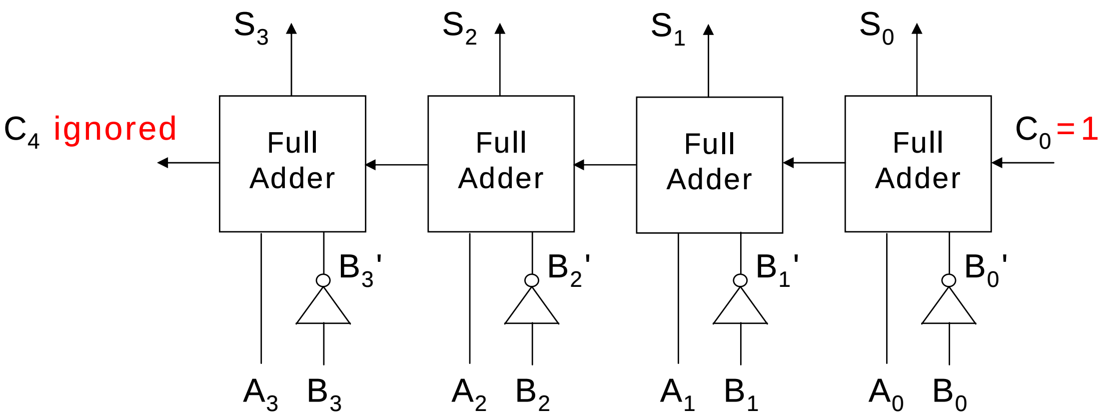
- Adder design for 4-bit signed binary numbers
  - 2's complement:
    - C_4 ignored
    - Because of C_0=0, the first full adder can be simplified to a **half adder**, with $S_0=A_0 \oplus B_0$ and $C_1= A_0B_0$
  - 1's complement:
    - End-around carry

Overflow condition: $V = A_3'B_3'S_3 + A_3B_3S_3'$



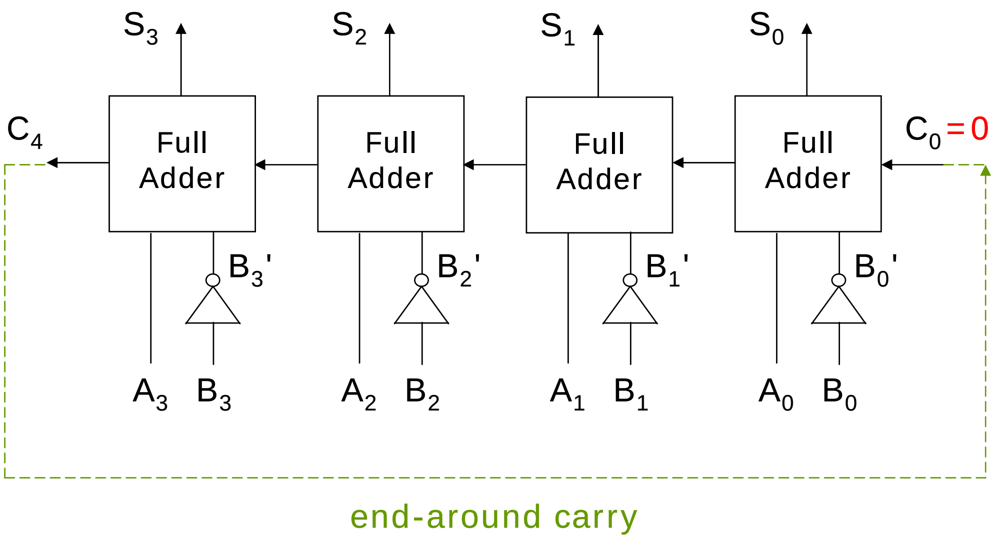
Design of Binary Subtracters

- Subtracter design using full adders
  - A–B = A+(–B) with –B in the 2's complement



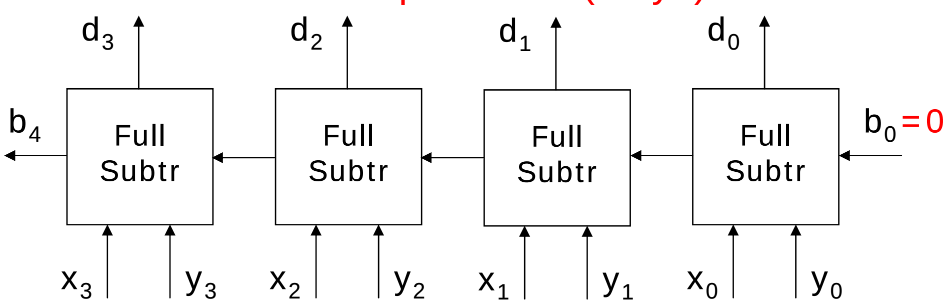
Design of Binary Subtracters

- Subtracter design using full adders
  - A–B = A+(–B) with –B in the 1's complement



Design of Binary Subtracters

- Subtracter design using full subtracters
  - Work also for 1's and 2's complements (why?)



x_i y_i b_i	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

	Column i before borrow	Column i after borrow
x_i	0	10
$-\ y_i$	–1	–1
$-\ b_i$	–1	–1
d_i		0 ($b_{i+1}=1$)