EECS 3201:
Digital Logic Design
Lecture 6

Ihab Amer, PhD, SMIEEE, P.Eng.

Synthesis by Boolean Algebra

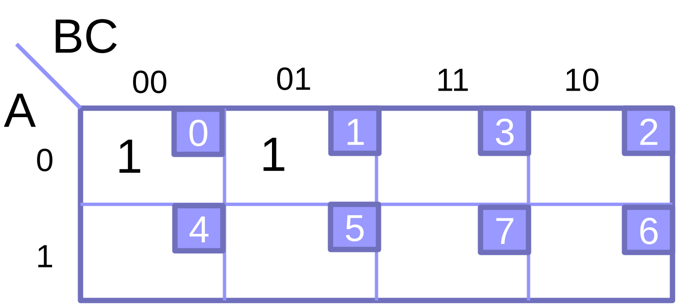
- We used a combination of Boolean theorems to implement more efficient circuits
  - not obvious how to apply these
  - often tedious

- Karnaugh Map a much more systematic method
  - a graphical approach
  - a judicious application of the combining property
    - $a \bullet b + a \bullet \overline{b} = a$
    - $(a + b) \bullet (a + \overline{b}) = a$
  - combine terms with complementary variables

Karnaugh Maps (K-Maps)

- Boolean expressions can be minimized by combining terms
- K-maps minimize equations graphically
- $PA + P\overline{A} = P$ (*combining property*)
- Translate truth table into grid with corresponding
- K-map arranged such that adjacent grids vary by only one literal

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

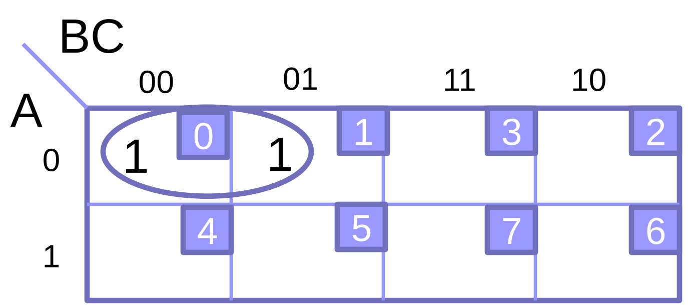


K-Map

- Circle 1's in adjacent squares

- In Boolean expression, include only literals whose true and complement form are ***not*** in the circle

A	*B*	*C*	*Y*
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



- $Y = \bar{A}\bar{B}$

K-Map Rules

- Every 1 must be circled at least once
- Each circle must span a power of 2 (i.e. 1, 2, 4) squares in each direction
- Each circle must be as large as possible
- A circle may wrap around the edges
- A “don't care” (X) is circled only if it helps minimize the equation

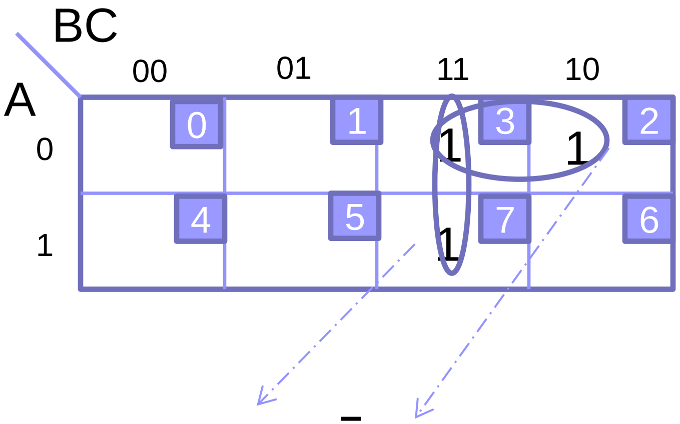
K-Map Minimization Technique

- It is kind of a greedy approach
- Start by circling the largest possible #of 1’s (according to the rules)
- Look at the remaining 1’s, and circle the largest possible #of remaining 1’s (your circle is allowed to cover 1’s that were circled before)
- Stop when every 1 is circled at least once

3-Input K-Map

Truth Table

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



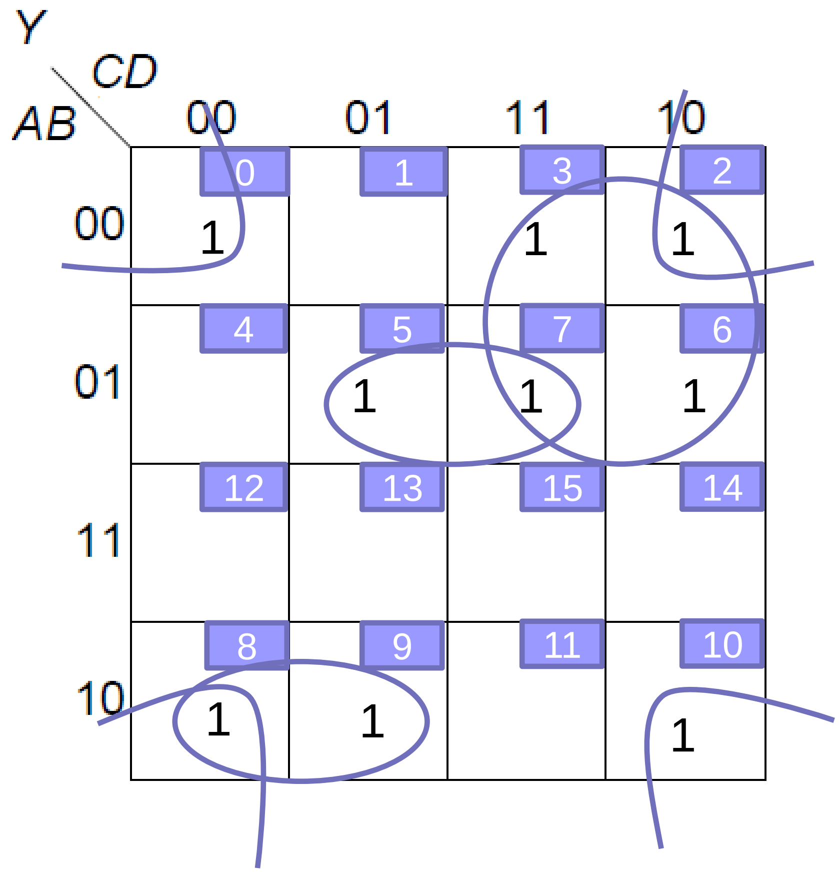
$Y = BC + \bar{A}B$

Example

$f(x_1,x_2,x_3) = \Sigma\, m(1,3,4,6,7)$

4-Input K-Map

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



$Y = \bar{A}C + \bar{B}\bar{D} + \bar{A}BD + A\bar{B}\bar{C}$

K-Map with Don't Cares

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

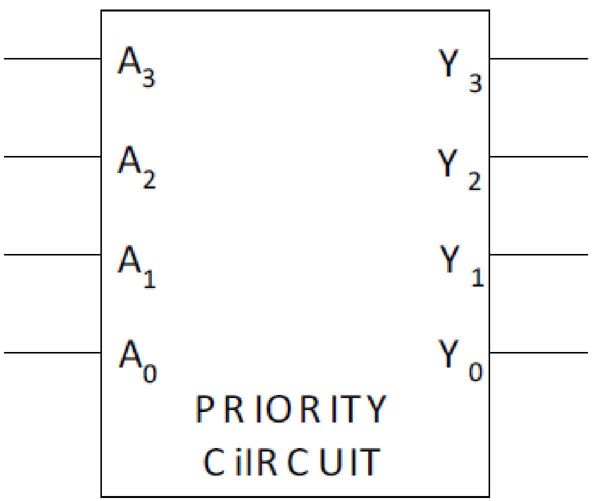
Y

AB \ CD	00	01	11	10
00				
01				
11				
10				

An Aside on Don't Cares

- **Example: Priority Circuit**

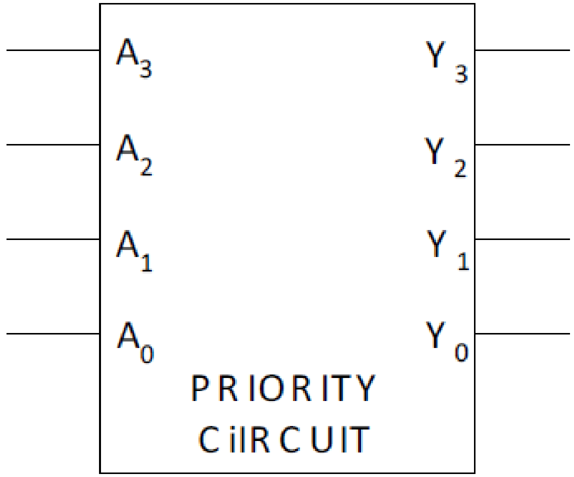
  Output asserted corresponding to most significant TRUE input



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

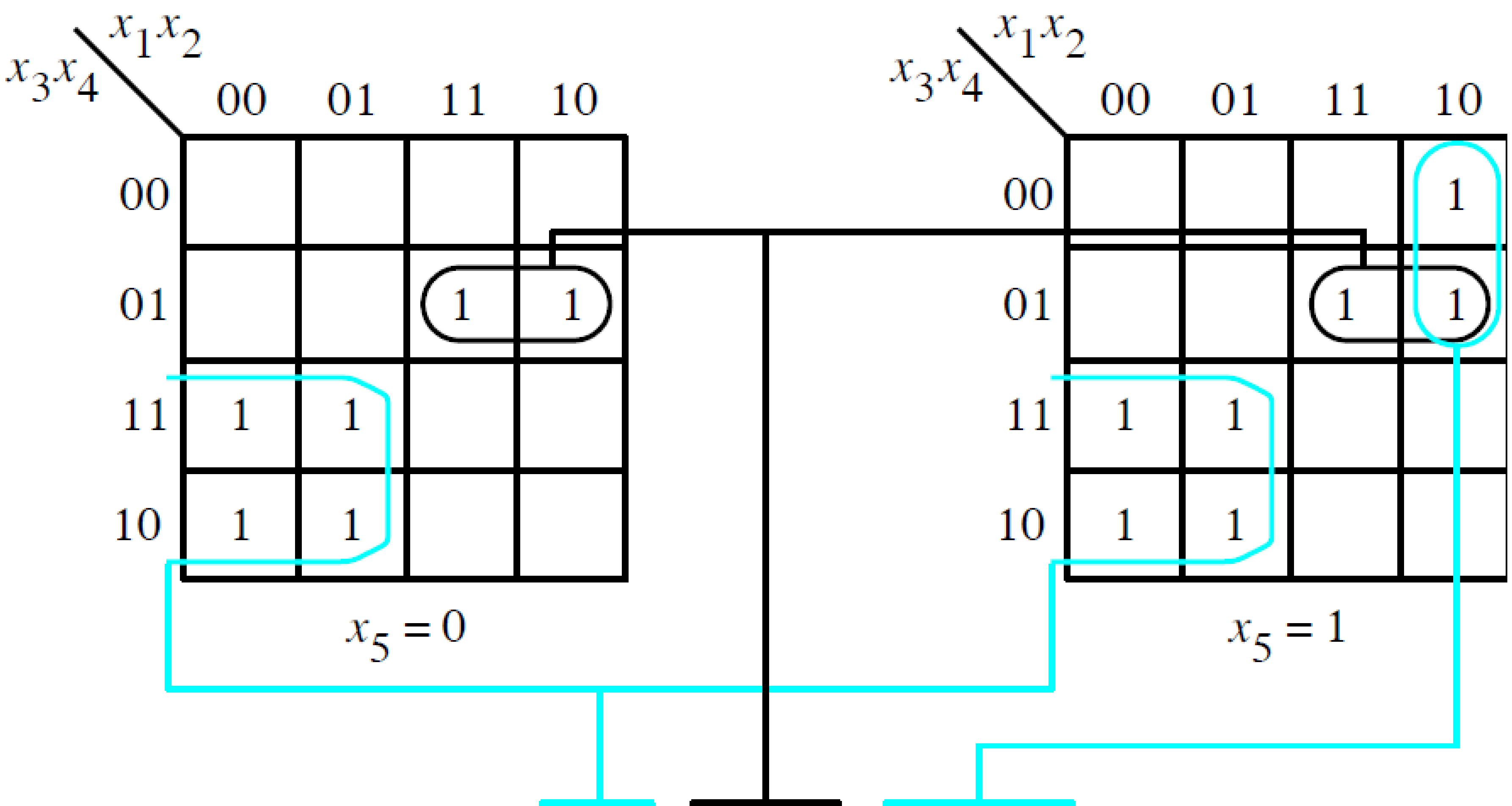
An Aside on Don't Cares

- **Example: Priority Circuit**
  Output asserted corresponding to most significant TRUE input



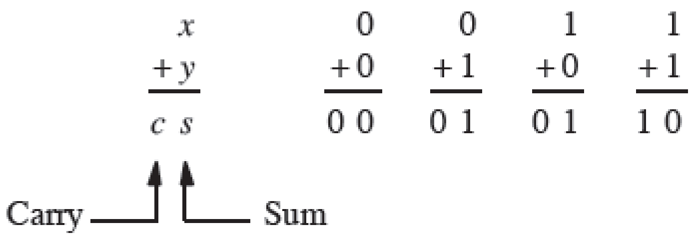
A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

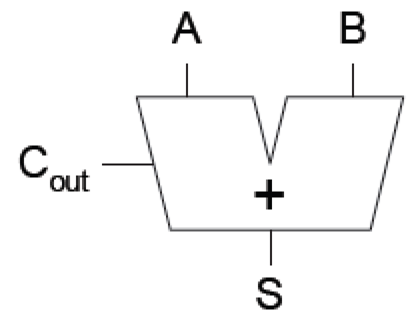
5-Variable Map

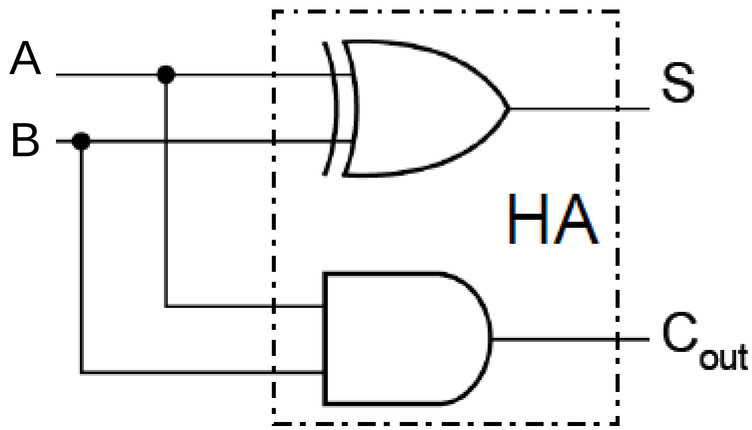


$$f_1 = \bar{x}_1x_3 + x_1\bar{x}_3x_4 + x_1\bar{x}_2\bar{x}_3x_5$$

1-bit Addition – The Half Adder







A	B	C_{out}	S
0	0		
0	1		
1	0		
1	1		

S =

C_{out} =

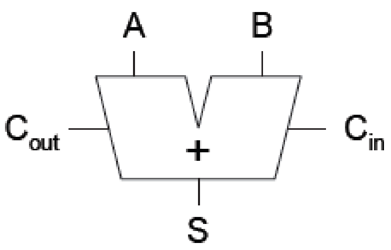
Multi-Bit Addition

Generated carries →	1 1 1 0	
$X = x_4x_3x_2x_1x_0$	0 1 1 1 1	$(15)_{10}$
$+\,Y = y_4y_3y_2y_1y_0$	+ 0 1 0 1 0	$+\,(10)_{10}$
$S = s_4s_3s_2s_1s_0$	1 1 0 0 1	$(25)_{10}$

…	c_{i+1}	c_i	…
…	…	x_i	…
…	…	y_i	…
…	…	s_i	…

Bit position *i*

1-bit Addition – The Full Adder



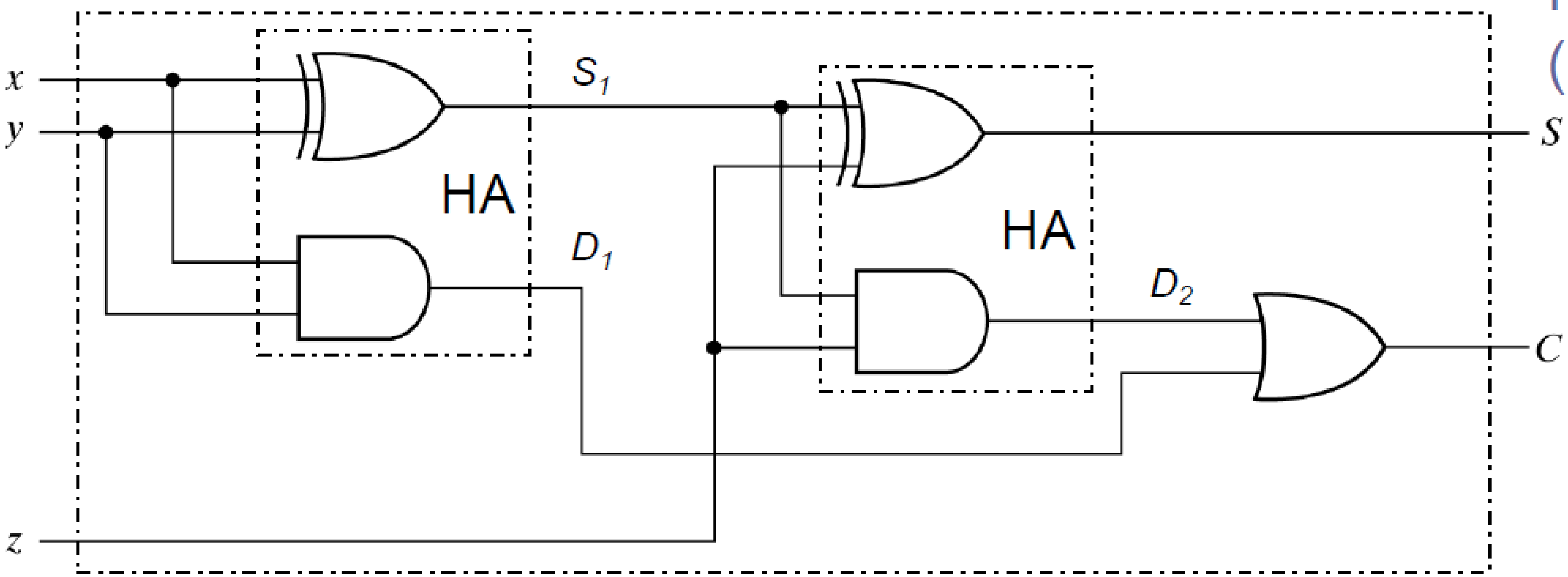
C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$S =$

$C_{out} =$

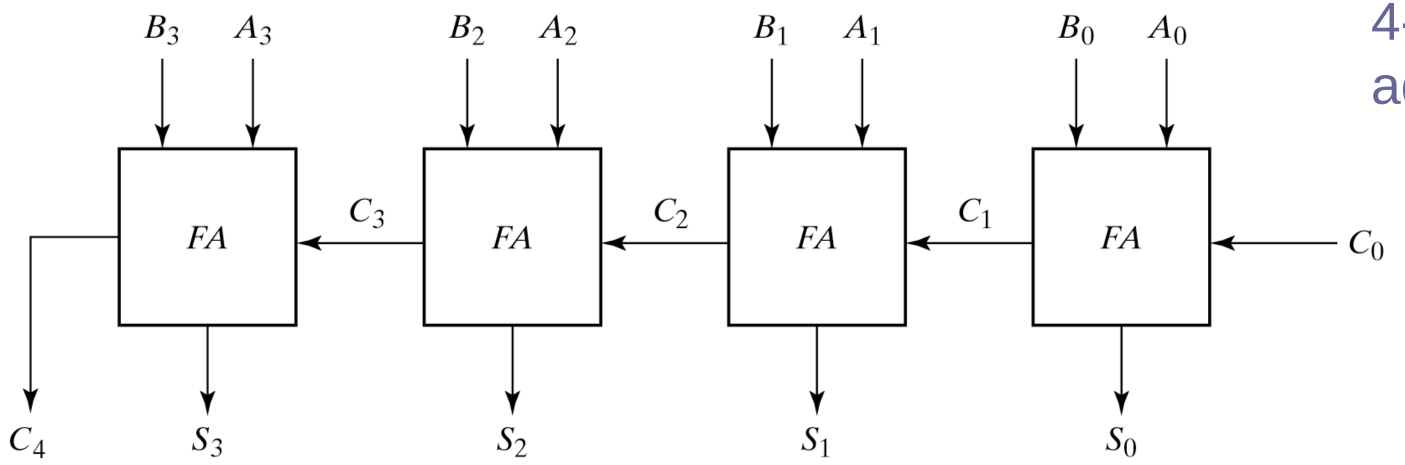
- A straightforward schematic

Full Adder with 2 half adders



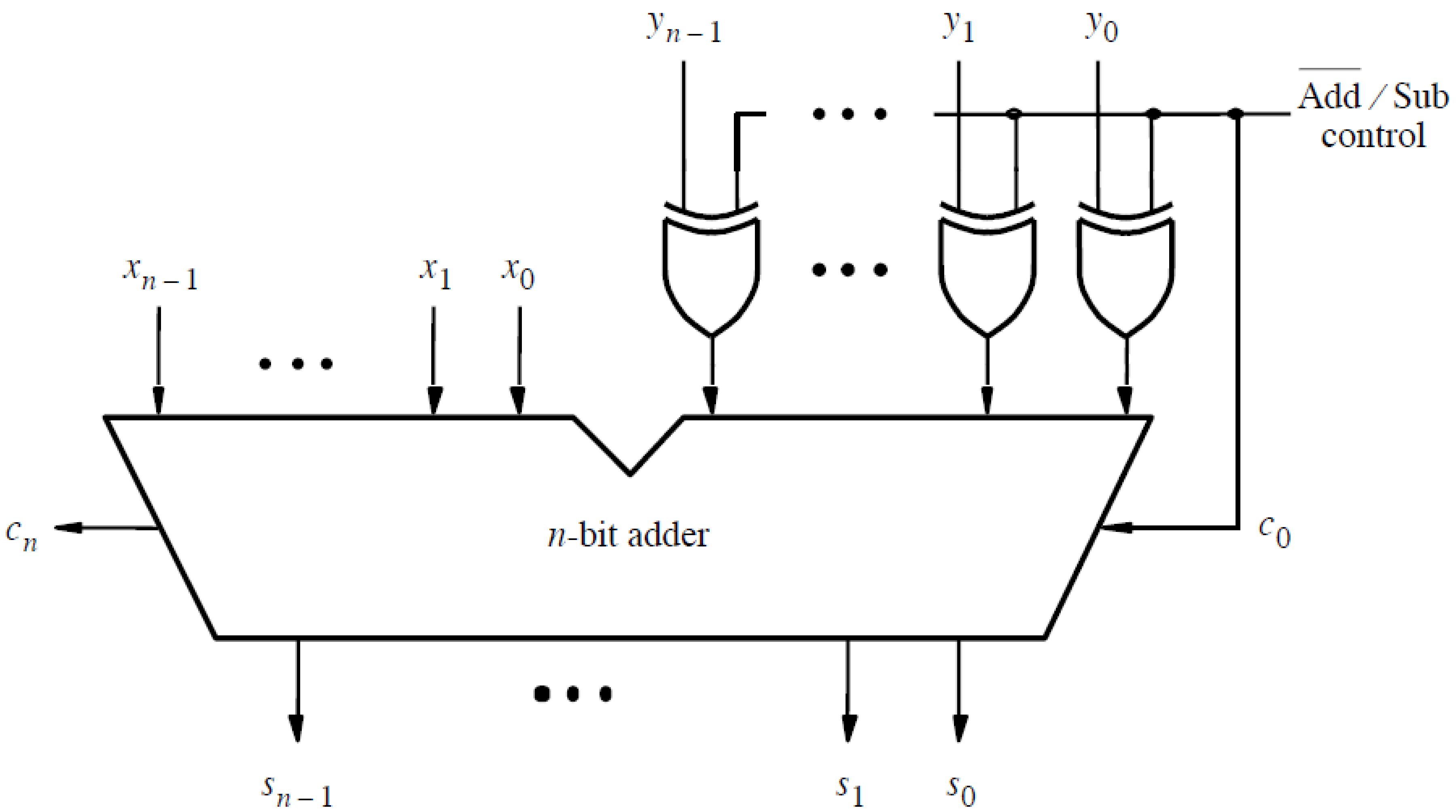
Full Adder (FA)

Carry-Ripple Adder



4-bit binary adder

Adder/Subtractor



References

- Lecture Notes of Dr. Sebastian Magierowski – Fall 2013