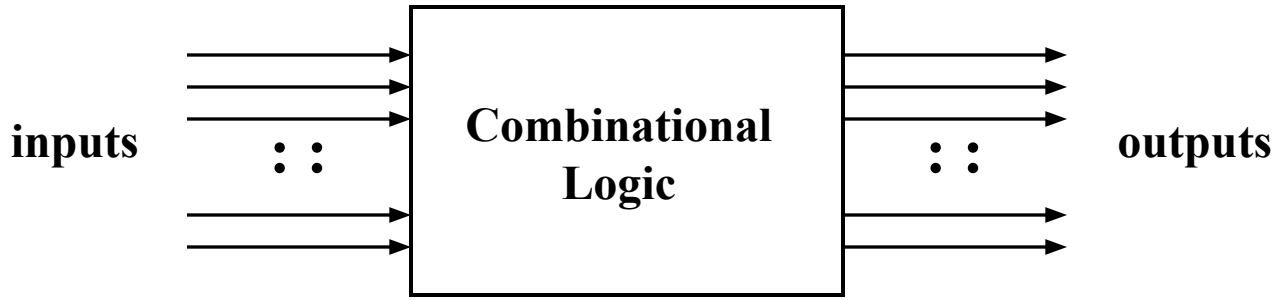
CSE 260 DIGITAL LOGIC DESIGN

Combinational Circuits- 1

Introduction

- Two classes of logic circuits:
  - combinational
  - sequential
- **Combinational Circuit**:



Combinational Circuit vs. Sequential Circuit

Combinational

- A combinational circuit consists of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs.
- A combinational circuit performs a specific information-processing operation fully specified logically by a set of Boolean functions.

Sequential

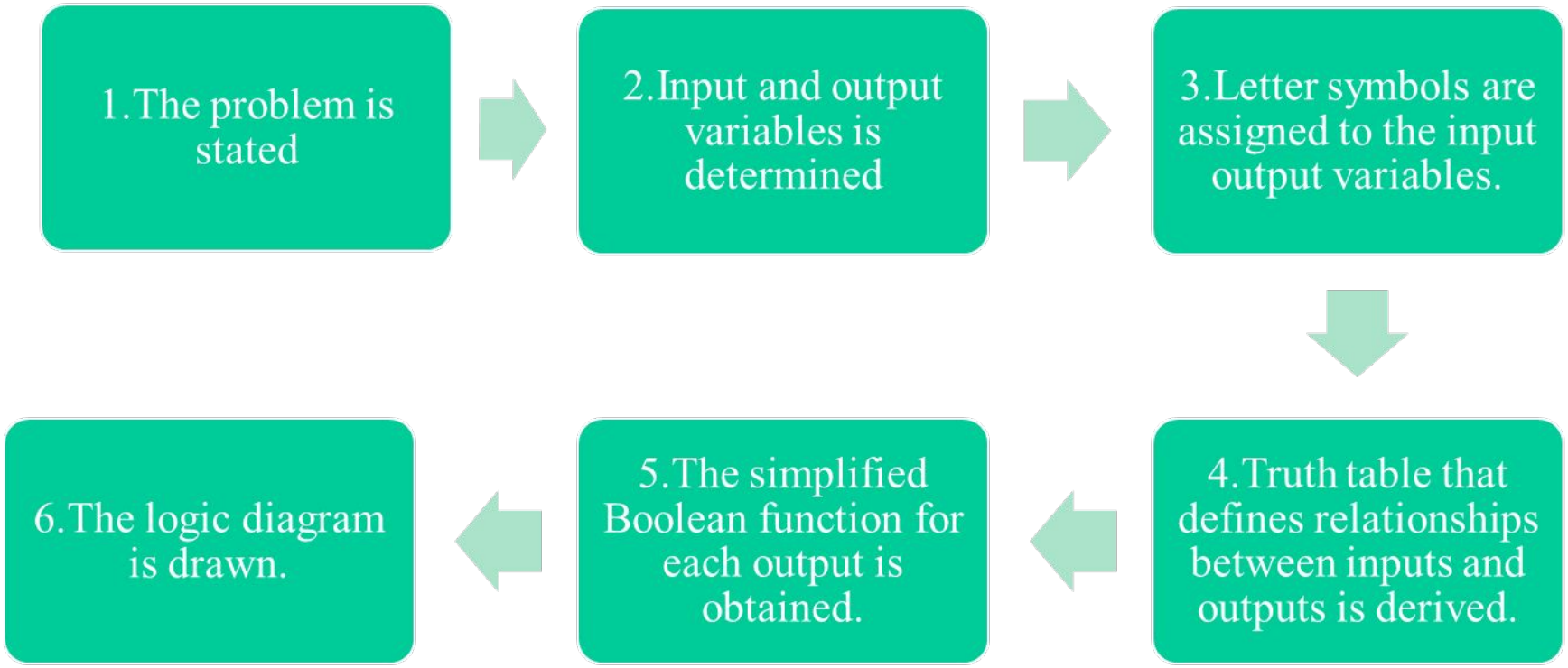
- Sequential circuits employ memory elements in addition to logic gates. Their outputs are a function of the inputs and the state of memory elements
- The state of memory elements, in turn, is a function of previous inputs. As a consequence, the outputs of a sequential circuit depend not only on present inputs, but also on past inputs.

- A combinational circuit consists of input variables, logic gates and output variables.
- A combinational circuit can be described by m Boolean functions, one for each output variable. Each output function is expressed in terms of the n input variables.
- There **is no feedback path** or presence of memory element

Design Procedure:

The design of combinational circuits starts from the verbal outline of the problem and ends in a logic circuit diagram.

The procedure involves the following steps:



Hint: Aren't these the steps used for solving car-garage alarm system?!!!!So we are familiar with combinational circuit design! ☺

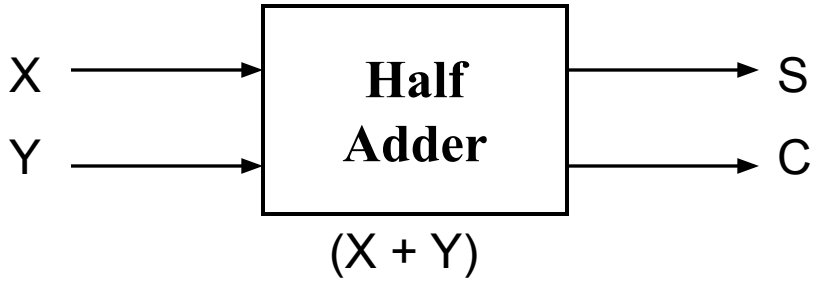
Design Procedure: Half Adder

- The combinational circuit that performs the addition of two bits is called a half-adder**.**
- **1)** State Problem

  Example: Build a Half Adder to add two bits

  2) Determine and label the inputs & outputs of circuit.

  Example: Two inputs and two outputs labeled, as follows:



3) Draw truth table.

Design Procedure: Half Adder

4) Obtain simplified Boolean function.

Example: C = XY

S = X'Y + XY' = X⊕Y

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

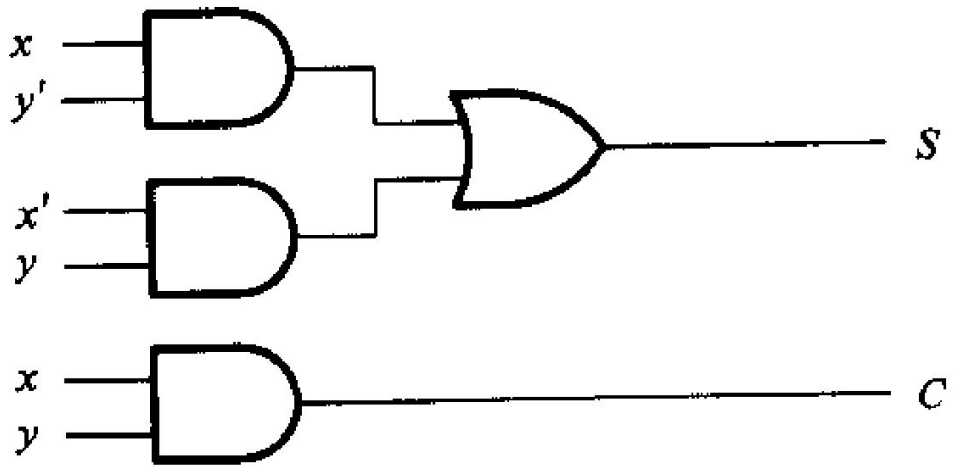
5) Draw logic diagram.

Half Adder

Ways to draw half-adder

- Method 1:
  S=xy'+x'y

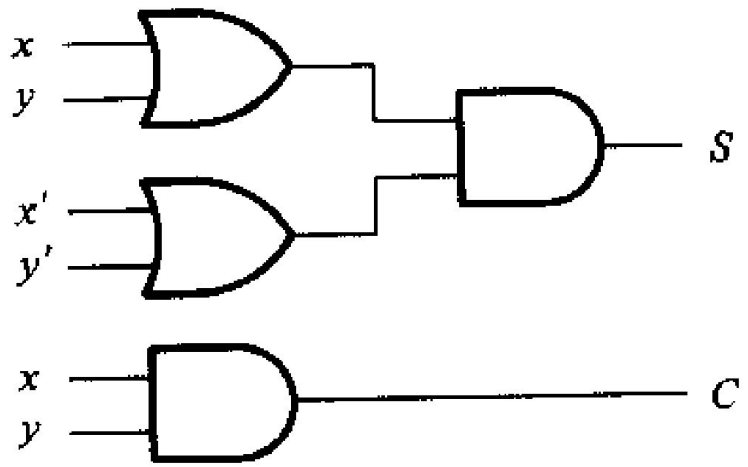
  C=xy



(a) $S = xy' + x'y$
$C = xy$

Ways to draw half-adder

- Method 2:
  S=xy'+x'y=(x+y)(x'+y') (i.e. POS format)
  C=xy

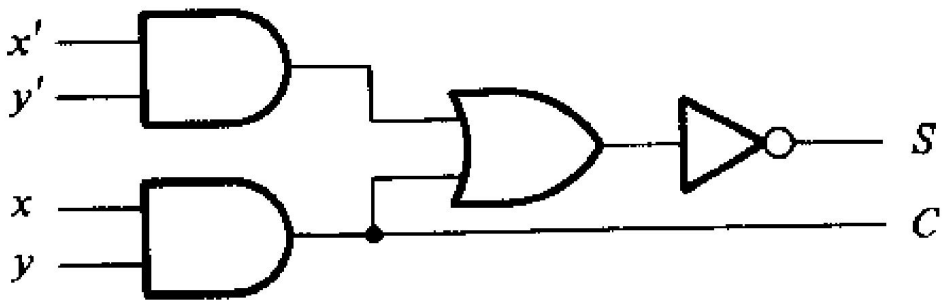


(b) $S = (x + y)(x' + y')$
$C = xy$

Ways to draw half-adder

- Method 3:
  S=xy’+x’y
  S=(S’)’=(xy+x’y’)’
  =(C+x’y’)’

  C=xy



(c) $S = (C + x'y')'$
$C = xy$

Ways to draw half-adder

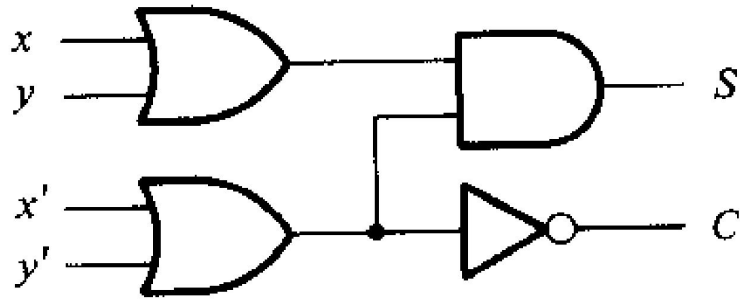
- Method 4:

  S=xy’+x’y

  =(x+y)(x’+y’)

  C=(C’)’=((xy)’)’

  =(x’+y’)’



(d) $S = (x + y)(x' + y')$
$C = (x' + y')'$

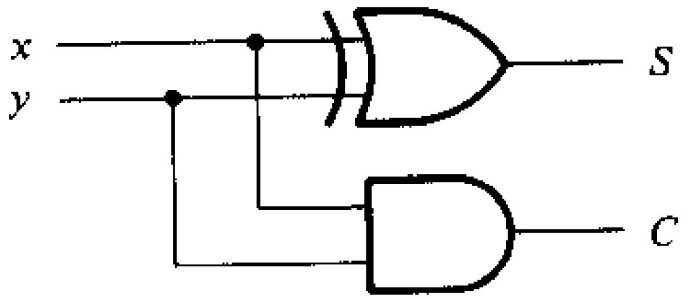
Ways to draw half-adder

- Most commonly used

  S=xy'+x'y

  $=x \oplus y$

  C=xy



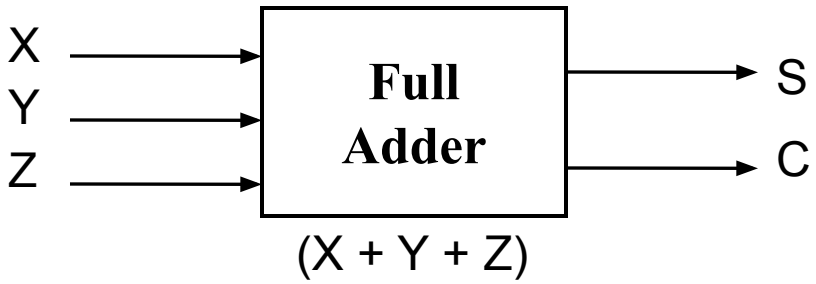
(e) $S = x \oplus y$
$C = xy$

Design Procedure : Full Adder

- Half-adder adds up only two bits.
- To add two binary numbers, we need to add 3 bits (including the *carry*).
- Example:

	1	1	1		carry
	0	0	1	1	X
+	0	1	1	1	Y
	1	0	1	0	S

- Need Full Adder (so called as it can be made from two half-adders).



Design Procedure : Full Adder

- Truth table:

x	y	z	C	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

Note:

Z - carry in (to the current position)

C - carry out (to the next position)

- From truth table

C = X'YZ+XY'Z+XYZ'+XYZ

S = X'Y'Z + XYZ + X'YZ'+XY'Z'

Design Procedure : Full Adder

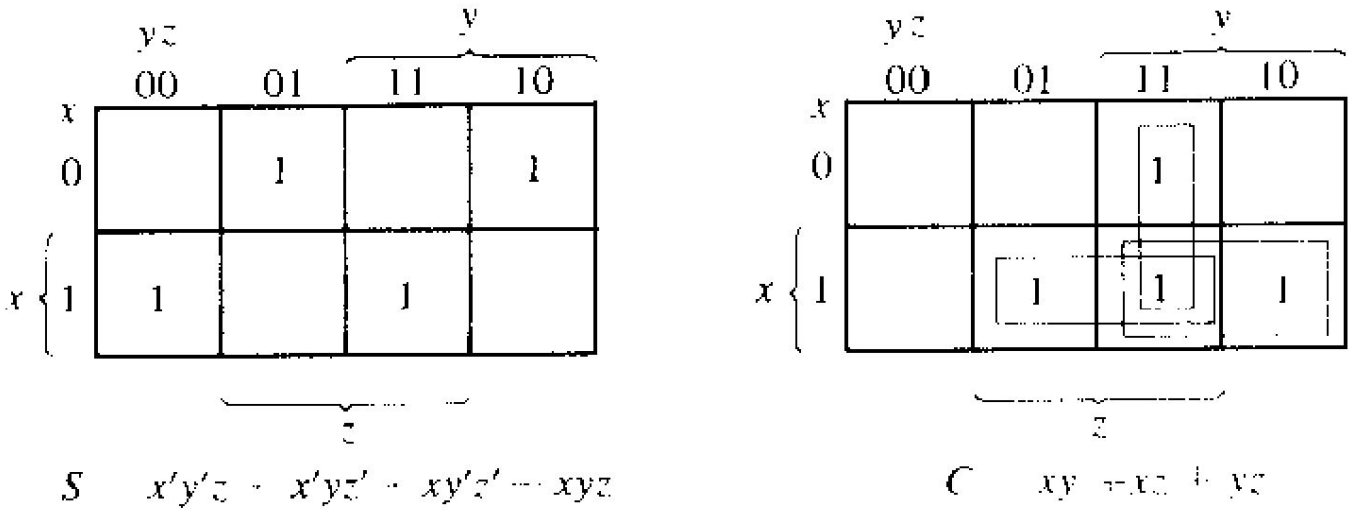


FIGURE 4-3
Maps for a full-adder

- Alternative formulae using algebraic manipulation, Using K-map, simplified SOP form:

  C = XY + XZ + YZ

  S = X'Y'Z + XYZ + X'YZ'+XY'Z'

Gate-level (SSI) Design: Full Adder

$$C = XY + XZ + YZ$$
$$= XY + (X + Y)Z$$
$$= XY + ((X \oplus Y) + XY)Z$$
$$= XY + (X \oplus Y)Z + XYZ$$
$$= XY + (X \oplus Y)Z$$

$$S = X'Y'Z + X'YZ' + XY'Z' + XYZ$$
$$= X'(Y'Z + YZ') + X(Y'Z' + YZ)$$
$$= X'(Y \oplus Z) + X(Y \oplus Z)'$$
$$= X \oplus (Y \oplus Z) \quad \text{or} \quad (X \oplus Y) \oplus Z$$

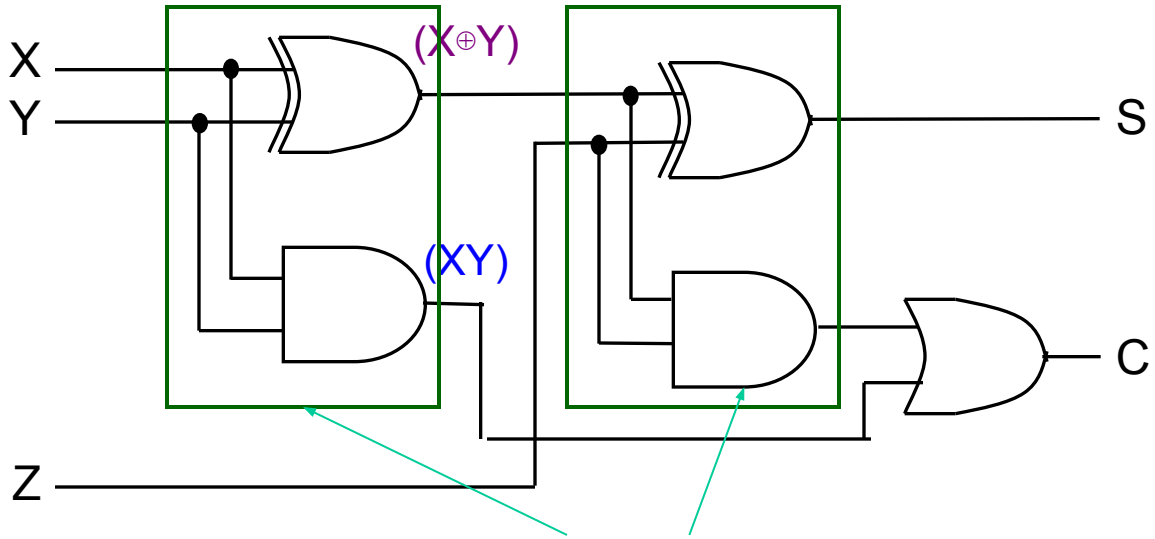
Note:
How to turn (X+Y) to ((X⊕Y) + XY)
If F=(X+Y)
=(X+Y)(Y+Y')
=XY+XY'+YY+YY'
=XY+XY'+Y
=XY'+XY+Y
=XY'+Y(X+1)
=XY'+Y
=XY'+Y(X'+X)
=XY'+X'Y+XY
=(X⊕Y) + XY)

Gate-level (SSI) Design: Full Adder

- Circuit for above formulae:

$$C = XY + (X \oplus Y)Z$$

$$S = (X \oplus Y) \oplus Z$$



Full Adder made from two Half-Adders (+ OR gate).

Design Methods

- Different combinational circuit design methods:
  - Gate-level method (with logic gates)
  - Block-level design method
- Design methods make use of logic gates and useful functional blocks.
  - These are available as Integrated Circuit (IC) chips.

Why is it better to use IC than classical gate method?

- IC consists of gates all packed up together internally, which keeps the gates safe inside as well as makes it economical by reducing external interconnection wires
- As number of input increase, making the truth table and k-map gets cumbersome

Design Methods

- Type of IC chips (based on packing density) :
  - Small-scale integration (SSI): up to 12 gates
  - Medium-scale integration (MSI): 12-99 gates
  - Large-scale integration (LSI): 100-9999 gates
  - Very large-scale integration (VLSI): 10,000-99,999 gates
  - Ultra large-scale integration (ULSI): > 100,000 gates

- ***Main objectives of circuit design:***
  - (i) reduce cost
    - reduce number of gates (for SSI circuits)
    - reduce IC packages (for complex circuits)
  - (ii) increase speed
  - (iii) design simplicity (reuse blocks where possible)

Block-Level Design Method

- More complex circuits can be built using block-level method.
- In general, block-level design method (as opposed to gate-level design) relies on algorithms or formulae of the circuit, which are obtained by decomposing the main problem to sub-problems recursively (until small enough to be directly solved by blocks of circuits).

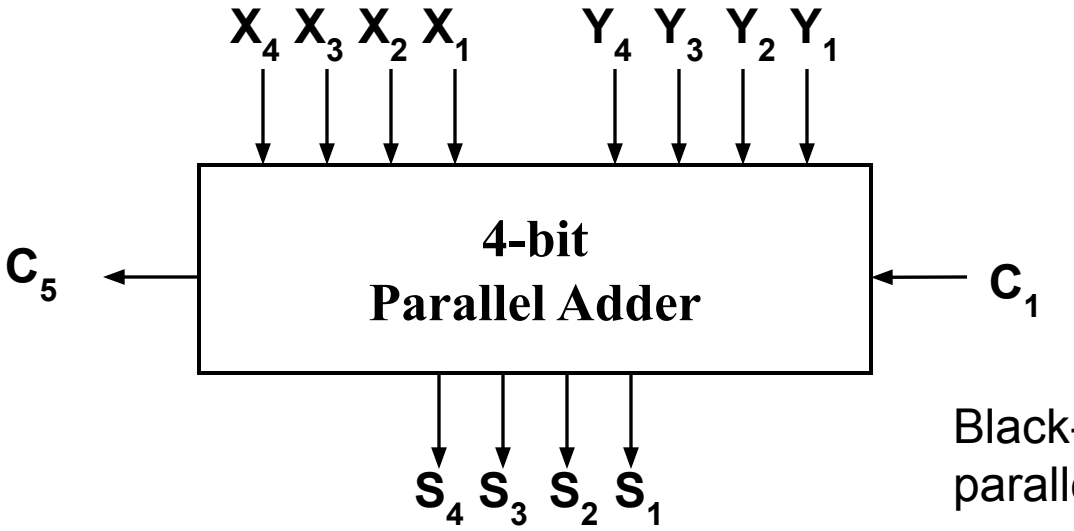
What if we want to add more than 3 bit?

N-bit Full adder Can be

- Series adder: consist of 1 full adder and a storage for saving Carry out (which becomes Carry in for next bits addition)
- Parallel adder: Consist of n adder and all n-bits of both number are given as input simultaneously. Carry input, in the least significant position, must be 0.

4-bit Parallel Adder

- Consider a circuit to add two 4-bit numbers together and a carry-in, to produce a 5-bit result:



Black-box view of 4-bit parallel adder

- 5-bit result is sufficient because the largest result is:

$$(1111)_2+(1111)_2+(1)_2 = (11111)_2$$

4-bit Parallel Adder

- SSI design technique should not be used.
- Truth table for 9 inputs very big, i.e. 2^9=512 entries:

$X_4X_3X_2X_1$	$Y_4Y_3Y_2Y_1$	C_1	C_5	$S_4S_3S_2S_1$
0 0 0 0	0 0 0 0	0	0	0 0 0 0
0 0 0 0	0 0 0 0	1	0	0 0 0 1
0 0 0 0	0 0 0 1	0	0	0 0 0 1
...	...	...	...	...
0 1 0 1	1 1 0 1	1	1	0 0 1 1
...	...	...	...	...
1 1 1 1	1 1 1 1	1	1	1 1 1 1

- Simplification very complicated as no. of input is 9 so 512(=2^9) possible combination will be there in truth table!

4-bit Parallel Adder

- Alternative design possible.
- Addition formulae for each pair of bits (with carry in).

output

$$C_{i+1}S_i = X_i + Y_i + C_i$$

input

has the same function as a full adder.

$$C_{i+1} = X_i Y_i + (X_i \oplus Y_i) C_i$$

$$S_i = X_i \oplus Y_i \oplus C_i$$

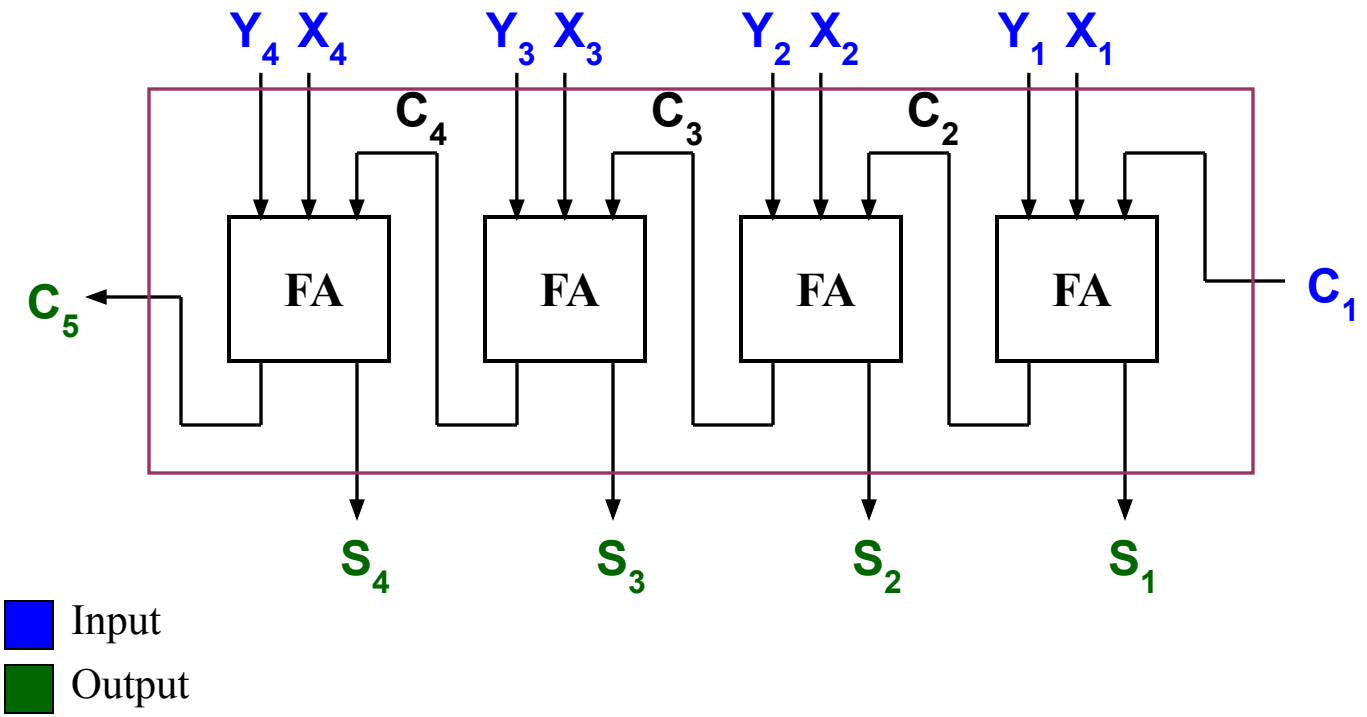
NB: Similar to FA equation

$C = XY + (X \oplus Y)Z$

$S = (X \oplus Y) \oplus Z$

4-bit Parallel Adder

- Cascading 4 full adders via their carries, we get:



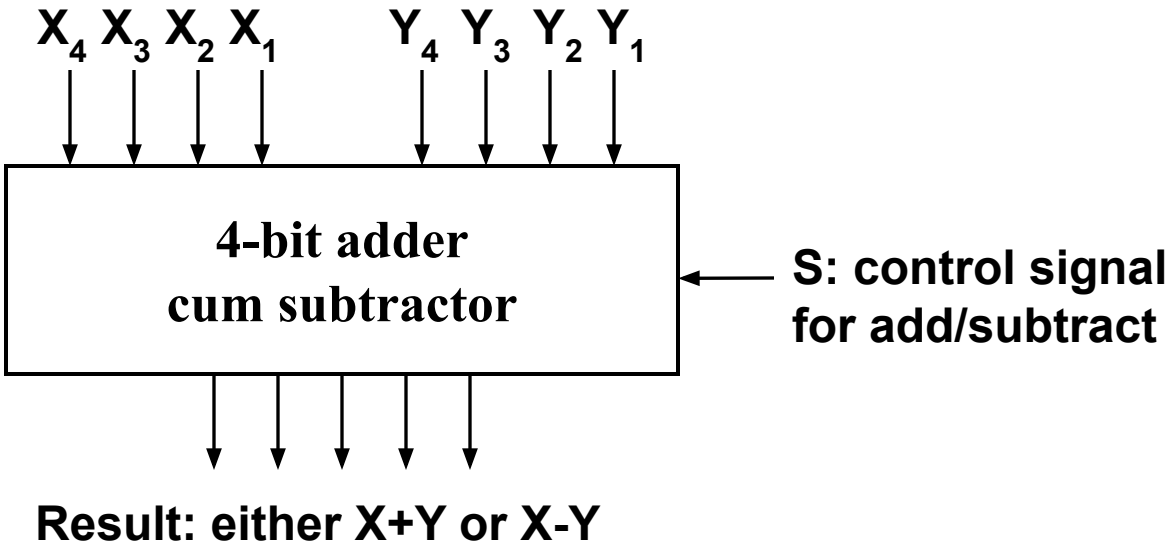
n-bit full adder requires *n* full- adders

Examples

- Simple examples using 4-bit parallel adder as building blocks:
  - 16-Bit Parallel Adder
  - Adder cum Subtractor
  - BCD to excess 3 code converter

4-bit Parallel Adder cum Subtractor

- Subtraction can be performed through addition using 2s-complement numbers.
- Hence, we can design a circuit which can perform both addition and subtraction, using a parallel adder.



4-bit Parallel Adder cum Subtractor

- The control signal **S=0** means add
  S=1 means subtract

- Recall that:

$$X-Y = X + (-Y)$$
$$= X + (2\text{'s complement of } Y)$$
$$= X + (1\text{'s complement of } Y) + 1$$
$$X+Y = X + (Y)$$

4-bit Parallel Adder cum Subtractor

- Design requires:

  (i) XOR gates:

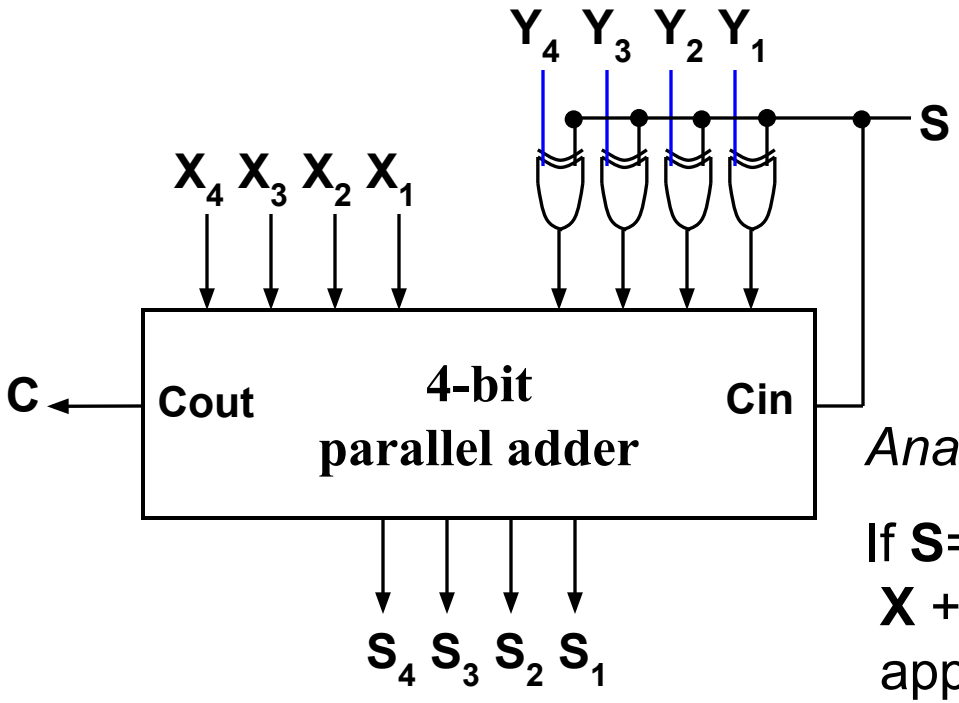
such that: output = Y when S=0
= Y' when S=1

(ii) S connected to carry-in.

S	Y	Output
0	0	0
0	1	1
1	0	1
1	1	0

4-bit Parallel Adder cum Subtractor

- Adder cum subtractor circuit:



A 4-bit adder cum subtractor

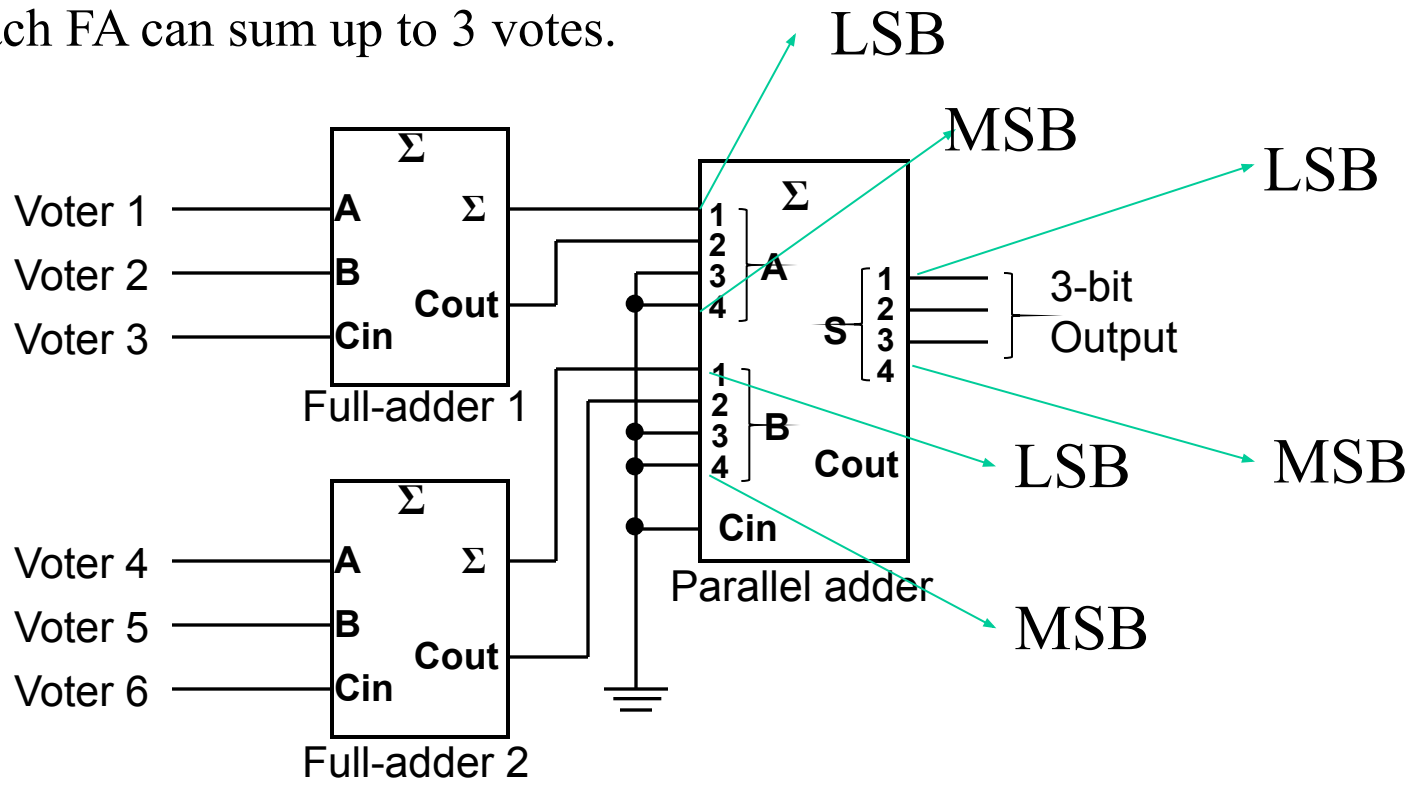
Analysis:

If **S**=1, then
X + (1's complement of **Y**) +1
appears as the result.

If **S**=0, then **X**+**Y** appears as the result.

Arithmetic Circuits: Cascading Adders

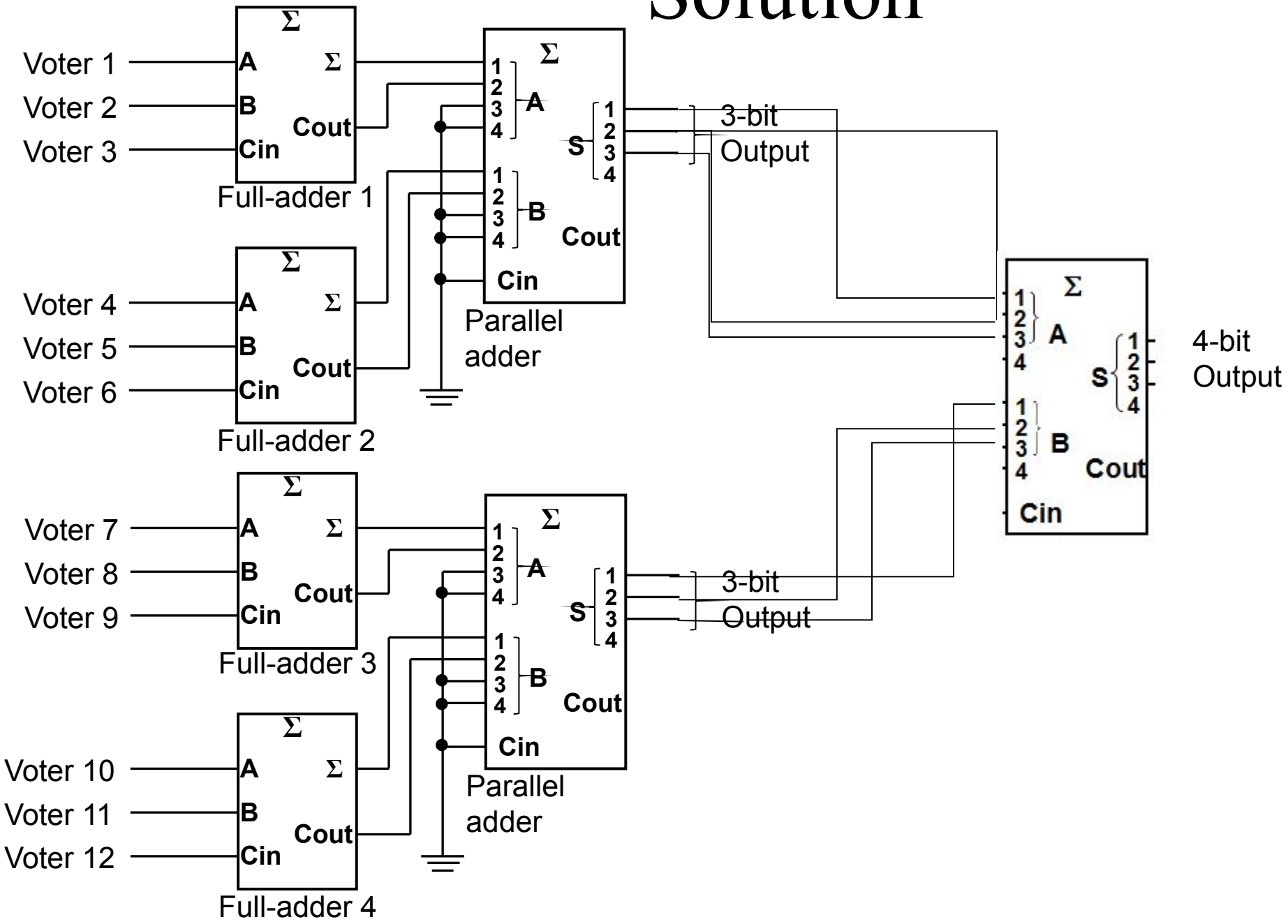
- Application: 6-person voting system.
  - Use FAs and a 4-bit binary parallel adder.
  - Each FA can sum up to 3 votes.



Try it yourself

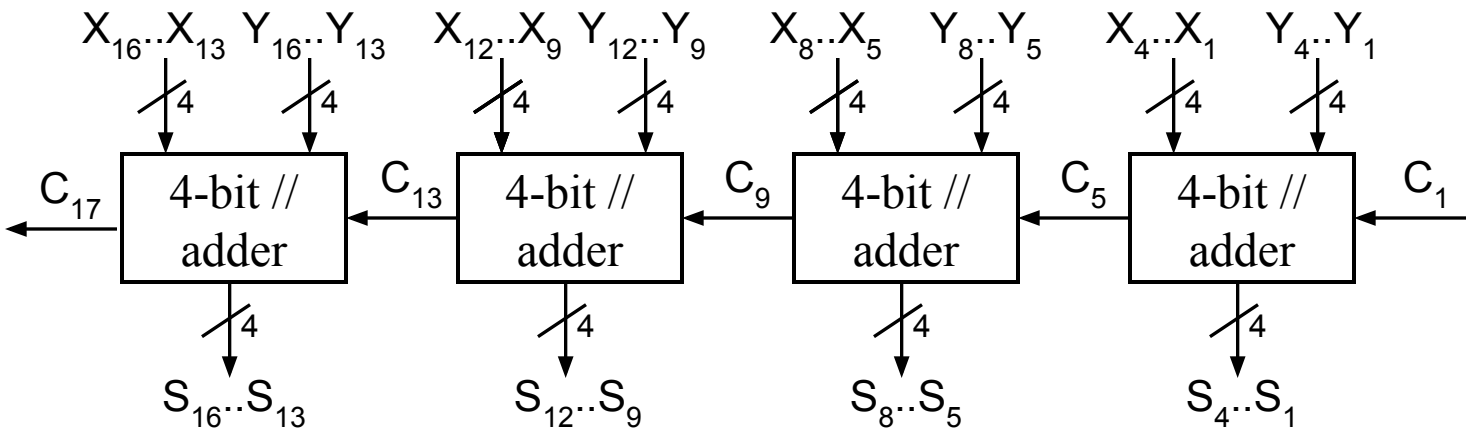
- Design a 12 person voting system

Solution


Voter 1
Voter 2
Voter 3
Full-adder 1
Voter 4
Voter 5
Voter 6
Full-adder 2
Parallel adder
3-bit Output
Voter 7
Voter 8
Voter 9
Full-adder 3
Voter 10
Voter 11
Voter 12
Full-adder 4
Parallel adder
3-bit Output
4-bit Output
A
B
Cin
Cout
S
Σ

16-bit Parallel Adder

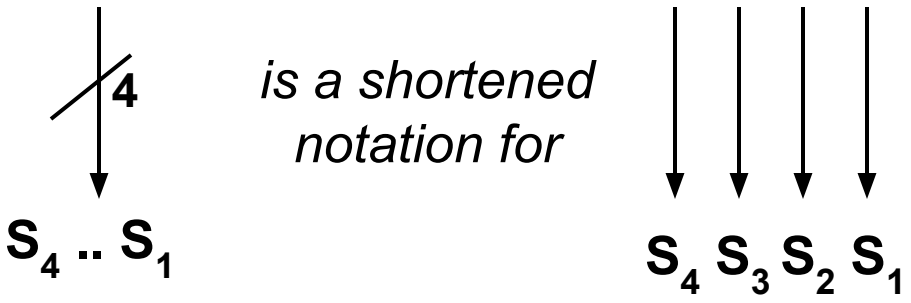
- Larger parallel adders can be built from smaller ones.
- Example: a 16-bit parallel adder can be constructed from four 4-bit parallel adders:



A 16-bit parallel adder

16-bit Parallel Adder

- Shortened notation for multiple lines.

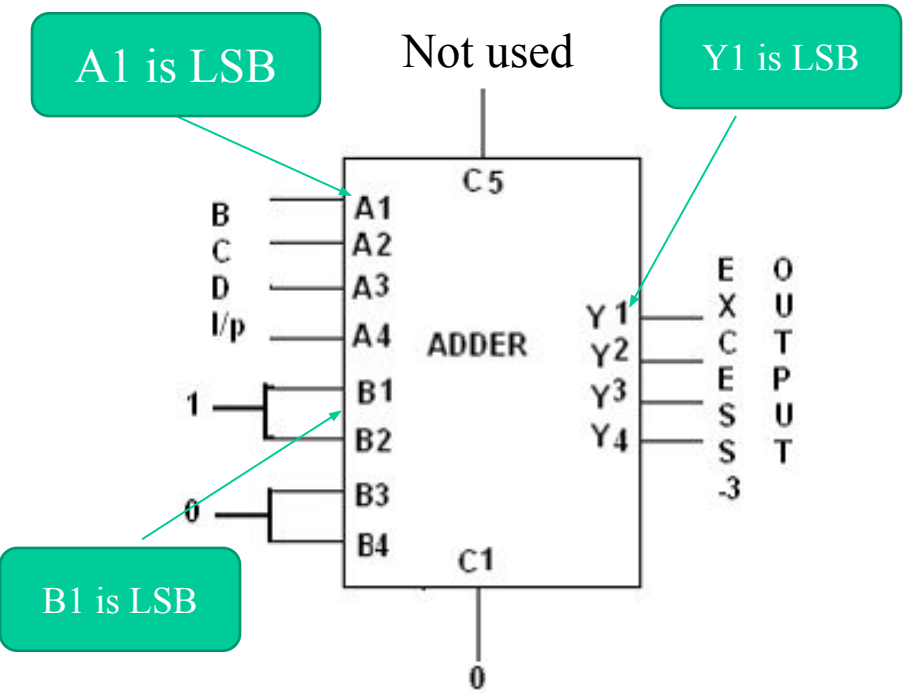


16-bit parallel adder ripples carry from one 4-bit block to the next.
Such ripple-carry circuits are “slow” because of long delays needed to propagate the carries.

Try it yourself

- BCD to excess 3 code converter using **4-bit Parallel adder**

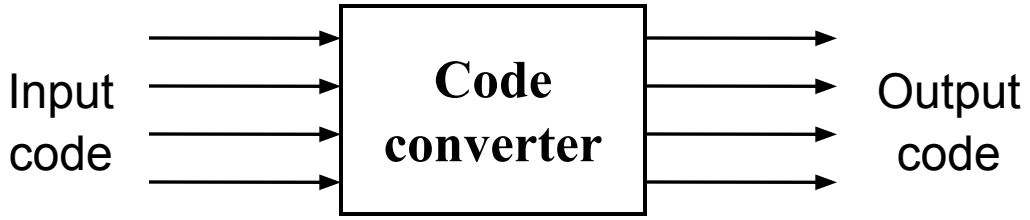
BCD to excess 3 code converter



Inspection of truth table of ‘excess 3 from bcd’ shows that we can get excess-3 code from bcd by adding ‘0011’ to each BCD number. This addition can be implemented by means of 4-bti full adder MSI circuit.

Code Converters

- Code converters – take an input code, translate to its equivalent output code.



- Example: BCD to Excess-3 Code Converter.

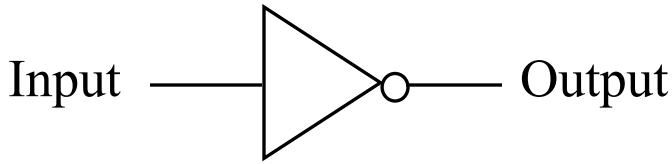
  Input: BCD digit

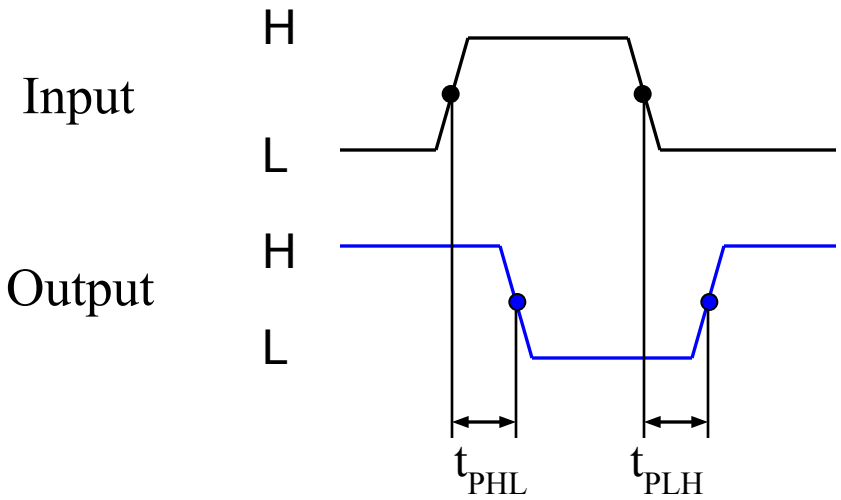
  Output: Excess-3 digit

Propagation Delay

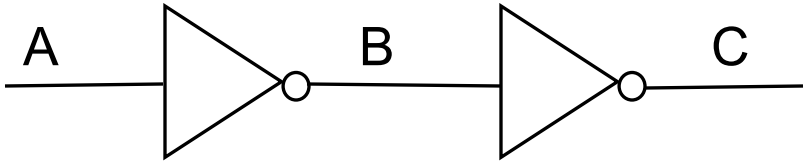
- Every logic gate experiences some delay (though very small) in propagating signals forward.
- This delay is called Gate (Propagation) Delay.
- Formally, it is the average transition time taken for the output signal of the gate to change in response to changes in the input signals.
- Three different propagation delay times associated with a logic gate:
  - t_{PHL}: output changing from the High level to Low level
  - t_{PLH}: output changing from the Low level to High level
  - $t_{PD}=(t_{PLH} + t_{PHL})/2$ (average propagation delay)

Propagation Delay

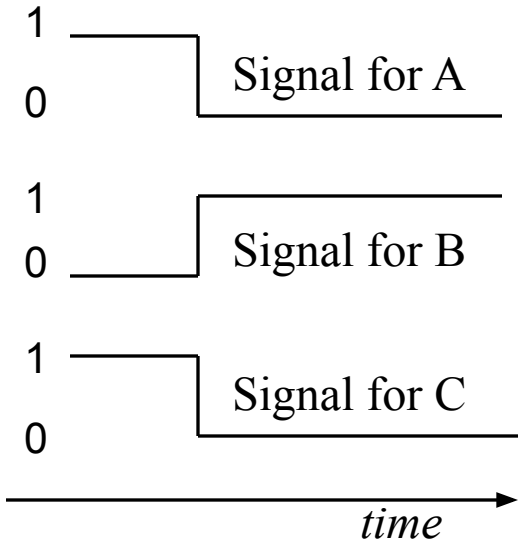




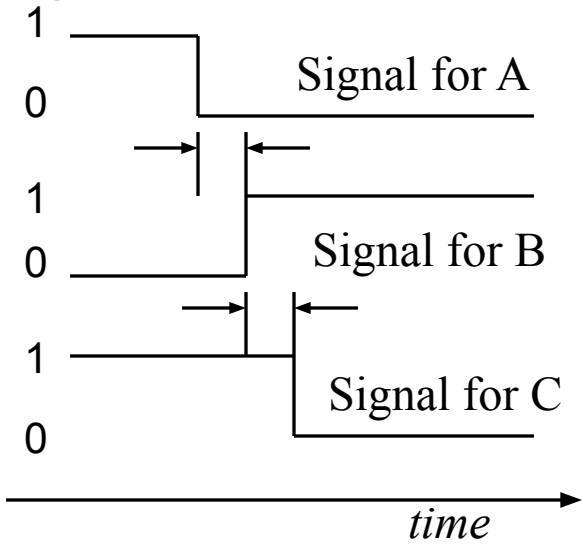
Propagation Delay



- Ideally, no delay:

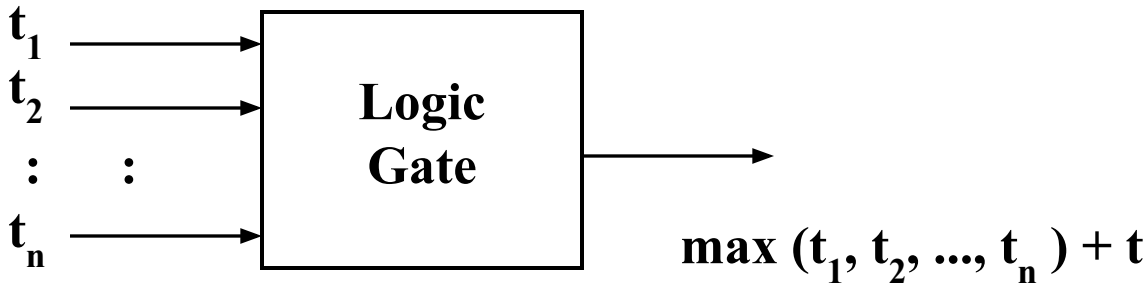


- In reality, output signals normally lag behind input signals:



Calculation of Circuit Delays

- In general, given a logic gate with delay, t.



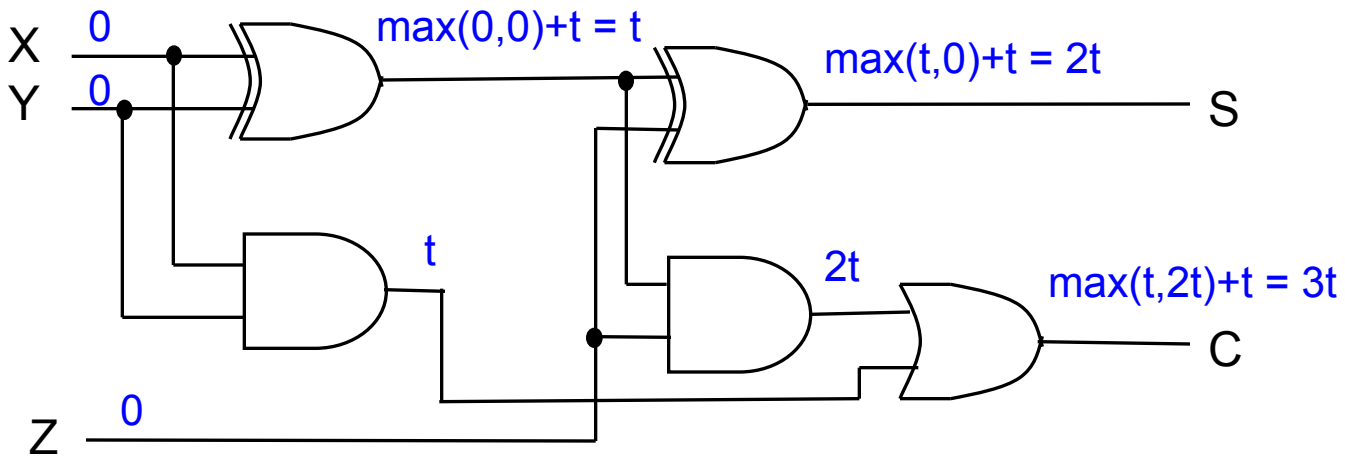
If inputs are stable at times t_1,t_2,..,t_n, respectively; then the earliest time in which the output will be stable is:

$$\max(t_1, t_2, .., t_n) + t$$

- To calculate the delays of all outputs of a combinational circuit, repeat above rule for all gates.

Calculation of Circuit Delays

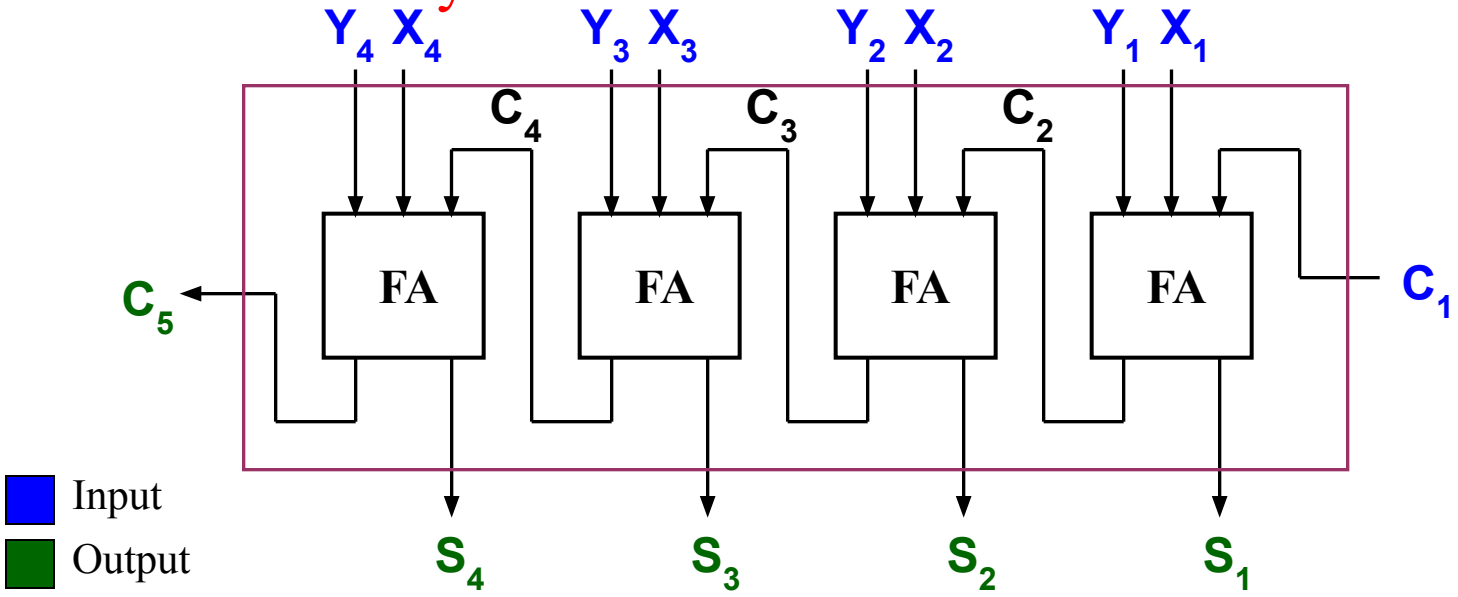
- As a simple example, consider the full adder circuit where all inputs are available at time 0. (Assume each gate has delay t.)



where outputs **S** and **C**, experience delays of 2t and 3t, respectively.

4-bit Parallel Adder

- Propagation time= no. of gates* propagation delay of each gate.
- Longest propagation delay is equal to carry to propagate through all 4 full adder. Only after the arrival of carry, S4 and C5 reaches final steady state.



n-bit full adder requires *n* full- adders

HomeWork for next class

- Obtain a NAND logic diagram of a single full-adder from the Boolean functions:
  * C= xy+xz+yz
  *S=C’(x+y+z)+xyz

A Must Self study

- ‘Chapter 4.9 : Exclusive OR and Equivalence Function’ (page 145 to page 149) of Morris Mano