Topic 5

Combination Circuit

What is Combinational Circuit?

- Combinational Circuit
  - A digital circuit whose output depends only upon the ***present combination*** of its inputs
  - Output changes only when inputs change
  - Output changes once inputs change
- As oppose to Sequential Circuit which is
  - A digital circuit whose output depends not only upon the present input values, but also the history of input and output values
  - Have feedbacks from outputs
  - More complicated and difficult to analyze
- Typical modern digital circuit involves both combination and sequential circuits

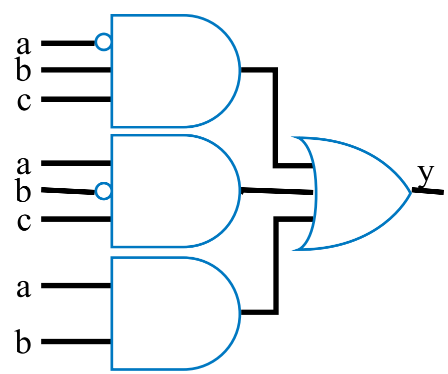
How to Design Combinational Circuits?

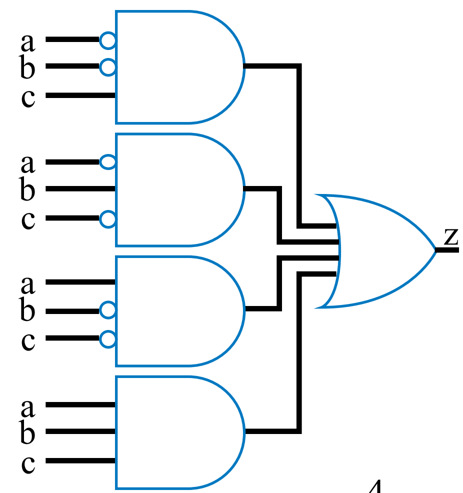
	Step	Description
Step 1	**Capture** the function	Create a truth table or equations, ***whichever is most natural for the given problem***, to describe the desired behavior of the combinational logic.
Step 2	**Convert** to equations	This step is only necessary if you captured the function using a truth table instead of equations. Create an equation for each output by ORing all the minterms for that output. Simplify the equations if desired.
Step 3	**Implement** as a logic circuit	For each output, create a circuit corresponding to the output's equation. (Sharing gates among multiple outputs is OK optionally.)

Example: Count Number of 1s

- Problem: Output in binary the number of 1s on three inputs
- E.g.: 010→01   101→10   000→00
  - **Step 1: Capture** the function
    - Truth table or equation?
      - Truth table is straightforward
  - **Step 2: Convert** to equation
    - y = a'bc + ab'c + abc' + abc
    - z = a'b'c + a'bc' + ab'c' + abc
  - **Step 3: Implement** as a gate-based circuit

Inputs			(# of 1s)	Outputs	
a	b	c		y	z
0	0	0	(0)	0	0
0	0	1	(1)	0	1
0	1	0	(1)	0	1
0	1	1	(2)	1	0
1	0	0	(1)	0	1
1	0	1	(2)	1	0
1	1	0	(2)	1	0
1	1	1	(3)	1	1

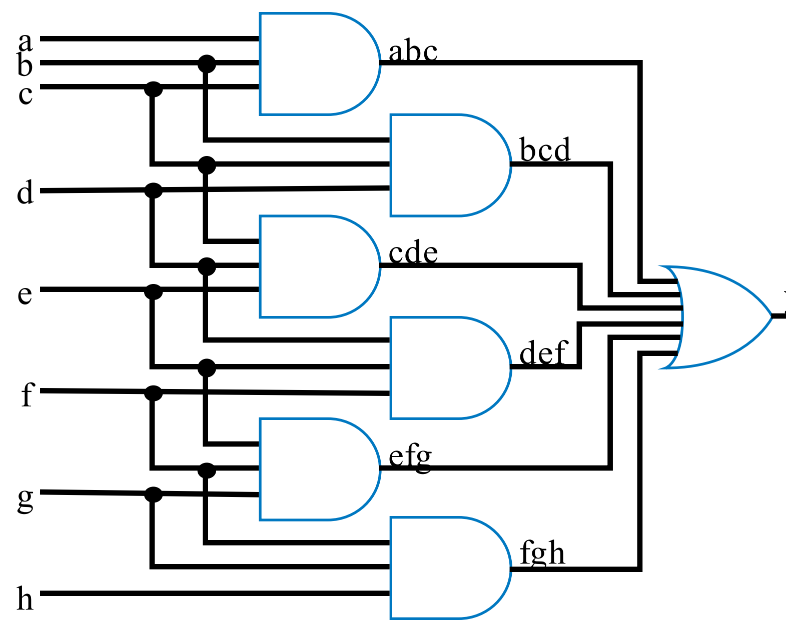




Example: Three 1s Detector

- Problem: Detect three consecutive 1s in 8-bit input: abcdefgh
  - 000**111**01 → 1   10101011 → 0
    11110000 → 1
  - **Step 1: Capture** the function
    - Truth table or equation?
      - Truth table too big: 2^8=256 rows
      - Equation: create terms for each possible case of three consecutive 1s
    - **y = abc + bcd + cde + def + efg + fgh**
  - **Step 2: Convert** to equation -- already done
  - **Step 3: Implement** as a gate-based circuit

a

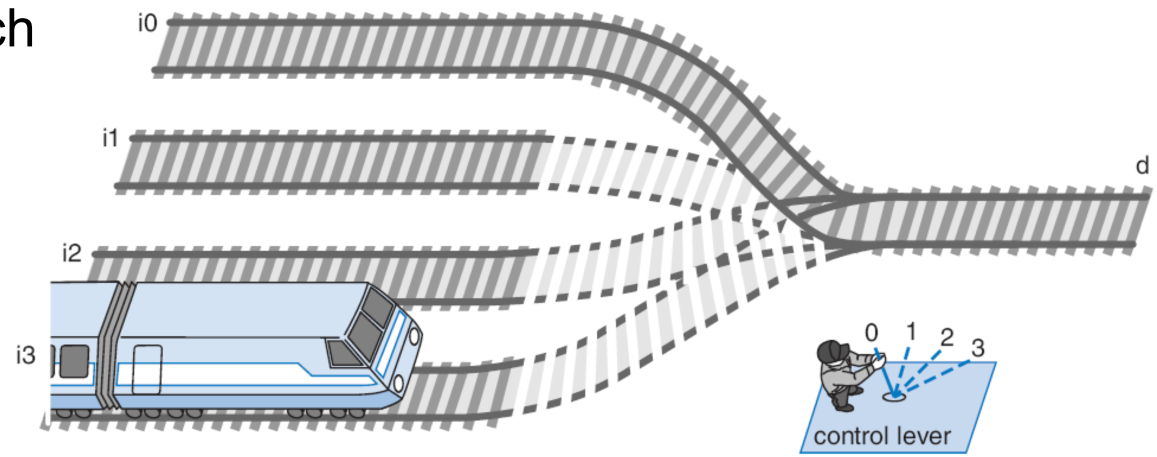


Combinational Building Blocks

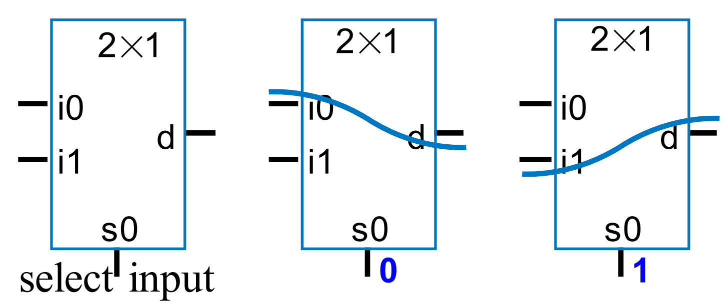
- Combinational building blocks are digital components that
  - Are combinational circuits
  - Implement specific fundamental functions
  - Are used to build bigger combinational circuits
- To learn
  - Multiplexor (MUX)
  - Half adder
  - Full adder
  - Carry-ripple adder
  - Decoder
  - Buffer
  - Tri-state buffer

Multiplexor (Mux)

- Mux: A popular combinational building block
  - Like a railyard switch



  - Routes one of its N data inputs to its one output, based on binary value of select inputs
    - 2 inputs → needs 1 select bit (2 different combinations) to select which input to connect to the output
    - 4 inputs → 2 select bits
    - 8 inputs → 3 select bits
    - N inputs → $\log_2(N)$ select bits



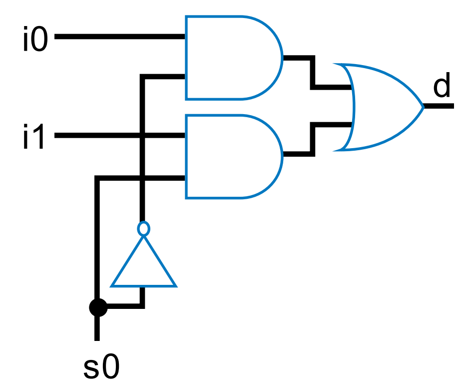
2 to 1 or ***2 by 1*** or ***2x1*** mux

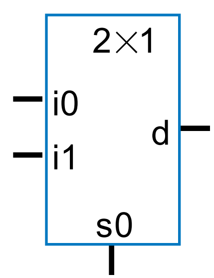
Mux Internal Design

Connect i0 to d when s0 is 0, otherwise connect i1 to d

s0	i0	i1	d
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

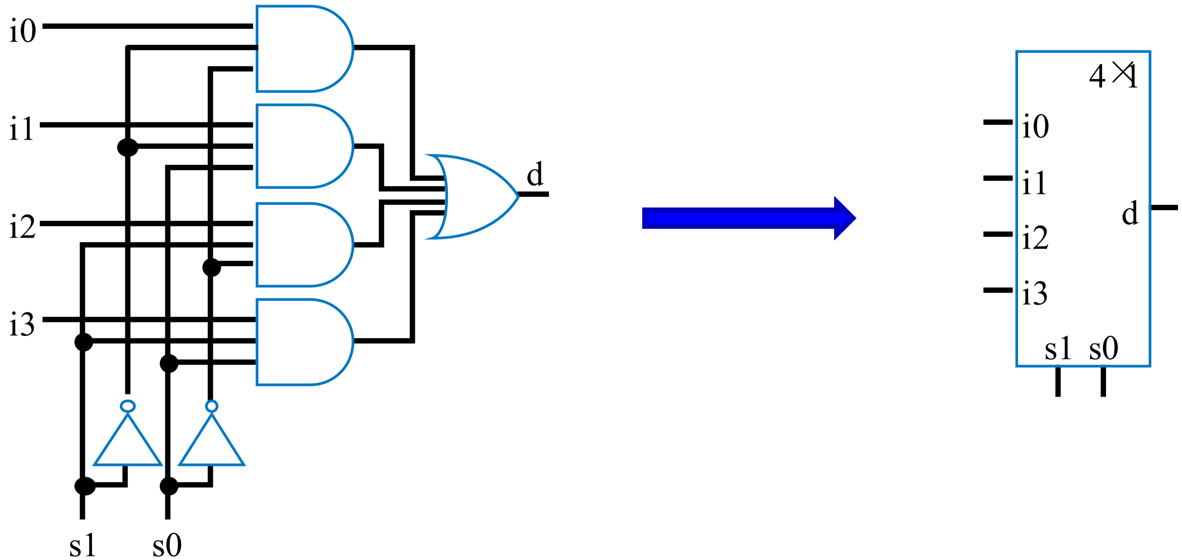
$$d = s0'\, i0\, i1' + s0'\, i0\, i1 + s0\, i0'\, i1 + s0\, i0\, i1$$
$$= s0'\, i0 + s0\, i1$$





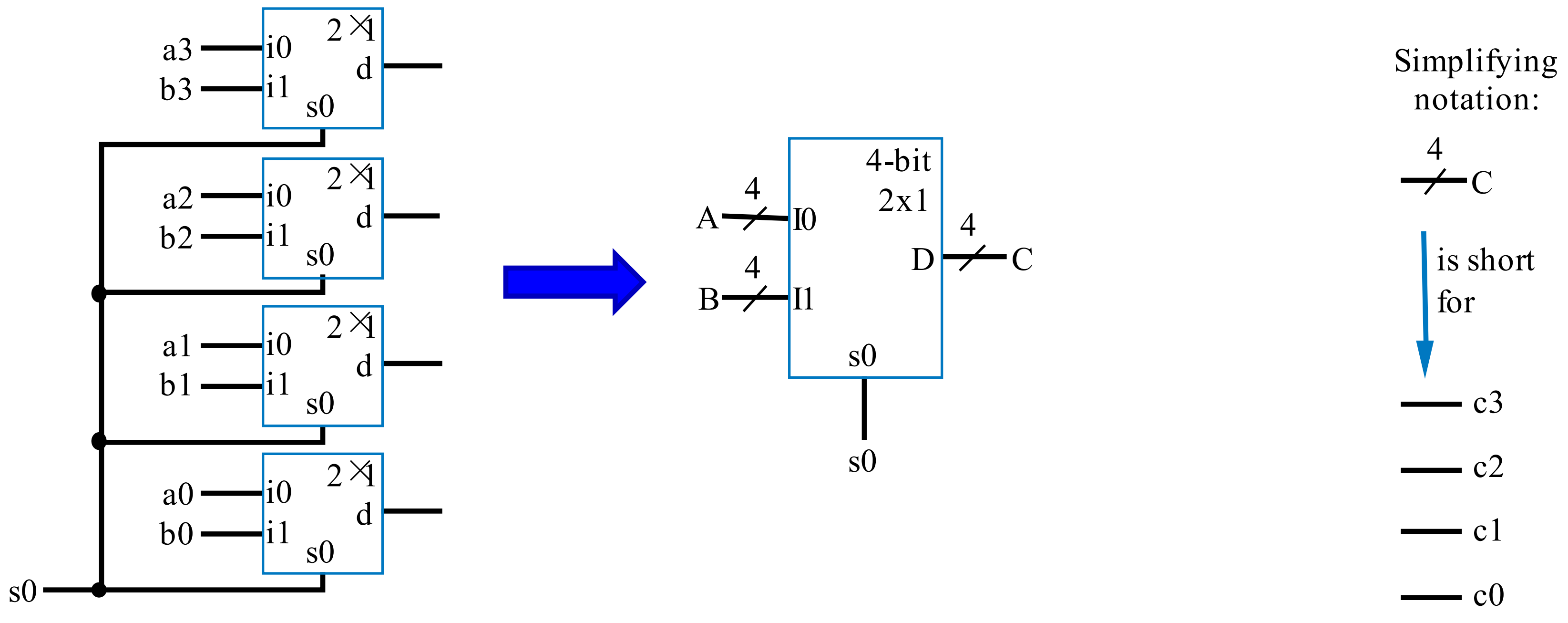
2x1 mux

Mux Internal Design



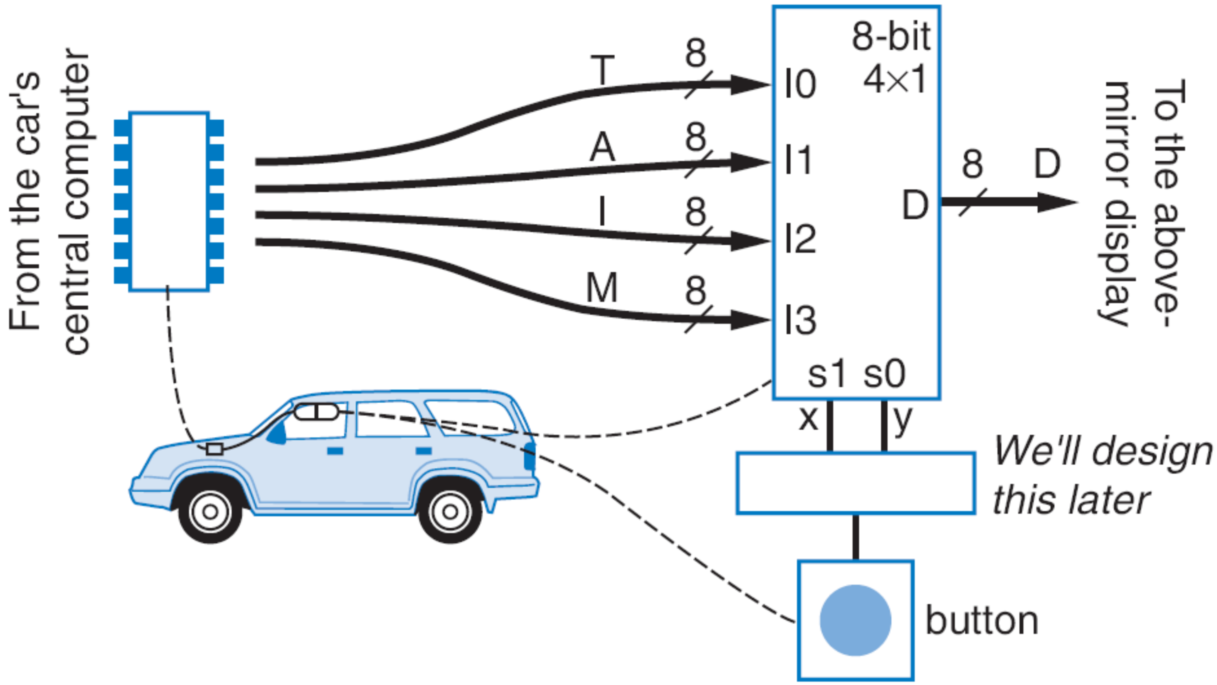
4x1 mux

4-bit 2x1 Mux



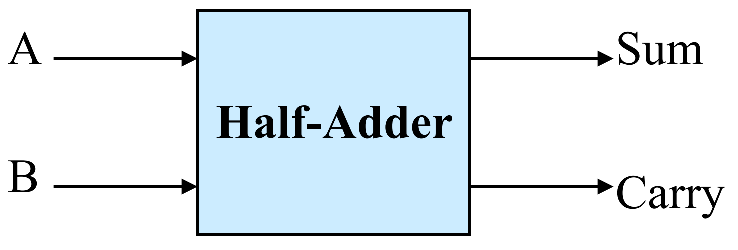
- Ex: Two 4-bit inputs, A (a3 a2 a1 a0), and B (b3 b2 b1 b0)
  - 4-bit 2x1 mux (just four 2x1 muxes sharing a select line) can select between A or B
- Width of input channels may be any, 8, 18, 32, 64, …

N-bit Mux Example



- Four possible display items
  - Temperature (T), Average miles-per-gallon (A), Instantaneous mpg (I), and Miles remaining (M) -- each is 8-bits wide
  - Choose which to display using two inputs x and y
  - Use 8-bit 4x1 mux

Half Adder



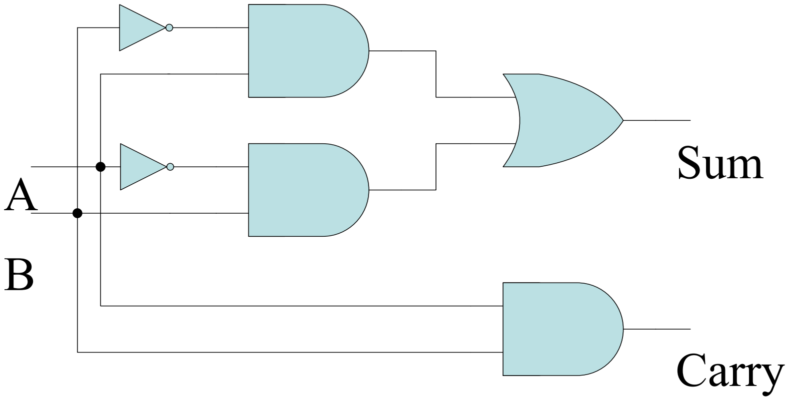
- Addition of two single bits A, B
- Based on the operations it performs, a truth table can be built

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Derive Boolean functions (sum-of-minterms) from the truth table for both outputs

$$\text{Sum} = A'B + AB' = m1 + m2 = \Sigma\,(1, 2)$$
$$= (A+B)(A'+B') = M0 \bullet M3 = \Pi\,(0, 3)$$

$$\text{Carry} = AB = m3$$
$$= (A+B)(A+B')(A'+B)$$
$$= M0 \bullet M1 \bullet M2$$
$$= \Pi\,(0, 1, 2)$$

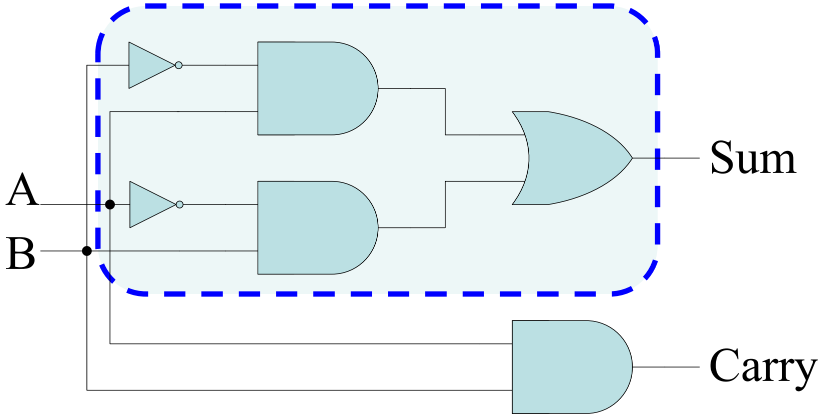


Application of XOR in Half-Adder

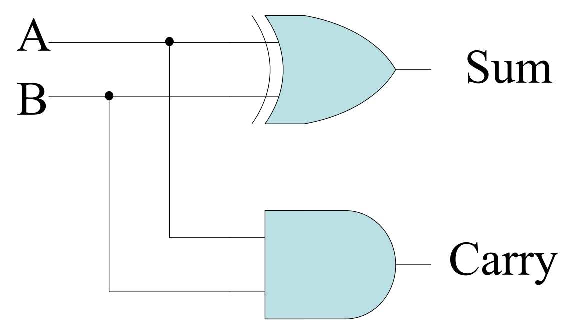
- Half adder

  Sum = A'B + AB' = A ⊕ B

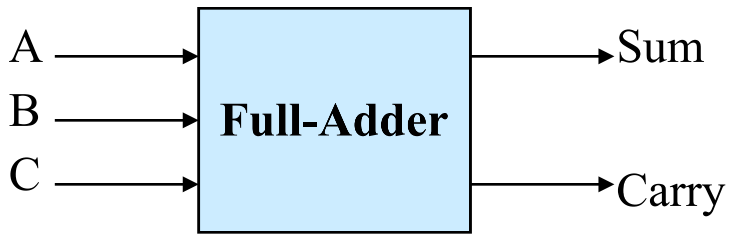
  Carry = AB



⇔



Full Adder



- Derive minterm/Maxterm expressions from the truth table for both outputs

Sum = ?

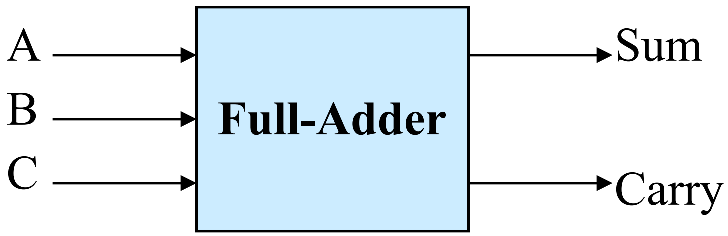
Carry = ?

- Based on the operations it performs, a truth table can be built

A	B	C	Sum	Carry
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1	1	1

Logic Circuit?

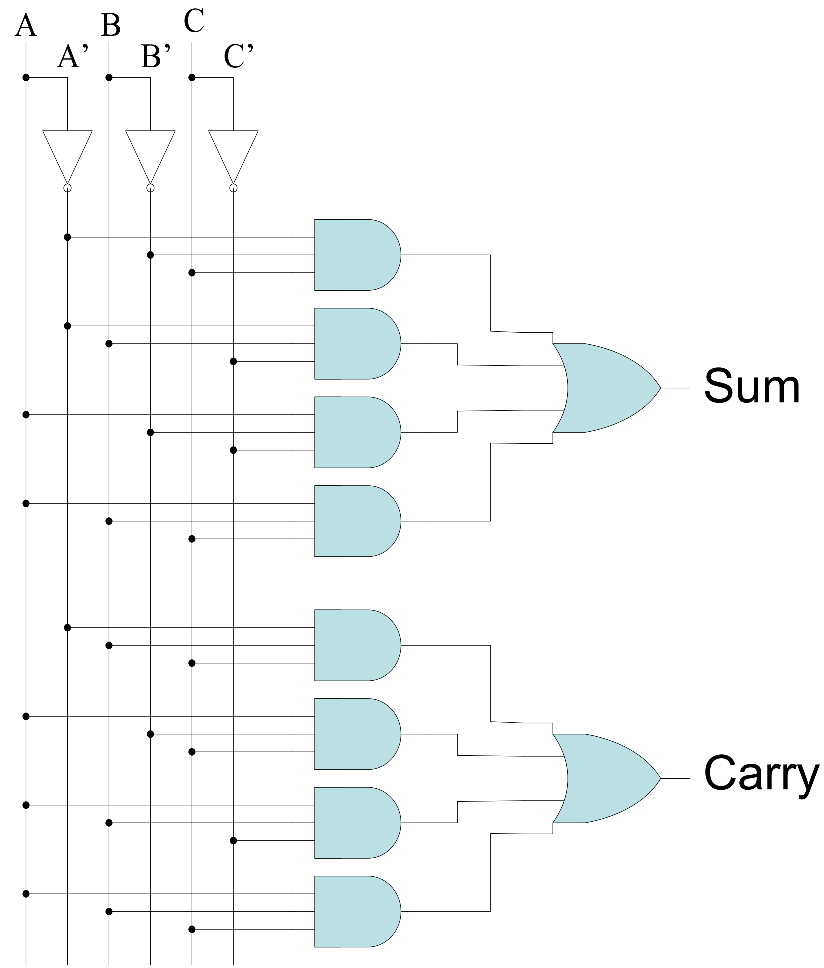
Full Adder



A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Sum = A'B'C + A'BC' + AB'C' + ABC
$= \Sigma\, m(1, 2, 4, 7)$

Carry = A'BC + AB'C + ABC' + ABC
$= \Sigma\, m(3, 5, 6, 7)$



Notice the circuit draw convention

Application of XOR in Full Adder

- Full adder

  Sum = A'B'C + A'BC' + AB'C' + ABC

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

  = A'(B ⊕ C) + A(B ⊕ C)' (let B ⊕ C = D)

  = A'D + AD'

  = A ⊕ D

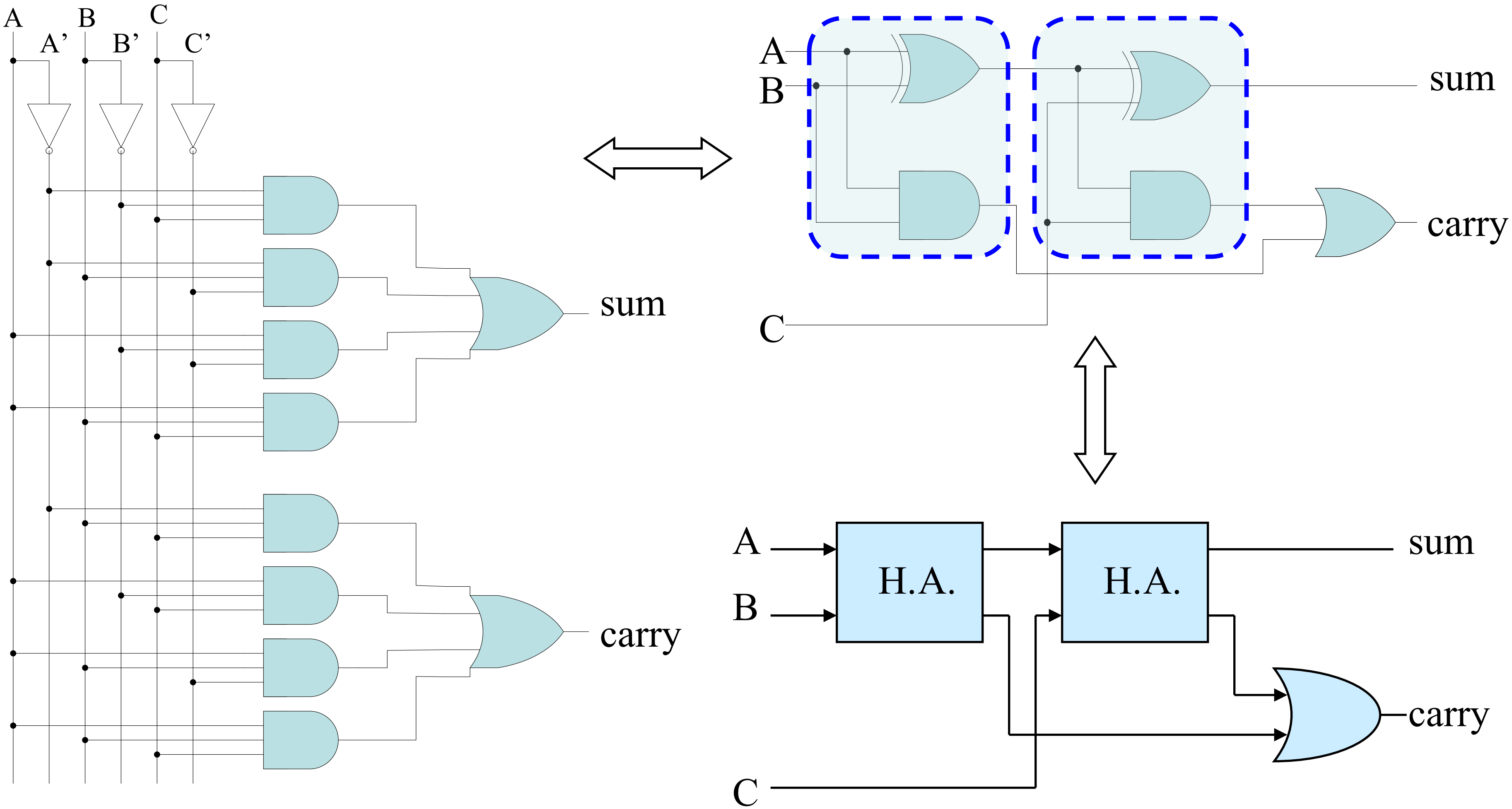
  = A ⊕ B ⊕ C

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

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

  = (A ⊕ B)C + AB

Application of XOR in Full Adder



Multi-bit Number Addition

```
Carry:  0 1 0 0 0
    A:    0 1 0 0  = 4
 +  B:    0 1 0 1  = 5
-----------------------
 Sum:     1 0 0 1  = 9
```

```
Carry:  0 0 0 0 0
          0 1 1 0  = +6
 +        0 0 0 1  = +1
-----------------------
 Sum:     0 1 1 1  = +7
```

A carry bit is the carry output (C_out) from previous stage, it's also the carry input (C_in) to the present stage

Without previous stage, carry should be always 0

```
Carry:  1 1 0 0 0
    A:    1 1 0 0  = 12
 +  B:    1 1 0 1  = 13
-----------------------
 Sum:   1 1 0 0 1  = 25
```

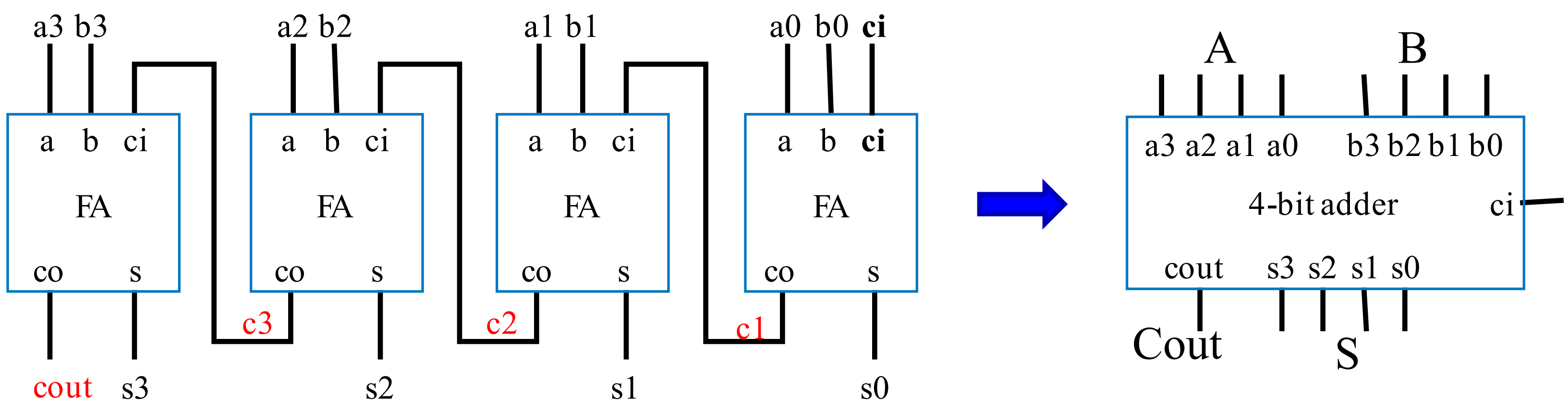
Need one more bit to hold a number bigger than 2^n-1 = 15

Operation may be performed by a full adder

Carry-Ripple Adder

- ***carry-ripple adder***
  - 4-bit adder: Adds two 4-bit numbers, generates 5-bit output
    - 5-bit output can be considered 4-bit “sum” plus 1-bit “carry out”
  - Can easily build any size adder

carries:		c3	c2	c1	cin
B:		b3	b2	b1	b0
A:	+	a3	a2	a1	a0
	cout	s3	s2	s1	s0



(a) (b)

Two's Complement Subtractor

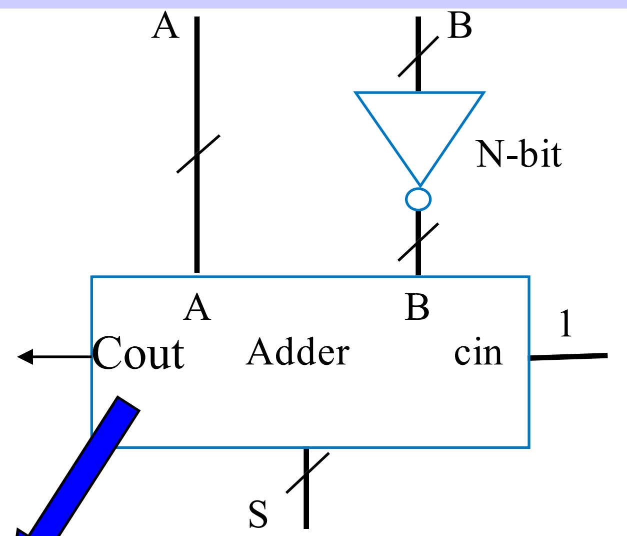
- Using two's complement representation

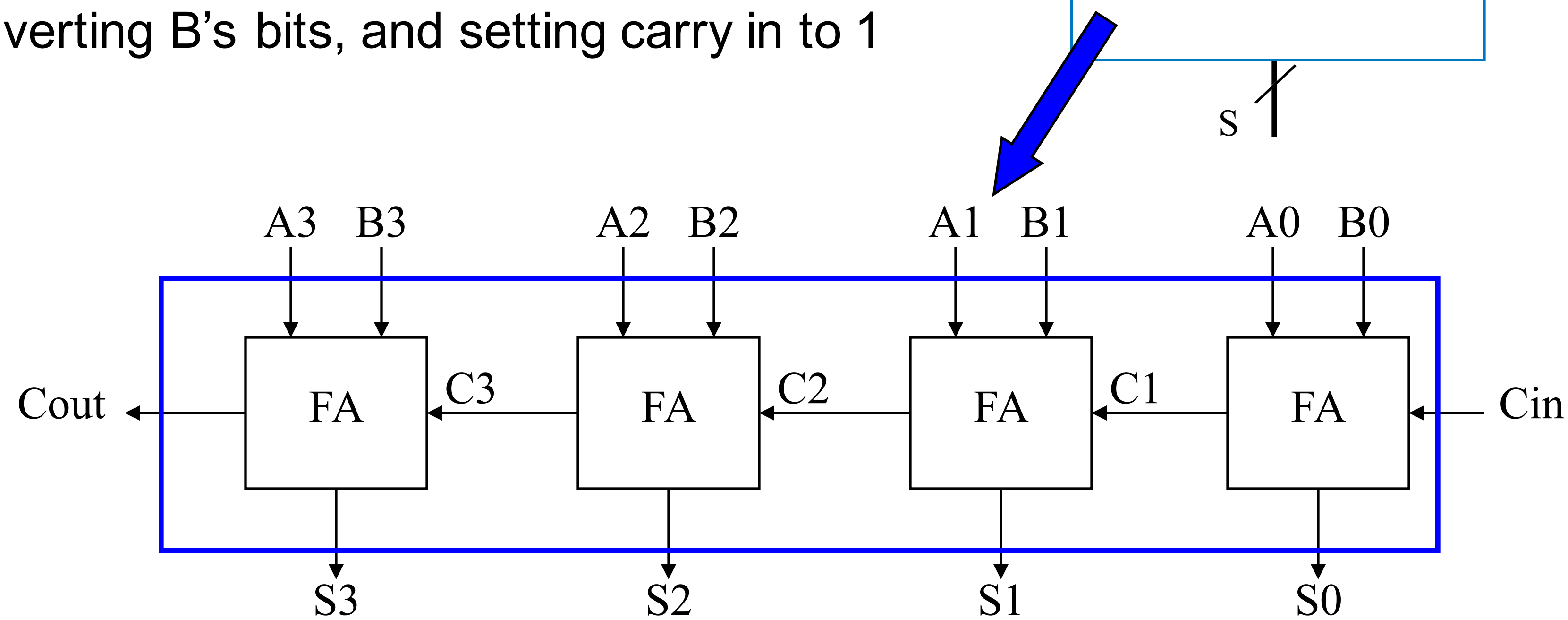
  A – B = A + (-B)

  = A + (two's complement of B)

  = A + invert_bits(B) + 1

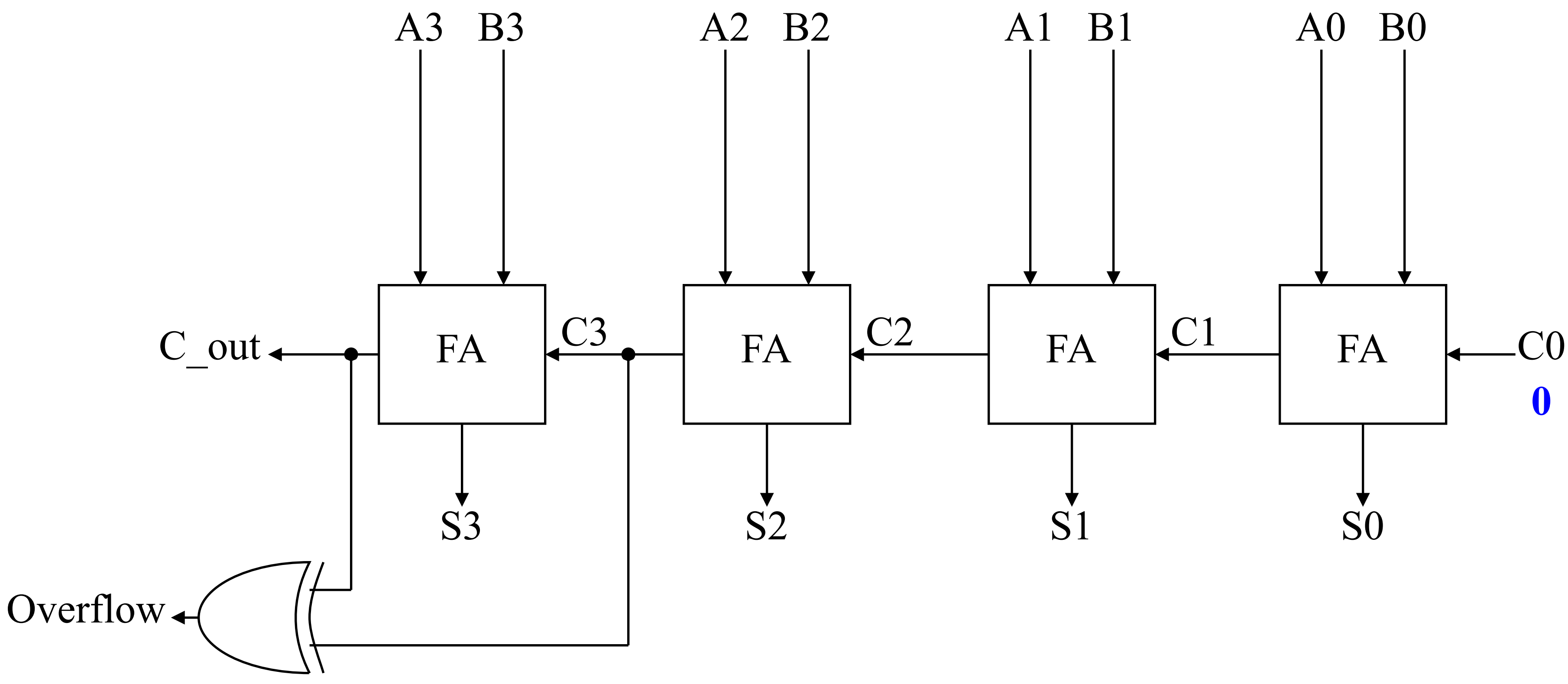
- So build subtractor using adder by inverting B's bits, and setting carry in to 1



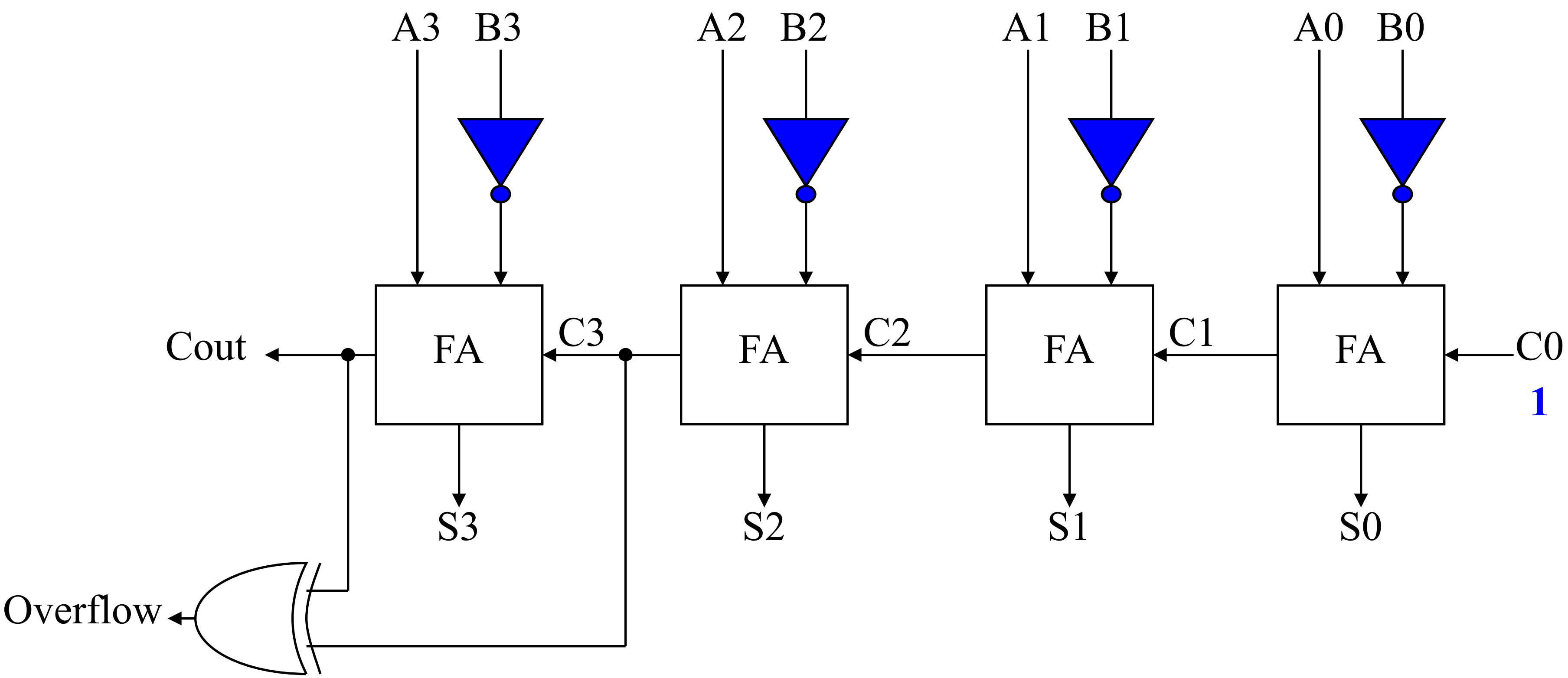


Or
Lookahead Adder

4-Bit 2's Complement Adder



4-Bit 2’s Complement Subtractor



Arithmetic-Logic Unit: ALU

- ***ALU***: Component that can perform any of various arithmetic (add, subtract, increment, etc.) and logic (AND, OR, etc.) operations, based on control inputs
- Key component in computer

TABLE 4.2 Desired calculator operations

Inputs			Operation	Sample output if A=00001111, B=00000101
x	y	z		
0	0	0	S = A + B	S=00010100
0	0	1	S = A - B	S=00001010
0	1	0	S = A + 1	S=00010000
0	1	1	S = A	S=00001111
1	0	0	S = A AND B (bitwise AND)	S=00000101
1	0	1	S = A OR B (bitwise OR)	S=00001111
1	1	0	S = A XOR B (bitwise XOR)	S=00001010
1	1	1	S = NOT A (bitwise complement)	S=11110000

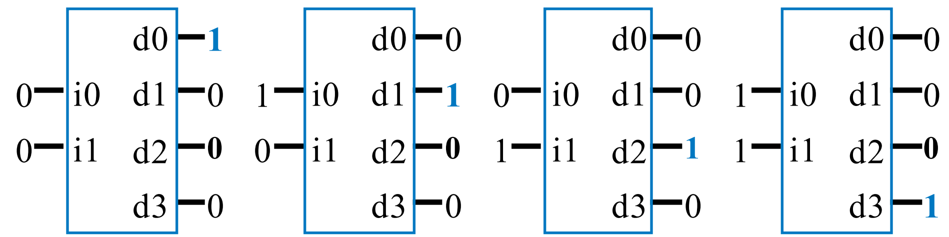
Encoder

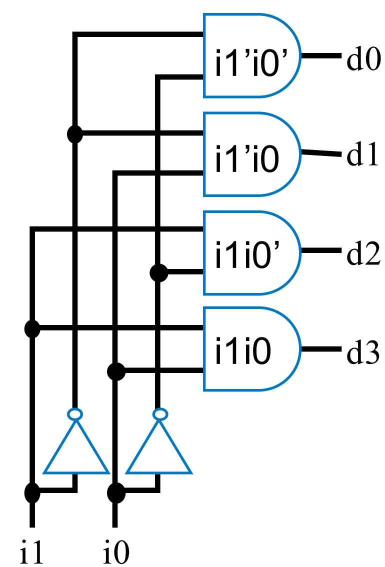
- **Each input is given a binary code**

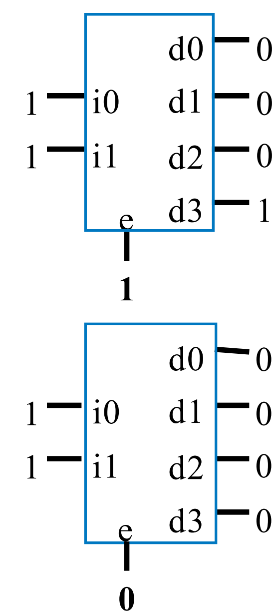
Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**

Decoder

- **Decoder**: Popular combinational logic building block, in addition to logic gates
  - Converts input binary number to one high output
- 2-input decoder: four possible input binary numbers
  - So has four outputs, one for each possible input binary number
- Internal design
  - AND gate for each output to detect input combination
- Decoder with enable e
  - Outputs all 0 if e=0
  - Regular behavior if e=1
- n-input decoder: 2^n outputs





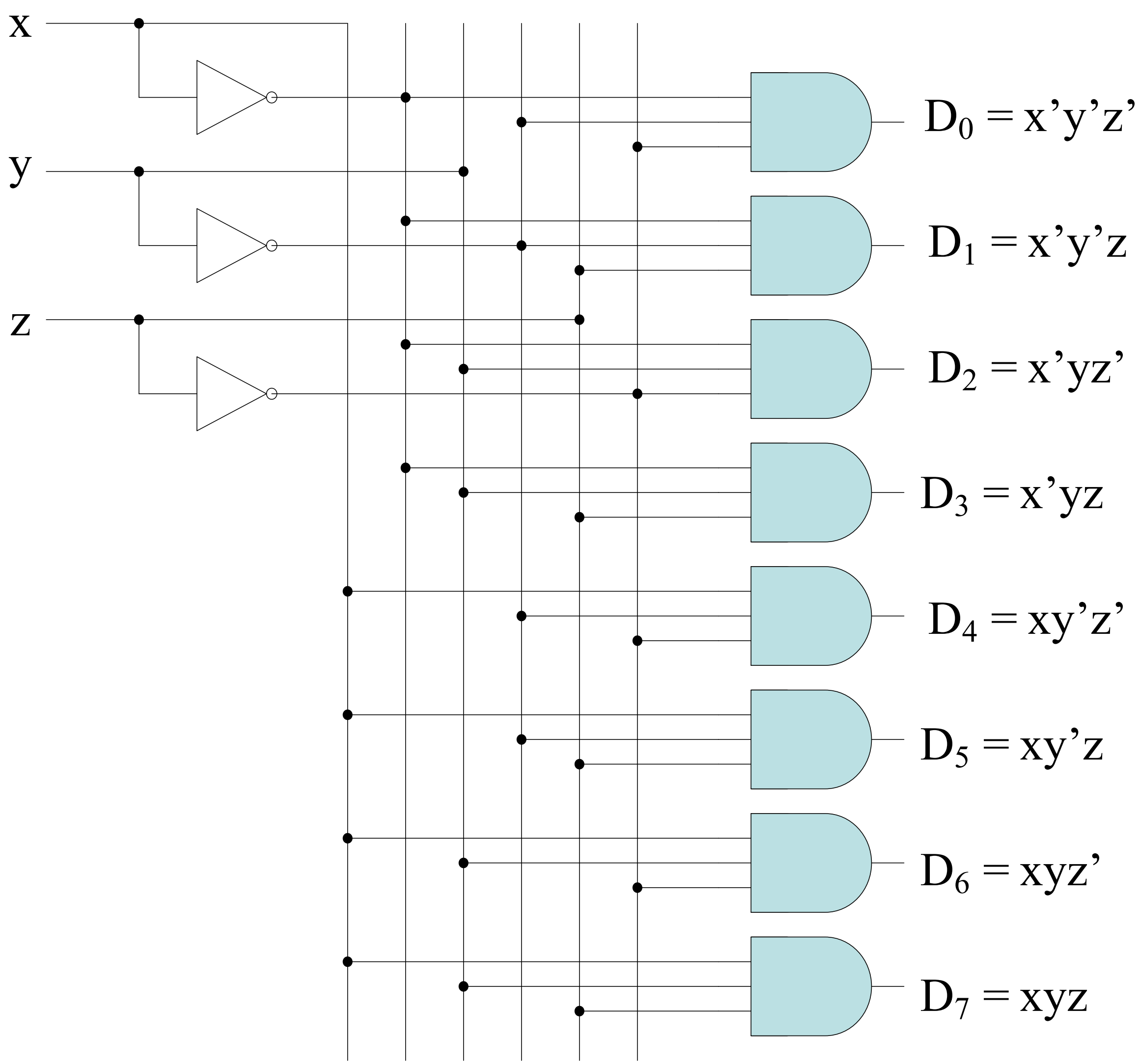


Truth Table of 3×8 Decoder

- **For any input combination, only one of the outputs is turned on**

Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
1	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
1	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

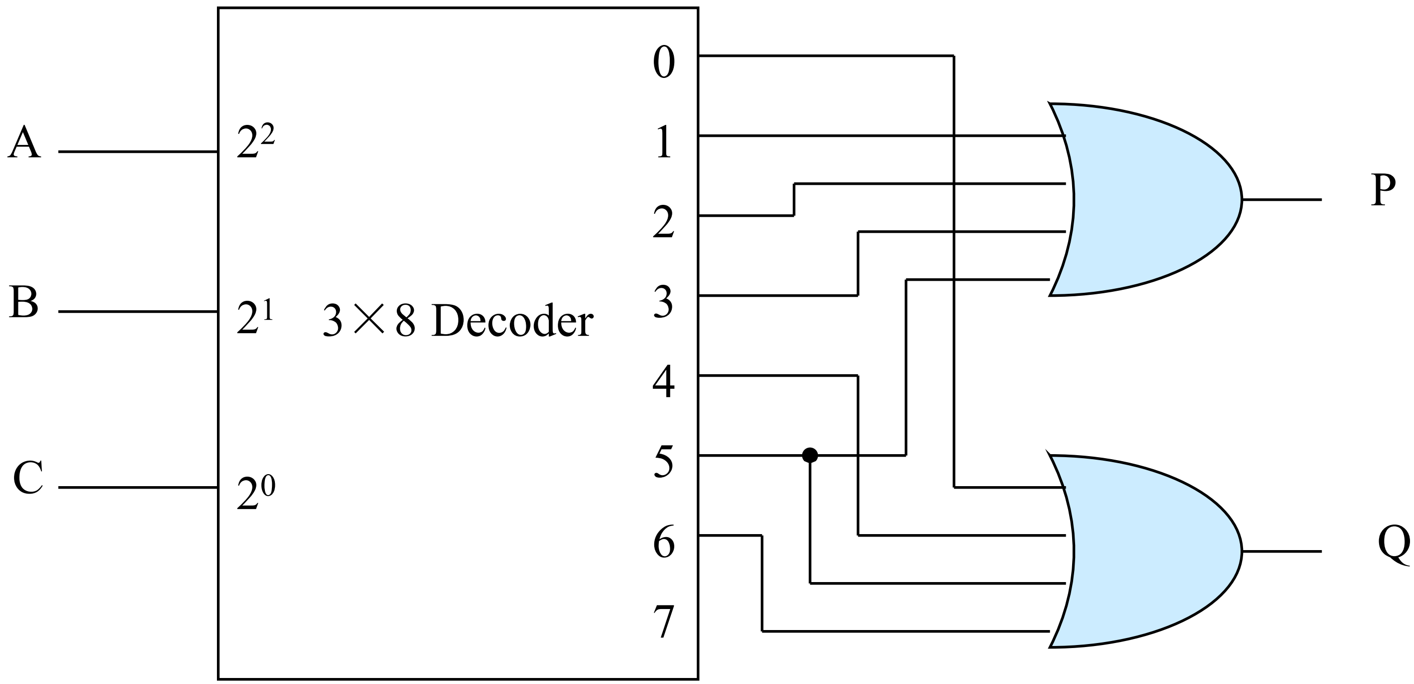
Gate-Level Implementation of 3×8 Decoder



Minterm Generator

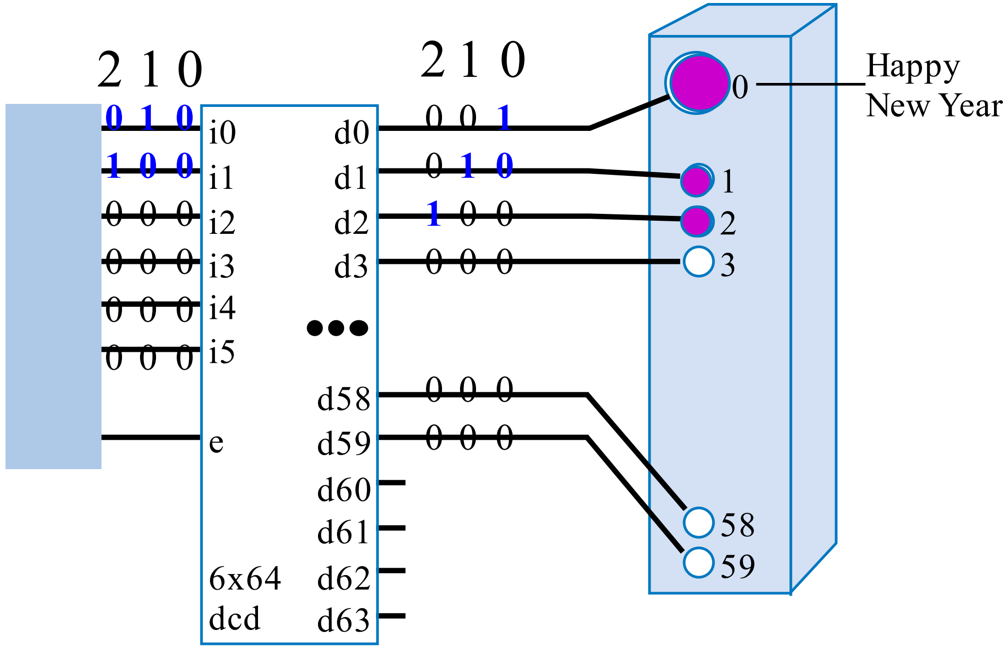
Implementation of Any Circuit with a Decoder

- **P (A, B, C) = Σ m(1, 2, 3, 5) and Q (A, B, C) = Σ m(0, 4, 5, 6)**
- **Since there are three inputs and total of 8 minterms, we can implement the circuits with a 3-to-8-line decoder**



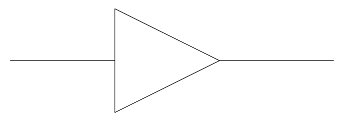
Decoder Example

- New Year's Eve Countdown Display
  - Microprocessor counts from 59 down to 0 in binary on 6-bit output
  - Want illuminate one of 60 lights for each binary number
  - Use 6x64 decoder
    - 4 outputs unused

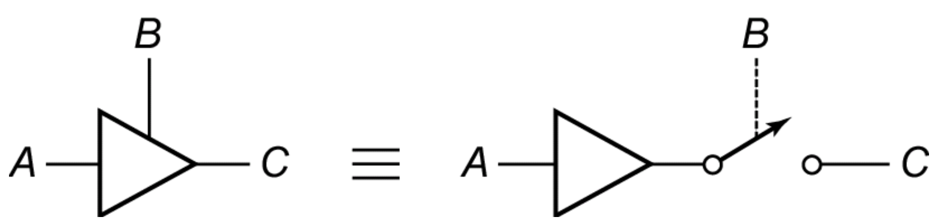


Buffer and Three State (Tri-state) Buffer

- A buffer is a one directional transmission logic device
  - somewhat like a NOT gate without complementing the binary value
  - amplify the driving capability of a signal
  - insert delay
  - Protect input from output

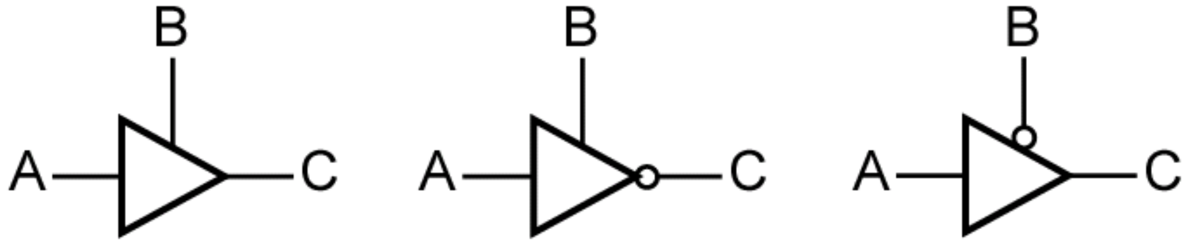


- A three state (Tri-state) buffer can have three different output values, controlled by an enable bit
  - 0 (off) when B = 1, A = 0;
  - 1 (on) when B = 1, A = 1;
  - Z (high impedence, no voltage) when B = 0, A = x;



Tri-state Buffers

- Different types of tri-state buffers

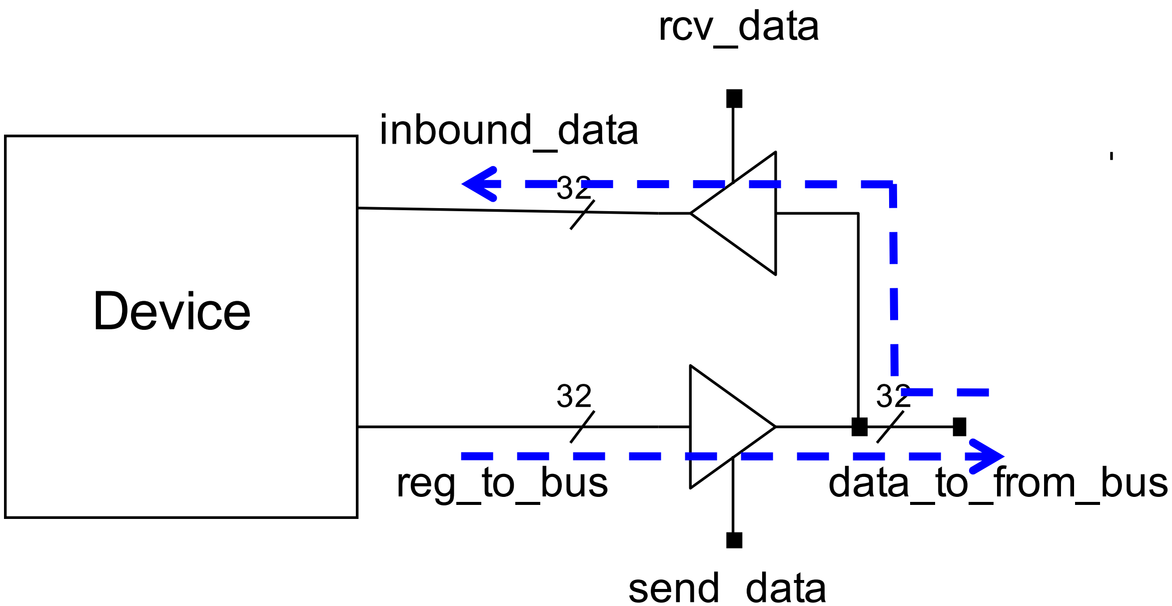


B	A	C
0	0	Z
0	1	Z
1	0	0
1	1	1

B	A	C
0	0	Z
0	1	Z
1	0	1
1	1	0

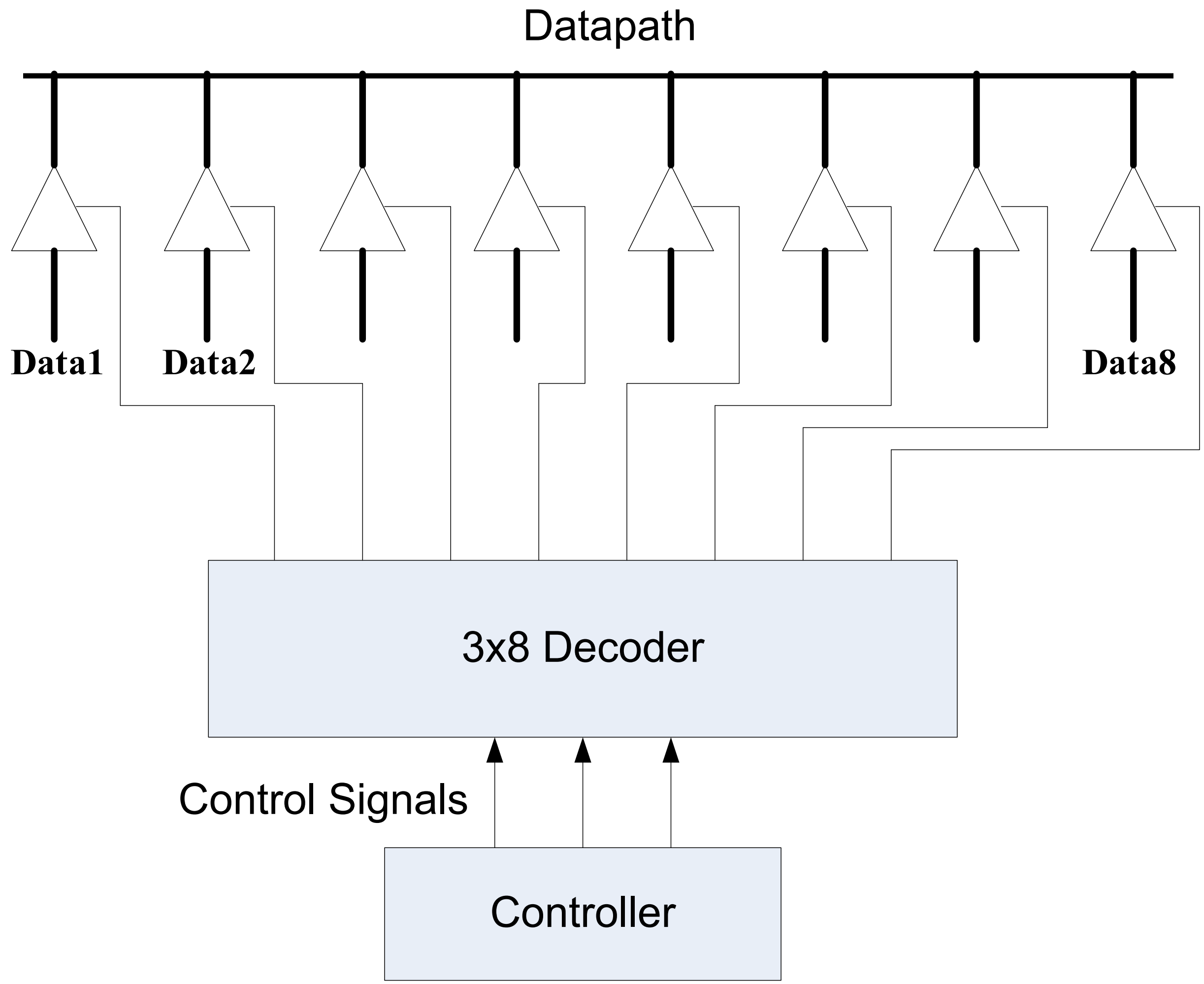
B	A	C
0	0	0
0	1	1
1	0	Z
1	1	Z

Tri-state Buffer Application – Bidirectional I/O Port



Applications of Decoders and Tri-state Buffers

- **Based on the control signals, only one path will be connected**



Alternative Implementation of Datapath Control

- **Based on the control signals, only one path will be connected**