Computer Systems

Kasun Gunawardana

E-mail: kgg

UCSC

University of Colombo School of Computing

Combinational Logic Circuits

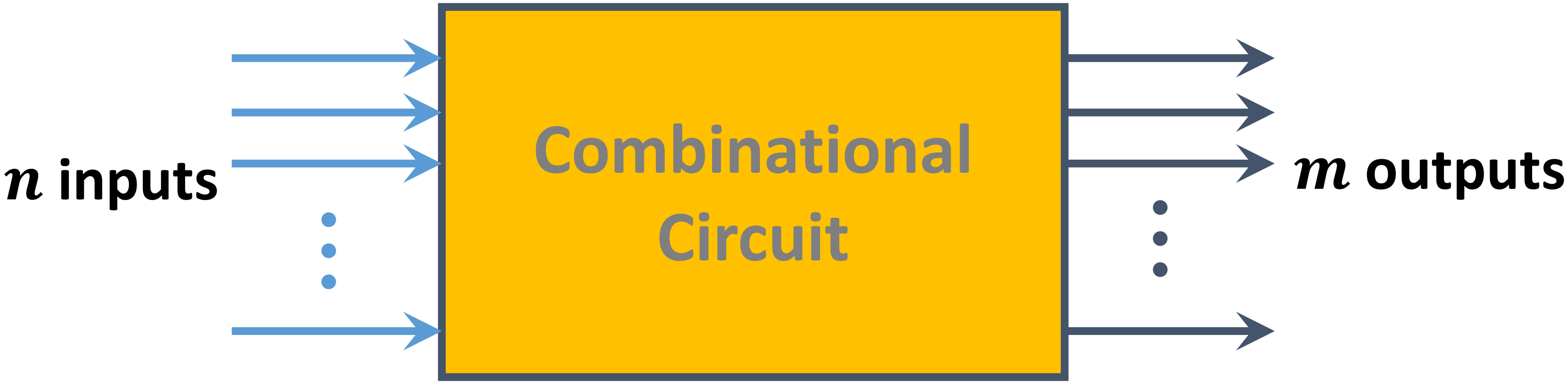
Digital Logic Circuits

- There are two types of digital logic circuits.
  - Combinational Logic Circuits
  - Sequntial Logic Circuits

Combinational vs Sequential

- The output of a combinational circuit depends **only** on its currrent inputs.
    - Purely built upon logic gates
    - No feedback paths, No memory elements
- But, there are times we need circuits that consider not only its present inputs, but also the previous outputs that are stored in storage elements.
- Such circuits are considered as sequential circuits.
- Sequential circuits store previous outputs.

Combinational Circuit



- 2^n possible input patterns
- Circuit can be represented by a truth table
- Circuit can be defined as $\boldsymbol{m}$ Boolean function

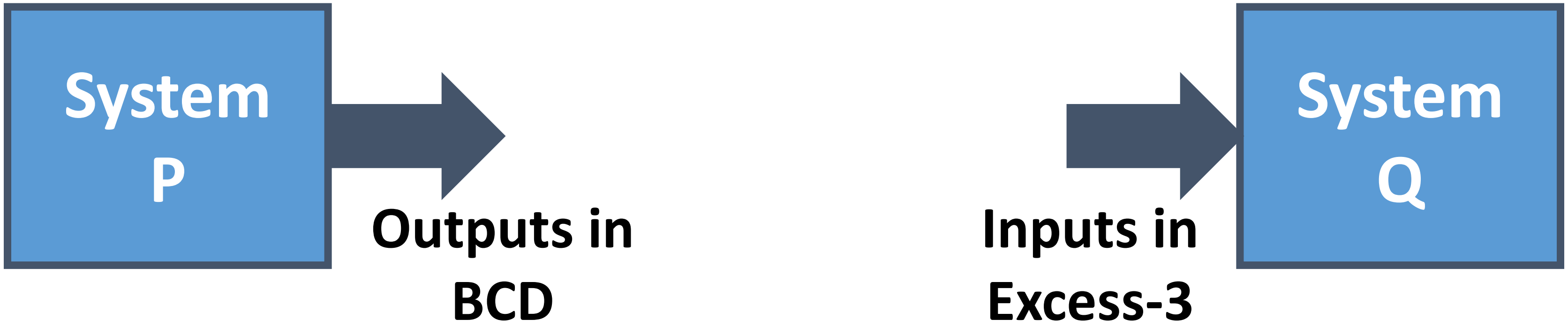
Digital Systems

- There are standard units of combinational circuit in digital systems.
  - Adders, decoders, encoders, multiplexers, etc.
- These are available as integrated circuits (ICs)
  - MSI – Medium Scale Integration circuits
  - VLSI – Very Large Scale Integration circuits

Challenge #1

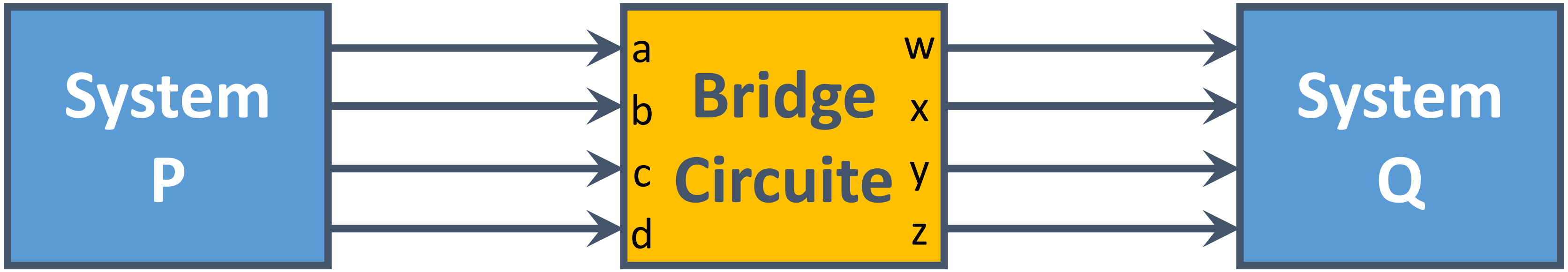
- Assume there are two digital systems, P and Q.
- System P generates 0-9 decimal digits. But digits are encoded in Binary (Binary Coded Decimal).
- System Q takes decimal digit as input, but it should be in Excess-3 encoding.
- Two systems should be conncted as P’s output will be fed into Q.
- But two systems operates on two different encoding systems.
- Therefore, code tranlator should be designed in between P and Q to bridge two systems.

Challenge #1: Block Diagram



- P generates 10 different binary patterns.
  - P has 4 output channels
- 10 different Excess-3 patterns should be fed into Q.
  - Q has 4 input channels

Challenge #1: Block Diagram



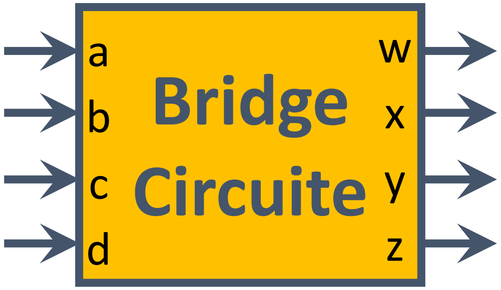
- P generates 10 different binary patterns.
- Therefore we need minimum 4 binary digits for Bridge circuit input.
- Q accepts 10 different excess-3 patterns.
- Therefore we need minimum 4 binary digits for Bridge circuit output.

Conversion Table

Decimal	Binary Coded Decimal				Excess-3			
	a	b	c	d	w	x	y	z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Designing Bridge Circuit

a	b	c	d
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1



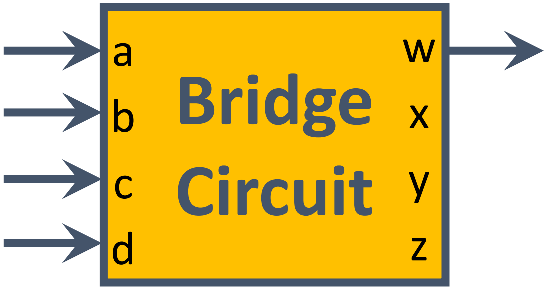
w	x	y	z
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0

Bridge Circuit – Design Approach

- The Bridge Circuit has 4 inputs.
- 2^4=16 input patterns are possible.
- But there are only 10 defined input patterns.
- Remaining patterns are the "Don't Care Conditions"
- There are 4 outputs for the Bridge Circuit.
- Draw K-Maps for all output variables and derive the simplified Boolean logic expressions.

Output w

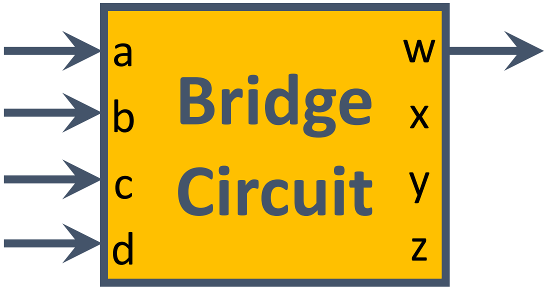
a	b	c	d	w
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1

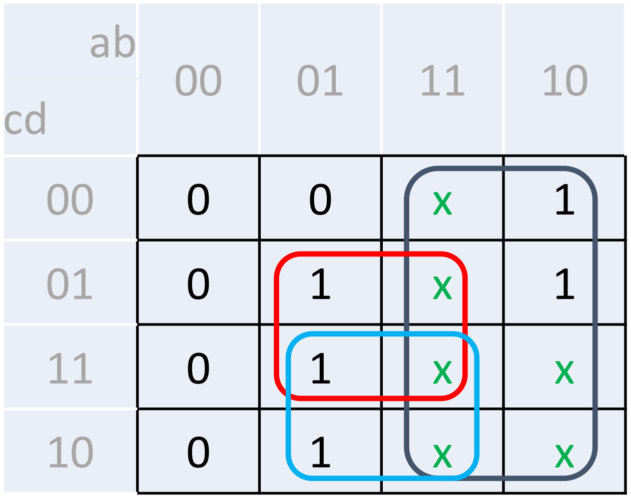


ab cd	00	01	11	10
00	0	0	x	1
01	0	1	x	1
11	0	1	x	x
10	0	1	x	x

Output w

a	b	c	d	w
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
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



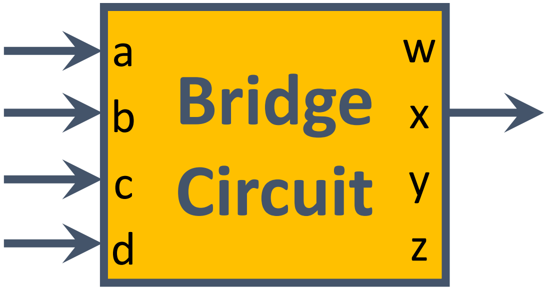


cd \ ab	00	01	11	10
00	0	0	x	1
01	0	1	x	1
11	0	1	x	x
10	0	1	x	x

$$\boxed{w = a + bc + bd}$$

Output x

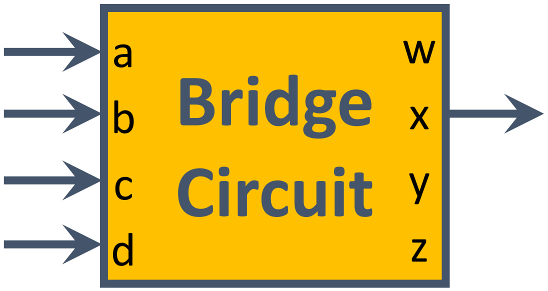
a	b	c	d	x
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1

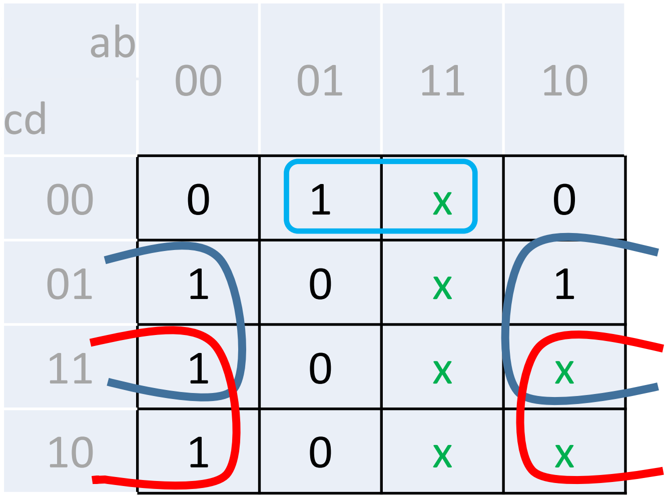


ab cd	00	01	11	10
00	0	1	x	0
01	1	0	x	1
11	1	0	x	x
10	1	0	x	x

Output x

a	b	c	d	x
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1



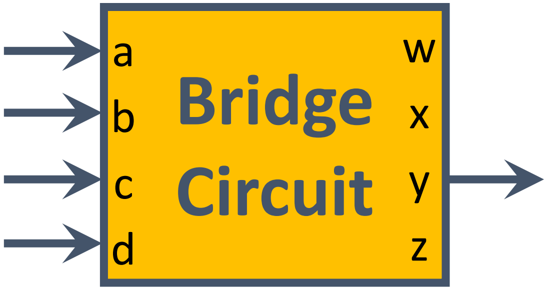


ab / cd	00	01	11	10
00	0	1	x	0
01	1	0	x	1
11	1	0	x	x
10	1	0	x	x

$$x = b'c + b'd + bc'd'$$

Output y

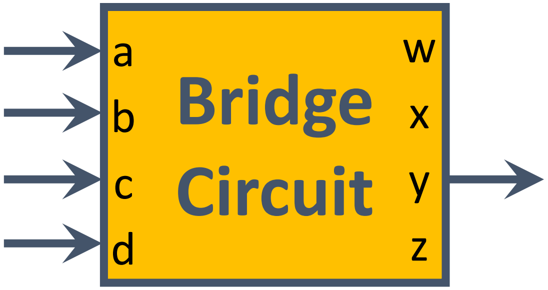
a	b	c	d	y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0



ab cd	00	01	11	10
00	1	1	x	1
01	0	0	x	0
11	1	1	x	x
10	0	0	x	x

Output y

a	b	c	d	y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0

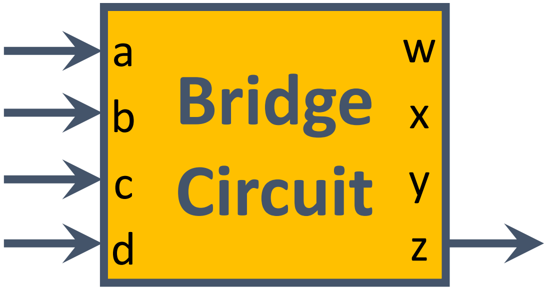


ab cd	00	01	11	10
00	1	1	x	1
01	0	0	x	0
11	1	1	x	x
10	0	0	x	x

$$y = c'd' + cd$$

Output z

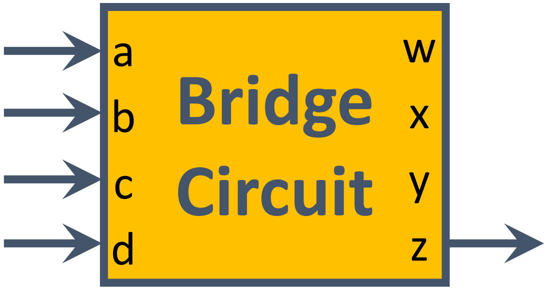
a	b	c	d	z
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0

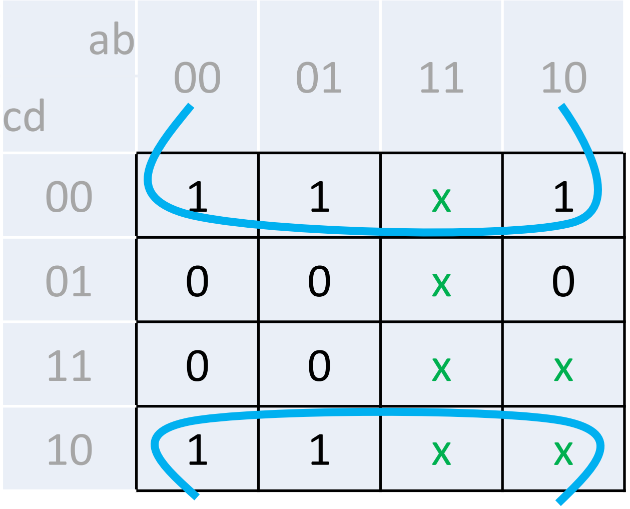


ab cd	00	01	11	10
00	1	1	x	1
01	0	0	x	0
11	0	0	x	x
10	1	1	x	x

Output z

a	b	c	d	z
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0

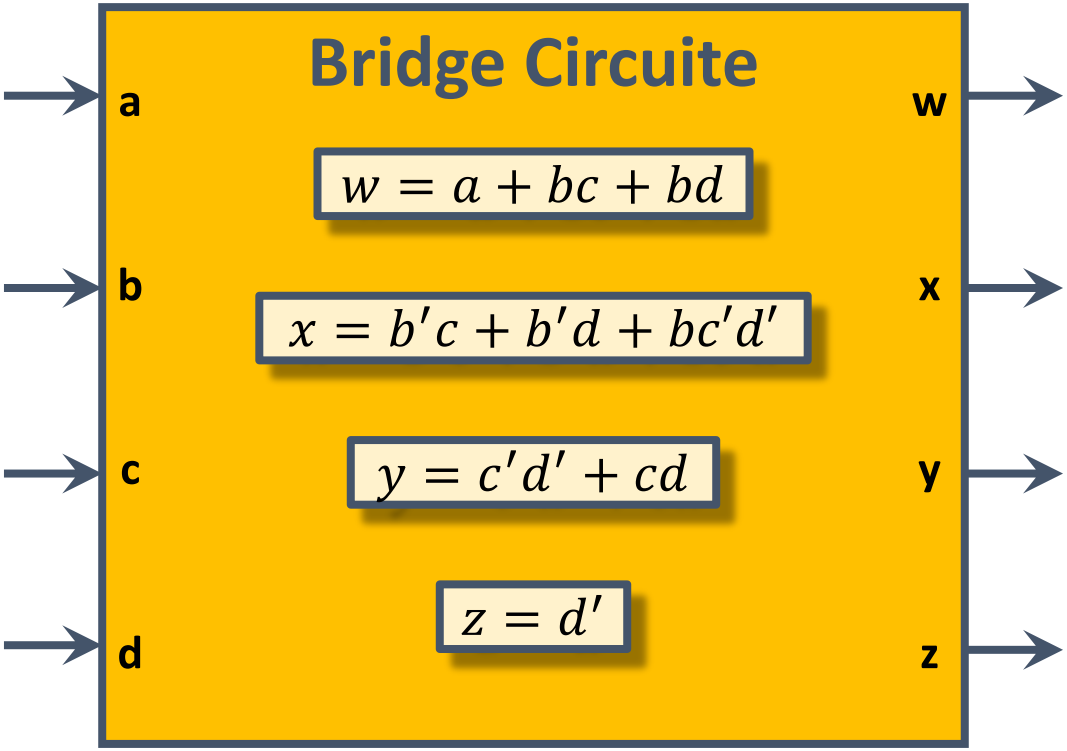




ab cd	00	01	11	10
00	1	1	x	1
01	0	0	x	0
11	0	0	x	x
10	1	1	x	x

$$z = d'$$

K-Maps for Four w, x, y and z



Bridge Circuite

$$w = a + bc + bd$$

$$x = b'c + b'd + bc'd'$$

$$y = c'd' + cd$$

$$z = d'$$

Next…

Binary Adder

Computer Systems

Kasun Gunawardana

E-mail: kgg

University of Colombo School of Computing

Binary Adders

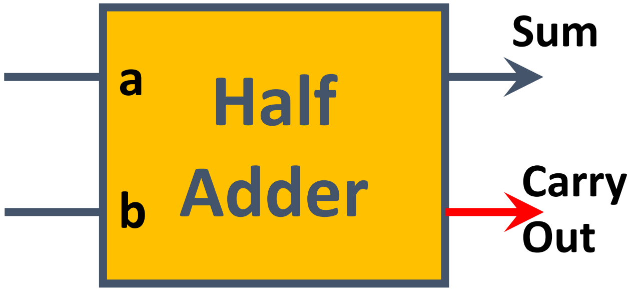
Binary Adder

- **ADD** is an important operation.
- Most basic version of **ADD** is adding two digits.
- Combinational circuit that adds two bits is called "Half Adder".
- Half adder cannot accommodate a carry in bit.

Half Adder

- Adds only two bits.
- Four possible cases.

0	0	1	1
0	1	0	1
0	1	1	1 0



- In the final case there is a carry out.
- Therefore, Half Adder circuit has 2 inputs and 2 outputs.

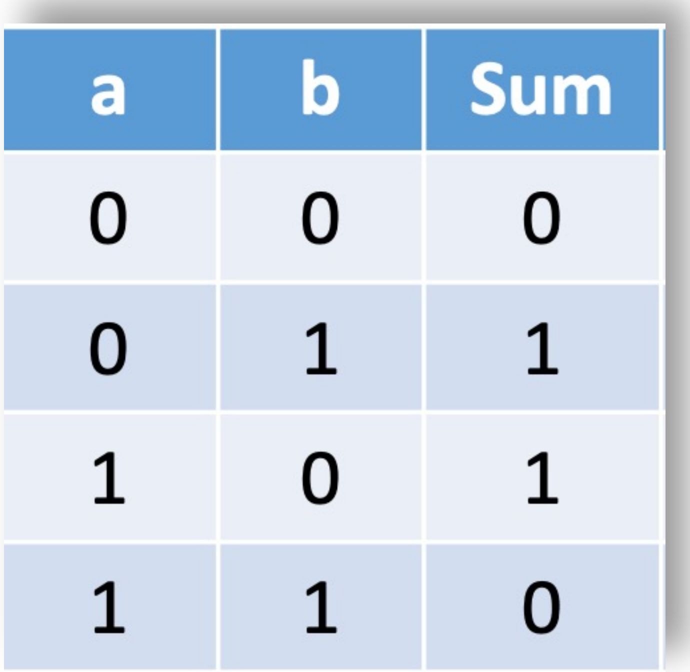
Half Adder Ciricuit Design

- Let’s have the truth table.
- Two inputs, one for each bit
- Two outputs, Sum and Carry
- We can derive logic expressions for two outputs separately.

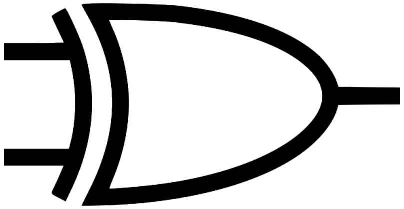
Inputs		Outputs	
a	b	Sum	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half Adder Ciricuit Design

Inputs		Outputs	
a	b	Sum	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



a	b	Sum
0	0	0
0	1	1
1	0	1
1	1	0

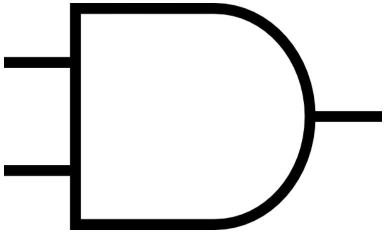


XOR Gate

Half Adder Ciricuit Design

Inputs		Outputs	
a	b	Sum	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

a	b	Carry Out
0	0	0
0	1	0
1	0	0
1	1	1

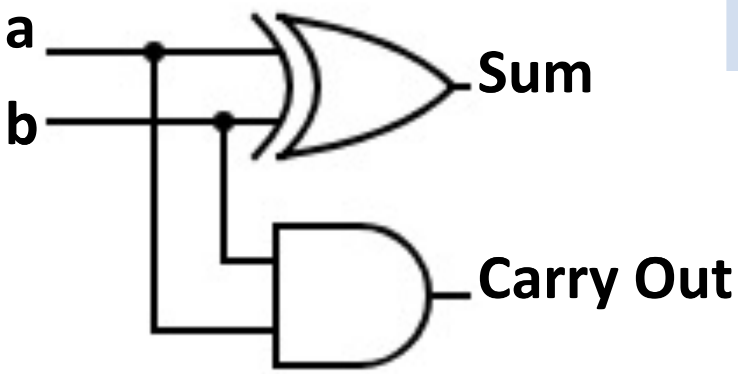


AND Gate

Half Adder - Circuit

- Half Adder circuit can be designed as below.

Inputs		Outputs	
a	b	Sum	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



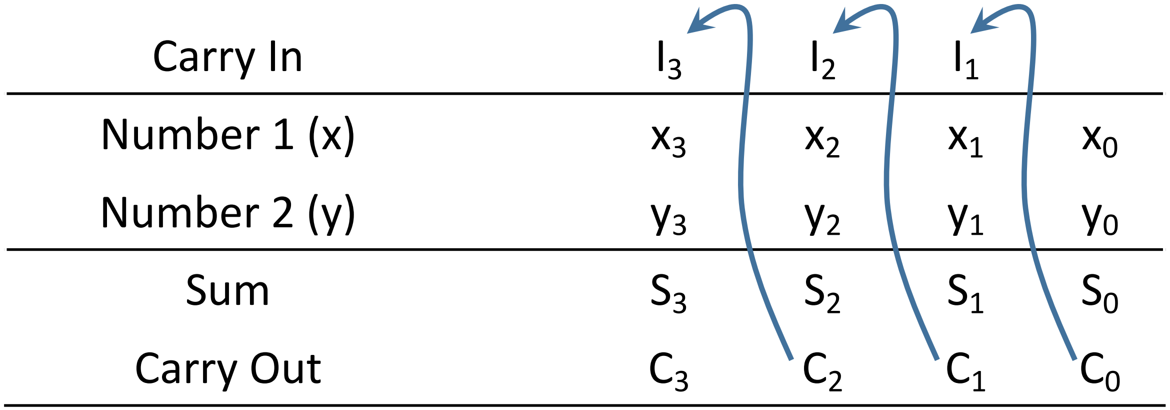
Full Adder

- Half Adder can only add two single bit numbers.
- But digital systems should be able to add two numbers where each number has more than 1 bit.
- Let’s assume two 4-bit numbers, x and y,

Carry In	I_3	I_2	I_1	
Number 1 (x)	x_3	x_2	x_1	x_0
Number 2 (y)	y_3	y_2	y_1	y_0
Sum	S_3	S_2	S_1	S_0
Carry Out	C_3	C_2	C_1	C_0

Full Adder

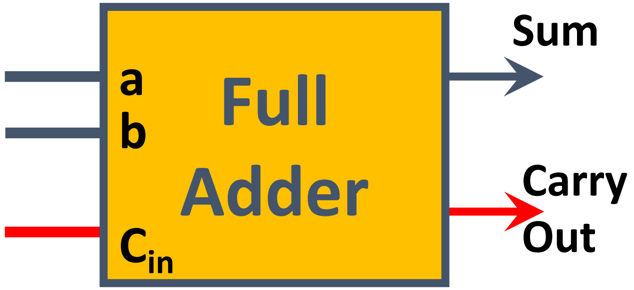
- Half Adder can only add two single bit numbers.
- But digital systems should be able to add two numbers where each number has more than 1 bit.
- Let’s assume two 4-bit numbers, x and y,



Carry In	I_3	I_2	I_1	
Number 1 (x)	x_3	x_2	x_1	x_0
Number 2 (y)	y_3	y_2	y_1	y_0
Sum	S_3	S_2	S_1	S_0
Carry Out	C_3	C_2	C_1	C_0

Full Adder

- To be a complete addition unit, a circuit should have,
  - INPUT: Two bits to be added
  - INPUT: A bit represents Carry in
  - OUTPUT: A bit to hold Sum of all inputs
  - OUTPUT: A bit to hold Carry out



Full Adder - Truth Table

Inputs			Outputs	
a	b	C_{in}	Sum	C_{out}
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

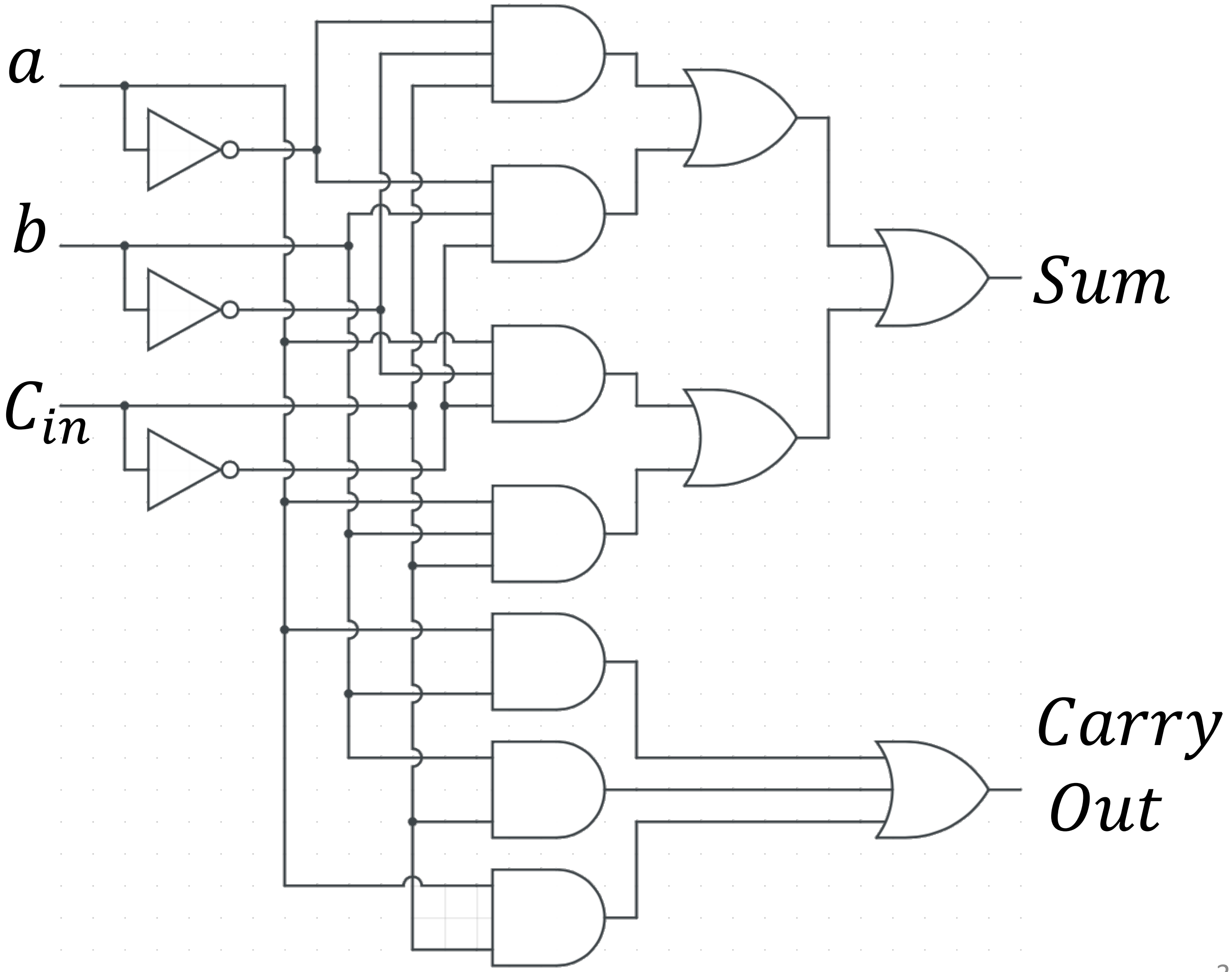
$After\ simplification$

$$Sum = a'b'C_{in} + a'bC_{in}' + ab'C_{in}' + abC_{in}$$

$$C_{out} = ab + bC_{in} + aC_{in}$$

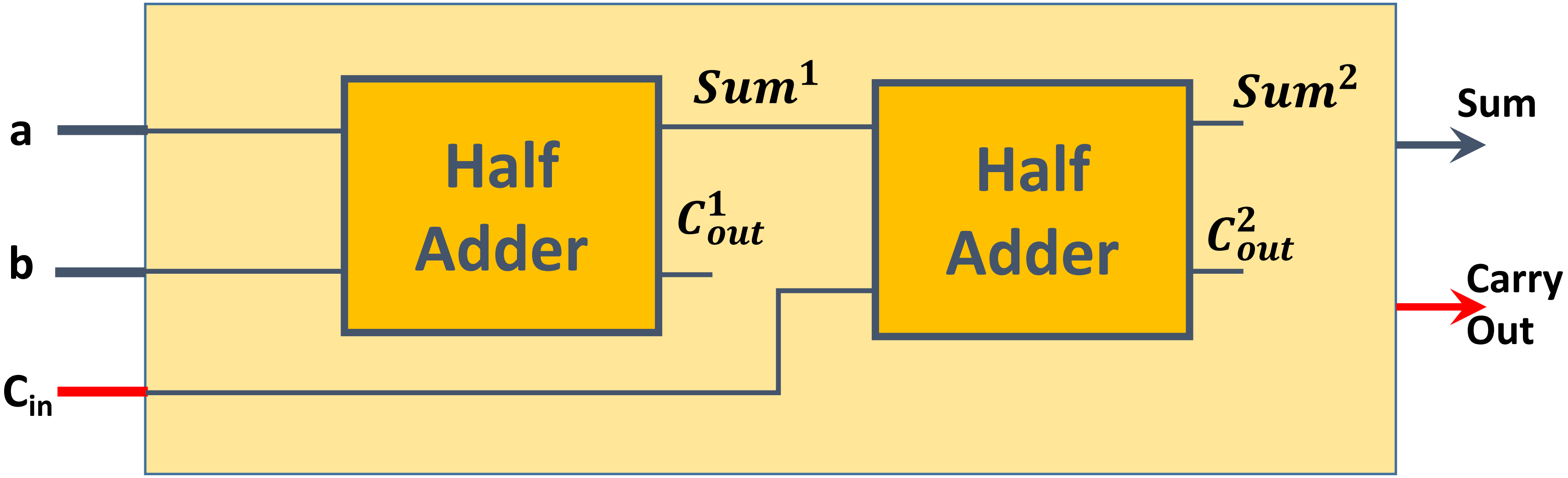
$$Sum = a'b'C_{in} + a'bC_{in}' + ab'C_{in}' + abC_{in}$$

$$C_{out} = ab + bCin + aC_{in}$$



Full Adder – Alternative Approach

- Full adder takes 3 inputs (3 bits).
- Half adder can work with 2 inputs (2 bits)



Sum

a	b	Sum^1	C_{in}	Sum^2	Sum (Expected)
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	0	0	0	0
0	0	0	1	1	1
0	1	1	1	0	0
1	0	1	1	0	0
1	1	0	1	1	1

Carry Out

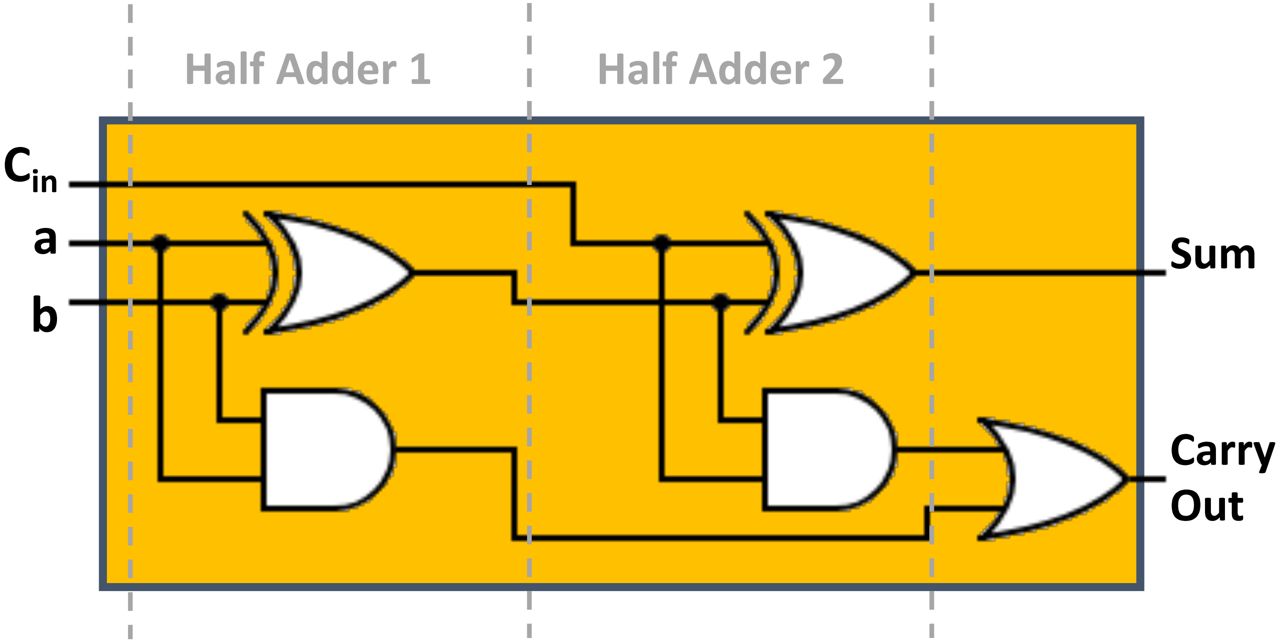
a	b	S_1	C_{out}^1	C_{in}	S_2	C_{out}^2	Carry Out (Expected)
0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0
1	0	1	0	0	1	0	0
1	1	0	1	0	0	0	1
0	0	0	0	1	1	0	0
0	1	1	0	1	0	1	1
1	0	1	0	1	0	1	1
1	1	0	1	1	1	0	1

Carry Out

a	b	S_1	C^1_{out}	C_{in}	S_2	C^2_{out}	C_{out}
0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0
1	0	1	0	0	1	0	0
1	1	0	1	0	0	0	1
0	0	0	0	1	1	0	0
0	1	1	0	1	0	1	1
1	0	1	0	1	0	1	1
1	1	0	1	1	1	0	1

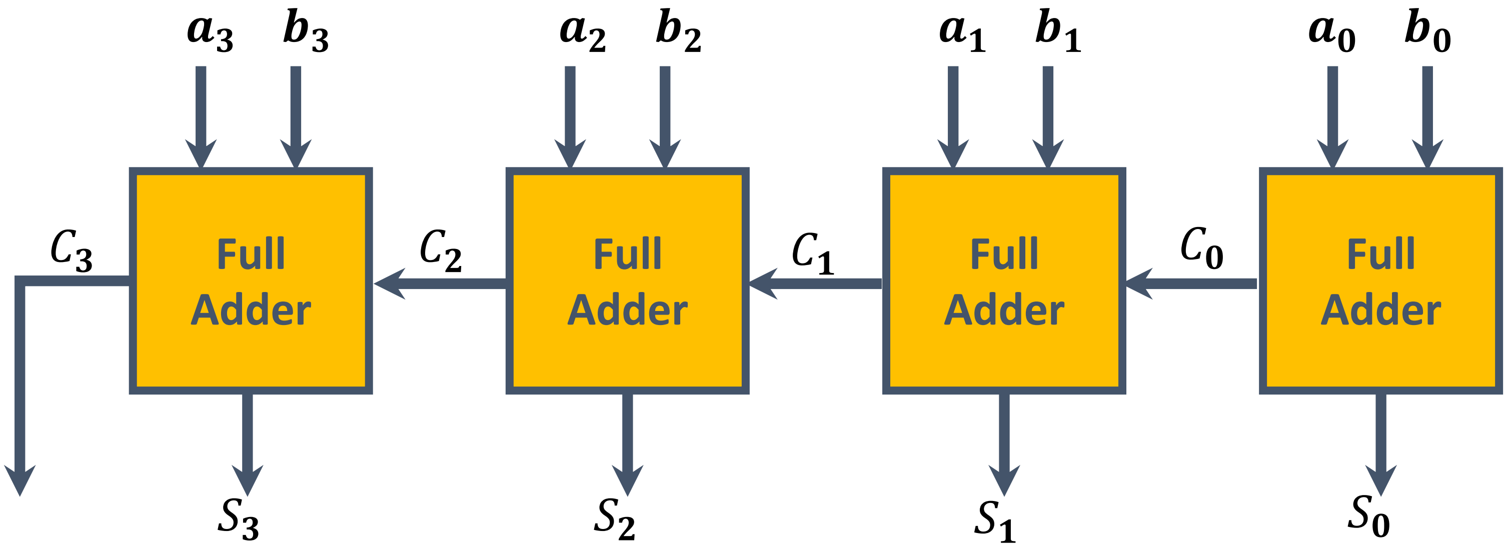
C^1_{out}	C^2_{out}	C_{out}
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**X**

Full Adder with Half Adders



4-bit Adder

Number 1	$a_3\ a_2\ a_1\ a_0$
Number 2	$b_3\ b_2\ b_1\ b_0$



Ripple Carry Adder

Next…

Binary Subtractor

Computer Systems

Kasun Gunawardana

E-mail: kgg

UCSC

University of Colombo School of Computing

Binary Subtractor

Subtractor

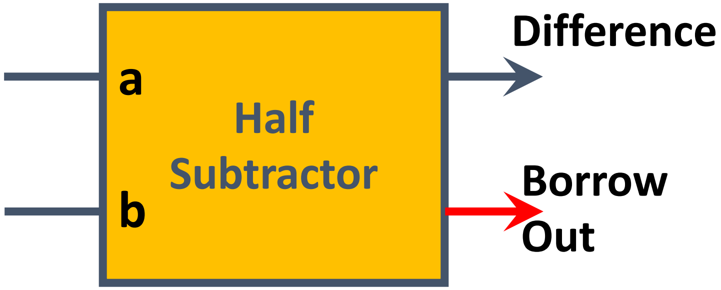
- Similar to Adders we can design,
  - Half Subtractor
  - Full Subtractor

Number 1	0	0	1	1	(Minuend)
Number 2	0	1	0	1	(Subtrahend)
Difference	0	1	1	0	
Borrow Out	0	1	0	0	

The **borrow out** is set when the subtractor needs to borrow from the next digit in a multi-digit subtraction.

Half Subtractor

- INPUTS: Two single bit numbers (two bits)
- OUTPUTS: Difference and Borrow out
- Does not consider *Borrowing in*.



Half Subtractor - Truth Table

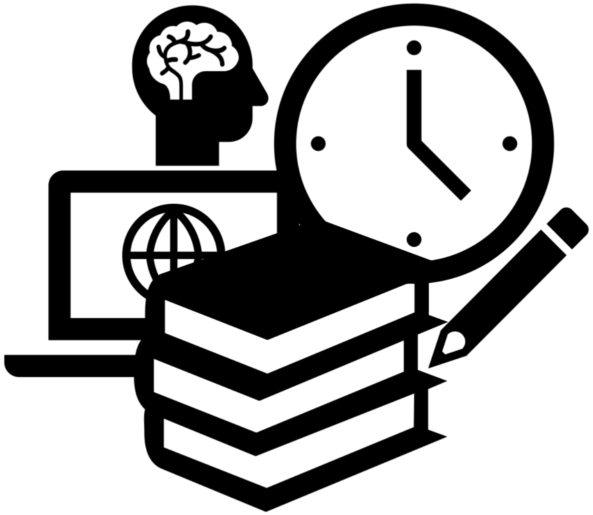
0	0	1	1
0	1	0	1
0	1	1	0
0	1	0	0

Inputs		Outputs	
a	b	Diff	B. Out
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- The borrow out signal is set when the subtractor needs to borrow from the next digit in a multi-digit subtraction.

Half Subtractor - Circuit

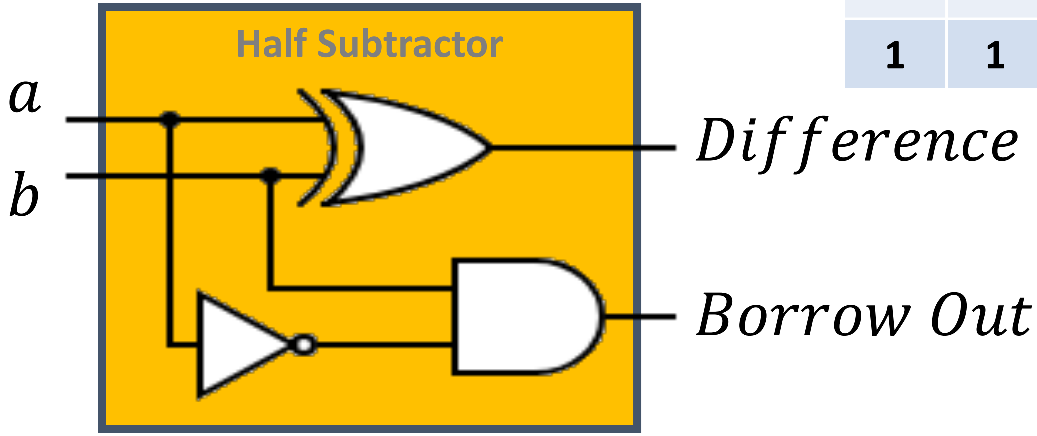
- Try it out by yourself..
- Compare the truth table and the circuit with Half Adder
- Can Half Adder be employed to construct Half Subtractor ?



Half Subtractor

- Circuit Diagram

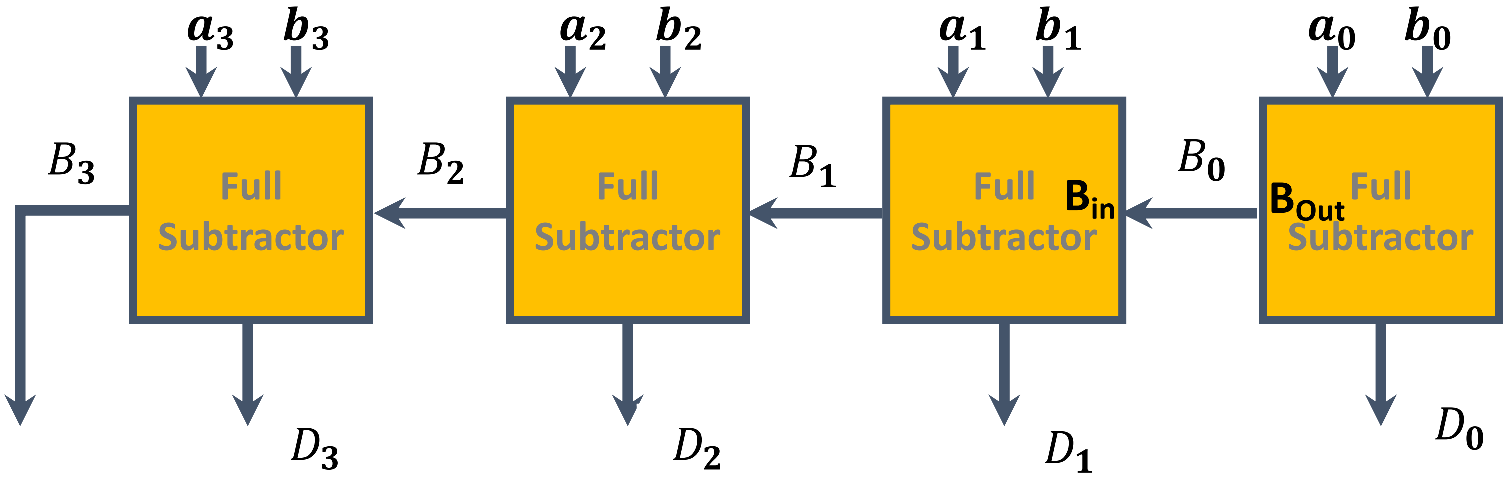
Inputs		Outputs	
a	b	Diff	B. Out
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Note: Half subtractor circuit implements $a - b$ and not $b - a$

Full Subtractor

a_3	a_2	a_1	a_0
b_3	b_2	b_1	b_0
D_3	D_2	D_1	D_0



- Four bit subtractor – series of full subtractors should be able to subtract two binary numbers with multiple bits.

Full Substractor

Inputs			Outputs	
$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{B_{in}}$	$\boldsymbol{Diff}$	$\boldsymbol{B_{out}}$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$\boldsymbol{B_{in}}$ is set when the previous digit is borrowed from $\boldsymbol{a}$

Full Substractor

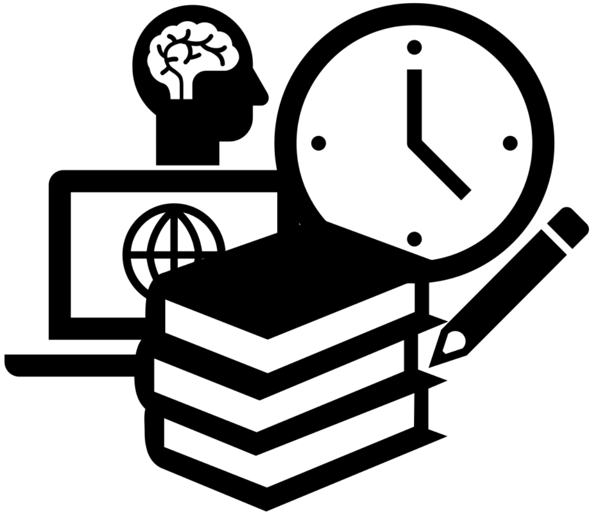
Inputs			Outputs	
a	b	B_{in}	$Diff$	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Example

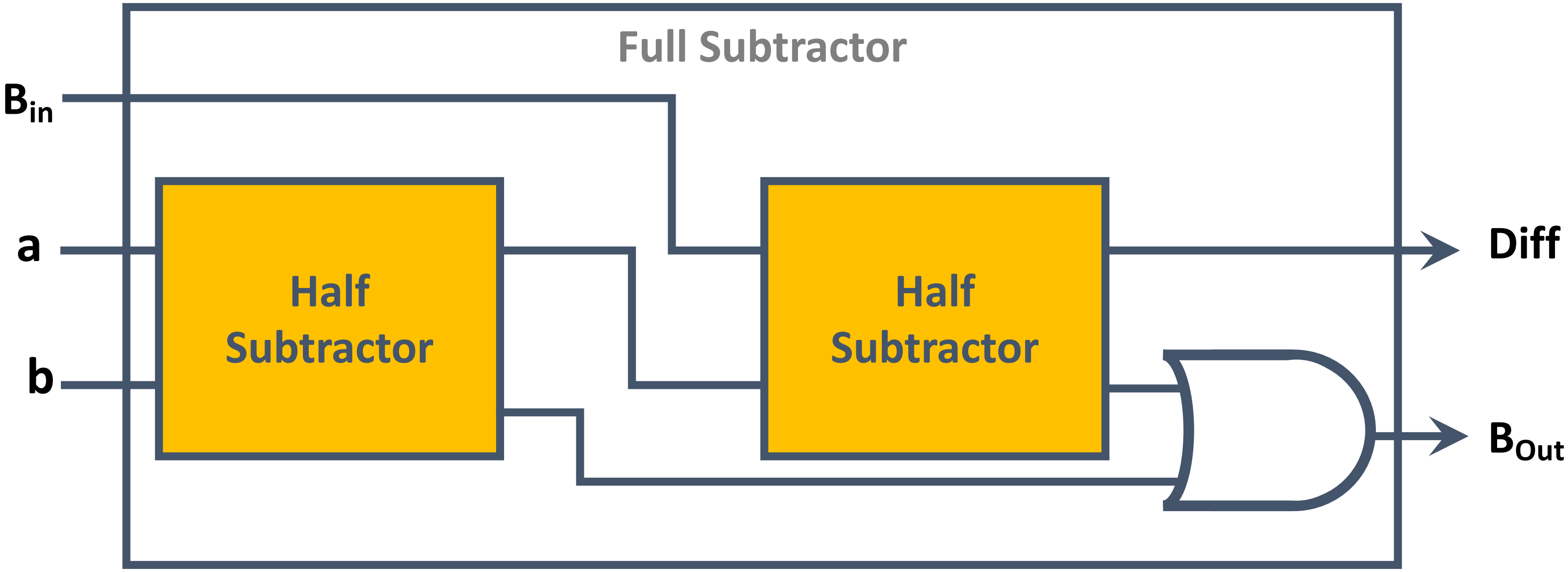
$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 1 \\ \hline \end{array}$$

Full Subtractor – Alternative Appr.

- Construct Full Subtractor from Half Subtractors



Full Subtractor – Alternative Appr.



External reference: https://en.wikipedia.org/wiki/Subtractor

Multipliers

- How do you design a two bit multiplier?
- How do you design a multiplier for multiple bits?

ALU

- ALU contains many circuit units to perform different operations.
  - Arithmatic: +,−, ×,÷
  - Logic: &, |, ~, <<, >>

Next…

Decoders & Multiplexers

Computer Systems

Kasun Gunawardana

E-mail: kgg

University of Colombo School of Computing

Decoders & Multiplexers

Decoders

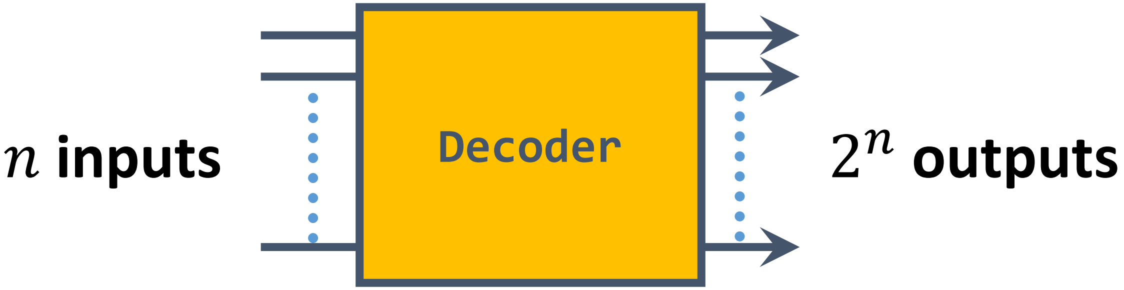
- Decoder is a special circuit unit to interpret a given code word.
- A binary code space of n bits can have 2^n number of codes.

1	2	3	.....	n

- Decoder can interpret maximum of 2^n of codes by activating one of its 2^n number of output channels, given a n bit input code word.
- Each **`minterm`** activates a distinct output channel.

Decoders (Cont.)

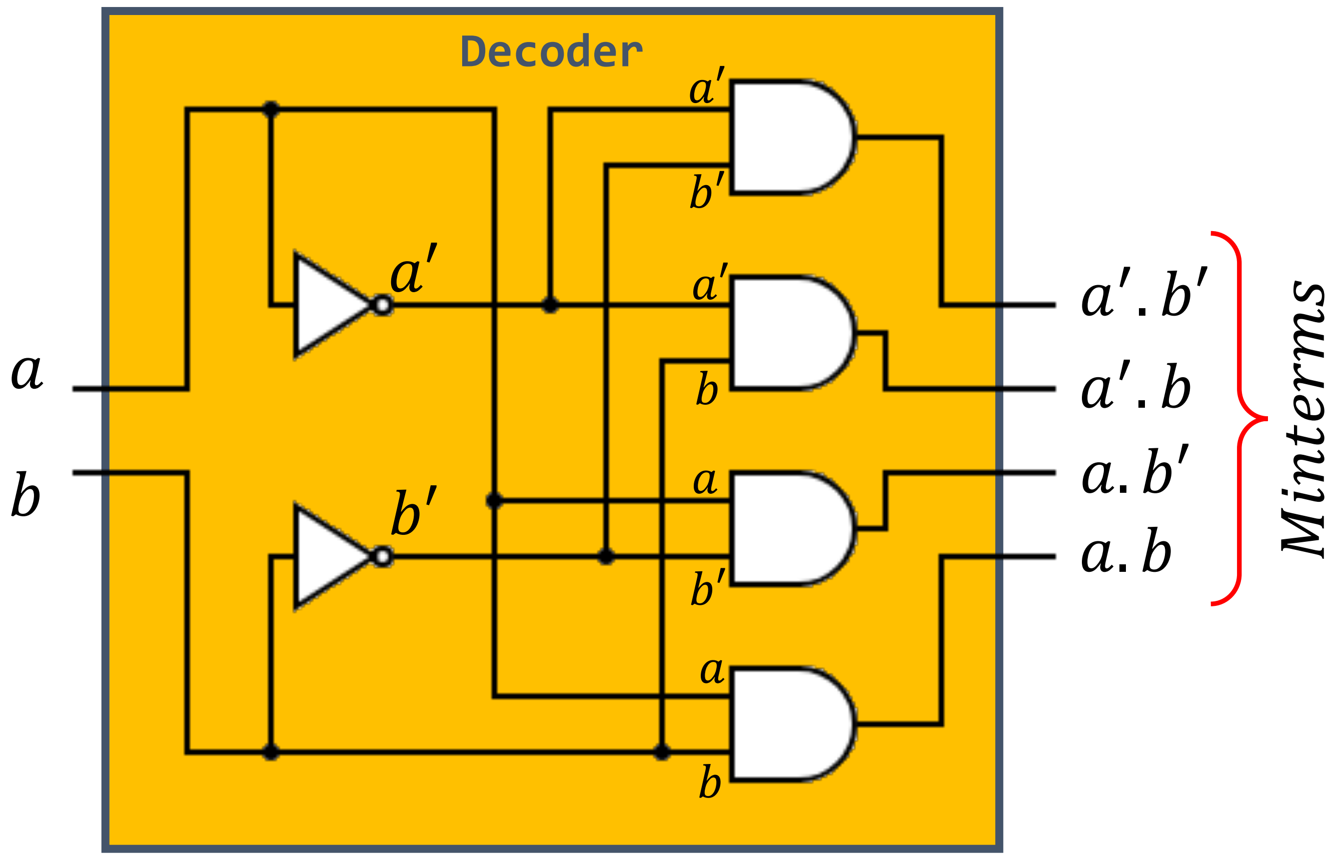
- INPUTS: n bits
- OUTPUTS: 2^n bits



- Given an input code, only one output channel must be activated.

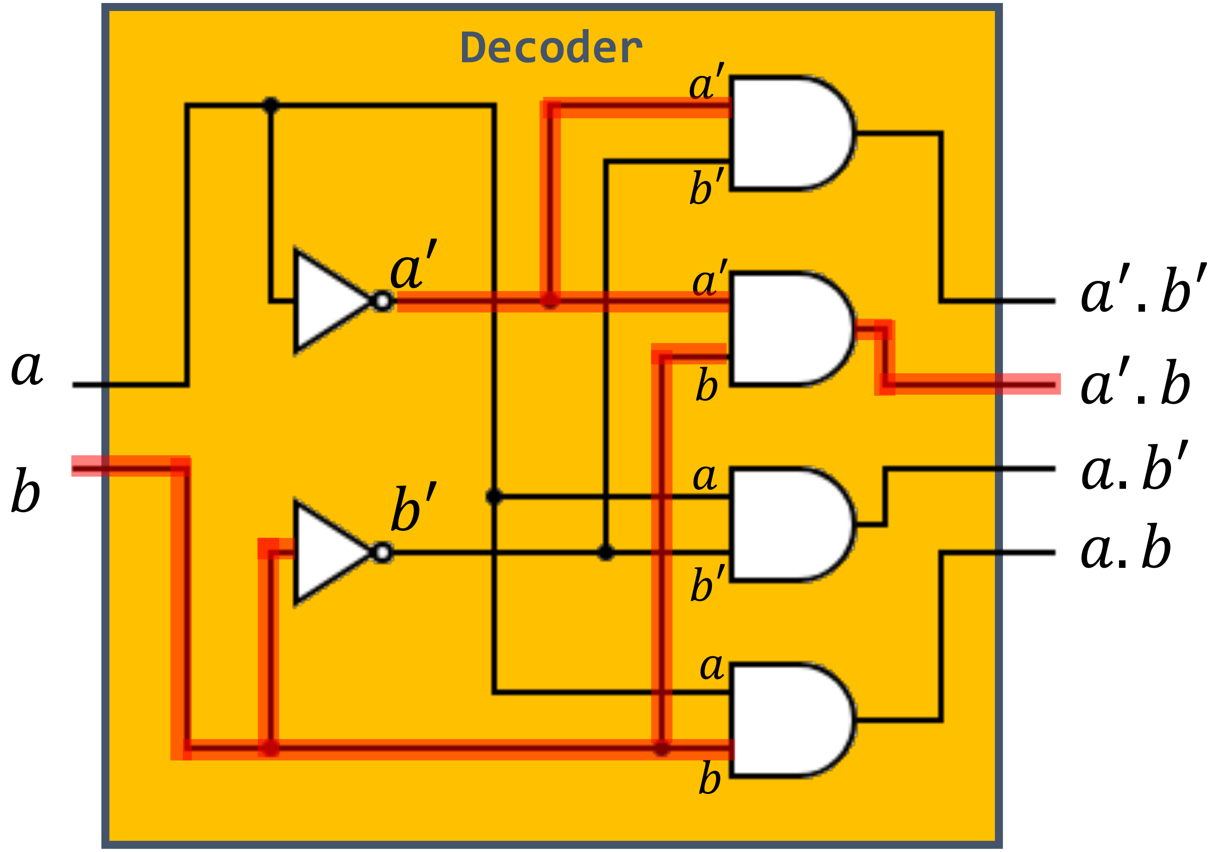
- Note: Decoders can have less than 2^n outputs.

2×4 Decoder



2×4 Decoder

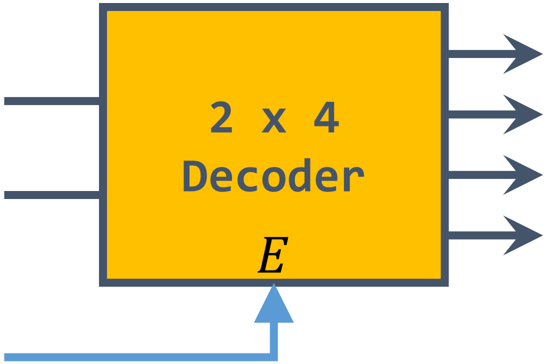
$a = 0$
$b = 1$



Boolean state 1 *channels are highliged in RED*

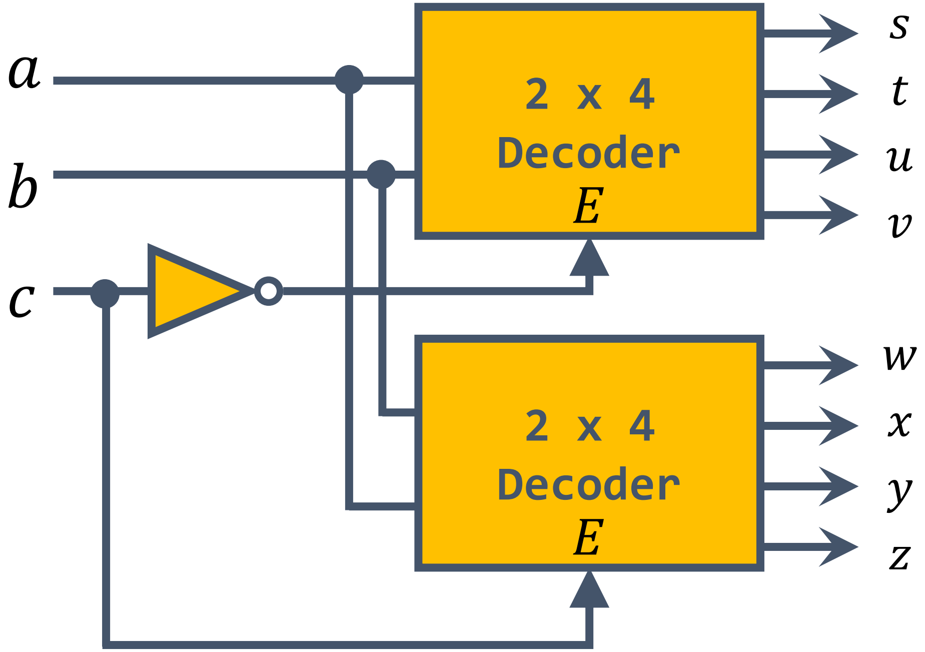
Decoder with Enable Input

- A single controller E input to enable/ disable
  - If $E = 1$ then the decoder will operate normal
  - If $E = 0$ then the decoder outpus will be inactive



Exercise..

- Derive the truth table for the following circuit and discuss the equlent single circuit unit that you can use.



Answer

- It's your turn…

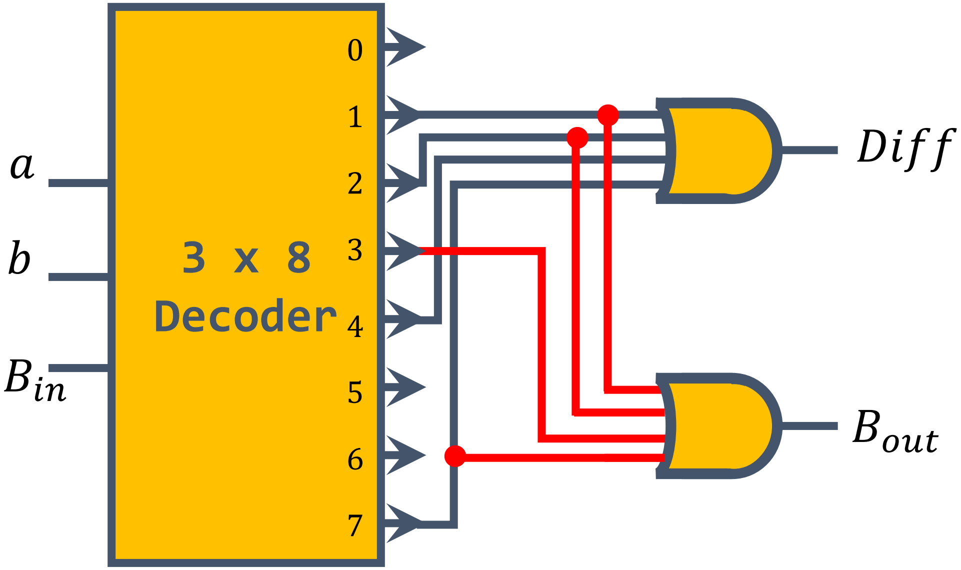
Decoder as Circuit Builder

- Any Boolean function can be represented in Sum of Minterms form.
- A decoder with **n** inputs can can generate all the Minterms for **n** input variables.
- Logical Sum can be provided with an external **OR** gate.

Full Subtractor with Decoder

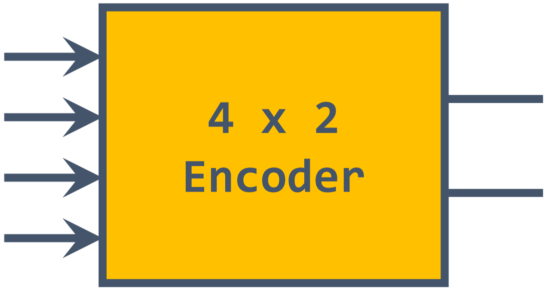
Inputs			Outputs	
a	b	B_{in}	Diff	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$Diff = \sum(1, 2, 4, 7) \qquad B_{out} = \sum(1, 2, 3, 7)$$



Encoders

- Encoder perfomrs the inverse operation of a decoder.
- Encoder can have 2^n of inputs and maximum of n output channels.

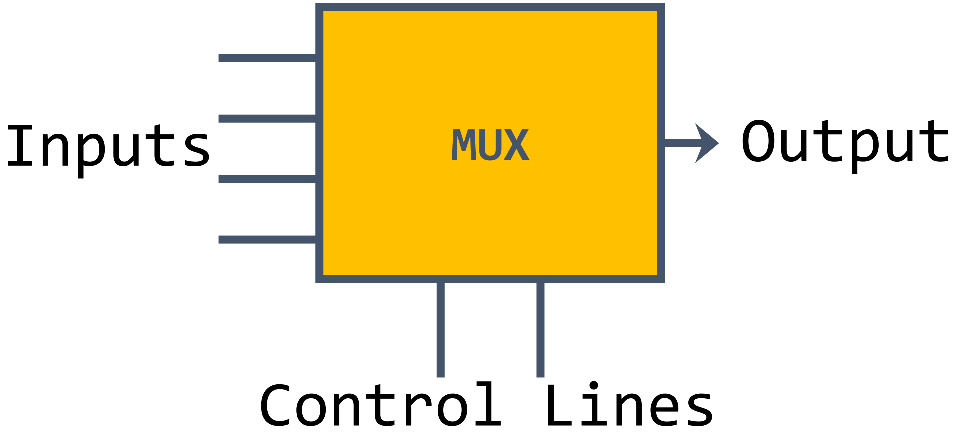


Exercise..

- Design a circuit for 8 x 3 Encoder
  - Search..!
  - Read..!
  - Explore..!

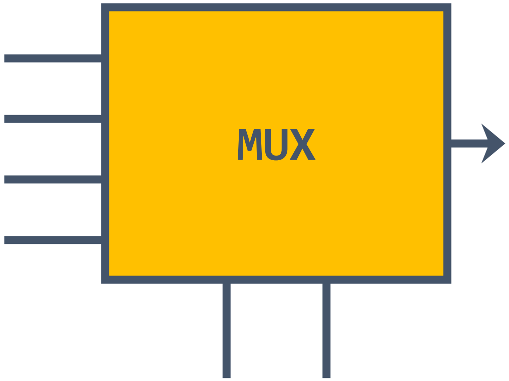
Multiplexers

- One of many input channels is directed to the single output chanel.
  - Several input channels
  - Single output channel
  - Several control lines to direct an input channel to the output channel



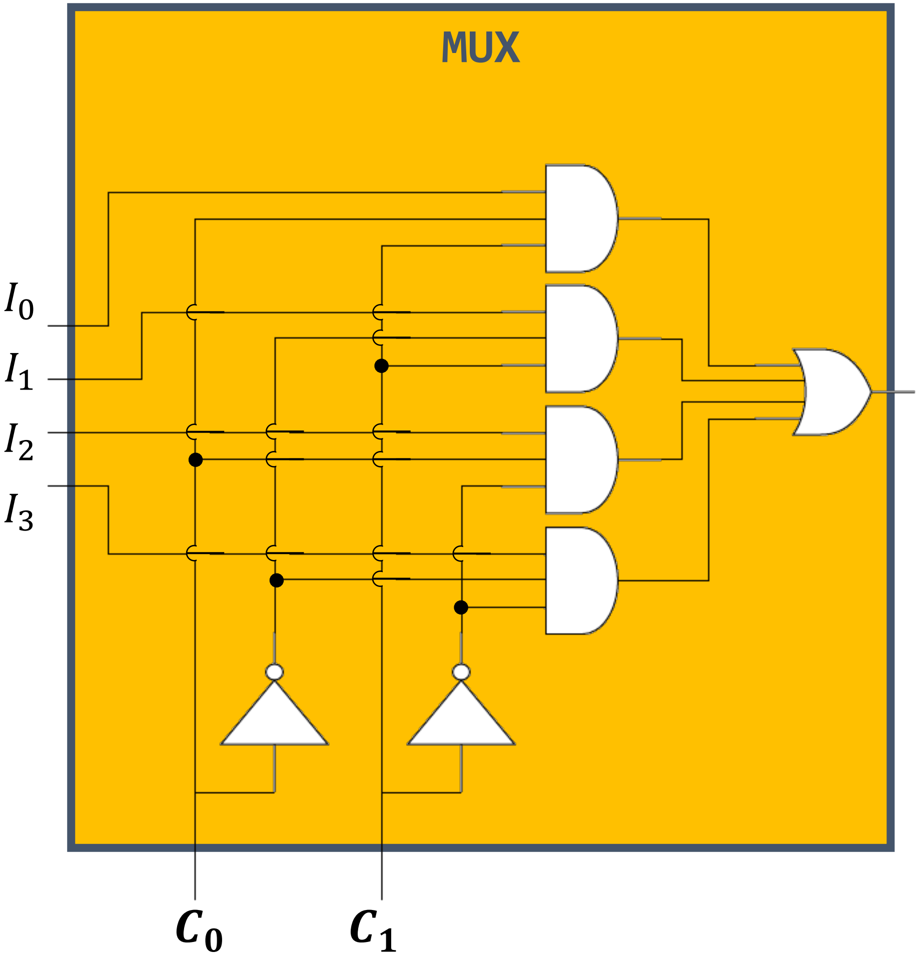
Multiplexer

- Depends on the value set on control lines, only one input will be directed to the output.
- Dedicated control line pattern as the pass code for each input.
- 4 patterns from 2 control lines.
- 4 inputs can be directed using 2 control lines.
- Thus, n control lines for 2^n inputs.



Multiplexer Circuit

Input	Control Code		Input Activation
	C_0	C_1	
I_0	0	0	$I_0 . C_0' . C_1'$
I_1	0	1	$I_1 . C_0' . C_1$
I_2	1	0	$I_2 . C_0 . C_1'$
I_3	1	1	$I_3 . C_0 . C_1$



Reading Exercise

- Read about Demultiplexers.

Exercise..

- What are the uses of the following circuit units in a computer system?
  - Decoder
  - Multiplexer

Thank You..!