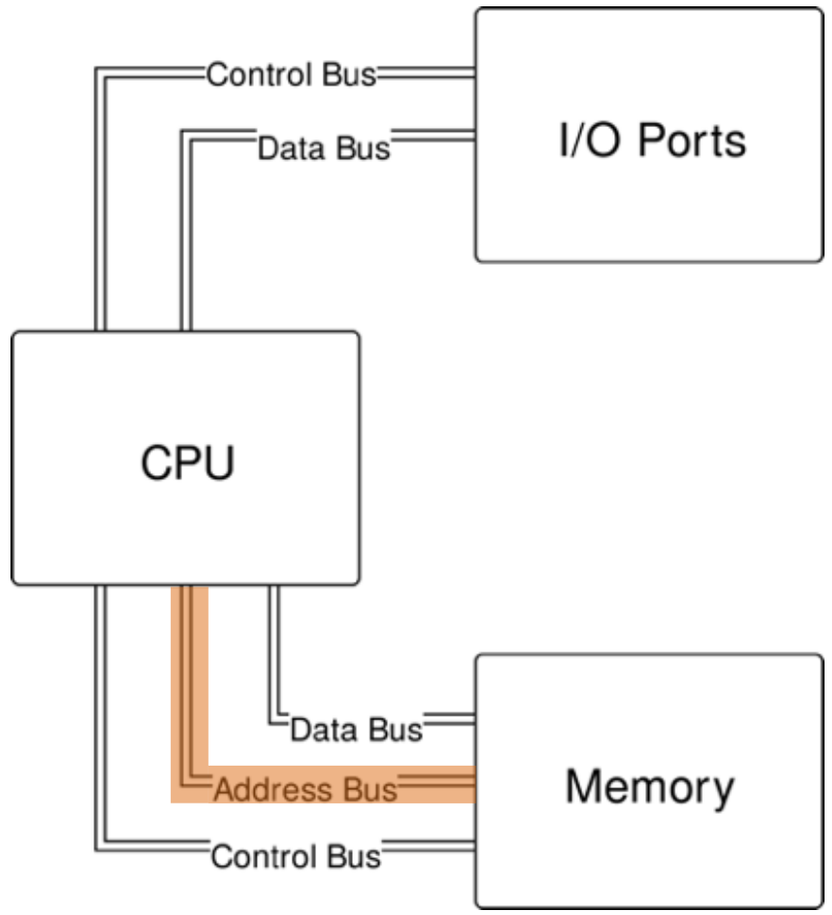
Combinational Logic Circuits

ENCE260: Computer Architecture Topic 7

Recap

- Each byte in memory has its own address, which is transmitted on the address bus.

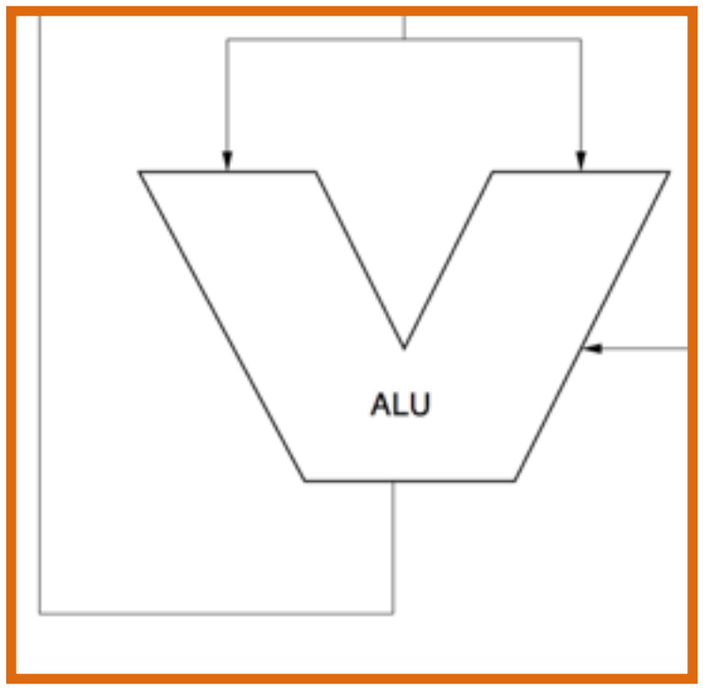


	Byte 0								
Address	**Bit 0**	**Bit 1**	**Bit 2**	**Bit 3**	**Bit 4**	**Bit 5**	**Bit 6**	**Bit 7**	
0000									**Word 0**
0001									**Word 1**
N									**Word N**

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Recap

- The ALU performs arithmetic operations and logic comparisons, but only sends the result of one operation at a time to its output.



Ciaran Moore
ciaran.moore@canterbury.ac.nz

Challenge

- What sort of hardware do we need to translate the values on the address bus to a single word-specific activation signal?
- How can we select which ALU operation's result is sent to the ALU output?

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Combinational Logic Circuits

ENCE260: Computer Architecture Topic 7

Selected diagrams from Tocci, Widmer & Moss: Digital Systems

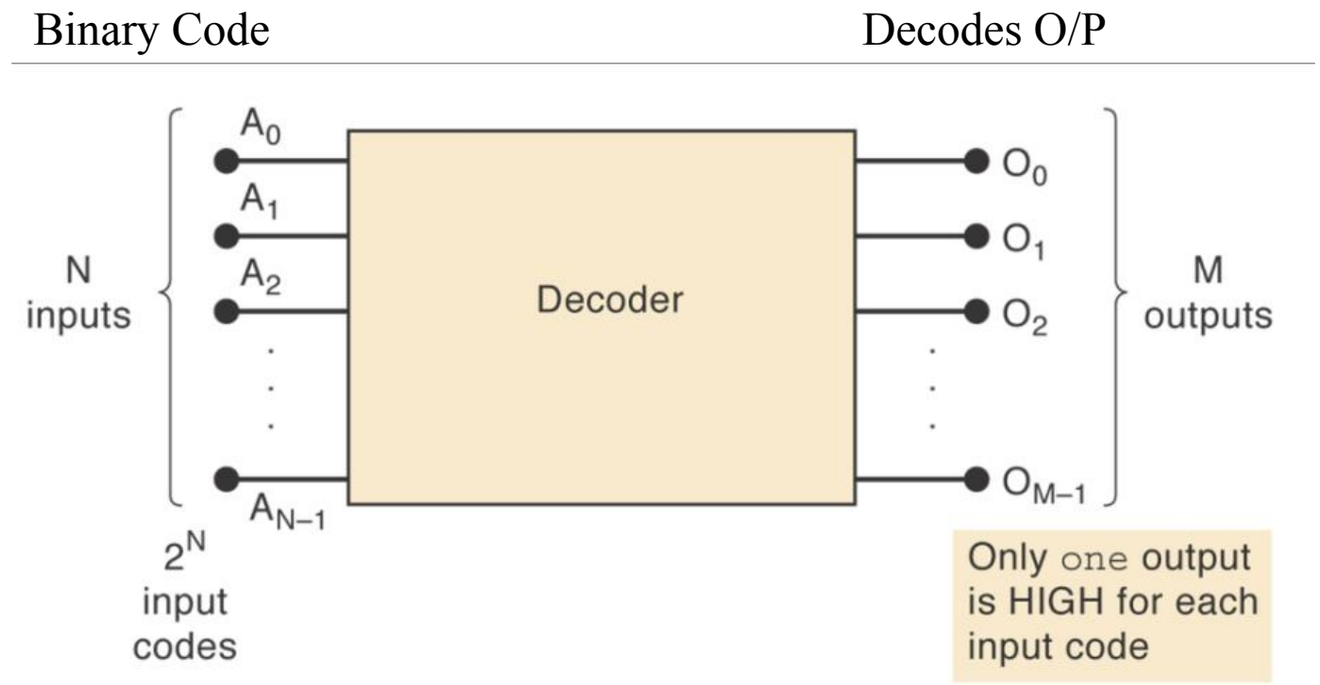
Common Combinational Logic Circuits

- Decoders & Encoders
- Multiplexers & Demultiplexers
- Multibit Adders
- Multifunction Logic

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Decoder

- Typically used for *selecting* a memory location given an address:
- N address bits → 2^N locations
- We want a separate output to be high for each possible combination of inputs.



Ciaran Moore
ciaran.moore@canterbury.ac.nz

Decoder

- 2-to-4 Decoder

Code Input A_1	A_0	D_0	D_1	D_2	D_3	Decoded Output
0	0	1	0	0	0	
0	1	0	1	0	0	
1	0	0	0	1	0	
1	1	0	0	0	1	

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Decoder

- 2-to-4 Decoder

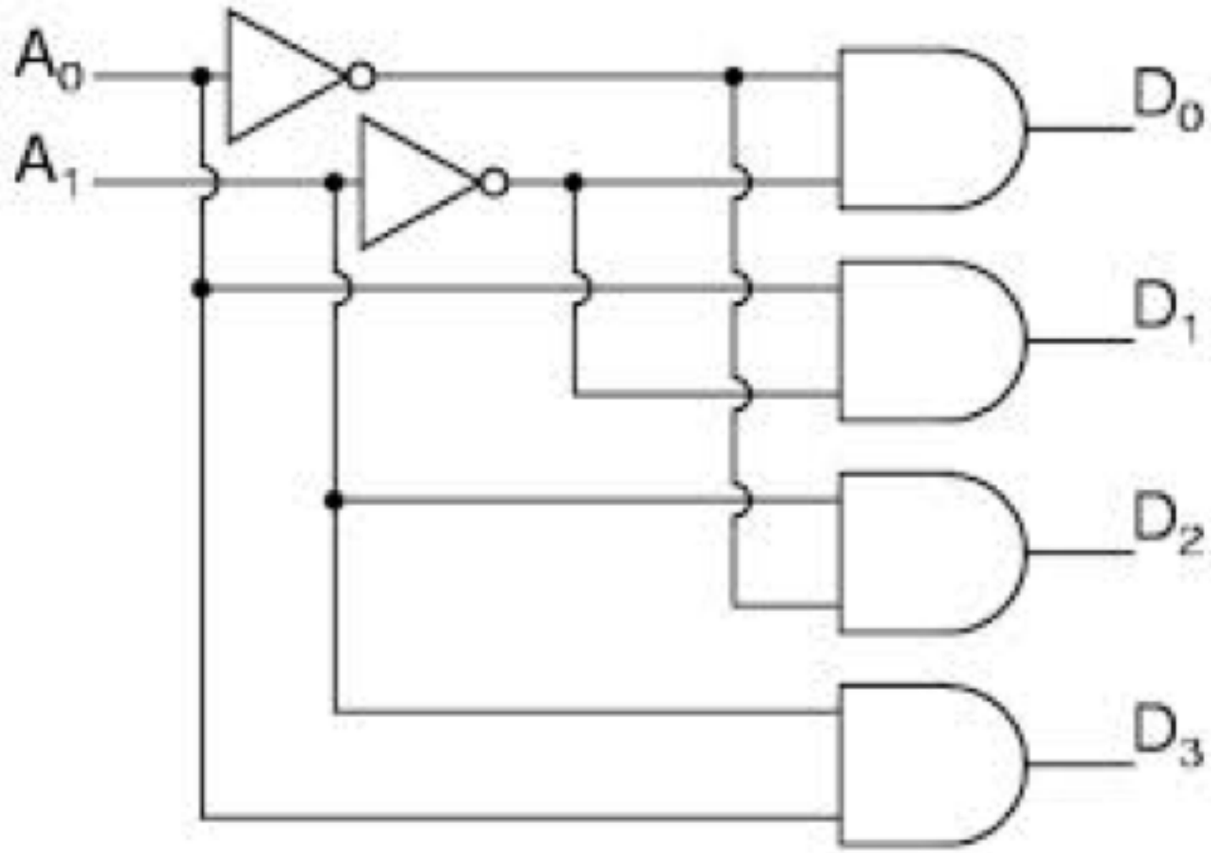
Code Input A_1	A_0	D_0	D_1	D_2	D_3	Decoded Output
0	0	1	0	0	0	
0	1	0	1	0	0	
1	0	0	0	1	0	
1	1	0	0	0	1	

$$D_0 = \overline{A_1}\ \overline{A_0} \qquad D_2 = A_1\ \overline{A_0}$$
$$D_1 = \overline{A_1}\ A_0 \qquad D_3 = A_1\ A_0$$

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Decoder

- 2-to-4 Decoder



$$D_0 = \overline{A_1}\ \overline{A_0}$$

$$D_1 = \overline{A_1}\ A_0$$

$$D_2 = A_1\ \overline{A_0}$$

$$D_3 = A_1\ A_0$$

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Decoder

- 2-to-4 Decoder

TEXAS INSTRUMENTS

SN74LVC1G139

SCES602E – AUGUST 2004 – REVISED JANUARY 2018

SN74LVC1G139 2-to-4 Line Decoder

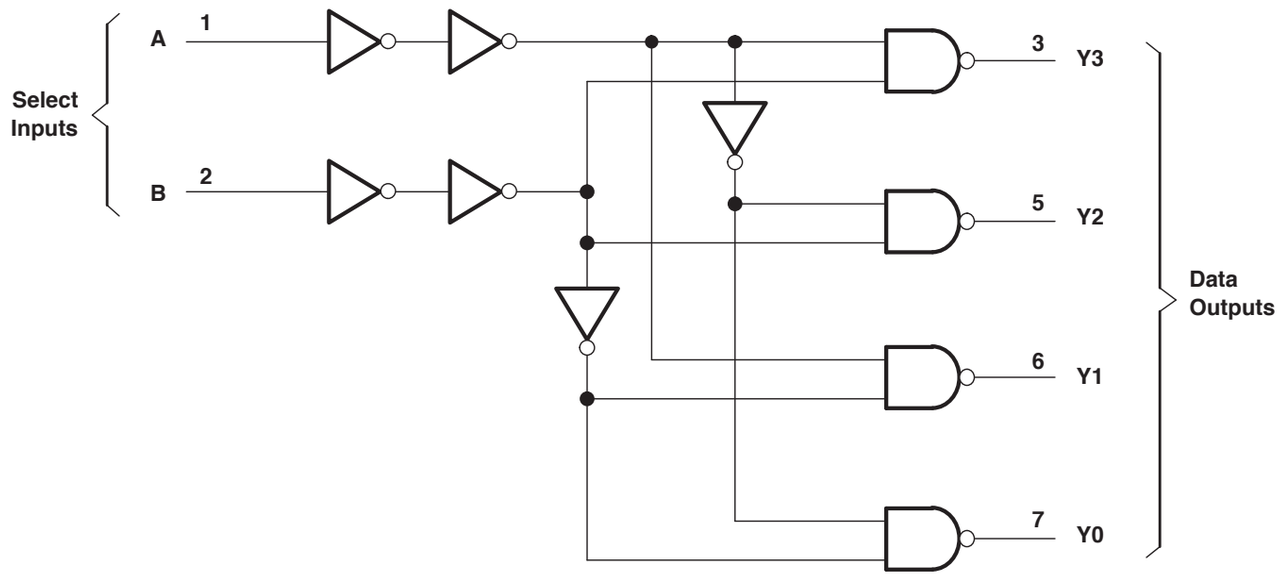
1 Features

- Available in the Texas Instruments NanoStar™ and NanoFree™ Packages
- Supports 5-V V_{CC} Operation

3 Description

This SN74LVC1G139 2-to-4 line decoder is designed for 1.65-V to 5.5-V V_{CC} operation.

The SN74LVC1G139 2-line to 4-line decoder is



Ciaran Moore
ciaran.moore@canterbury.ac.nz

Decoder

- 2-to-4 Decoder

TEXAS INSTRUMENTS

SN74LVC1G139

SCES602E – AUGUST 2004 – REVISED JANUARY 2018

SN74LVC1G139 2-to-4 Line Decoder

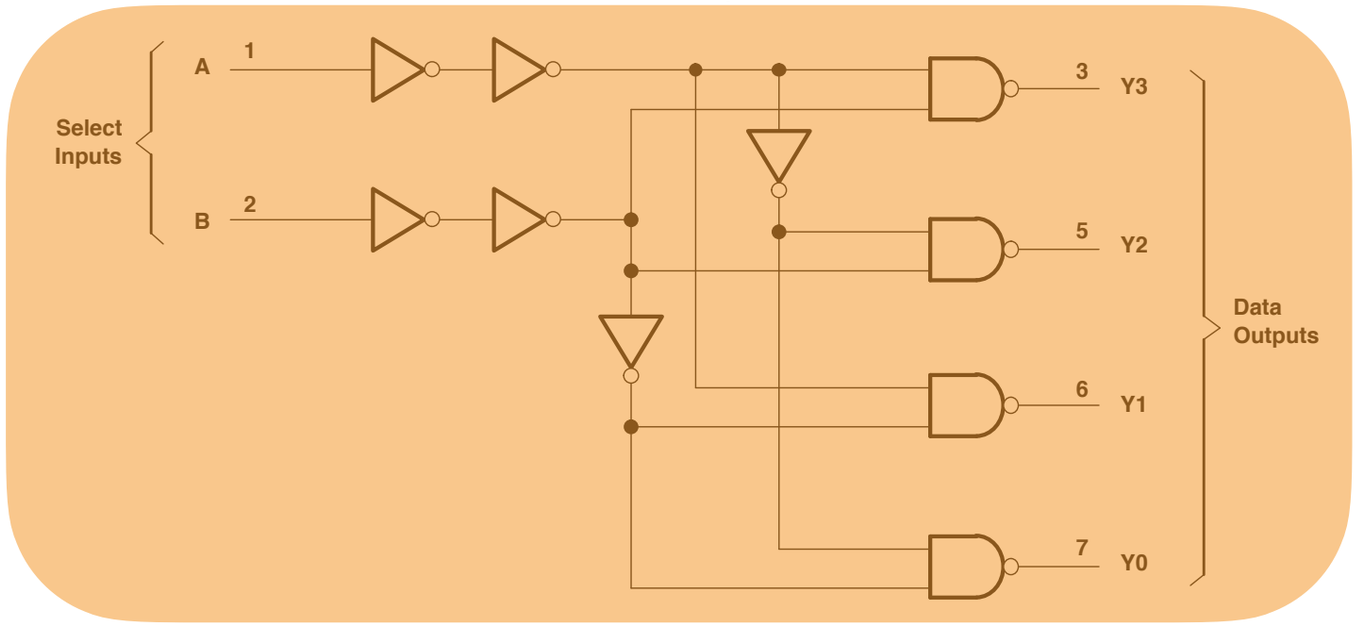
1 Features

- Available in the Texas Instruments NanoStar™ and NanoFree™ Packages
- Supports 5-V V_{CC} Operation

3 Description

This SN74LVC1G139 2-to-4 line decoder is designed for 1.65-V to 5.5-V V_{CC} operation.

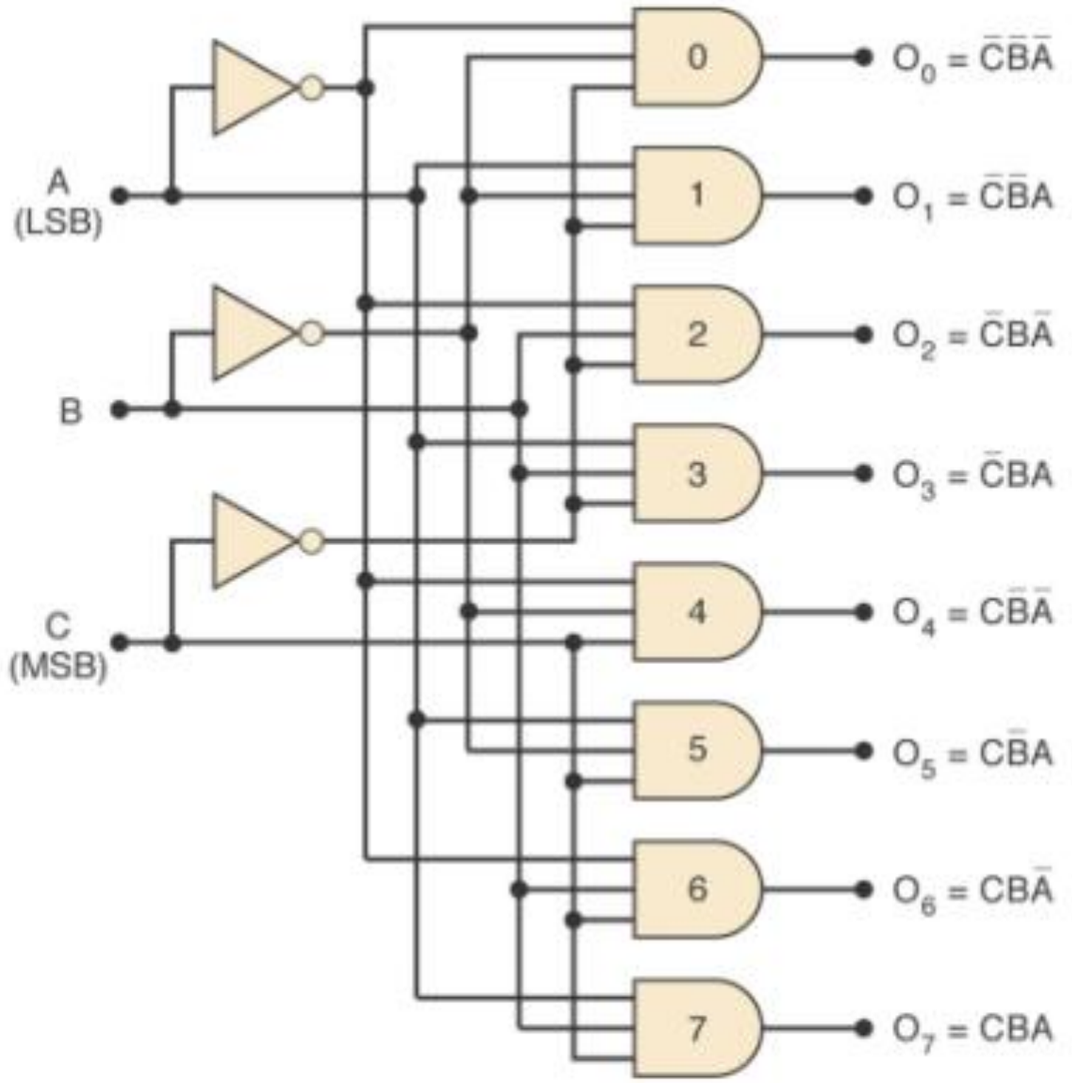
The SN74LVC1G139 2-line to 4-line decoder is



Ciaran Moore
ciaran.moore@canterbury.ac.nz

Decoder

- 3-to-8 Decoder



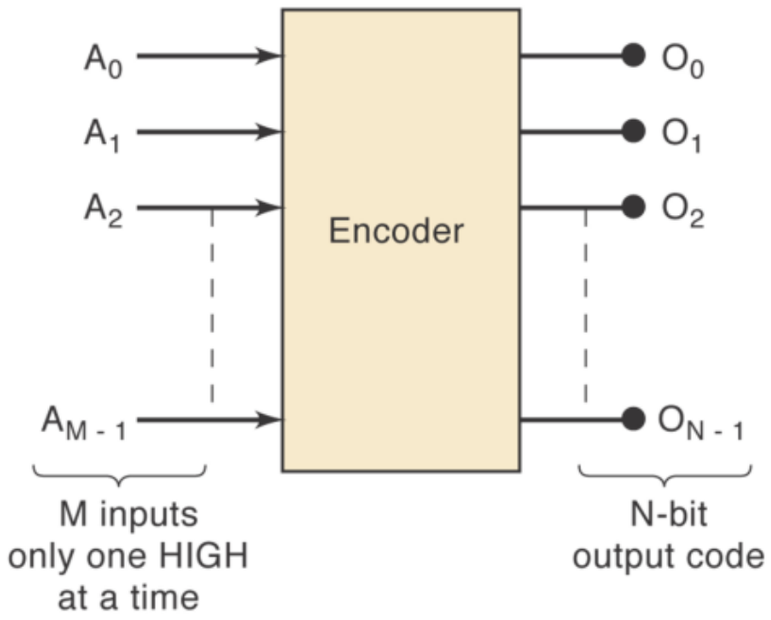
Ciaran Moore
ciaran.moore@canterbury.ac.nz

Encoder

Raw Input

D_0	D_1	D_2	D_3	A_1	A_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Encoded Output

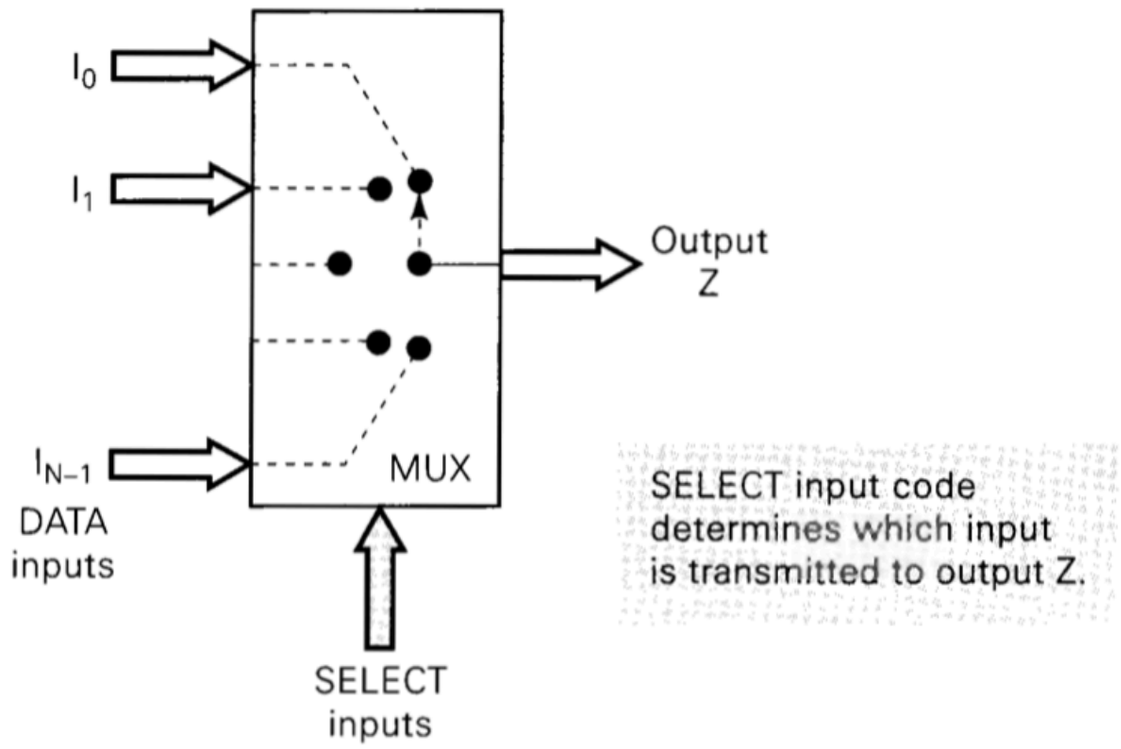


Tutorial: what would the logic circuit look like for this device?

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Multiplexer

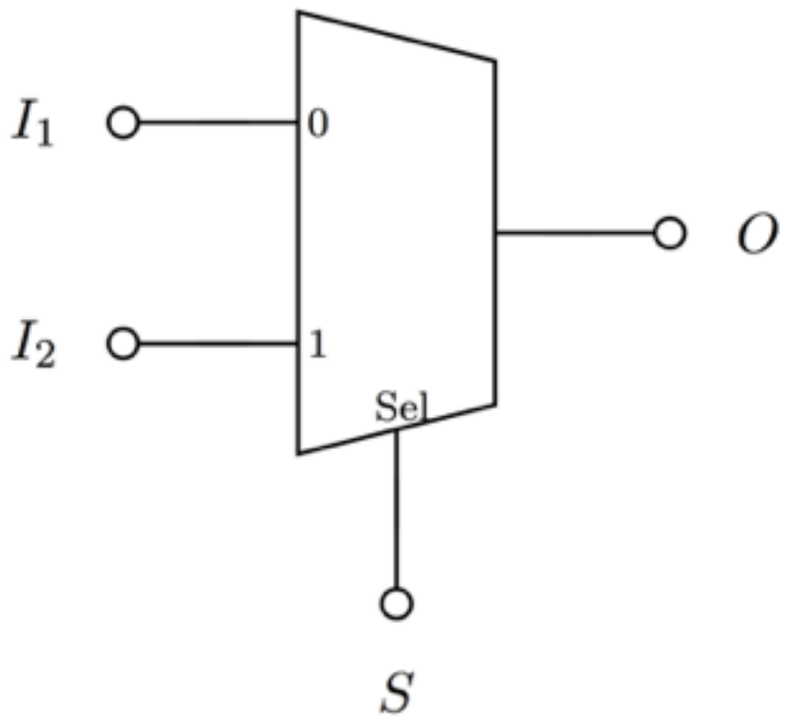
- Multiplexers (muxes) select one or more of several inputs and send them to the output.
- N-to-1 muxes are most common, but can also get N-to-N/2 or N-to-N/4 muxes.



Ciaran Moore
ciaran.moore@canterbury.ac.nz

Multiplexer

- 2:1 Multiplexer

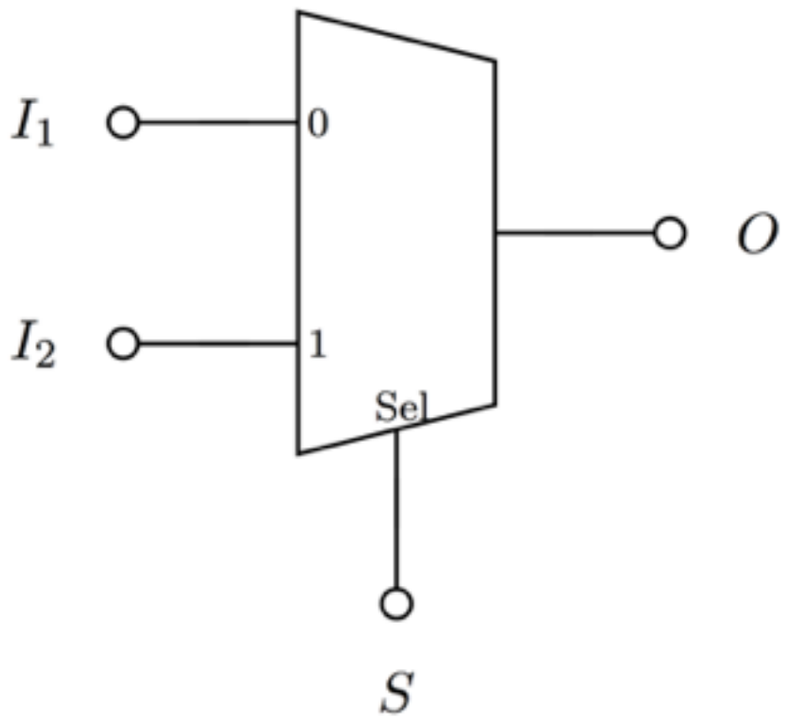


Input			Output
S	I_1	I_2	O
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Multiplexer

- 2:1 Multiplexer



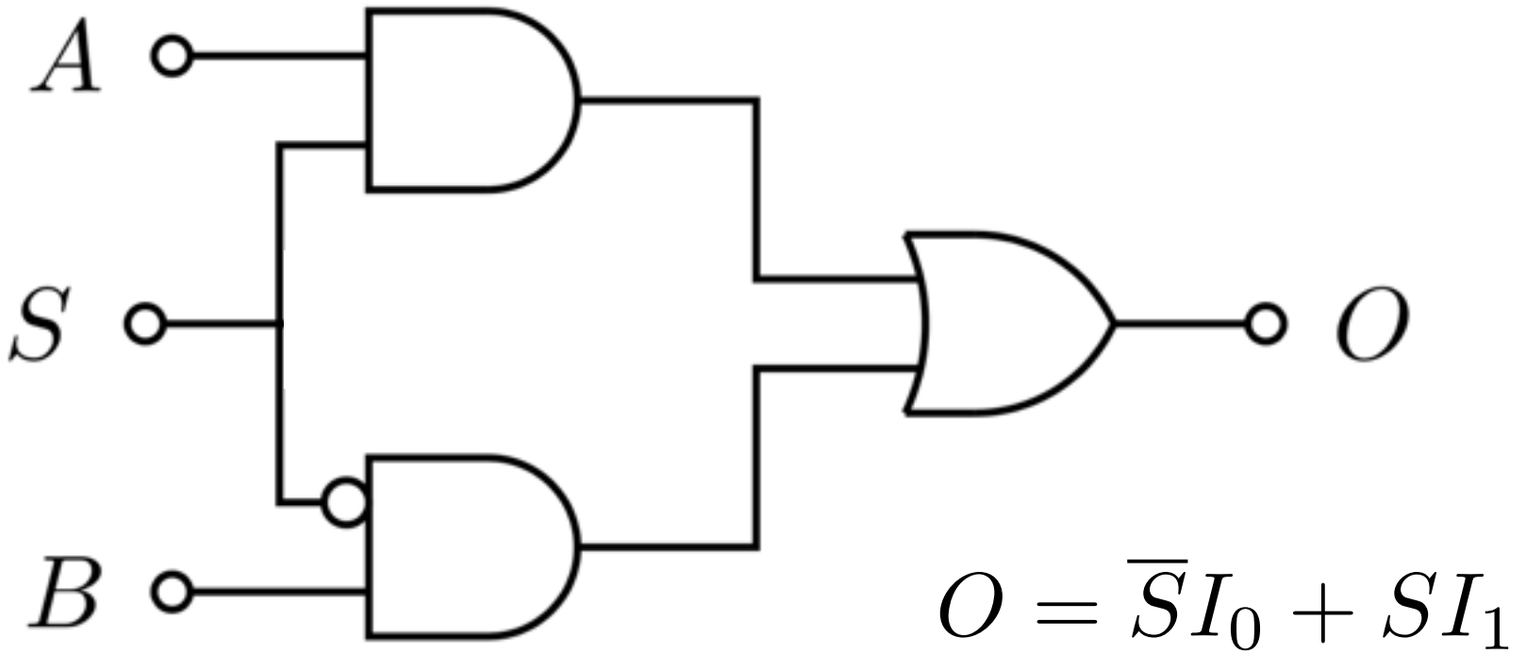
Input S	Output O
0	I_0
1	I_1

$$O = \overline{S} I_0 + S I_1$$

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Multiplexer

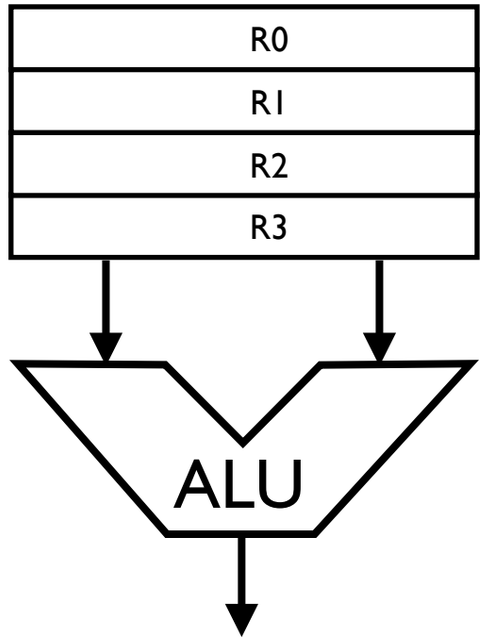
- 2:1 Multiplexer



Ciaran Moore
ciaran.moore@canterbury.ac.nz

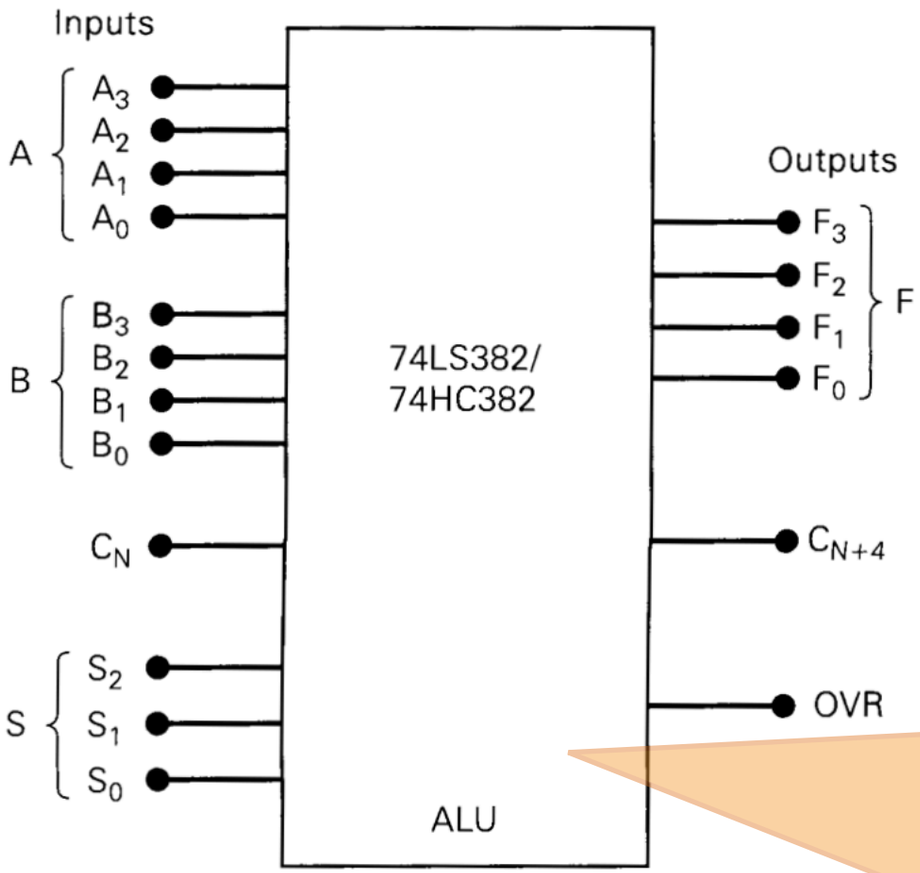
Example: ALU

- ALUs can perform arithmetic calculations as well as logical comparisons.
- In practice, they do both a calculation and a comparison on every pair of inputs they are given.
- A mux is used to determine whether the arithmetic or logical result is sent to the output.



Ciaran Moore
ciaran.moore@canterbury.ac.nz

Example: ALU

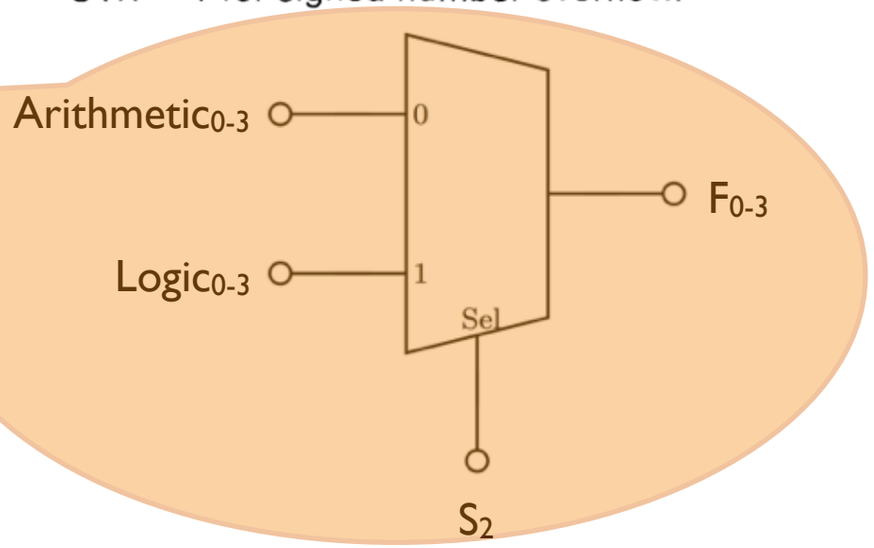


A = 4-bit input number
B = 4-bit input number
C_N = carry into LSB position
S = 3-bit operation select inputs
F = 4-bit output number
C_{N+4} = carry out of MSB position
OVR = overflow indicator

Function Table

S_2	S_1	S_0	Operation	Comments
0	0	0	CLEAR	$F_3F_2F_1F_0 = 0000$
0	0	1	B minus A	Needs $C_N = 1$
0	1	0	A minus B	
0	1	1	A plus B	Needs $C_N = 0$
1	0	0	A ⊕ B	Exclusive-OR
1	0	1	A + B	OR
1	1	0	AB	AND
1	1	1	PRESET	$F_3F_2F_1F_0 = 1111$

Notes: S inputs select operation.
OVR = 1 for signed-number overflow.



Ciaran Moore
ciaran.moore@canterbury.ac.nz

Multiplexer

- Tutorial:
  - Carry out the design process (truth table, logic expression and circuit diagram) for a 4:1 multiplexer.
  - Combine two 2:1 multiplexers to make a 4:2 multiplexer.

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Demultiplexer

- Conceptually similar to a mux, but with swapped inputs and outputs.

1:4 Demux

Inputs			Outputs			
I	S_1	S_2	Y_0	Y_1	Y_2	Y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	X	X	0	0	0	0

image: https://learn.digilentinc.com/Documents/299

Ciaran Moore
ciaran.moore@canterbury.ac.nz

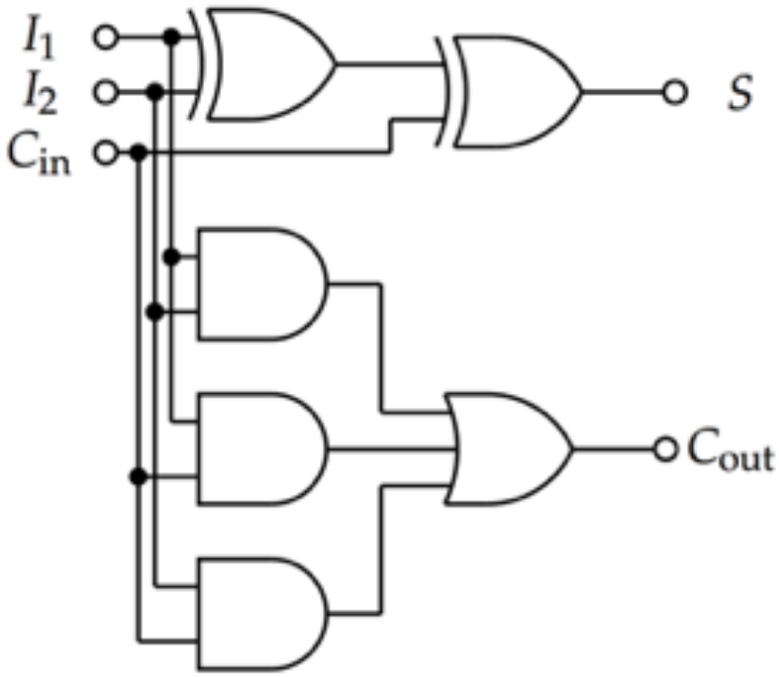
Adder

- We have already seen a full adder circuit that adds two 1-bit numbers, with carry in and carry out.
- We have also seen how to do binary addition on 4-bit 2's complement numbers (Tute 2).
- ALUs combine N 1-bit full adders, where N is the word length of the processor.

Ciaran Moore
ciaran.moore@canterbury.ac.nz

1-Bit Adder

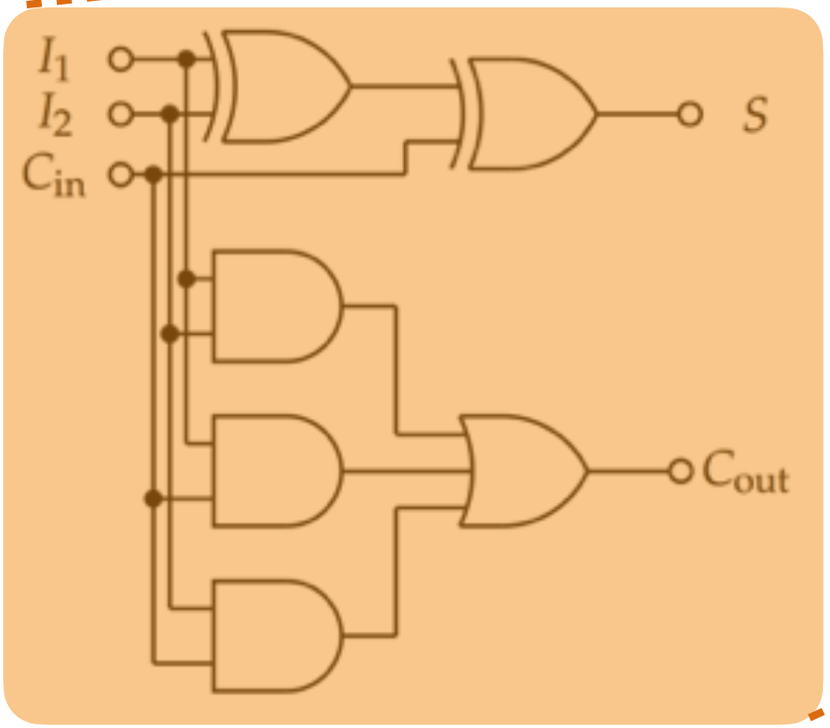
Full Adder

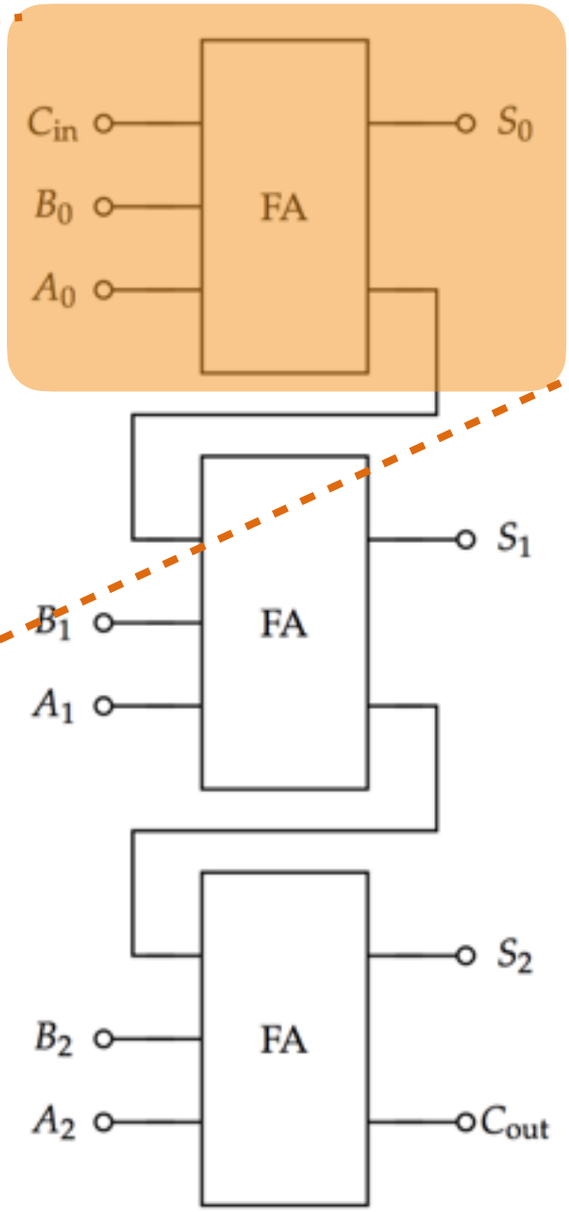


Ciaran Moore
ciaran.moore@canterbury.ac.nz

3-Bit Adder

Full Adder



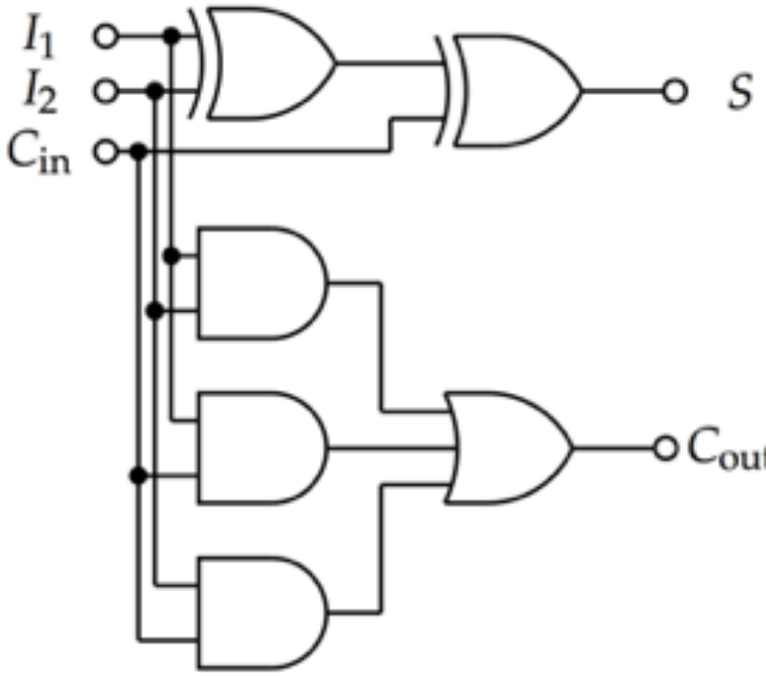


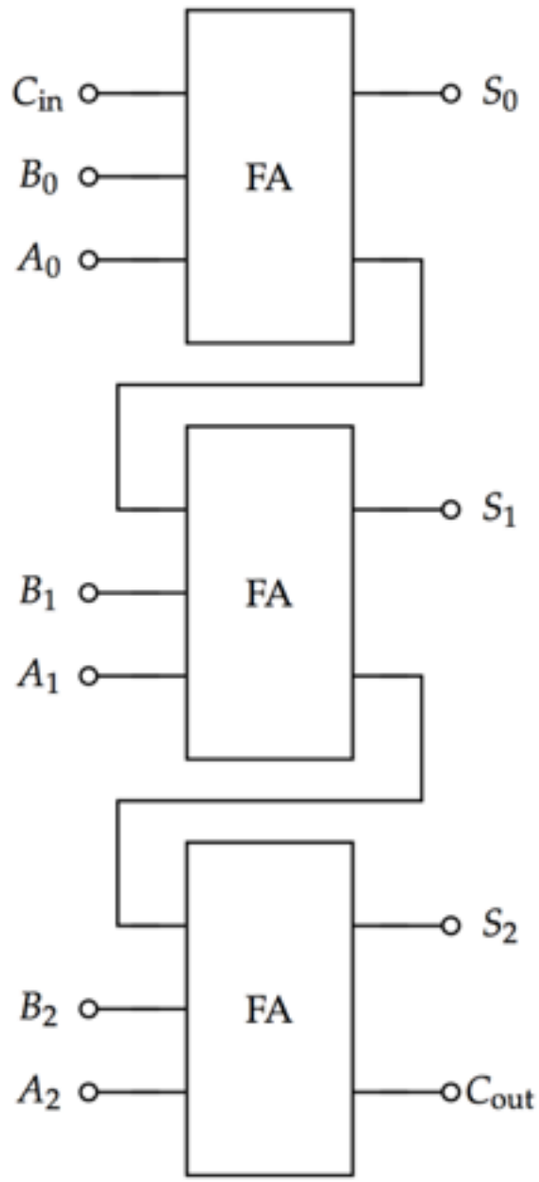
3-Bit Ripple Carry Adder

Ciaran Moore
ciaran.moore@canterbury.ac.nz

3-Bit Adder

Full Adder



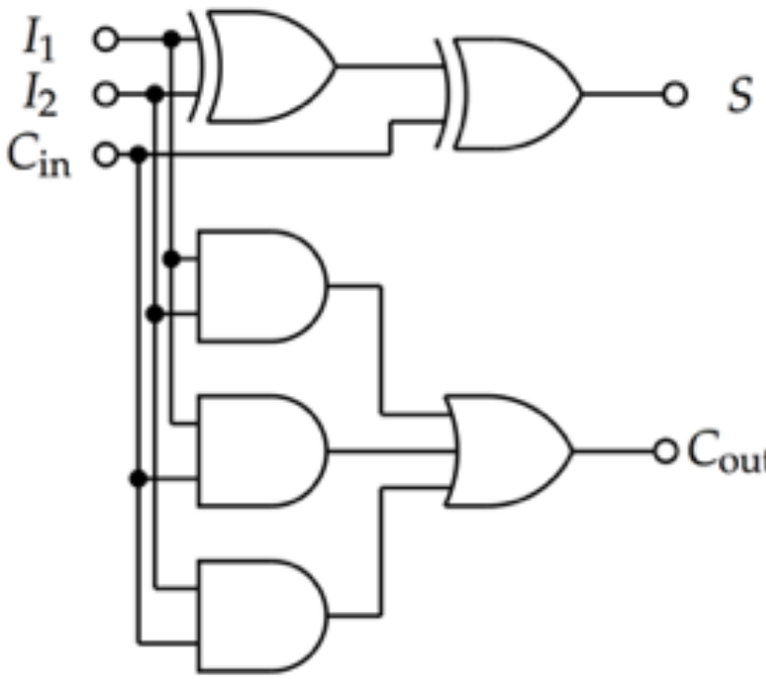


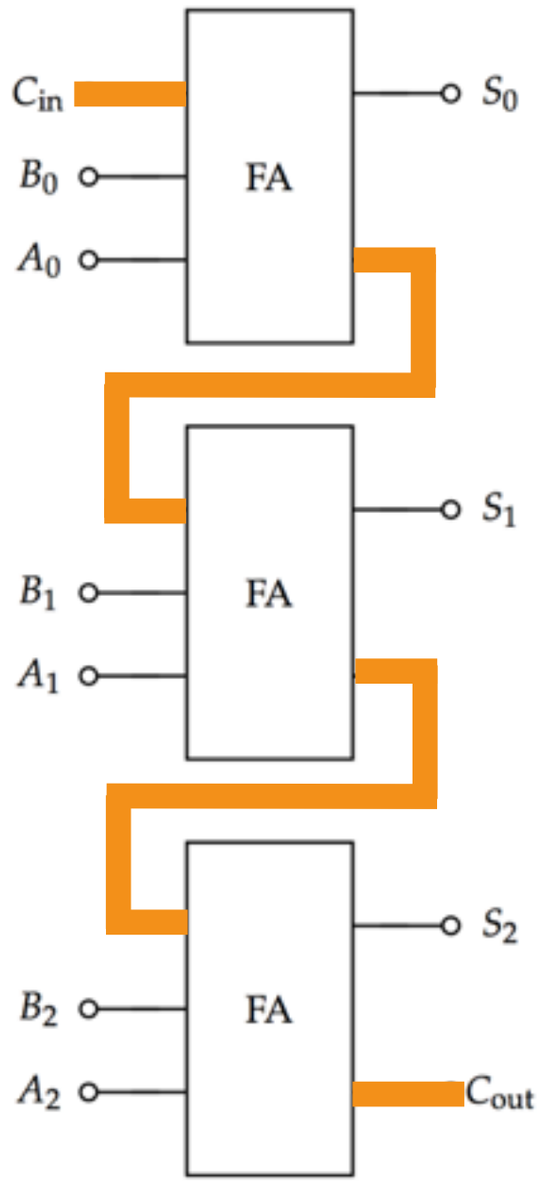
3-Bit Ripple Carry Adder

Ciaran Moore
ciaran.moore@canterbury.ac.nz

3-Bit Adder

Full Adder





3-Bit Ripple Carry Adder

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Subtraction

- We *could* design a circuit to perform binary subtraction, e.g. 3 – 4 = ?
- Instead, we use the properties of 2's complement encoding and our adder circuitry, since 3 – 4 = 3 + (-4).

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Subtraction

minuend $3 - 4 = -1$ difference

subtrahend

- To perform a subtraction, convert the subtrahend to a 2's complement negative number:

	0	1	0	0	= +4
	1	0	1	1	1's complement
+	0	0	0	1	add 1
	1	1	0	0	= -4

- Then add the result to the minuend:

	0	0	1	1	
+	1	1	0	0	
	1	1	1	1	= -1

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Subtraction

- So how do we implement a 2's complement negative conversion using digital logic?
- 1's complement:

Inputs		Output
Sub?	*In*	*Out*
0	0	0
0	1	1
1	0	1
1	1	0

Ciaran Moore
ciaran.moore@canterbury.ac.nz

Subtraction

- So how do we implement a 2's complement negative conversion using digital logic?
- 1's complement:

$$Out = Sub \oplus In$$

Inputs		Output
Sub?	*In*	*Out*
0	0	0
0	1	1
1	0	1
1	1	0

Ciaran Moore
ciaran.moore@canterbury.ac.nz

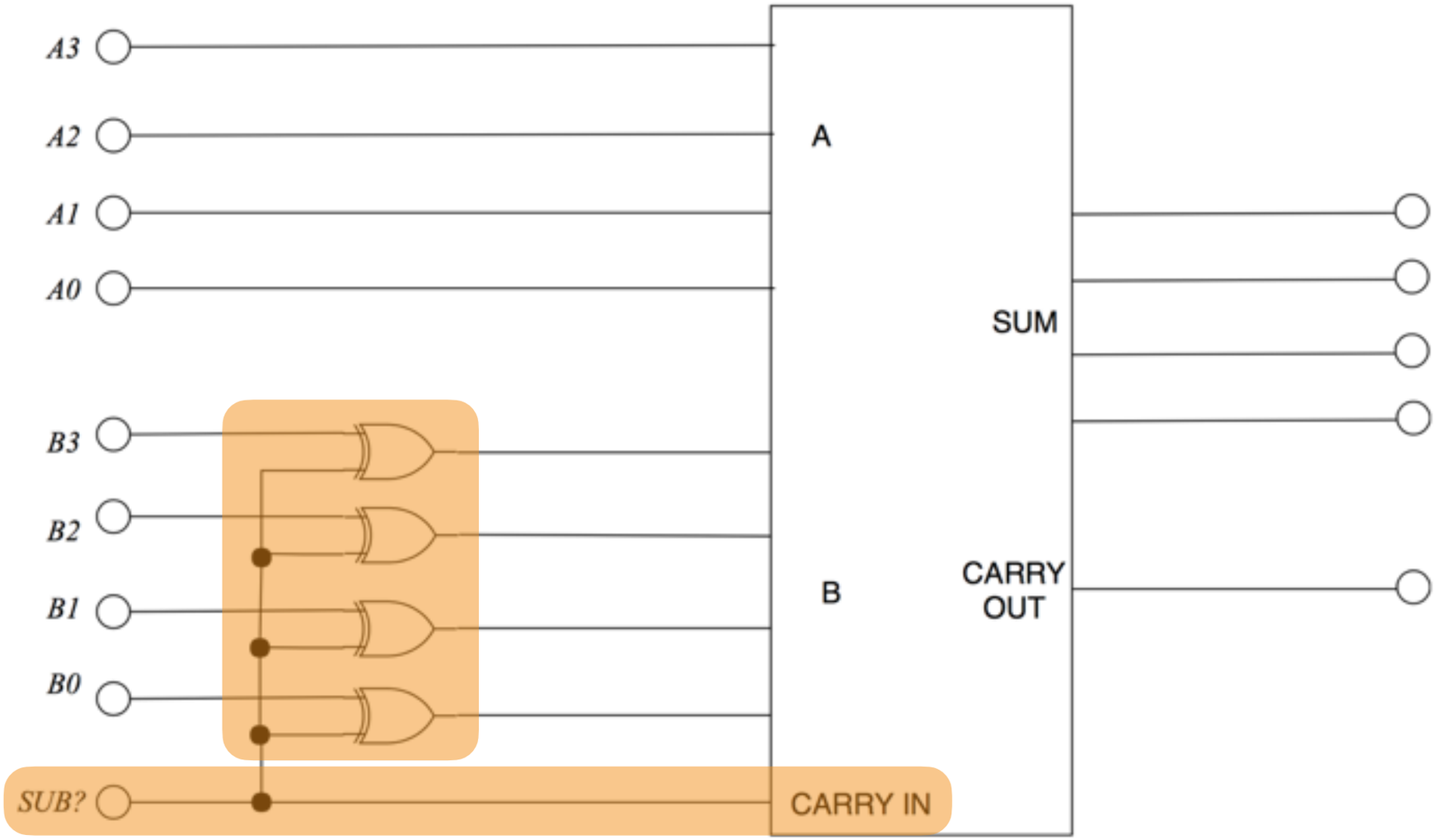
Subtraction

- So how do we implement a 2's complement negative conversion using digital logic?
- 1's complement:
- Add 1: $C_{in} = 1$.

Inputs		Output
Sub?	*In*	*Out*
0	0	0
0	1	1
1	0	1
1	1	0

Ciaran Moore
ciaran.moore@canterbury.ac.nz

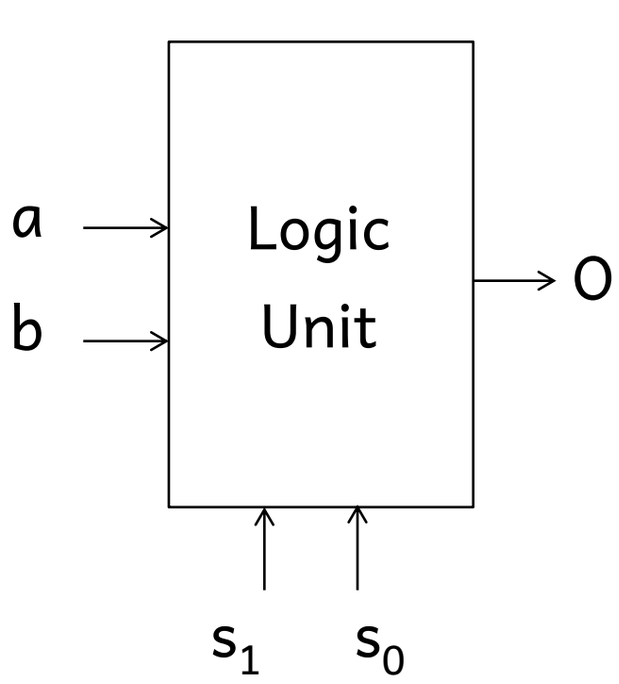
Subtraction



Ciaran Moore
ciaran.moore@canterbury.ac.nz

Multifunction Logic

- When is an AND gate not an AND gate?



s_1	0	1	1
s_0	1	0	1
Inputs		Output O	
a	b	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Ciaran Moore
ciaran.moore@canterbury.ac.nz

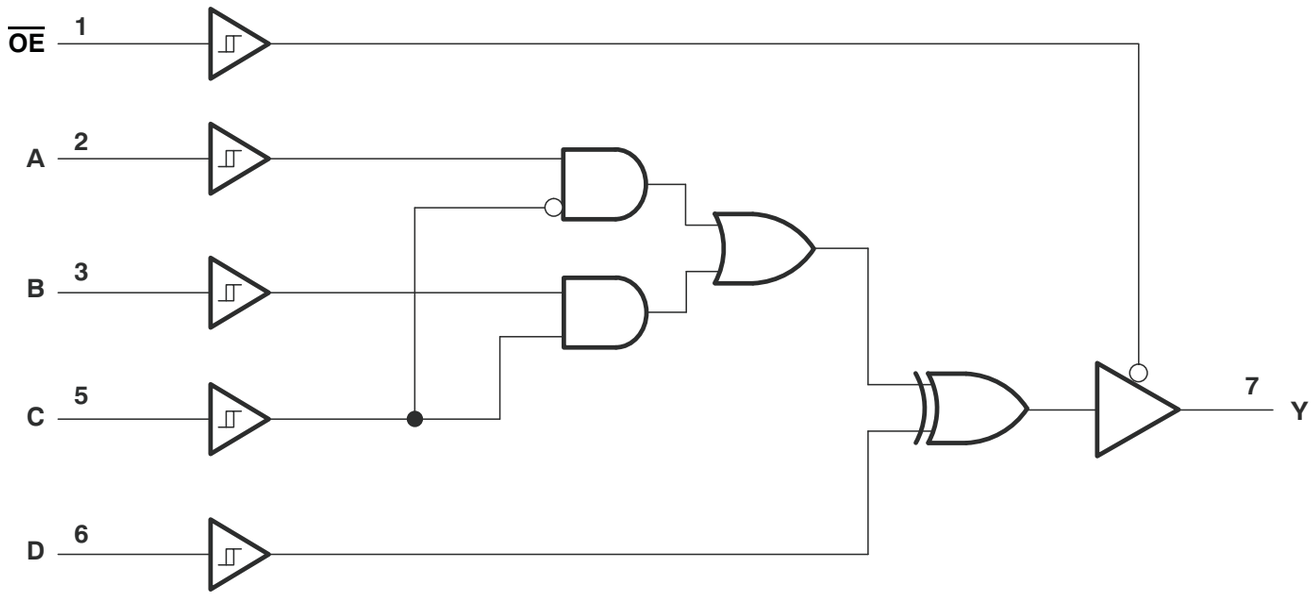
Multifunction Logic

TEXAS INSTRUMENTS

SN74LVC1G99

Ultra-Configurable Multiple-Function Gate With 3-State Output

Check for Samples: SN74LVC1G99



NO. OF INPUTS	AND/NAND FUNCTION	OR/NOR FUNCTION	$\overline{OE}$	A	B	C	D
2	3-state AND	3-state NOR, both inputs inverted	L	L	Input 1	Input 2	L

Summary

- Combinational devices like muxes, encoders and adders are like high-level constructs in a programming language: they allow us to implement sophisticated functionality with minimal effort.

Ciaran Moore
ciaran.moore@canterbury.ac.nz