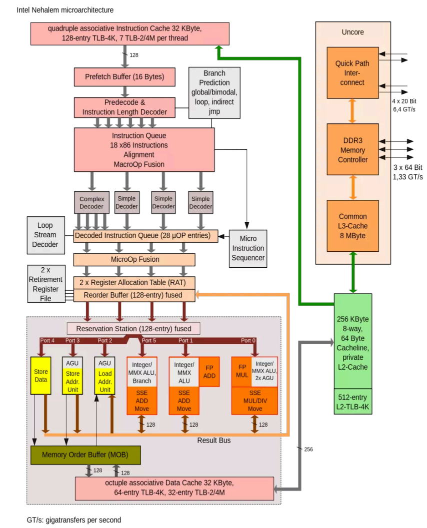
CSC 411

Computer Organization (Spring 2022)
Lecture 15: Basics of logic design (Appendix)

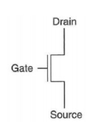
Prof. Marco Alvarez, University of Rhode Island

Goal of Chapter 4

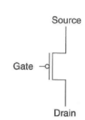


Transistors

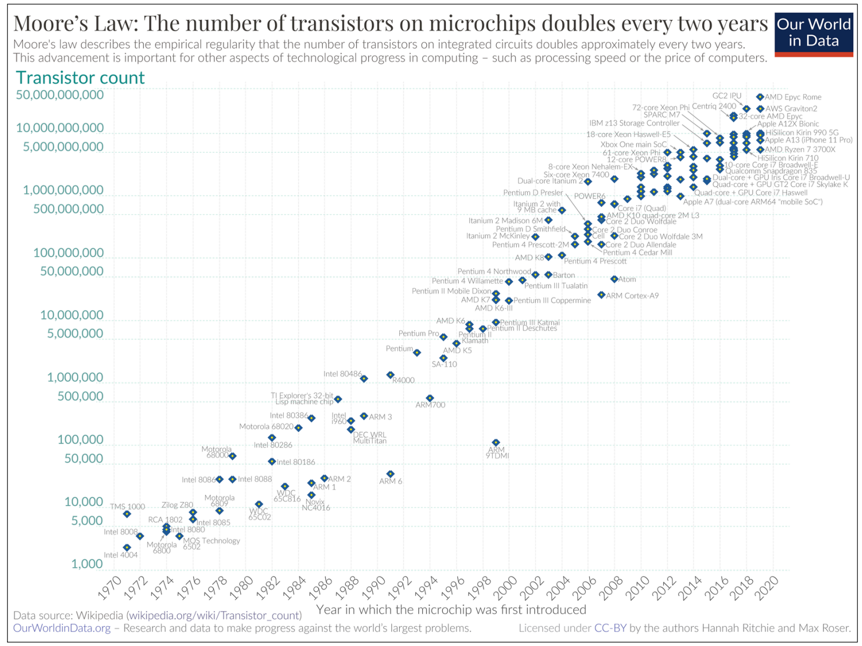
- Building block of computers
- ICs contain billions
- Transistors act as switches
  - can be combined to implement simple logic gates
  - AND OR NOT
- Two types of MOS transistors
  - metal-oxide semiconductors



n-type



p-type



NOT Gate

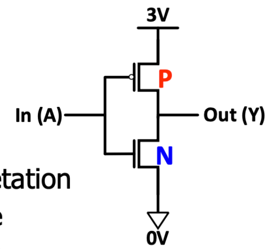
This is actually the CMOS NOT Gate

Why do we call it NOT?

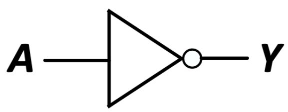
- If A = 0V then Y = 3V
- If A = 3V then Y = 0V

Digital circuit: one possible interpretation

- Interpret 0V as logical (binary) 0 value
- Interpret 3V as logical (binary) 1 value



$$Y = \bar{A}$$



We call it a NOT gate
or an inverter

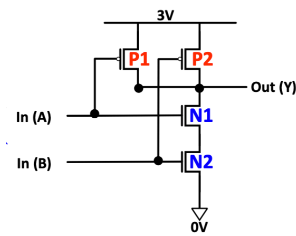
Truth table: shows what is the logical output of the circuit for each possible input

A	Y
0	1
1	0

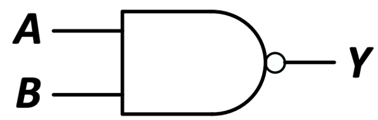
https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=onur-digitaldesign_comparch-2021-lecture4-combinational-logic-afterlecture.pdf

NAND Gate

Let's build more complex gates!

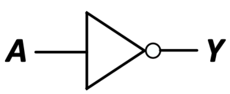


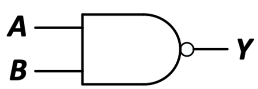
$$Y = \overline{A \cdot B} = \overline{AB}$$

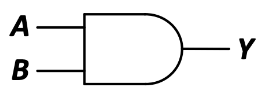


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=onur-digitaldesign_comparch-2021-lecture4-combinational-logic-afterlecture.pdf



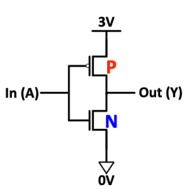


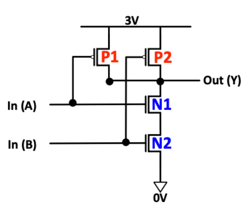


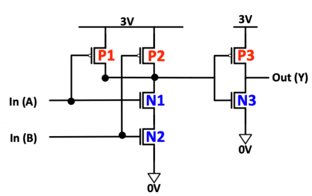
A	Y
0	1
1	0

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

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



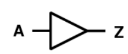




https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=onur-digitaldesign_comparch-2021-lecture4-combinational-logic-afterlecture.pdf

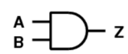
Common logic gates

Buffer



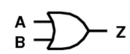
A	Z
0	0
1	1

AND



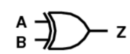
A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR



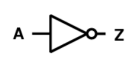
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

XOR



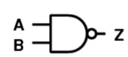
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Inverter



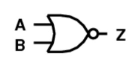
A	Z
0	1
1	0

NAND



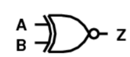
A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR



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

XNOR



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

https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=onur-digitaldesign_comparch-2021-lecture4-combinational-logic-afterlecture.pdf

Building logic circuits

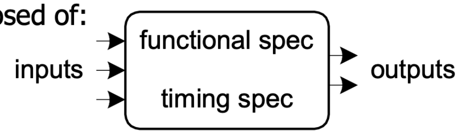
Now, we understand the workings of the basic logic gates

What is our next step?

Build some of the logic structures that are important components of the microarchitecture of a computer!

A logic circuit is composed of:

- Inputs
- Outputs

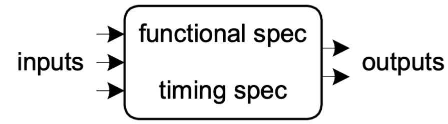


Functional specification (describes relationship between inputs and outputs)

Timing specification (describes the delay between inputs changing and outputs responding)

https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=onur-digitaldesign_comparch-2021-lecture4-combinational-logic-afterlecture.pdf

Types of logic circuits



Combinational Logic

- Memoryless
- Outputs are strictly dependent on the combination of input values that are being applied to circuit *right now*
- In some books called Combinatorial Logic

Later we will learn: Sequential Logic

- Has memory
  - Structure stores history → Can "store" data values
- Outputs are determined by previous (historical) and current values of inputs

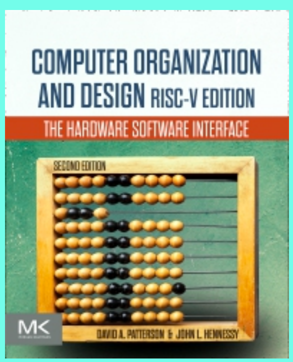
https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=onur-digitaldesign_comparch-2021-lecture4-combinational-logic-afterlecture.pdf

Disclaimer

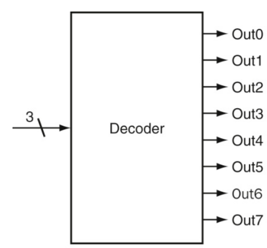
Some of the following slides are adapted from:

Computer Organization and Design (Patterson and Hennessy)

The Hardware/Software Interface



3-bit decoder



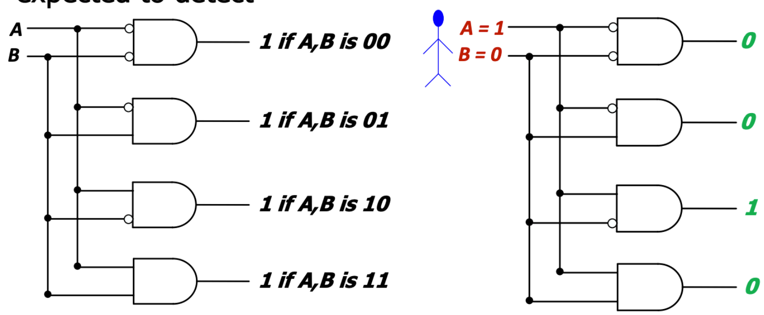
a. A 3-bit decoder

Inputs			Outputs							
12	11	10	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

b. The truth table for a 3-bit decoder

Decoder

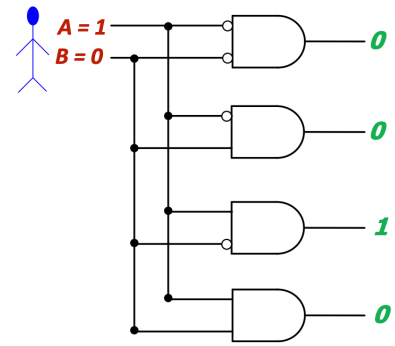
- n inputs and 2^n outputs
- Exactly one of the outputs is 1 and all the rest are 0s
- The one output that is logically 1 is the output corresponding to the input pattern that the logic circuit is expected to detect



https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=onur-digitaldesign_comparch-2021-lecture4-combinational-logic-afterlecture.pdf

Decoder

- The decoder is useful in determining how to interpret a bit pattern
  - **It could be the address of a row in DRAM, that the processor intends to read from**
  - **It could be an instruction in the program and the processor has to decide what action to do! (based on *instruction opcode*)**

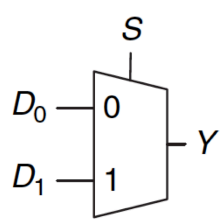


https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=onur-digitaldesign_comparch-2021-lecture4-combinational-logic-afterlecture.pdf

Multiplexor

- Selects one of the *N* inputs to connect it to the output
  - based on the value of a $\log_2 N$-bit control input called select
- Example: 2-to-1 MUX

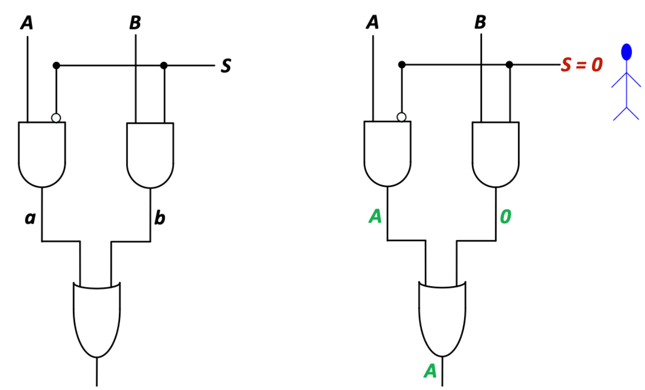
S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



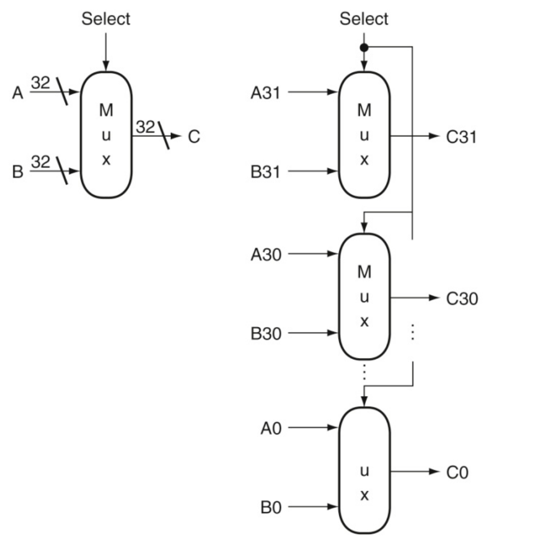
https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=onur-digitaldesign_comparch-2021-lecture4-combinational-logic-afterlecture.pdf

Multiplexor

- Selects one of the *N* inputs to connect it to the output
  - based on the value of a $\log_2 N$-bit control input called select
- Example: 2-to-1 MUX



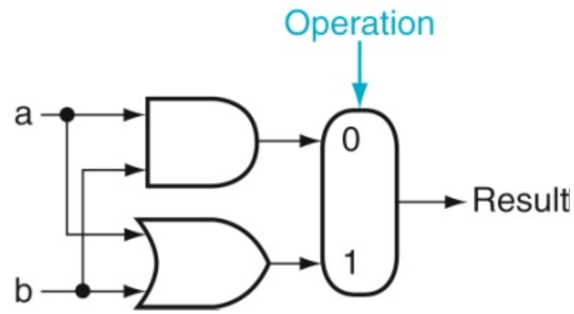
https://safari.ethz.ch/digitaltechnik/spring2021/lib/exe/fetch.php?media=onur-digitaldesign_comparch-2021-lecture4-combinational-logic-afterlecture.pdf



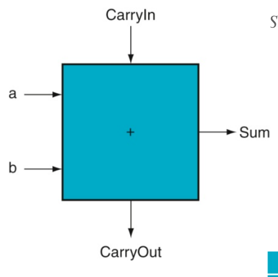
a. A 32-bit wide 2-to-1 multiplexor

b. The 32-bit wide multiplexor is actually an array of 32 1-bit multiplexors

1-bit logical unit for AND and OR



1-bit adder

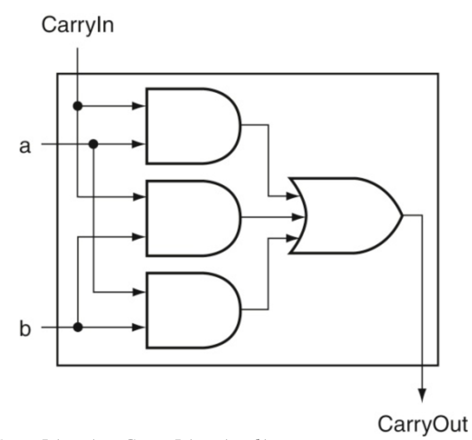


$$Sum = (a \cdot \overline{b} \cdot \overline{CarryIn}) + (\overline{a} \cdot b \cdot \overline{CarryIn}) + (\overline{a} \cdot \overline{b} \cdot CarryIn) + (a \cdot b \cdot CarryIn)$$

$$CarryOut = (b \cdot CarryIn) + (a \cdot CarryIn) + (a \cdot b)$$

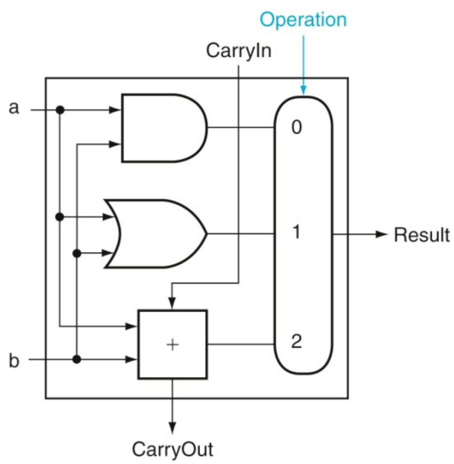
Inputs			Outputs		
a	b	CarryIn	CarryOut	Sum	Comments
0	0	0	0	0	$0 + 0 + 0 = 00_{two}$
0	0	1	0	1	$0 + 0 + 1 = 01_{two}$
0	1	0	0	1	$0 + 1 + 0 = 01_{two}$
0	1	1	1	0	$0 + 1 + 1 = 10_{two}$
1	0	0	0	1	$1 + 0 + 0 = 01_{two}$
1	0	1	1	0	$1 + 0 + 1 = 10_{two}$
1	1	0	1	0	$1 + 1 + 0 = 10_{two}$
1	1	1	1	1	$1 + 1 + 1 = 11_{two}$

Adder hardware for CarryOut

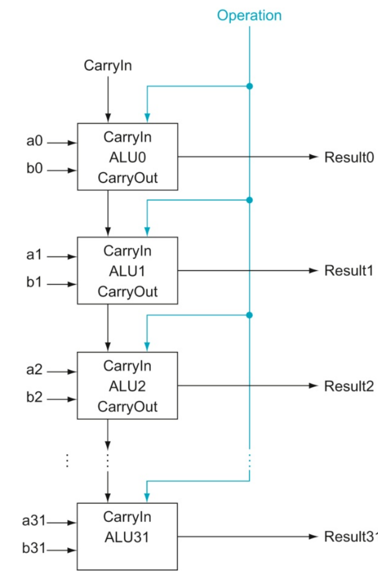


$$CarryOut = (b \cdot CarryIn) + (a \cdot CarryIn) + (a \cdot b)$$

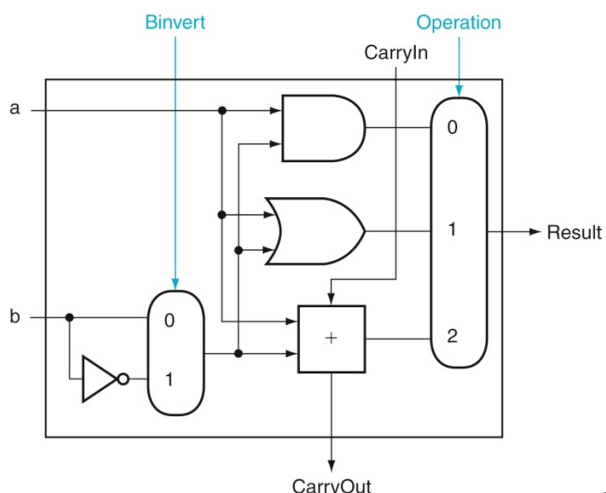
1-bit ALU (AND, OR, Add)



32-bit ALU

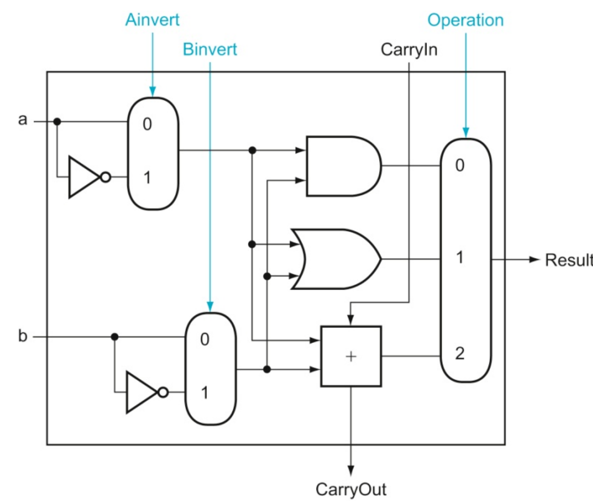


1-bit ALU (AND, OR, Add, Sub)



invert and set CarryIn = 1

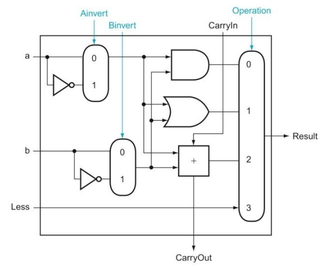
1-bit ALU (AND, OR, Add, Sub, NOR)

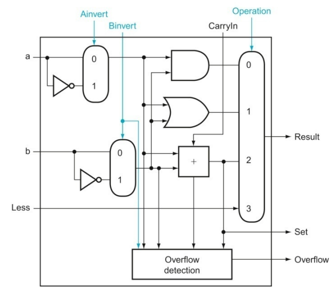


(Top) A 1-bit ALU

(Bottom) a 1-bit ALU for the most significant bit.

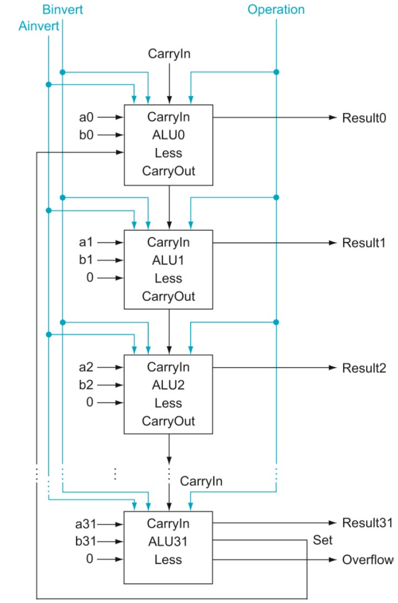
The top drawing includes a direct input that is connected to perform the set on less than operation; the bottom has a direct output from the adder for the less than comparison called Set.





32-bit ALU

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract



Adding a zero detector