Lesson 7

Combinational Logic Circuits

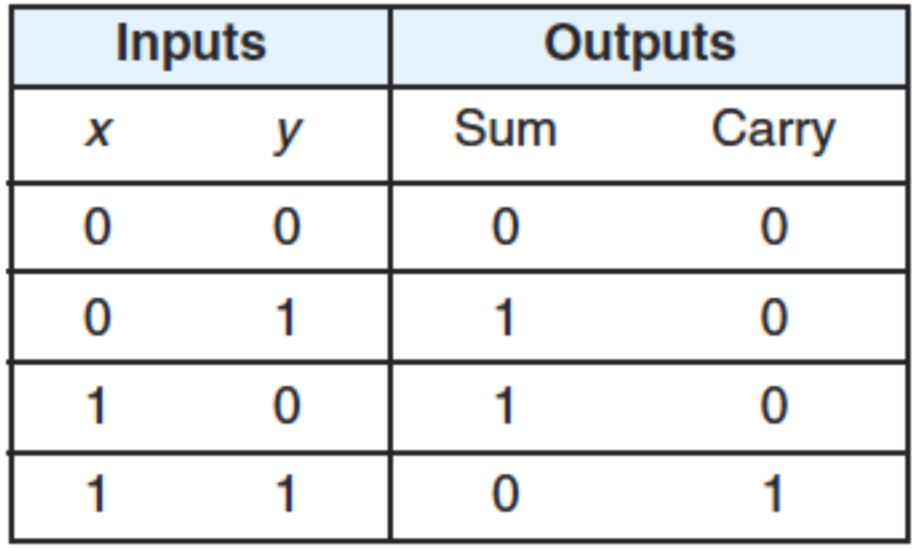
Concepts Introduced

- Adders
- Multiplexers
- Decoders
- basic Arithmetic/Logic Unit

Review Half-Adder

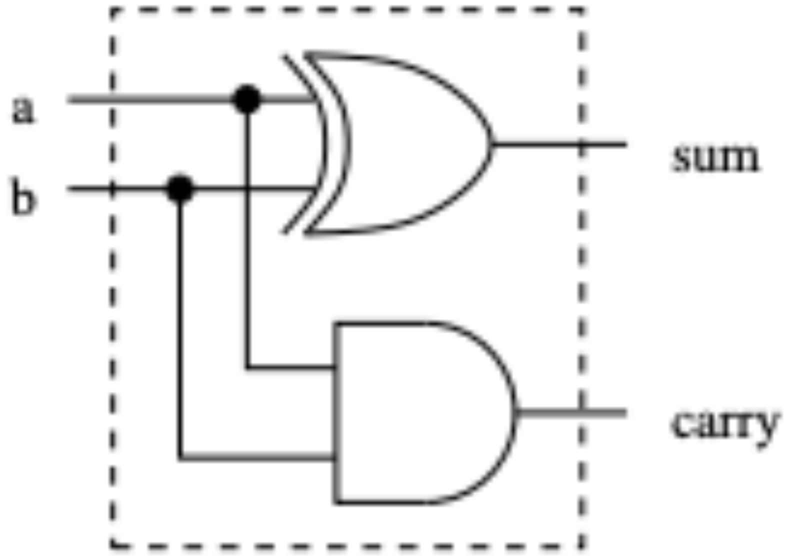
- A half adder takes two inputs, a and b, and generates two outputs, the carry and the sum.
- A half adder is called a (2,2) adder as it takes two inputs and produces two outputs.
- The circuit for the carry can use an AND gate.
- The circuit for the sum can use an XOR gate.

Truth Table for a Half-Adder



Inputs		Outputs	
x	*y*	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

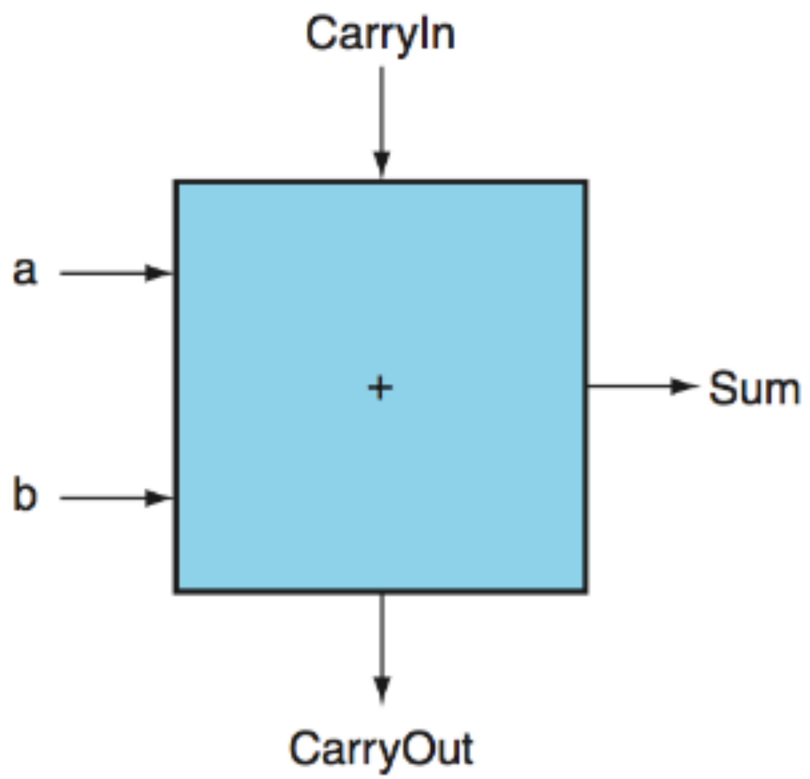
Half-adder



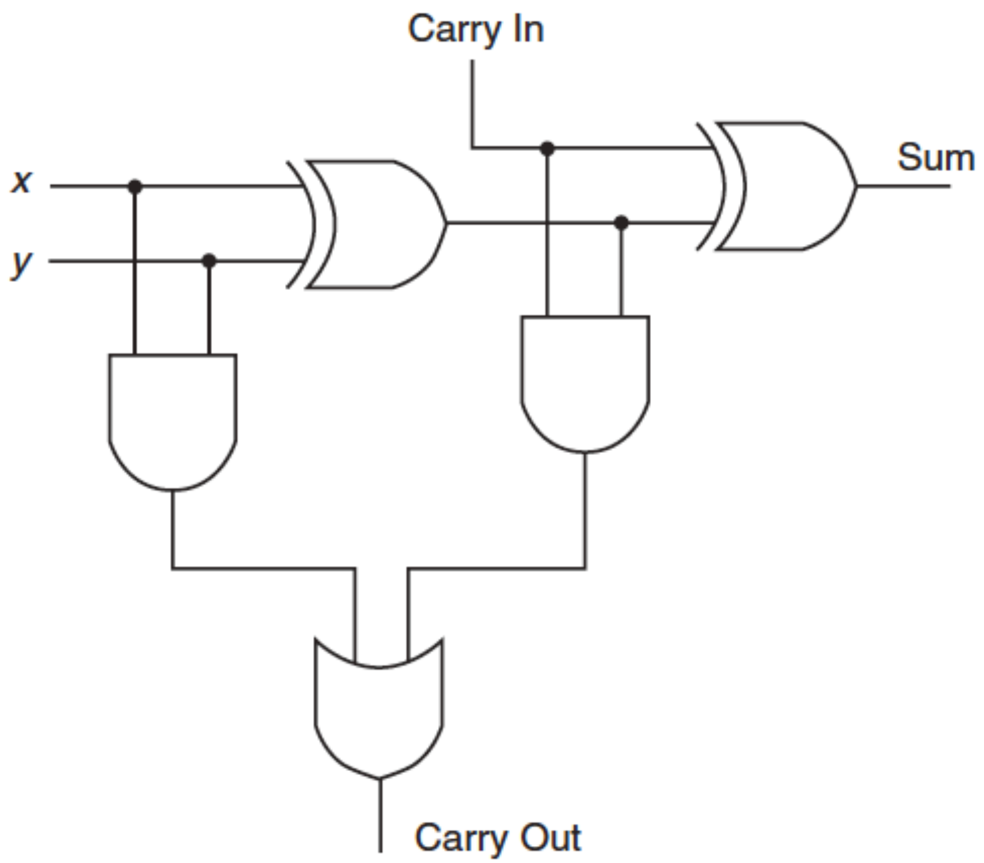
Full-adder

- Full-adder is composed of two half-adders and an OR gate.
- The full-adder is a three input and two output combinational circuit.
- The first two inputs are A and B and the third input is an input carry as C-IN.
- The output carry is designated as C-OUT and the normal output is designated as S which is SUM.
- Therefore, a full adder adds binary numbers and accounts for values carried in as well as out
- A full adder adds binary numbers and accounts for values carried in as well as out.

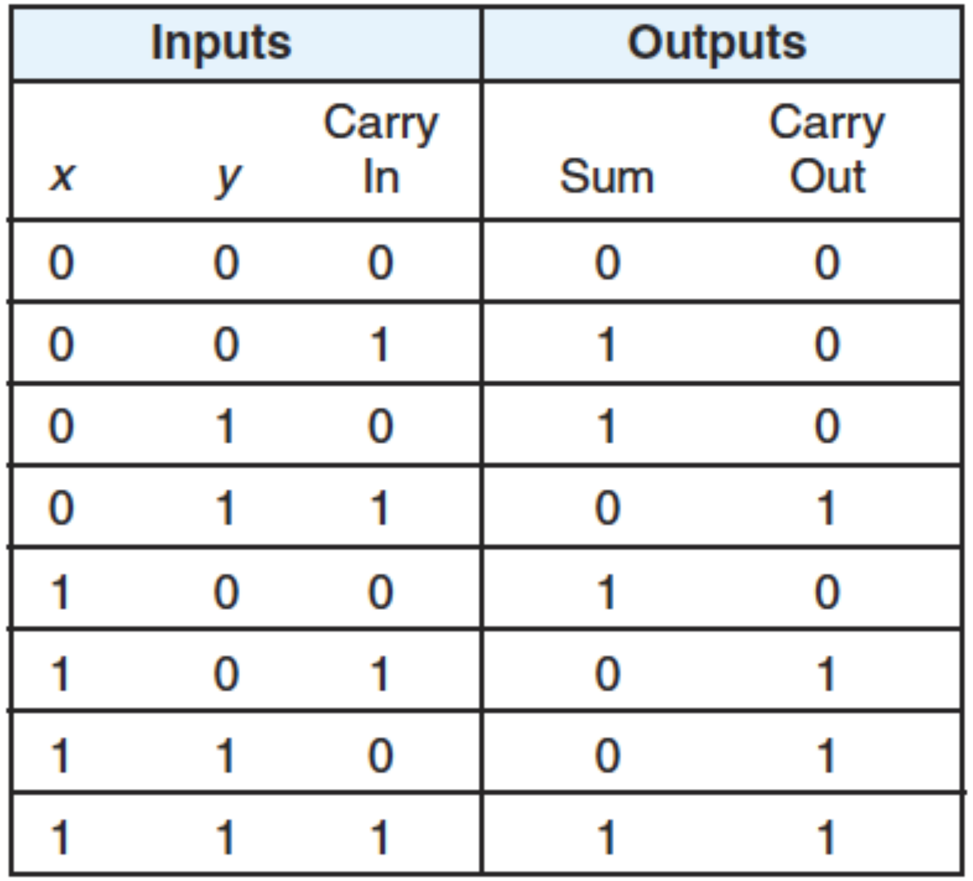
Full-adder



A Logic Diagram for a Full-Adder



A Truth Table for a Full-Adder



Inputs			Outputs	
x	*y*	Carry In	Sum	Carry 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

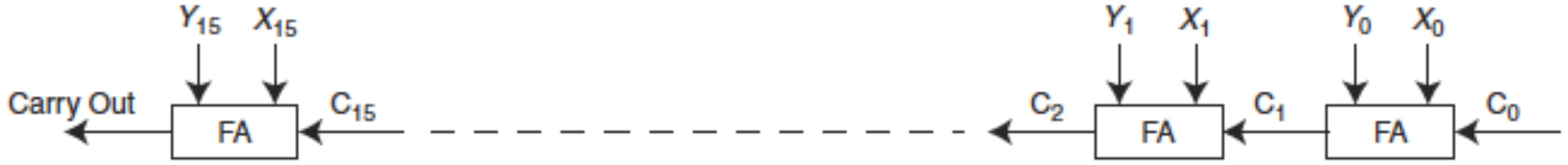
Sum = x’.y’.CarryIn + x’.b.CarrryIn’ + a.b’.CarryIn’ + a.b.CarryIn

CarryOut = x’.y.CarryIn + x.b’.CarrryIn + a.b.CarryIn’ + a.b.CarryIn

Ripple-carry adder

- We can be able to create a logical circuit using multiple adders to add N bit numbers.
- Each full adder inputs a C_{in}, which is the C_{out} of the previous adder,
- we can build an adder capable of adding two 16-bit words, for example, by replicating the above circuit 16 times, feeding the Carry Out of one circuit into the Carry In of the circuit immediately to its left.
- This circuit is called a ripple-carry adder

Ripple-Carry Adder



Adders Summary

- Adders are very important circuits—a computer would not be very useful if it could not add numbers
- Another important operation that all computers use often is decoding binary information from a set of n inputs to a maximum of 2^n outputs.

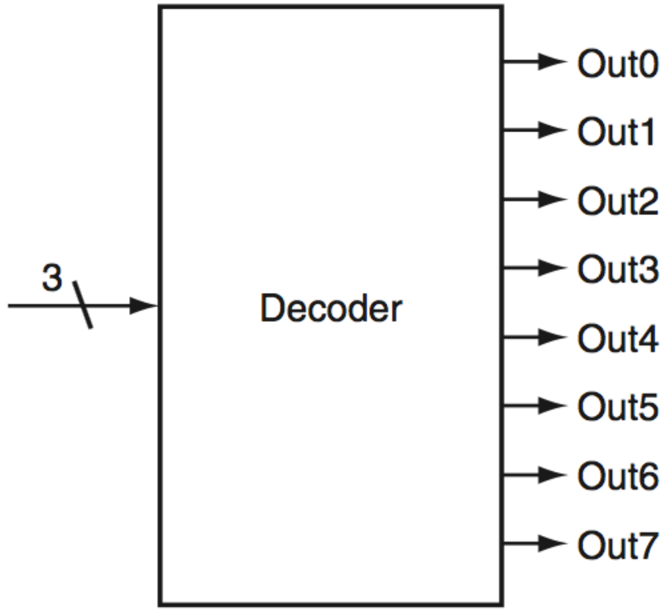
Decoder

- A decoder uses the inputs and their respective values to select one specific output line. By selecting we mean that one unique output line is asserted, or set to 1 while the other output lines are set to zero.
- Decoders are defined by the number of inputs and the number of outputs. a decoder that has 3 inputs and 8 outputs is called a 3-to-8 decoder.
- All memory addresses in a computer are specified as binary numbers. When a memory address is referenced, the computer first has to determine the actual address. This is done by using a decoder

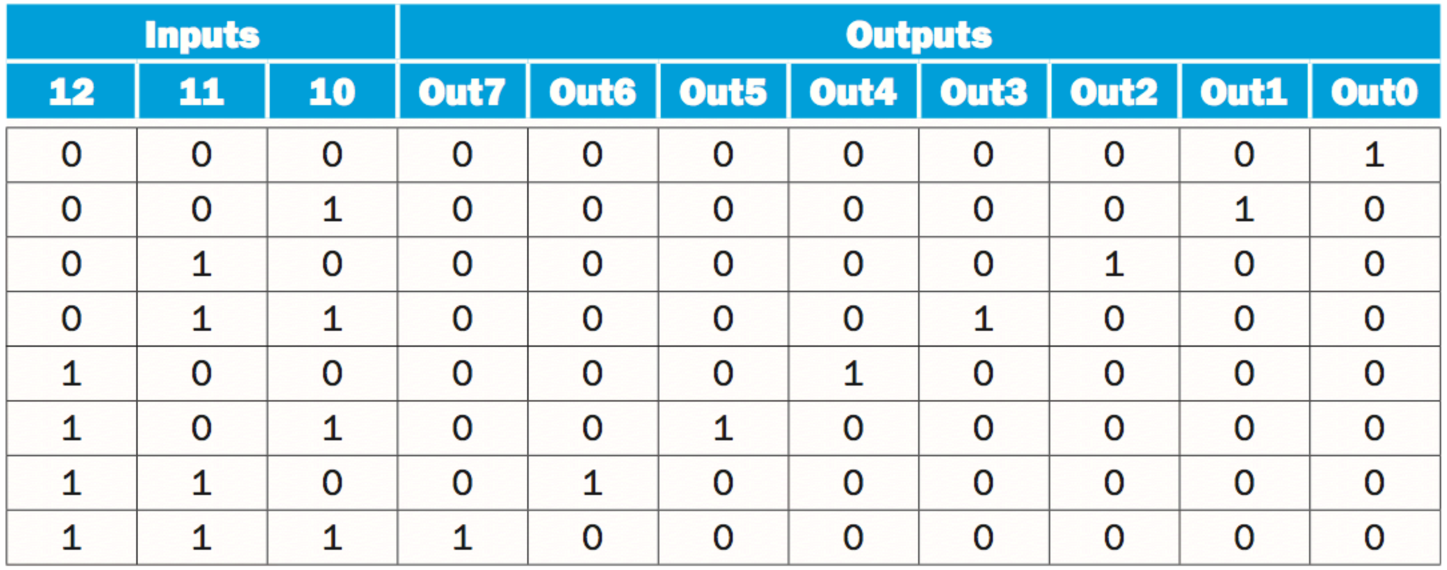
Decoders Continued

- The most common type of decoder has an *n*-bit input and 2^n outputs, where only one output is asserted for each input combination.
- This decoder translates the *n*-bit input into a signal that corresponds to the binary value of the *n*-bit input.

A 3-bit decoder has 3 inputs, called 12, 11, and 10, and 2^3 = 8 outputs, called Out0 to Out7



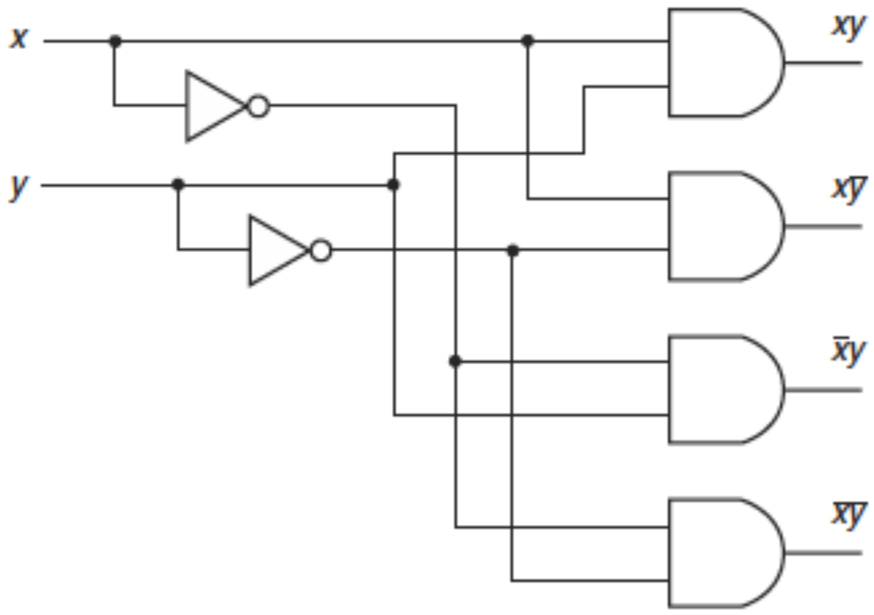
a. A 3-bit decoder

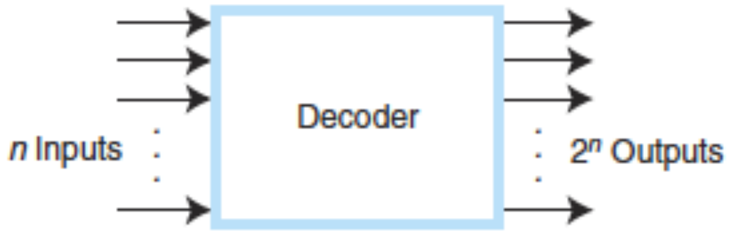


Inputs			Outputs							
12	11	10	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

b. The truth table for a 3-bit decoder

Decode and Decoder Symbol (left to right)





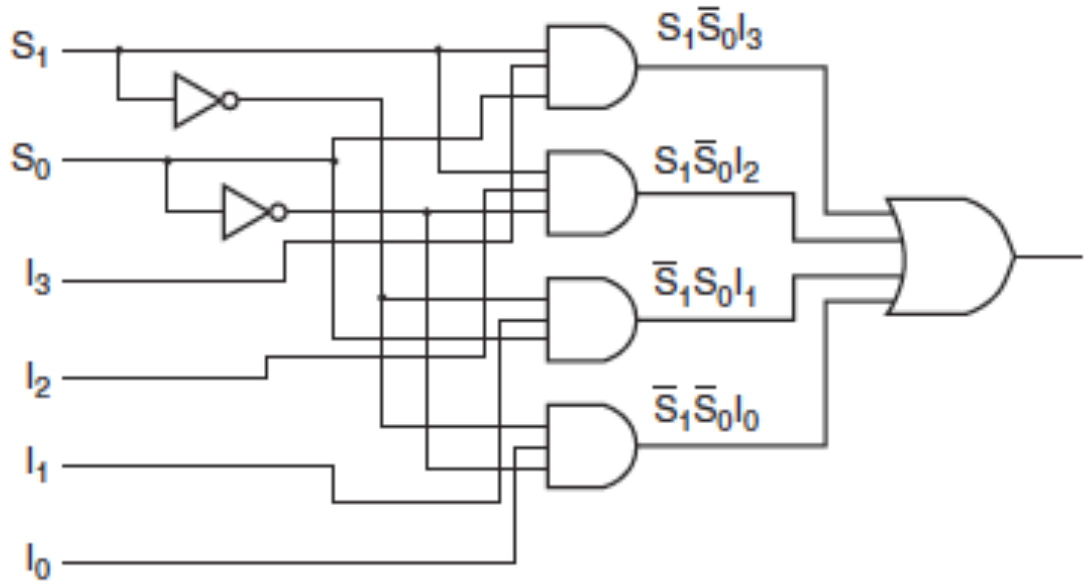
Multiplexors

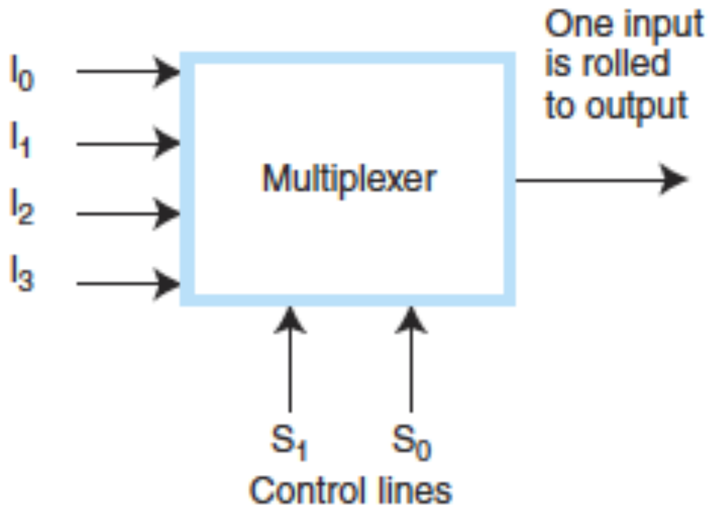
- A ***multiplexor*** is also called a ***selector***, since its output is one of the inputs that is selected by a control.
- Multiplexors can be created with an arbitrary number of data inputs.
- When there are only two inputs, the selector is a single signal that selects one of the inputs if it is true (1) and the other if it is false (0)
- If there are *n* data inputs, there will need to be $\lceil \log_2 n \rceil$ selector input

Parts of a multiplexor

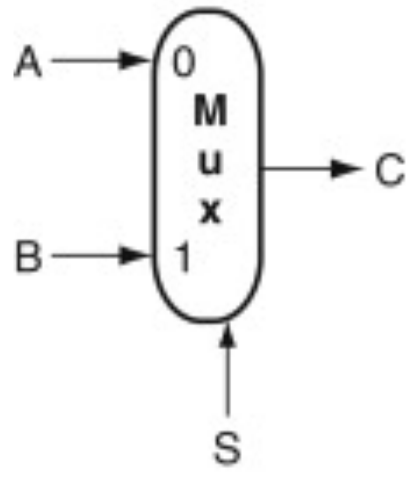
- A decoder that generates *n* signals, each indicating a different input value
- An array of *n* AND gates, each combining one of the inputs with a signal from the decoder
- A single large OR gate that incorporates the outputs of the AND gates

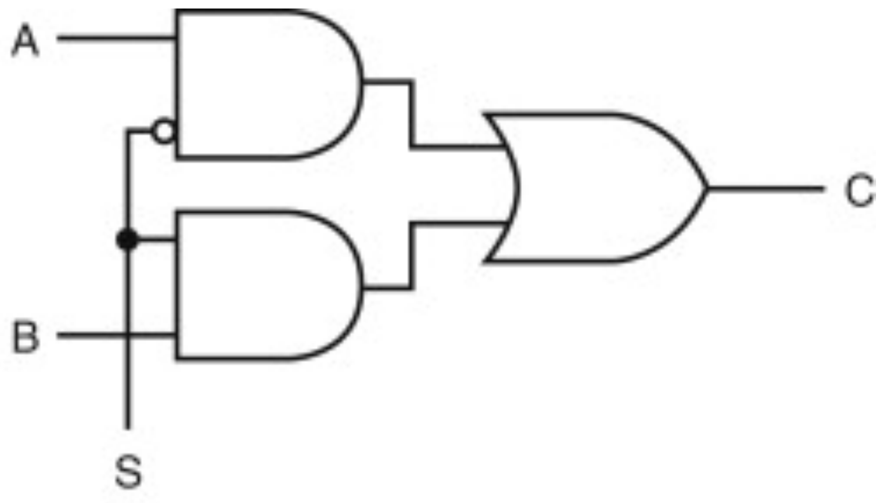
A Look Inside a Multiplexer and A Multiplexer Symbol





two-input multiplexor on the left and its implementation with gates on the right

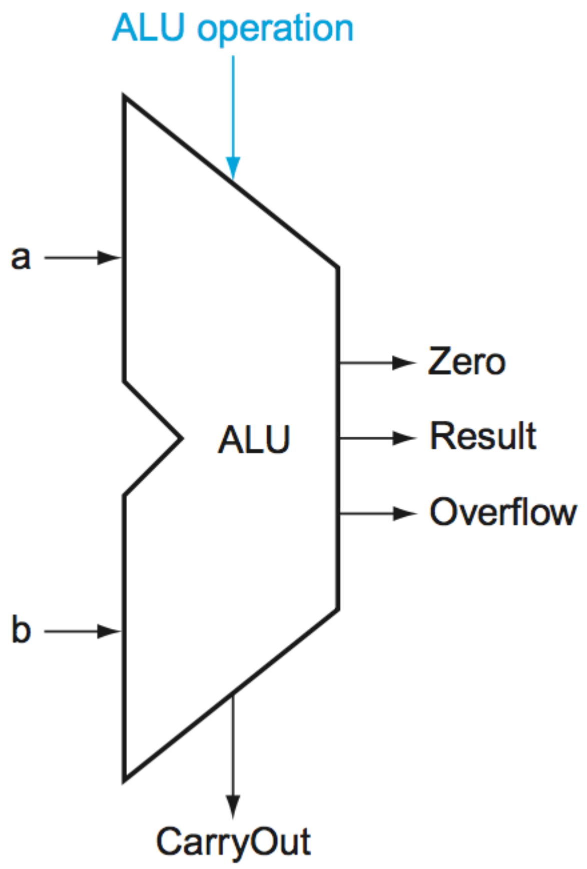




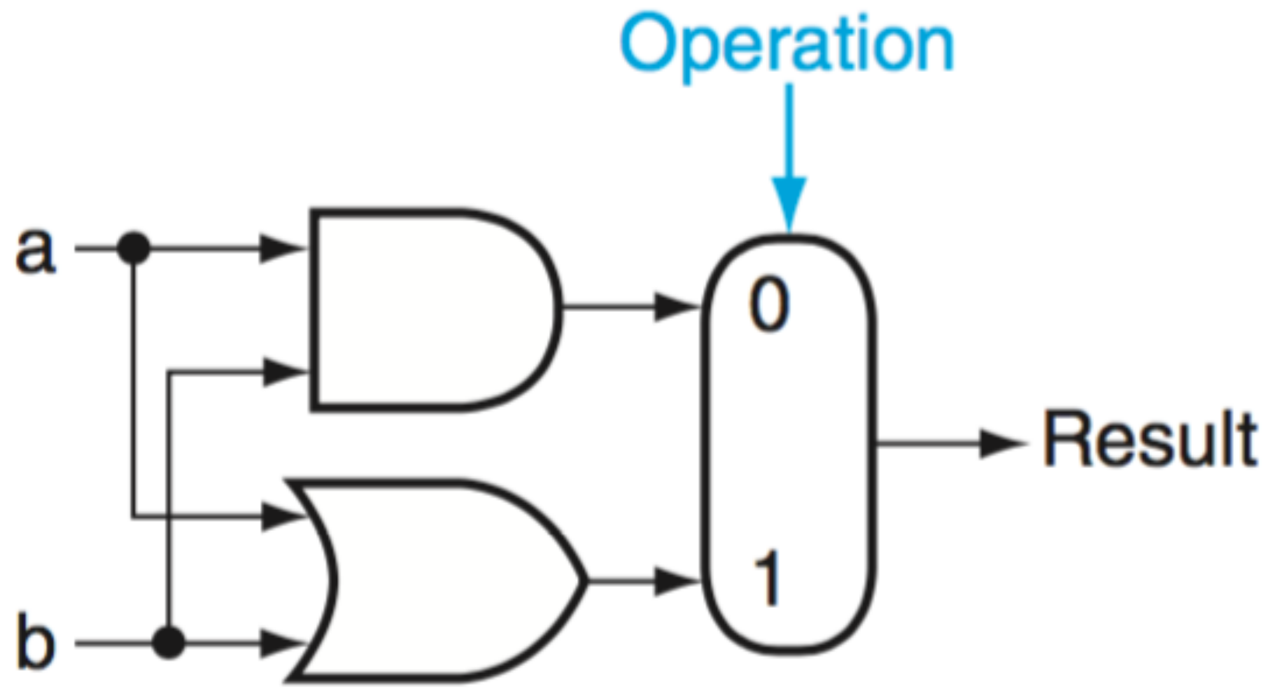
Arithmetic Logic Unit (ALU)

- We have discussed enough combinational circuit to build an ALU
- The arithmetic logic unit (ALU) carries out the logic operations (such as comparisons) and arithmetic operations (such as add or multiply) required during the program execution.
- the device that performs the arithmetic operations like addition and subtraction or logical operations like AND and OR.
- Generally an ALU has two data inputs and one data output.
- Operations performed in the ALU often affect bits in the status register
- The ALU knows which operations to perform because it is controlled by signals from the control unit.

The symbol to represent an ALU



1-bit logical unit for AND and OR



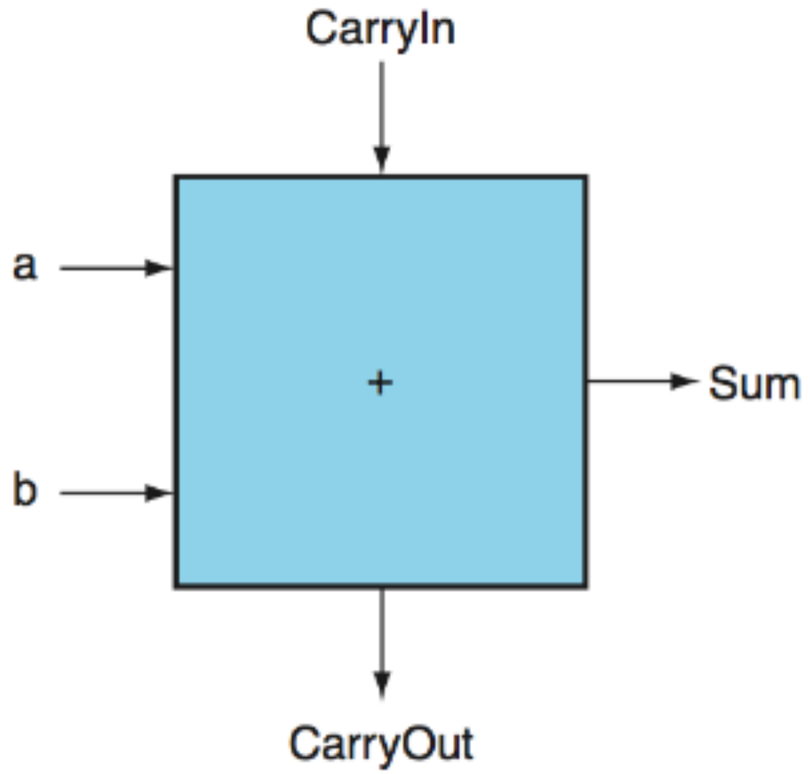
1-bit logical unit for AND and OR

- In the 1-bit logical unit for AND and OR, the multiplexor on the right then selects *a* AND *b* or *a* OR *b*, depending on whether the value of *Operation* is 0 or 1
- Both the AND and OR operations are always performed, but the output produced depends on the selector to the multiplexor
- A 32-bit logical unit for AND and OR operations would just be an array of these 1-bit logical units.

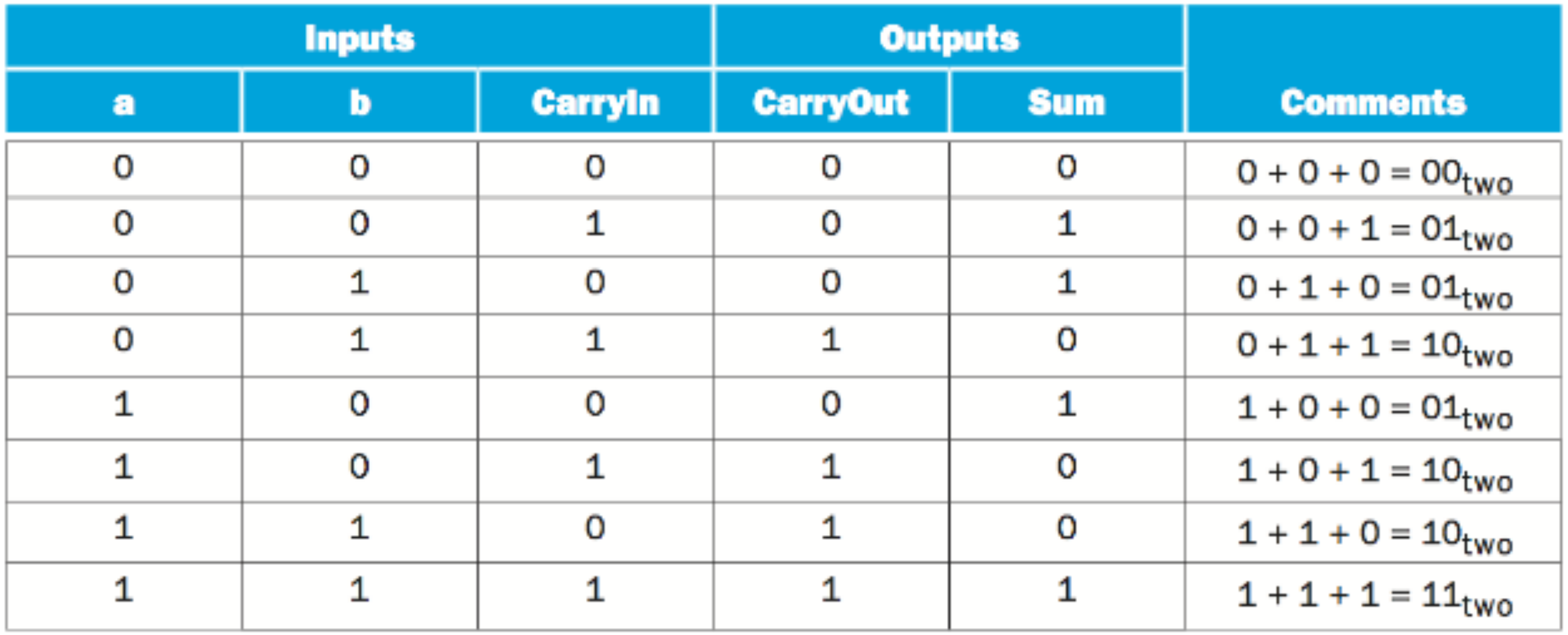
ALU

- We next include addition to the ALU
- An adder must have two inputs for the operands and a single-bit output for the sum. There must be a second output to pass on the carry, called *CarryOut*.
- We also need a third input. This input is called *CarryIn*.

Full-adder

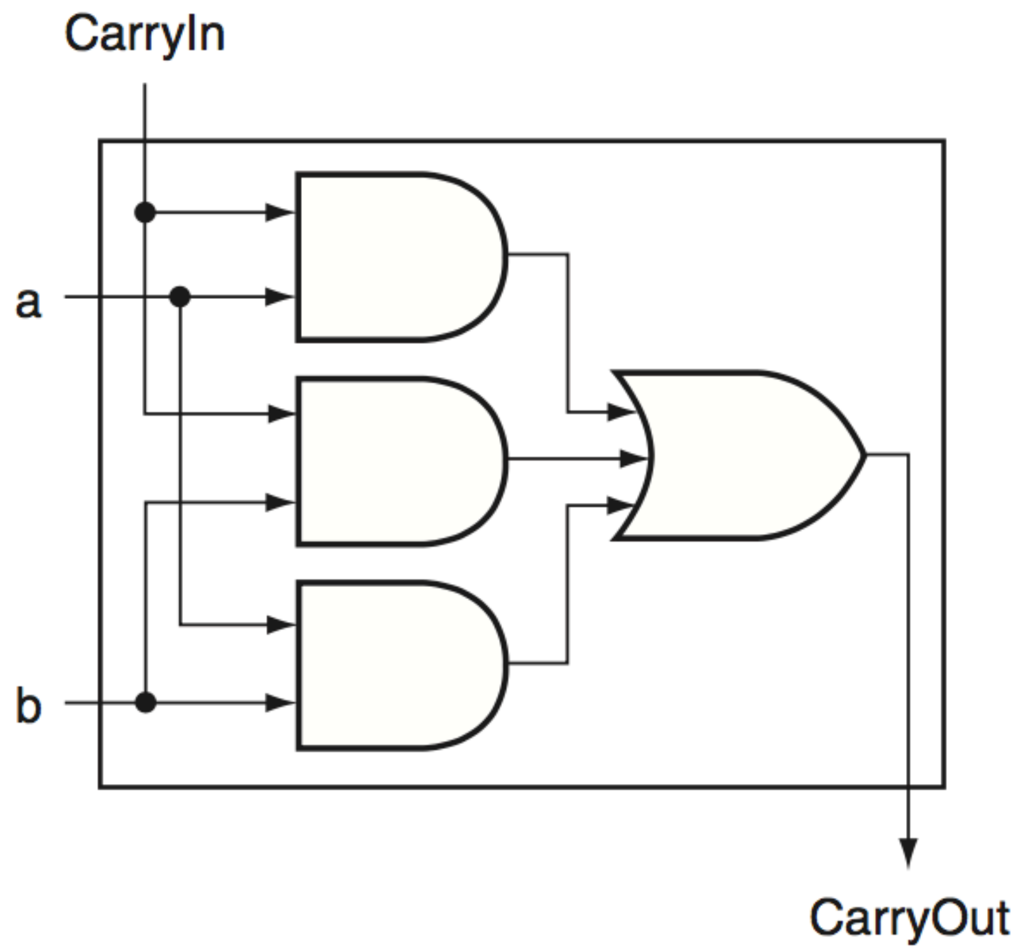


Input and output specification for a 1-bit adder

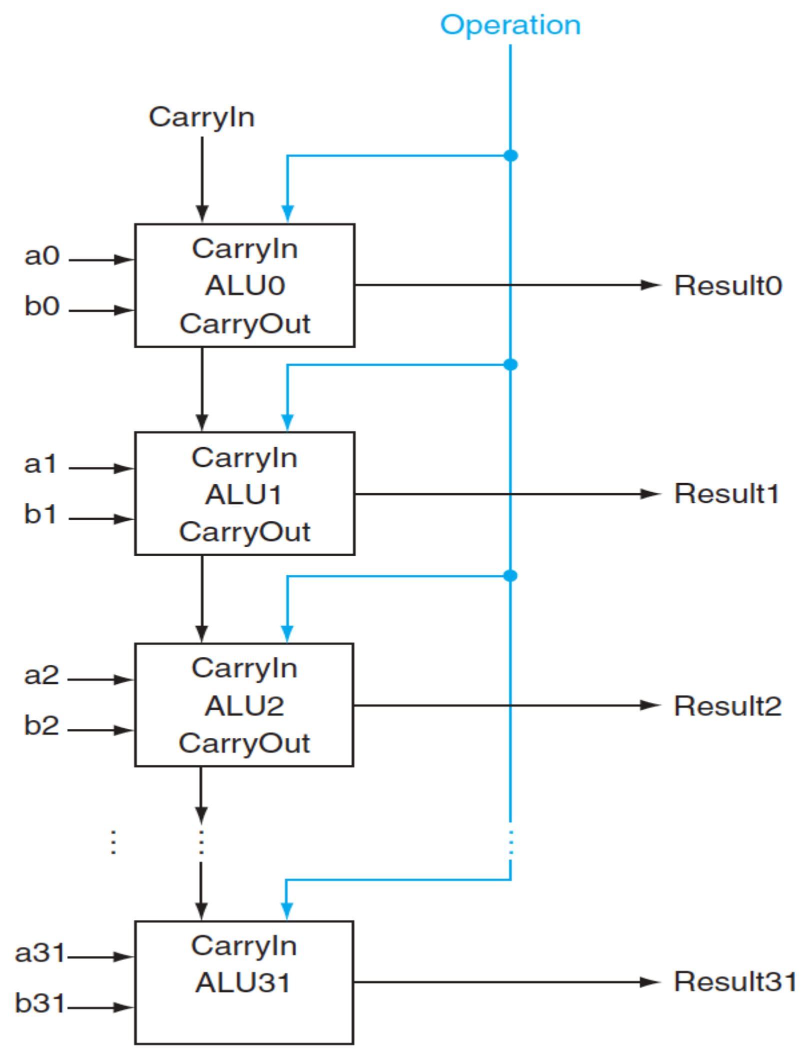


Inputs			Outputs		
a	b	CarryIn	CarryOut	Sum	Comments
0	0	0	0	0	$0 + 0 + 0 = 00_{two}$
0	0	1	0	1	$0 + 0 + 1 = 01_{two}$
0	1	0	0	1	$0 + 1 + 0 = 01_{two}$
0	1	1	1	0	$0 + 1 + 1 = 10_{two}$
1	0	0	0	1	$1 + 0 + 0 = 01_{two}$
1	0	1	1	0	$1 + 0 + 1 = 10_{two}$
1	1	0	1	0	$1 + 1 + 0 = 10_{two}$
1	1	1	1	1	$1 + 1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

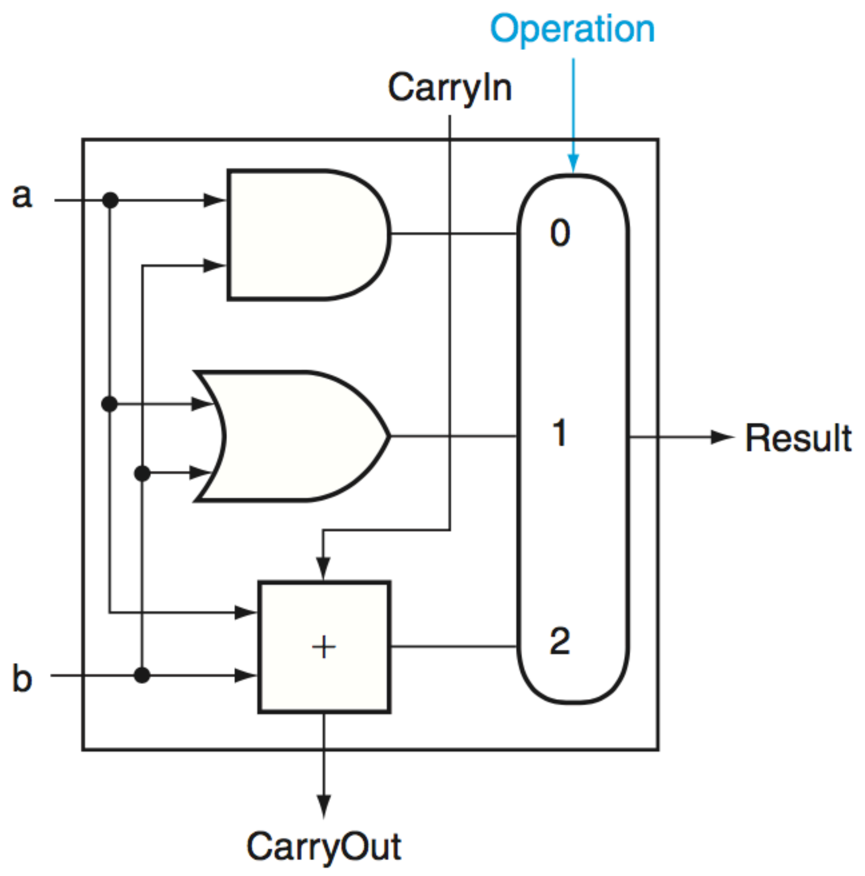
CarryOut from the Full Adder



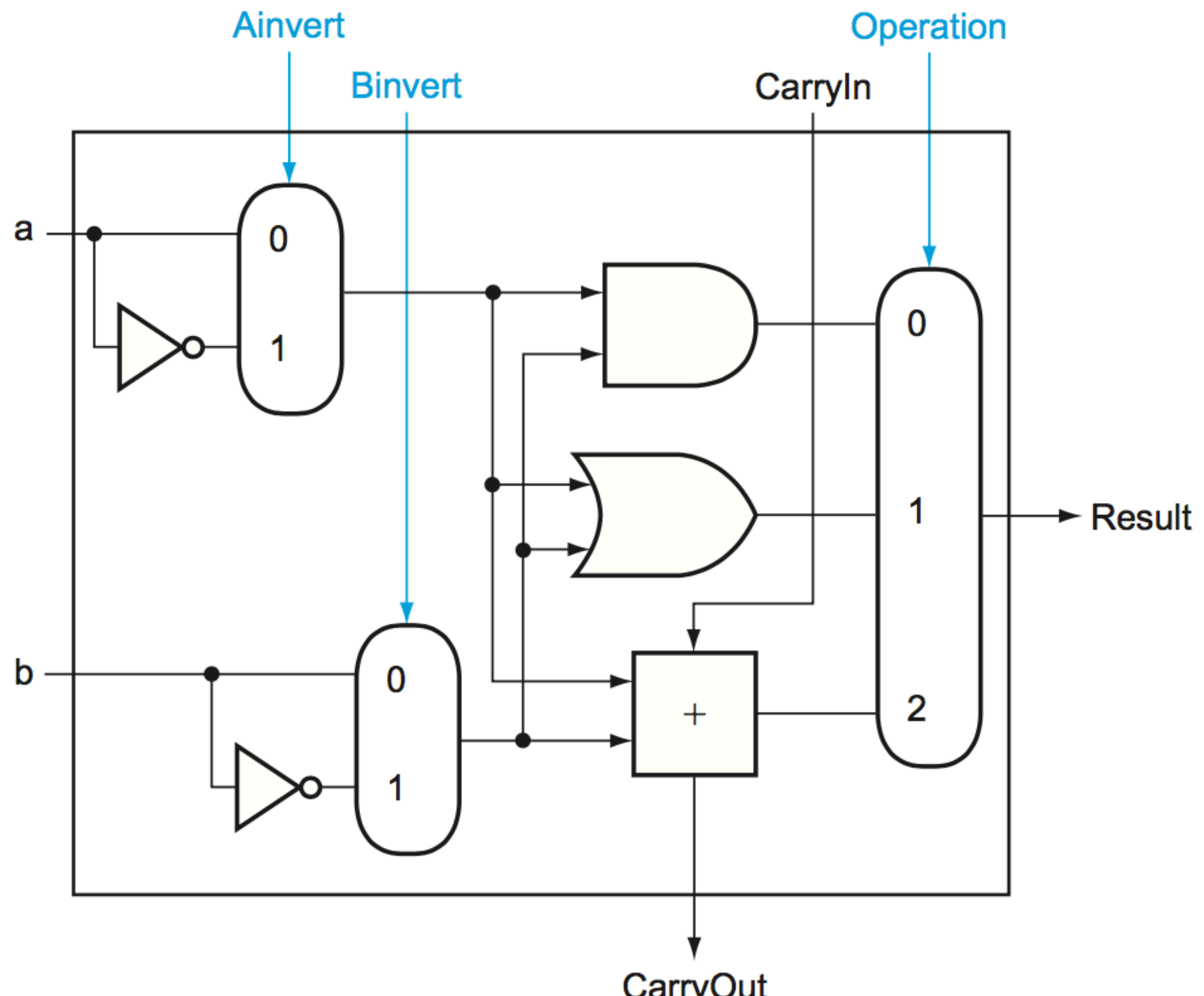
A 32-bit ALU constructed from 32 1-bit ALUs



A 1-bit ALU that performs AND, OR, and addition



A 1-bit ALU that performs AND, OR, and addition on a and b or not a and not b



Reading

- Hennessy and Patterson Chapter 8.3 and 8.5 (Appendix B)