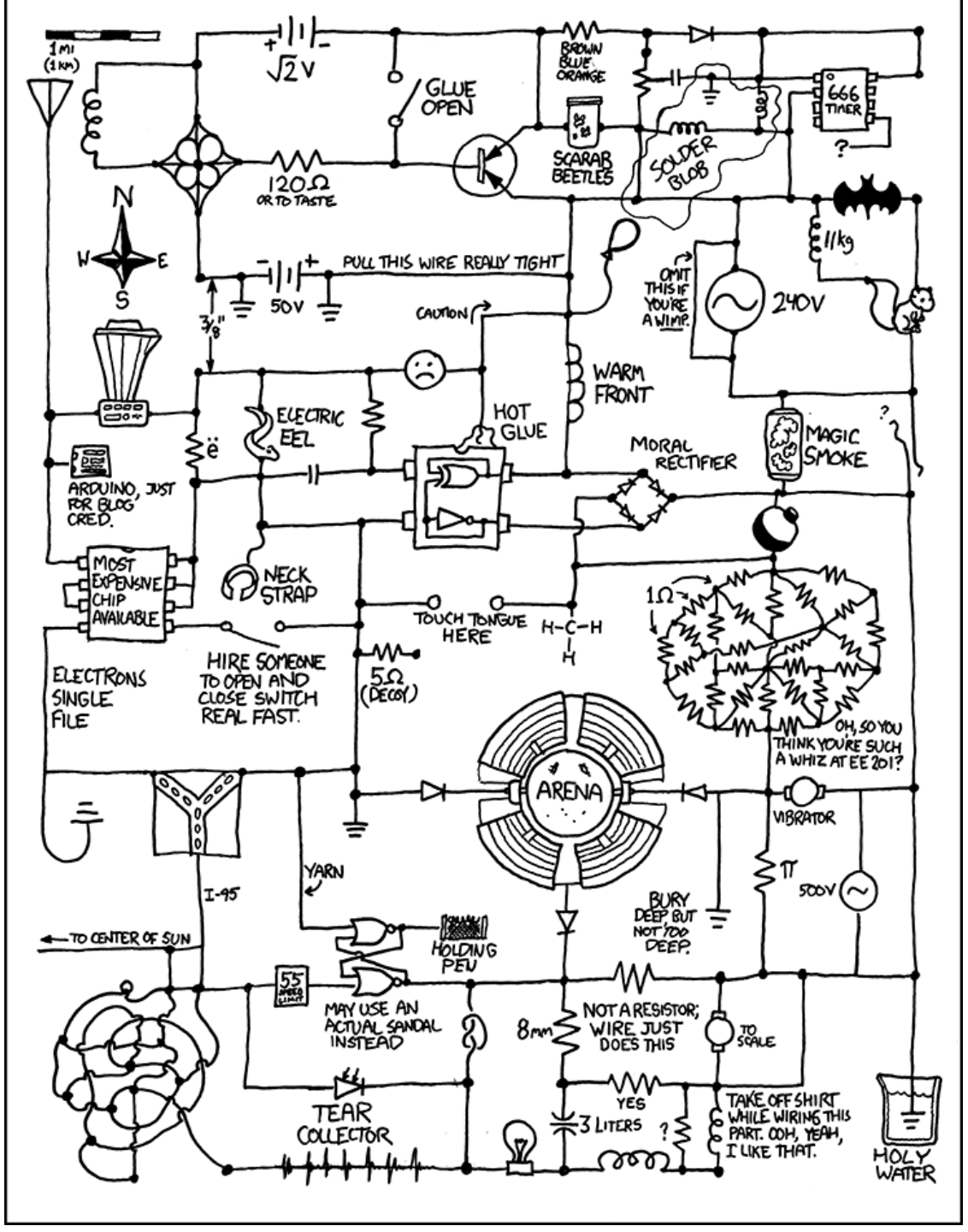
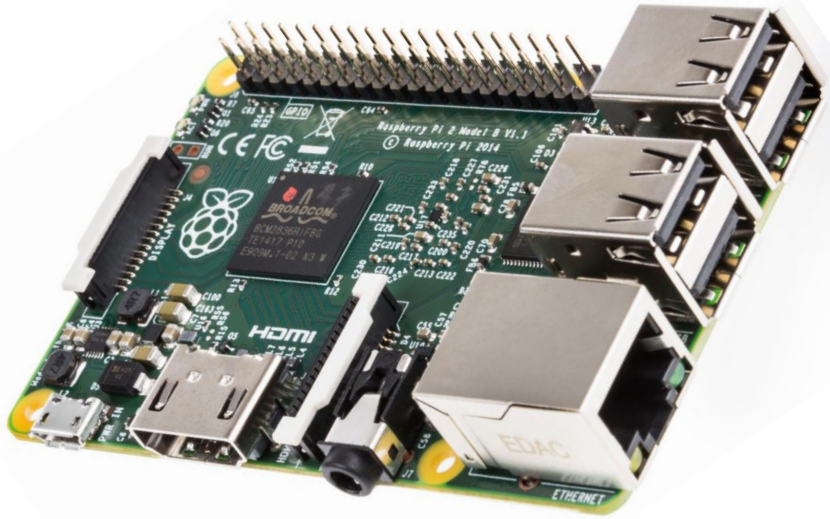
Combinatorial Circuits



Overview

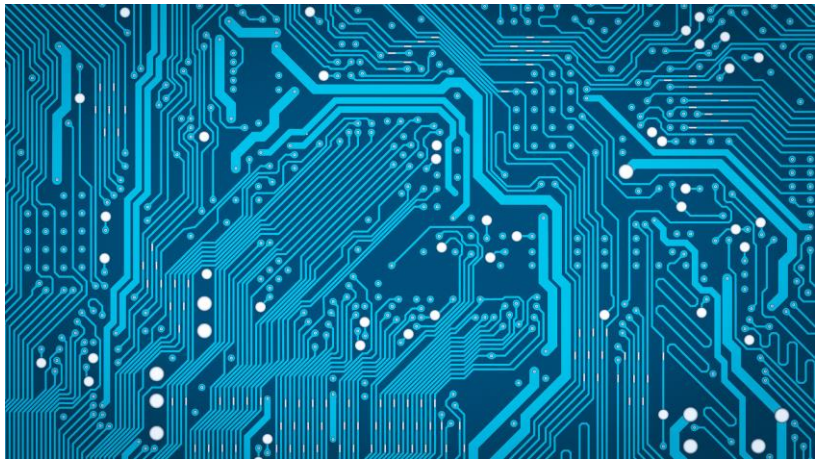
- How to build an adder
- Common combinational circuits
  - Decoders and encoders
  - Multiplexors
  - Read-only memory
  - Programmable logic array
  - Arithmetic logic unit

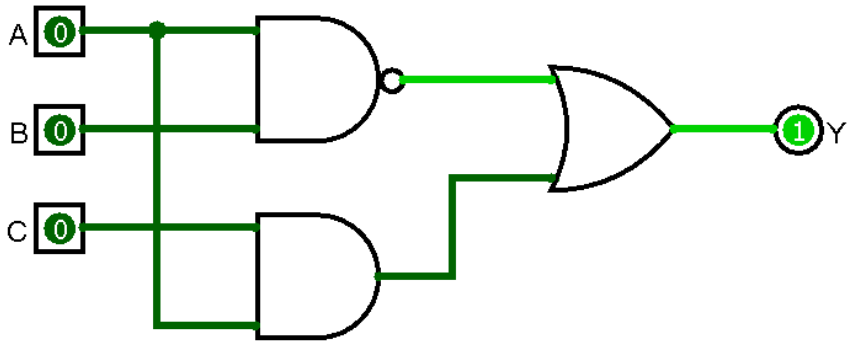
The Big Picture Again



Combinational Circuits

- **Combinational/Combinatorial circuit:** a circuit that implements a Boolean function



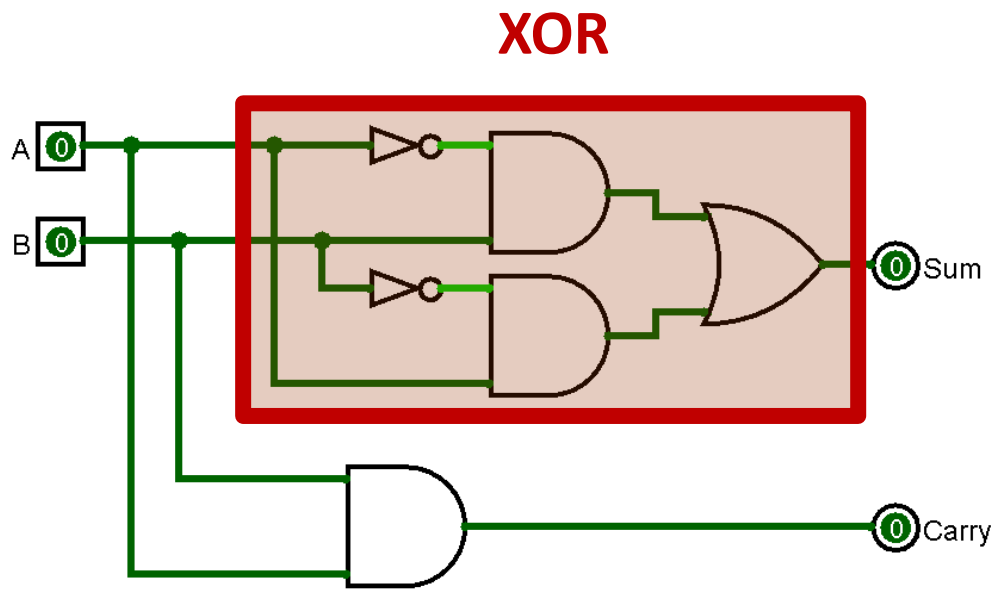


Circuits for Binary Arithmetic

1-Bit Half Adder

- Inputs: two bits A and B
- Outputs: sum and carry out

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1
Sum = A'B + AB' **Carry = AB**			

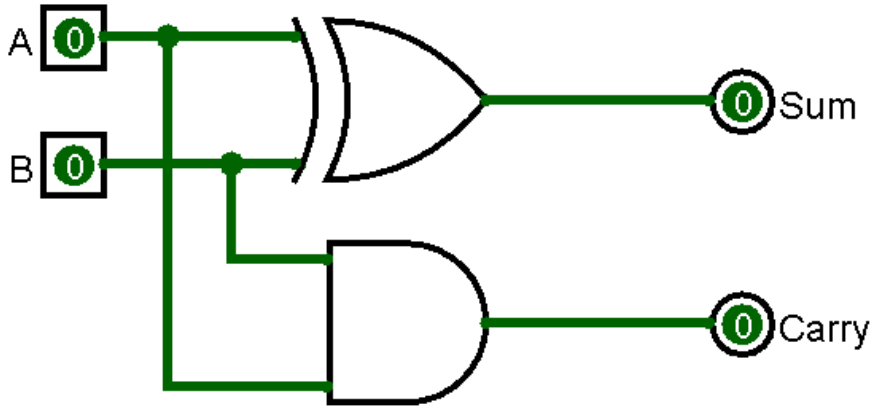


1-Bit Half Adder

- Inputs: two bits
- Outputs: sum and carry out

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

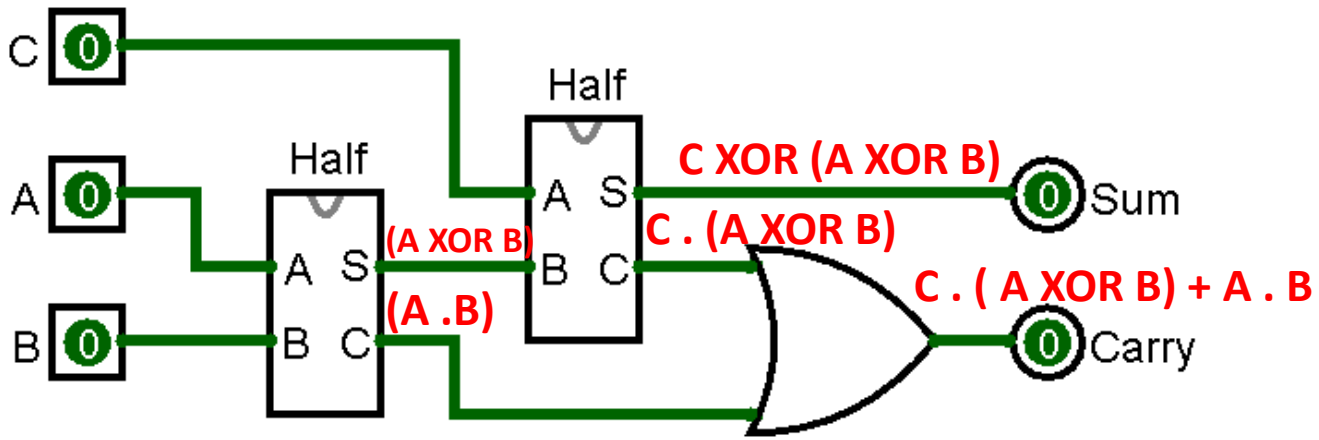
Sum = A XOR B
Carry = AB



1-Bit Full Adder

- Inputs: two bits (A, B) + carry from the previous bit (C)
- Outputs: sum and carry

C	A	B	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

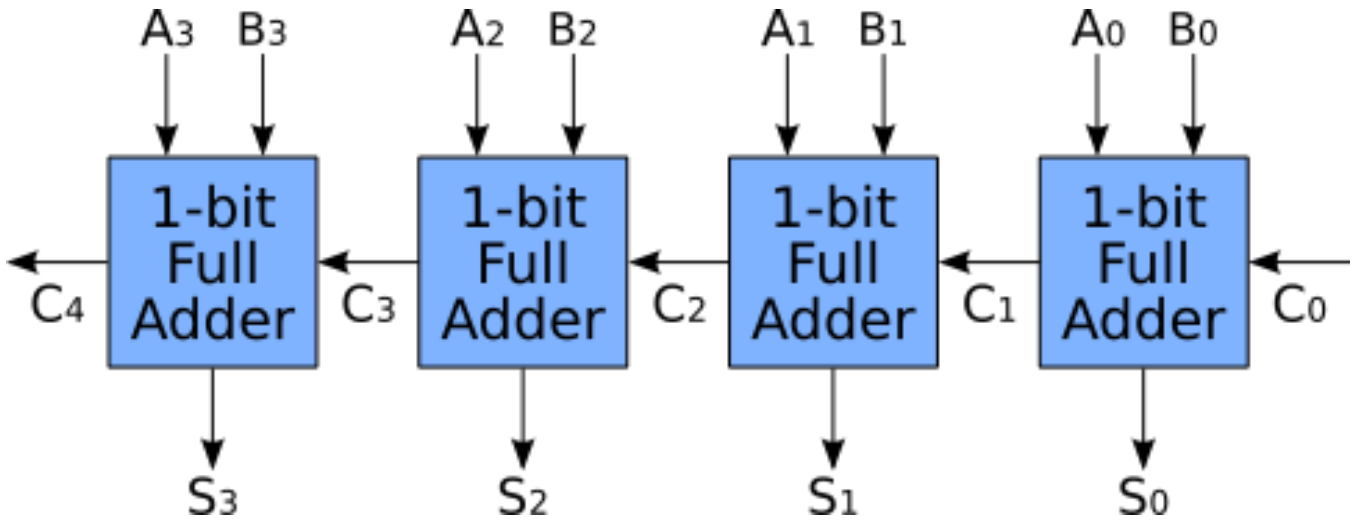


Sum = C' A' B + C' A B' + C A' B' + C A B
= C' (A' B + A B') + C (A' B' + AB)
= C' (A XOR B) + C (A XOR B)'
= C XOR (A XOR B)

Carry = C'AB + CA'B + CAB' + CAB
= C'(AB) + C (A'B + A' B) + C (AB)
= (C'+C)AB + C(A XOR B)
= AB + C(A XOR B)

Multi-Bit Adder

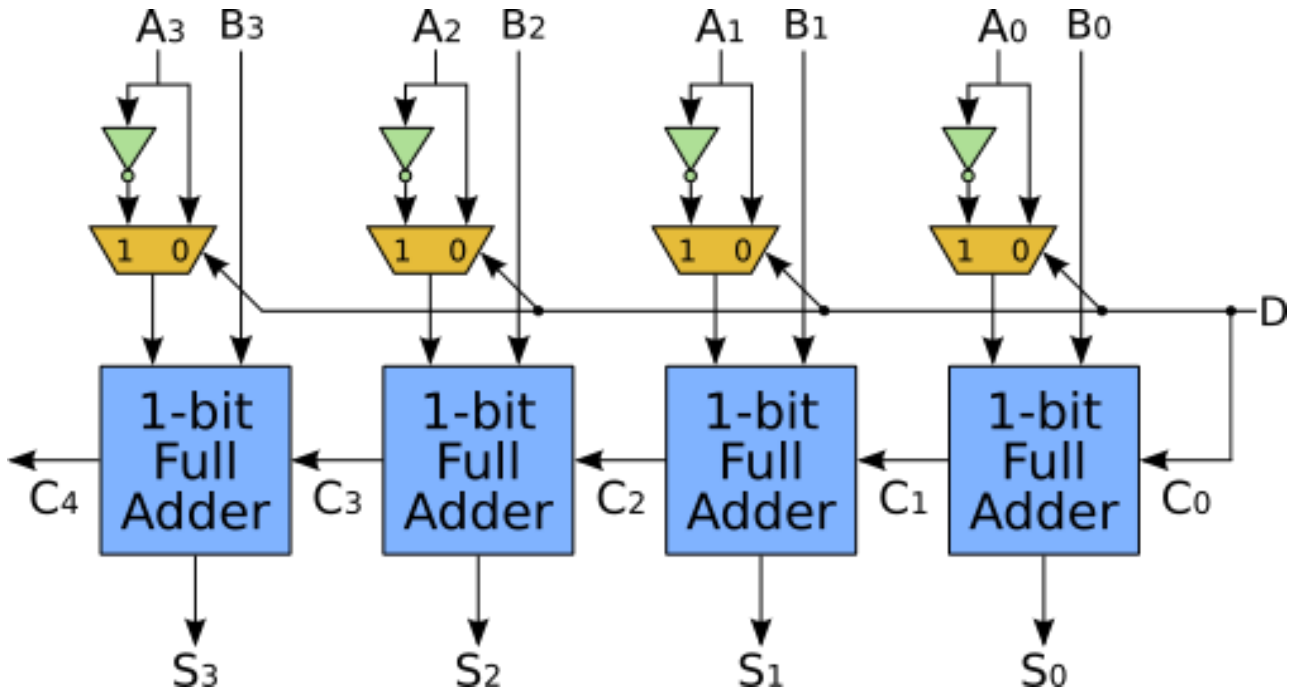
- Chain together adders!



https://commons.wikimedia.org/wiki/File:4-bit_ripple_carry_adder.svg

Multi-bit Subtractor

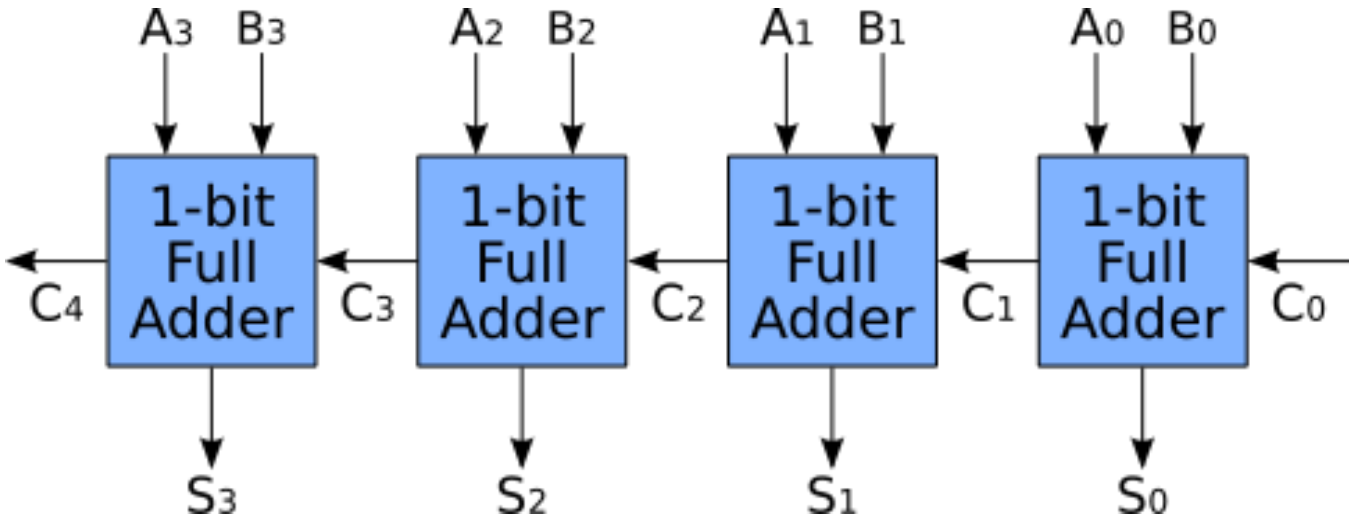
- B - A = B + (-A)
- Inverted bits of A + B. Add one using the first carry in.



https://commons.wikimedia.org/wiki/File:4-bit_ripple_carry_adder-subtracter.svg

Potential Problems?

- Carry must ripple (propagate) through each adder
- The MSB depends on the carry of each preceding bits
- In the case of 32-bit or 64-bit integer, the **delay** is significant



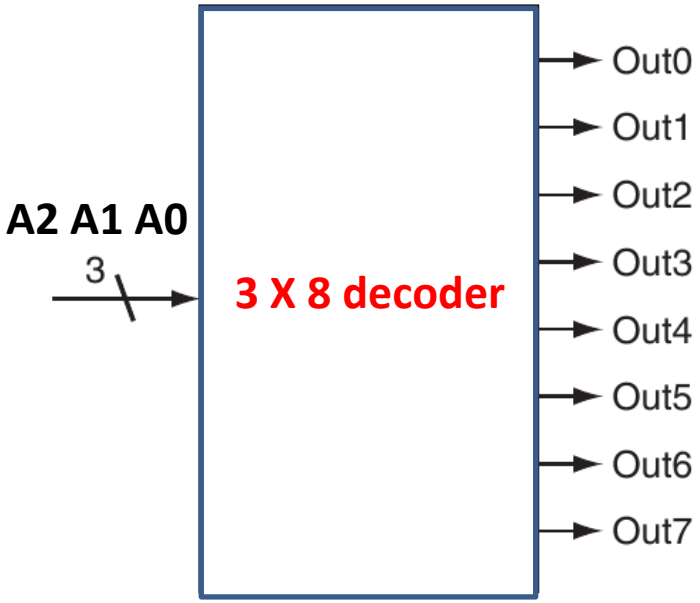
- Solutions: Carry look ahead addition, Carry-save adder (B.6)

https://commons.wikimedia.org/wiki/File:4-bit_ripple_carry_adder-subtracter.svg

Common Circuits

Decoders

- A **decoder** has n-bit input and 2^n outputs
- Only one output active for each input combination



a. A 3-bit decoder

Inputs			Outputs							
A2	A1	A0	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

Encoders

- Opposite of a decoder
  - Output only specified if one input is 1

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Assume only one button can be pressed at any time

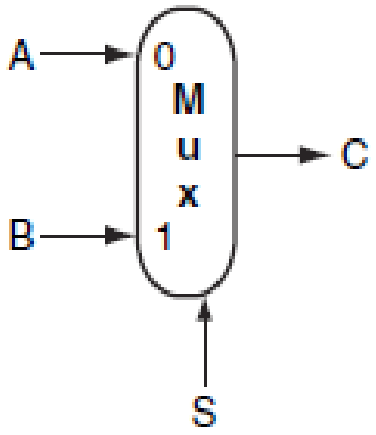
$$A_0 = D_1 + D_3 + D_5 + D_7$$

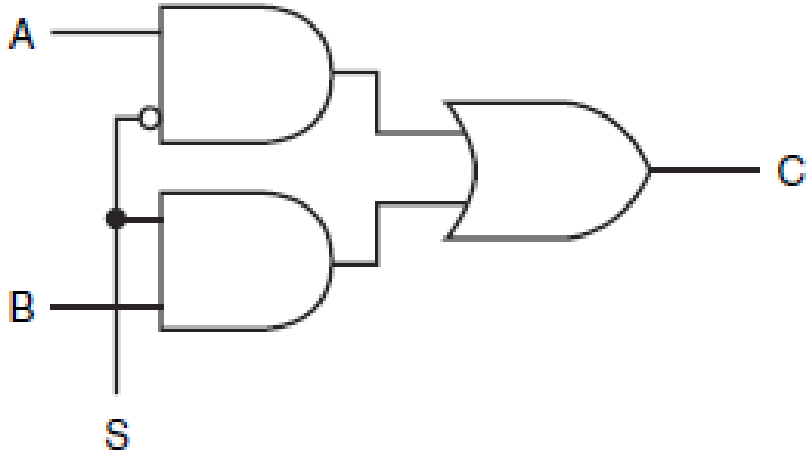
$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

Multiplexor

- The Selector. Use often
  - 2 signals **A**, and **B**, and a selector **S**
  - If **S = 0**, output = **A**
  - Otherwise, output = **B**





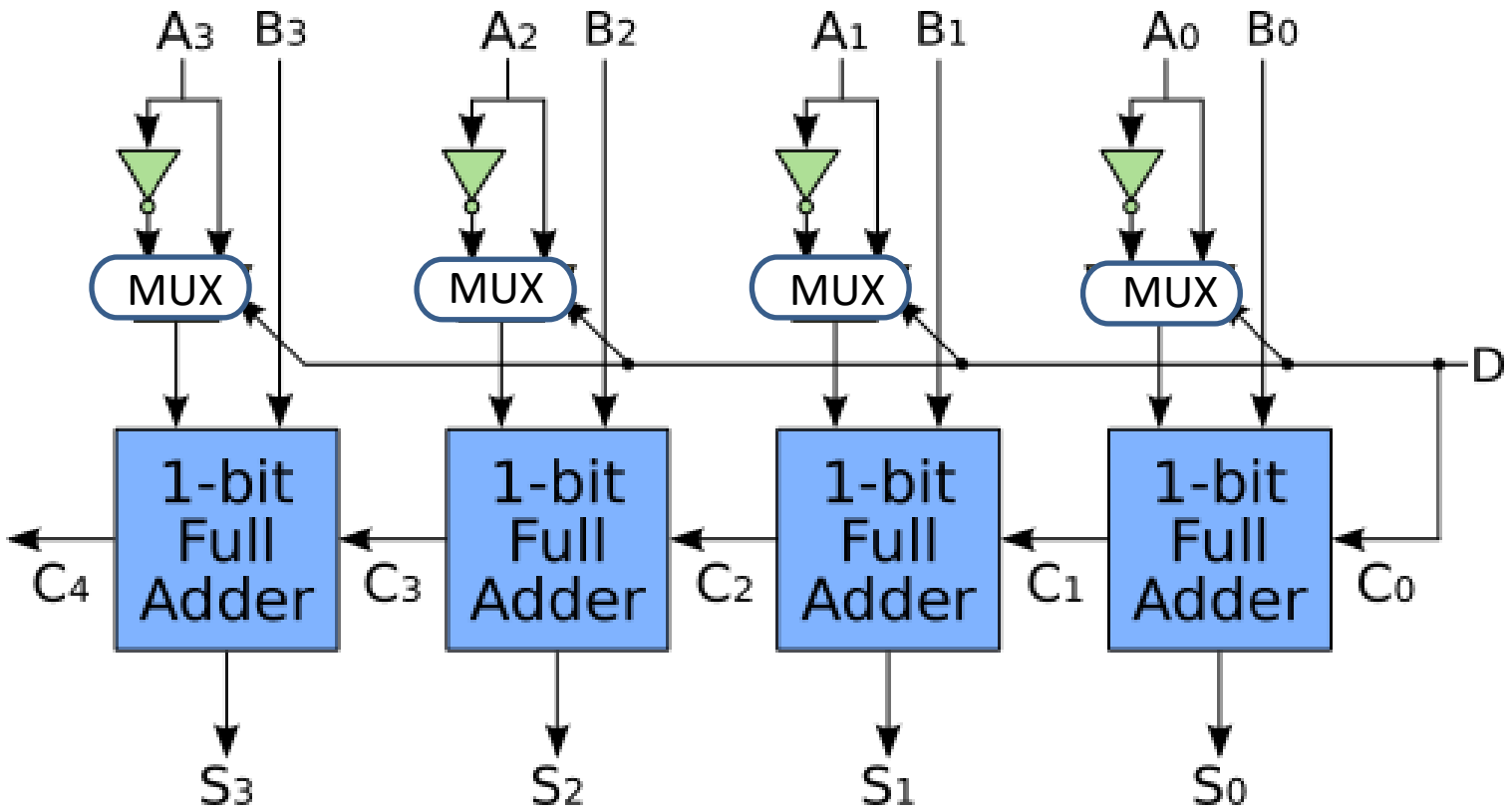
S	A	B	C
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

C = S'A + SB

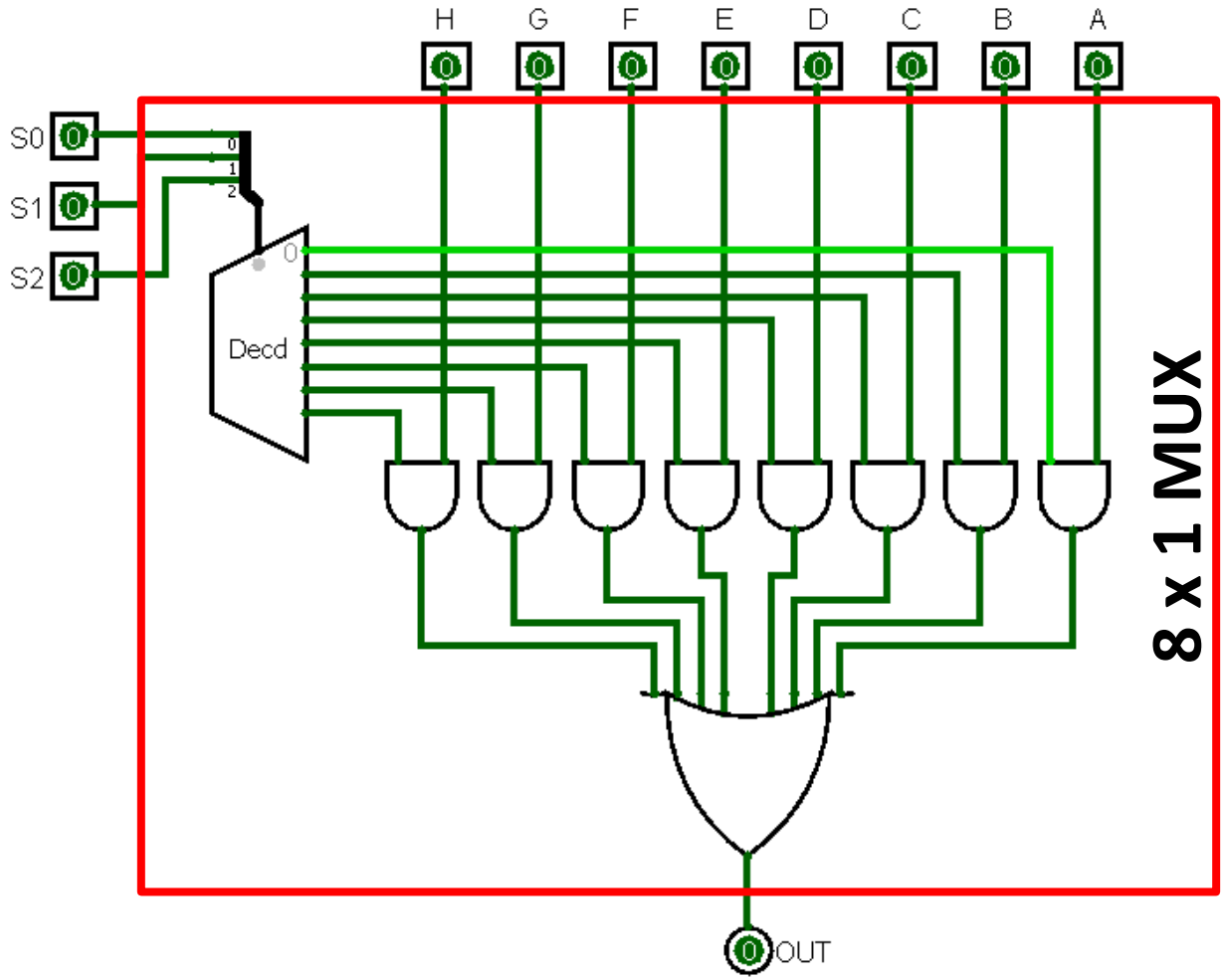
Multiplexor

- Can have any number of inputs
  - Given n signals, need $\lceil \log_2 n \rceil$ selector inputs
- Three step recipe for building a multiplexor
  - A decoder that generates n signals
  - An array of n AND gates to combine the signals
  - A single n input OR gate to compute the result

Full adder circuit



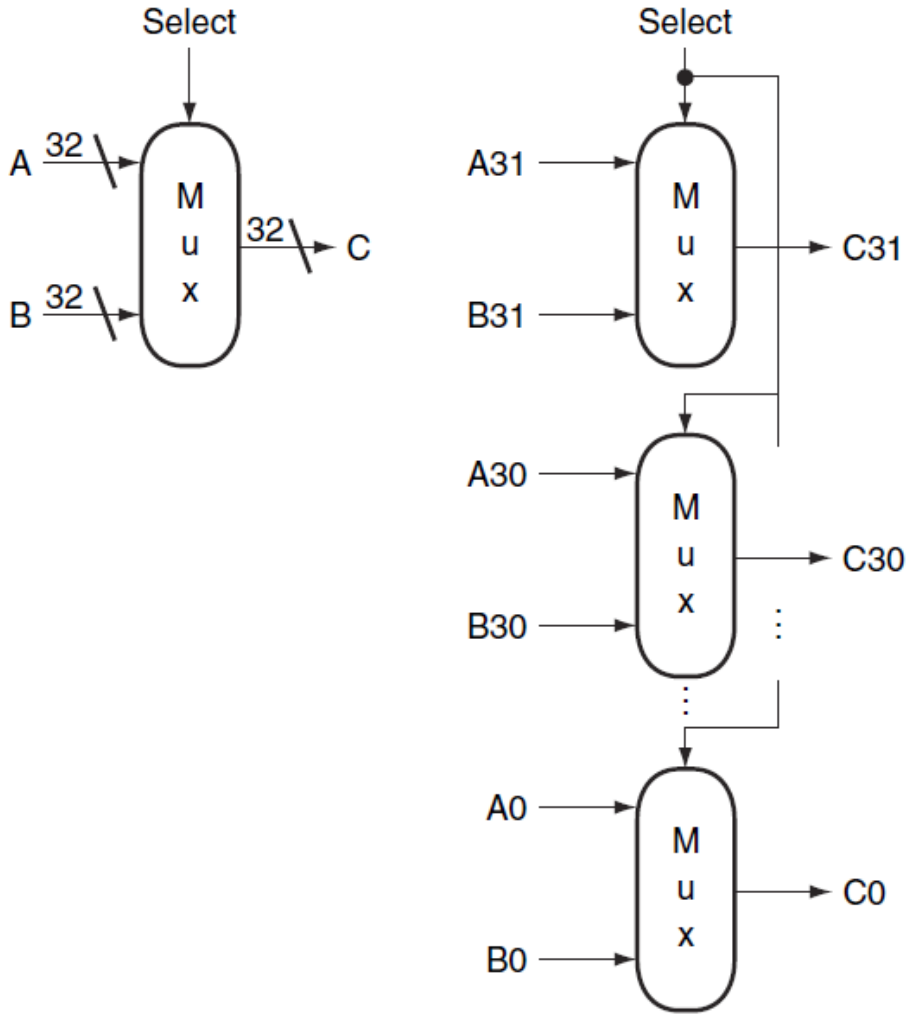
Multiplexor



- Eight different signals (A-H)
- 3-8 Decoder for selecting the inputs

Arrays of Logic Elements

- Many operations need to be done on an entire word (32 bits)
- For instance, a multiplexor that selects one of two words:



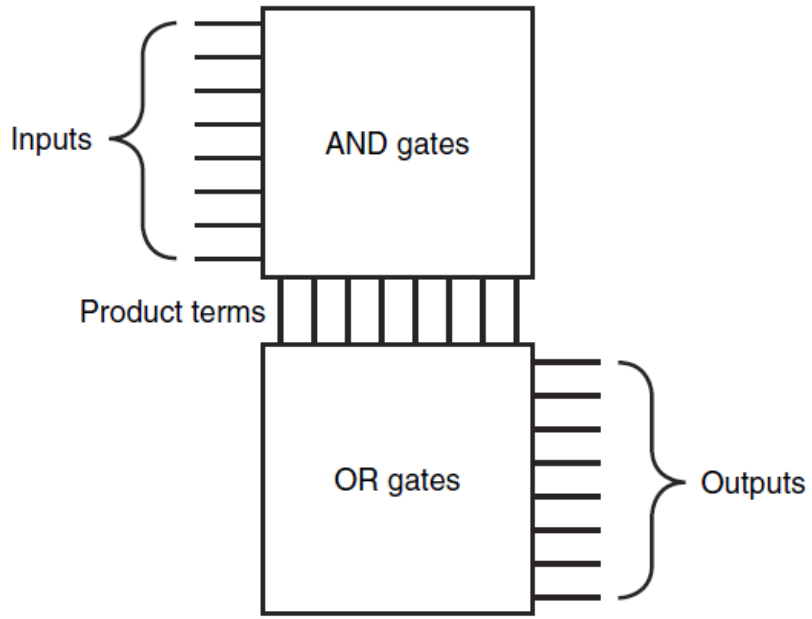
Read-only Memory (ROM)

- Inputs encode the memory address
- Outputs are the contents at each address
- Consider a circuit as a "hardwired" memory

A_1	A_0	b_2	b_1	b_0
0	0	0	1	1
0	1	0	0	1
1	0	0	0	0
1	1	1	0	0

Programmable Logic Array (PLA)

Canonical form: only two levels of gates



- The **AND gates array** is a product term consisting of any number of inputs or inverted inputs
- The **OR gates array** is a sum term consisting of any number of these product terms.

Programmable Logic Array (PLA)

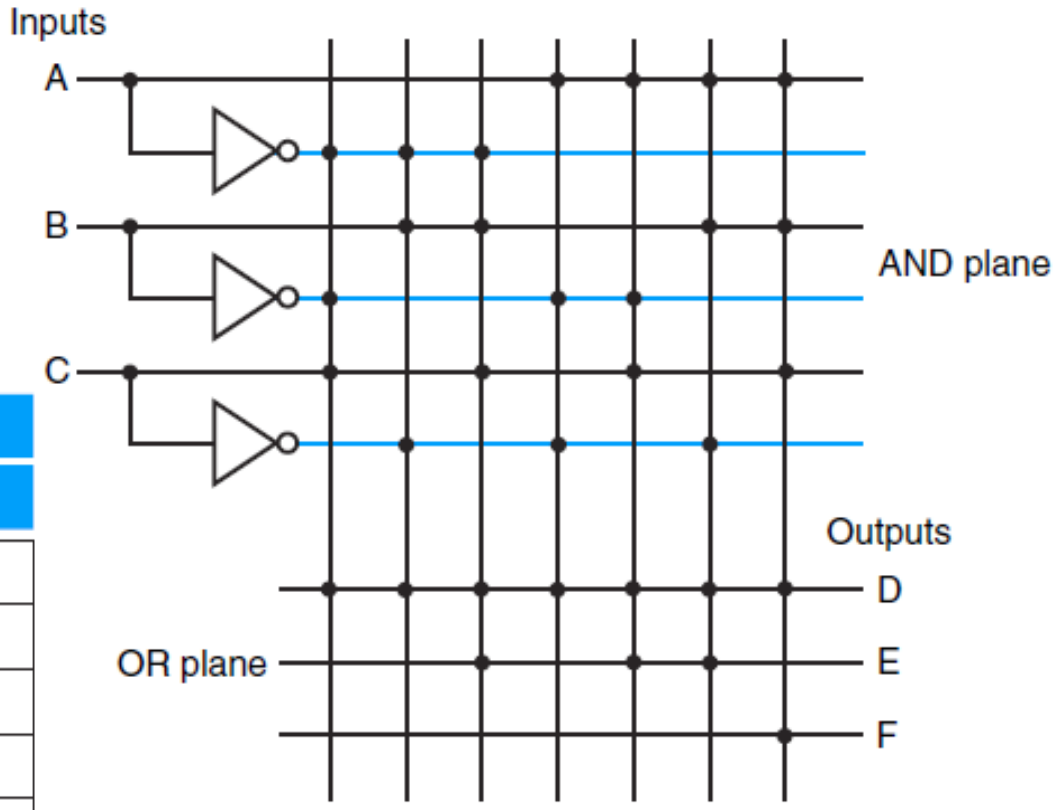
D = A + B + C

D= (A'B'C + A'BC' + ….. + ABC'+ ABC)

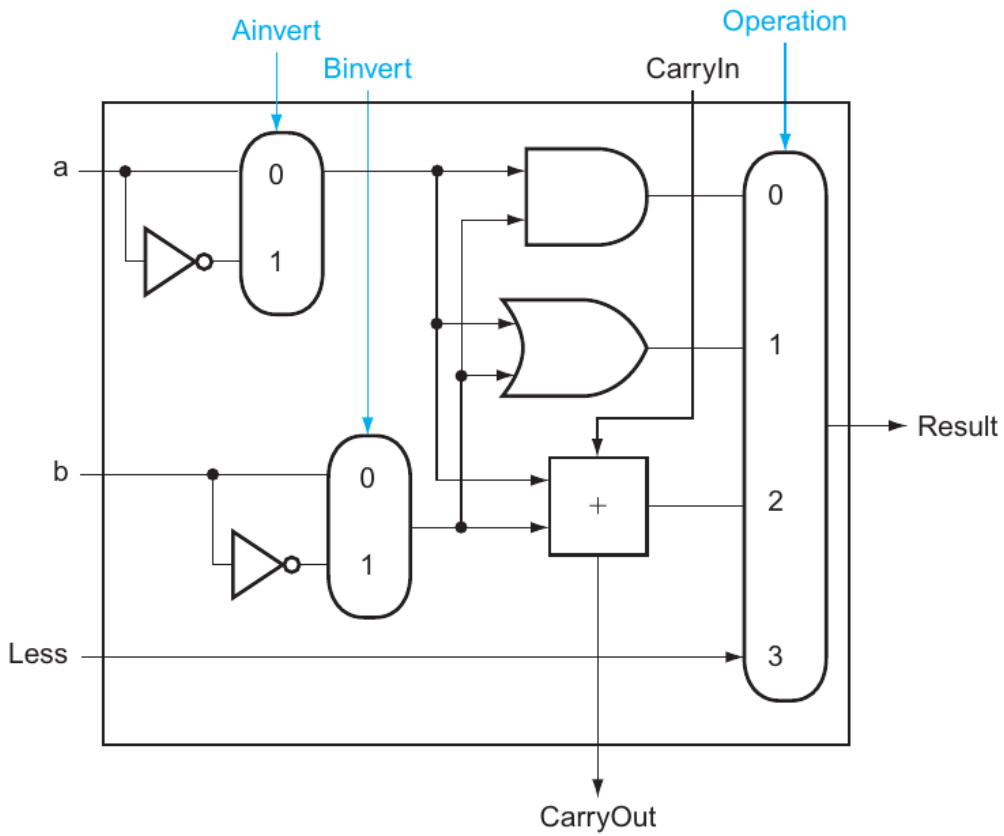
E = (A · B · C') + (A · C · B') + (B · C · A')

F = A · B · C

Inputs			Outputs		
A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1



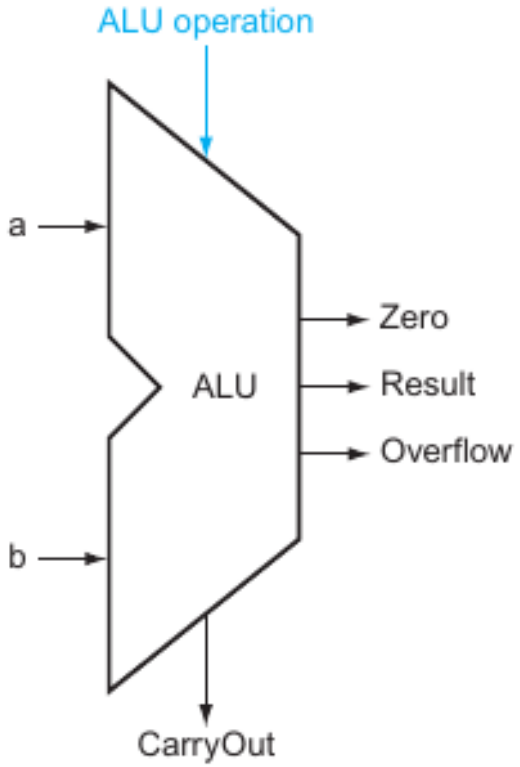
Arithmetic Logic Unit (ALU)



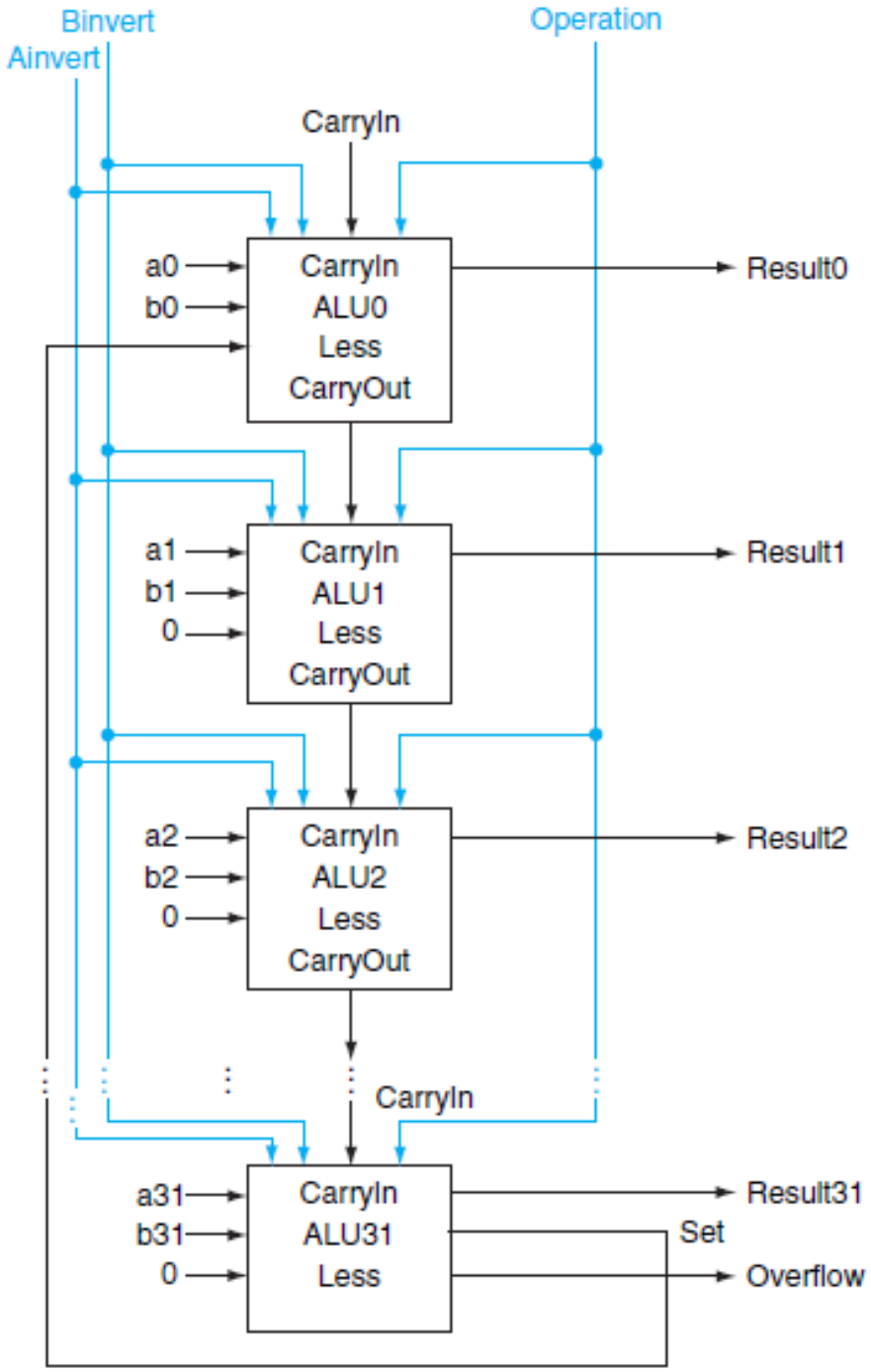
1-Bit ALU

Instruction	Ainvert	Binvert	CarryIn	Operation
ADD	0	0	0	2
SUB	0	1	1	2
AND	0	0	x	0
OR	0	0	x	1
NOR	1	1	x	0
SLT	0	1	1	3

Arithmetic Logic Unit (ALU)



32-Bit ALU



Review

- Adder
- Common circuits
  - Decoder and Encoder
  - Multiplexor
  - Read-only Memory (ROM)
  - Programmable Logic Array (PLA)
  - Arithmetic Logic Unit (ALU)

- Textbook: Appendix B.8 of 5th and 6th edition

Sequential Circuits

Agenda

- RS Latch
- D Latch
- D Flip flop
- T Flip flop

Sequential Circuits

- **Combinatorial circuits** have no memory
  - Output is simply a function of inputs
- **Sequential circuits** contain "state"
  - Combinatorial circuits + memory
  - The mechanism for remembering information (i.e., bits) inside the CPU and in the main memory.
  - How to remember things?
    - Write it down
    - Repeat it to yourself (sequential circuits style)

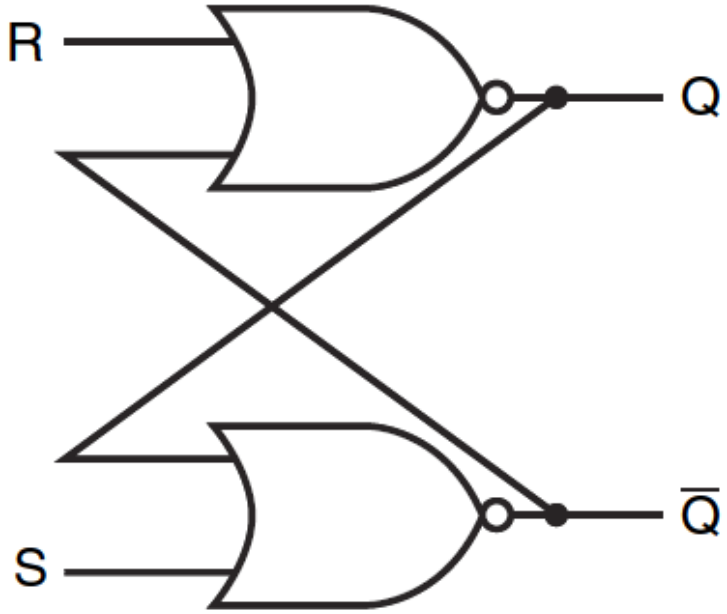
Latch

RS flip-flop

- Two inputs S and R
  - S = 1, **set** output to 1
  - R = 1, **reset** output to 0
  - S, R are both 0, **hold** the values

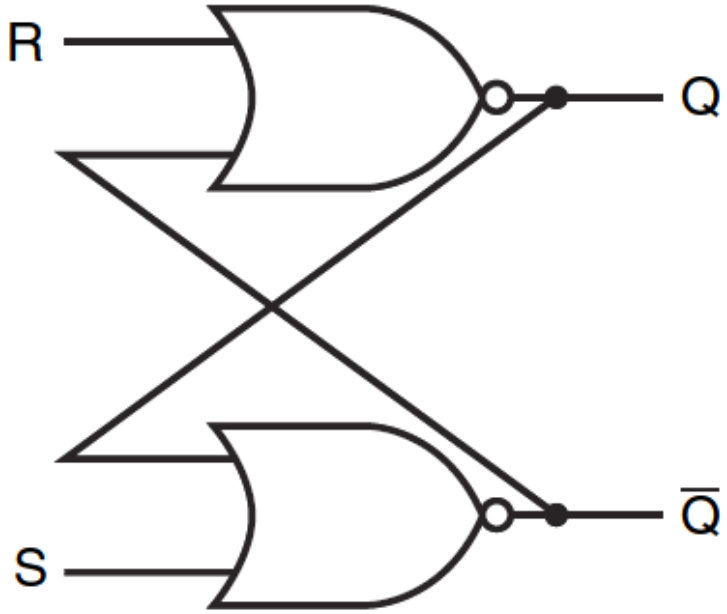
A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

R	S	Q	Q'
0	0	Q	Q'
0	1	1	0
1	0	0	1



RS flip-flop

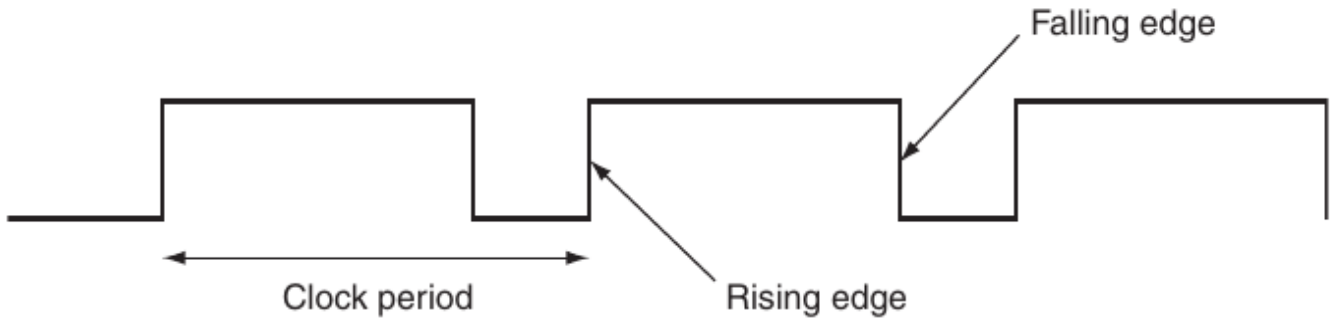
- Problem:
  - When S, R are both 1, **invalid**



R	S	Q	Q'
0	0	Q	Q'
0	1	1	0
1	0	0	1
1	1	invalid	invalid

Clocks

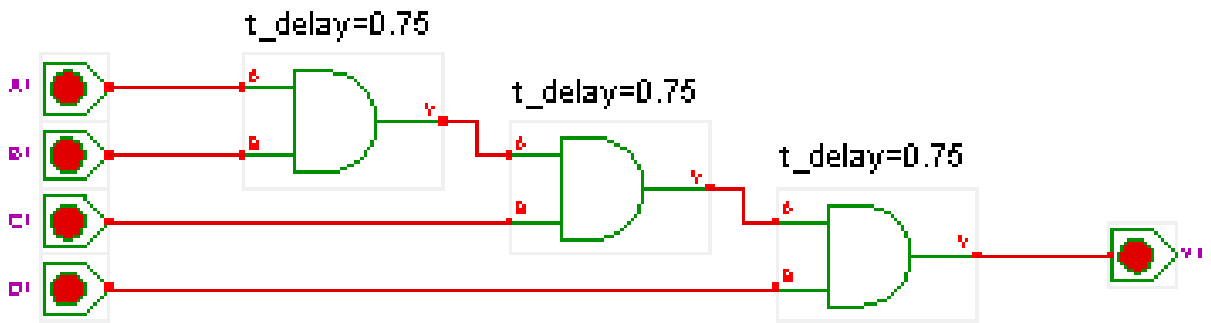
- Mechanism for synchronizing inputs and outputs.
- Metronome
- Clock speed is in Gigahertz



https://bestanimations.com/gifs/Black-And-White-Old-Metronome.html

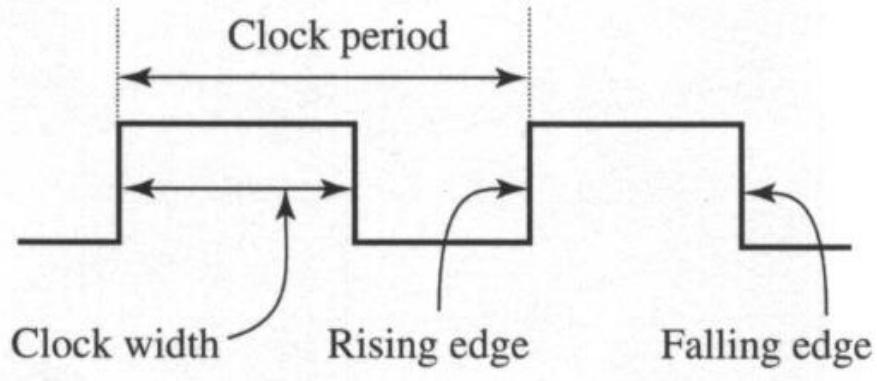
Clocks

- Recall that it takes time for combinatorial circuits to produce results.
  - e.g., carry ripple in adder



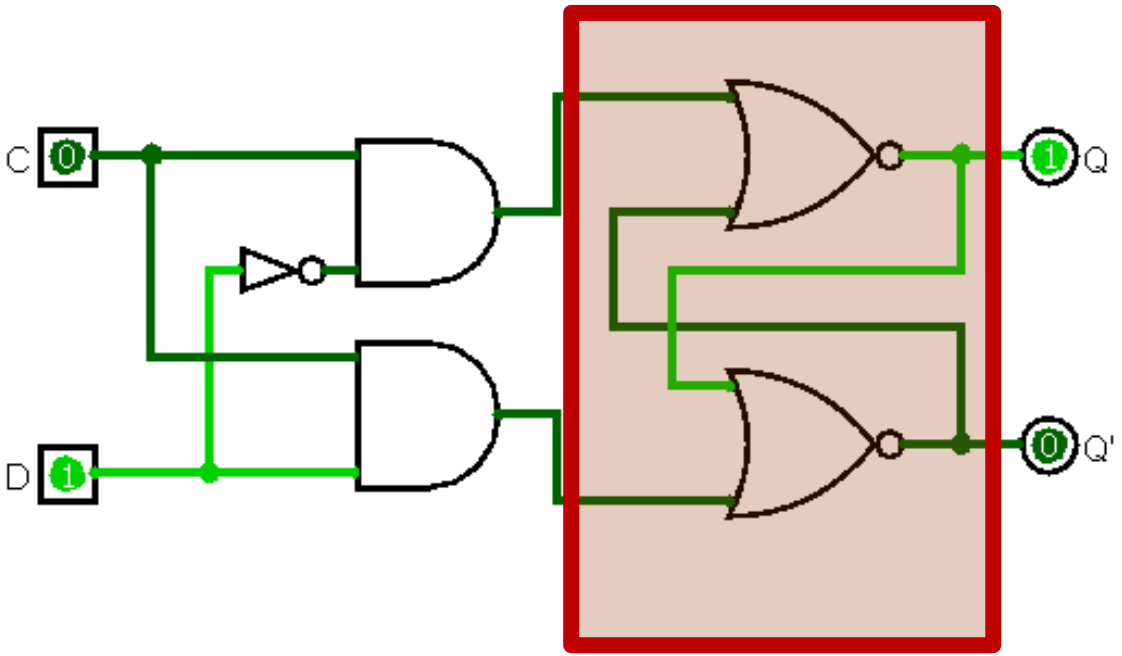
http://goo.gl/n3fKv

- Want results to be finished before we store the result somewhere



D Latch

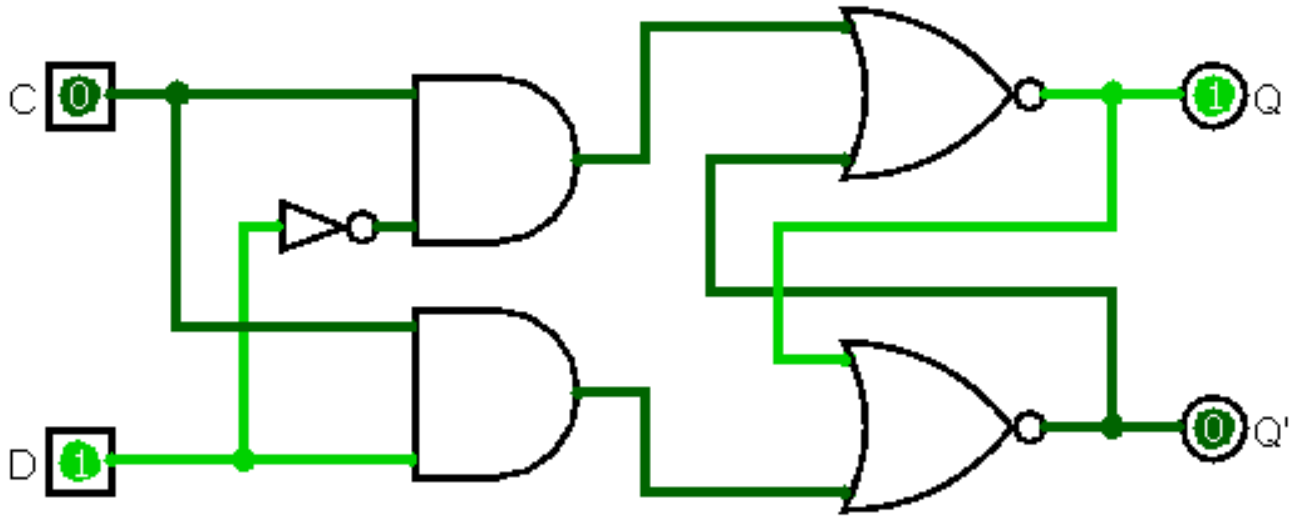
- D for “Data”
- Q records state of D (holds previous value of D while C = 0)



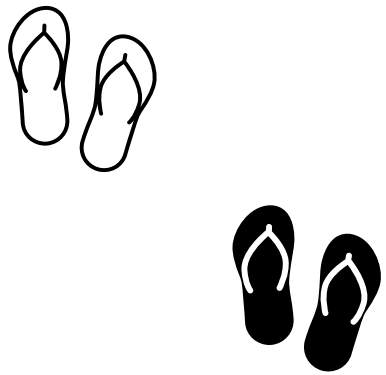
SR Latch

D Latch

- When C = 0, hold values of Q and Q'
- When C = 1, D = 1, then set Q = 1, Q' = 0
- When C = 1, D = 0, then reset Q = 0, Q' = 1
- Impossible for R and S to be both 1

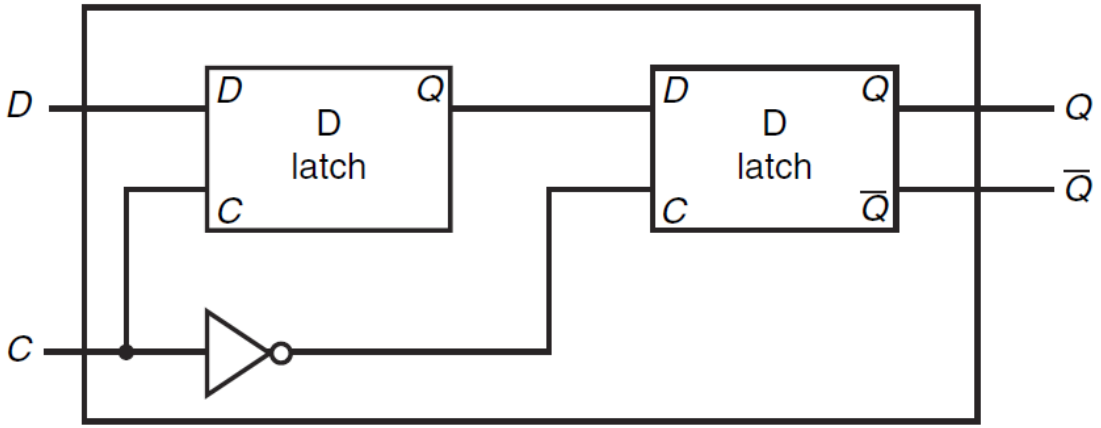


Flip-flop



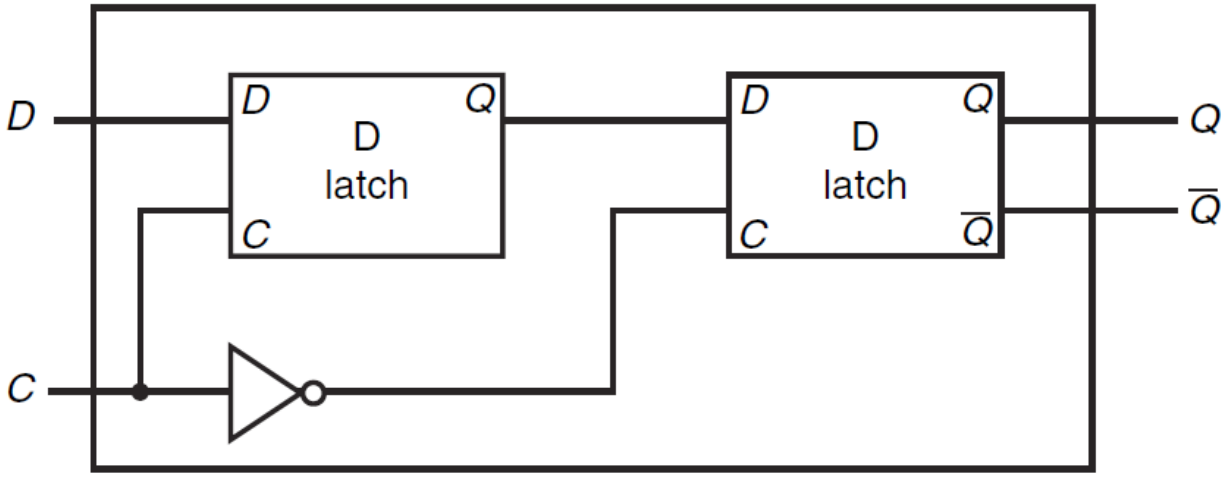
D flip-flop

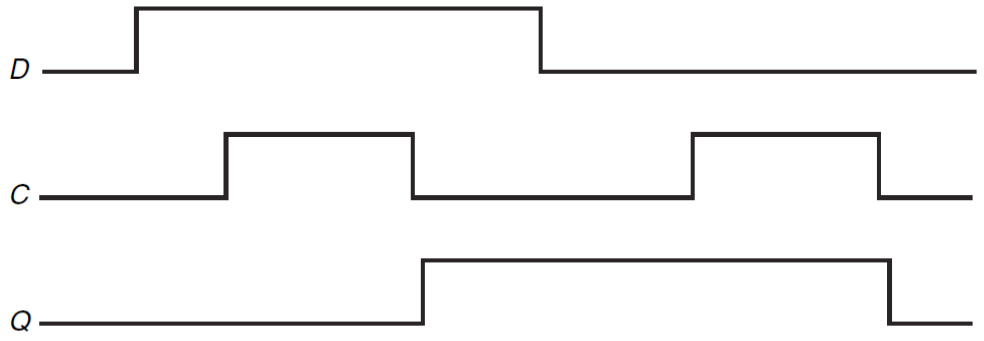
- **Two D latches**. One for reading and one for writing
- Mechanism prevents data from being read/write at the same time



- By putting an inverter, change when the **falling edge** is triggered

D flip-flop

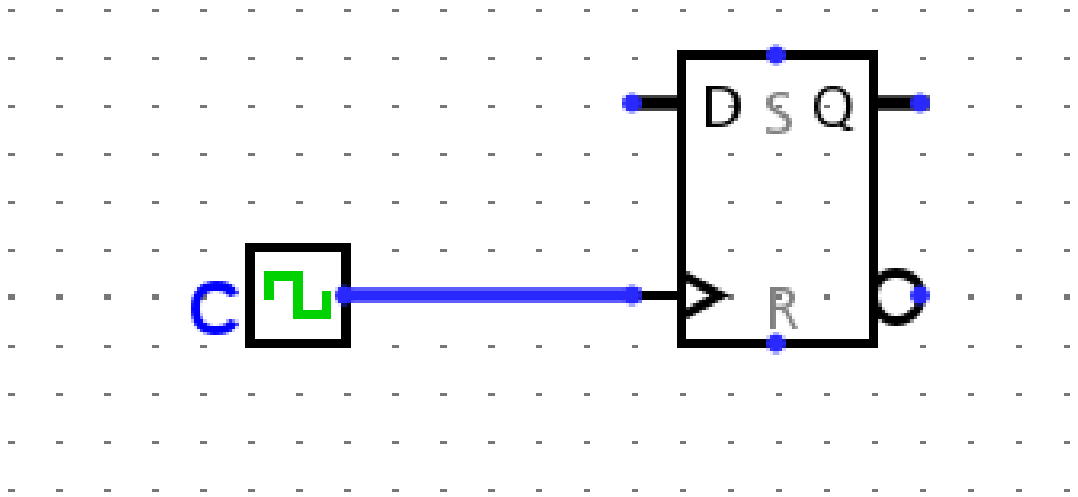




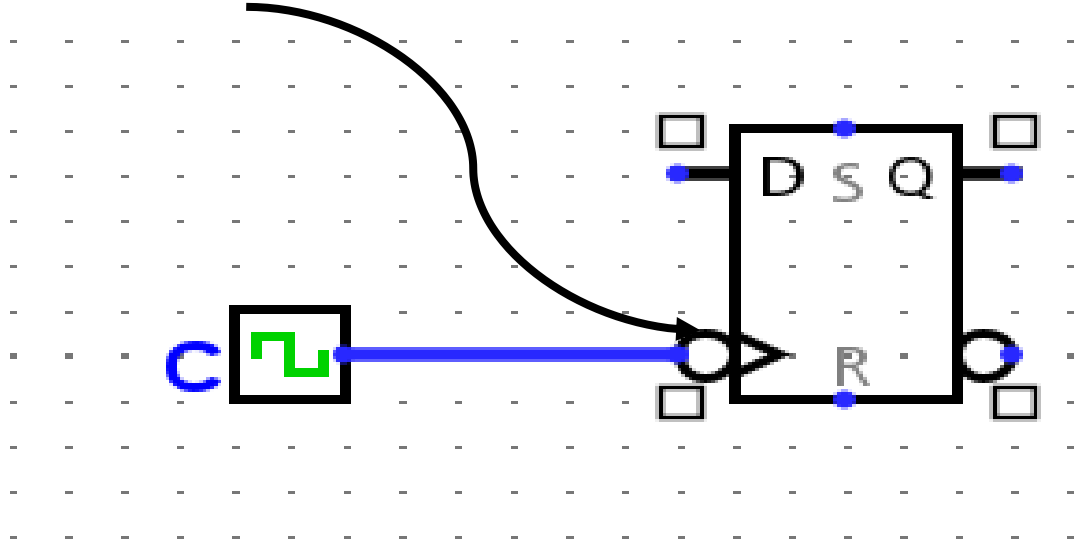
- When C = 1
  - write D into the first D latch
  - Q does not change
- When C = 0
  - Stop writing D into the first D-latch.
  - The D value from the first D-latch is written into the second D-latch

D flip-flop

Rising edge D flip-flop

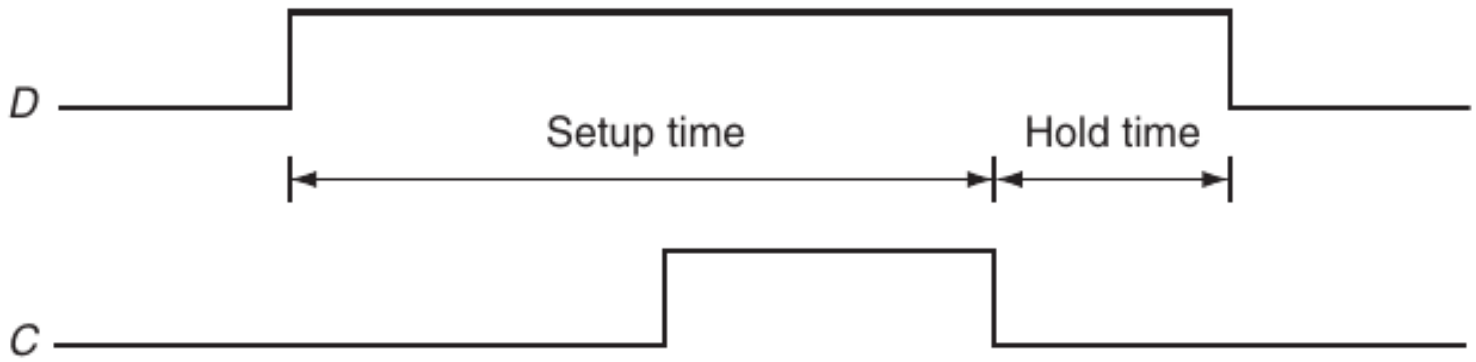


Falling edge D flip-flop (small circle)



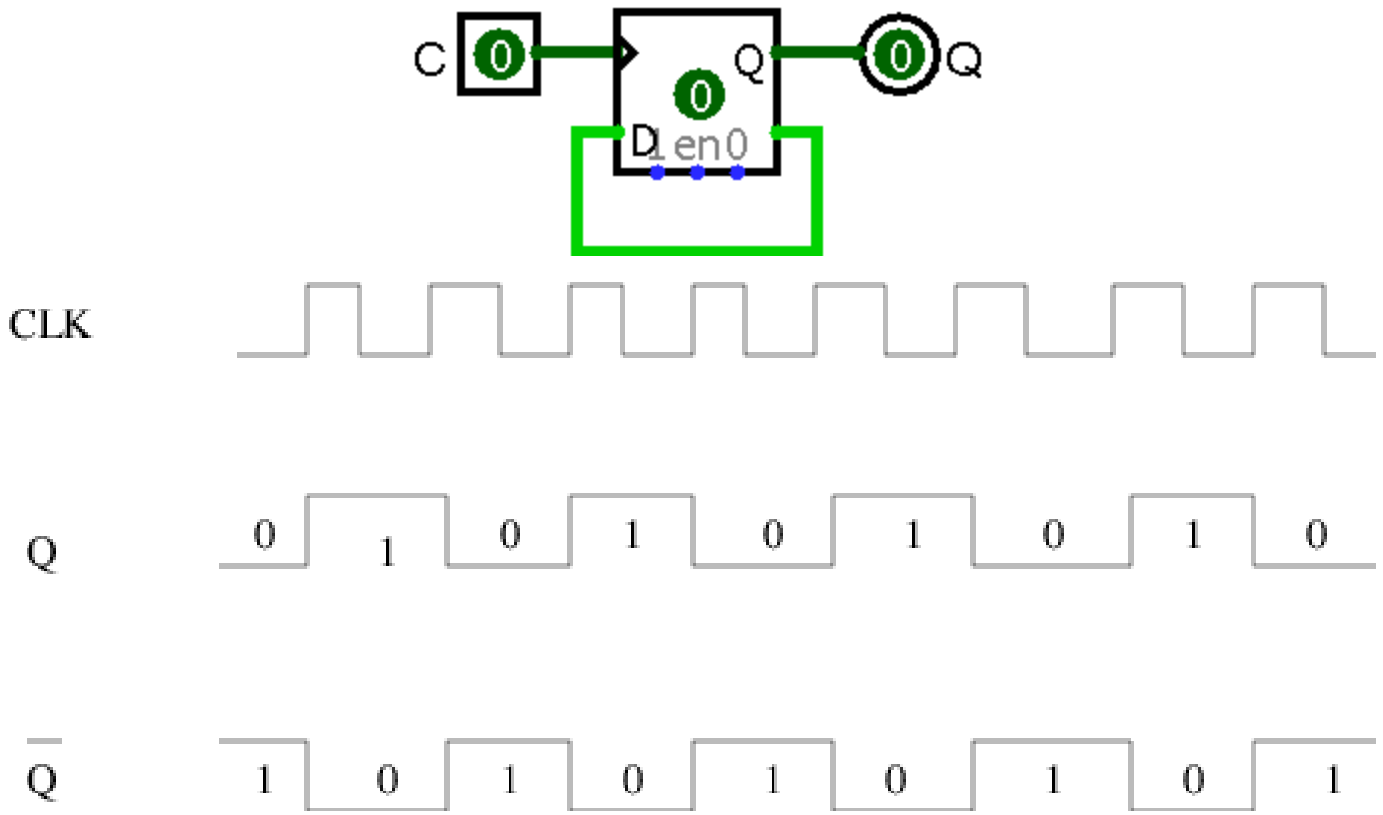
D flip-flop Setup and Hold

- **Setup time**: Input must be stable before the falling clock edge
- **Hold time**: Input must also be stable after clock edge
  - Hold times typically *very small*, or zero



Toggle Flip Flop

- Hook Q' up to D, and clock becomes a **toggle**



Review and more information

- RS Latches
- D Latch
- D Flip flop
- T Flip flop

- Textbook: Appendix B.7 and B.8 of 5th and 6th edition
- Next topic: Registers (use sequential circuits)