INF-1100
Digital logic structures

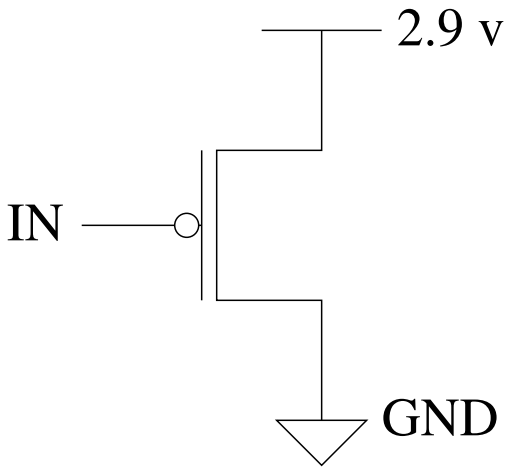
Åge Kvalnes

University of Tromsø, Norway

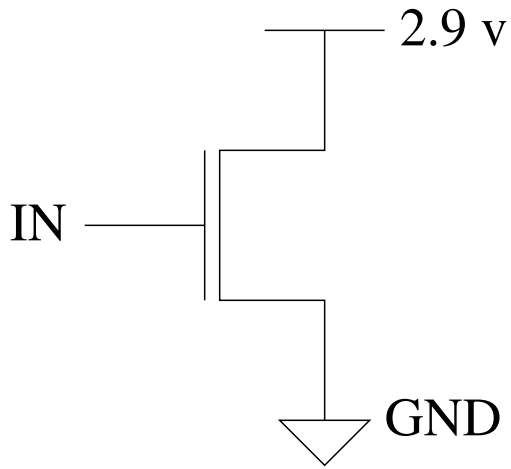
September 4, 2014

The transistor

A transistor is a semiconductor component that acts like a switch, controlling the flow of an electric current.



(a) **p-type: conducts when IN=0**

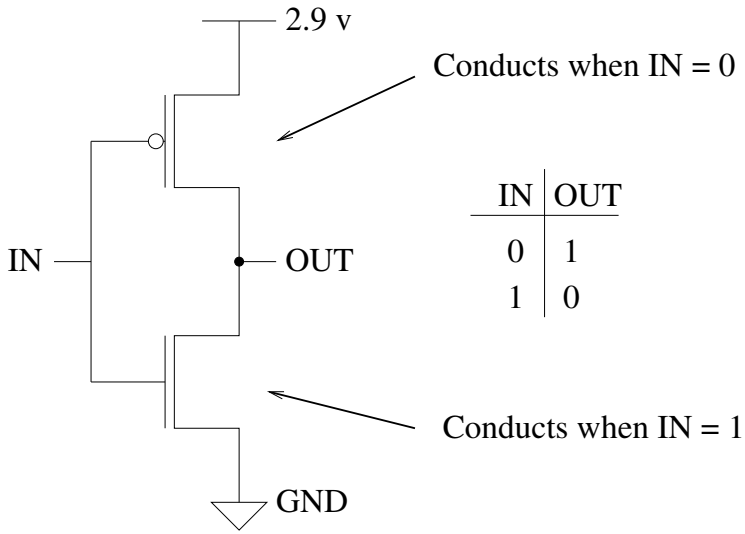


(b) **n-type: conducts when IN=1**

Think of IN as a binary digit. The value 0 is a voltage low enough to keep the p-type transistor open. The value 1 is a voltage high enough to keep the n-type transistor open. And vice versa.

Logic gates: NOT gate

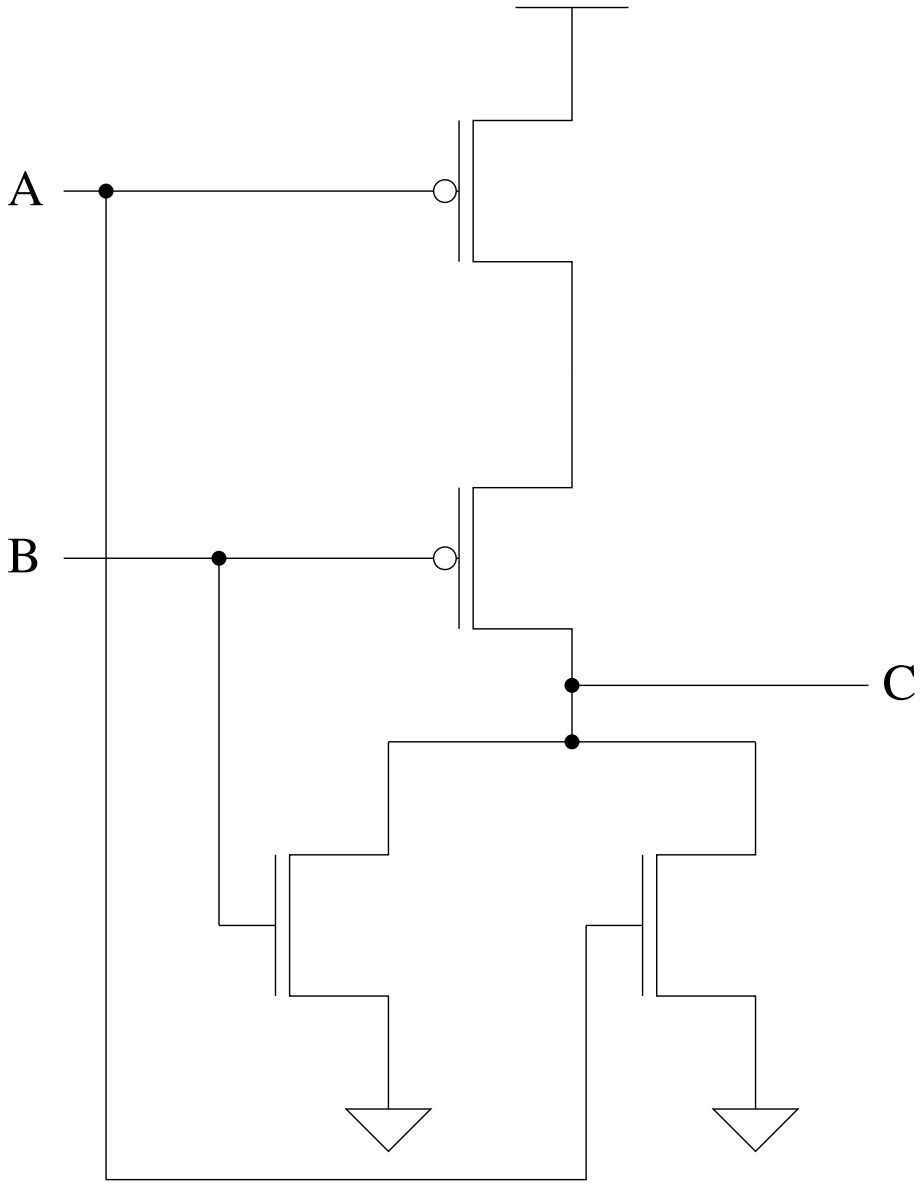
Combine a p-type and an n-type transistor to invert a binary digit. 1 becomes 0 and 0 becomes 1.



IN	OUT
0	1
1	0

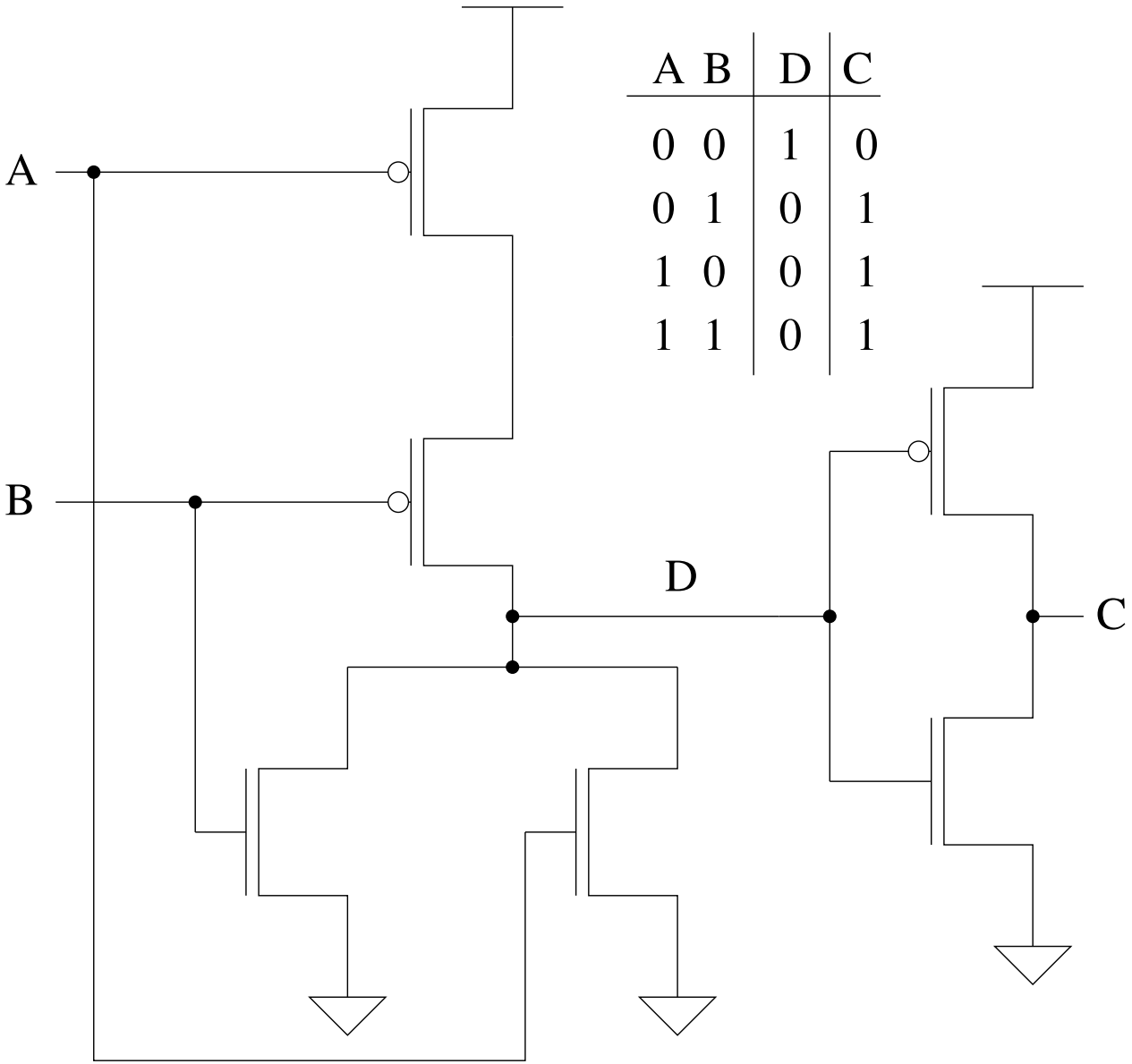
IN represents the binary digit. Depending on the value of IN, OUT is either GND (value 0) or 2.9v (value 1).

Logic gates: NOR gate



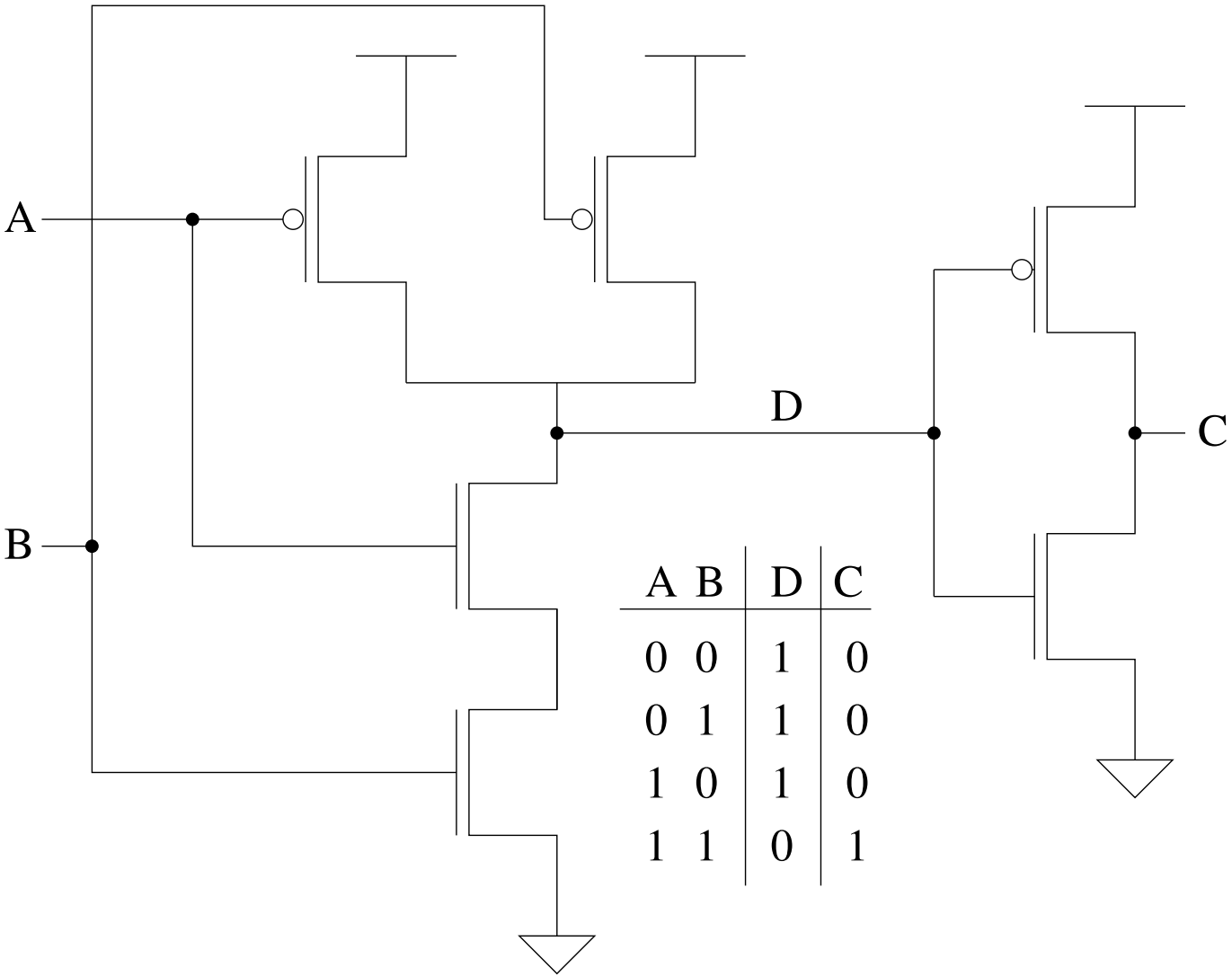
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

Logic gates: OR gate, or NOR-NOT gate



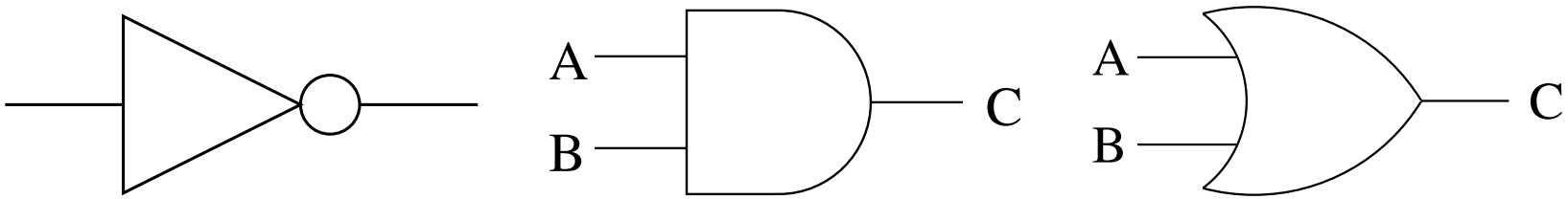
A	B	D	C
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

Logic gates: AND gate



A	B	D	C
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Raising the abstraction level



(a) **NOT gate** (b) **AND gate** (c) **OR gate**

a) Inverts 0 to 1, and 1 to 0.

b) C is 1 only if A **and** B are 1.

c) C is 1 if either A **or** B are 1.

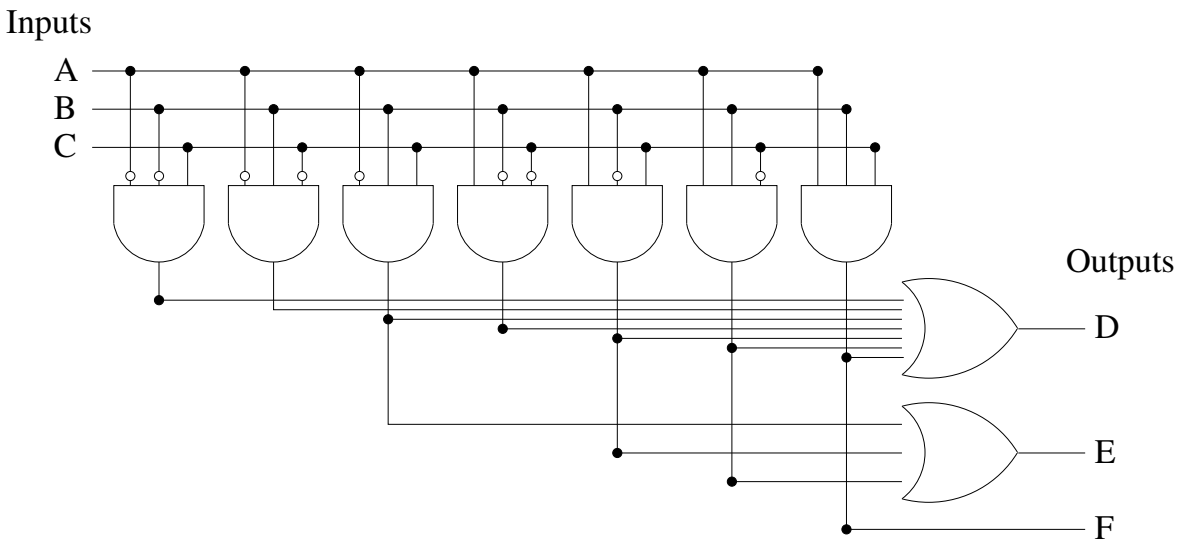
Logical completeness

Any truth table can be implemented using AND, OR, NOT gates.

How do we implement this truth table?

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

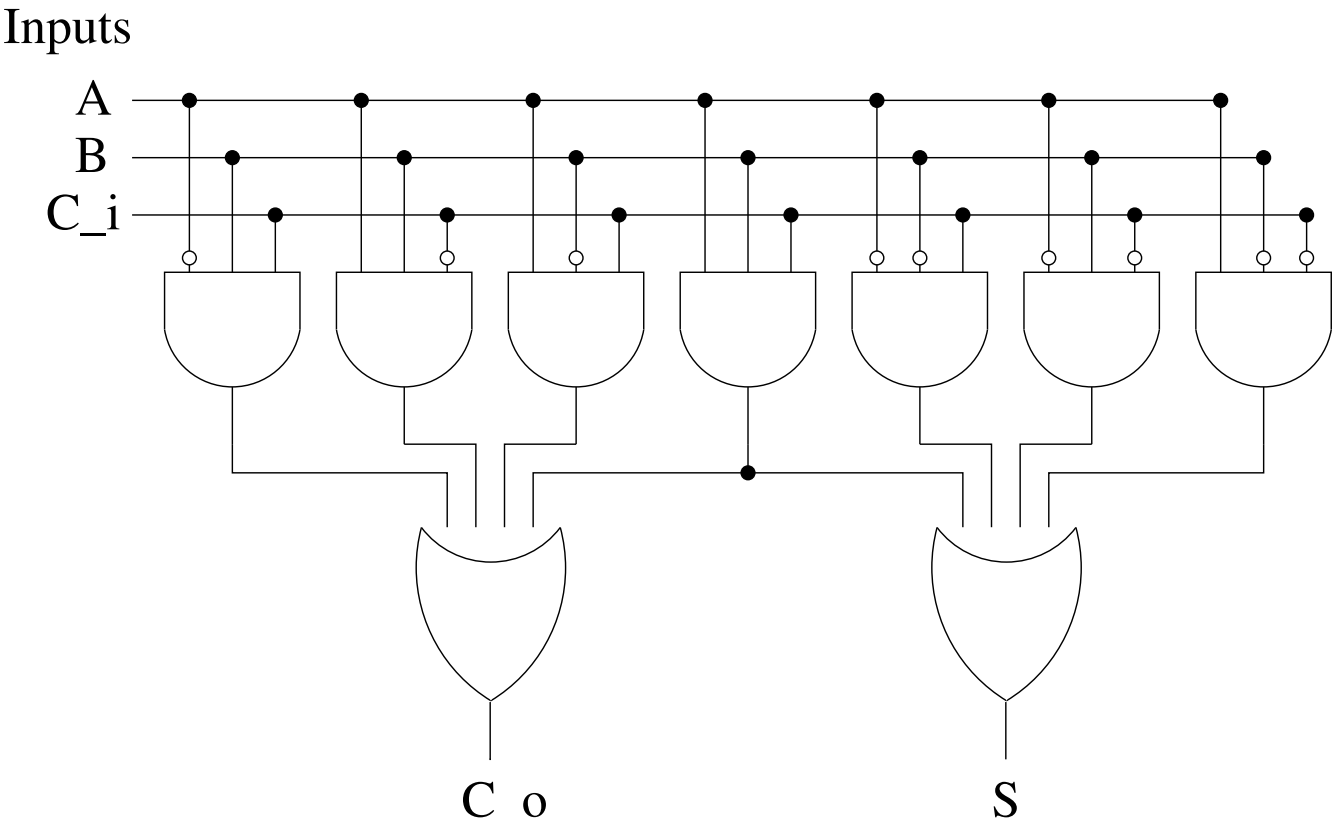
Implementing a truth table



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

AND terms that yield 1 in the truth table, then OR the results together.

Building functions from logic gates: Adding with carry

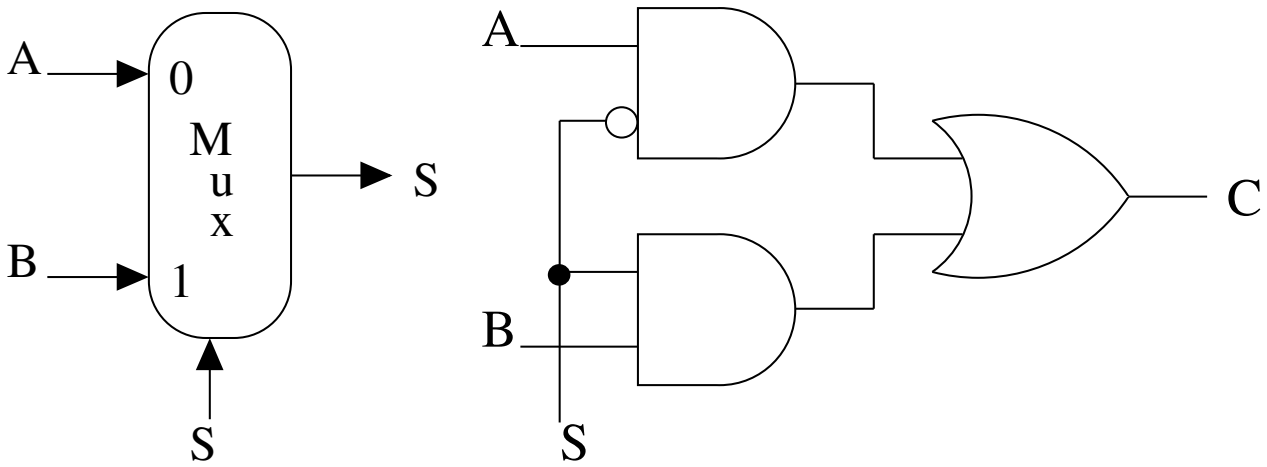


A	B	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Sum A, B, and carry in. Produce sum and carry out.

Building functions from logic gates: The Multiplexor

The output of a multiplexor is one of the inputs. A control signal is used to select which input that should be the output.



1. If $S = 0$, output = A.
2. If $S = 1$, output = B.

Combinational and sequential logic

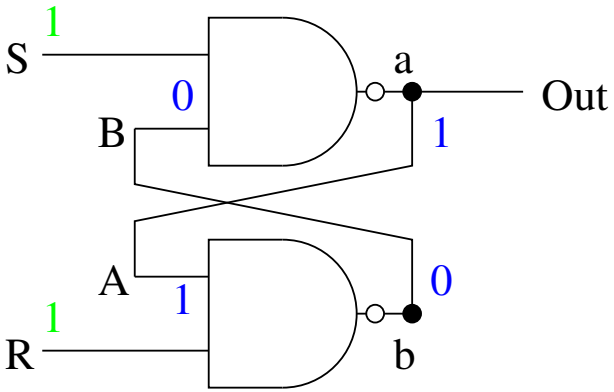
Combinational logic:

- Output depends **only** on the current input.
- Stateless

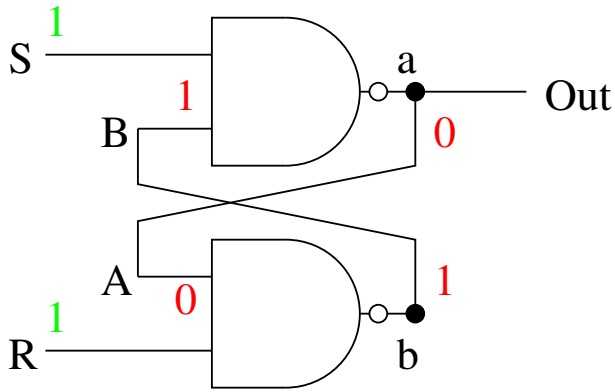
Sequential logic:

- Output depends on the current input **plus** previously stored state.
- Stateful

Stateful logic: The R-S Latch



(f) **Storing the value 1**

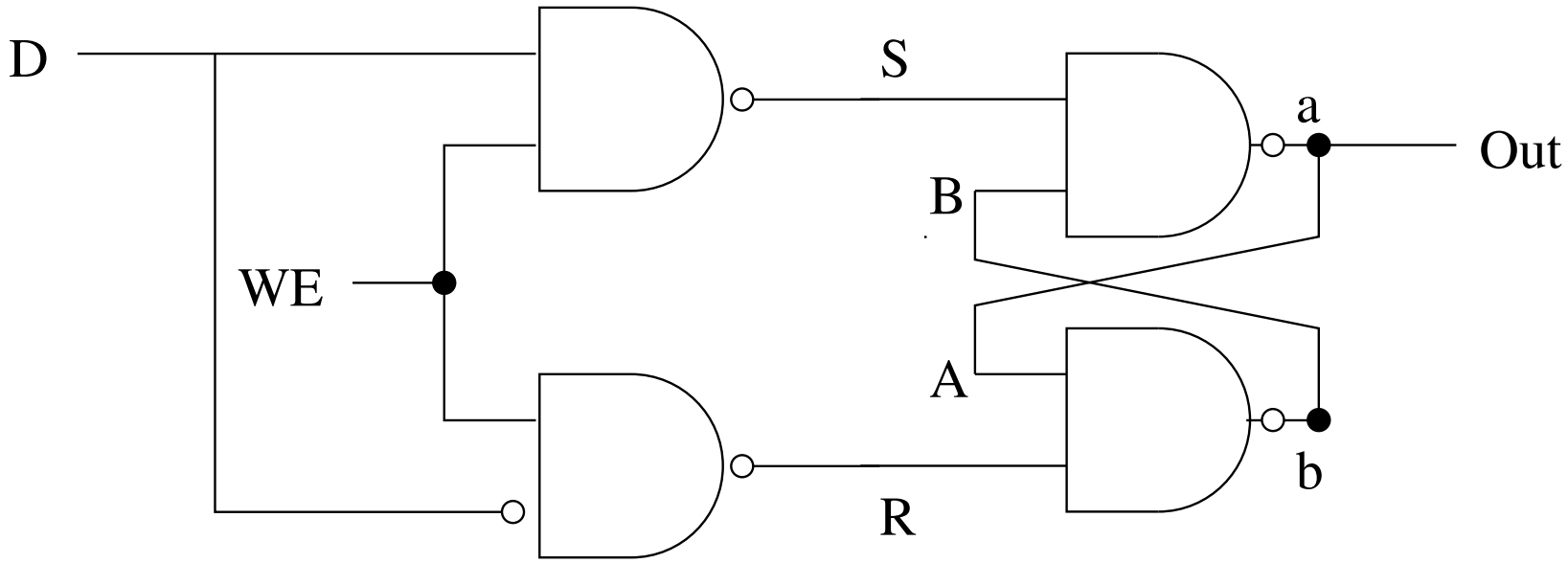


(g) **Storing the value 0**

- The latch can be set to 1 by momentarily setting S to 0 (R must be 1).
- The latch can be set to 0 by momentarily setting R to 0 (S must be 1).

If S and R are set to 0 at the same time, Out is undefined.

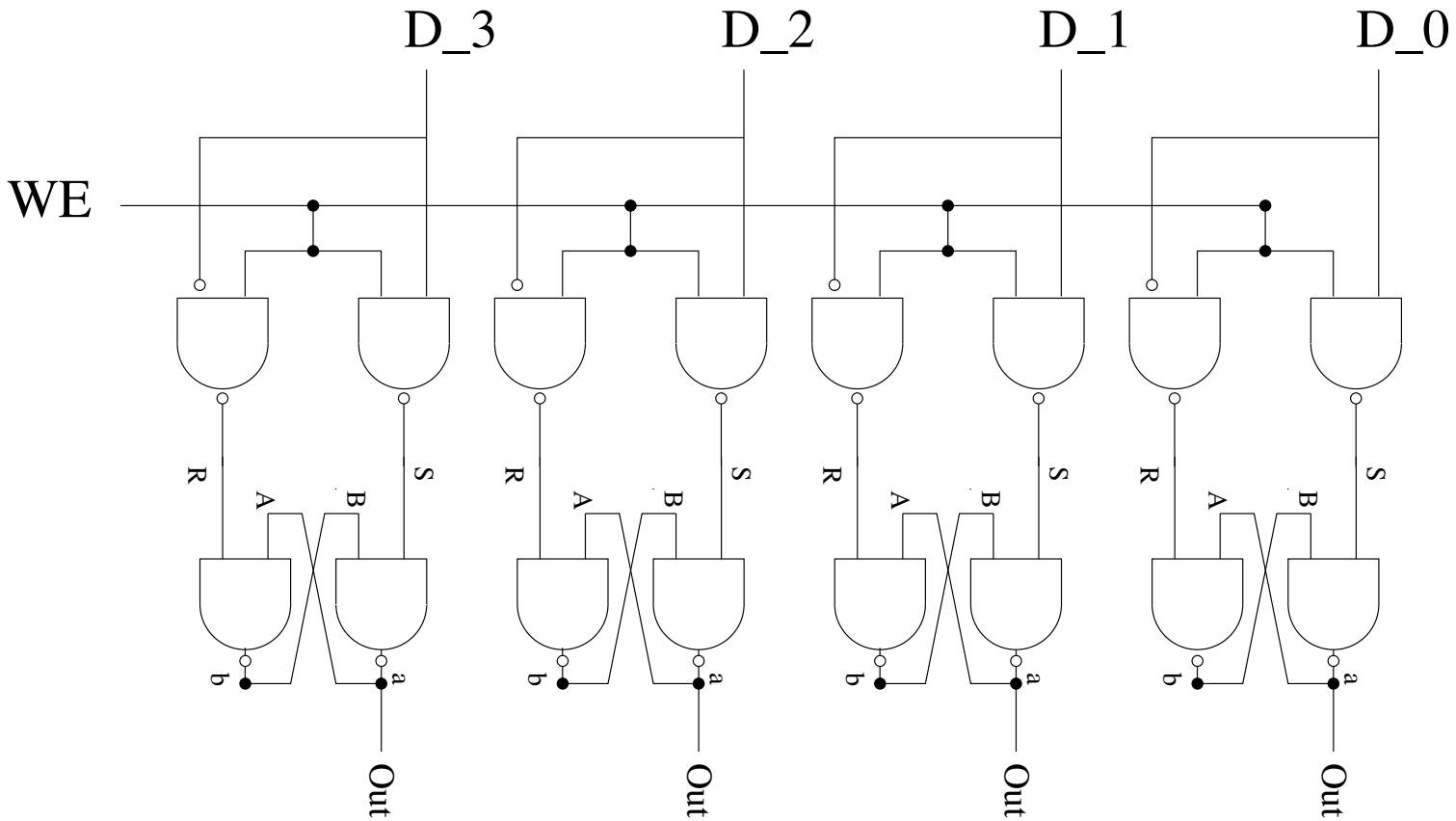
Stateful logic: The Gated D Latch



Controlling when to set or clear an R-S latch.

- If WE (write enable) is 1, latch assumes value of D.
- If WE is 0, latch holds previous value.

Stateful logic: The Register



D latches chained together to form a 4-bit register. WE is shared between all latches.

Stateful logic: Memory

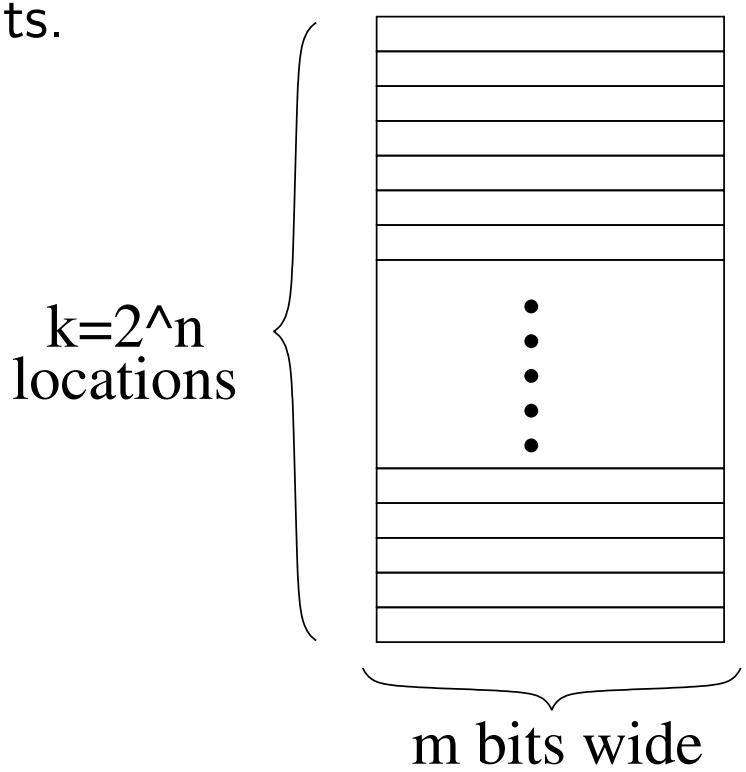
Memory is essentially a $k * m$ matrix of bits.

Address space:

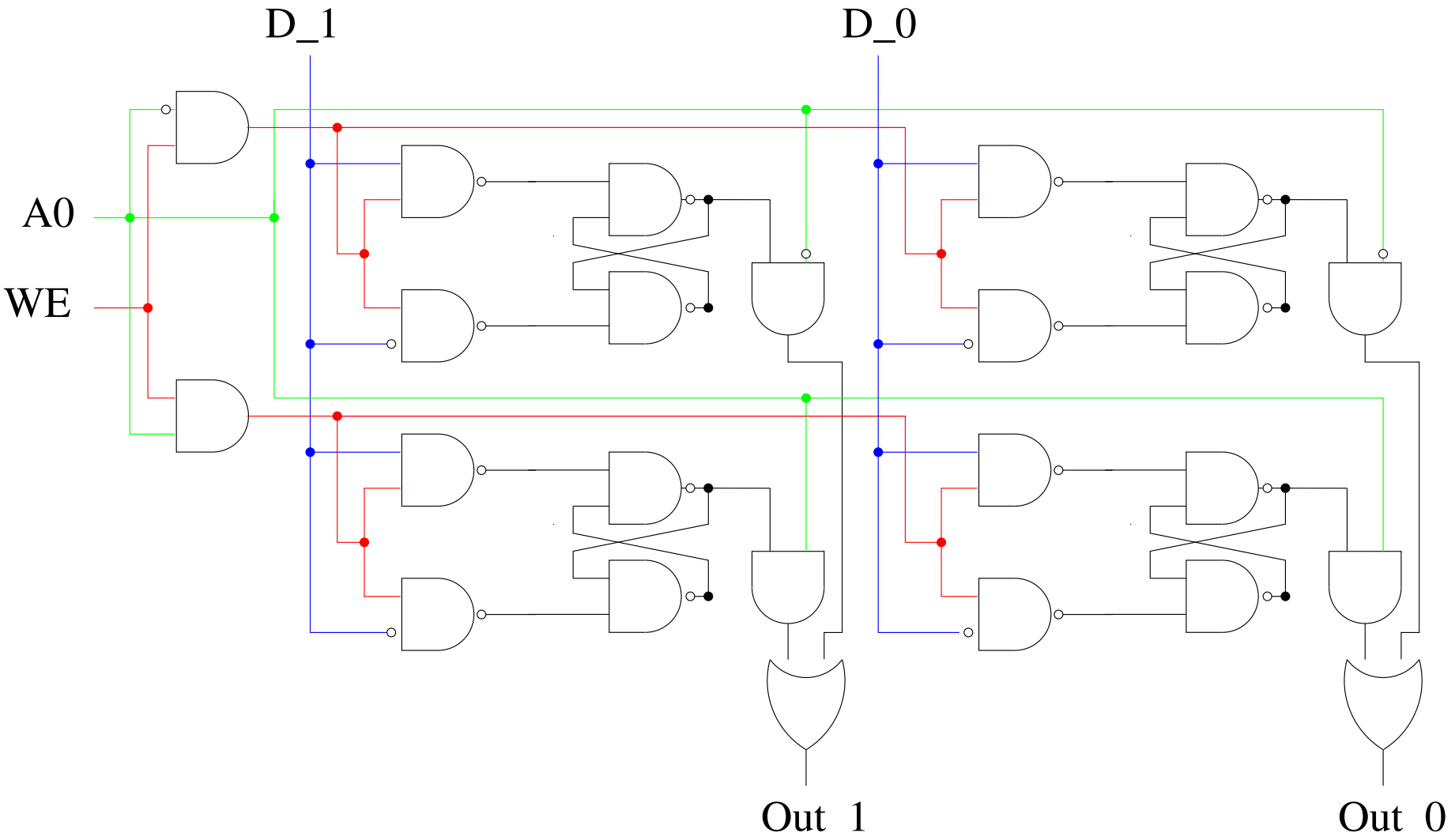
- Number of locations (k).
- Usually a power of 2.

Addressability:

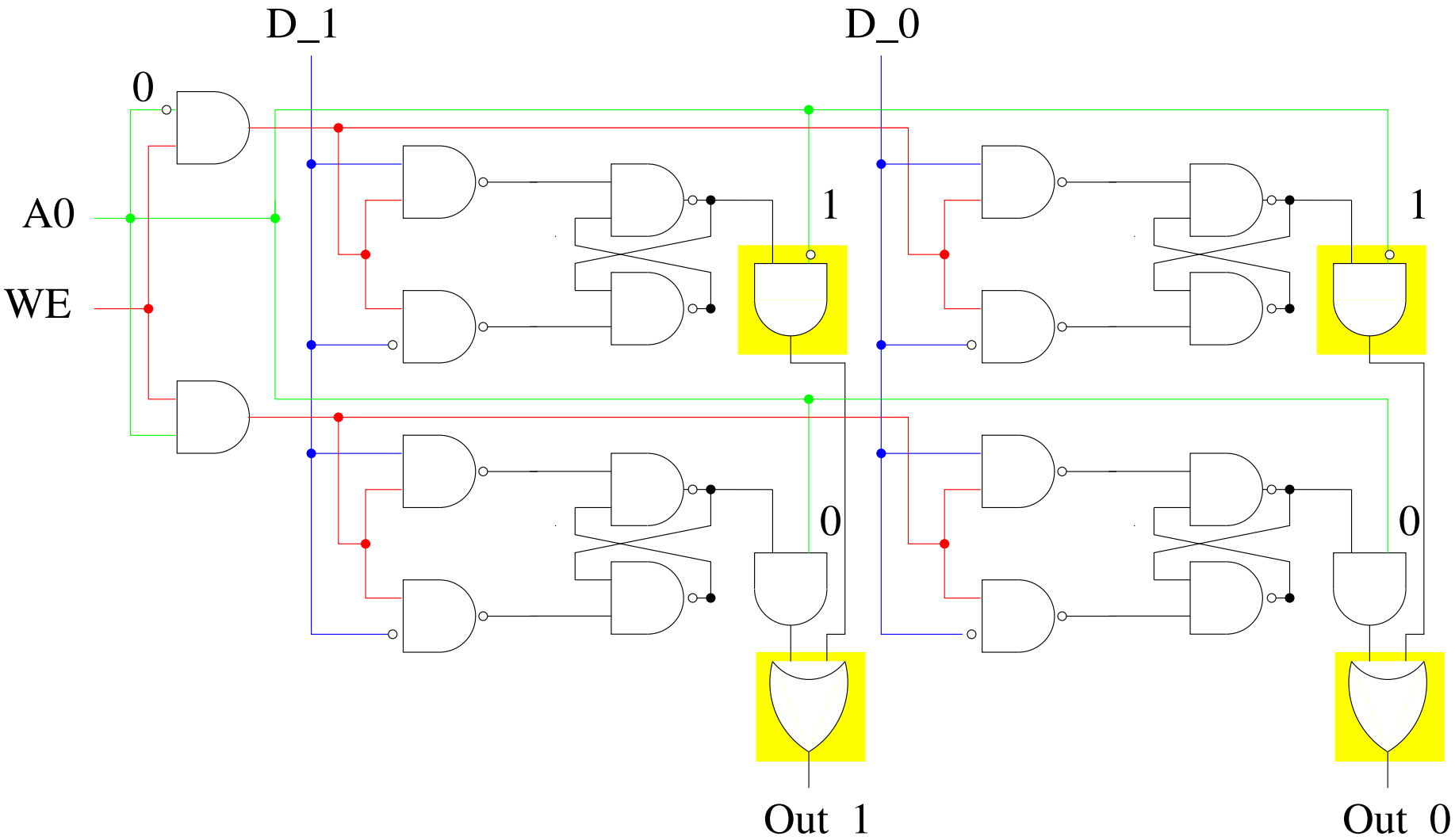
- Number of bits per location.
- Usually 8 bit (a byte).



Stateful logic: A 2x2 bit memory



Stateful logic: Reading from address 0

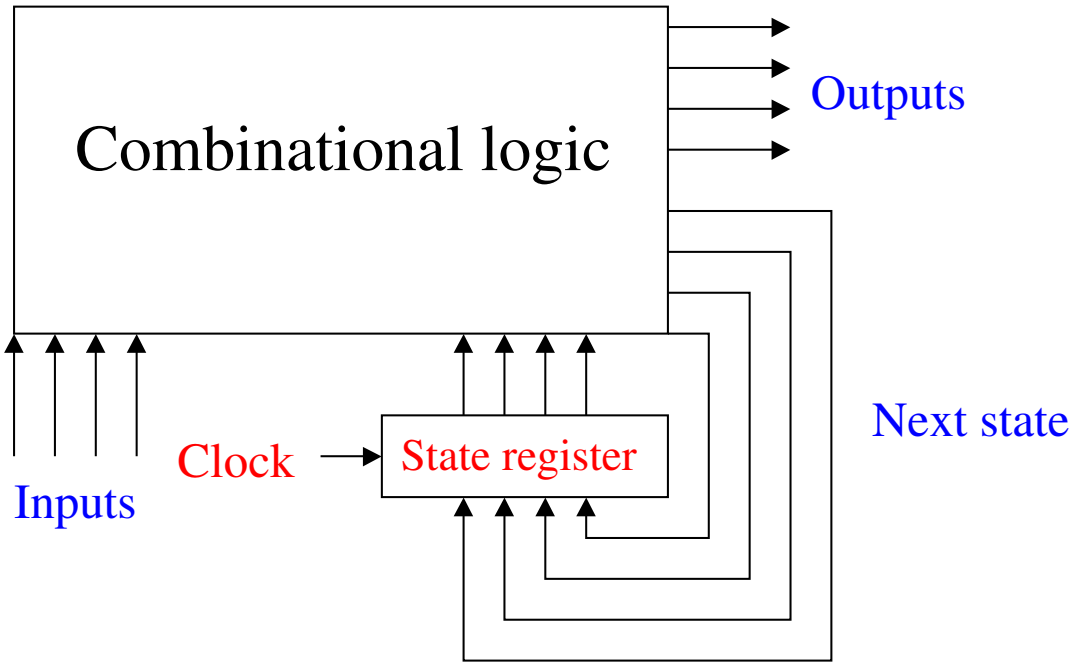


Stateful logic: Memory

In reality memory is implemented somewhat differently.

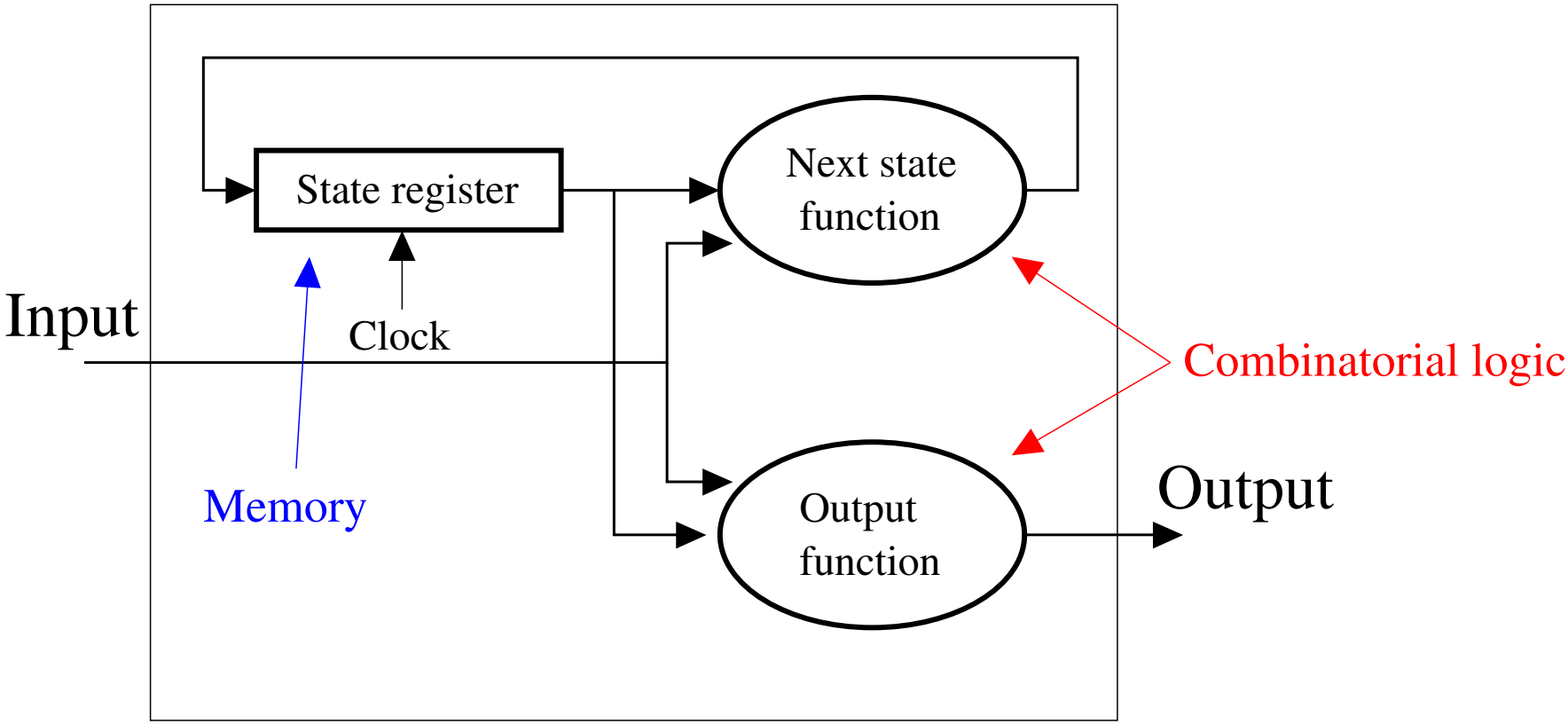
- SRAM: Static random access memory. The type of memory shown in the previous figure.
  - Very fast.
  - Requires 6 transistors per bit.
  - Very little power consumption.
  - Cache memory is typically built using SRAM.
- DRAM: Dynamic random access memory.
  - Slow.
  - Requires only 1 transistor per bit (connected to a capacitor).
  - The capacitor must be recharged thousands of times per second (as its charge leaks gradually).
  - High power consumption (due to recharge of capacitor).

The state machine

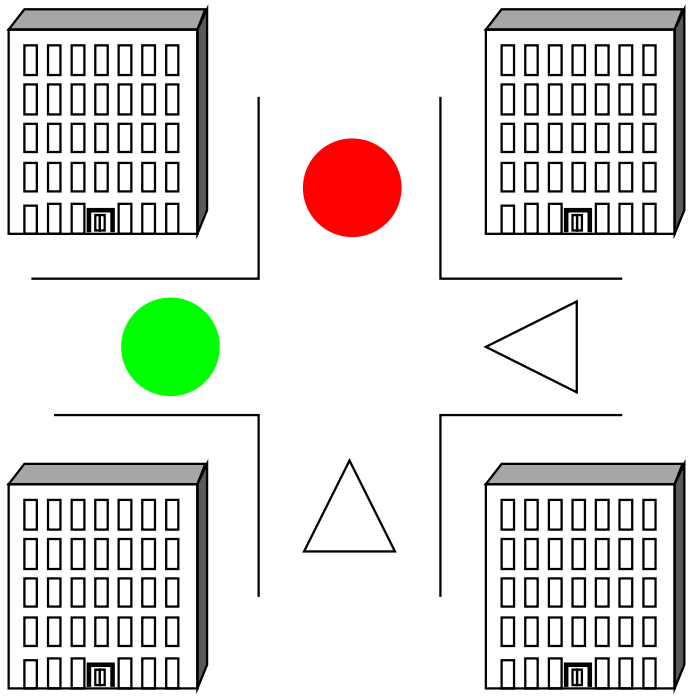


- Output is dependent upon current input **and** state of memory.
- State of memory can change based on input, output, and previous memory state.
- A **clock signal** is used to control when to change memory state.

The state machine



Example: Traffic light



Rules:

- Only RED and GREEN lights.
- RED and GREEN are mutually exclusive in south-north (SN)/east-west (EW) direction.
- If no car in either lane, light should not change.

Example: Traffic light

Outputs:

- SNlight: When asserted, SN light is green. When deasserted, SN light is red.
- EWlight: When asserted, EW light is green. When deasserted, EW light is red.

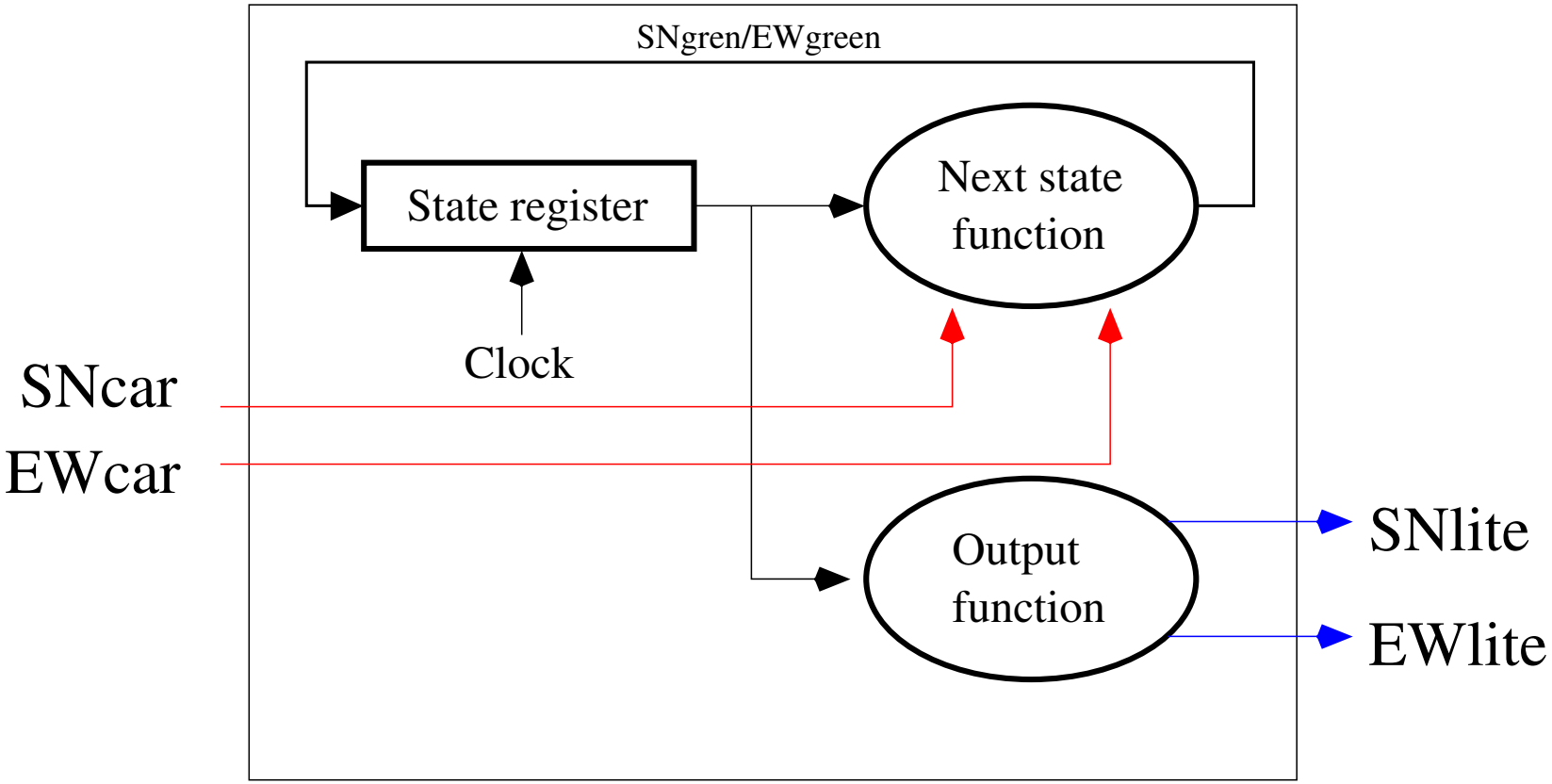
Inputs:

- SNcar: Asserted if car wants to drive north.
- EWcar: Asserted if car wants to drive west.
- Input drawn from detector system.

States:

- SNgreen: Traffic light is green in SN direction.
- EWgreen: Traffic light is green in EW direction.
- Two states, can be encoded using 1 bit.

Example: Traffic light



Next state function

Current state	Inputs		Next state
	SNcar	EWcar	
SNgreen	0	0	SNgreen
SNgreen	0	1	EWgreen
SNgreen	1	0	SNgreen
SNgreen	1	1	EWgreen
EWgreen	0	0	EWgreen
EWgreen	0	1	EWgreen
EWgreen	1	0	SNgreen
EWgreen	1	1	SNgreen

Output function

Current state	Outputs	
	SNlight	EWlight
SNgreen	1	0
EWgreen	0	1