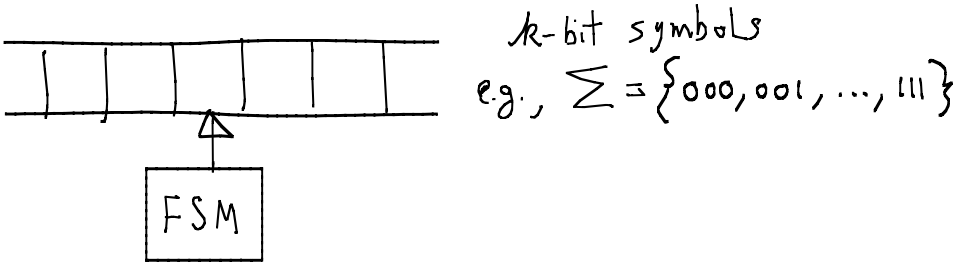
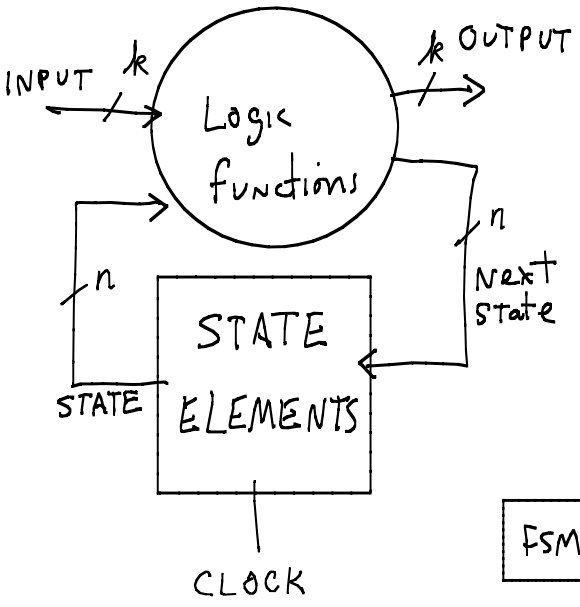
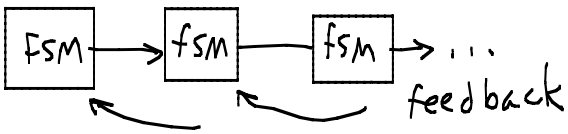
FSM-implementation



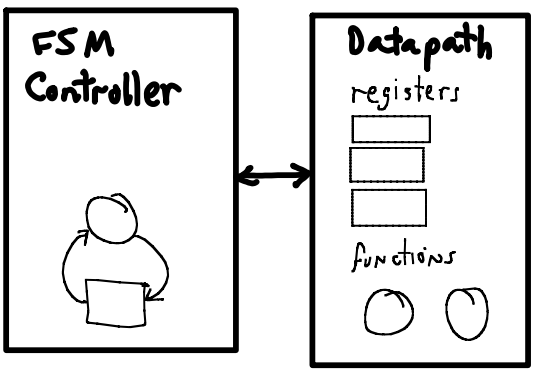




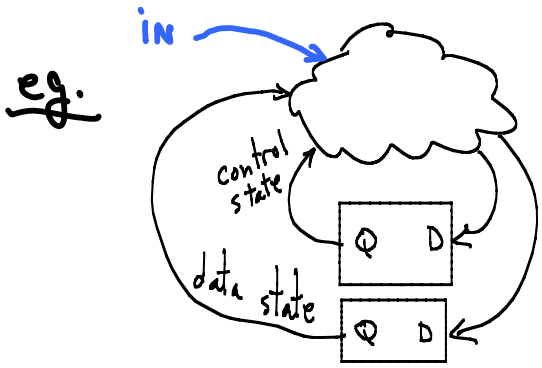
FSM has input/output, but from/to where?

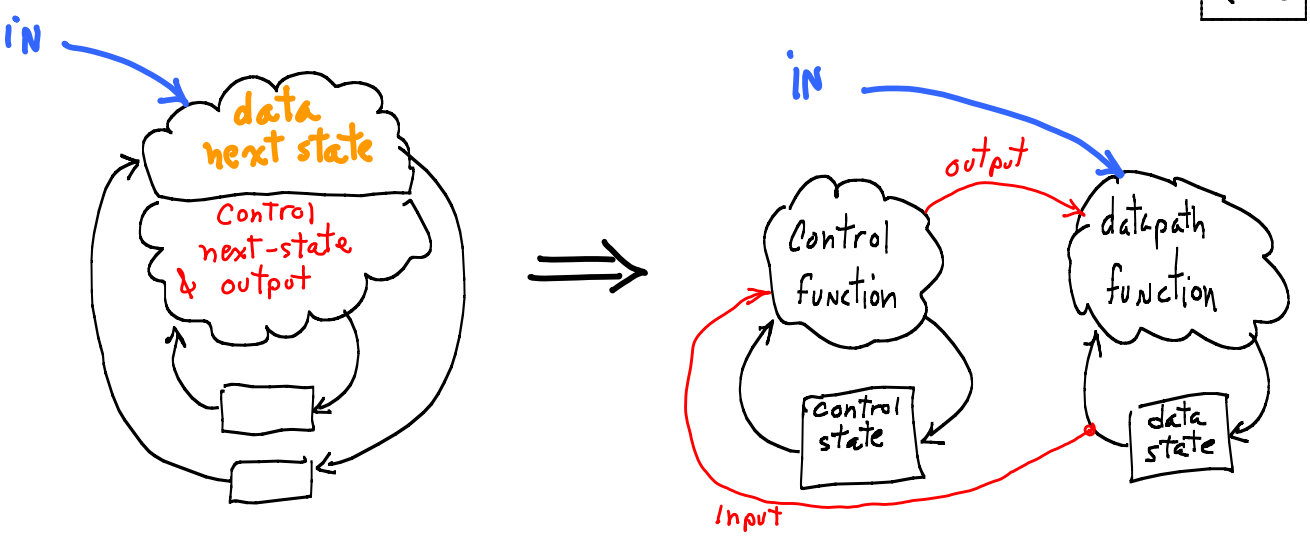
(1) Other FSMs in same machine

(2) Feedback loops

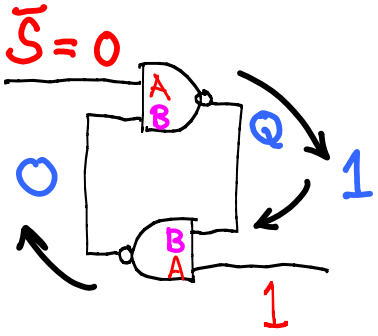


We like to separate state into two "types", control and data state. Eg., some state elements are for "control" state, and some are for "data" state.





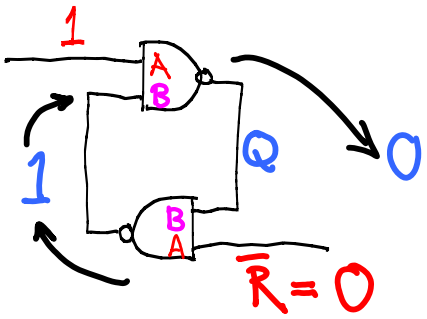
Basic Sequential elements



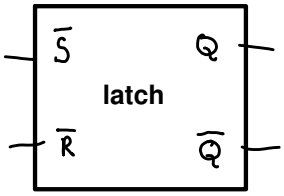
NAND

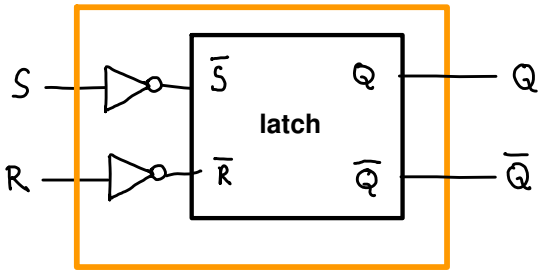
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

A = 0 ⇒ Q = 1

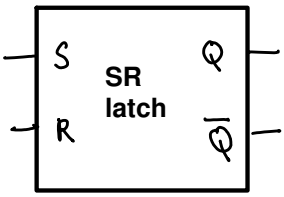


$\bar{S}$	$\bar{R}$	
1	1	stable
0	1	set Q = 1
1	0	reset Q = 0



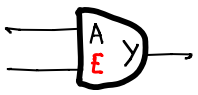


SR latch



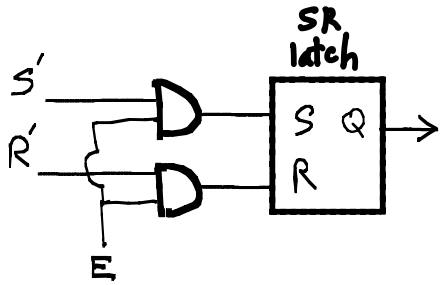
a basic element for state

CLOCKING, part 1: Gating D-Latch



E = 0
Y = 0

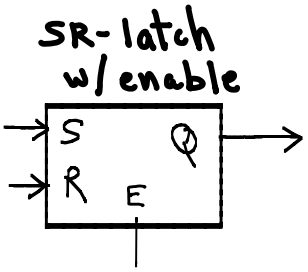
E = 1
Y = A



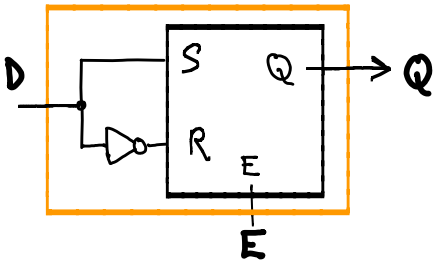
E	A	Y = E·A
0	0	0
0	1	0
1	0	0
1	1	1

E = 0 : input ignored

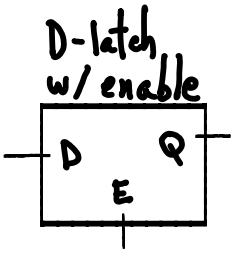
E = 1 : input passed



simplify ⇒



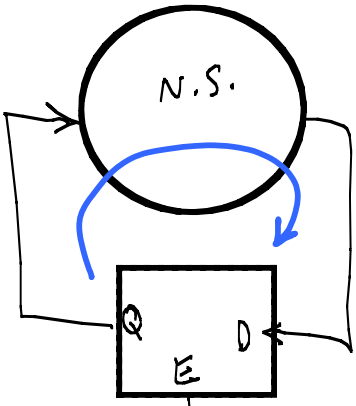
⇒



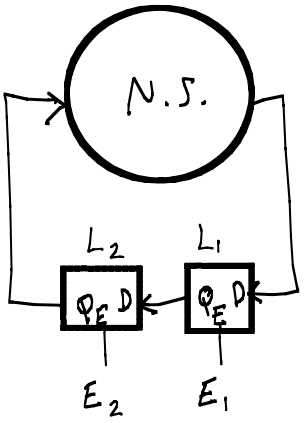
D = 0 : S = 0, R = 1 → **Q = 0**
D = 1 : S = 1, R = 0 → **Q = 1**
} if E = 1

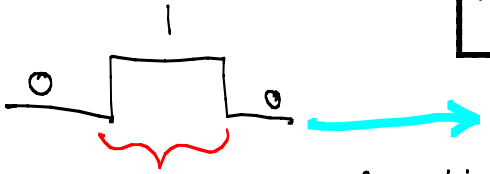
2-phase clocking, D-FF

Fix Problem



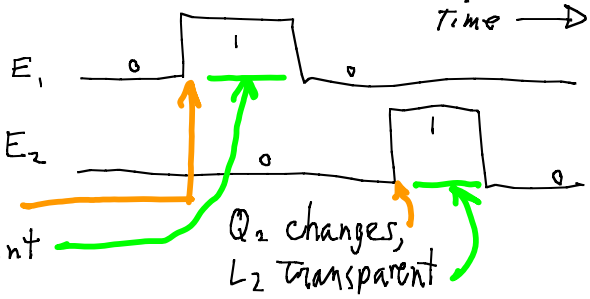
⇒



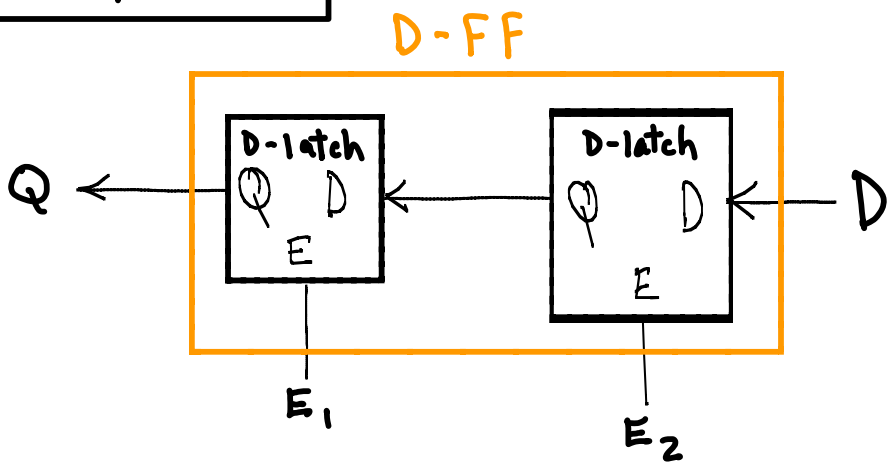


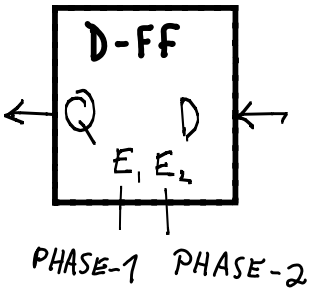
Transparent during this time, enough time to cause state change?

Make sure system never has an open feedback path. ⇒



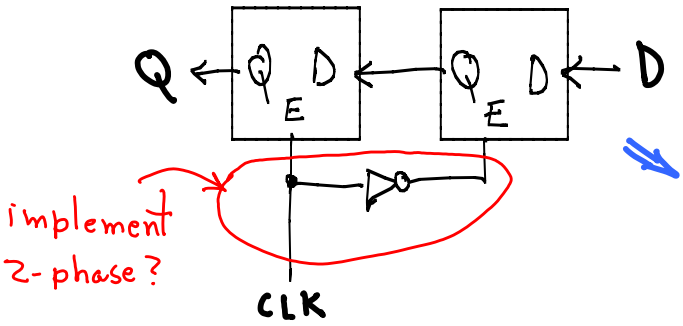
D-FF

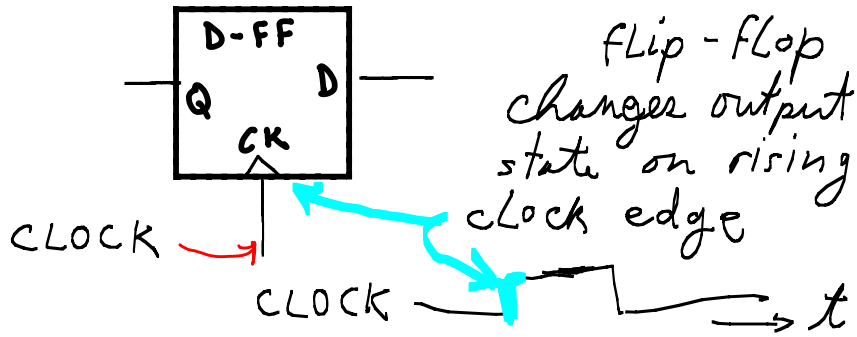




2-Phase Clocking

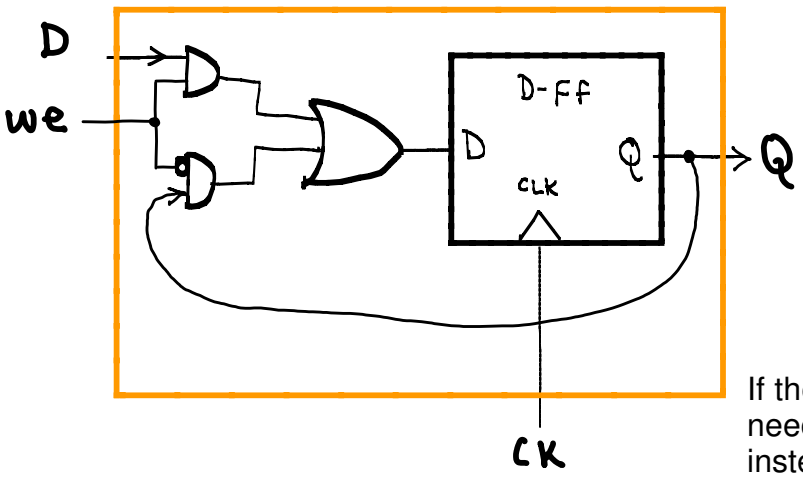
Separate signals for each latch's enable in FlipFlop. On breadboard we connect PHASE-1 to one data switch, PHASE-2 to another.

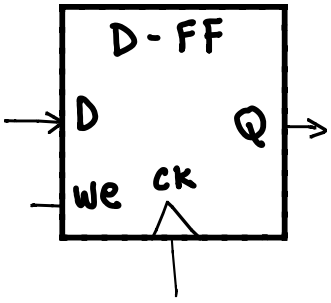




D-FF, Pos. edge triggered w/enable

We often want to control whether or not the FF will be written into when the clock pulse arrives: add an "enable" input. When enable is 0, the current state is written back into the FF. Otherwise, D is written.

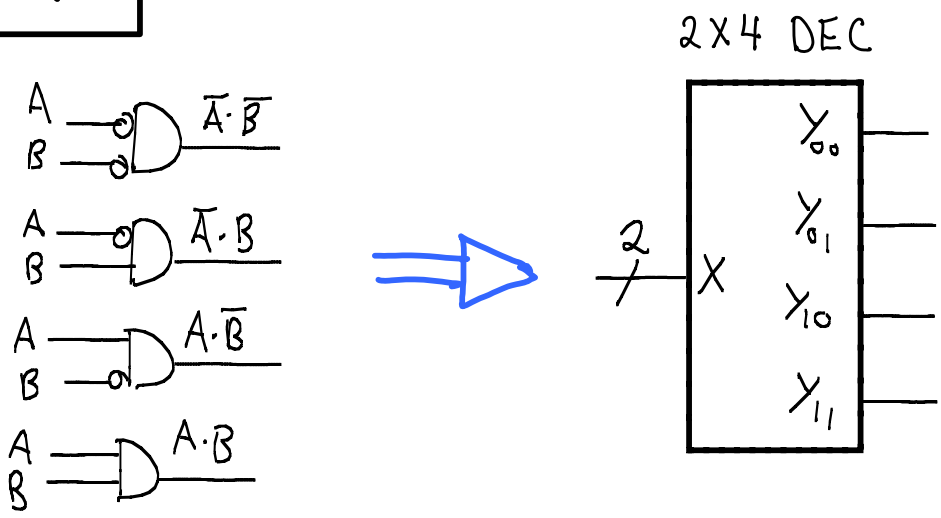




D-FF pos-edge Triggered w/ write enable

If there is no feedback path from Q to D, we do not need a flip-flop, we can use a write-enable latch instead. Datapaths sometimes can use latches.

PLA



Probrammable Logic Array consists of two parts:

PLA, part 1.

A **decoder**, can be thought of as:

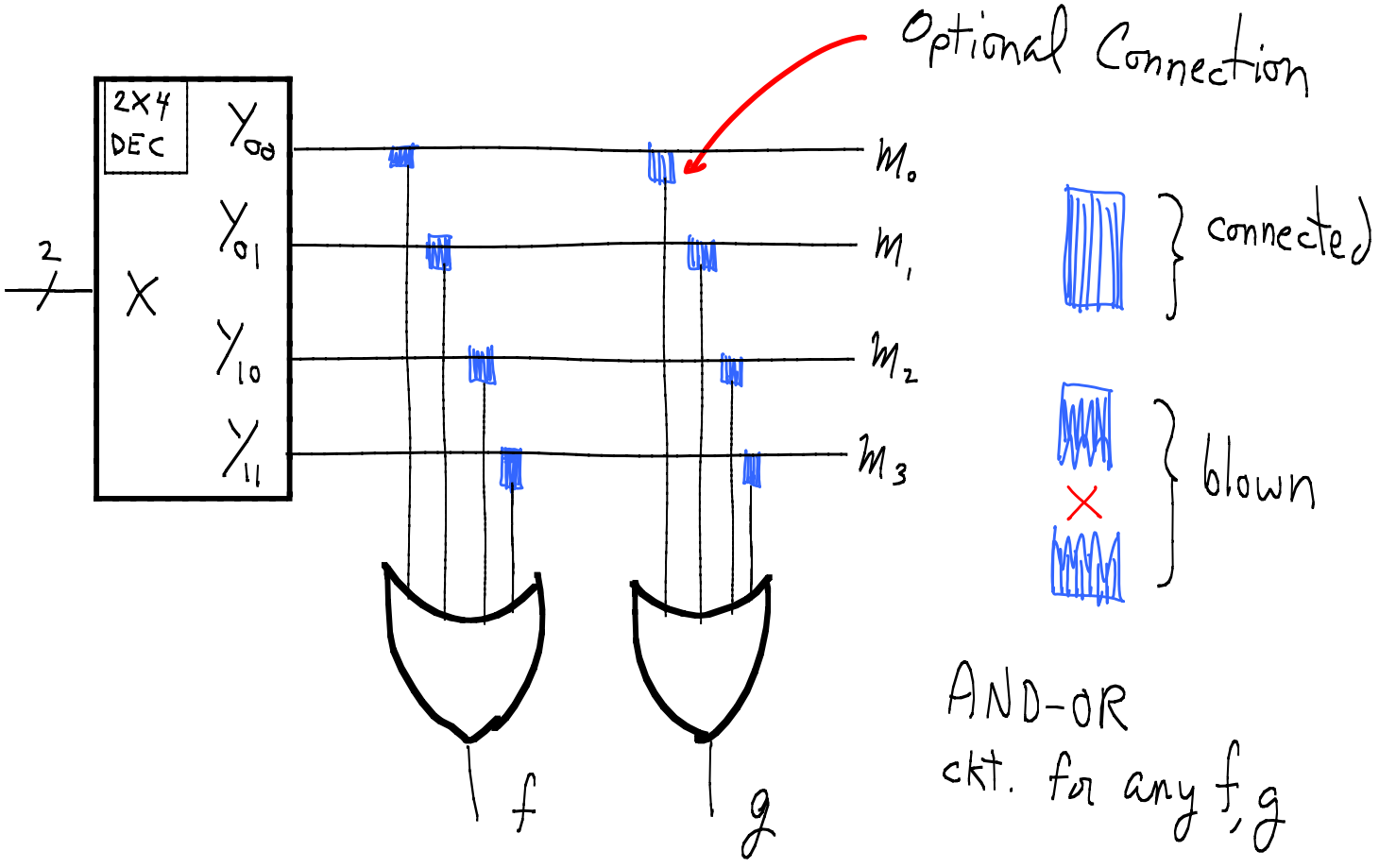
a) **activates exactly one output** from **input code**.

b) **generates all minterms** from **input**.

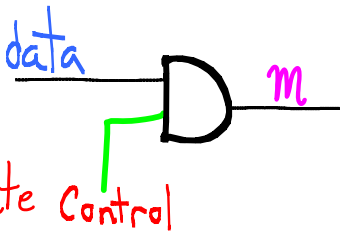
PLA, Part 2.
A means of **OR'ing minterms** to produce function outputs.

Can **share the same minterms**: we can economically produce multiple functions at once.

Programmable: minterm **lines can be "blown" to disconnect** them:
selects minterms included in the function.



MUX

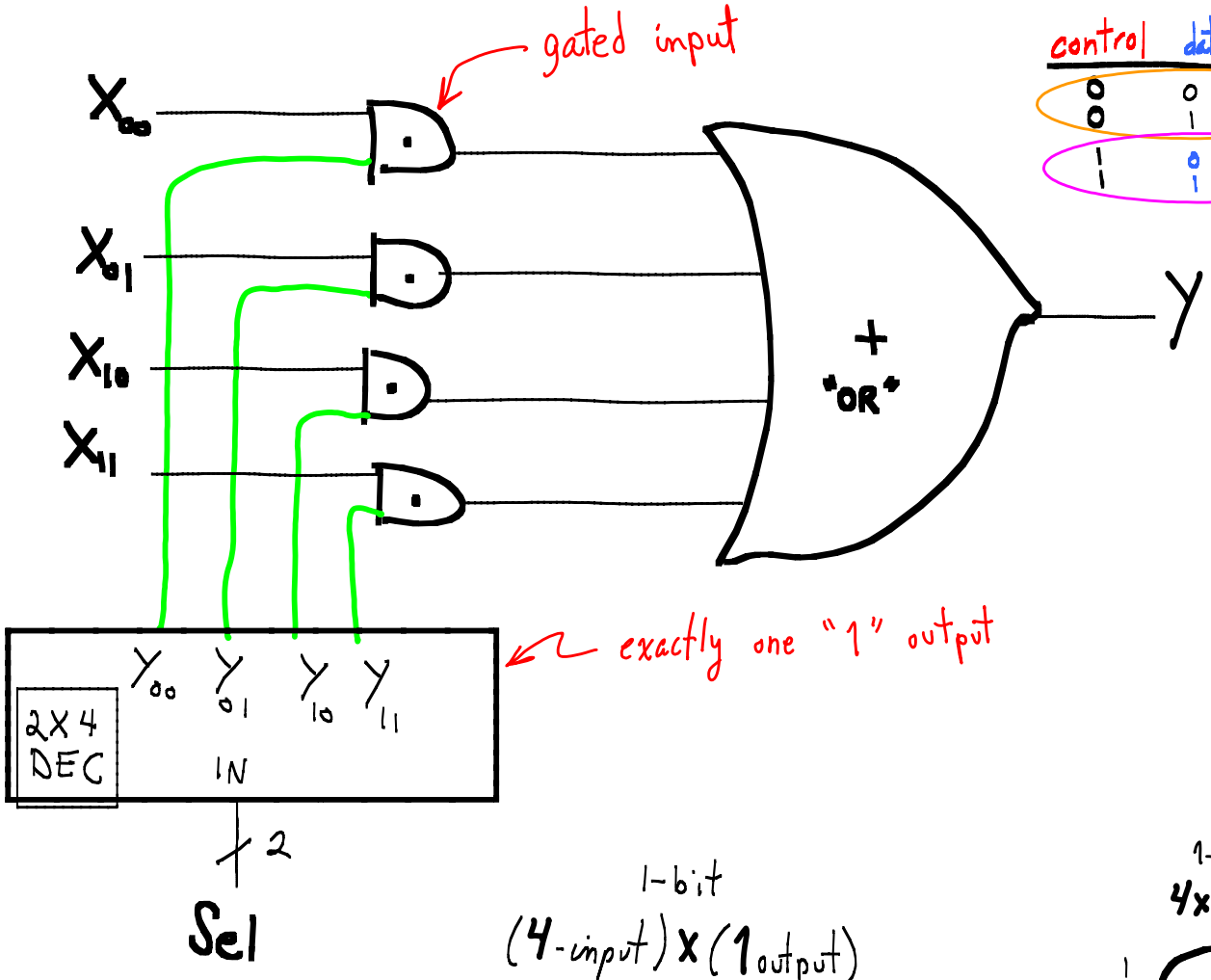


"gating"

Control = 0 : m = 0

Control = 1 : m = data

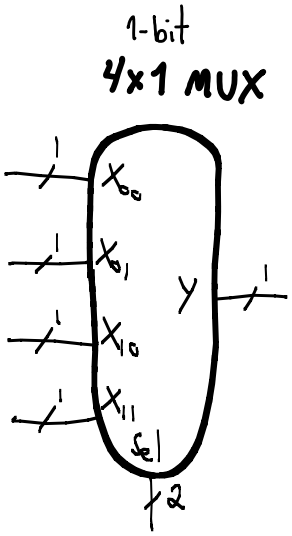
control	data	m
0	0	0
0	1	0
1	0	0
1	1	1



1-bit

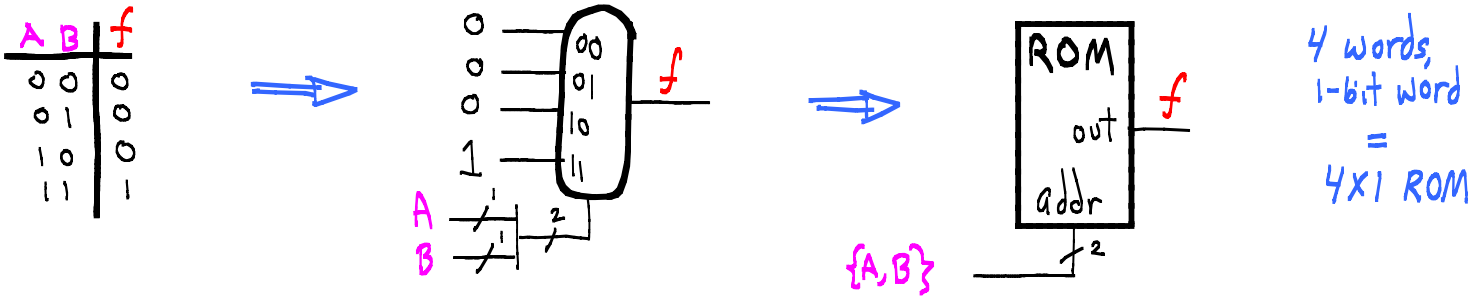
(4-input) x (1 output)

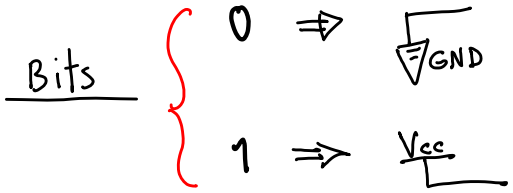
MUX



IMPLEMENTING FUNCTIONS as ROM

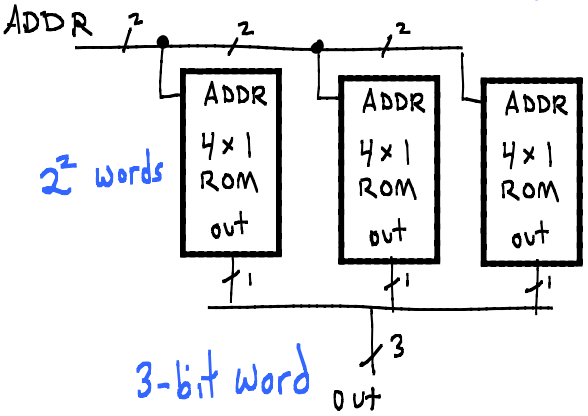
Logic = ROM = MUX + bits



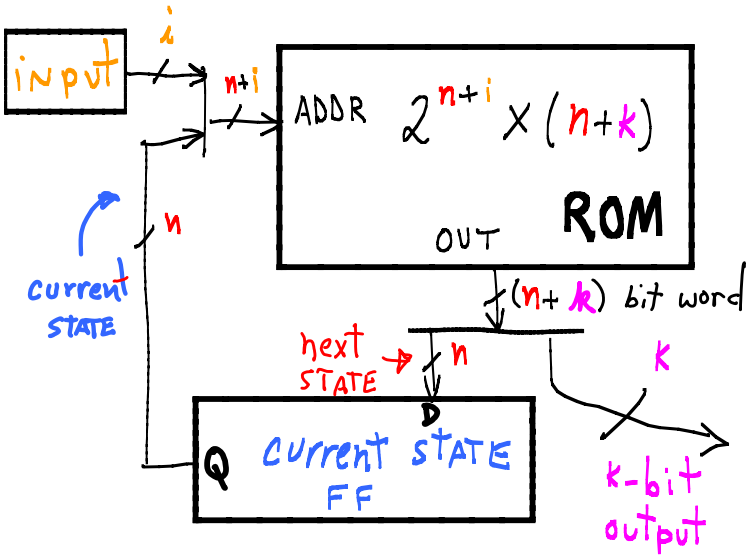


Bigger ROM

$2^2 \times 3$ ROM



FSM = ROM + STATE



FSM functions in ROM

i input bits to FSM
k output bits
n state bits (2^**n** states)

possible (**state**, **input**) pairs == 2^(**n**+**i**)

===> 2^(**n+i**) words in ROM
n+i address bits

ROM word == (**next-state**, **output**)

====> (**n + k**)-bit word size

FSM in ROM (n-bit state, i-bit input, k-bit FSM output)

(STATE, INPUT) is ROM address
n bits + i bits ===> 2^(n+i) ROM locations

(NEXT-STATE, FSM-OUTPUT) is ROM output
n bits + k bits ====> (n+k) bits per location

===> 2^(n+i) location by (n+k)-bit word ROM

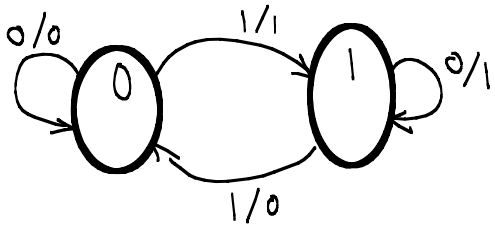
ANY FSM (Mealy or Moore) can be built as a ROM

NOTE: A Moore machine's output depends only on state ===> use n-bit addresses, one ROM location per state.

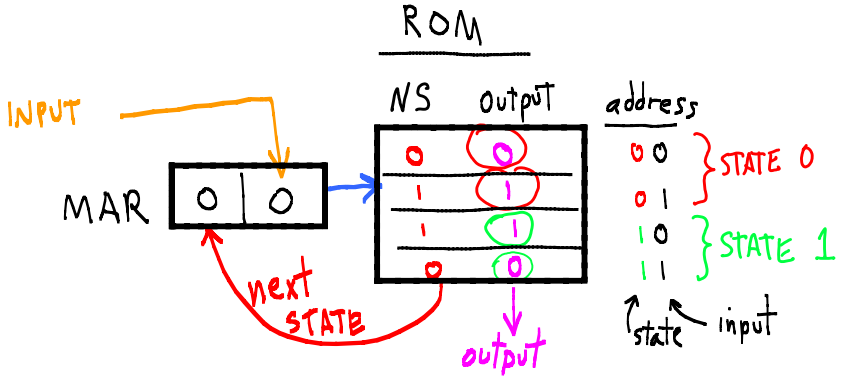
BUT, next-state depends on current-state+input. Encode part of next-state function in ROM word as NS-CODE, and use external logic to calculate next-state function: next-state = f(INPUT, NS-CODE). This is what is done in the LC3's micro-coded controller.

Every possible FSM can be built as a ROM.

ROM is very large since there is a word for every possible {state, input} combination.



address		ROM data word (ROW)	
STATE	INPUT	next-state	OUTPUT
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

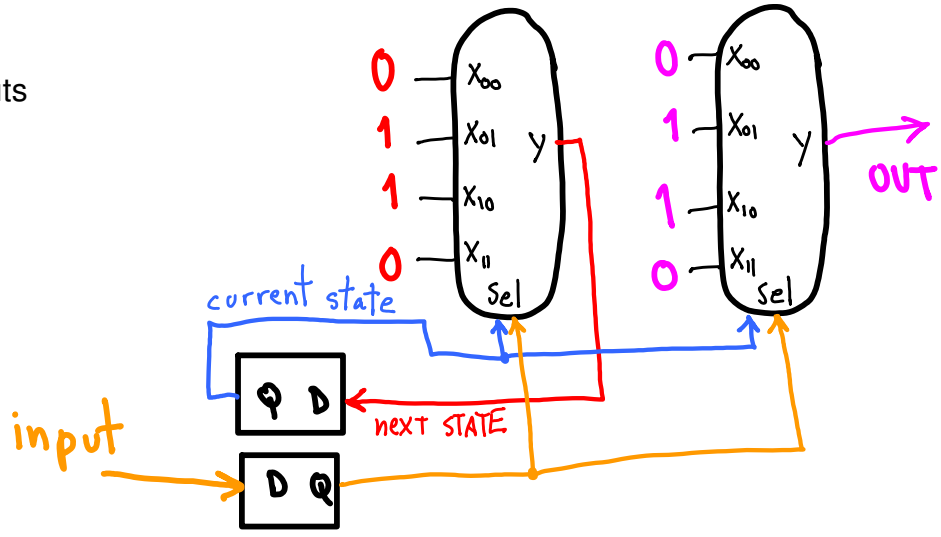


at clock tick:

-- { current state, current input } captured
-- output changes to match captured state/input

-- Every state row has same output
===> Moore Machine

-- Rows for state S have differing outputs
===> Mealy Machine.



We can enumerate all ROMs (and consequently all TMs/digital-computers):

Concatenate ROM content from all words:

address	content
00	00
01	11
10	11
11	00

==> 01111000

List all n = i = k = 1 machines:
FSM-0, FSM-1, ..., FSM-256

List all n = i = k = 2 machines:
FSM-257, FSM-258, ...

and so on.