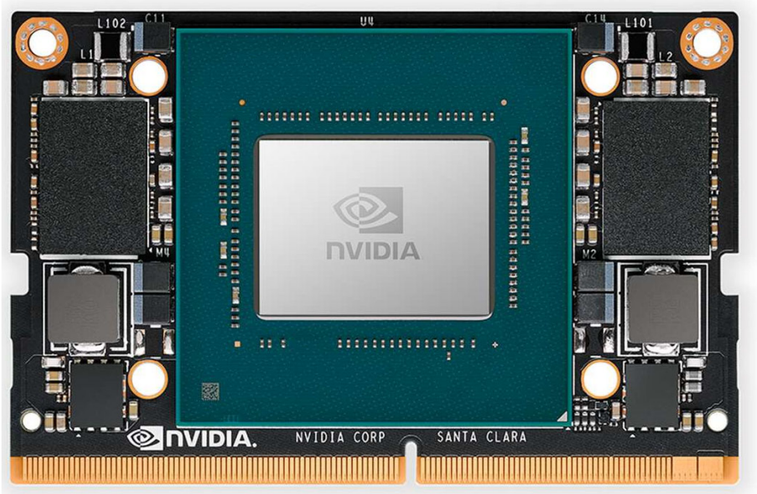
EECS151 : Introduction to Digital Design and ICs

Lecture 18 – SRAM

Bora Nikolić and Sophia Shao

Nvidia Shrinks AI 'Supercomputer' to Credit Card Size

Nov. 6, 2019. Nvidia's Jetson Xavier NX board is smaller than a credit card and provides 1.4 TOPS/W (Image: Nvidia)



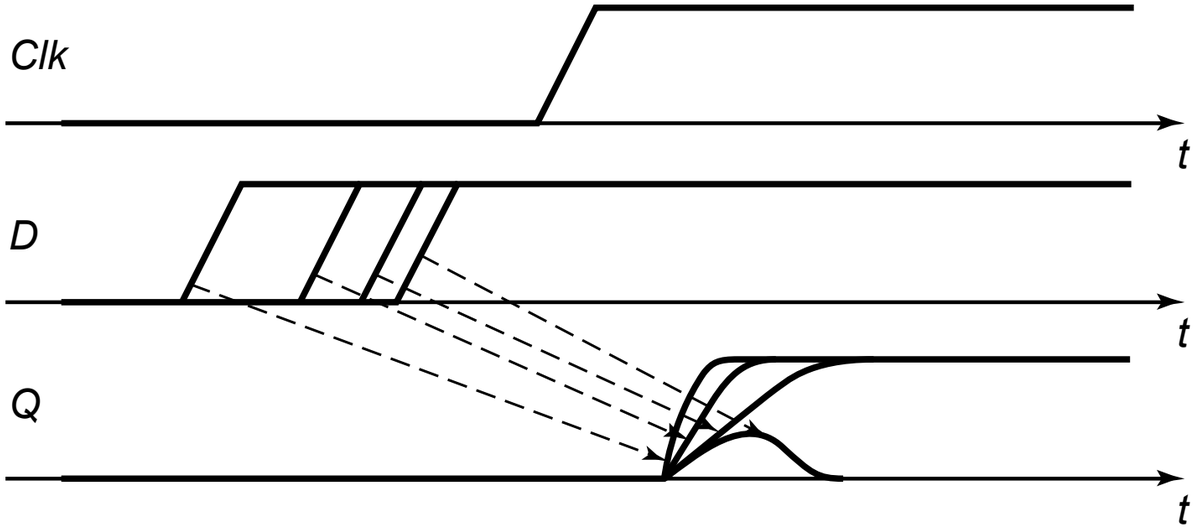
https://www.eetimes.com/document.asp?doc_id=1335268

Review

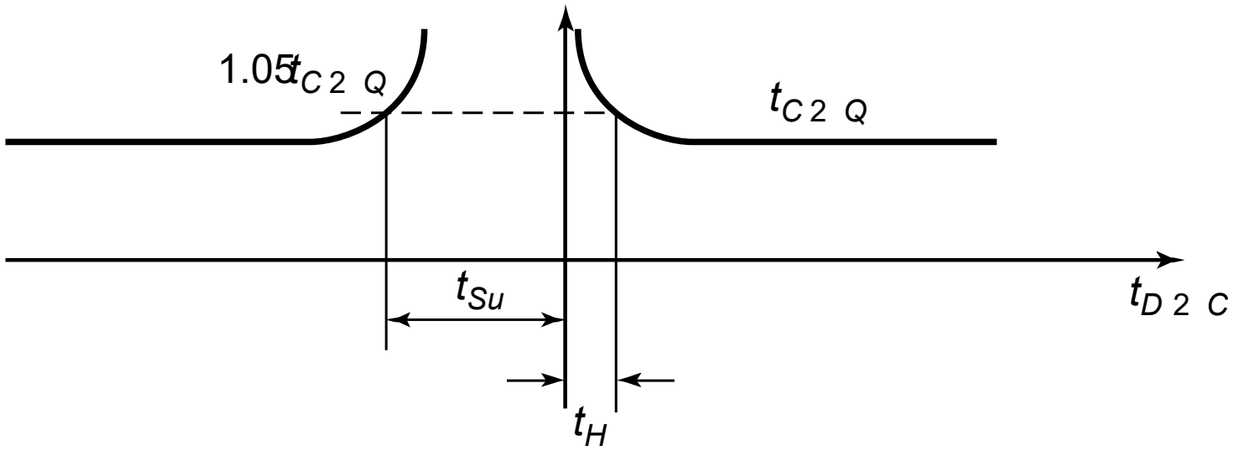
- Flip-flops are edge-triggered
- Long timing paths need to meet the setup time of receiving flip-flop
- Short timing paths have to be longer than hold time + $t_{clk\text{-}Q}$
- Flip-flops are often designed as latch pairs

Setup and Hold Times

Setup and Hold Times



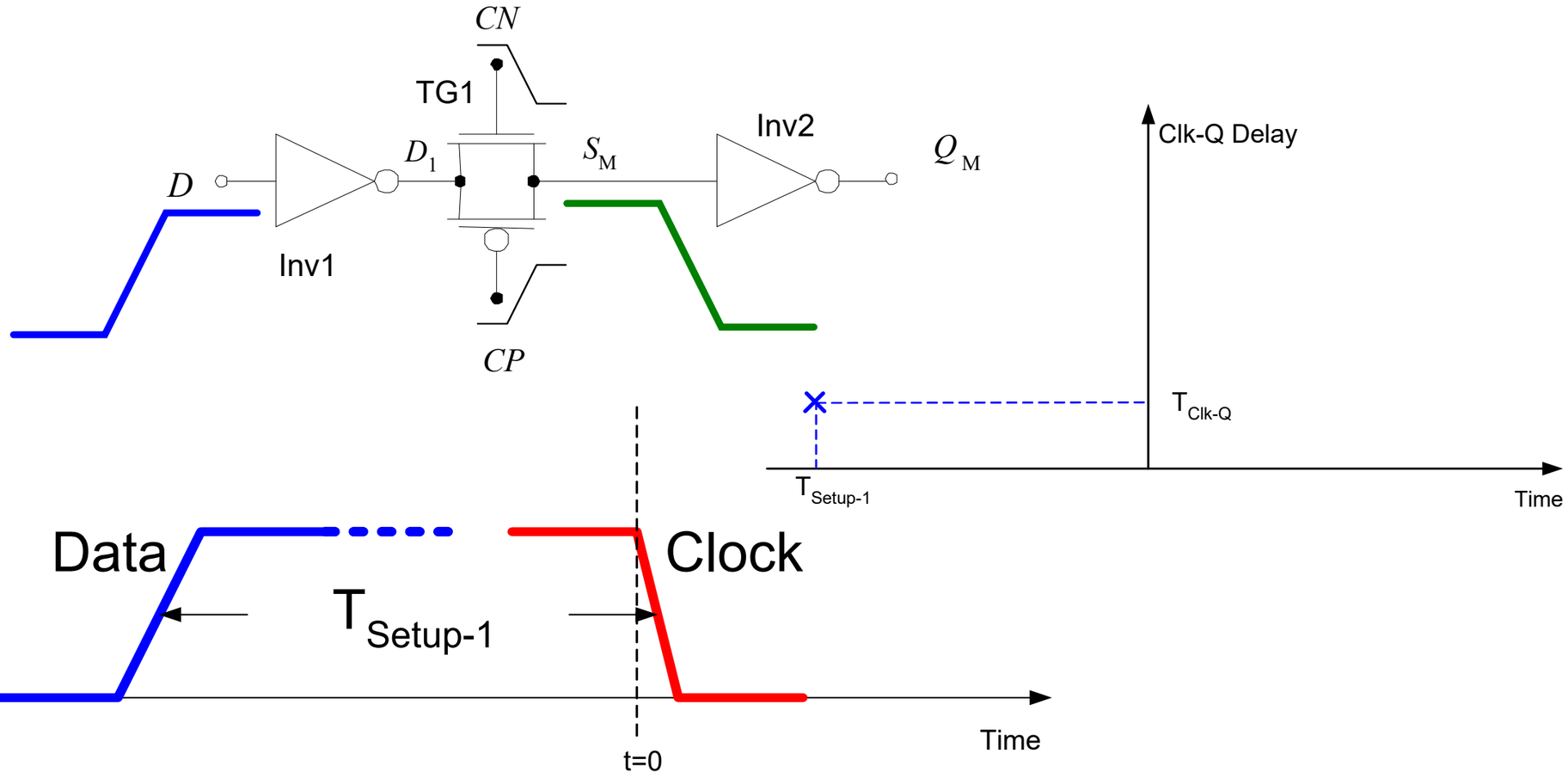
(a)



(b)

Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)

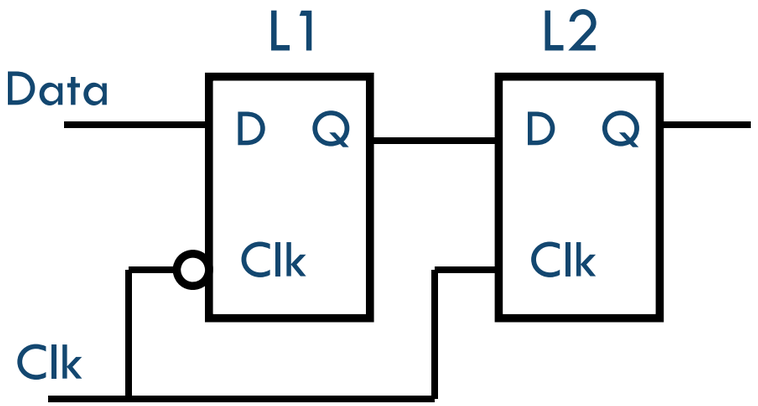


Flip-Flops

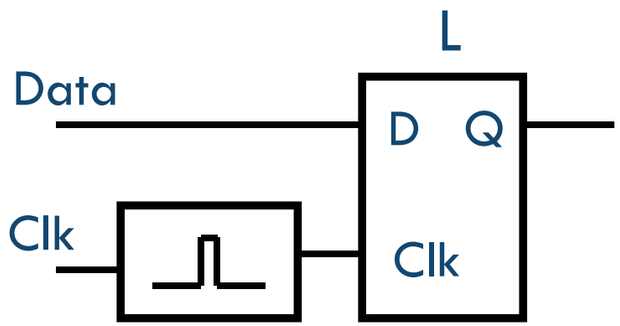
Types of Flip-Flops

Latch Pair

(Master-Slave)



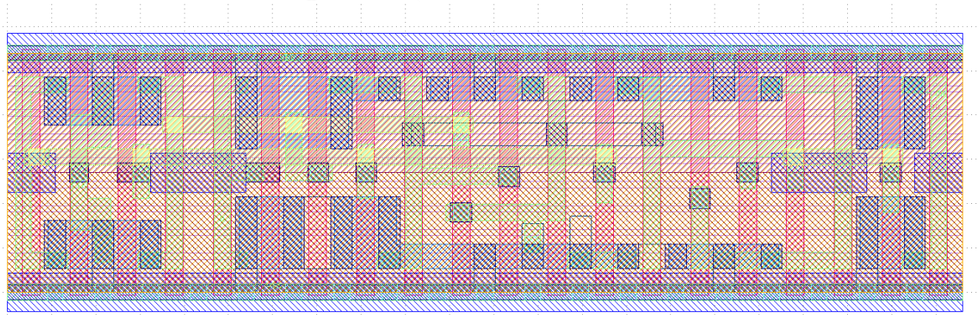
Pulse-Triggered Latch

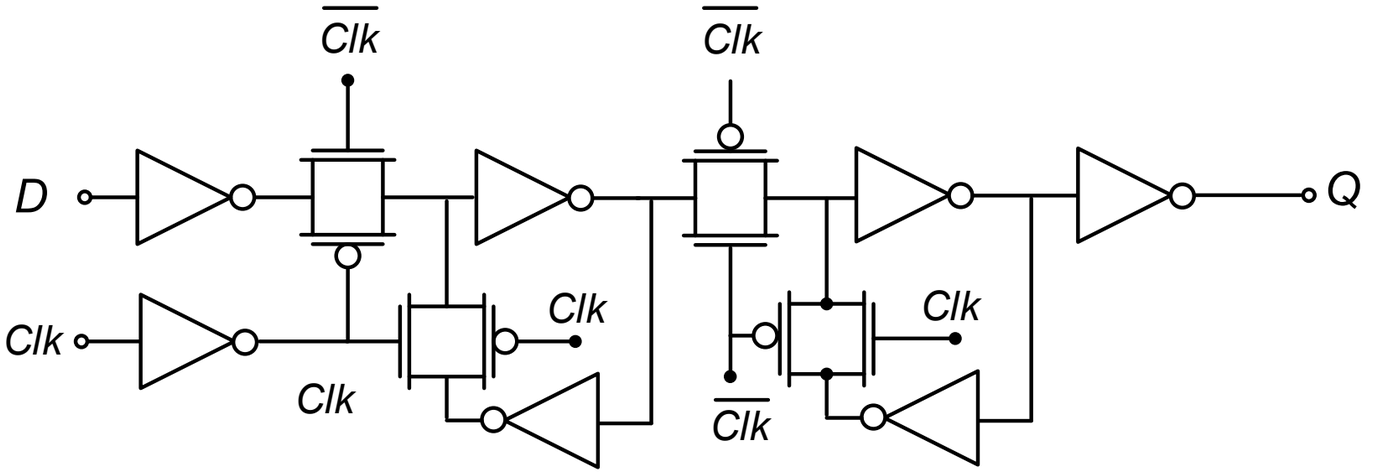


Transmission Gate Flip-Flop

- Two back-to-back latches

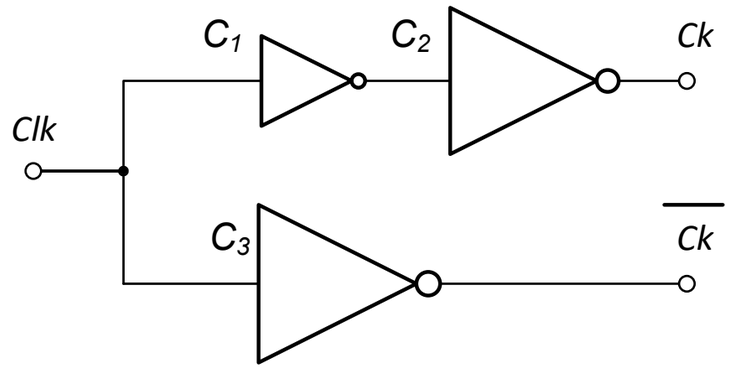
- In ASAP7



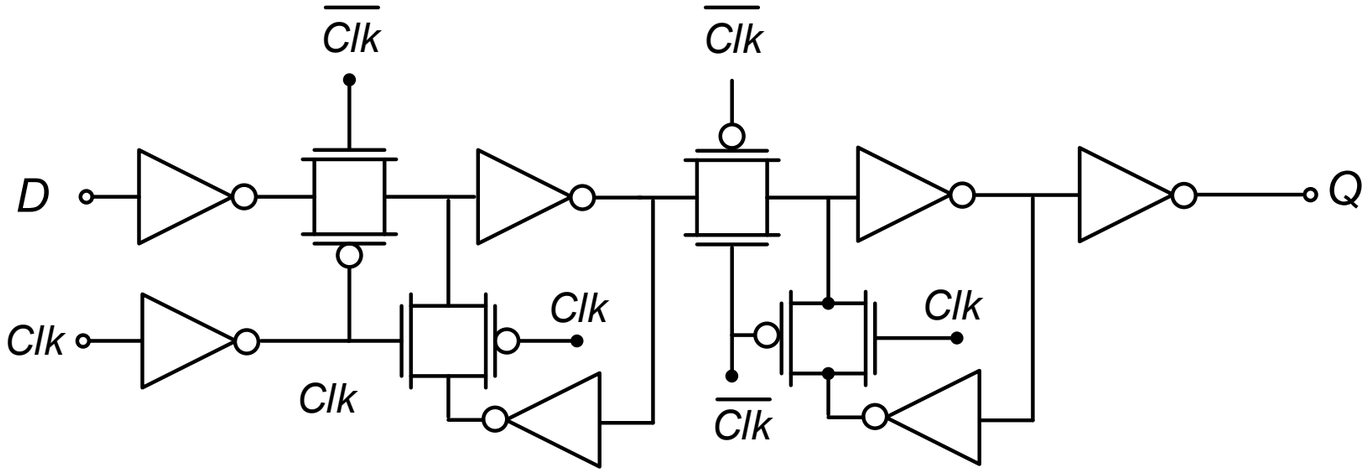


Aside: Inverter Fork

- Often found in flip-flops: equalize Ck, Ckb delays



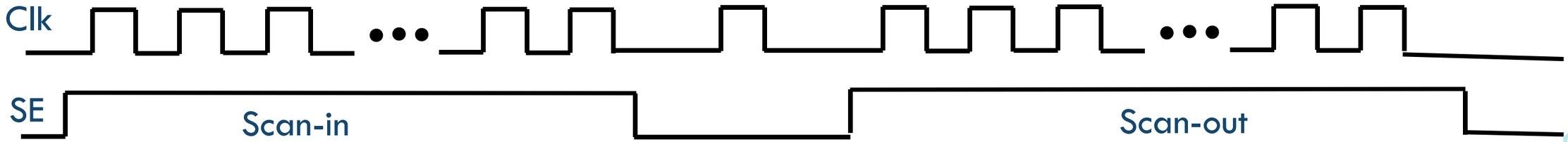
Clk-Q, Setup and Hold Times



Scan

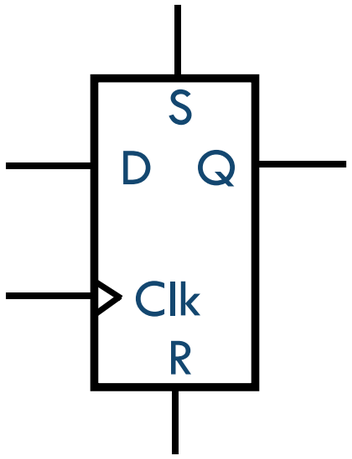
- Scan input is used for functional test of the chip

- During test, a desired pattern is 'scanned' into the chain of flip-flops



Set, Reset

- Set and reset can be synchronous or asynchronous
  - Always watch for additional timing paths!



D-flip-flop with synchronous reset

```
module dff_sync_clear(
  input d, r, clk,
  output reg q);

  always @(posedge clk)
    begin
      if (!r) q <= 1'b0;
      else q <= d;
    end
endmodule
```

`always` block entered only at each positive clock edge

D-flip-flop with asynchronous reset

```
module dff_async_clear(
  input d, r, clk,
  output reg q);

  always @(negedge r or posedge clk)
    begin
      if (!r) q <= 1'b0;
      else q <= d;
    end
endmodule
```

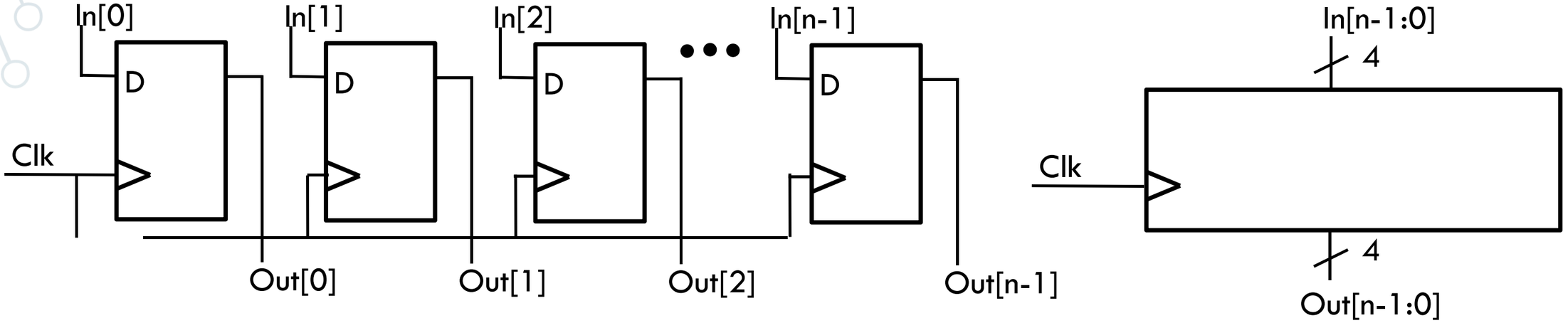
`always` block entered immediately when (active-low) r is asserted

Flip-Flop Timing Characterization

- Combinational logic delay is a function of output load and input slope
- Sequential timing (flip-flop):
  - $t_{clk\text{-}q}$ is function of output load and clock rise time
  - t_{Su}, t_H are functions of D and Clk rise/fall times

Registers, Register files

- Register is often built out of flip-flops

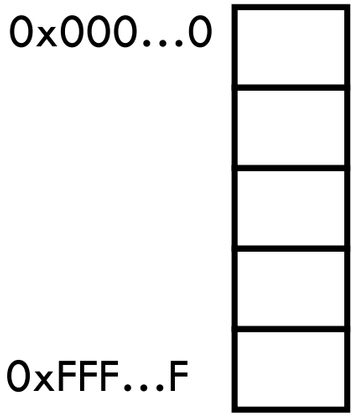


- Register file can be built out of registers
  - Ok for small register files
  - Large register files are generally built with latches and custom designed (like memory arrays)

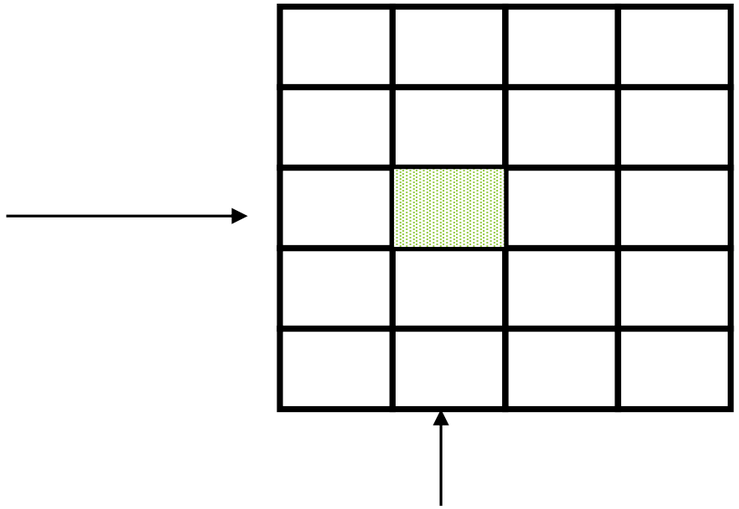
SRAM

Random Access Memory Architecture

- Conceptual: Linear array of addresses
  - Each box holds some data
  - Not practical to physically realize
    - millions of 32b/64b words

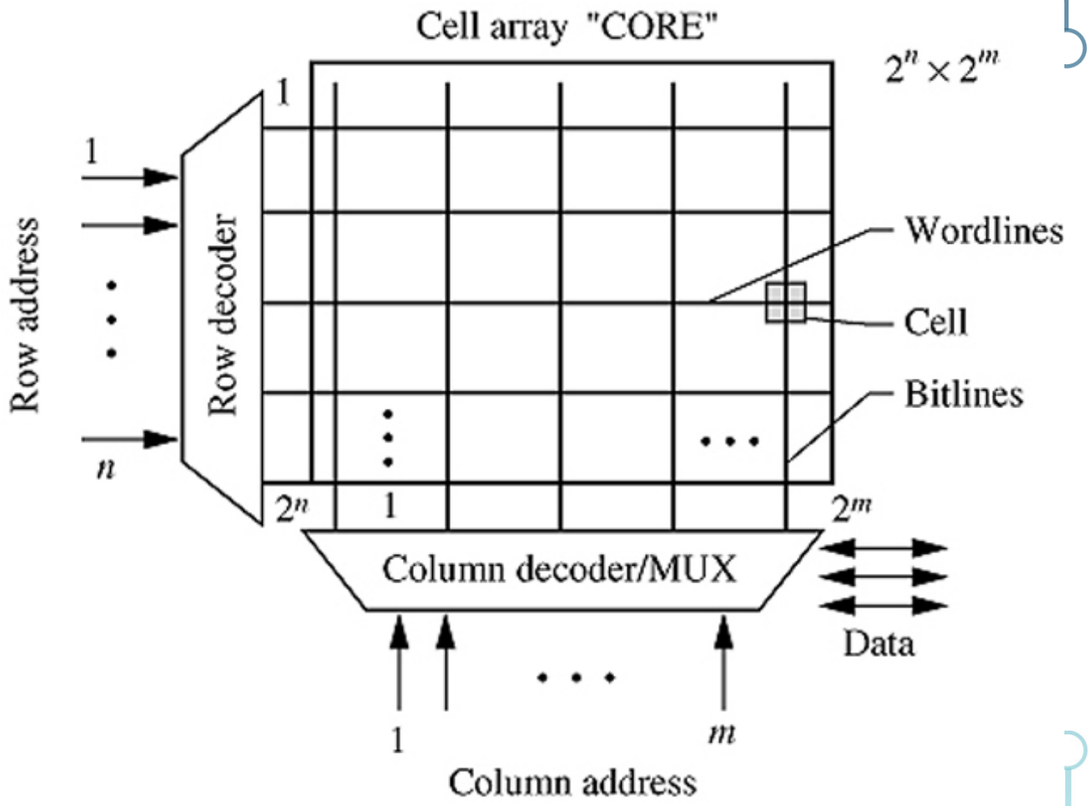


- Create a 2-D array
  - Decode Row and Column address to get data

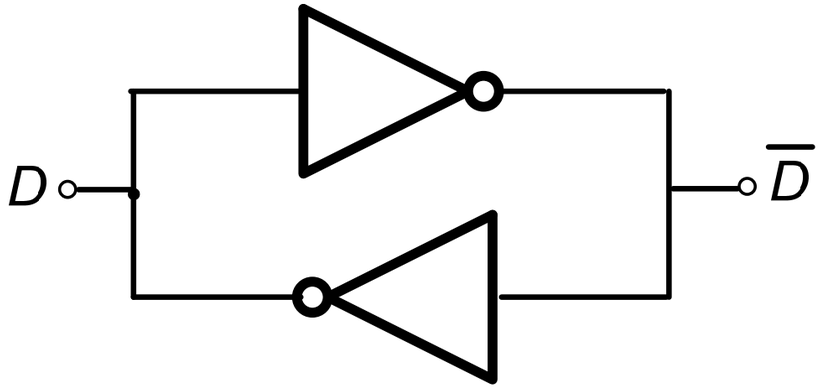


Basic Memory Array

- CORE
  - Wordlines to access rows
  - Bitlines to access columns
  - Data multiplexed onto columns
- Decoders
  - Addresses are binary
  - Row/column MUXes are ‘one-hot’ - only one is active at a time



Basic Static Memory Element

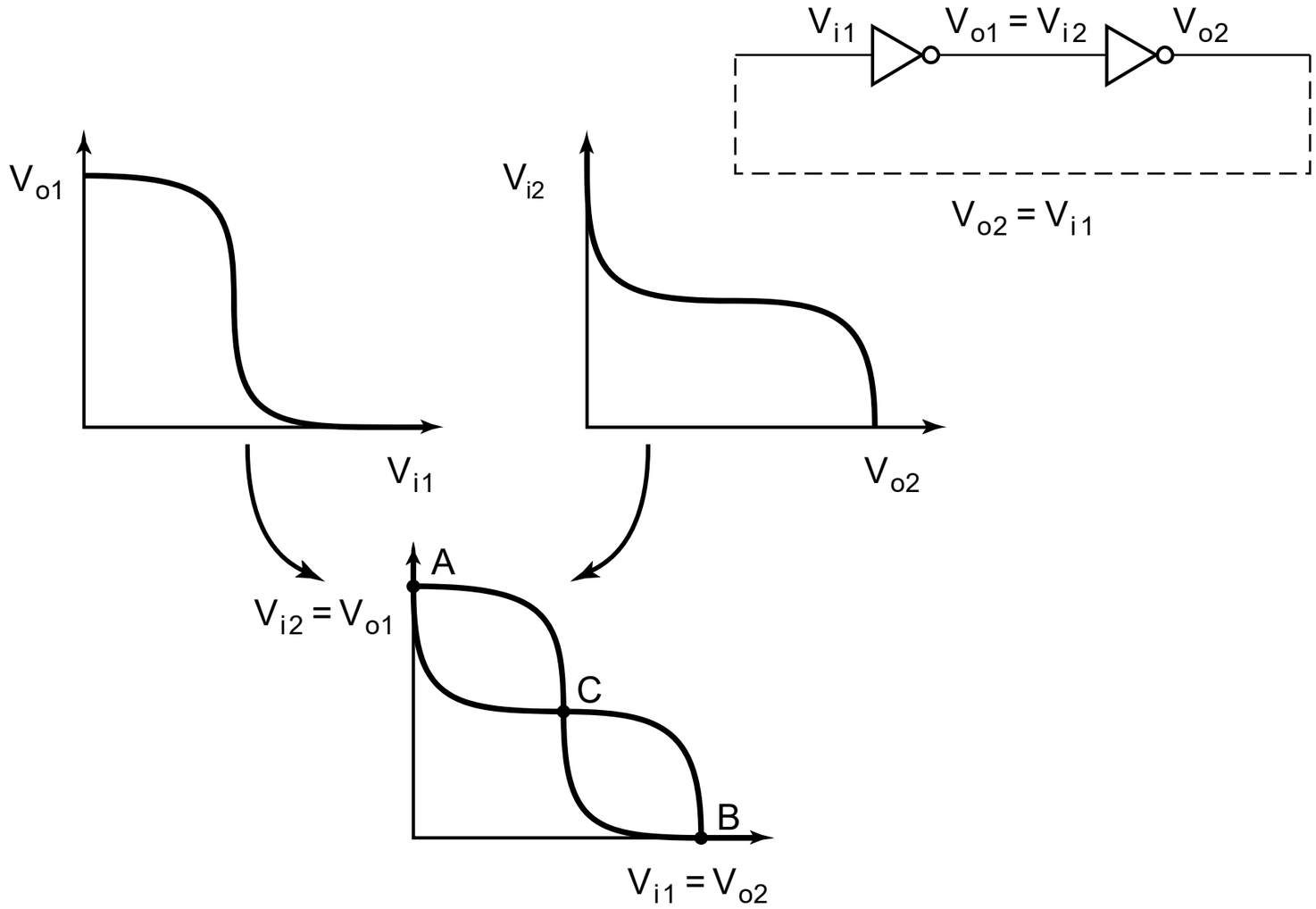


- If D is high, $\overline{D}$ will be driven low
  - Which makes D stay high

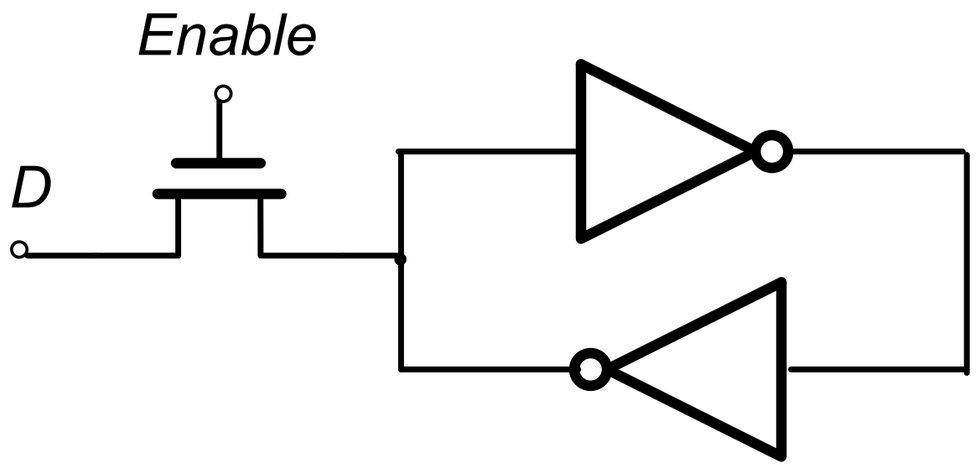
- Positive feedback
- Same principle as in latches

Positive Feedback: Bi-Stability

- As in latches



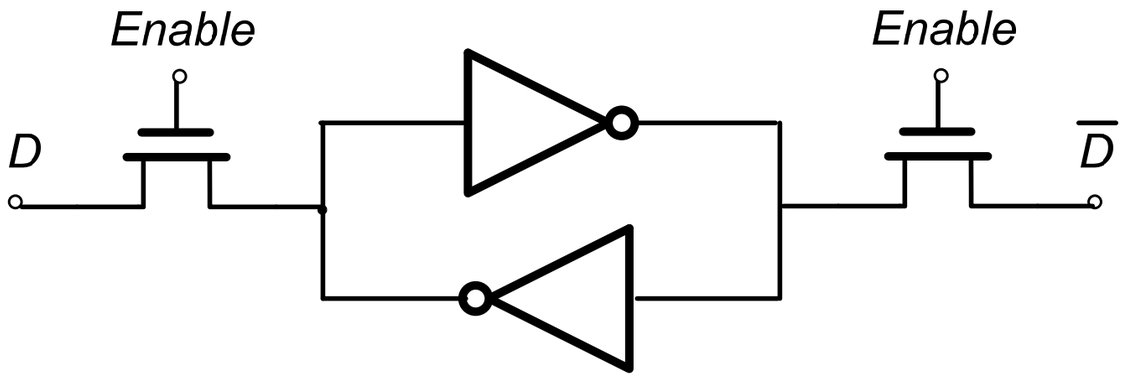
Writing into a Cross-Coupled Pair



- This is a 5T SRAM cell
  - Access transistor must be able to overpower the feedback; therefore must be large
  - Easier to write a 0, harder to write 1
- Can implement as a transmission gate as well; single-ended 6T cell

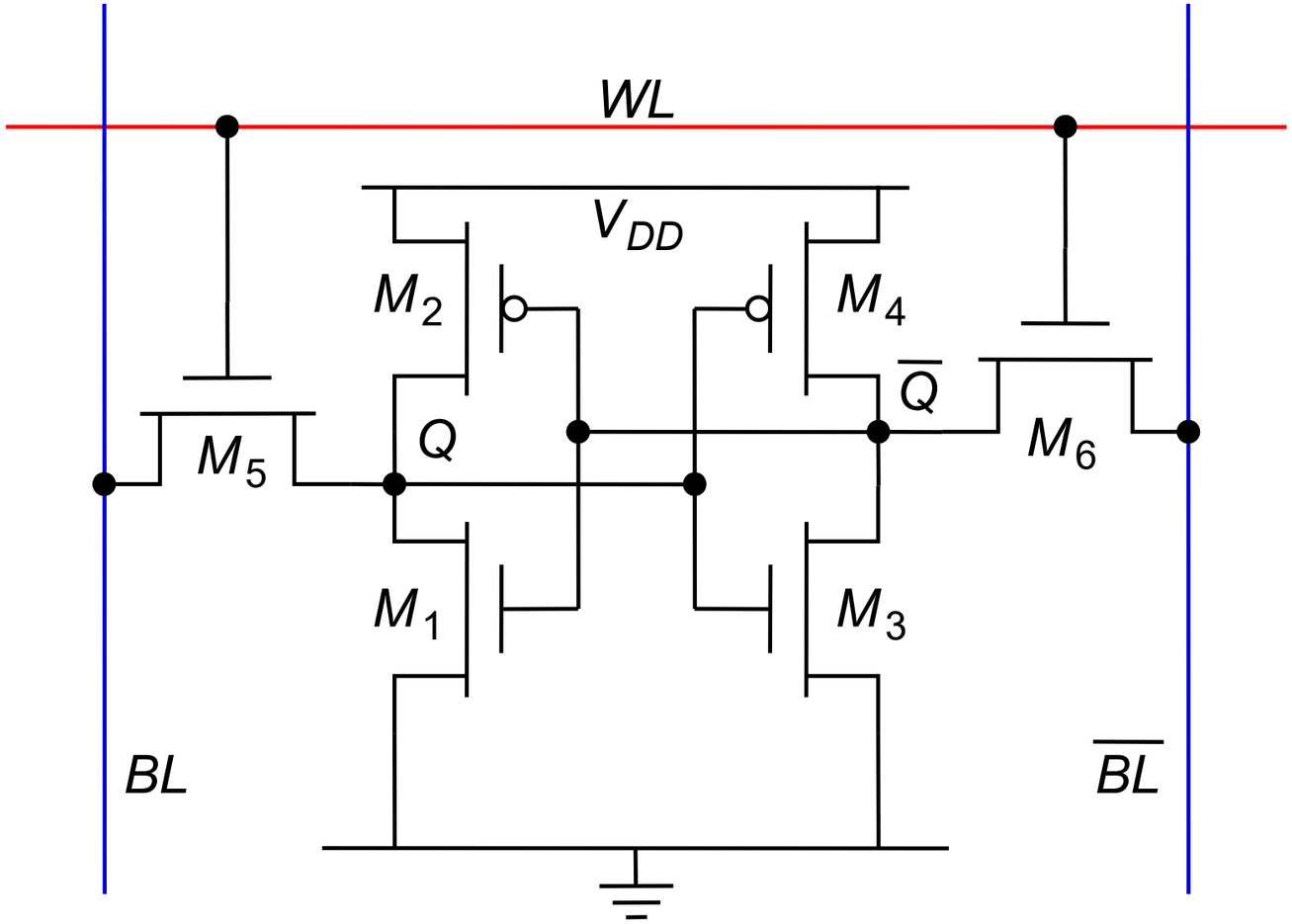
- There is a better solution…

SRAM Cell



Since it is easier to write a 0 through NMOS, write only 0s, but on opposite sides!
When reading, measure the difference

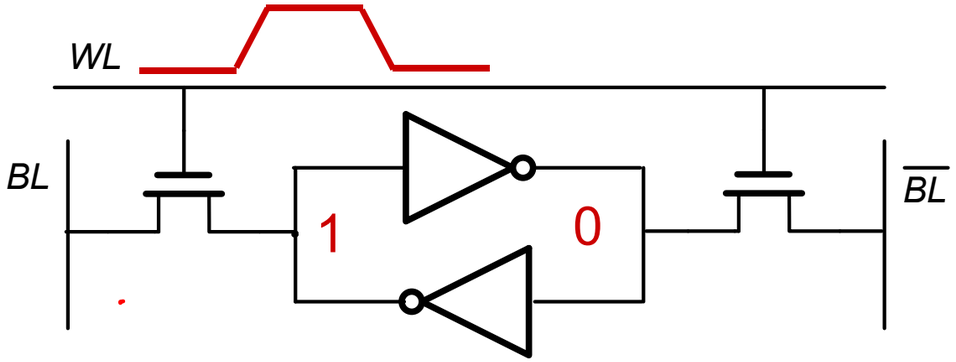
6-transistor CMOS SRAM Cell



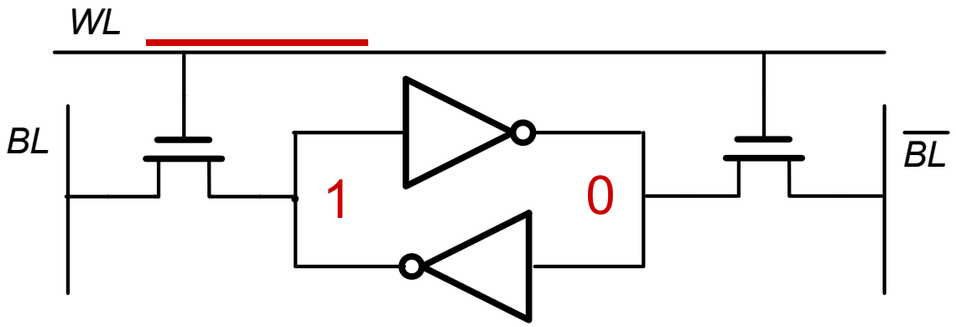
- Wordline (WL) enables read/write access for a row
- Data is written/read differentially through shared BL, $\overline{BL}$

SRAM Operation

Write

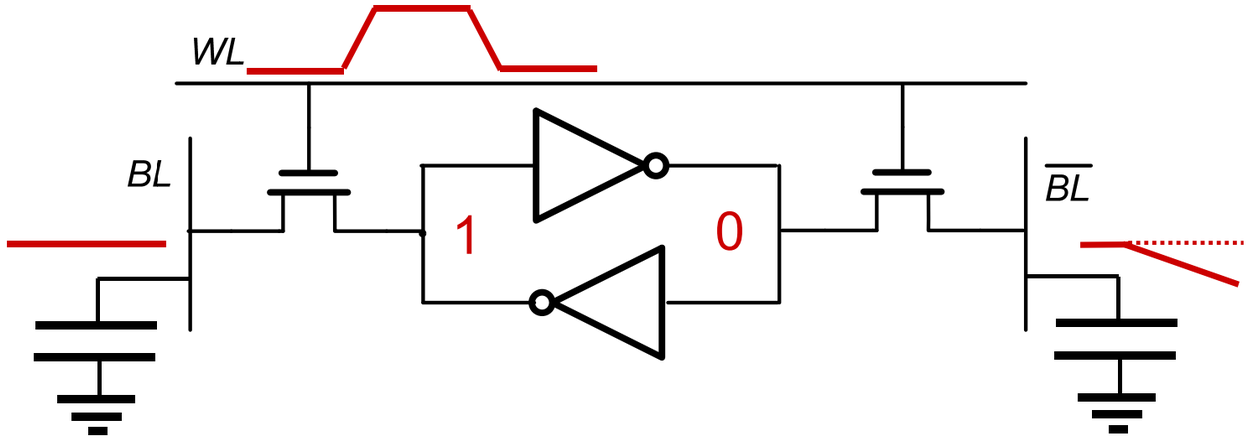


Hold



SRAM Operation

Read

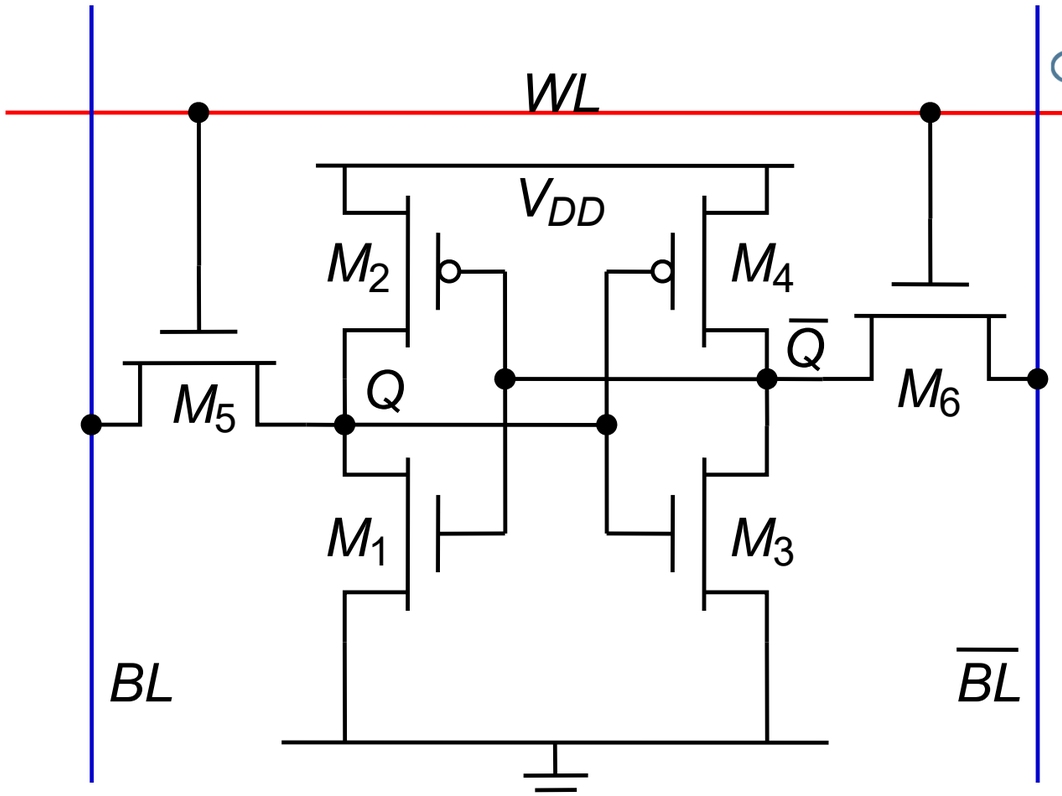


SRAM read in non-destructive

- Reading the cell should not destroy the stored value

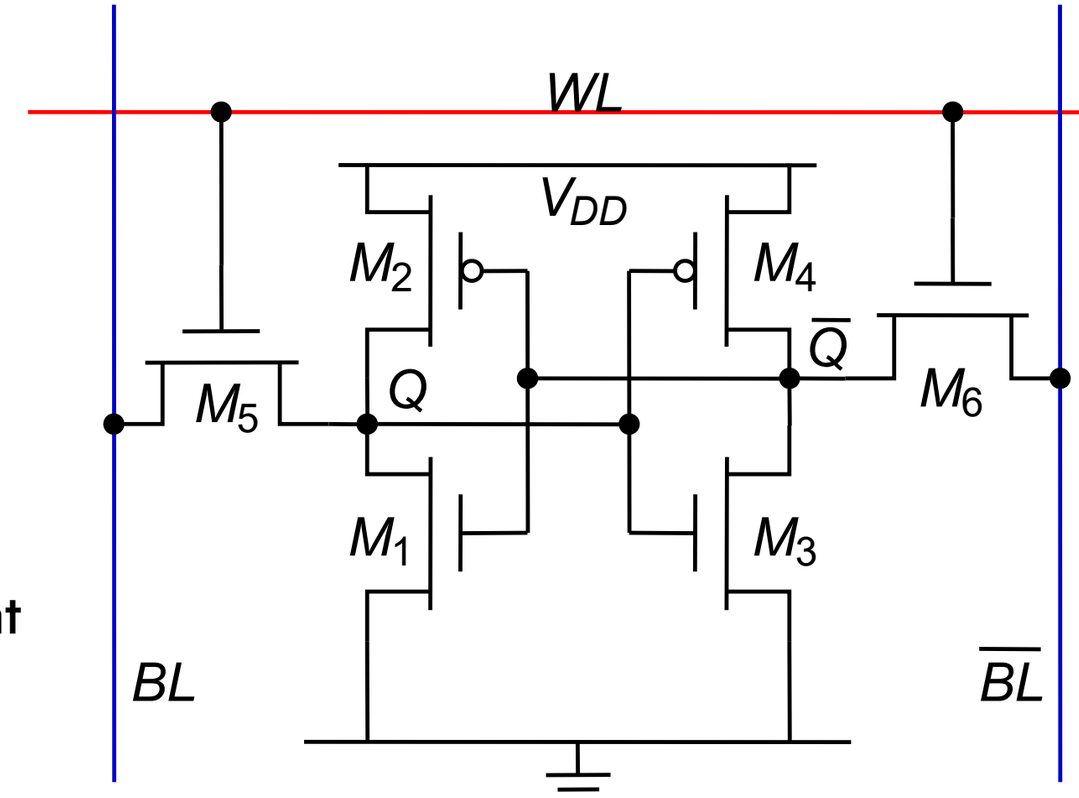
Sizing SRAM Cell

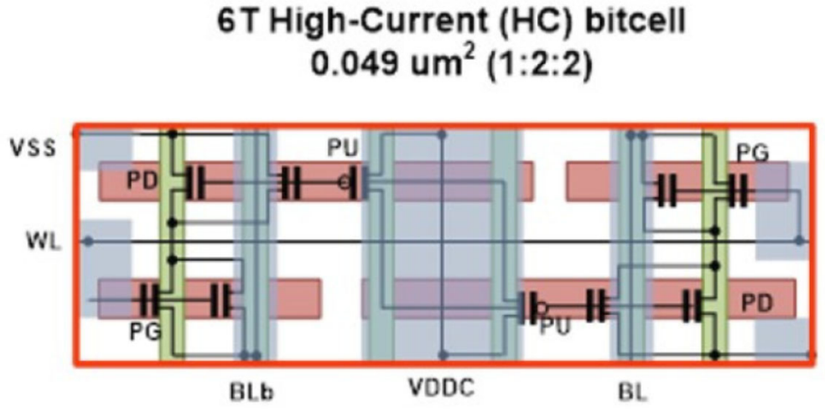
- Read stability: Cell should not change value during read
  - Q = 0: M_5, M_1 both on
  - Voltage divider between M_5, M_1
  - V_Q should stay low, not to flip M_4-M_3 inverter
  - $(W/L)_1 > (W/L)_5$
- Typically $(W/L)_1 = 1.5\ (W/L)_5$
  - In finFETs: $(W/L)_1 = 2(W/L)_5$
- Read speed: Both M_5 and M_1



Sizing SRAM Cell

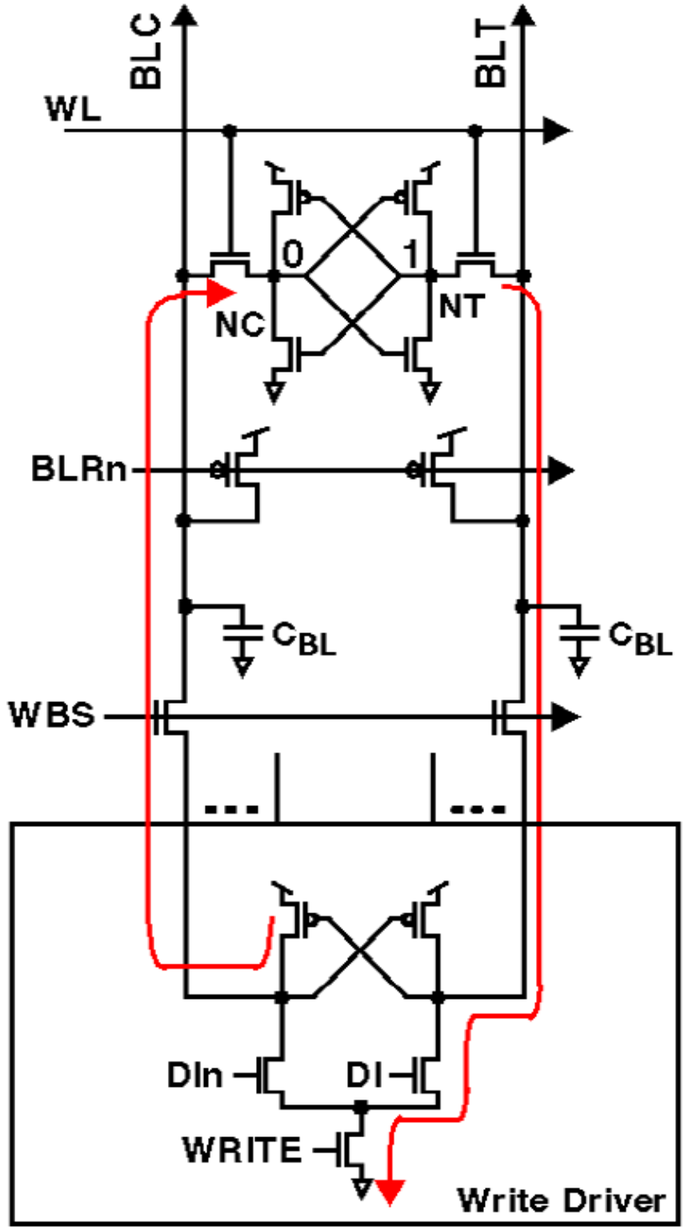
- Writeability: Cell should be writeable by pulling BL low
  - Q = 1, M_5, M_2 both on
  - Voltage divider between M_5, M_2
  - V_Q should pull belowthe switching point of M_4-M_3 inverter
  - $(W/L)_5 > (W/L)_2$
- Typically $(W/L)_5 = (W/L)_2$ in planar
  - In finFETs: $(W/L)_5 = 2(W/L)_2$
  - 1:2:2 and 1:2:3 sizing

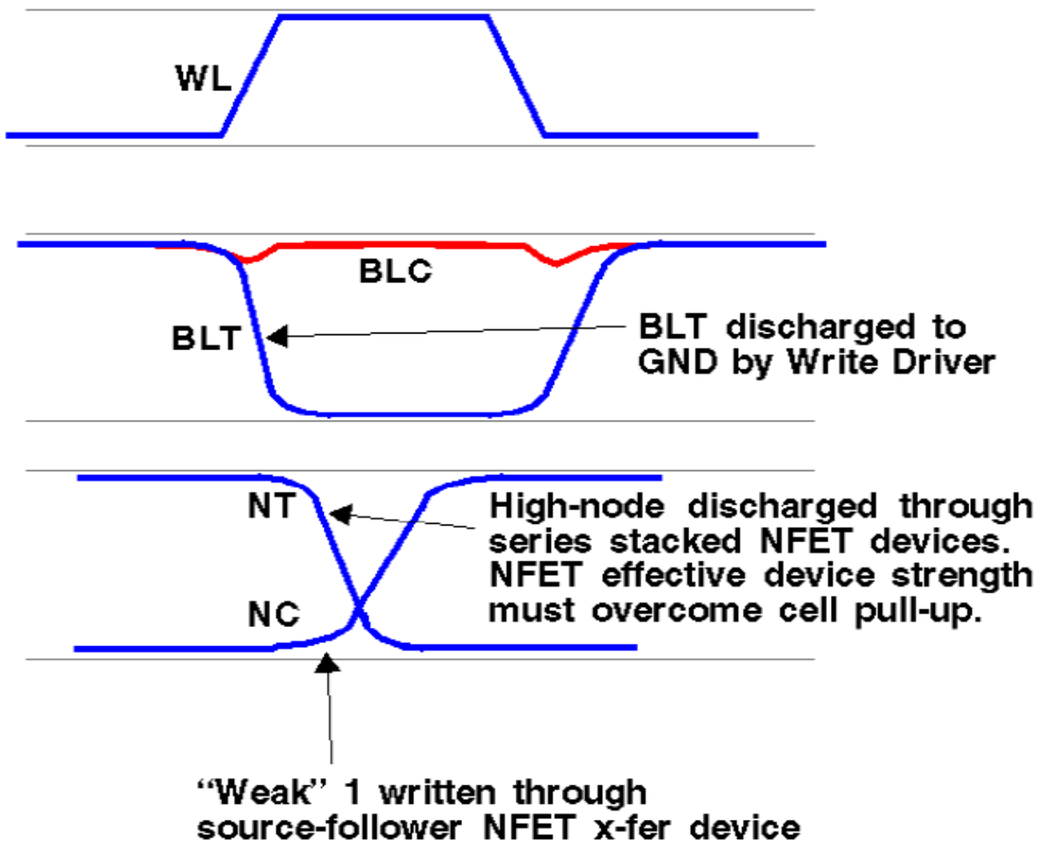




Song, ISSCC'16

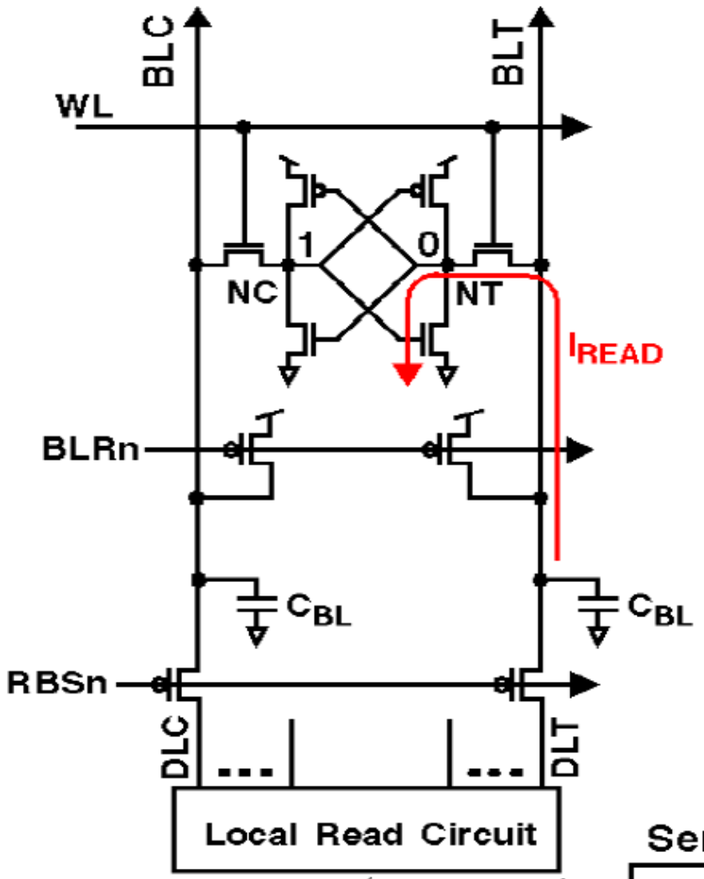
SRAM Column: Write

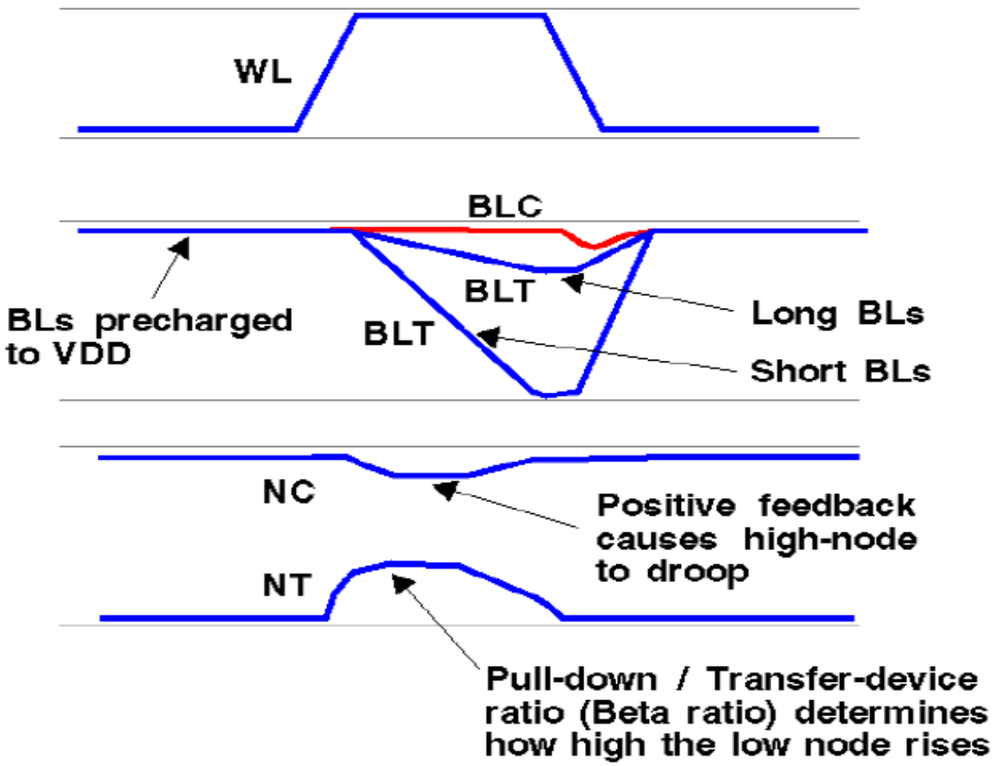




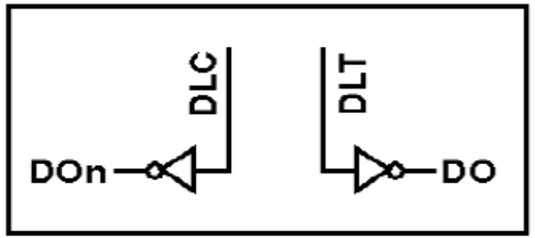
H. Pilo, *IEDM* 2006H. Pilo, *IEDM* 2006

SRAM Column: Read

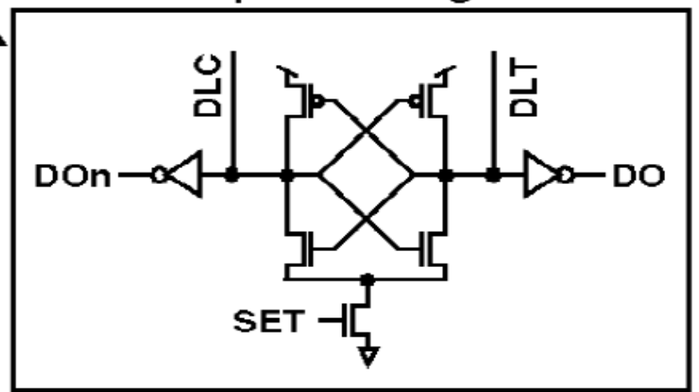




Domino: Short BLs, SOI

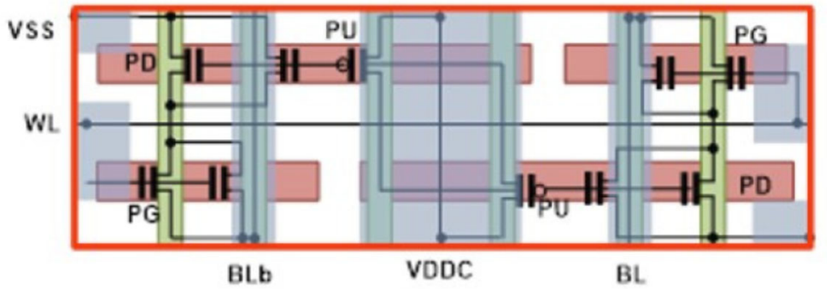


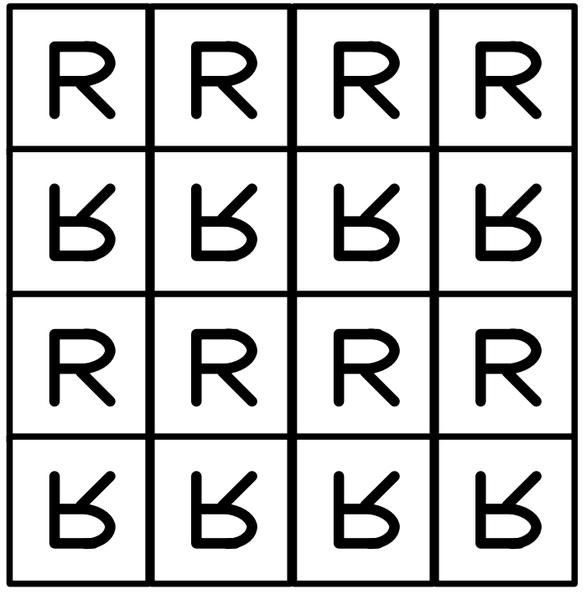
Sense Amplifier: Longer BLs

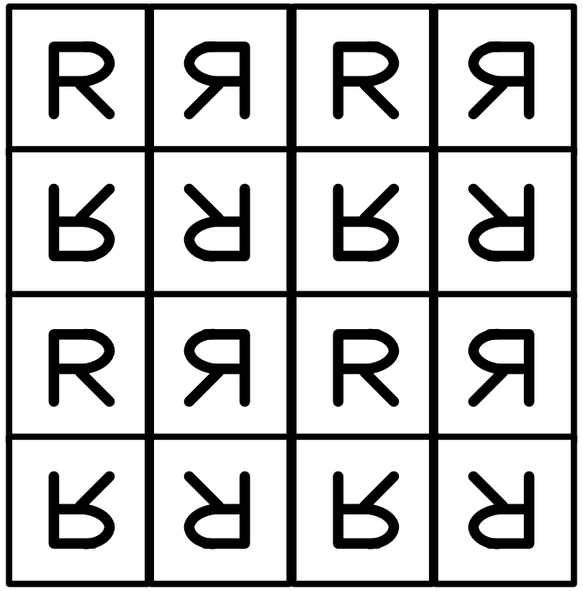


H. Pilo, *IEDM* 2006

SRAM Array Layout

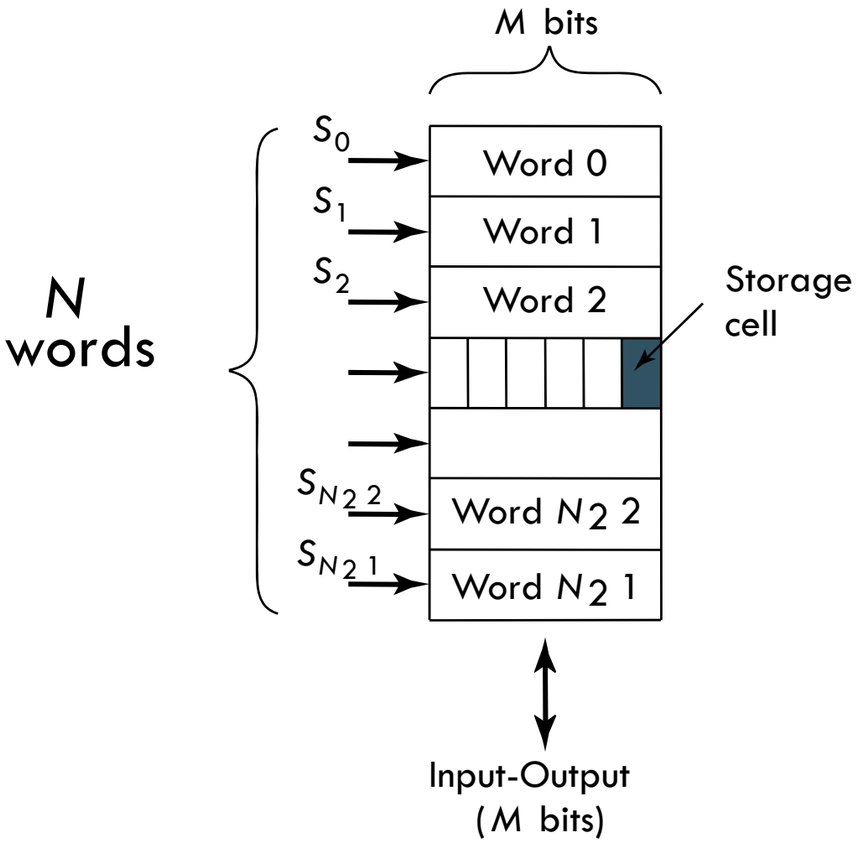


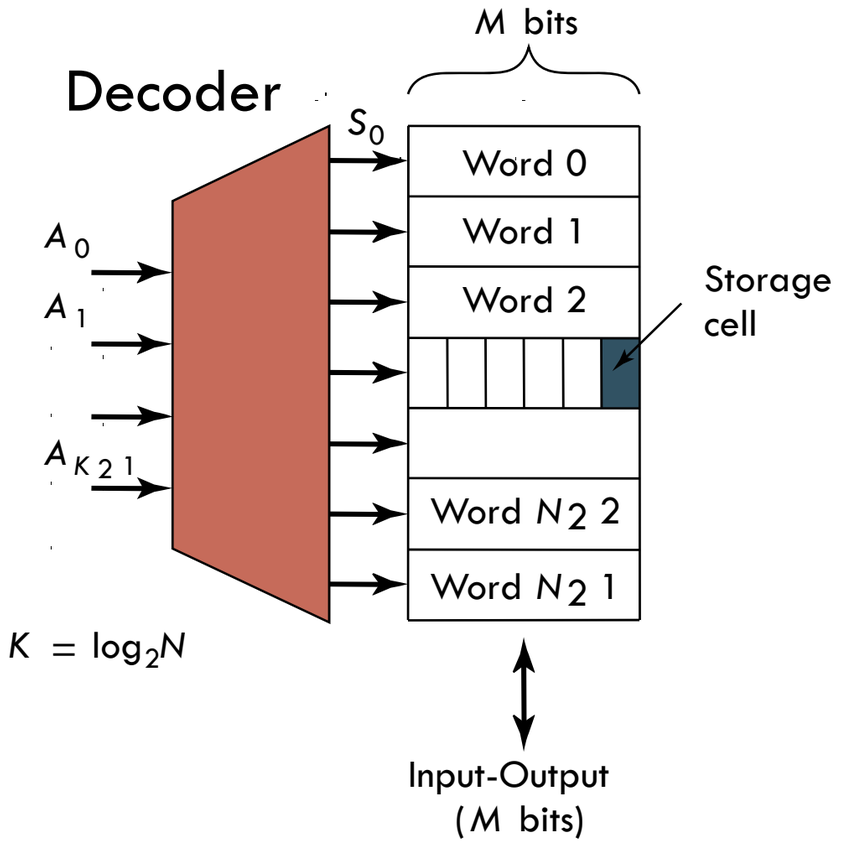




Memory Decoders

Decoders





Intuitive architecture for N x M memory
Too many select signals:
N words = N select signals

Decoder reduces the number of select signals
K = log 2N

Row Decoders

Collection of 2^M complex logic gates
Organized in regular and dense fashion

(N)AND Decoder

$$WL_0 = \overline{A_0}\,\overline{A_1}\,\overline{A_2}\,\overline{A_3}\,\overline{A_4}\,\overline{A_5}\,\overline{A_6}\,\overline{A_7}\,\overline{A_8}\,\overline{A_9}$$

$$WL_{511} = A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 \overline{A_9}$$

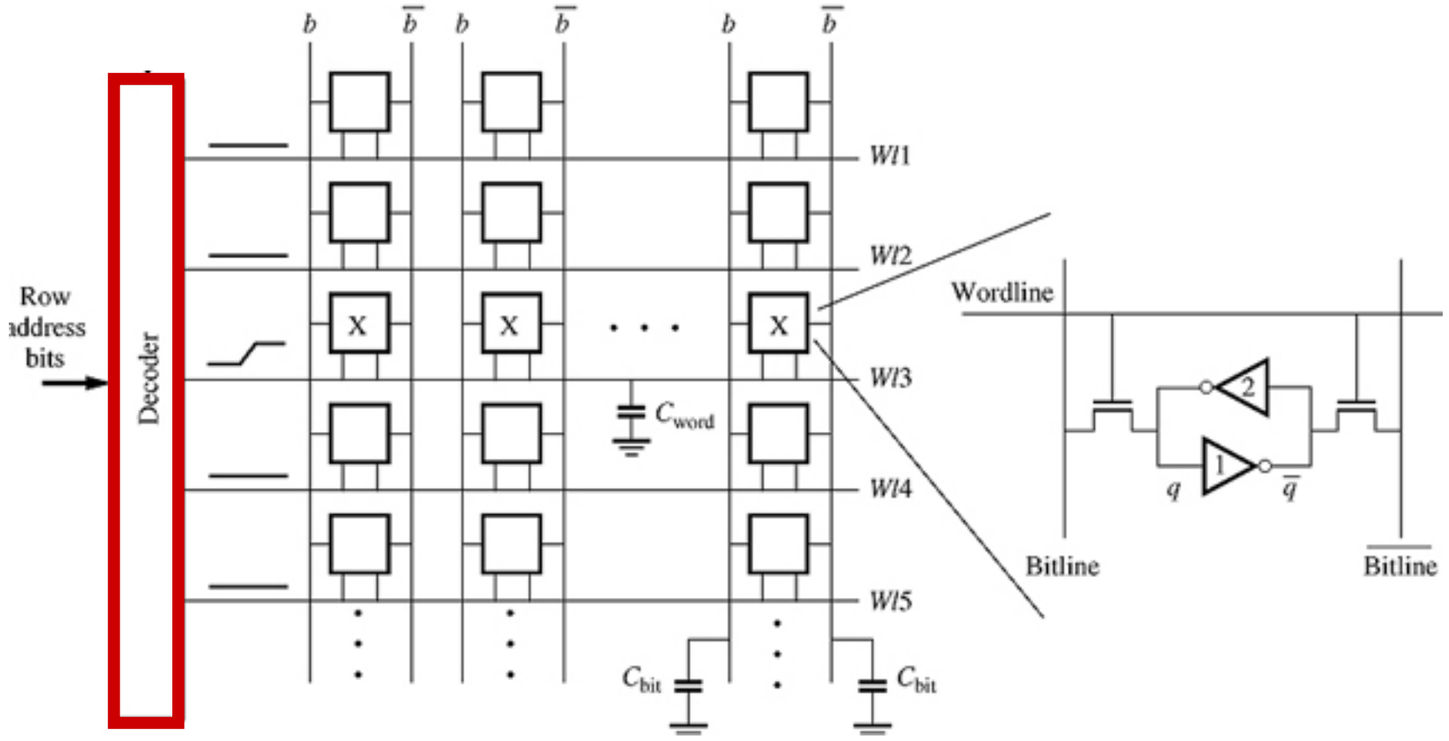
NOR Decoder

$$WL_0 = \overline{A_0 + A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8 + A_9}$$

$$WL_{511} = \overline{\overline{A_0} + \overline{A_1} + \overline{A_2} + \overline{A_3} + \overline{A_4} + \overline{A_5} + \overline{A_6} + \overline{A_7} + \overline{A_8} + A_9}$$

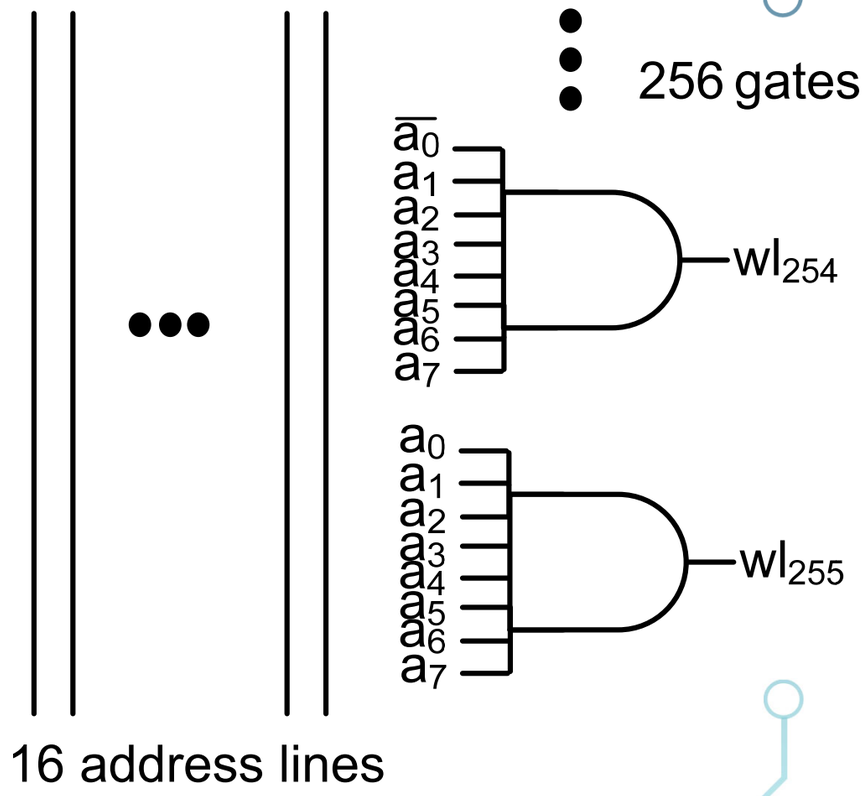
Decoder Design Example

- Look at decoder for 256x256 memory block (8KBytes)



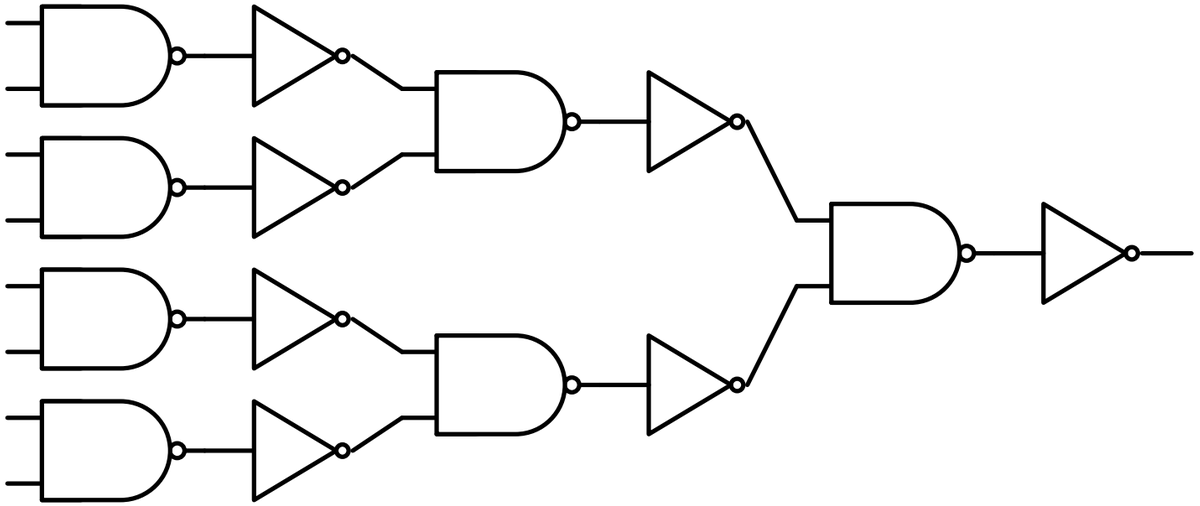
Possible Decoder

- 256 8-input AND gates
  - Each built out of a tree of NAND gates and inverters
- Need to drive a lot of capacitance (SRAM cells)
  - What's the best way to do this?



Possible AND8

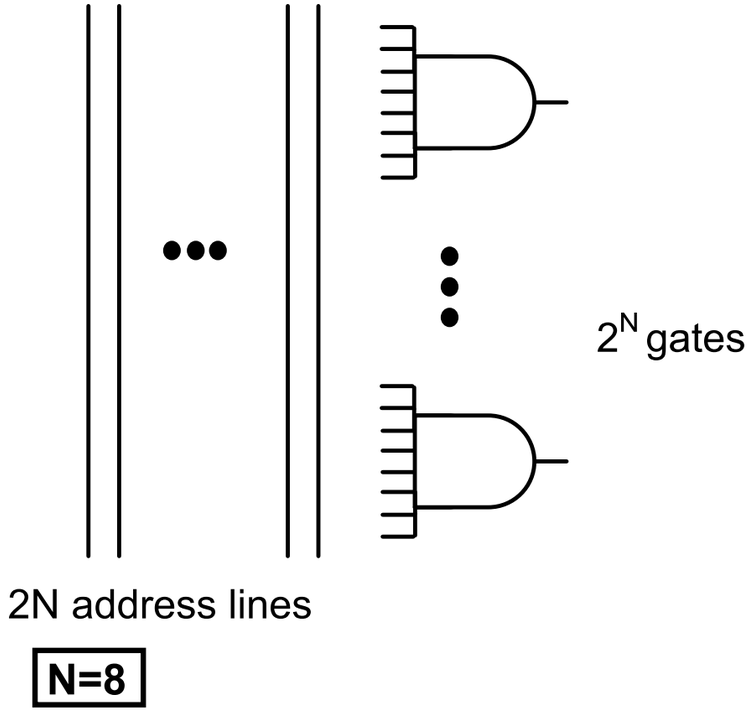
- Build 8-input NAND gate using 2-input gates and inverters
- Is this the best we can do?
- Is this better than using fewer NAND4 gates?



Problem Setup

- Goal: Build fastest possible decoder with static CMOS logic

- What we know
  - Basically need 256 AND gates, each one of them drives one word line



Problem Setup (1)

- Each wordline has 256 cells connected to it
- $C_{WL} = 256*C_{cell} + C_{wire}$
  - Ignore wire for now
- Assume that decoder input capacitance is $C_{address}=4*C_{cell}$

Problem Setup (2)

- Each address bit drives $2^8/2$ AND gates
  - A0 drives ½ of the gates, A0_b the other ½ of the gates

Problem Setup (3)

- Total fanout on each address wire is:

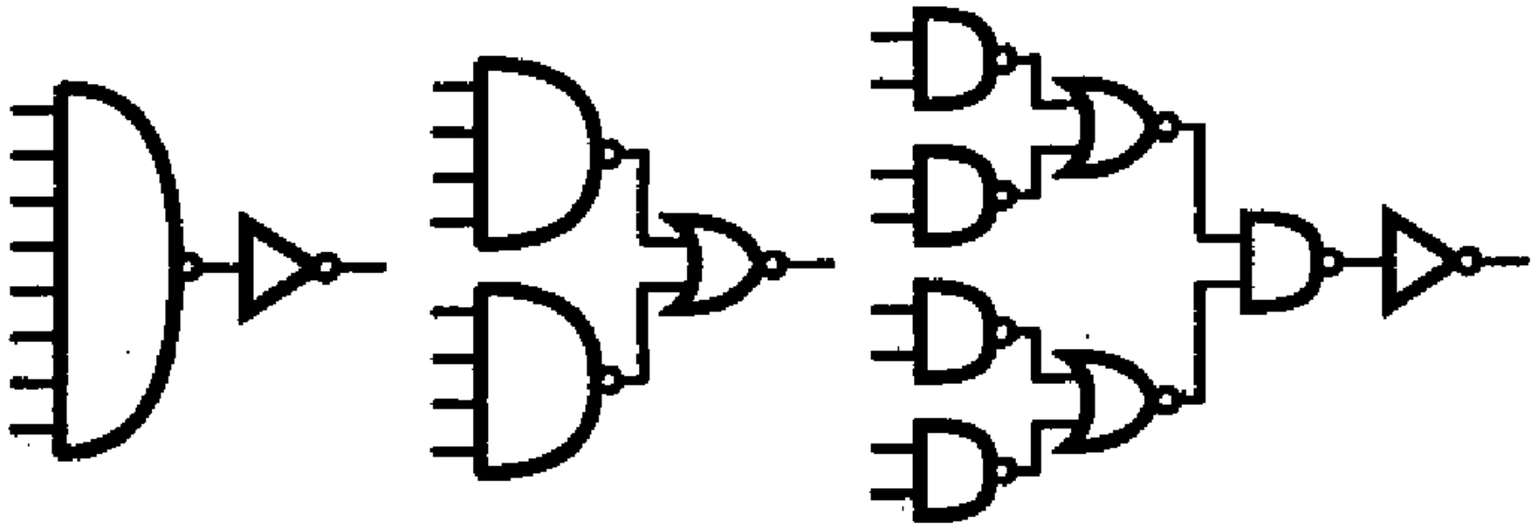
$$F = \Pi B \frac{C_{load}}{C_{in}} = 128 \frac{\left(256 C_{cell}\right)}{4 C_{cell}} = 2^7 \frac{\left(2^8 C_{cell}\right)}{2^2 C_{cell}} = 2^{13}$$

Decoder Fan-Out

- F of 2^{13} means that we will want to use more than $\log_4(2^{13}) = 6.5$ stages to implement the AND8

- Need many stages anyways
  - So what is the best way to implement the AND gate?
  - Will see next that it's the one with the most stages and least complicated gates

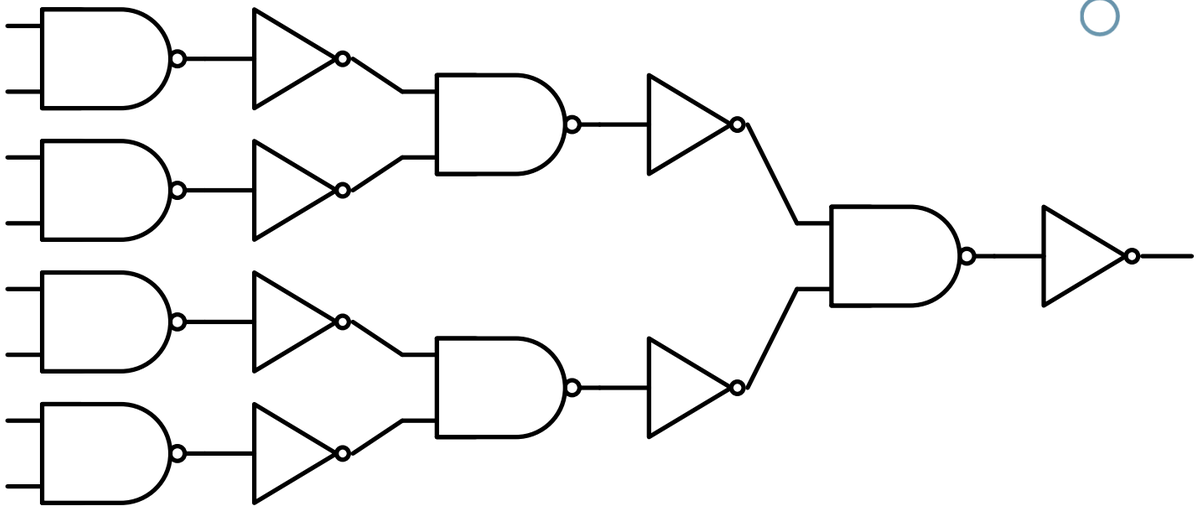
8-Input AND



LE : 9/2 1	***LE*** : 5/2 3/2	***LE*** : 3/2 3/2 3/2 1
ΠLE = 9/2	ΠLE = 15/4	ΠLE = 27/8
P = 8 + 1	P = 4 + 2	P = 2 + 2 + 2 + 1

8-Input AND

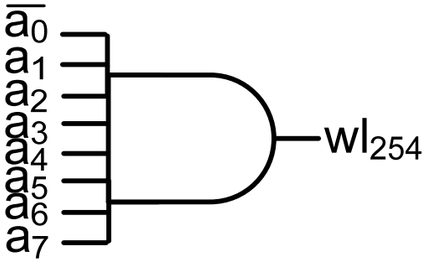
- Using 2-input NAND gates
  - 8-input gate takes 6 stages
- Total LE is $(3/2)^3 \approx 3.4$
- So PE is $3.4*2^{13}$ – optimal N of ~7.4

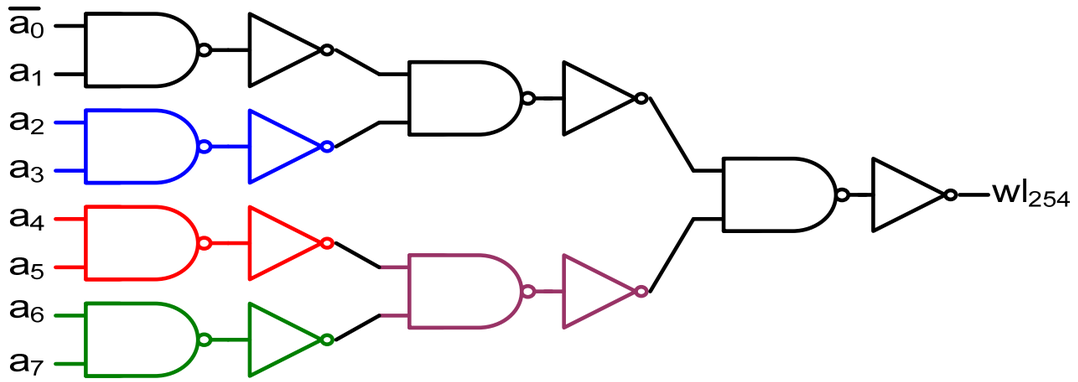


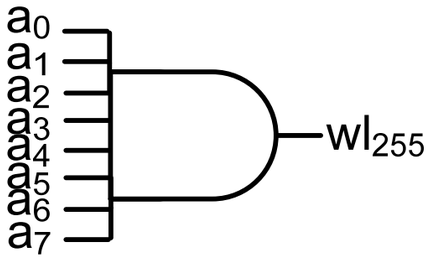
Decoder So Far

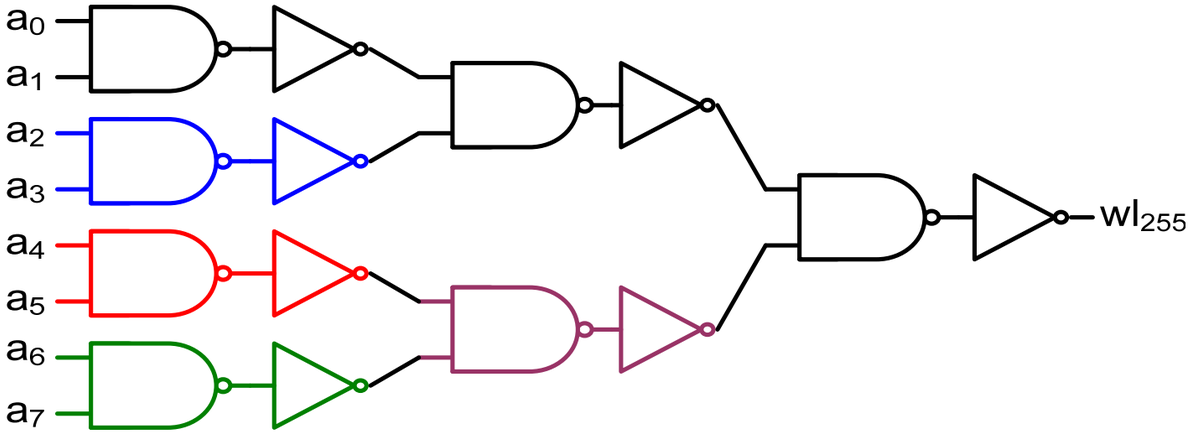
- 256 8-input AND gates
  - Each built out of tree of NAND gates and inverters
- Issue:
  - Every address line has to drive 128 gates (and wire) right away
  - Forces us to add buffers just to drive address inputs

Look Inside Each AND8 Gate







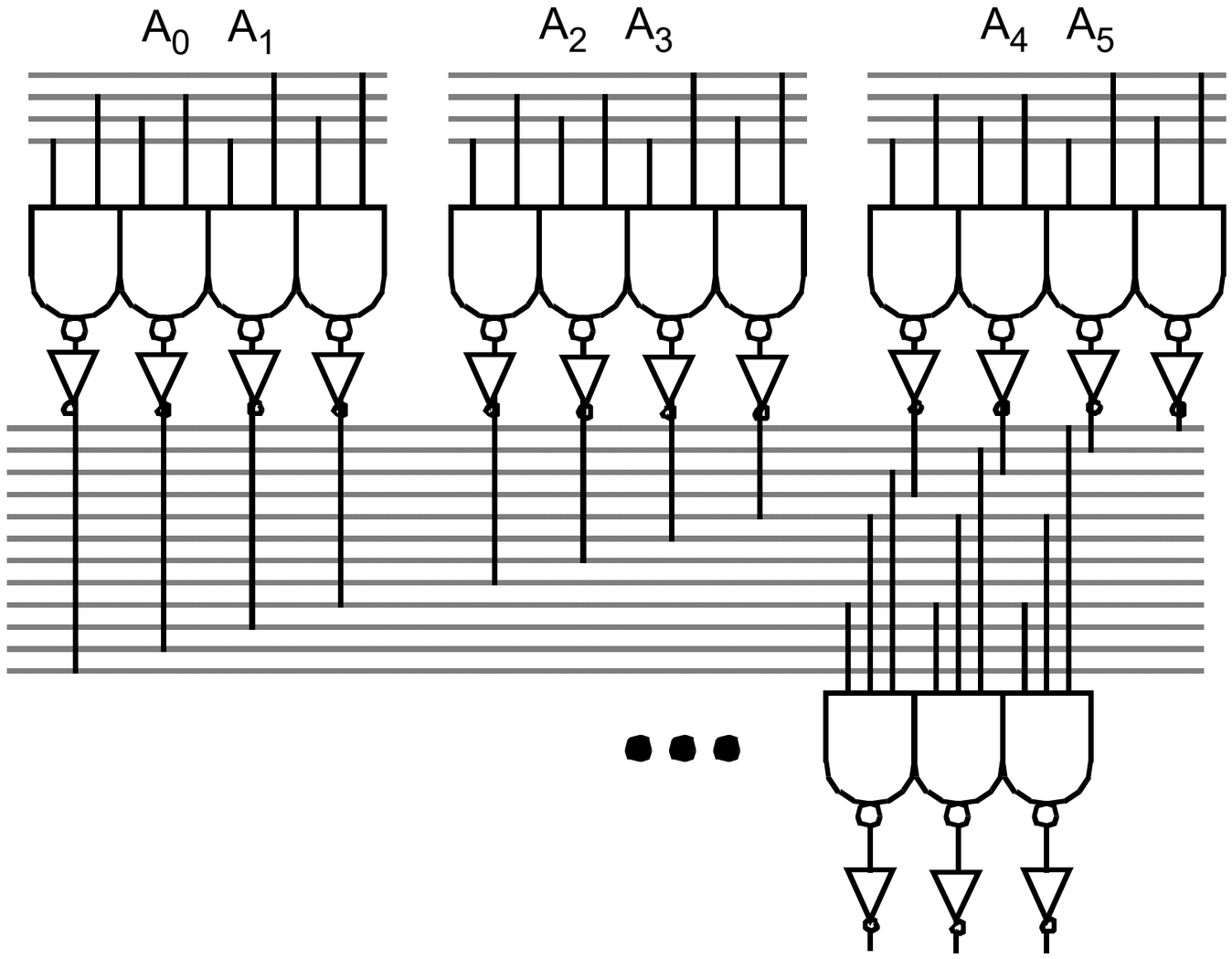


Predecoders

- Use a single gate for each of the shared terms
  - E.g., from A_0, $\overline{A_0}$, A_1, and $\overline{A_1}$, generate four signals: A_0A_1, $\overline{A_0}A_1$, $A_0\overline{A_1}$, $\overline{A_0}\,\overline{A_1}$

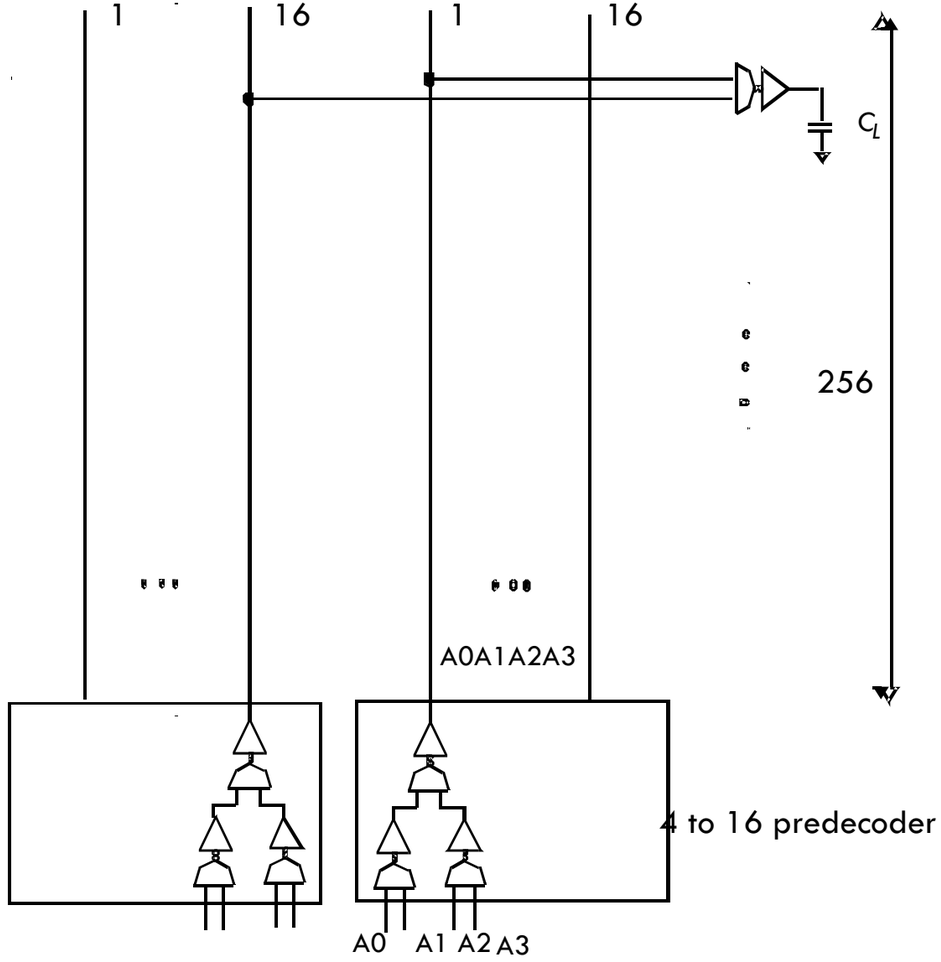
- In other words, we are decoding smaller groups of address bits first
  - And using the "predecoded" outputs to do the rest of the decoding

Predecoder and Decoder



Predecode Options

- Larger predecode usually better:
- More stages before the long wires
  - Decreases their effect on the circuit
- Fewer number of long wires switches
  - Lower power
- Easier to fit 2-input gate into cell pitch



Review

- Flip-flop is typically a latch pair
- Setup and hold times are defined at constant percentage increases over clk-q delay
- SRAM has unique combination of density, speed, power
- SRAM cells sized for stability and writeability