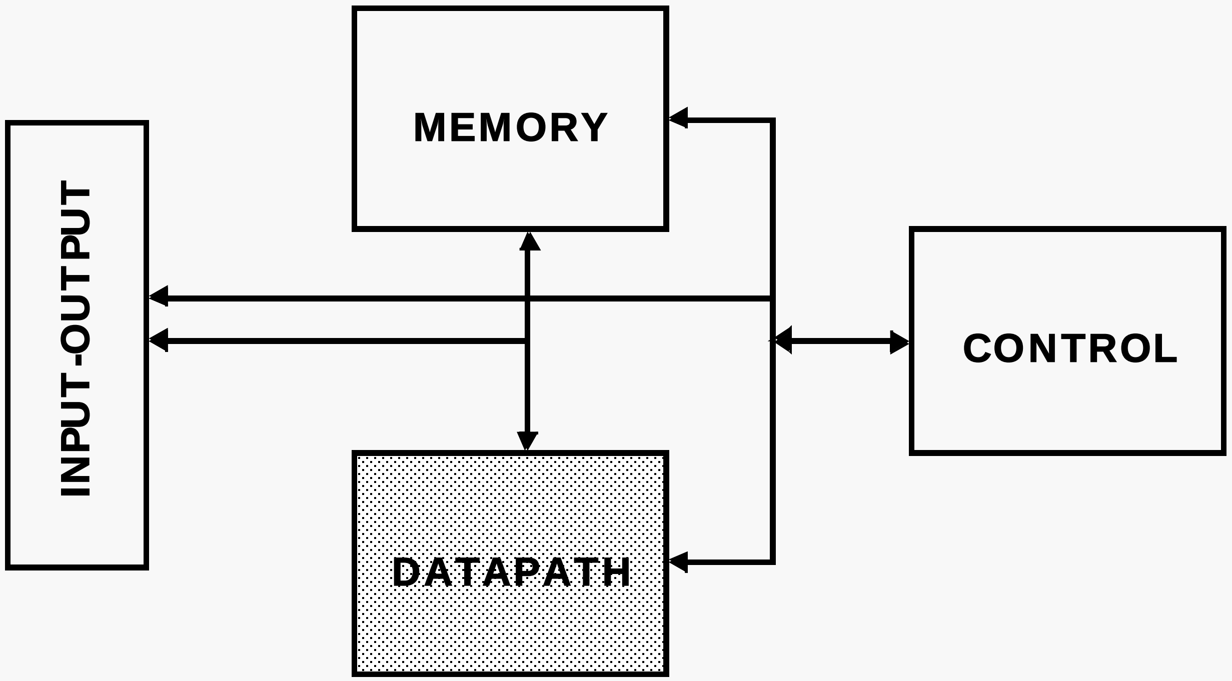
Design of System Elements

Basics of VLSI

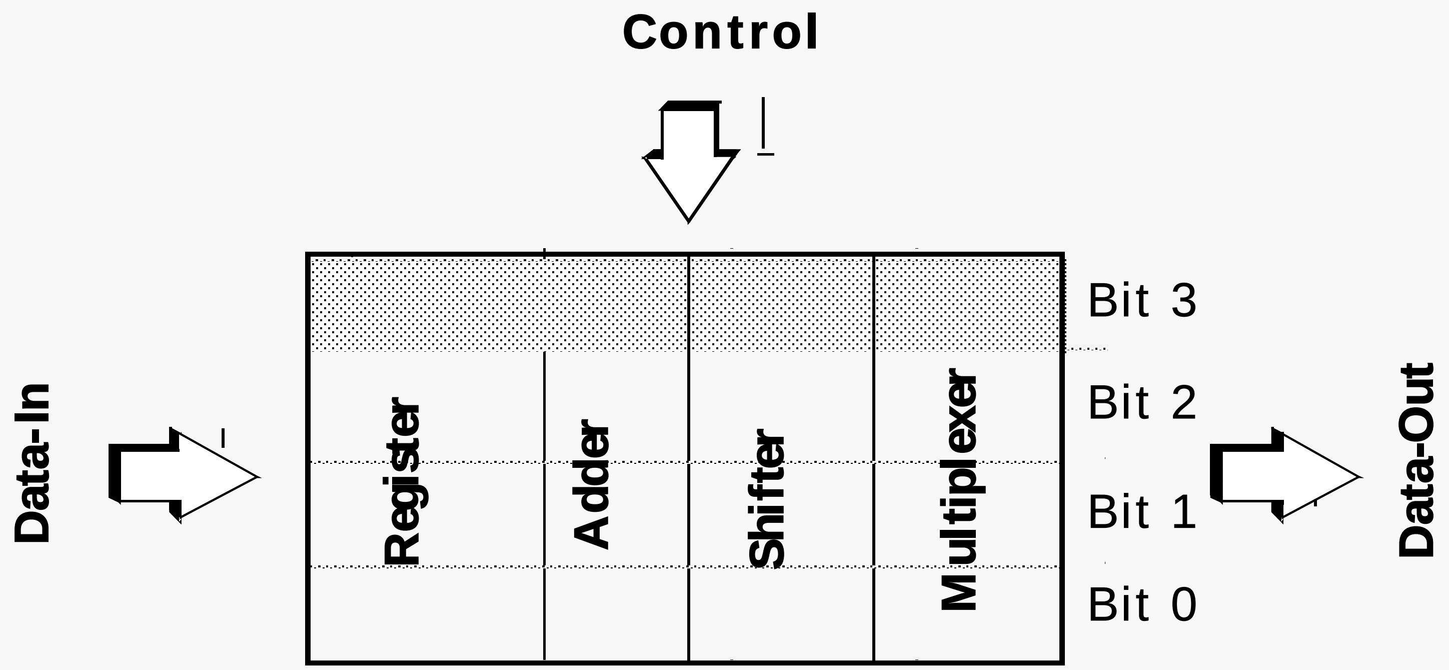
A Generic Digital Machine



Building Blocks for Digital Architectures

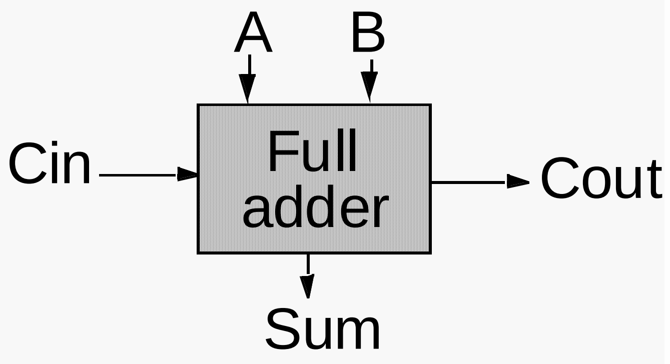
- Arithmetic unit
  - Data path (adder, multiplier, shifter, comparator): Bit slice technique
- Memory
  - RAM, ROM, PROM, Flash Memory.......
- Random logic
  - Finite state machine, Switches, Arbiters
- Data vehicle
  - Buses for data, address and control

Bit-Sliced Design



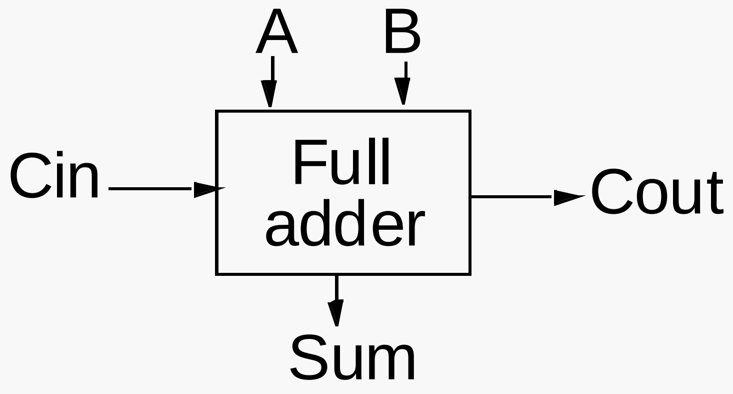
Tile identical processing elements

Full-Adder



A	B	C_i	S	C_o	*Carry status*
0	0	0	0	0	delete
0	0	1	1	0	delete
0	1	0	1	0	propagate
0	1	1	0	1	propagate
1	0	0	1	0	propagate
1	0	1	0	1	propagate
1	1	0	0	1	generate
1	1	1	1	1	generate

The Binary Adder



$$S = A \oplus B \oplus C_i$$

$$= A\bar{B}\bar{C}_i + \bar{A}B\bar{C}_i + \bar{A}\bar{B}C_i + ABC_i$$

$$C_o = AB + BC_i + AC_i$$

Express Sum and Carry as a function of P, G, D

Define 3 new variable which ONLY depend on A, B

Generate (G) = AB

Propagate (P) = A ⊕ B

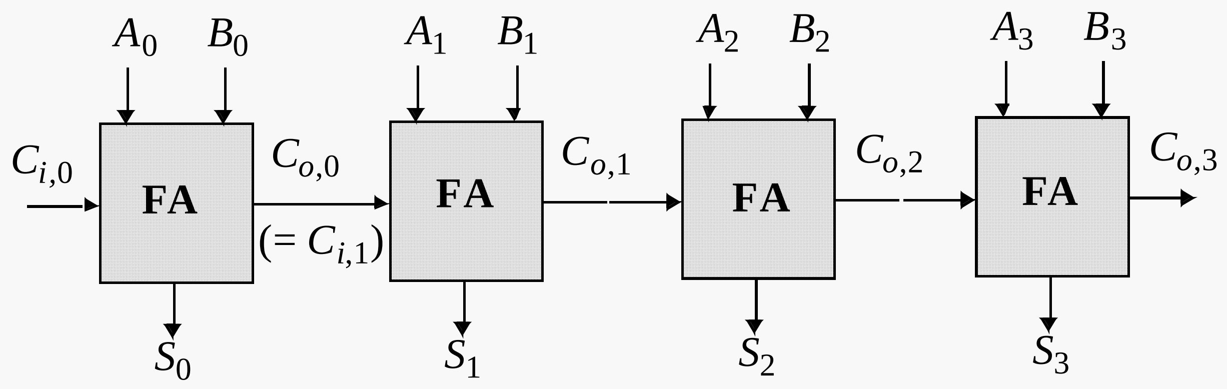
Delete = $\overline{A}\,\overline{B}$

$$C_o(G, P) = G + PC_i$$

$$S(G, P) = P \oplus C_i$$

Can also derive expressions for *S* and C_o based on *D* and *P*

The Ripple-Carry Adder



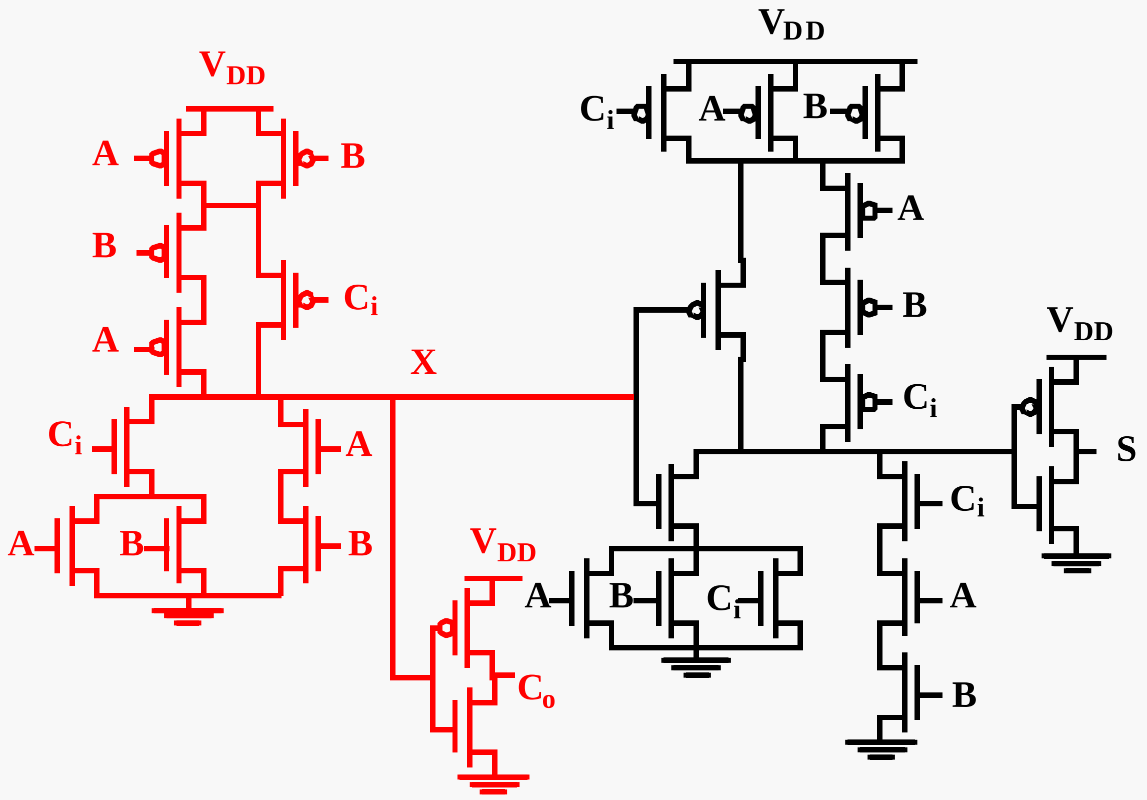
Worst case delay linear with the number of bits

$t_d = O(N)$

$$\mathbf{t_{adder} \approx (N-1)t_{carry} + t_{sum}}$$

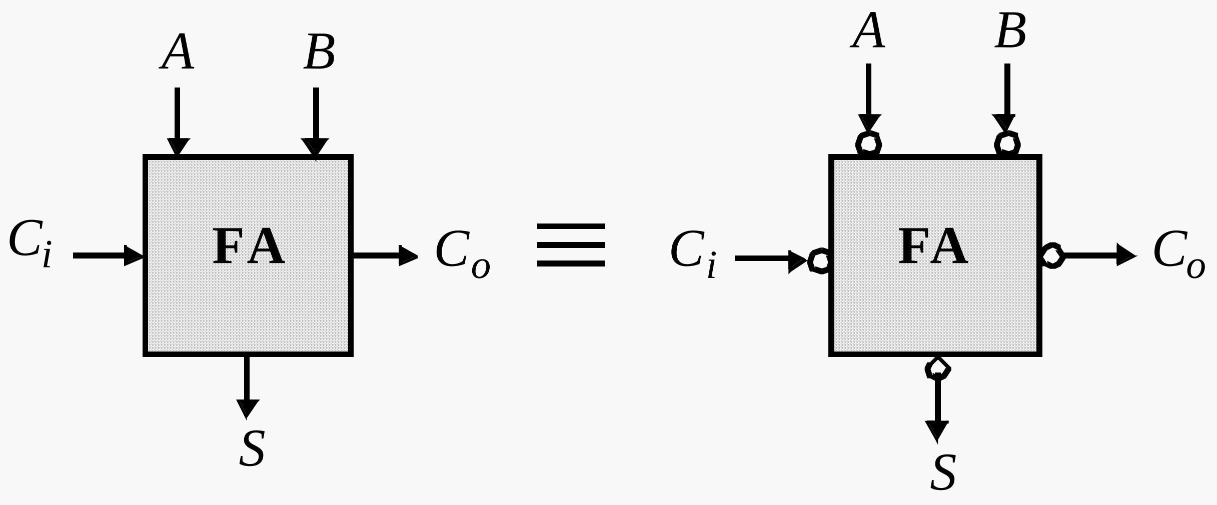
Goal: Make the fastest possible carry path circuit

Complimentary Static CMOS Full Adder



28 Transistors

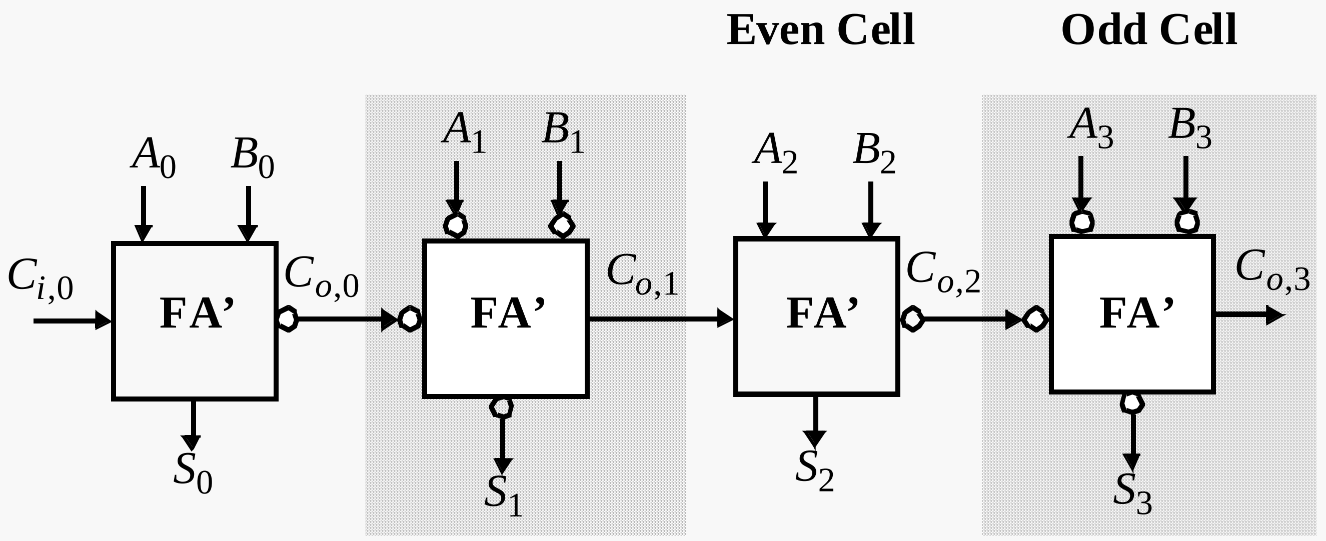
Inversion Property



$$\bar{S}(A, B, C_i) = S(\bar{A}, \bar{B}, \bar{C_i})$$

$$\overline{C_o}(A, B, C_i) = C_o(\bar{A}, \bar{B}, \bar{C_i})$$

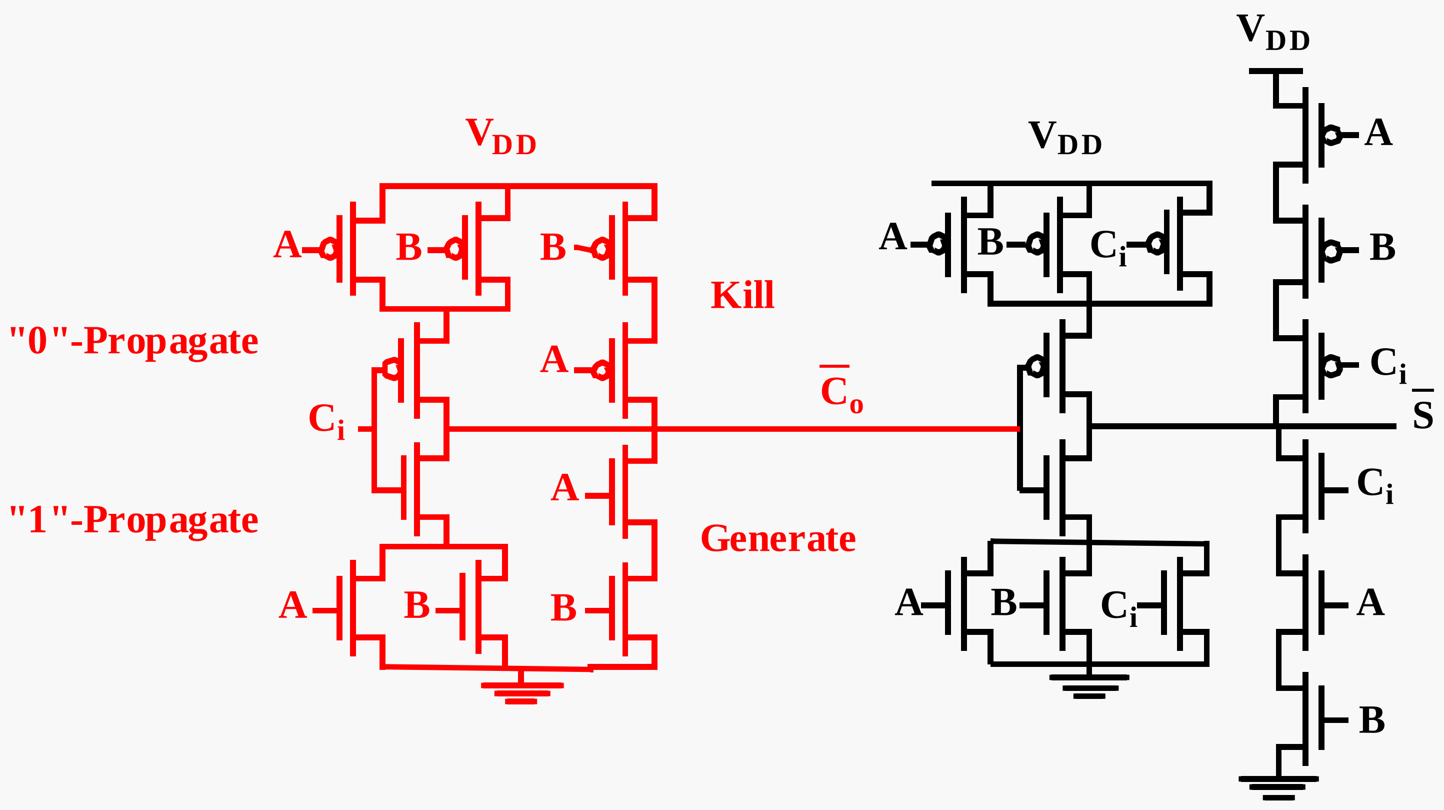
Minimize Critical Path by Reducing Inverting Stages



Exploit Inversion Property

Note: need 2 different types of cells

The better structure: the Mirror Adder

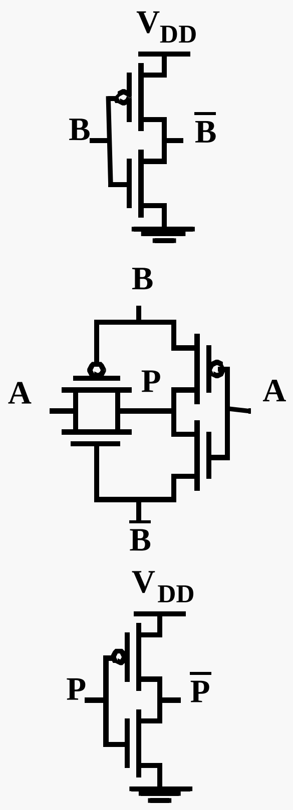


24 transistors

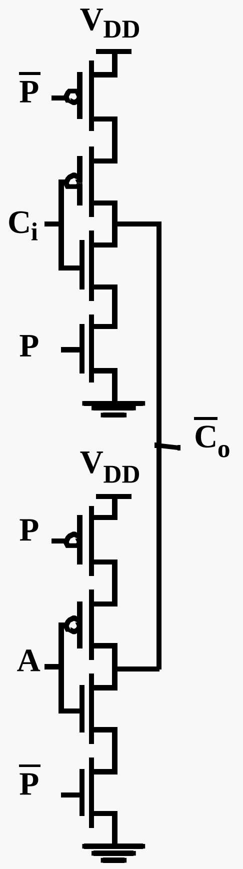
The Mirror Adder

- The NMOS and PMOS chains are completely symmetrical
  - guarantees identical rising and falling transitions if devices are properly sized
  - A maximum of two series transistors can be observed in the carry-generation circuitry.
- When laying out the cell, the most critical issue is the minimization of the capacitance at node C_o
  - reduction of the diffusion capacitances is particularly important.
- The capacitance at node C_o = 4 diffusion capacitances + 2 internal gate capacitances + 6 gate capacitances in the connecting adder cell
- The transistors connected to C_i are placed closest to the output
- Only the transistors in the carry stage have to be optimized for optimal speed
  - All transistors in the sum stage can be minimal size.

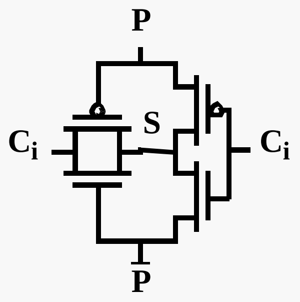
Quasi-Clocked Adder



Signal Setup

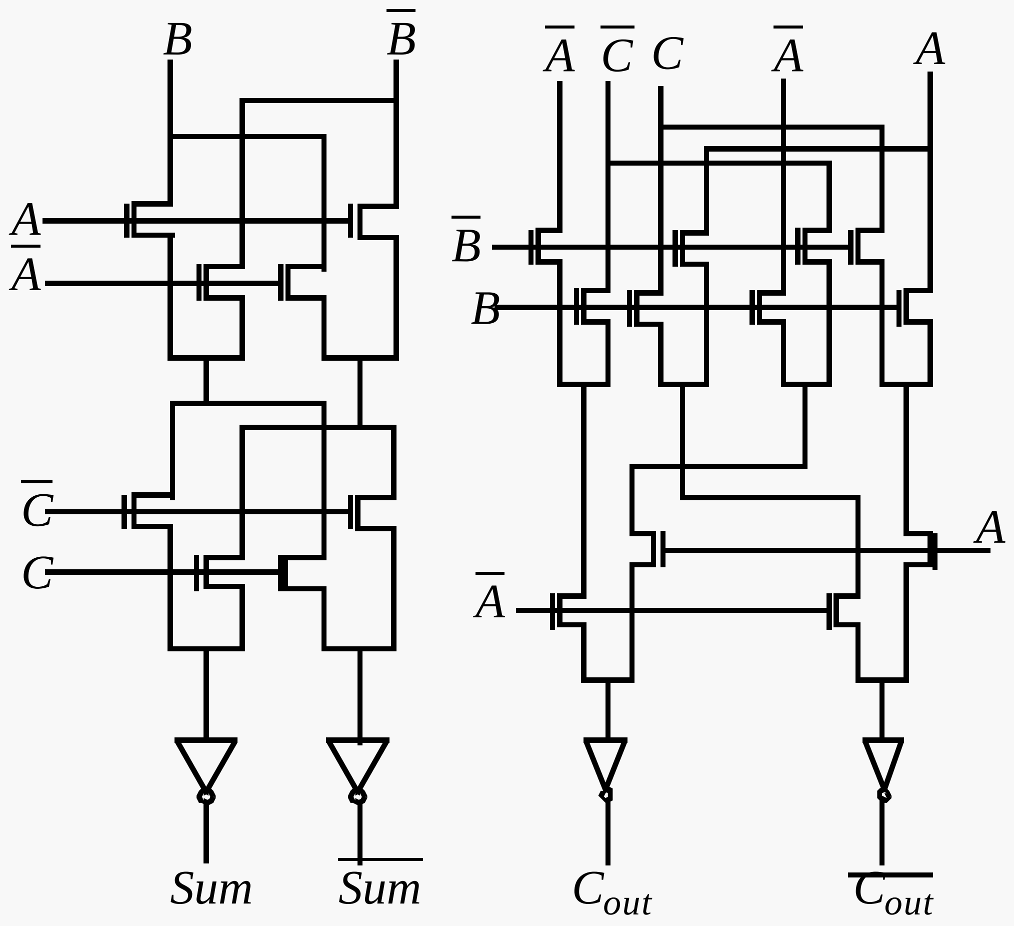


Carry Generation



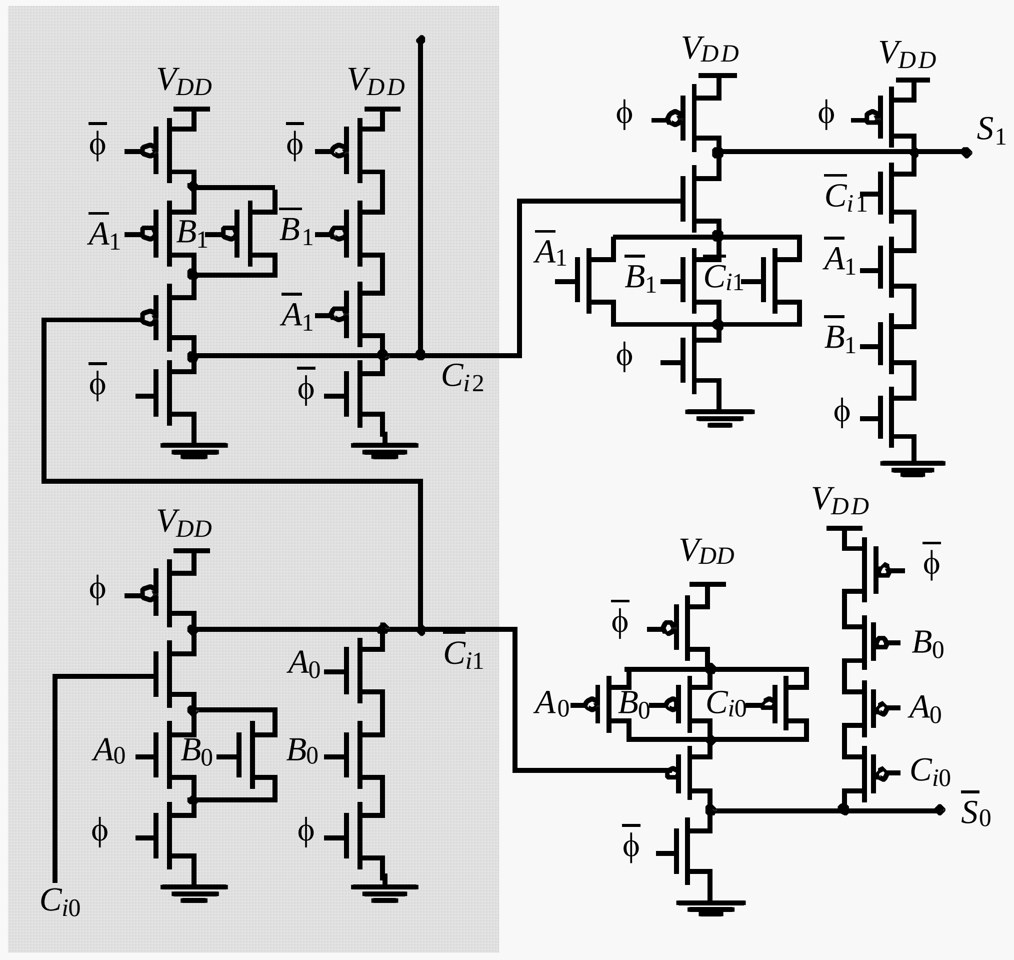
Sum Generation

NMOS-Only Pass Transistor Logic



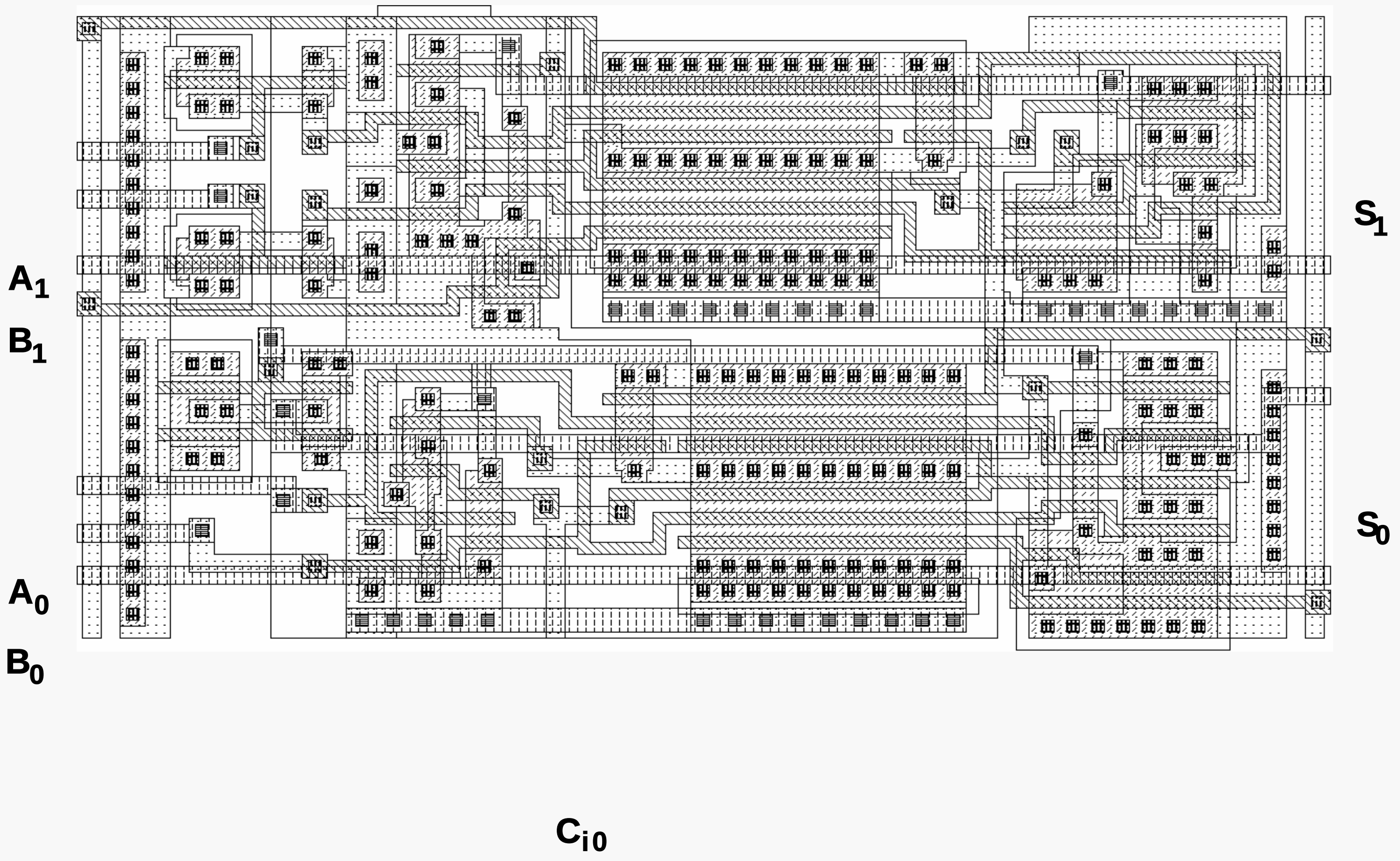
Transistor count (CPL) : 28

NP-CMOS Adder

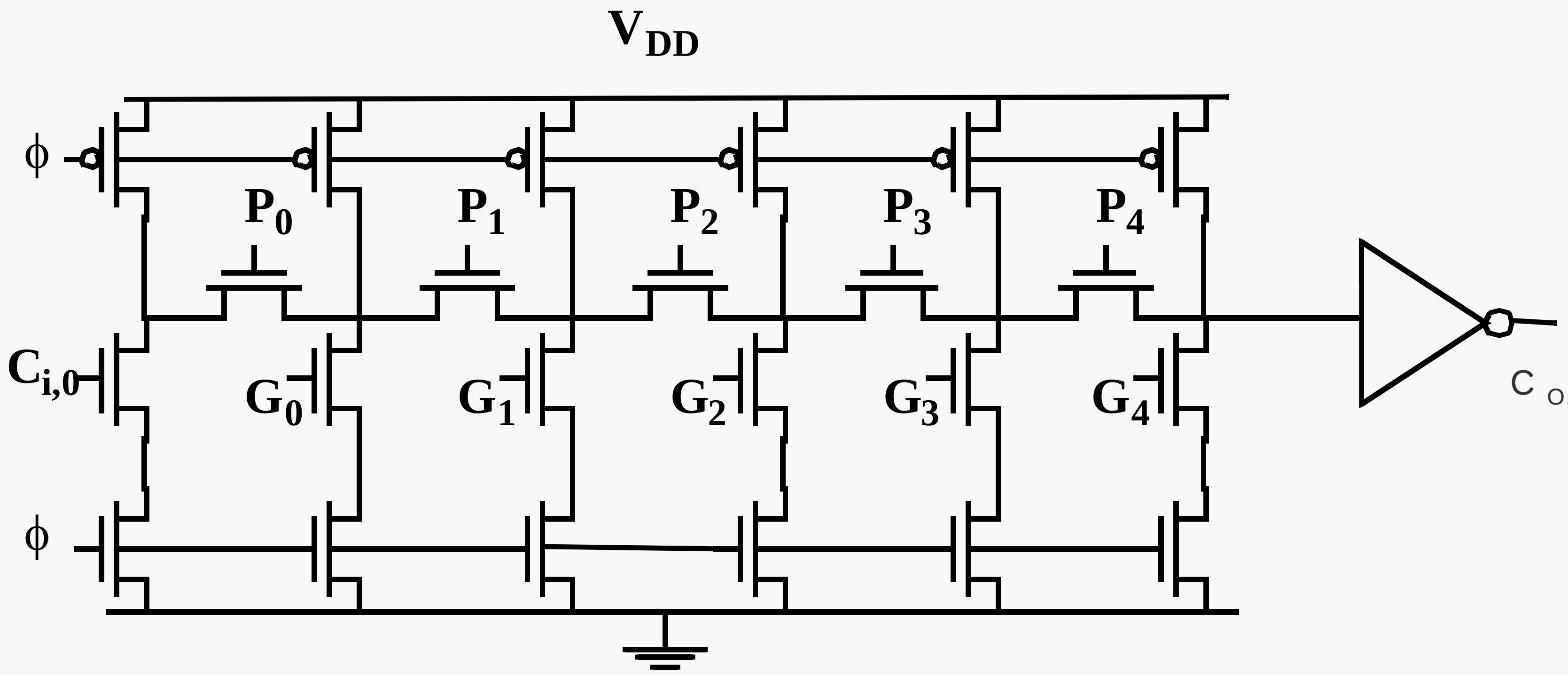


Carry Path

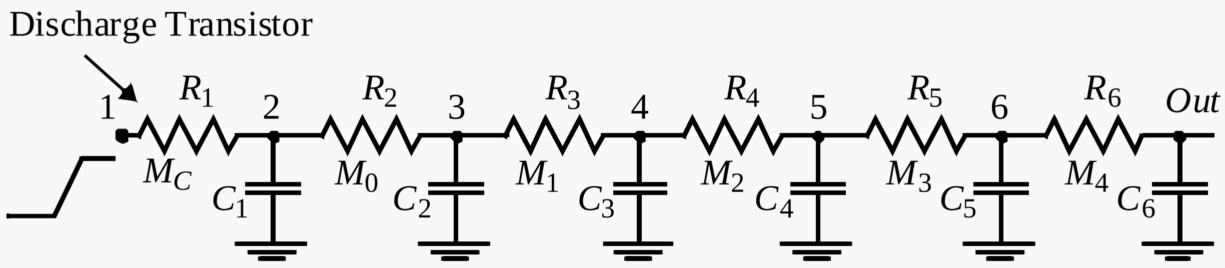
NP-CMOS Adder



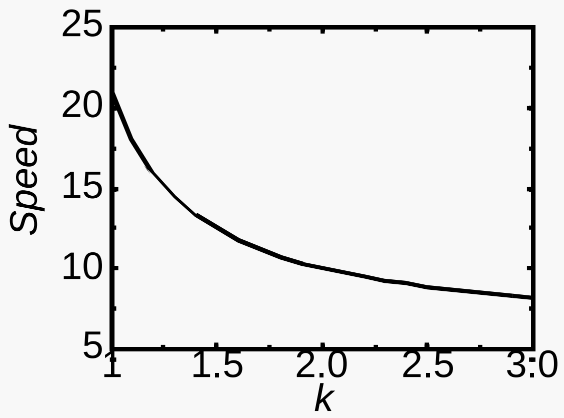
Manchester Carry Chain



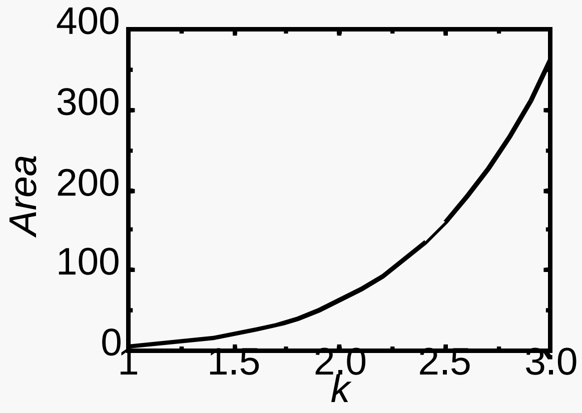
Sizing Manchester Carry Chain



$$t_p = 0.69 \sum_{i=1}^{N} C_i \left(\sum_{j=1}^{i} R_j \right)$$

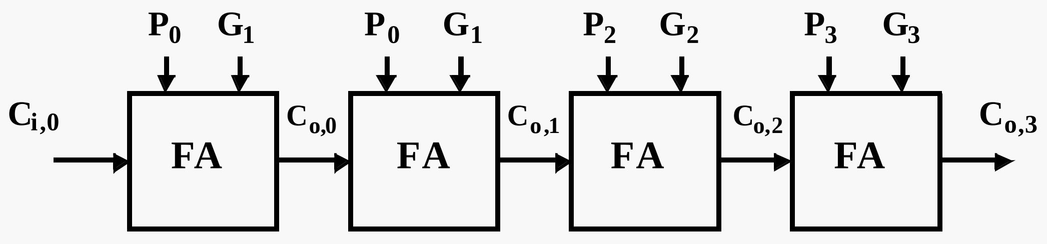


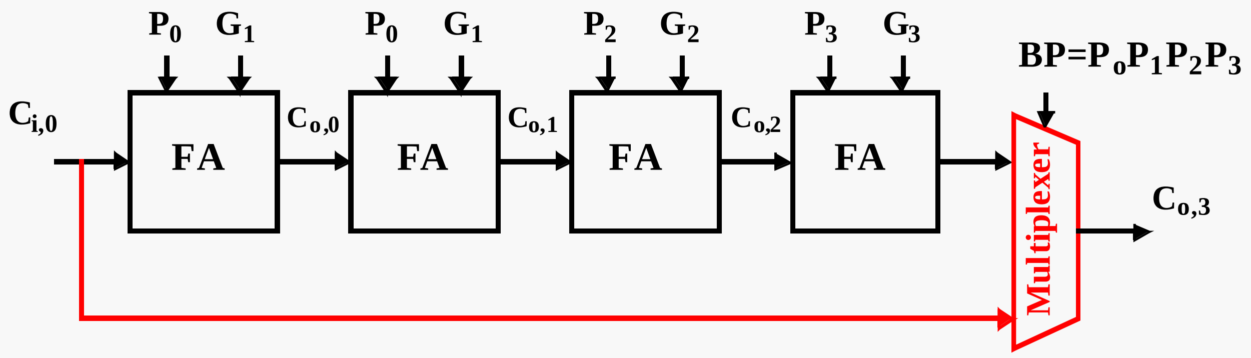
Speed (normalized by 0.69*RC*)



Area (in minimum size devices)

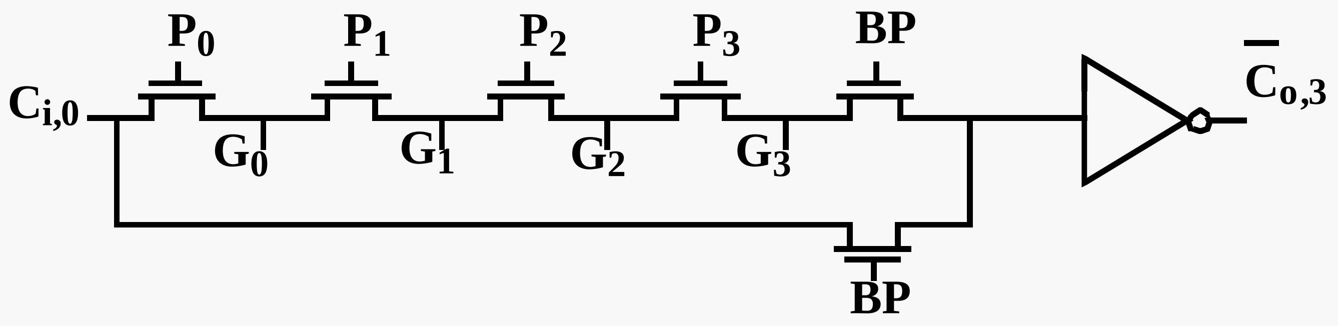
Carry-Bypass Adder



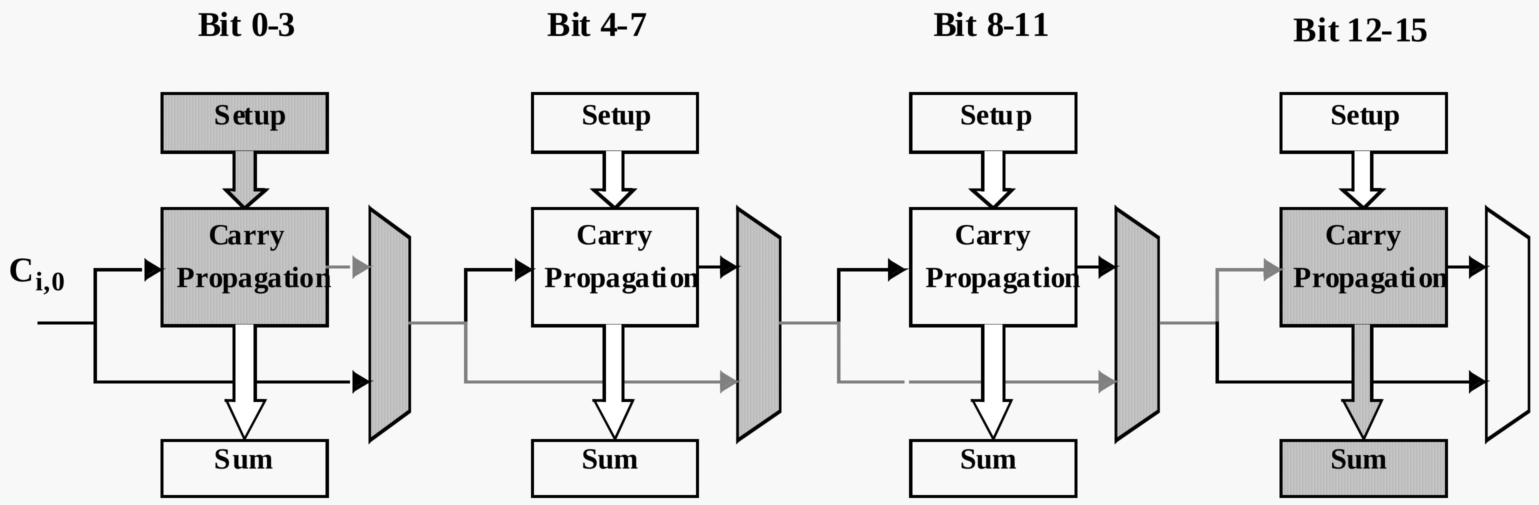


Idea: If (P0 and P1 and P2 and P3 = 1)
then $C_{o3} = C_0$, else “kill” or “generate”.

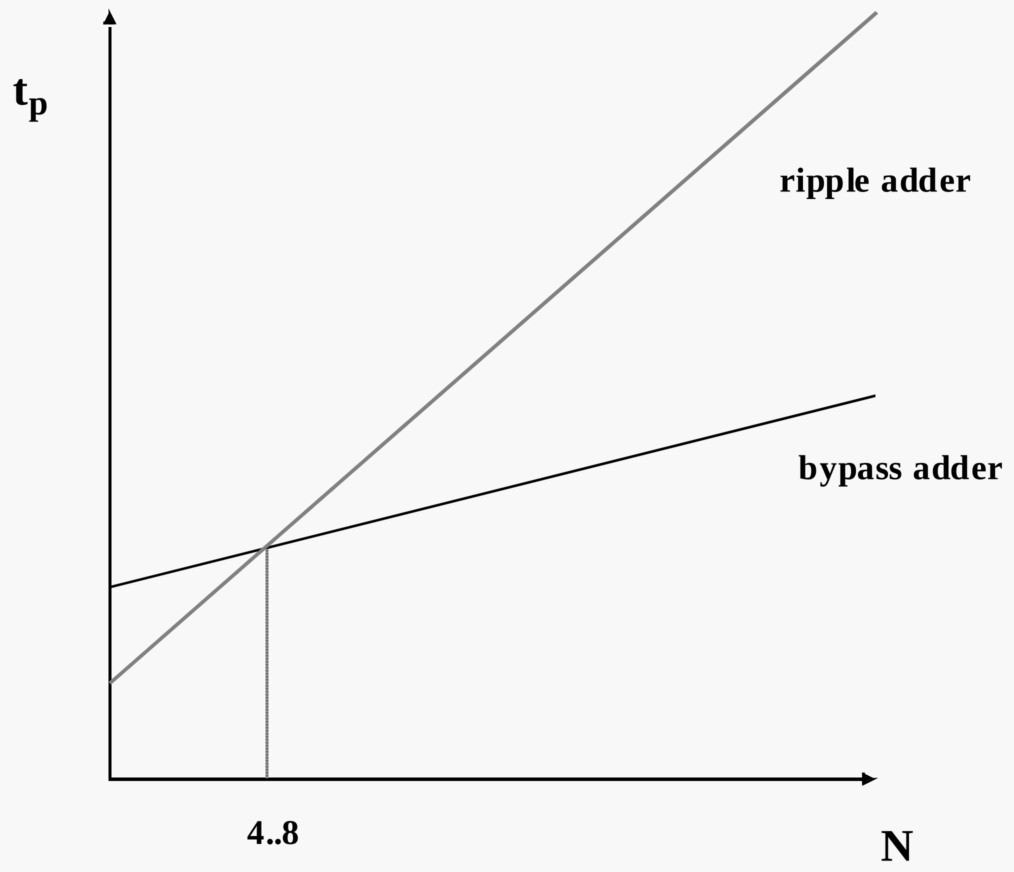
Manchester-Carry Implementation



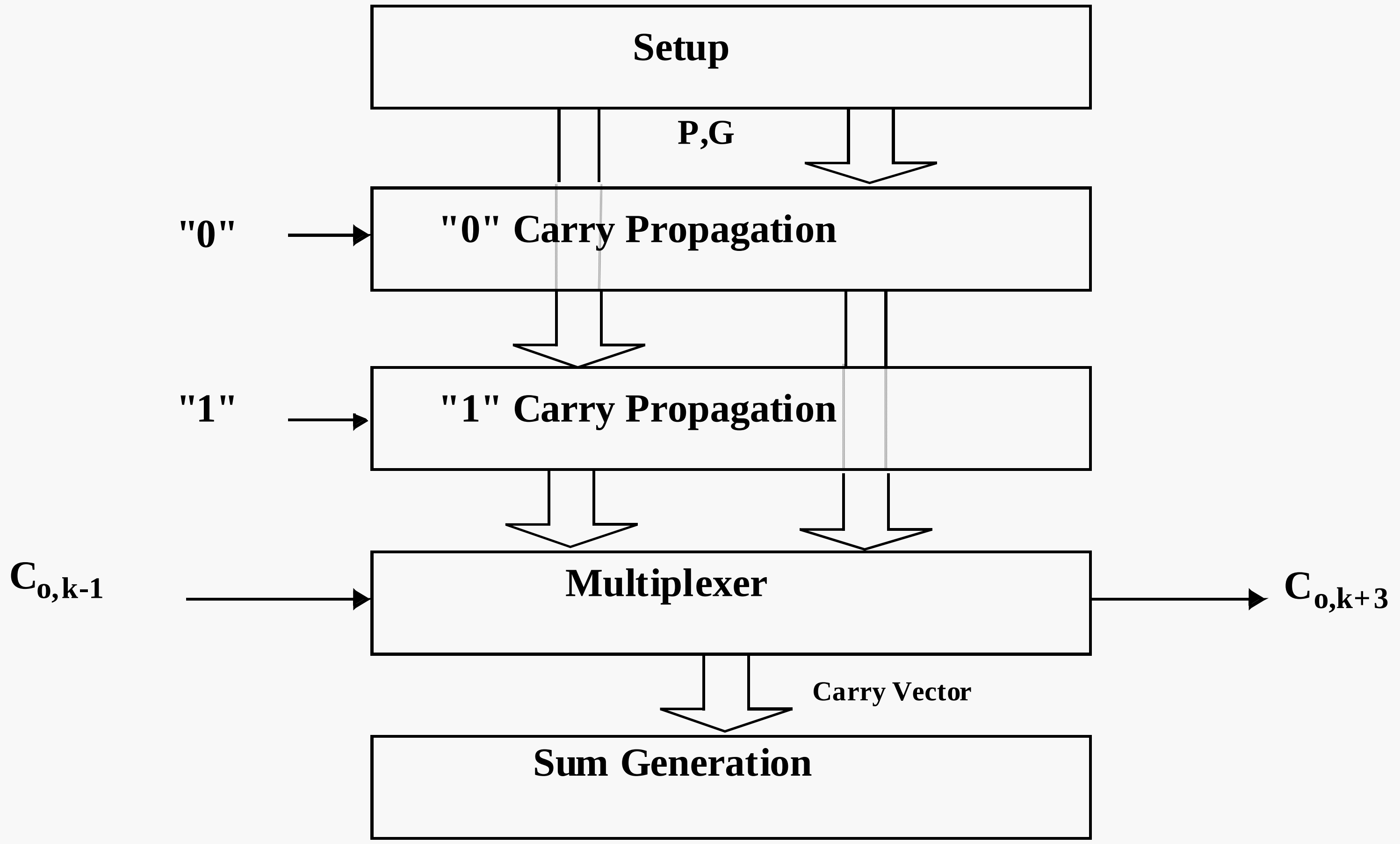
Carry-Bypass Adder (cont.)



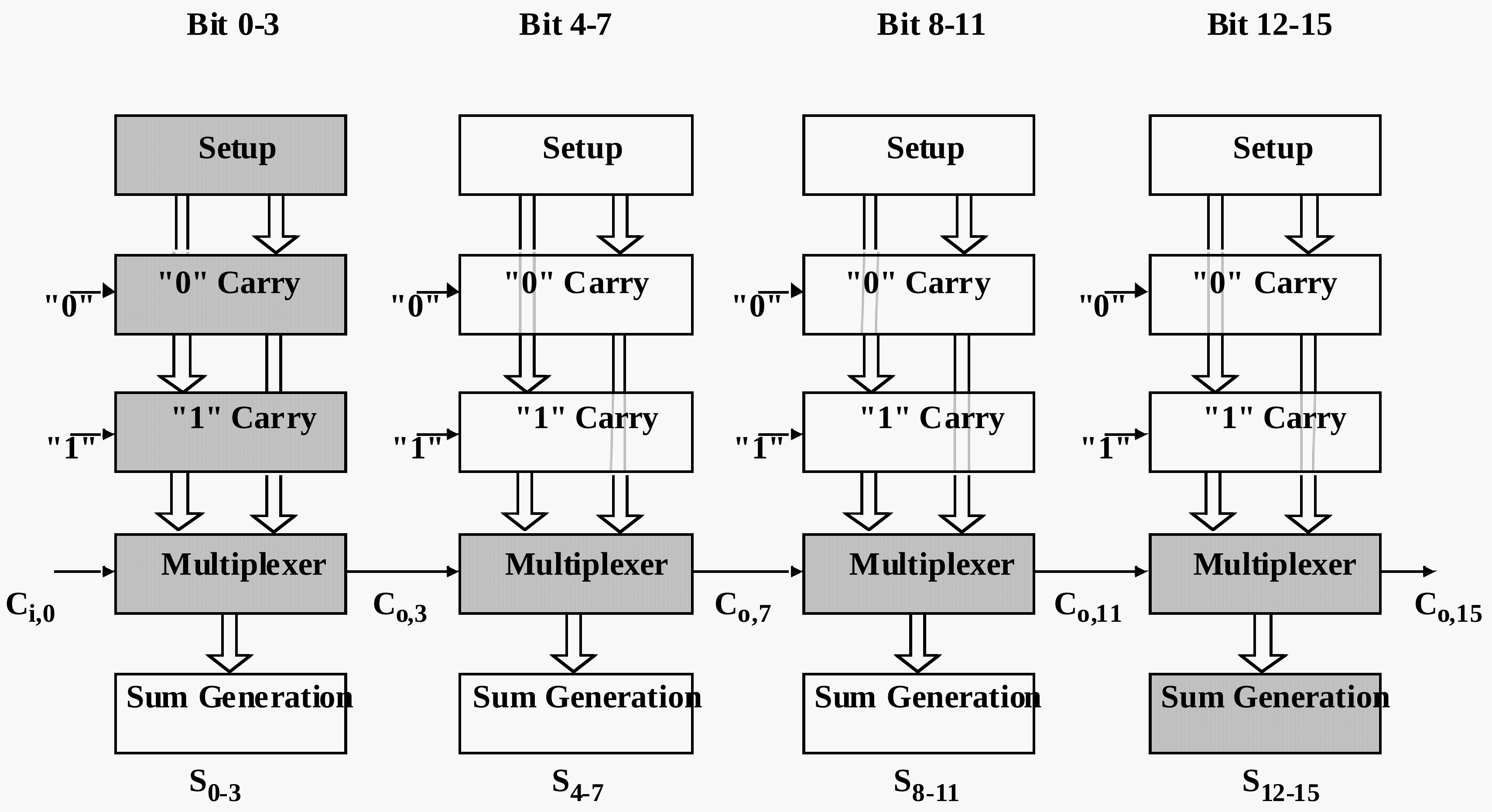
Carry Ripple versus Carry Bypass



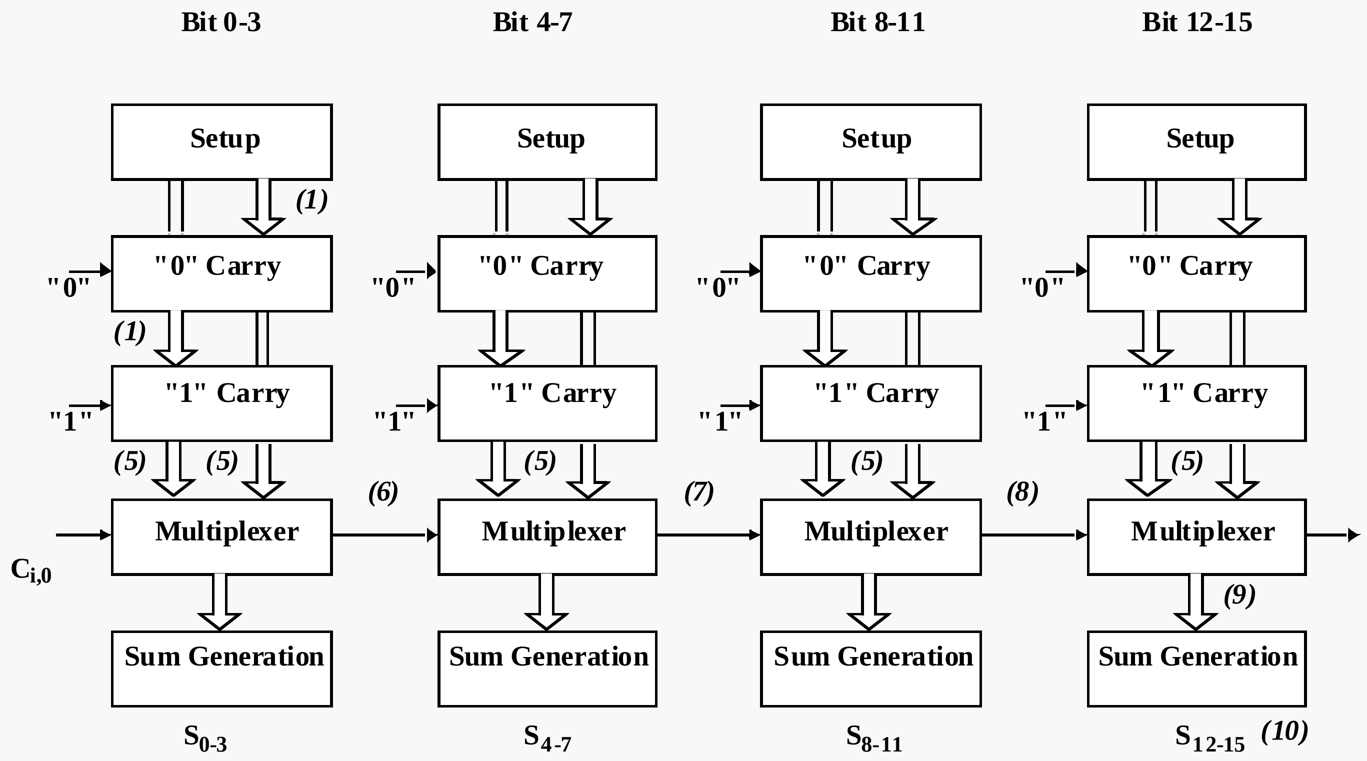
Carry-Select Adder



Carry Select Adder: Critical Path

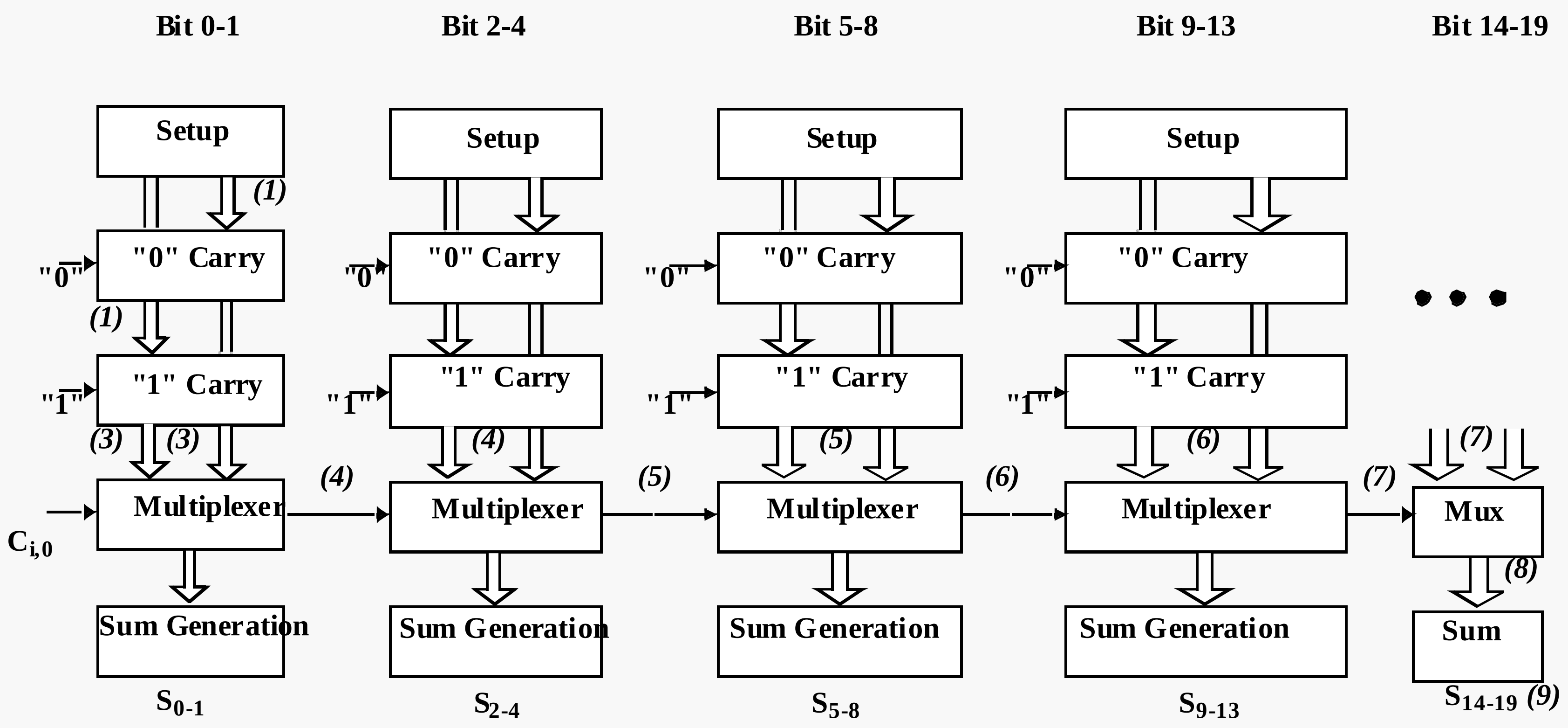


Linear Carry Select



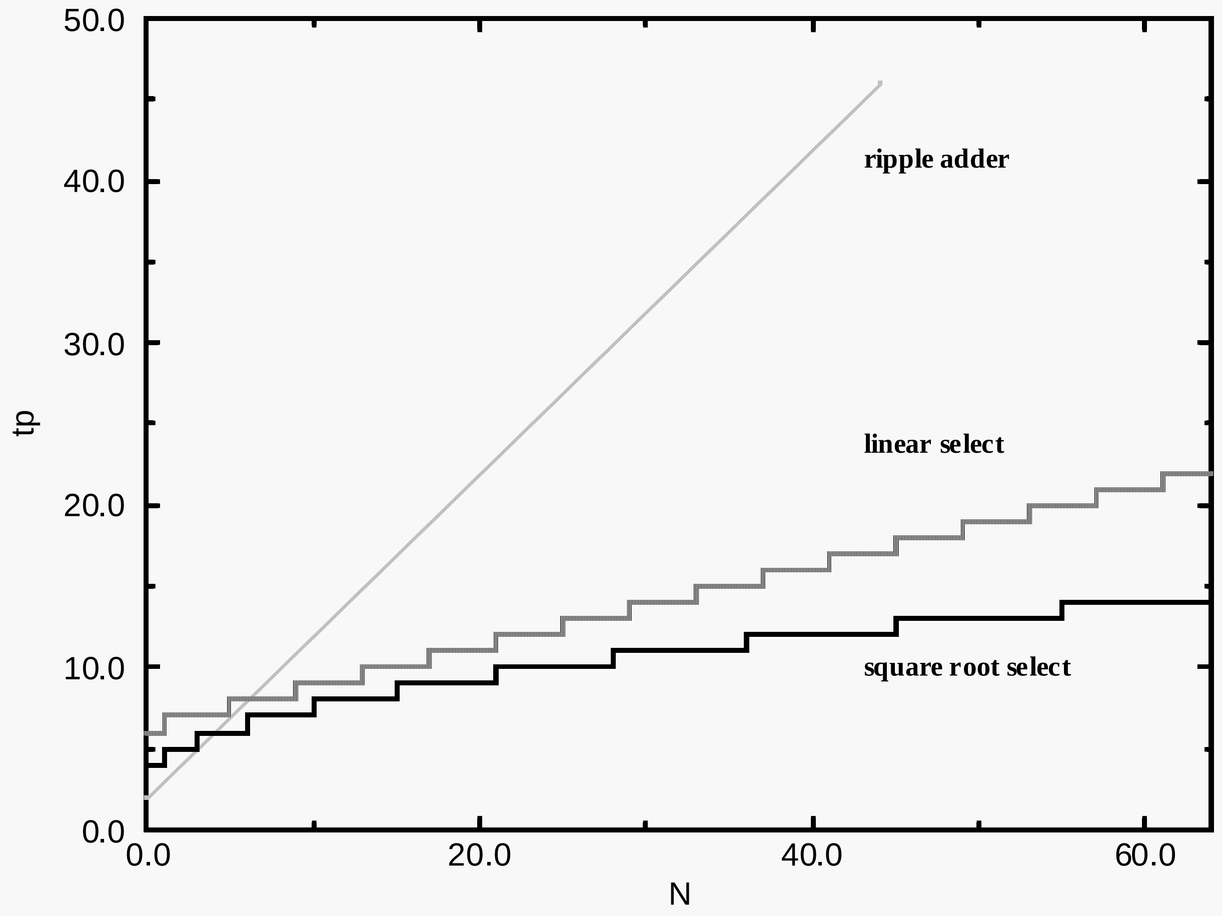
$$t_{add} = t_{setup} + \left(\frac{N}{M}\right) t_{carry} + M t_{mux} + t_{sum}$$

Square Root Carry Select

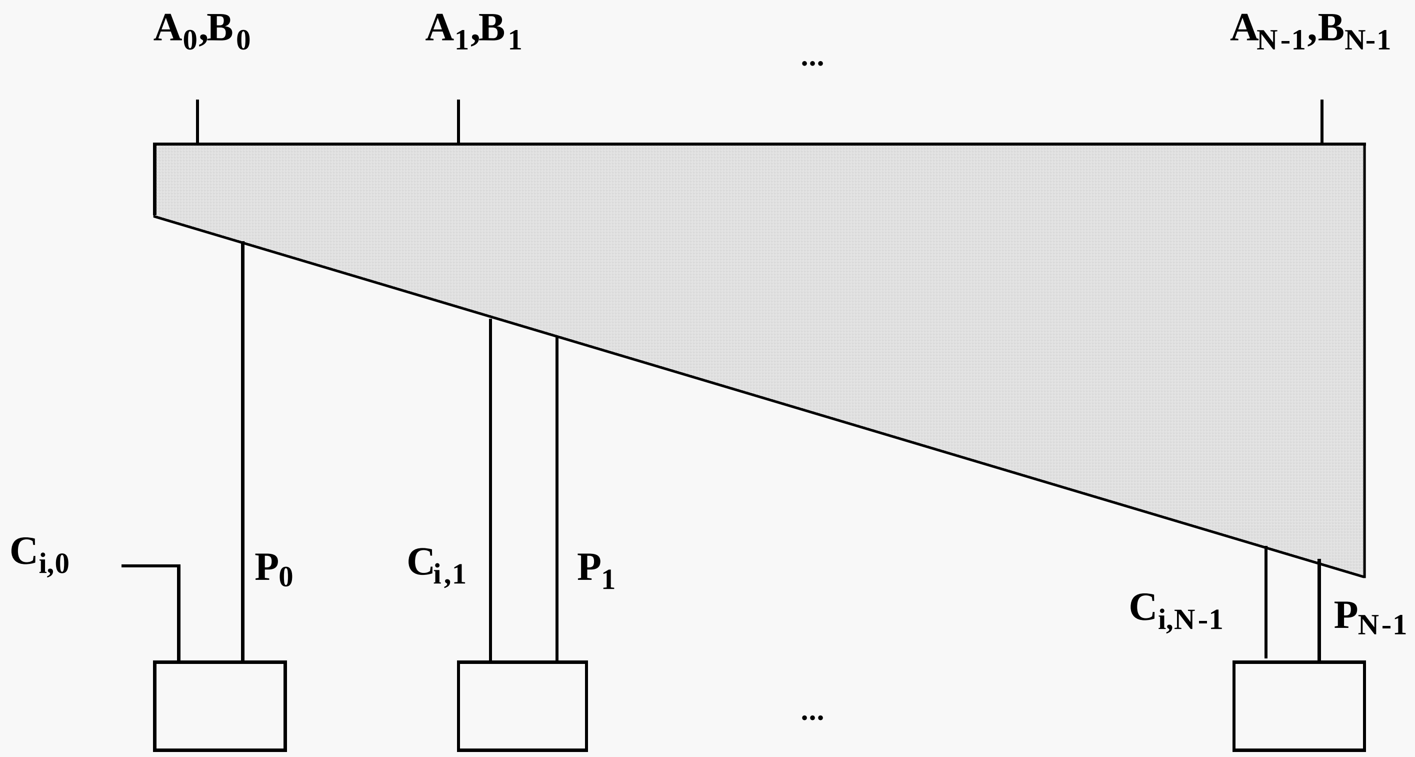


$$t_{add} = t_{setup} + P \cdot t_{carry} + (\sqrt{2N})t_{mux} + t_{sum}$$

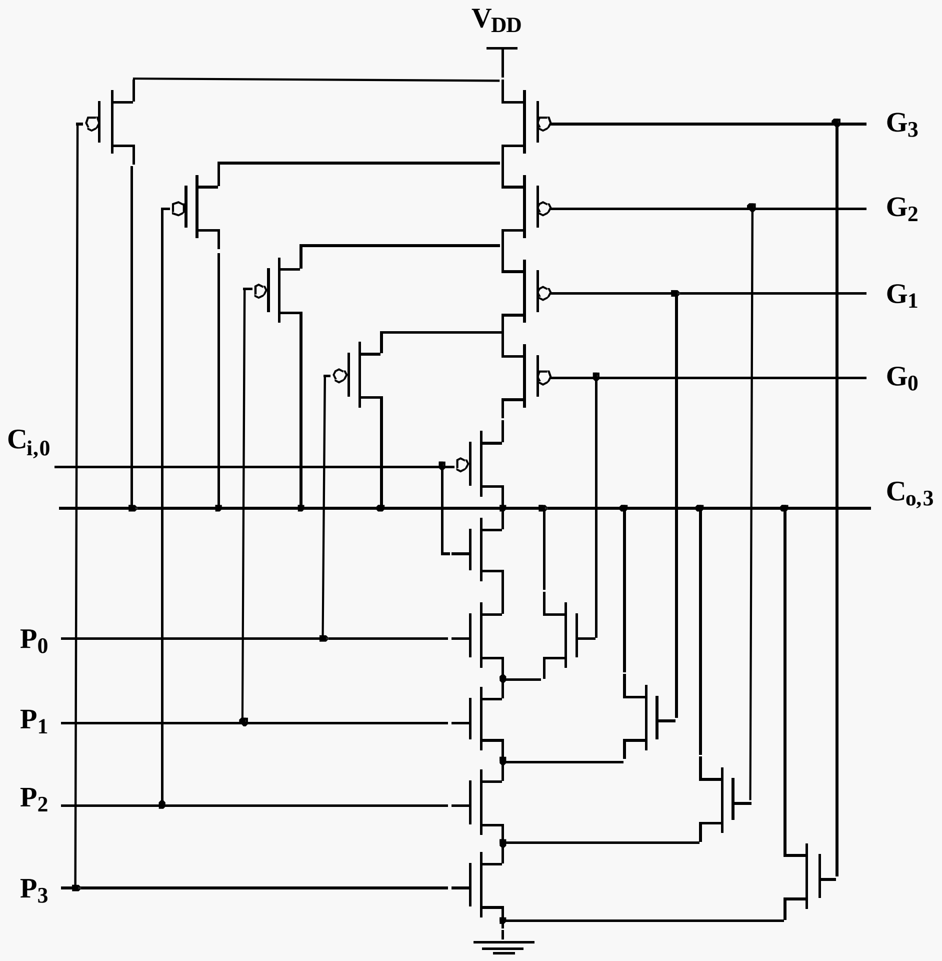
Adder Delays - Comparison



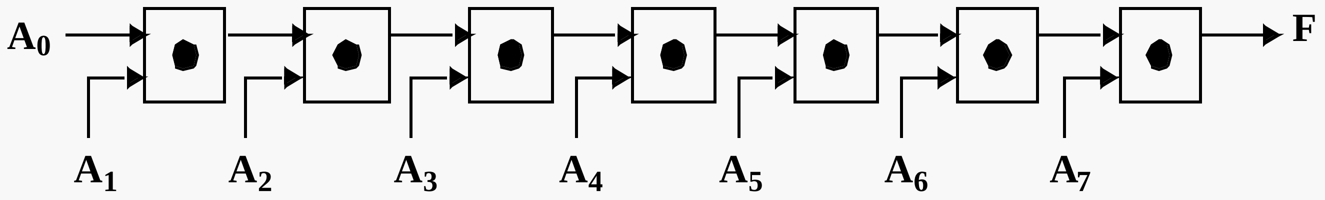
Look Ahead - Basic Idea



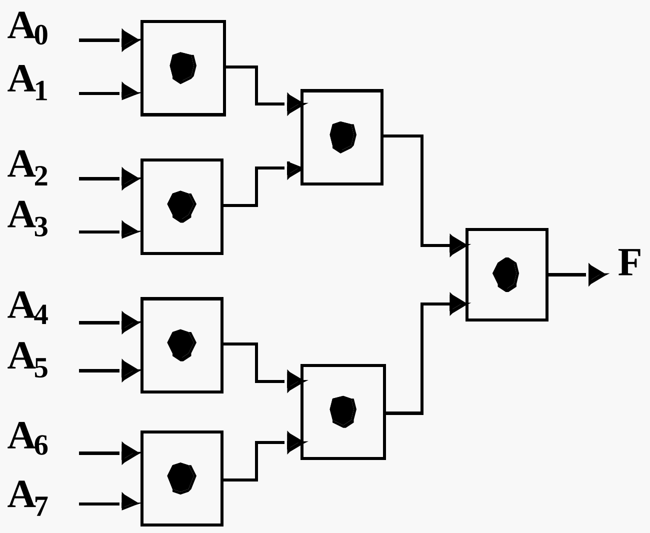
Look-Ahead: Topology



Logarithmic Look-Ahead Adder

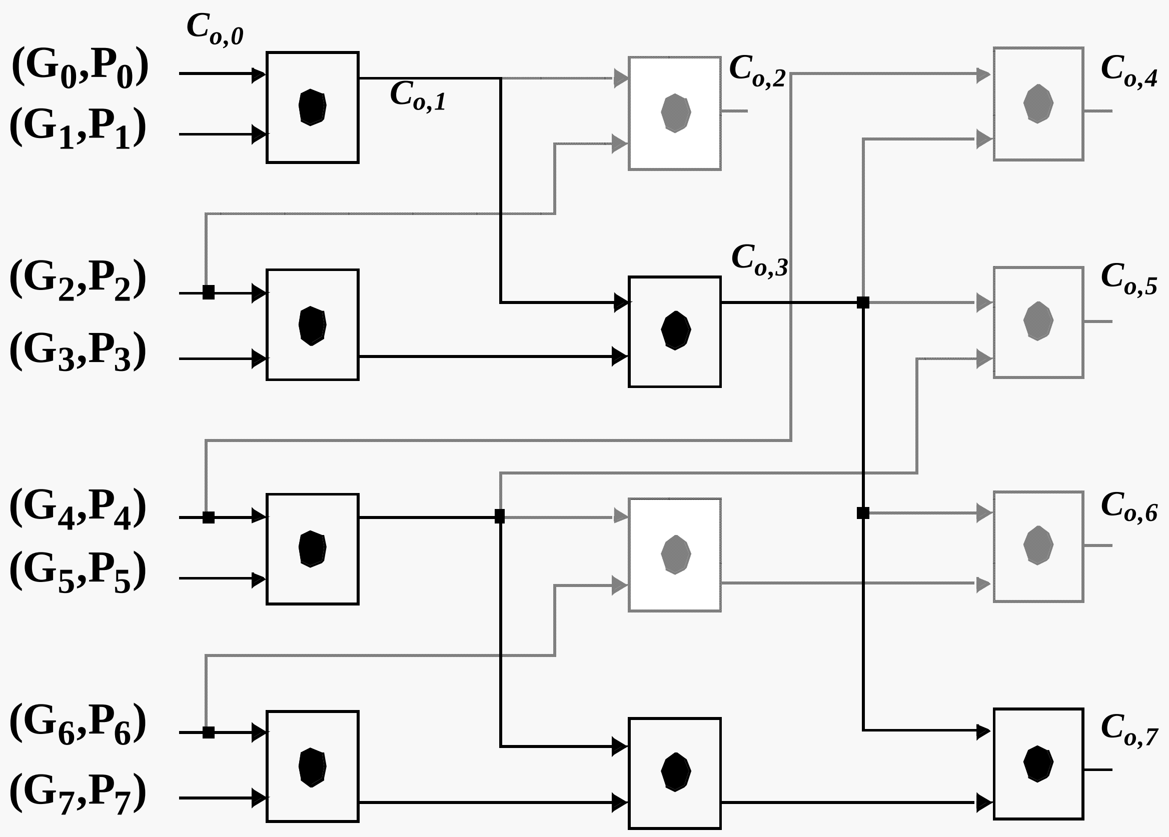


$t_p \sim N$



$t_p \sim \log_2(N)$

Brent-Kung Adder



$$t_{add} \sim \log_2(N)$$

The Binary Multiplication

$$Z = \ddot{X} \times Y = \sum_{k=0}^{M+N-1} Z_k 2^k$$

$$= \left(\sum_{i=0}^{M-1} X_i 2^i\right)\left(\sum_{j=0}^{N-1} Y_j 2^j\right)$$

$$= \sum_{i=0}^{M-1}\left(\sum_{j=0}^{N-1} X_i Y_j 2^{i+j}\right)$$

with

$$X = \sum_{i=0}^{M-1} X_i 2^i$$

$$Y = \sum_{j=0}^{N-1} Y_j 2^j$$

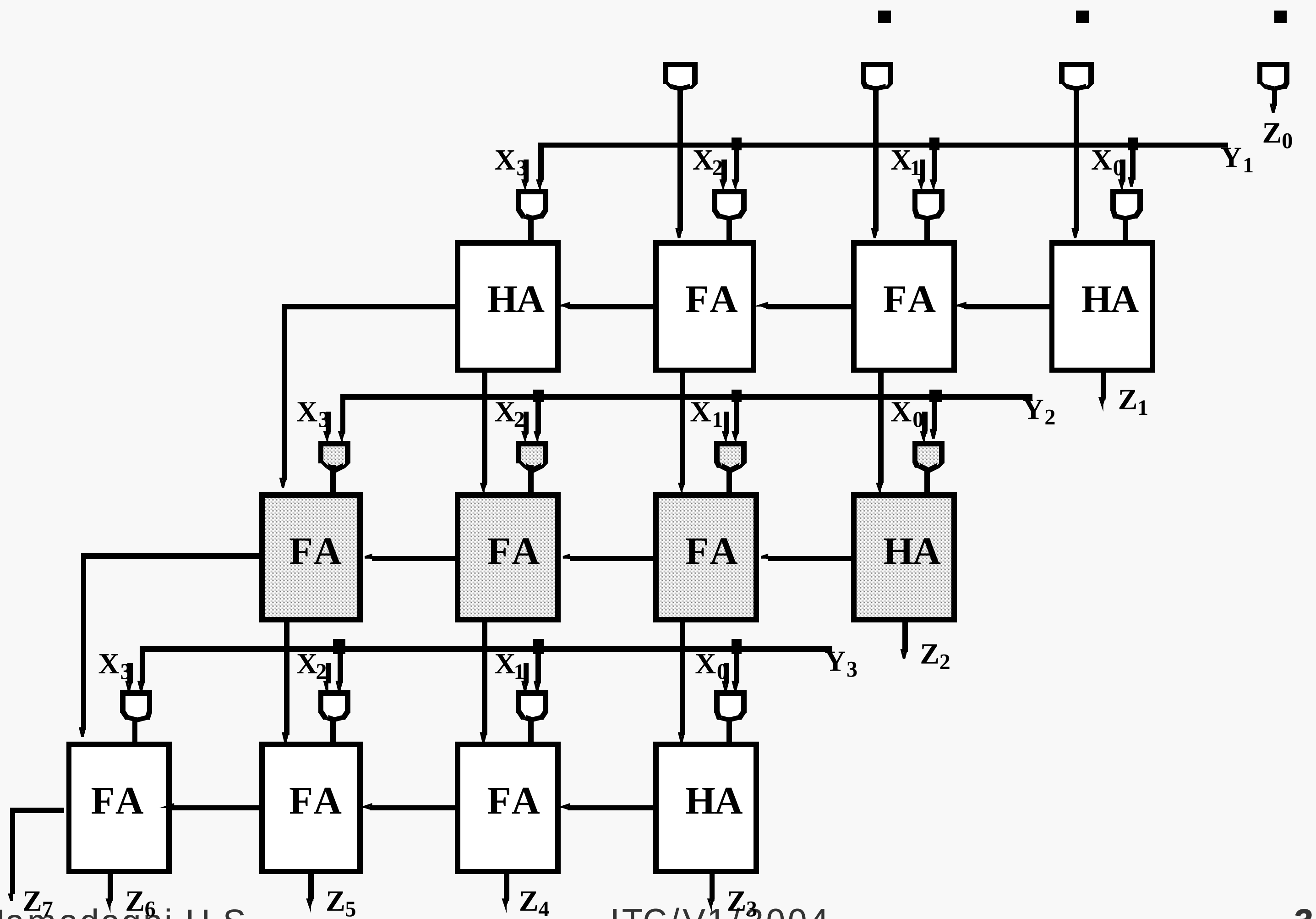
The Binary Multiplication

```
            1 0 1 0 1 0
×               1 0 1 1
------------------------
            1 0 1 0 1 0
          1 0 1 0 1 0
        0 0 0 0 0 0
+     1 0 1 0 1 0
------------------------
      1 1 1 0 0 1 1 1 0
```

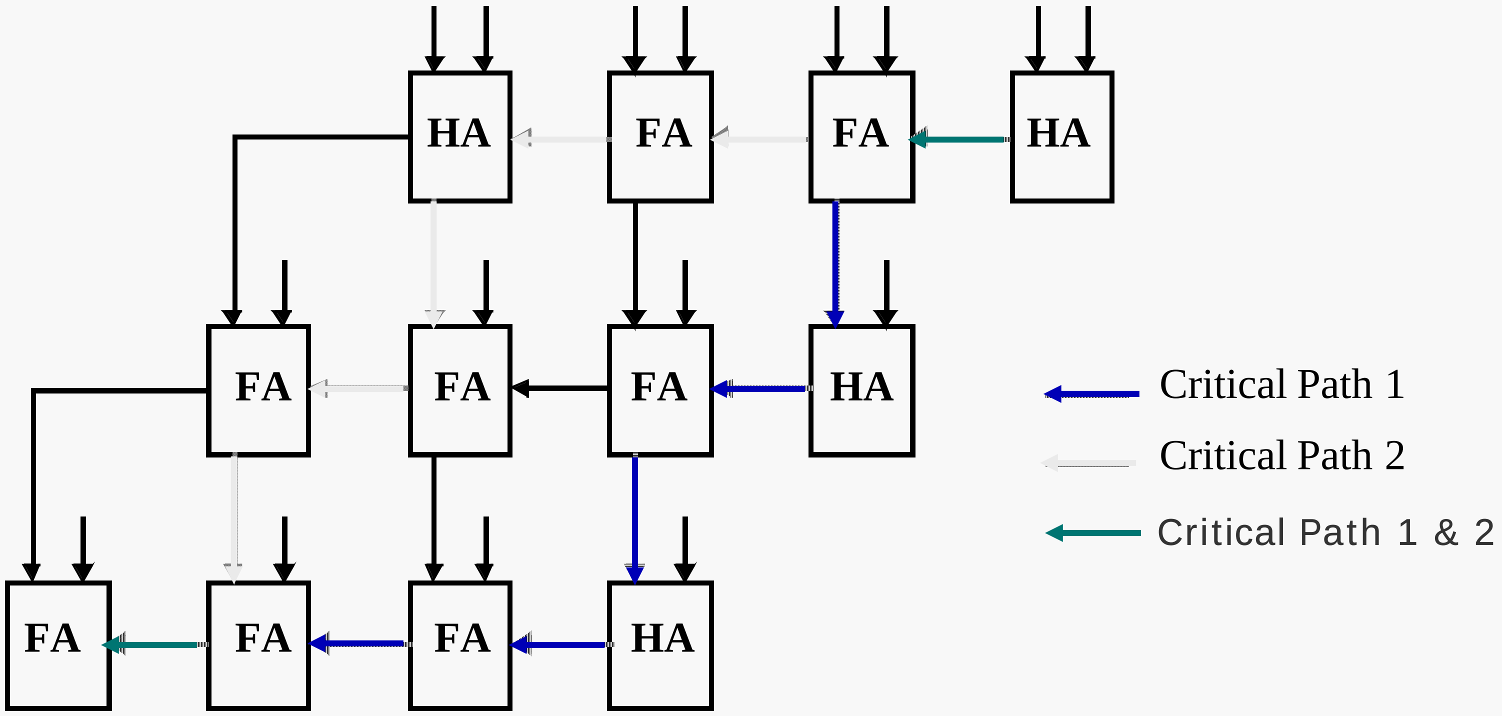
AND operation

Partial Products

The Array Multiplier

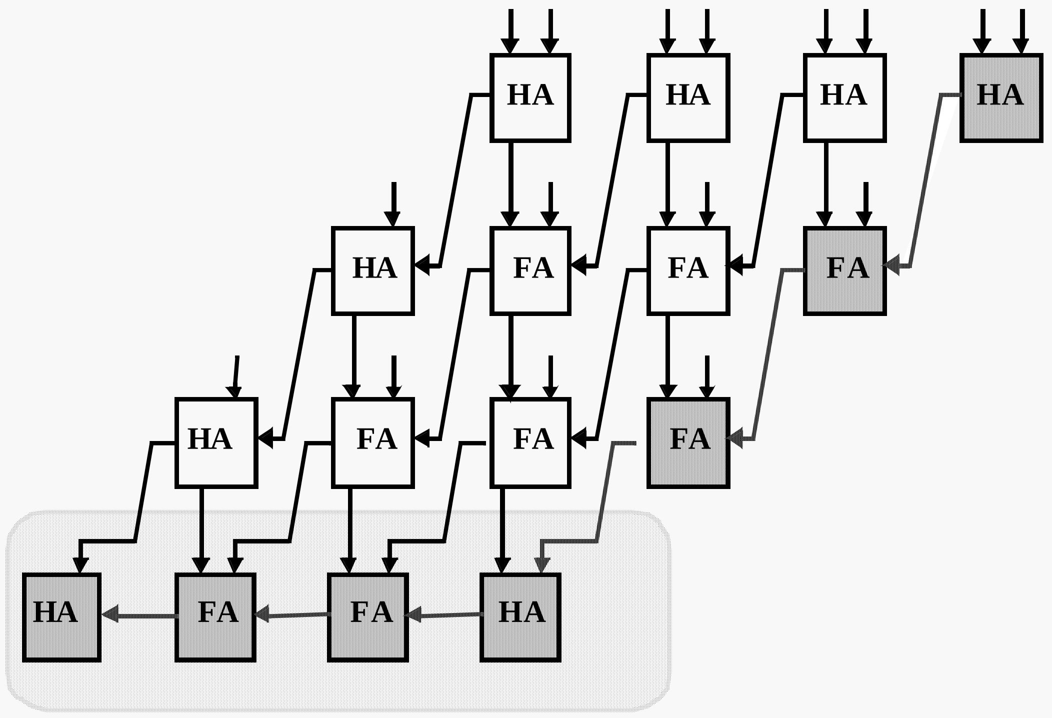


The MxN Array Multiplier — Critical Path



$$t_{mult} \approx [(M-1)+(N-2)]t_{carry}+(N-1)t_{sum}+(N-1)t_{and}$$

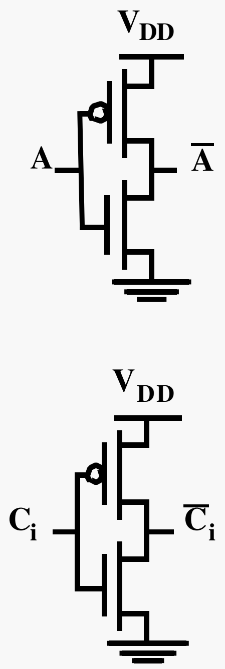
Carry-Save Multiplier

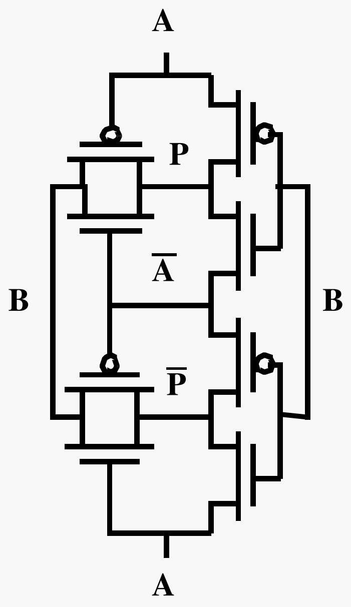


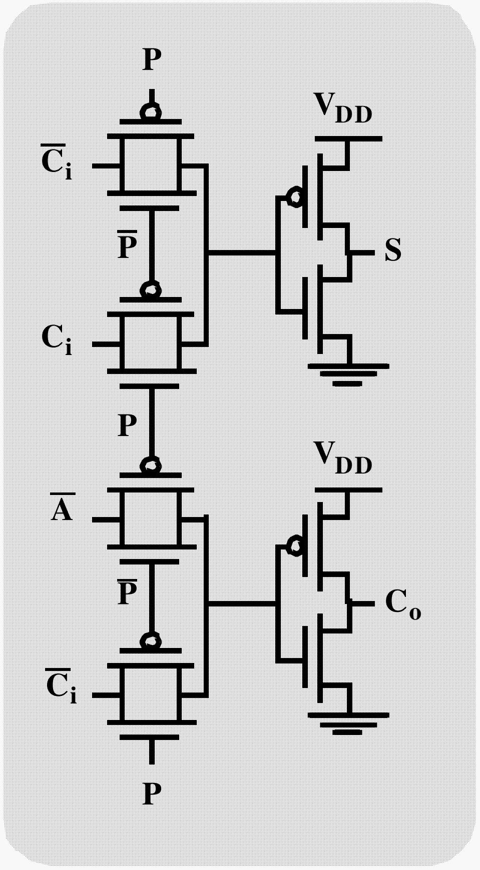
Vector Merging Adder

$$t_{mult} = (N-1)t_{carry} + (N-1)t_{and} + t_{merge}]$$

Adder Cells in Array Multiplier

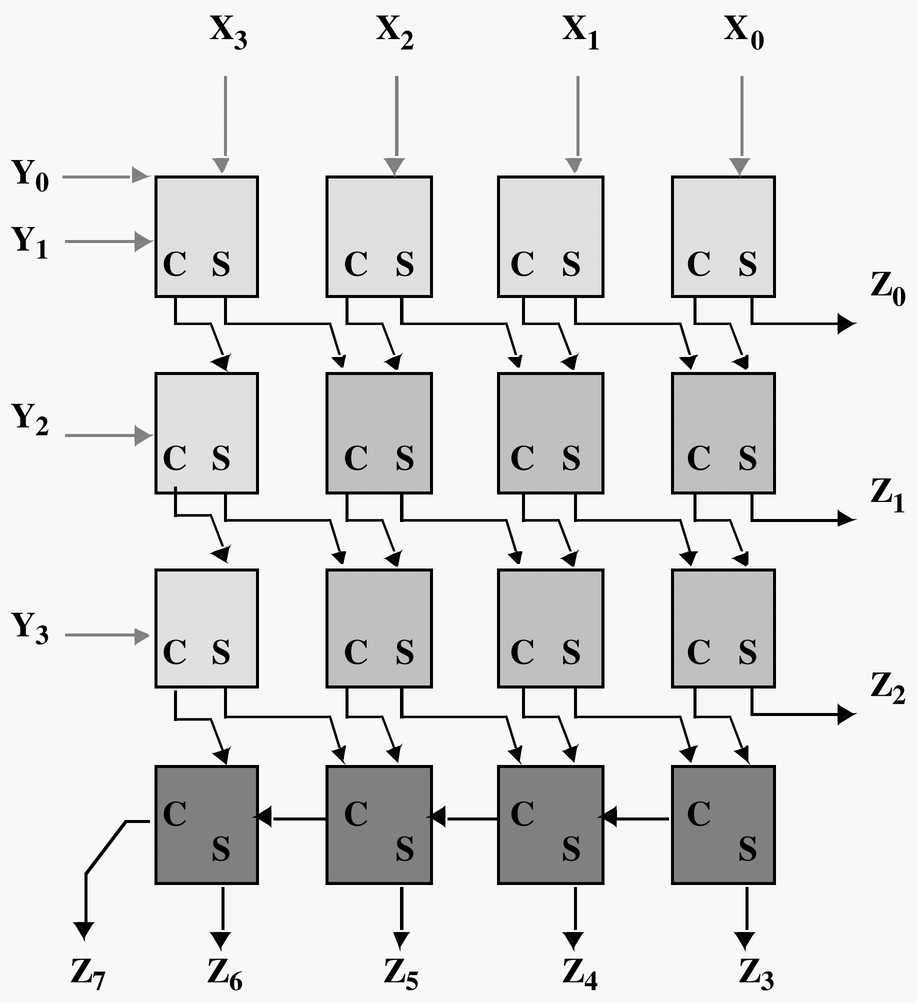






Identical Delays for Carry and Sum

Multiplier Floorplan



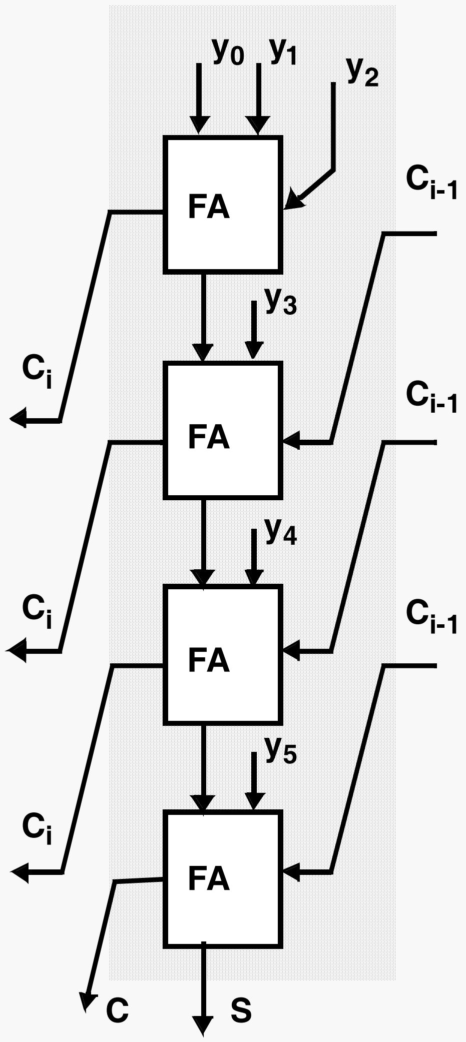
▯ HA Multiplier Cell

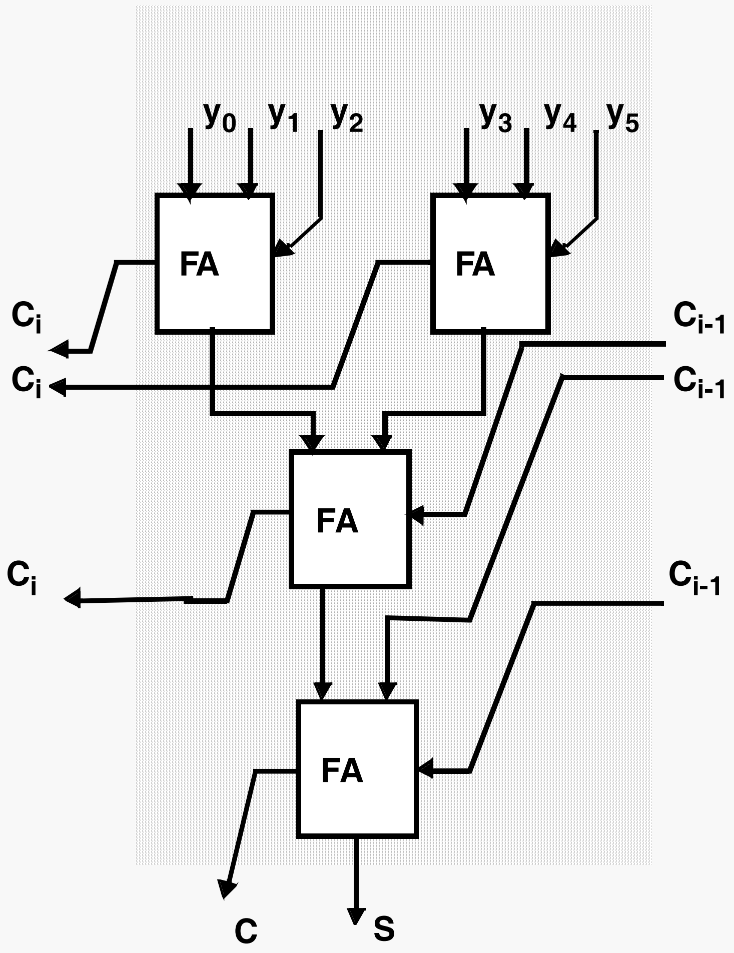
▨ FA Multiplier Cell

■ Vector Merging Cell

X and Y signals are broadcasted through the complete array. ($\longrightarrow$)

Wallace-Tree Multiplier



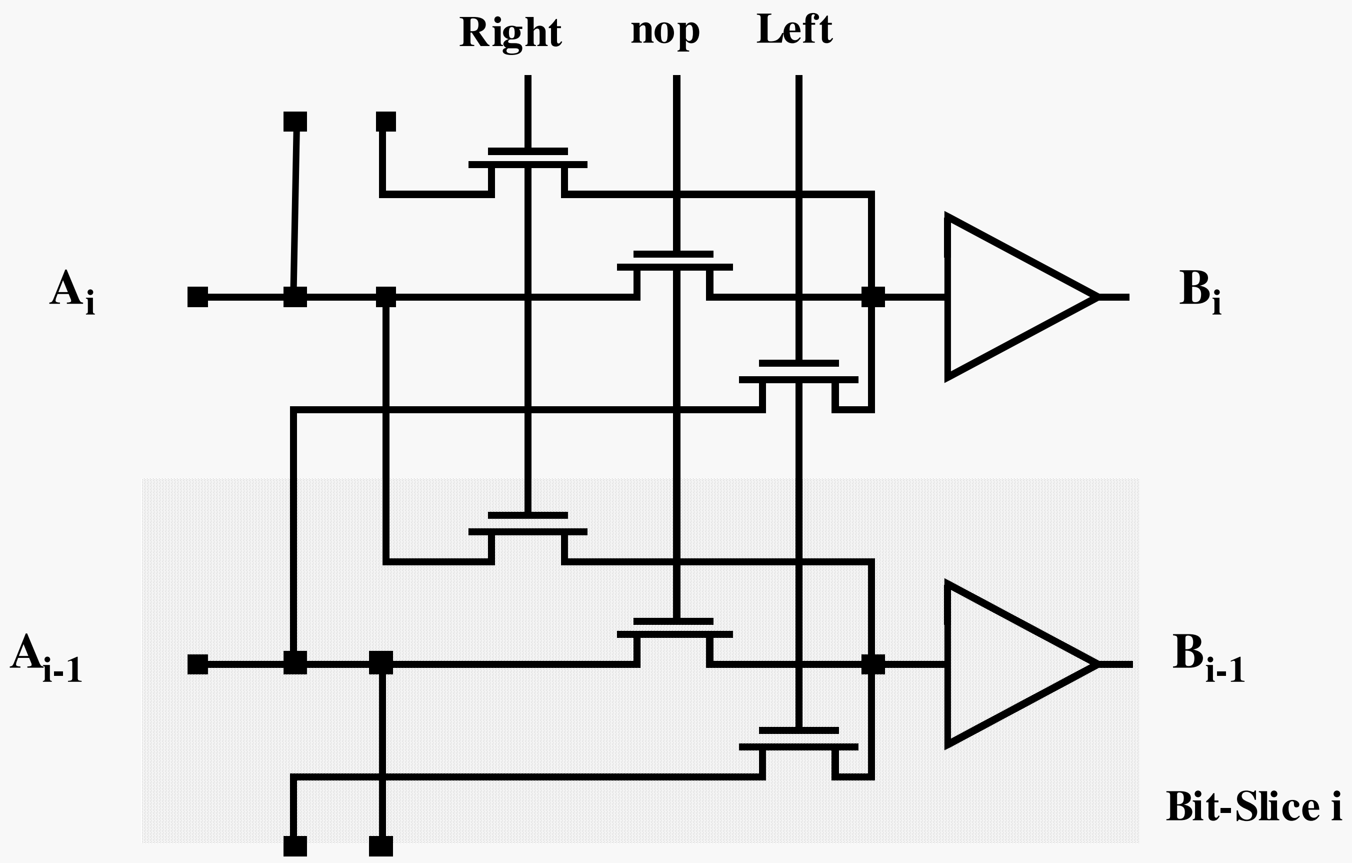


Multipliers —Summary

- Optimization Goals Different Vs Binary Adder
- Once Again: Identify Critical Path
- Other possible techniques
  - Logarithmic versus Linear (Wallace Tree Mult)
  - Data encoding (Booth)
  - Pipelining

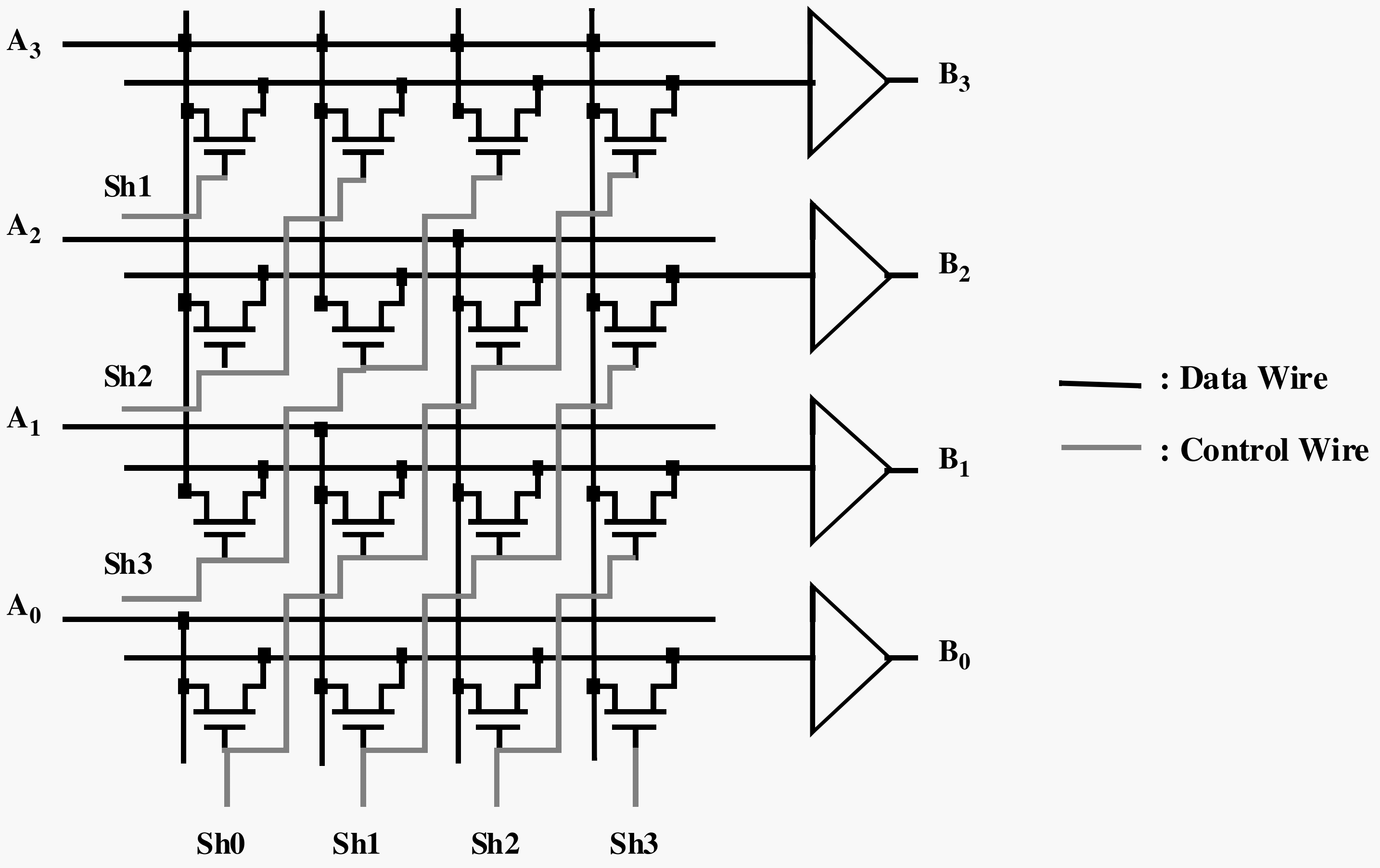
FIRST GLIMPSE AT SYSTEM LEVEL OPTIMIZATION

The Binary Shifter



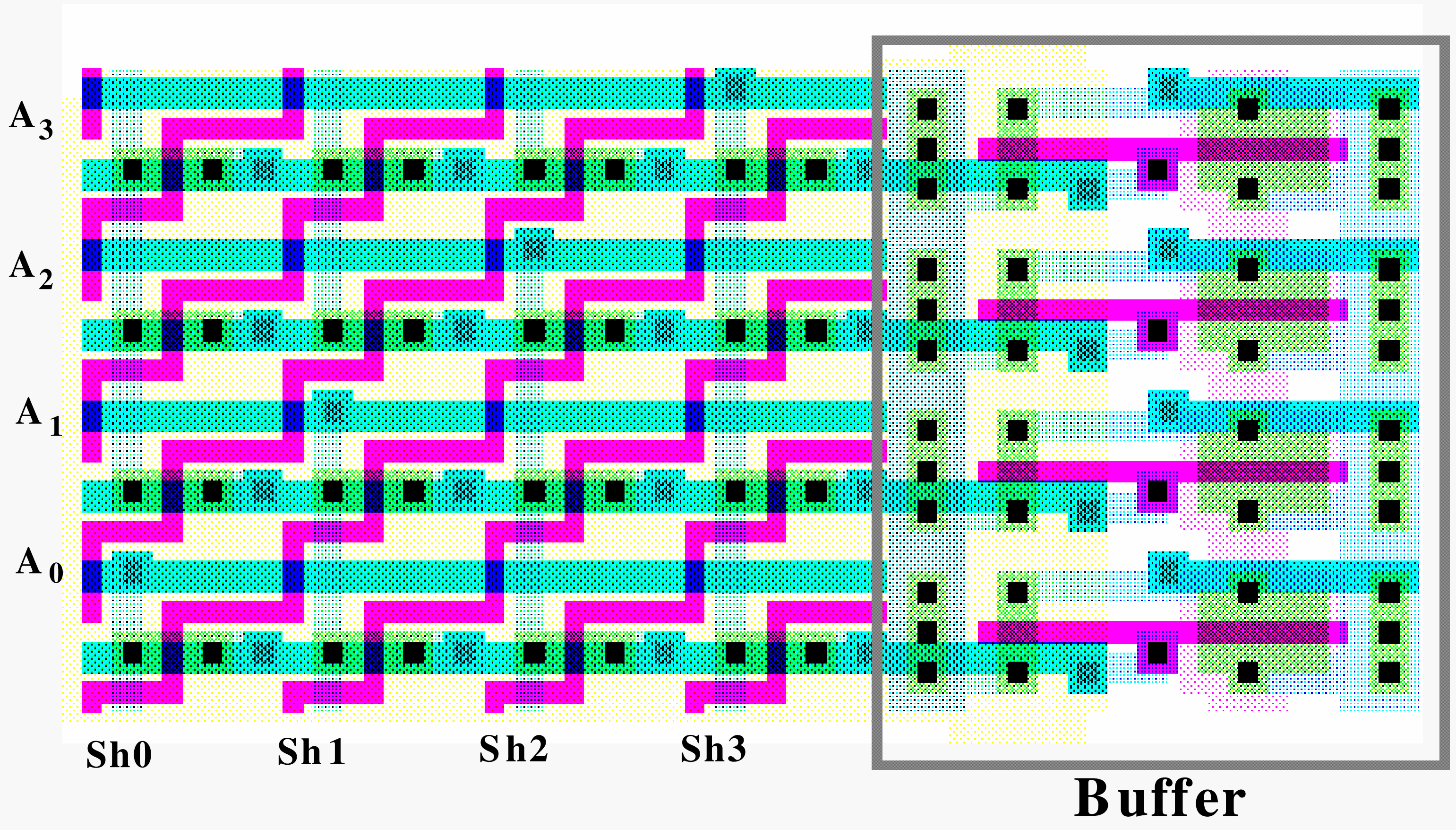
...

The Barrel Shifter



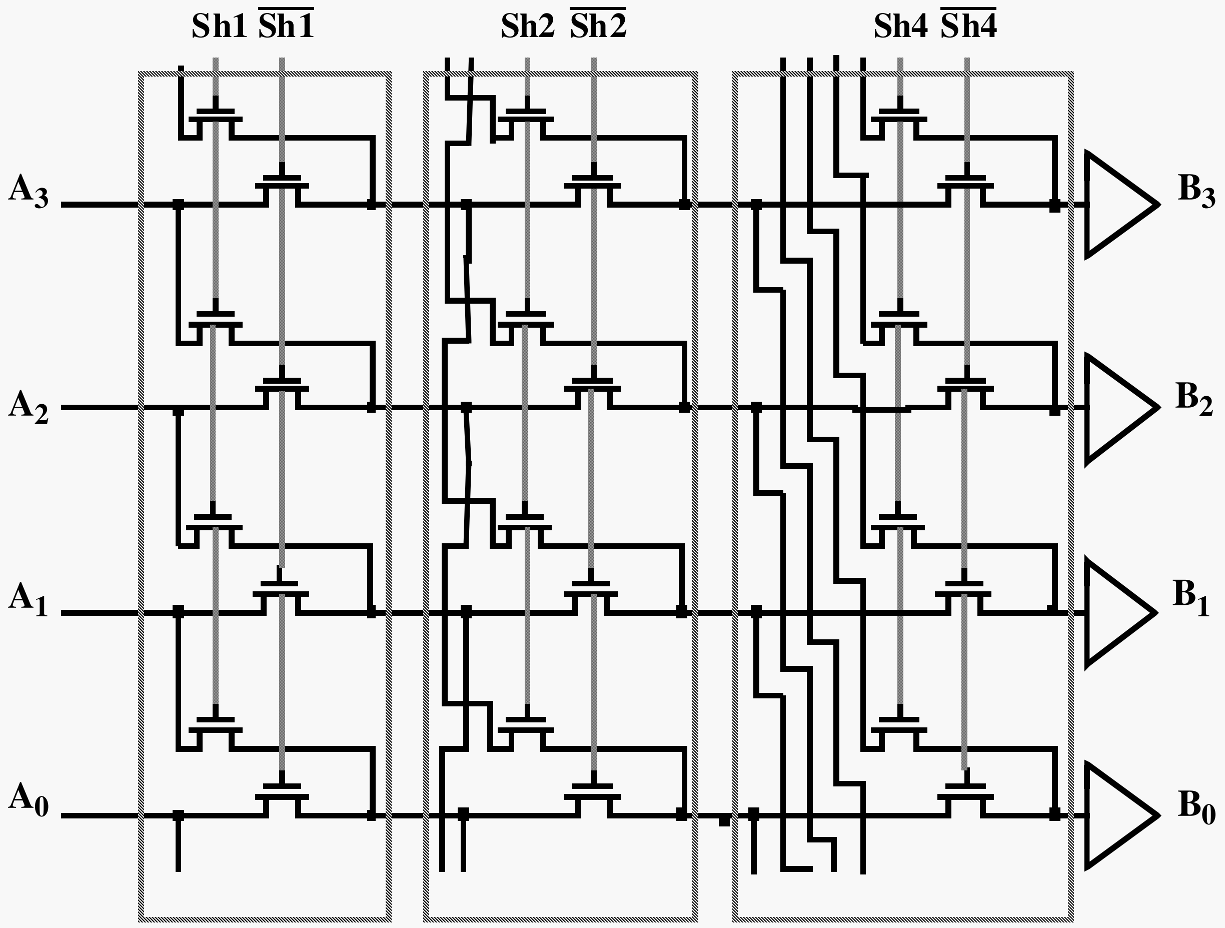
Area Dominated by Wiring

4x4 barrel shifter

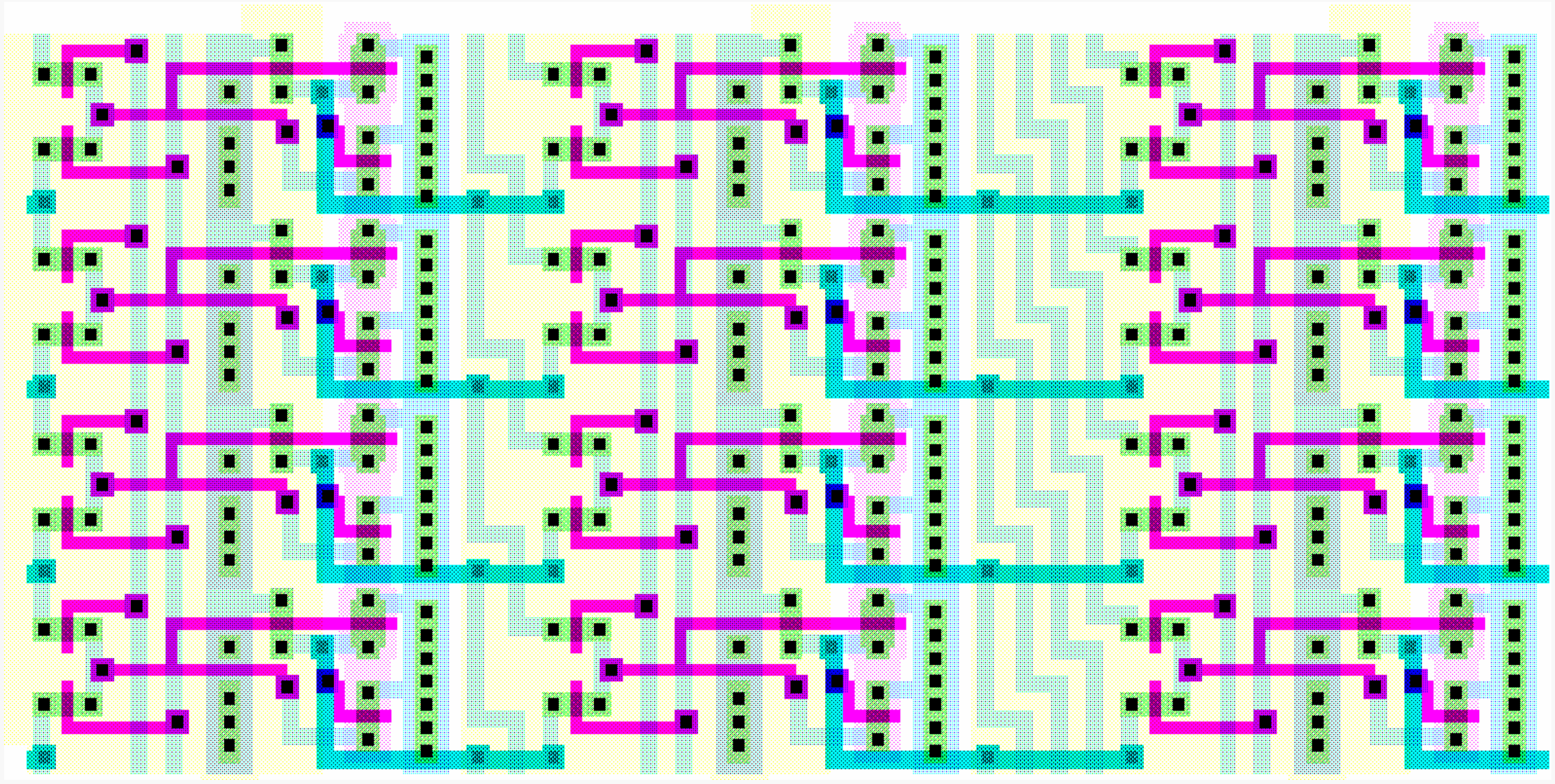


$$\text{Width}_{\text{barrel}} \sim 2\, p_m\, M$$

Logarithmic Shifter

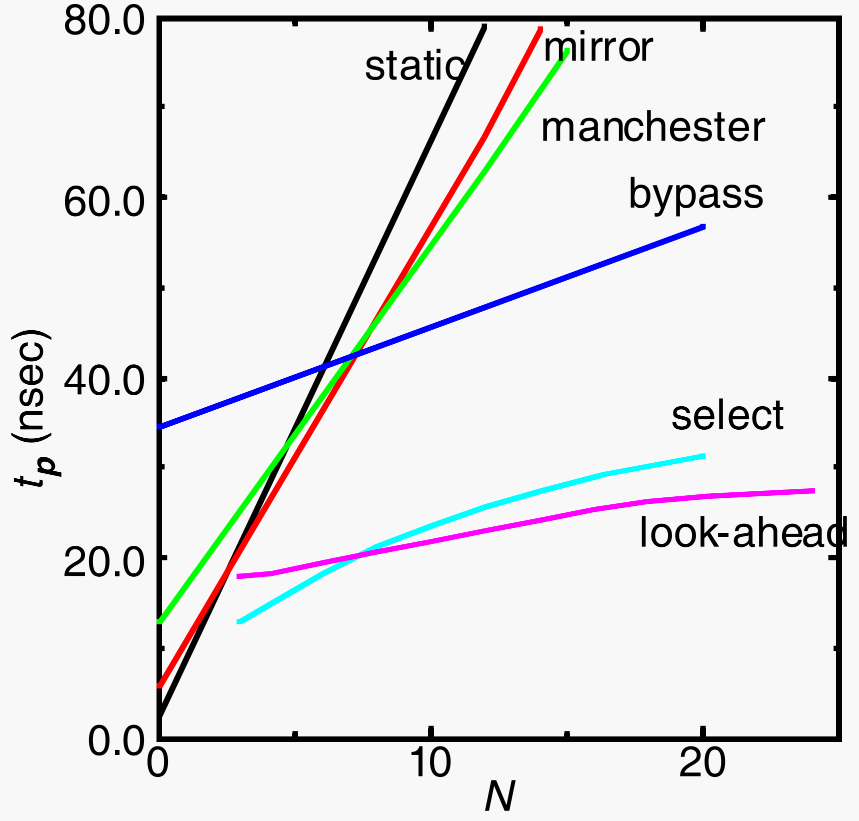

Sh1 $\overline{\text{Sh1}}$
Sh2 $\overline{\text{Sh2}}$
Sh4 $\overline{\text{Sh4}}$
A_3
A_2
A_1
A_0
B_3
B_2
B_1
B_0

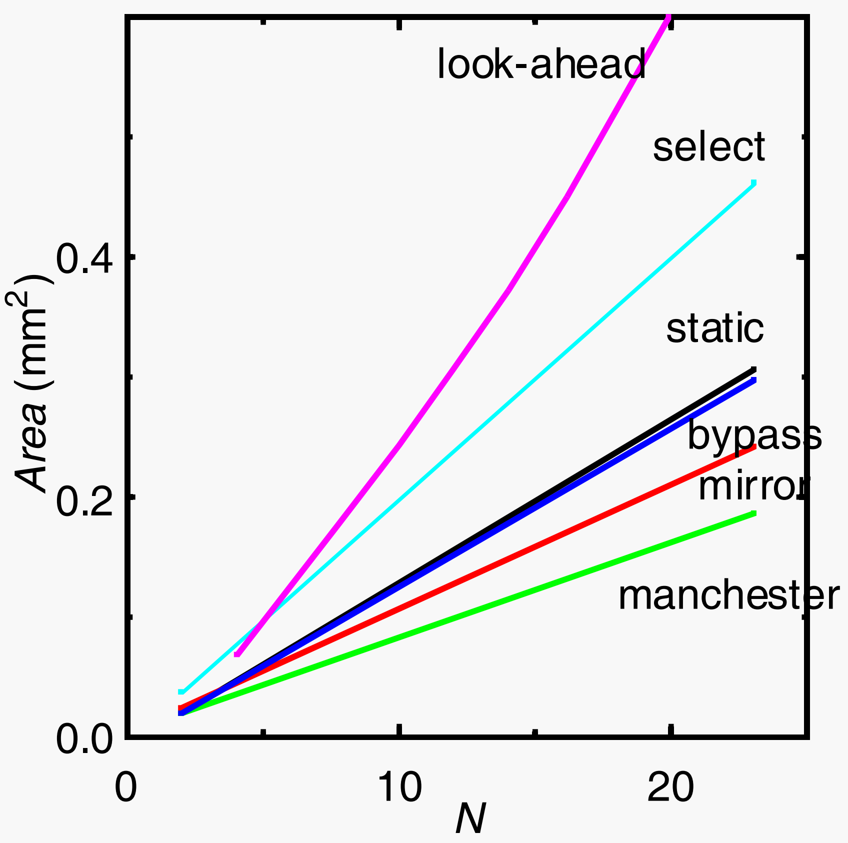
0-7 bit Logarithmic Shifter



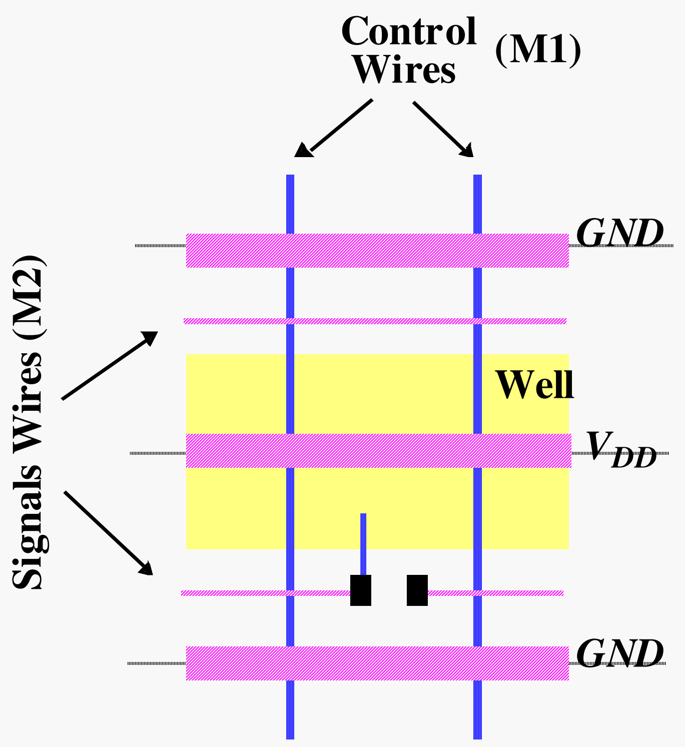
$$width_{\log} \approx p_m\left(2K + \left(1 + 2 + \ldots + 2^{K-1}\right)\right) = p_m\left(2^K + 2K - 1\right)$$

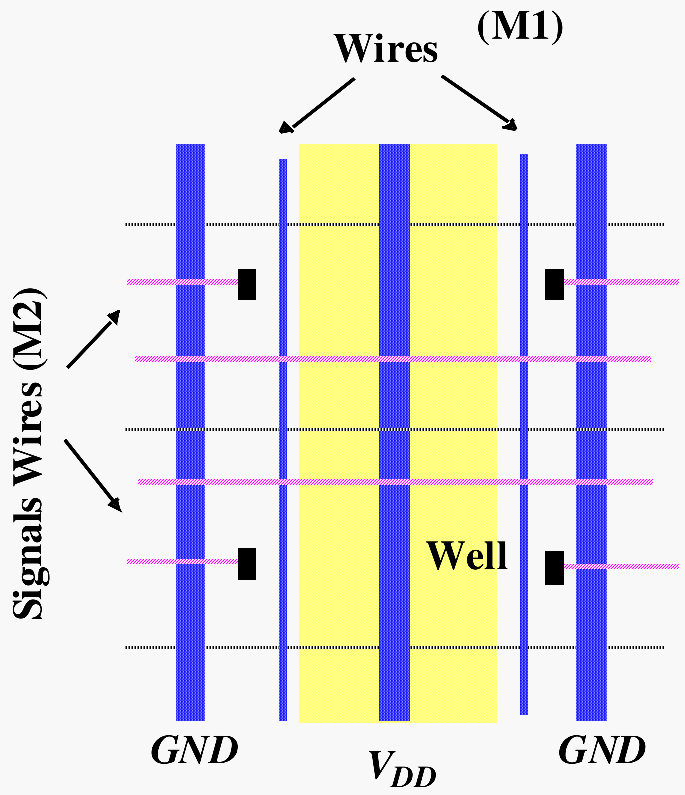
Design as a Trade-Off





Layout Strategies for Bit-Sliced Datapaths





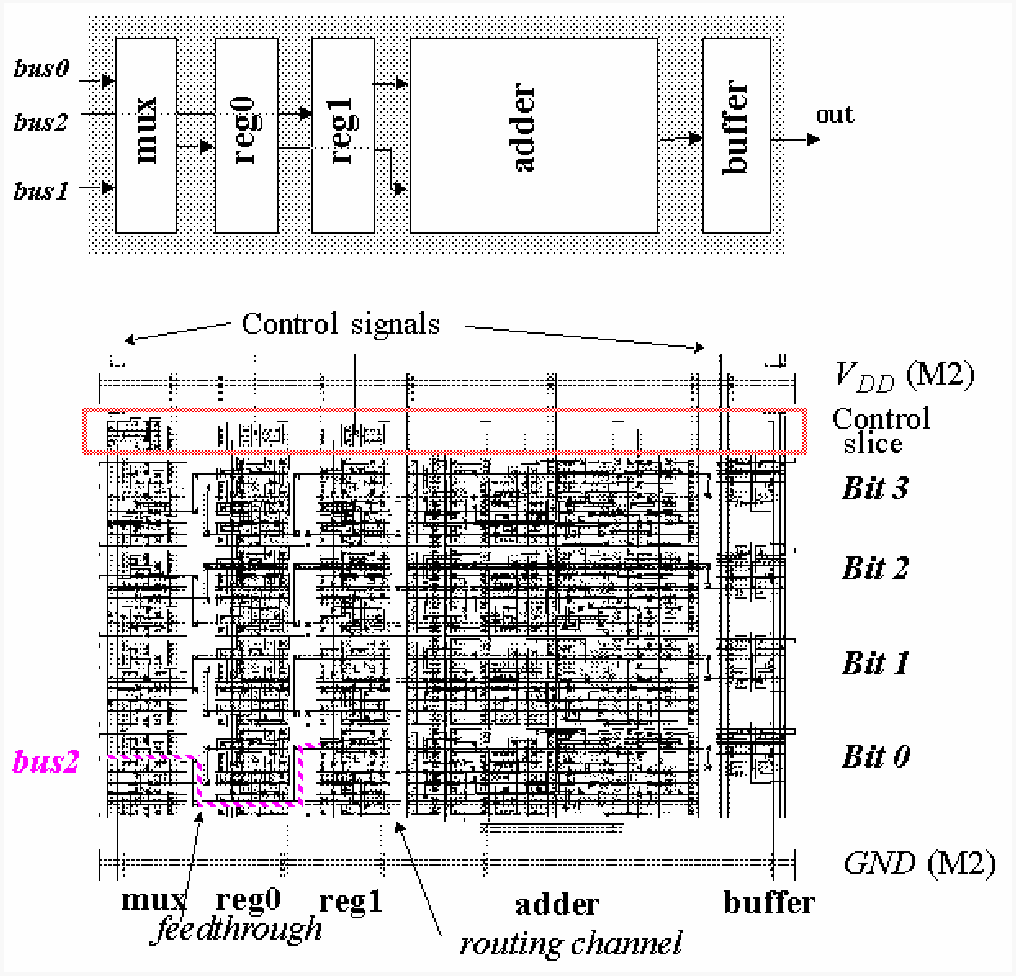
Approach I —

Signal and power lines parallel

Approach II —

Signal and power lines perpendicular

Layout of Bit-sliced Datapaths

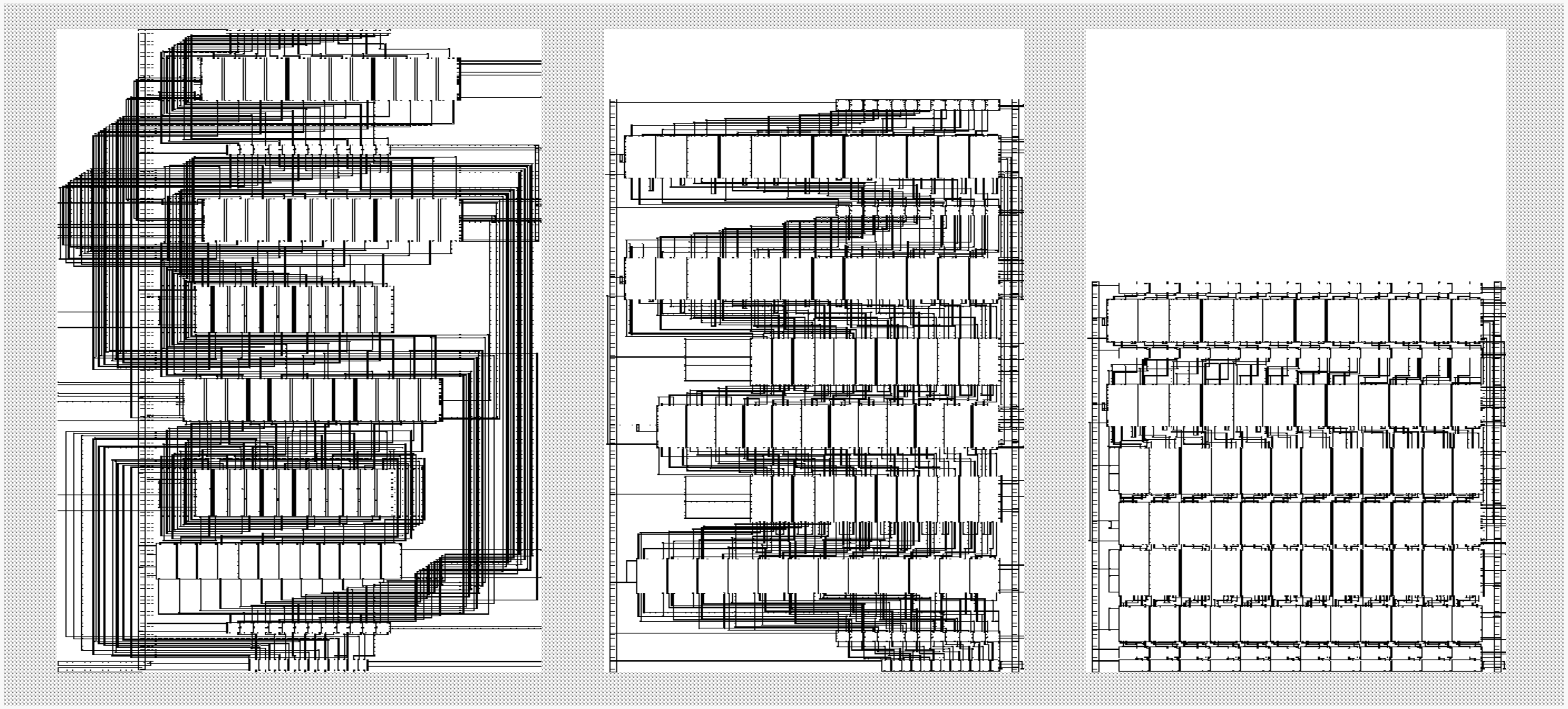


Layout of Bit-sliced Datapaths

(a) Datapath without feedthroughs and without pitch matching (area = 4.2 mm^2).

(b) Adding feedthroughs (area = 3.2 mm^2)

(c) Equalizing the cell height reduces the area to 2.2 mm^2.



1. INTRODUCTION TO LOW POWER VLSI

Power consumption, speed of operation, and silicon area occupied are the most important attributes of integrated circuit implementation of electronic circuits. Most integrated circuit designs focus on these issues and try to optimize one or more of these attributes. Of these, low power consumption is becoming the most important as markets for portable, handheld equipment is growing. Also, new generations of semiconductor processing technologies need to reduce power dissipation of digital circuits due to large device density and higher operating speed.

Analysis of power consumption and its optimization has been traditionally carried out for analog circuit design. It is now becoming important in the digital design influencing all aspects of the design.

Power is the rate at which energy is delivered or exchanged. In a VLSI, electrical energy is converted to thermal energy during the operation of the circuit. Power dissipated by a circuit is the rate at which energy is taken from the source minus the rate at which energy delivered by the circuit to a load. The energy converted to heat is removed from the chip for maintaining a reasonable temperature of the semiconductor substrate.

Low power circuit design for VLSI includes analysis of power dissipation in the circuit and its optimization. In analysis we try to estimate the power or energy dissipation at different phases of the design process. Its purpose is to increase confidence of the design with the assurance that the power consumption specifications are not violated. Accuracy of analysis depends on the availability of design information. In early phases of design, we attempt to get power dissipation estimates using only the system model of the circuit to be designed as we don’t have much information on the detailed design. Therefore, in this phase results obtained are not accurate. However, it helps us to take an appropriate approach to circuit design for low power. As the design proceeds to gate, and circuit level, a more accurate analysis is carried out. These analyses produce more accurate estimates but take longer computation time.

Analysis techniques also form the basis for design optimization. Optimization is the process of generating the best design, given an optimization goal, without violating design specifications. An automatic design optimization algorithm requires fast analysis to evaluate the merits of the design choices. Manual optimization also demands a reliable analysis tool to provide accurate estimation of power dissipation. A decision to apply a particular low power technique often involves trade-offs from conflicting requirements. In such contexts, main criteria to be considered are the impact on the circuit delay, and the chip area. Other factors of chip design such as design cycle time, testability, quality, reliability, reusability, and risk may be affected by a particular design decision to achieve the low power.

1.1. Need for Low Power VLSI

In the early stages of VLSI design, the device density and operating frequency were low enough and the power dissipation of the chip was within reasonable limits. With steady changes in technology, VLSI now has a larger device density, and a higher operating frequency. Therefore, power dissipation per unit area of Silicon increasing. Therefore, low power VLSI design has become a necessity.

Also, an increased market demand for the portable electronic goods supplied by batteries is an important driver for low power design. Unfortunately, batteries have not experienced a rapid density growth compared to VLSI circuits. Stored energy per unit weight of batteries roughly doubles in about a decade. Therefore, the battery technology alone will not solve the low power problem in the near future.

The power dissipation of high performance microprocessors could be several tens of Watts and CPUs in some personal computers require cooling fans directly mounted on the chip due to this power dissipation. A chip that operates at 3V, consuming 10W of power draws an average current of 3A. The transient current could be as high as 10A. This creates problem in the design of power supply rails and poses big challenges in the assuring adequate noise immunity.

With a push towards generation and use of “clean energy”, it has also become necessary to save power usage to extent possible. In fact power saving is an alternative strategy to generate more power. Since electricity generation is a major source of pollution, inefficient energy usage in computing equipment indirectly contributes to environmental pollution. With promotion of energy compliance for electronic equipment in various countries, low power design assumes a central position in saving power in these equipment.

1.2. Mechanisms of power dissipation in VLSI

There are two types of power dissipation in CMOS circuits: *dynamic* and *static.*

- Dynamic power dissipation is mainly due to charging of load capacitances during switching activities of the circuits. A higher operating frequency leads to more frequent switching activities in the circuits and results in increased power dissipation. A second cause of dynamic dissipation is due to short circuit current that flows between the supply and ground rails of the CMOS circuit for a brief period during switching when the CMOS elements between these rails are both on.
- In CMOS logic, leakage current is the source of static power dissipation. This current flows from the supply rail to the ground rail as a DC due to the device which is expected to be off conducting a leakage current due to several effects present in a MOS device. Also, deviations from the CMOS style logic can cause additional static current to be drawn.

The most dominant source of dynamic power dissipation in CMOS circuits is the charging and discharging of load capacitances. Most digital CMOS circuits do not require capacitors for their intended operations. The capacitances are found in circuits due to parasitic effects of interconnection wires and transistors. Such parasitic capacitances cannot be avoided and it has a significant impact on the power dissipation of the circuits. The estimation and analysis of parasitic capacitance is crucial for estimating power as well as signal delays.

Figure 1.1 shows the equivalent circuit of charging and discharging output capacitance of a CMOS logic gate. We use a toggling switch to model the charging and discharging cycles.

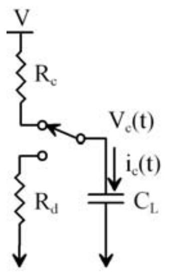


Fig 1.1: Capacitance charging and discharging in a CMOS circuit

In fig 1.1 *V* is an ideal constant voltage source and R_c and R_d are charging and discharging resistances of circuit respectively. The voltage $v_c(t)$ and the current $i_c(t)$ of a capacitance C_L at time *t* are given by

$$i_c(t) = C_L \frac{dv_c(t)}{dt} \tag{1.1}$$

During the charging cycle from time t_0 to t_1, the energy E_s drawn from the voltage source is

$$E_s = \int_{t_0}^{t_1} V \bullet i_c(t) \bullet dt \tag{1.2}$$

Initially the capacitor contains no charge and the voltage across its terminals is zero, that is, $v_c(t_0) = 0$. Assuming that the capacitor is fully charged at the end of the charging cycle, we have $v_c(t_1) = V$. Substituting Equation (1.1) into (1.2), we have

$$E_s = C_L \bullet V \int_{t_0}^{t_1} \left(\frac{dv_c(t)}{dt} \right) \cdot dt = C_L \bullet V \int_{t_0}^{t_1} \left(dv_c \right) = C_L \bullet V^2 \tag{1.3}$$

Part of the electrical energy E_s drawn from the voltage source is stored in the capacitor and the rest is dissipated in the resistor R_c. The energy E_c stored in the capacitor at the end of the charging cycle is

$$E_c = \int_{t_0}^{t_1} v_c(t) \cdot i_c(t) \cdot dt = C_L \int_{t_0}^{t_1} v_c(t) \cdot \frac{dv_c(t)}{dt} \cdot dt = C_L \cdot \int_{0}^{V} v_c(t) \cdot dv_c = \frac{1}{2} C_L V^2 \tag{1.4}$$

From Equations (1.3) and (1.4), the energy dissipated in R_c during charging is therefore

$$E_R = E_s - E_c = \frac{1}{2} C_L V^2 \tag{1.5}$$

Now consider the discharging cycle from t_1 to t_2, we assume that the capacitor is fully discharged, that is, $v_c(t_1) = V$ and $v_c(t_2) = 0$. The energy E_d dissipated in the discharge resistor R_d is

$$E_d = -\int_{t_1}^{t_2} v_c(t) \cdot i_c(t) \cdot dt = -C_L \int_{V}^{0} v_c(t) \cdot dv_c = \frac{1}{2} C_L V^2 \tag{1.6}$$

E_d is equal to the energy stored in the capacitance at the beginning of the discharging cycle. If we charge and discharge the capacitance at the frequency of *f* cycles per seconds, the power dissipation of the system is

$$P = E_f \cdot f = C_L V^2 f \tag{1.7}$$

Equation (1.7) is the most important equation in digital VLSI chip design. The equation is easy to compute, and is applicable to almost all digital CMOS circuit because only a few assumptions have been used in its derivation.

We observe that during charging, $C_L V^2$ energy is drawn from the energy source, half of which is dissipated in the charging resistance R_c and the other half is stored in the capacitor. During discharge, the energy stored in the capacitor is dissipated in the discharging resistor R_d. Assumptions in the above calculations are:

- C_L is constant
- V is constant
- C_L is fully charged and discharged respectively at t_0 and t_2.

The result is independent of the charging and discharging circuit resistances (R_c, R_d), length of charging, discharging cycle (t_0, t_1, t_2), and the voltage or current waveforms $v_c(t)$, $i_c(t)$. Moreover, R_c, R_d can be nonlinear, and time varying. The equation 1.7 is valid as long as the above three assumptions are satisfied. Most CMOS digital circuits designed today satisfy the above assumptions during logic operations. Therefore, for most CMOS circuits operating at medium to high frequency, this is the dominant mode of power dissipation. Equation (1.7) is only the power dissipation caused by a single capacitor C_L. In general, the total power should be summed over each capacitance C_i in a circuit giving the following equation.

$$P = \sum_i C_i V_i^2 f_i \qquad (1.8)$$

Here V_i is the voltage swing across the capacitor C_i switching at frequency f_i. For CMOS circuits, V is typically the same for all capacitance C_i. If we further assume that f_i is constant, (for example, by taking the average of all f_i's) we get

$$P = V^2 f \sum_i C_i = C_{total} V^2 f \qquad (1.9)$$

Here C_{total} is the sum of all capacitance, f is the average frequency and V is the voltage swing.

In today's typical CMOS process with minimum feature size of 90 to 180 nm, typical values of C_i are in the order of 1fF to 1pF, charging and discharging frequency several hundred MHz, with V of 1 to 3V.

1.2.1. Example

A 16 bit bus operating at 3.3 V and 100 MHz clock rate is driving a capacitance of 20 pF/bit. Each bit has a toggling probability of 0.25 at each clock cycle. What is the power dissipation in operating the bus?

C = 16 x 20 = 320 pF

V = 3.3 V

F = 0.25 x 100 = 25 MHz

Power dissipation = 320 pF x 3.3^2 V^2 x 25 MHz = 87.12 mW

1.3. Short circuit Current in CMOS Circuit

1.3.1. Short circuit Current of the Inverter

Figure 1.2 shows a CMOS inverter with transistor threshold voltages of V_{tn} and V_{tp} as shown on the transfer curve. When the input signal level is above V_{tn}, the N-transistor is on. Similarly, when the signal level is below V_{tp}, the P-transistor is on. When the input signal v_i switches, there is a short duration during which the input level is between V_{tn} and V_{tp} and both transistors are turned on. This causes a short circuit current from V_{dd} to ground that dissipates power.

Figure 1.3 shows the approximate current waveforms during the transition. The current is zero when the input signal is below V_{tn} or above V_{tp}. The current increases as v_i rises beyond V_{tn} and decreases as it approaches V_{tp}. Integration of the current over time multiplied by the supply voltage is the energy dissipated during the input transition period.

The shape of the short circuit current is dependent on the duration and slope of the input signal, the transfer characteristics of the P and N transistors (which in turn depend on their sizes, process technology, temperature, etc), and the output loading capacitance of the inverter.

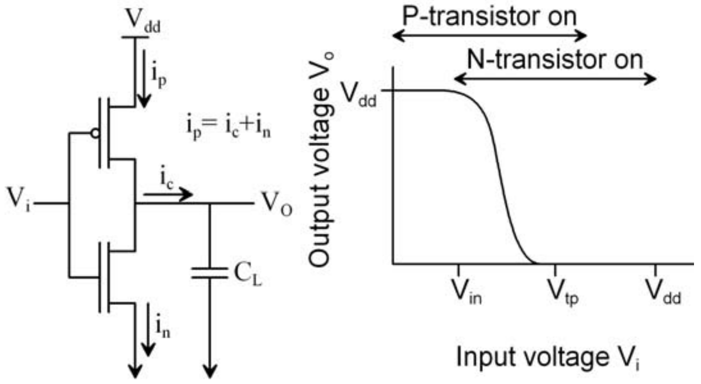
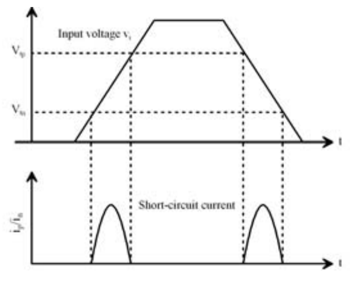


Fig.1.2 Inverter and its characteristics

Assuming the input to be ramp signal, an analysis of the short circuit current shows that the energy dissipated is

$$E_{short} = \frac{\beta}{12} \bullet \tau \bullet \left(V_{tp} - V_{tn}\right)^2 \qquad (1.10)$$

Here β is dependent on the size transistors and τ is the rising duration of the input signal. This equation is useful in only indicative of the relationship of various parameters to the short circuit energy and power dissipation. In practical circuits energy dissipation is a more complex function of the parameters shown in equation 1.10. The equation also assumes that the output loading capacitance of the CMOS inverter is zero which is not true in real circuits.



1.3: Short circuit current in an Inverter

However, the short circuit dissipation can be found accurately using circuit simulators such as SPICE. To obtain the short circuit current when the inverter input changes from low to high, we measure the source-drain current of the N-transistor.

1.3.2. Short circuit current variation with output load

Short circuit current magnitude and the shape varies with respect to the output loading capacitance and the input signal slope of a CMOS inverter. These characteristics of the short circuit current have some implications on the analysis and optimization of the power efficiency for CMOS circuits. We have observed that duration of short circuit current depends on the transition period of the input signal. The short circuit current has also been known to depend on the output loading capacitance.

Consider the case when the input voltage is falling and the output voltage is rising. Based on the on-off properties of the transistors, the short circuit current is non-zero only when the input level is between V_{tn} and V_{tp}. When the inverter input is at $V_{dd}/2$, its output voltage is between 0 and $V_{dd}/2$ assuming a symmetrical transfer curve. If the output capacitance is large, the output voltage is nearly zero and the voltage across the source and drain of the N-transistor is only slightly above zero. The low source-drain potential difference results in small short circuit current. Conversely, if the output capacitance is small, the output voltage rises faster and the source-drain voltage is much higher, causing a larger short circuit current.

It can be shown (typically by SPICE simulation) that the short circuit current envelope is the largest when the output capacitance is the smallest. As the output capacitance increases, the current envelope becomes. The duration of the short circuit current is independent of the output capacitance because it depends on the input signal slope only. The short circuit current is non-zero only when the input voltage is between V_{tn} and V_{tp}. Therefore, increasing the output loading capacitance reduces the short circuit energy per transition, for the same input signal conditions.

This means that increasing the output capacitance decreases the short circuit power dissipation. However, the dynamic dissipation increases as the dissipation component due to increased load capacitance increases. Therefore, low power digital design should always try to reduce load capacitances. A major contribution to this is the input capacitance of the gate receiving the signal. The size of the receiving gate is constrained by its speed requirement and driving a signal faster than necessary results in wasted power. Thus for low power design, a good design practice is to choose the minimum gate size that meets the speed constraint to minimize capacitance. This agrees with the minimum area design goal and presents no conflicting requirements in the area-power trade-off.

1.3.3. Short circuit Current Variation with Input Signal Slope

It can be shown by simulation that as the input signal slope decreases (that is, the duration of the slope increases), the short circuit current peak and duration increase. The width of total current (that is, short circuit + current through the load capacitance) increases and its peak decreases. However, the integration of the total current over time (energy) is increases. The capacitive charging and discharging energy remains identical because the capacitance remains constant.

Therefore, to reduce power dissipation of a CMOS circuit, we should use the fastest input signal slope for all signals. However, while the fast signal reduces power dissipation of the signal *receiving* gate, the transistor sizes of the signal *driving* gate have to be increased to sustain the steep slope. A larger driver gate means more power consumption due to increased input capacitance and short circuit current. Thus, sharp signal slopes presents conflicting effects on the power dissipation of the signal driver and receiver gates. A rule-of-thumb used as a compromise is the duration of the input signal (say, as measured from 10% to 90% voltage) should be comparable to the propagation delay of the gate.

1.4. Static power dissipation and leakage Current

Static power dissipation in a CMOS circuit is due to the leakage current, which is due to a phenomenon of the semiconductor device operation. In circuits that use MOS transistors, there are two major sources of leakage current: 1. reverse biased PN-junction current and, 2. subthreshold channel conduction current.

1.4.1. Reverse Biased PN-junction

This source of leakage current occurs when the source or drain of an N-transistor (P-transistor) is at V_{dd} (V_{ss}). PN-junctions are formed at the source or drain of transistors because of a parasitic effect of the bulk CMOS device structure. As shown in Figure 1.7, the junction current at the source or drain of the transistor is picked up through the bulk or well contact. The magnitude of the current depends on the temperature, process, bias voltage and the area of the PN-junction. The reverse biased PN-junction current is

$$I_r = I_s\left[\exp\left(\frac{V}{V_{th}}\right) - 1\right] \tag{1.11}$$

I_s is the reverse saturation current dependent on the fabrication process and the PN-junction area. V_{th} is the *thermal voltage* where

$$V_{th} = \frac{kT}{q} \tag{1.12}$$

In the above equation, k is the Boltzmann's constant, q is the electronic charge and T is the device operating temperature. At room temperature, T = 300K and V_{th} = 25.9mV.

In equation (1.11) V is negative as the PN-junction is in reversed bias. If $|V| >> V_{th}$, we have $e^{V/Vth}$ is nearly 0 and $I_r = - I_s$. Thus, a small reverse voltage is sufficient to induce current saturation. For all practical purposes, the current is independent of the circuit operating voltage.

The reverse saturation current I_s is of the order of 1 pA/um^2 for many processes. Fo example, for a two-million-transistor chip with an average junction area of 1 x 10um^2 per transistor, the current is about 20uA. This current is very small compared to the dynamic current. For example, an inverter driving a capacitor of 0.5pF operating at 3.3V, 20MHz will consume I = CVf = 0.5pF x 3.3V x 20MHz = 33uA, which is larger than the junction leakage of all two million transistors. The saturation current I_s is highly dependent on the temperature. As a rule of thumb, the saturation current I_s doubles for every ten-degree increase in temperature. The leakage current is becoming significant in recent 90 nm technology. For future technologies like the 65 nm and lower, the leakage current will become very significant. For technologies with larger feature size, this component is negligible at usual operating temperature of the chip.

1.4.2. Subthreshold current

The second source of leakage current is the subthreshold leakage through a MOS device channel. Even though a transistor is logically turned off, there is a non-zero leakage current through the channel, as illustrated in Figure 1.5. This current is known as the *subthreshold leakage* because it occurs when the gate voltage is below its threshold voltage.

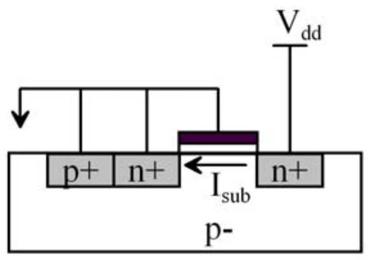


Fig 1.5 Subthreshold current in a MOS transistor

The subthreshold conduction current I_{sub} depends on the device dimension, fabrication process, gate voltage V_{gs}, drain voltage V_{ds} and temperature. The dependency on V_{ds} is insignificant if it is much larger the thermal voltage. In CMOS circuits subthreshold current occurs when $V_{ds} = V_{dd}$ and therefore, the current depends mainly on V_{gs}, device dimension and operating temperature. Subthreshold conduction current is given by

$$I_{sub} = I_0 \exp\left(\frac{V_{gs} - V_t}{\alpha V_{th}}\right) \qquad (1.13)$$

Where V_t is the device threshold voltage, V_{th} is the thermal voltage (25.9mV at room temperature), and I_0 is the current when $V_{gs} = V_t$. The parameter α is a constant depending on the device fabrication process, ranging from 1.0 to 2.5. The expression $(V_{gs} - V_t)$ has a negative value so that I_{sub} drops exponentially as V_{gs} decreases.

A typical plot of I_{sub} with V_{gs} of a MOS transistor is illustrated in Figure 1.6. The slope at which I_{sub} decreases is an important parameter in low power design. By taking the logarithm and differentiation of Equation (1.13), the slope can be expressed as

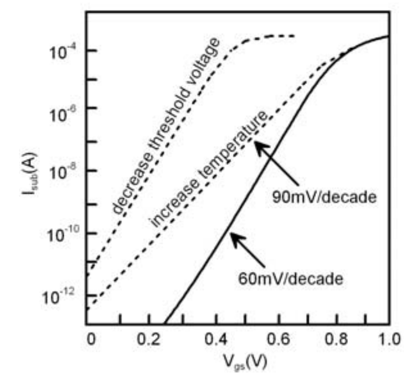


Fig 1.6 Subthreshold current Vs V_{gs}

$$\frac{\Delta V_{gs}}{\Delta\left(\log I_{sub}\right)} = \frac{\alpha V_{th}}{\log e} = 2.3\,\alpha\,\frac{kT}{q} \qquad (1.14)$$

At room temperature with V_{th} = 25.9mV, the slope ranges from an ideal 60mV/decade to 150m V/decade. A smaller value is desirable because the decrease in subthreshold per unit changes in gate voltage V_{gs} is larger. The slope flattens as the temperature T rises.

Subthreshold conduction current is becoming a limiting factor in low voltage and low power chip design. When the operating voltage is reduced, the device threshold voltage V_t has also to be reduced to compensate for the loss in switching speed. Consider a conventional CMOS circuit operating at V_{dd} = 5 V and V_t = 0.9V. Assuming an ideal slope of 60mV/decade in equation (1.14), the subthreshold current is about 900/60 = 15 decades smaller from V_{gs} = V_t to V_{gs} = 0. Such low leakage is negligible. However, in a low power design with V_{dd} = 1.2V, V_t = 0.45V only 450/60 = 7.5 orders of magnitude increase in subthreshold conduction current as compared to the high voltage device. The problem is worsened at high temperature 125 degree Celsius as the slope becomes 80mV/decade and the subthreshold conduction current could deteriorate to 450/80 = 5.6 decades. Decreasing subthreshold current is an important factor in low power design.

1.4.3. Summary on leakage current

Subthreshold leakage and reverse-biased junction leakage have very similar characteristics. They are both in the order of pico-Ampere per device and very sensitive to process variation. Both increase dramatically with temperature and are relatively independent of operating voltage for a given fabrication process. Although the leakage current cannot be ignored in low power circuit design, the logic or circuit designs do not pay any special attention to reduce this current. Leakage current is difficult to predict, measure, or optimize. Most high performance digital circuits operate at high frequencies in which the dynamic power dissipation is several orders of magnitude larger than the leakage current.

1.5. Circuits other than CMOS

In circuits other than CMOS static currents and static dissipation may be significant. For example is the pseudo NMOS logic circuit static current flow from V_{dd} to ground causes power dissipation.

The pseudo NMOS circuit does not require a P-transistor network and saves half the transistors required for logic computation as compared to the CMOS logic. The circuit has a property that the current only flows when the output is at logic 0. When the output is at logic 1, all N-transistors are off and no static current flows except for the leakage current. This property may be exploited in a low power design. If a signal is known to have very high probability of logic 1 it may be better to implement pseudo NMOS logic. Also, if the signal probability is very close to zero, we may replace the NMOS logic with a load transistor of N type.

1.6. Principles of Low Power Design

Analysis of power dissipation mechanisms in Silicon ICs leads us to the following basic mechanisms of low power design.

1.6.1. Switching and supply voltage reduction

The dynamic power of digital chips expressed by Equation (1.8) is the largest portion of power dissipation. Due to the quadratic effect of the voltage term, reducing the switching voltage can achieve significant power savings. The easiest method to achieve this is to reduce the operating voltage of the CMOS circuit. Other methods reduce voltage swing by using well-known circuit techniques such as charge sharing, transistor threshold voltage. There are many trade-offs to be considered in supply voltage reduction. MOS transistors become slower at lower operating voltages as the threshold voltages of the transistors do not scale the operating voltage. Noise immunity is also reduced at low voltage swing.

1.6.2. Capacitance reduction

Reducing parasitic capacitance in is a good way to improve performance as well as reduce power. However, the real goal is to reduce the product of capacitance and its switching frequency. Signals with high switching frequency should be routed with minimum parasitic capacitance to conserve power. Conversely, nodes with large parasitic capacitance should not be allowed to switch at high frequency. Capacitance reduction can be achieved at many design abstraction levels: material, process technology, physical design (floor planning, placement and routing), circuit techniques, transistor sizing, logic restructuring, architecture transformation, and alternative computation algorithms.

1.6.3. Switching frequency reduction

Reducing switching frequency has the same effect as reducing capacitance. Again, frequency reduction is best applied to signals with large capacitance. The techniques are often applied to logic level design and above. Those applied at a higher abstraction level generally have greater impact. Reduction of switching frequency also has the side effect of improving the reliability of a chip as some failure mechanism is related to the switching frequency. An effective method of reducing switching frequency is to eliminate logic switching that is not necessary for computation. Other methods involve alternate logic implementation. The use of different coding methods, number representation systems, counting sequences, and data representations can directly alter the switching frequency of a design.

1.6.4. Leakage and Static Current reduction

Leakage current is not useful in digital circuits. However, we often have very little control over the leakage current as it is technology dependent. In technologies with large feature size (say, above 130 nm) the leakage power dissipation of a CMOS digital circuit is several orders of magnitude smaller than the dynamic power. The leakage power dominates in very low frequency circuits or circuits with "sleep modes" where dynamic activities are suppressed. Most leakage reduction techniques are applied at low-level design abstraction such as process, device, and circuit design. Memories that have very high device density are most susceptible to high leakage power.

Static current can be reduced by transistor sizing, layout techniques and careful circuit design. Circuit modules that consume static current could be turned off if not used. Sometimes, static current depends on the logic state of its output and we can consider reversing the signal polarity to minimize the probability of static current flow.

We note that no single low power technique is applicable to all situations. Design constraints should be viewed from all angles within the bounds of the design specification. Low power considerations should be applied at all levels of design abstraction and design activities. Chip area and speed are the major trade-off considerations but a low power design decision also affects other aspects such as reliability, design cycle time, reusability, testability and design complexity. Early design decisions have higher impact to the final results and therefore, power analysis should be initiated early in the design cycle.

1.7. Figure of merit for low power circuits

The simplest unit of measure of a low power circuit is the power consumption. This measure is useful for packaging considerations, system power supply and cooling requirements. Also the peak power is used for power ground wiring design, signal noise margin and reliability analysis.

An IC that operating at a higher frequency will consume more power. Since power is the rate at which energy is consumed over time, the measure of energy in the unit of Joule becomes another choice of measure, expressed often in nW/MHz. Lower this number, lower the power consumed to perform a given function at a given frequency.

However, when comparing two processors with different instruction sets or architecture, this measure may be misleading. Different processors require different number of clock cycles for the same instructions. A more objective measure in such a case is nW/MIPS or mA/MIPS. Normalizing power dissipation with respect to MIPS allows us to compare processors with widely different performance rating. The nW/MIPS measure has a unit of Watt-Second per Instruction. Thus, it is a measure of the energy consumed by a typical instruction. This figure of merit is useful when comparing power efficiency of processors with similar instruction sets, for example, two DSP processors of the same family. Because of the normalization, this measure is independent of the performance or clock rate of the processor.

When comparing processors with different architectures, nW/MIPS suffers from the same defect as the nW/MHz measure. A RISC processor requires more instructions than a CISC to complete a given task. Recent processor architecture such as the Very Long Instruction Word (VLIW) can actually accomplish many concurrent operations with a single instruction. Thus, a RISC machine will show a lower nW/MIPS number while it may be an inferior computation engine because it takes more instructions to achieve the same computation task. This leads to other measures by substituting MIPS with better measures of performance rating such as SPEC, resulting in nW/SPEC. SPEC is a measure of computation speed derived from executing some standard benchmark software programs written in machine independent high-level programming language.

Other measures such as the *energy delay product* are commonly used to assess the merits of a logic style. When battery life is of concern in portable electronics applications, we often define measures that involve the battery lifetime. Commercial batteries are most often rated with mA-Hour, which is a unit of stored energy when the operating voltage is constant.

The choice of the figure of merit depends on the type of analysis and application area. No single measure is necessary for a particular situation nor is it sufficient for all purposes. It is the designer's responsibility to define the appropriate measures for power analysis and optimization.

Test Problems:

1. Design a CMOS transistor circuit to realize the Boolean function Y= A (B+C).
    a. Simulate the circuit to measure power for input signal probabilities of 0.5 each. Vary the signal probabilities (say 0.3, 0.6, 0.8 etc) and repeat the power measurement.
    b. Restructure (input reordering) the transistors and repeat part (a)
2. Design an inverter chain, which drives a large capacitive load (C_{load}) for optimum delay, area, and power requirements. Determine the number of stages (N) required for optimum delay case. Draw P_d Vs K (stage ratio). Assume C_{load} = 1pF and the gate capacitance of a min. sized inverter C_0 = 26fF.
3. Design single and double edge-triggered flip-flops and find P_d in each case and compare them.
4. Design a four-bit asynchronous counter and measure the power dissipation as a function of frequency. Run this counter with half swing clock and measure the power dissipation and compare this with the earlier case.
5. Design a synchronous 4-bit counter and measure the power. Compare this with the measurements of problem 4. Compare this power with the one obtained with MSB switched off (i.e., when MSB is not in use = > clock to last FF disabled). Measure the power saving.

Chapter 2 SIMULATION POWER ANALYSIS

Computer simulation has been applied to VLSI design for several decades. Most simulation programs operate on mathematical models which mimic the physical laws and properties of the object under simulation. Today, simulation is used for functional verification, performance, cost, reliability and power analysis. Many simulation languages have been developed specifically for IC's. For example in digital logic simulation, VHDL (Very High Speed IC Hardware Description Language) and *Verilog* are two popular languages being used. Special purpose hardware has also been developed to speed up the simulation process.

Simulation-based power estimation and analysis techniques have been developed and applied to the VLSI design process [2.1] [2.3] [2.3] [2.4] [2.5] [2.6]. Simulation software operating at various levels of design abstraction is a key technology in the mainstream VLSI design. The main difference between simulation at different levels of design abstraction is the trade-off between computing resources (memory and CPU time) and accuracy of the results [2.7]. In general, simulation at a lower-level design abstraction offers better accuracy at the expense of increased computer resource. Circuit simulators such as SPICE attain excellent accuracy but cannot be applied to full-chip analysis. Logic simulation generally can handle full-chip analysis but the accuracy is not as good and sometimes the execution speed is too slow. Behavioural-level or functional-level simulation offers rapid analysis but the accuracy is sacrificed. Figure 2.1 summarizes the trade-off between computing resources and analysis accuracy at different levels of design abstraction.

Fig. 2.1

Since no single simulation technique is applicable to all levels of design, the top-down estimation, refinement and verification methodology is used. As an example, the designer may start with a simulation at the hardware behaviour level to obtain an initial power dissipation estimate. When the gate-level design is available, a gate-level simulation is performed to refine the initial estimate. If the initial estimate turns out to be inaccurate and the design fails the specification, the design is modified and verified again. The iteration continues until the gate-level estimate is within specification. The design is then taken to the transistor or circuit-level analysis to further verify the gate-level estimates. The refinement and verification steps continue until the completion of the design process, when the chip is suitable for mass production.

This chapter is dedicated to simulation techniques to estimate and analyze power dissipation of CLSI chips. The material is presented according to the familiar design abstraction levels. For each technique, we discuss its analysis models, application domains, advantages and disadvantages. The concept of *characterization* will be emphasized. Characterisation refers to the process of using lower-level analysis results as a basis to construct higher-level power models. IN this manner, the lower-level details can be abstracted to more compact higher-level models to enable more efficient analysis. For example, the SPICE circuit-level analysis can provide data to construct gate-level power models. Subsequently, the gate-level power simulation results are applied to the architecture-level components. The loss of accuracy as we move up to a higher-level abstraction is a natural outcome that cannot be avoided in order to provide faster analysis. It is important for the readers to appreciate the bottom-up propagation of analysis results and the relationship among analyses at different levels of abstraction.

2.1 SPICE Circuit Simulation

SPICE (Simulation Program with IC Emphasis) is the de facto power analysis tool at the circuit level. A wealth of literature has already been dedicated to SPICE and dozens of SPICE-like application software packages are available today. We will only briefly discuss its application to power analysis, assuming that the readers are already familiar with SPICE.

2.1.1 SPICE Basics

SPICE operates by solving a large matrix of nodal current using the Krichoff's Current Law (KCL). The basic components of SPICE are the primitive elements of circuit theory such as resistors, capacitors, inductors, current sources and voltage sources. More complex device models such as diodes and transistors are constructed from the basic components. The device models are subsequently used to construct a circuit for simulation. Basic circuit parameters such as voltage, current, charge, etc., are reported by SPICE with a high degree of precision. Hence the circuit power dissipation can be directly derived from SPICE simulation.

SPICE offers several analysis modes but the most useful mode for digital IC power analysis is called *transient analysis*. The analysis involves solving the DC solution of the circuit at time zero and makes small time increments to simulate the dynamic behaviour of the circuit over time. Precise waveforms of the circuit parameters can be plotted over the simulation time.

SPICE device models are derived from a characterization process. Each device model is described by dozens of parameters. The models are typically calibrated with physical measurements taken from actual test chips and can achieve a very high degree of accuracy. Lower-level analysis tools using Finite Element Methods or other physical simulation can also be used to produce the device model parameters.

2.1.1 SPICE Power Analysis

The strongest advantage of SPICE is of cause its accuracy. SPICE is perhaps the most versatile among all power analysis tools. It can be used to estimate dynamic, static and leakage power dissipation. MOS and bipolar transistor models are typically available and it also faithfully captures many low-level phenomena such as charge sharing, cross talk and transistor body effect. In addition, it can handle common circuit components such as diodes, resistors, inductors and capacitors. Specialized circuit components can often be built using the SPICE's modeling capability.

SPICE analysis requires intensive computation resources and is thus not suitable for large circuits. Most SPICE-based simulators start to experience memory or computation limitation at several hundred to several thousand devices. Some advanced SPICE simulators can handle circuits upto ten thousand devices but simulating the entire chip is not possible.

With the correct device models, SPICE simulations can reach an accuracy within a few percent of physical measurement. The main source of error in SPACE us seldom found in the computation process but the inherent difficulties in modeling physical component and devices. Like any mass production process, the fabrication of semi-conductor devices is subject to normal fluctuation and therefore SPICE's accuracy is often clouded by the variation of the chip production process. The most common method to cope with this problem is to apply extreme case analysis. Several sets of device models are generated to represent the typical and extreme conditions of the chip fabrication process and operating environment. The conditions are generally defined by the speed of the device so that the device so that the designer can predict the fastest and slowest operating extremes of the circuit. For example, most chip designers will simulate SPICE with three sets of parameters, TYPICAL, BEST and WORST case conditions based on the device speed.

The variation of semiconductor process could be large. The BEST and WORST case device speed could be 2X apart. The process variation of power dissipation is less but is still on the same order. For most circuits using conventional CMOS processes, faster device models generally correspond to higher power dissipation and vice versa. However, there are exceptions : for example, a low-speed worst case device model may cause slow signal slopes with high short-circuit power. For designs with marginal power d\budget, the analysis should be performed on some or all extreme cases to ensure specification compliance. The process variation problem affects all types of power and timing analysis using the bottom-up characterization approach. With SPICE, the problem is more severe due to the accuracy sought at this level of analysis.

2.2 Discrete Transistor Modeling and Analysis

In SPICE, a transistor is modeled with a set of basic components using mathematical equations. The solution of the node currents and voltages requires complex numerical analysis involving matrix operations. Equation (2.1) gives a simple SPICE transistor model to compute I_{ds} as a function of V_{gs} and V_{ds}. The model is obtained by the first order approximation of the nonlinear equation at the operating point V_{gso} and Vr

(2.1)

to form a linear approximation equation. In the small signal model, the equation can be simplified as

(2.2)

which leads to the model shown in Figure 2.2. The linear equation has to be numerically evaluated in SPICE whenever the operating point V_{gso} and V_{dso} changes, resulting in excessive computation requirements.

Fig. 2.2

2.2.1 Tabular Transistor Model

One way to speed up computation is to express the transistor models in tabular forms stored in a database. Now instead of evaluating equations, a simple table lookup is used. The process involves setting the voltage step increments and running SPICE to compute the current c\values for the lookup tables. During circuit analysis, a quick table-lookup is used to obtain i_{ds}, given the operating points V_{gso} and V_{dso}.

Circuit level analysis tools based on tabular transistor models have been reported [2.8]. A one-time characterization effort using SPICE simulation of the MOS transistors with various sizes is required to build the lookup tables for the discrete transistor models. The system was mainly designed for timing and power analysis of digital circuits. To speed up computation, it applies the *event-driven* approach, in which an event is registered when a significant change in node voltage occurs. If the event-driven approach fails, it rolls back to the circuit analysis method. A transient analysis method similar to SPICE is used to perform time step simulation. A notorious problem in simulating large circuits with SPICE is the DC convergence problem. Bulk of the simulation time is often attributed to find the initial DC solution of the circuit so that it converges to a legal state. By using the knowledge that the circuit is mainly digital, the logic values from the primary inputs can be propagated to the circuit to help solve the DC convergence problem.

The transistor model quantization process introduces inaccuracies but improves the speed of analysis. The maximum circuit size and analysis speed using the tabular transistor model improves nearly two orders of magnitude compared to SPICE. The accuracy loss is mostly tolerable for digital circuits. Even with the increased capacity. Transistor-level tools are often not able to perform full-chip power analysis of very large scale chips. The analysis speed is still several orders of magnitude slower than logic-level simulation. Chips with up to a hundred thousand transistors can be handled by transistor-level tools byt many VLSI chips today exceed that scale.

2.2.2 Switch Level Analysis

Most digital circuit analysis is restricted to several basic circuit components such as transistors, capacitors and resistors. Because of the restricted component types, computation speed and memory can be improved by using higher-level abstraction model with little loss in accuracy. One such analysis is called *switch-level* simulation.

The basic idea of the switch-level simulation is to view a transistor as a two-state on-off switch with a resistor, as shown in Figure 2.2. A transistor is turned on when its gate voltage exceeds the threshold voltage. Under this model, timing simulation can be performed using approximated RC calculation that is more efficient than transistor model analysis.

Simulation tools for switch-level timing analysis have been reported [2.9] [2.10]. Recently, power analysis tools based on these simulators have also been developed. The power dissipation is estimated from the switching frequency and capacitance of each node. Short-circuit power can also be accounted by observing the time in which the switches form a power-ground path. The accuracy of switch-level analysis is worse than circuit-level analysis but offers faster speed.

2.3 Gate-level Logic Simualtion

Simulation-based gate-level timing analysis has been a very mature technique in today's VLSI design. The component abstraction at this level is logic gates and nets. The circuit consists of components having defined logic behaviour at its inputs and outputs, such as NAND gates, latches and flip-clops. Most gate-level analysis can also handle capacitors and some can also handle resistors and restricted models of inter-connect wires. Gate-leel logic simulation software is one of the earliest CAD tools being developed. Today, many gate-level logic simulators are available, most of which can perform full-chip simulation up to several million gates.

2.3.1 Basics of Gate-level Analysis

The most popular gate-level analysis is based on the so called *event-driven* logic simulation. Events are zero-one logic switching of nets in a circuit at a particular simulation time point. As one switching event occurs at the input of a logic gate, it may trigger other events at the output of the gate after a specified time delay. Computer simulation of such events provides a very accurate pre-fabrication logic analysis and verification of digital chips. Most gate-level simulation also supports other logic states such as, “unknown”, “don’t care” and “high-impedance”, to help the designer to simulate the circuit in a more realistic manner. Some simulators offer an extensive set of logic states for added accuracy in timing analysis. *Verilog* and VHDL are two popular languages used to describe gate-level design.

Recently, the *cycle-based* simulators are being introduced into the design community. Such simulators assume that circuits are driven by synchronous master clock signals. Instead of scheduling events at arbitrary time points, certain nets of the circuit are only allowed a handful of events at a given clock cycle. This reduces the number of events to be simulated and results in more efficient analysis.

Many gate-level simulators are so mature that special purpose computer hardware has been used to speed up the simulation algorithms. The idea is similar to the graphic co-processor in a computer system. Instead of using a general purpose CPU to execute the simulation program, special purpose hardware optimized for logic simulation is used. This *hardware acceleration* technology generally results in several factors of speedup compared to using a general purpose computing system.

Another technology that offers several orders of magnitude speedup in gate-level analysis is called *hardware emulation*. Instead of simulating switching events using software programs, the logic network is partitioned into smaller manageable sub-blocks. The Boolean function of each sub-block is extracted and implemented with a hardware table mapping mechanism such as RAM or FPGA. A reconfigurable inter-connection network, carrying the logic signals, binds the sub-blocks together. Circuits up to a million gates can be emulated with this technology but this is also the most expensive type of logic simulator to operate and maintain because of the sophisticated high-speed hardware required. The simulation speed is only one to two orders of magnitude slower than the actual VLSI chips to be fabricated. For example, a 200 MHz CPU can be emulated with a 2MHz clock rate, permitting moderate real-time simulation.

2.3.2 Capacitive Power Dissipation

Gate-level power analysis based on logic simulation is one of the earliest power analysis tools developed. The basic principle of such tools is to perform a logic simulation of the gate-level circuit to obtain the switching activity information. The information is then used to derive the power dissipation of the circuit.

A major advantage of gate-level power analysis is that the $P = CV^2f$ equation can be computed precisely and easily. In a non-logic abstraction such as SPICE, the notion of the frequency of a node is not well defined because it has an analog waveform that is potentially non-periodic and non-digital. In logic simulation, the switching activities of each node can be monitored to determine its frequency. The power dissipated due to charging and discharging capacitors can be easily computed. Each net i of a gate-level circuit is associated with a capacitance C_i and a counter variable t_i. As simulation progresses, a logic switching at net i increments the counter t_i. At the end of the simulation, the frequency of net i is given by $f_i = t_i /(2T)$ where T is the simulation time elapsed. The capacitive power dissipation of the circuit is

(2.3)

The simple gate-level power calculator is very useful in providing a quick estimate of the chip power dissipation.

In the pre-layout phase, the capacitance C_i can be estimated, as will be discussed in Section 2.3.5. After floorplanning, the node capacitance can also be estimated from the partition and placement of the gates. At the post-layout phase, the capacitance of a node can be accurately extracted from the mask geometry. Many commercial CAD tools can perform the layout extraction for power and timing analysis.

2.3.3 Internal Switching Energy

Equation (2.3) only computes the power dissipated due to charging and discharging of node capacitance. If a node appears inside a logic cell, its switching activities are not accounted because the logic-level abstraction does not define internal nodes. Short-circuit power is also not captured by the equation. The dynamic power dissipated inside the logic cell is called *internal power*, which consists of short-circuit power and charging/discharging of internal nodes.

For a simple logic gate, the internal power consumed by the gate can be computed through a *characterization* process similar to that of timing analysis for logic gates [2.3]. The idea is to simulate the "dynamic energy dissipation events" of the gate with SPICE or other lower-level power simulation tools. For example, in a NAND gate with inputs A, B and output Y, the logic event "A = 1, B switches from 0 to 1" causes the output to switch from 1 to 0 and consumes some amount of dynamic energy internally. The energy is caused by short-circuit current or charging/discharging of internal nodes in the gate. The dynamic energy dissipation event can be easily observed during logic simulation.

The computation of dynamic internal power uses the concept of logic events. Each gate has a pre-defined set of logic events in which a quantum of energy is consumed for each event. The energy value for each event can be computed with SPICE circuit simulation. For example, a simple 4-transistor NAND gate has four dynamic energy dissipation events as shown in Figure 2.3(b). The typical energy consumption of each event is also showin in the figure. This energy accounts for the short-circuit current and charging or discharging of internal nodes of the gate. With the energy associated with each event, we only need to know the occurrence frequency of each event from the logic simulation to compute the power dissipation associated with the event. The computation is repeated for all events of all gates in the circuit to obtain the total dynamic internal power dissipation as follows

(2.4)

In the above equation, E(g,e) is the energy of the event *e* of gate *g* obtained from logic gate characterization and f(g,e) is the occurrence frequency of the event on the gate observed from logic simulation. The parameter E(g, e) depends on many factors : process conditions, operating voltage, temperature, output loading capacitance, input signal slopes, etc.

Fig. 2.3

Note that the dynamic energy dissipation events not only depend on the Boolean function of the gate, but also the implementation of the gate. Figure 2.4 shows two different implementations of a two-input NAND gate. The first implementation has only four energy dissipation events as shown but the second implementation has two additional events due to the switching of its internal nodes. Given a transistor netlist of a logic gate, the task of determining the complete set of dynamic energy dissipation events requires careful consideration to avoid errors in power analysis.

2.3.4 Static State Power

A similar event characterization idea can also be used to compute the static power dissipation of a logic gate. In this case, the power dissipation depends on the *state* of the logic gate. For example, a two-input NAND gate has four distinct states, as shown in Figure 2.3(c). Under different states, the transistors operate in different modes and thus the static leakage power of the gate is different. As we have discussed in Section 1.4, the leakage power is primarily determined by the subthreshold and reverse biased leakage of MOS transistors. During logic simulation, we observe the gate for a period T and record the fraction of time T(g,s)/T in which a gate *g* stays in a particular state*s.* We perform this observation for all states of the gate to obtain the static leakage of the gate and repeat the computation for all gates to find the total static power P_{stat} as follows :

Fig. 2.4

(2.5)

In the above equation, P(g,s) is the static power dissipation of gate *g* at state *s* obtained from characterization. The state duration T(g,s) is obtained from logic simulation. It is the total time the gate *g* stays at state *s.* The static power P(g,s) depends on process conditions, operating voltage, temperature, etc.

2.3.5 Gate-level Capacitance Estimation

Capacitance is the most important physical attribute that affects the power dissipation of CMOS circuits as evident from Equation (2.3). Capacitance also has a direct impact on delays and signal slopes of logic gates. Changes in gate delays may affect the switching characteristics of the circuit and influence power dissipation. Short-circuit current is affected by the input signal slopes and output capacitance loading (See Section 1.3). Thus, capacitance has a direct and indirect impact on power analysis. The accurate estimation of capacitance is important for power analysis and optimization. Two types of parasitic capacitance exist in CMOS circuits : 1. device parasitic capacitance; 2. wiring capacitance.

The parasitic capacitance of MOS devices can be associated with their terminals. The gate capacitance is heavily dependent on the oxide thickness of the gate that is process dependent. The design dependent factors are the width, length and the shape of the gate. Typically, the shape of a transistor gate is rectangular and the width and length of the gate determine its capacitance. For a gate that "bends", e.g., L-shaped, a correction factor can be used to find its equivalent rectangular width and length. The source and drain capacitance is also estimated from a similar method. The primary capacitance contribution of source and drain terminals is the area and shape of the diffusion regions. In general, a larger transistor has more capacitance in all of its terminals.

In the cell-based design environment, the design and layout of the library cells are made available before the chip design. The capacitance of each pin of a cell is therefore fixed by its circuit and layout. The pin capacitance of a cell can be accurately measured and stored in the cell library. One way to measure the pin capacitance is to use SPICE circuit simulation with the help of the capacitor I-V equation

(2.6)

We vary the pin voltage <V of the cell in time <T and observe the current *i* to obtain the capacitance C. This measurement can be performed during the characterization of the cell.

The second source of parasitic capacitance is wiring capacitance. Wiring capacitance depends on the layer, area and shape of the wire. Typically, the width of routing wires is et to the minimum and the wiring capacitance is estimated from the lengths of the wires. In practice, the process dependent factors of wiring capacitance are expressed by a capacitance-per-unit-length parameter that depends on the thickness of the wire, its distance from the substrate and its width. Once the length of a wire is known, wiring capacitance can be computed.

Since wiring capacitance depends on the placement and routing of the gate-level netlist, accurate estimation cannot be obtained before the physical design phase. However, there is still a need to perform capacitance *estimation* before physical design because the latter are lengthy processes. One way to solve this problem is to predict the wire length of a ent from the number of pins incident to the net. This is called the *wire-load* model in ASIC terms. A wire-load model provides a mapping of the net's pin-count to the wiring capacitance without actually knowing the exact length of the net. The mapping table can be constructed from historical data of existing designs. Sometimes, the function also depends on the number of cells of the circuit because the mapping of a ten-thousand-cell module and that of a one-thousand-cell module may be very different. Pre-layout wire-load model coupled with pin capacitance characterization can provide a good capacitance estimate for gate-level power analysis. At the post-layout phase, when the actual lengths of the wires are known, the true wiring capacitance of a net can be used to verify the pre-layout analysis.

2.3.6 Gate-level Power Analysis

The previous sections presented the techniques to obtain the capacitive, internal and static power dissipation of a gate-level circuit using logic simulation. The event-driven gate-leel power simulation is summarized as follows :

1. Run logic simulation with a set of input vectors
2. Monitor the toggle count of each net; obtain capacitive power dissipation P_{cap} with Equation (2.3)
3. Monitor the dynamic energy dissipation events of each gate; obtain internal switching power dissipation P_{int} using Equation (2.4)
4. Monitor the static power dissipation states of each gate; obtain static power dissipation P_{stat} with Equation (2.5)
5. Sum up all power dissipation components.

The total power dissipation of the circuit is the sum of the three power components expressed in Equations (2.3), (2.4) and (2.5)

(2.7)

The choice of simulation vectors needs to be considered carefully because power dissipation depends on them. The vectors should be chosen such that they reflect the desired operating conditions in which power analysis is sought. Generally, the same vectors used for functional simulation and verification can be used for power analysis. The selection of simulation vectors is application dependent and should be determined from the operating environment of the chip. The vectors for test analysis are obviously not suitable for average power measurement because a chip does not normally operate in the test mode. For example, the simulation vectors of a CPU can be obtained from its instruction trace of standard benchmark software with the proper instruction mix.

The static and internal power dissipation of a gate depends on several factors such as the operating voltage and temperature, output load capacitance, input signal slopes, fabrication process, etc. To capture the power dissipation variation due to such conditions, case analysis can be applied. The gate is simulated with SPICE for all possible conditions that can affect the power. The results are stored in a multi-dimensional table after cells are characterized. During analysis, the actual conditions of the circuit under simulation are specified by the user and the correct internal power or energy values will be used for analysis.

An automated power characterization and analysis tool using this technique has been reported [2.3]. The analysis speed of a gate-level tool is fast enough to allow full-chip simulation. Within the static and internal power characterization mentioned above, the accuracy within 10-15% of SPICE simulation is possible. Commercial CAD tools based on this analysis method have been introduced.

Detailed power dissipation of the logic network can be obtained with a simulation technique. For example, we can observe the power dissipation on a per-gate or per-net basis. We can compute the power dissipation of a module to determine the width of the power supply and ground lines to the module. Coupled with layout information, we can plot the temperature profile of the chip to detect regions with potential reliability problems.

A major disadvantage of gate-level analysis is that signal glitches cannot be modeled precisely. Signal glitches are inherently analog phenomena and the simplified zero-one logic model in gate-level analysis fails to capture their effects. The presence of glitches is very sensitive to the signal and gate delays of the circuit. Signal glitches can be a significant source of power dissipation in some VLSI circuits, as such cannot be ignored. However, it is difficult for any analysis model above the logic level to account for the signal glitches precisely. Some attempts to analyse glitches using probabilistic techniques have been reported [2.11]. One technique uses probability to express the presence of glitches but the unpredictable nature of signal glitches prohibits a precise deterministic analysis.

2.4 Architecture-level Analysis

Architecture-level abstraction is the next design abstraction level above logic gates. This is also called block-level or macro-level design. The basic building blocks at this level are registers, adders, multipliers, busses, multiplexors, memories, state machines, etc. Each component performs a dedicated "high-level" function as perceived by the designer.

Today, architecture-level power analysis is becoming more important because more digital circuits are now synthesized from architectural description. As VLSI chips increase in size and complexity, it has become inefficient to design each individual gate. Over the years, the primary design abstraction has moved from the mask, to transistors, to gates and now to the architecture level. The lower level descriptions, such as gates, transistors and masks can now be automatically generated.

The *dynamic event and static state* characterization method for logic gates mentioned in Section 2.3.3 and section 2.3.4 cannot be practically applied to the architectural components because there are too many events and states. Consider a 16-bit adder in which one of the least significant input bits switches from logic 0 to logic 1. Depending on the logic value of the other inputs, the switching may or many not trigger a carry propagation inside the adder. The power dissipation is thus different depending on the logic values of the other inputs. In the worst case, we may need a to enumerate $2^{(16+6)}$ (4.29 billion) possible events to fully characterize the 16-bit adder with the gate-level characterization method. The enumeration is finite but certainly not practical to compute.

2.4.1 Power Models based on Activities

A distinguished feature of most architecture-level components is that they have well-structured regularity. The components are typically constructed by cascading or repeating simpler units built from logic gates. One way to characterize the architectural components is to express the power dissipation as a function of the number of bits of the components and their operating frequencies [2.4]. For example, the power dissipation of an adder can be expressed as

(2.8)

where *n* is the number of bits, *f* is the frequency of the addition operation, K_1 and K_2 are empirical coefficients derived from characterization with a lower-level power analysis such as gate-level simulation. This simple power model depends only on the operating frequency and size of the adder. The model does not take into account the *data dependency* of the power dissipation. For example, if one input of the adder is always zero, we would expect the power dissipation to be less compared with the case when both inputs are changing.

A more accurate model [2.12] that can capture data dependency is to characterize the power dissipation as

(2.9)

for each input signal *i* of the component. Again we have to perform characterization to derive the coefficients k_i. In some cases, the number of coefficients can be reduced because of the particular characteristics of the component. For example, since addition is commutative, the coefficients for the i-th bit of input A and the i-th bit of input B can be set to the same value with only a slight loss in accuracy. In other components, more K_i's can be set to the same value, such as the coefficients for the data bits of a register file. For larger components with deep logic nesting, e.g., multipliers, the power might also be dependent on the switching activities of the outputs, resulting in a power model as follows :

(2.10)

Equations (2.9) and (2.10) require a factor K_i for each input *i*. Sometimes this is too tedious to characterize and we can simplify them to

(2.11)

The characterization of the coefficients involves generating random signals with the specified frequencies to exercise the architectural components. The power dissipations of the components are observed and the K factors are derived by regression fitting.

2.4.2 Power Model Based on Component Operations

Yet another method to model power dissipation is to express the power in terms of the frequency of some primitive operations of an architecture component. This works well because most architecture-level components only have a few well-defined operations. For example, the power dissipation of a small memory component can be written as

(2.12)

The parameters f_{read} and f_{write} are the frequencies of READ and WRITE operations, respectively. The parameters can be obtained from simulation ands the coefficients K_1 and K_2 are obtained from characterization and properties of the component.

Strictly speaking, the power dissipation of the READ and WRITE operations of a memory component is also dependent on the actual address and data values. However, modeling such fine details of the power dissipation could be prohibitive in terms of computation. The problem is similar to the logic event based modeling in which the enumeration of the logic conditions is too large for our computer to handle. The compromise is to use the average READ and WRITE energy of the operations in Equation (2.12), which introduces some inaccuracies, but improves the computation efficiency and generality of the power mode. If the address and data values of the memory operations are fairly random, this solution is often very effective in practice.

In some cases, the memory access pattern is skewed such that most of the READ and WRITE operations occur at a particular location, e.g., address zero. If this is known a priori, the power model can be modified to improve the accuracy. For example, we can characterize the zero address (ZA) and the non-zero address (NZA) operations as

(2.13)

2.4.3. Abstract Statistical Power Models

As we can see from the power models for the various architecture-level components, we have to make certain assumptions on the statistical behaviours of the components in order to model the power dissipation efficiently. The primary reason is that a precise model that characterizes the exact signal events of the architecture-level components is computationally inefficient. The large number of I/O pins of the components prohibits the use of logic events such as the one used for gate-level logic analysis. Power models based on the idea of pin toggling, operations, probabilities and other statistical quantities are used to analyse the power dissipation of the architecture components.

The techniques used for characterizing memory components in Section 2.4.2 can be generalized and formalized to the concept of power models based on abstract statistical parameters of the components. There are several fundamental issues in composing a power model:

1. Which parameters have major influences on the power dissipation ? Size, signal frequency, operation frequency, etc.
2. What is the relationship of the power dissipation with the chosen parameters ? Equation or lookup table.

The answers to the questions strongly depend on the type and implementation of the components. They also depend on the accuracy level tolerated by the analysis. Adders, multipliers, memories and registers all have very different operation characteristics and require different parameters for analysis.

In general, the architecture-level power models are not as robust and versatile as the gate-level models because some assumptions have to be made in the modeling and characterization to be computationally efficient. To maintain accuracy, the analysis tools have to observe the operation characteristics of the components to make sure that the assumptions made by the power models are not violated. Nevertheless, the inability to compose accurate power models does not diminish the need for power analysis at the architecture level. A good strategy is to compose power models or analysis methods that emphasize the relative accuracy rather than absolute accuracy. If an architecture design X consumes more power than another design Y, a good power estimation technique should provide such insight; even though the actual power dissipation of designs X and Y is difficult to estimate accurately. As the architecture-level abstraction is the focus of major design decisions, the analysis tools at this level will be used primarily for design trade-off and exploration.

2.5 Data Correlation Analysis in DSP Systems

In a Digital Signal Processing (DSP) system, a phenomenon known as *sample correlation* has been observed. Simple correlation refers to the property that successive data samples are very close in their numerical values and consequently their binary representations have many bits in common. This is not a coincidence but a direct result of sampling a band-limited analog signal with a higher sampling rate relative to the analog signal bandwidth./ Some data streams exhibit *negative correlation* (anti-correlation) in which successive samples jump from a large positive value to a large negative value. Negative correlation may occur in some digital signal coding schemes such as delta modulation. Examples of such data streams are depicted in Figure 2.5.

Obviously, positive or negative correlation has a significant effect on the power dissipation of a DSP system because of the switching activities on the system datapath. If we can find the relationship between the data correlation and power dissipation, we can develop a high-level power model without a sample-by-sample analysis of the data stream. Our goal is to estimate power dissipation of an architecture-level component based on the frequency and some correlation measures of the data stream.

Fig.2.5

2.5.1 Dual Bit Type Signal Model

The effect of data correlation on power dissipation often depends on the numerical representation of the digital system. In digital signal processing, the two's complement representation is the most widely used. Another popular numerical representation method is the signed magnitude. We will assume the use of two's complement representation as the analysis method for signed magnitude representation is similar.

Let us observe the toggle characteristics of the data signals under the influence of data correlation. If the data sample is positively correlated, successive data sample values are very close in their binary representation. This means that the Least Significant Bits (LSB) of the data bus toggle frequently while the Most Significant Bits (MSB) are relatively quiet. If we plot the bit-toggle frequencies of the signals, the characteristics shown in Figure 2.6 will be observed [2.13]. Some of the LSB bits toggle at approximately half the maximum frequency. This is called the *uniform white noise* region because the bits toggle in a random fashion. On the MSB side, the bits have a very low toggle rate and they are called the *sigh bit* region. Most of the toggling at this regions is the result of a sign change of the data samples, which occurs infrequently. There is also a grey area between the two regions where the toggle frequency changes from white noise to sign bit. In this region, the bit-toggle rate changes from near zero to 0.5, typically in a linear fashion. Note that the switching frequency is normalized with respect to the maximum toggle rate, which is half of the sampling frequency. For a negatively correlated data stream, the converse is observed. The sign bit region has a very high switching frequency and the noise bit region remains at random. If a data stream exhibits no correlation, all bit switching characteristics will all look like uniform white noise.

The above observation allows us to characterize the data stream with only a few parameters :

Fig. 2.6

1. Sample frequency
2. Data correlation factor from – 1.0 to +1.0
3. The sign bit and uniform white noise regions with two integers.

Such characterization of data signals is called the *dual bit type* model, proposed by Landman and Rebaey [2.13].

2.5.2 Datapath Module Characterization and Power Analysis

The dual bit type signal model provides a very compact representation of the switching characteristics. We like to develop power dissipation models (equations) under such signal excitation so that they can be used for power analysis of architectural components. The power models are sensitive to the signal correlation and the "bit type" of the signals. There are many different methods to characterize the power dissipation and we will describe several methods proposed in [2.13].

For illustration purposes, we first assume that the module to be characterized is a simple single-input single-output block such as a FIFO data queue. We further assume that there is no activity coupling between any two-bit pair so that we can examine a single bit of the component and generalize it to the entire module. We can use a lower-level power analysis tool, such as a gate-level tool, to analyze the power dissipation of a single bit under the uniform white noise signal at a particular frequency f_1 and voltage V_1 . Suppose that the power measured by the lower-level tool is P_1. The effective capacitance C_u of the bit is defined as

(2.14)

The effective capacitance C_u is approximately equal to the capacitance being switched under the white noise signal excitation. Once the effective capacitance is characterized, we can apply it to compute power at a different frequency f_2 and different voltage V_2 under white noise signal excitation

(2.15)

The concept of effective capacitance can also be used on the module bits under the sign bit signal excitation. However, the effective capacitance in this case is no longer a scalar quantity. Between successive data samples, the sign bit may or may not change sign. Therefore, there are four effective capacitance values : C_{++}, C_{+-}, C_{-+}, C_{--}. The subscript sign pairs represent the changes in sign bit polarity between two consecutive samples. For example, the effective capacitance C_{-+} is the capacitance switched when the first sample is negative and next sample is positive. In a FIFO data queue, it is most likely that $C_{+-} = C_{-+}$ and $C_{++} = C_{--}$. But in general, we could construct circuits in which all four effective capacitance variables have different values. With the four effective capacitance values characterized by a lower-level power analysis tool, we can construct a power equation similar to Equation (2.15). Let P_{++}, P_{+-}, P_{-+}, P_{--} be the probabilities that sign changes occur in the data stream. The power equation for the bit excitation under the sign bit signal is

(2.16)

where V is the operating voltage and *f* is the frequency of the data stream. Note that $P_{++} + P_{+-} + P_{-+} + P_{--} = 1$ and for long data sequence $p_{+-} = p_{-+}$ since the number of plus-minus transitions and that of the minus-plus transitions can differ by at most one.

The above discussion has developed the characterization and power evaluation equations for the bit-level power dissipation under white noise and signed signal excitation. For a module that consists of multiple bits, we need to distinguish the white noise bits from the sign bits. This can be done by taking the midpoint of the grey area in Figure 2.6. All bits to the left (right) of the midpoint are considered to have sign bit (white noise) signals. Let N_s (N_u) be the number of bits with sign bit (white noise) signals. The power dissipation P of the module is

(2.17)

The effective capacitance C_u, C_{++}, C_{+-}, C_{-+}, C_{--} are obtained during characterization, once per module design. The variables N_s, N_u, P_{++}, P_{+-}, P_{-+}, P_{--} , are obtained from the correlation factor and signal properties of the data stream, typically by analyzing the data stream from the behaviour-level simulation. Sometimes the variables can be estimated and supplied by the designer. If the data stream has high positive correlation, we will have P_{++}, $P_{--} >> P_{+-}$, P_{--} and the converse if the data-stream is negatively correlated.

Equation (2.17) assumes that there are no interactions among data bits in the module and allows us to characterization one bit and applies the effective capacitance to the other bits. This is no longer true for modules that have interactions among data bits such as barrel shifters. One way to solve this is to perform the characterization for all possible combinations of N_u and N_s. Since $N_u + N_s$ is equal to the number of bits N of the module, there are only N + 1 conditions to be characterized. In this case, Equation (2.17) will be changed to

(2.18)

where $C(N_u)$, $0 \leq N_u \leq N$ is the effective capacitance when there are N_u white noise bits at the module's input.

We now consider modules with two inputs and one output such as adders. The two inputs may have different sign and noise bit separation points. This creates a region at the output of the adder in which the sign and noise bit signals overlap, as shown in Figure 2.7. There are four possible polarity conditions in the sign bit portions of the inputs and output. Therefore, there are 4 x 4 x 4 = 64 possible types of signal transition patterns. A given capacitor in the module may or may not transition under a particular type of transition pattern. This means that there are 64 effective capacitances from $C_{++/++/++}$, $C_{++/++/+-}$ to $C_{--/--/--}$. The "u/ss" condition (IN1 has noise bits and IN2 has sign bits) requires another four effective capacitances and so is the "ss/u" condition. The "u/u" input combination only requires one effective capacitance value. In total, there are 64 + 4 + 4 + 1 = 73 effective capacitance values to be characterized using a lower-level power analysis tool. Note that for an adder, some transition patterns may not occur. For example, $C_{++/++/--}$ is zero because when both inputs have the positive sign, the output cannot have the negative sign. In general, all 73 effective capacitances have to be characterized for an arbitrary two-input one-output module. The power dissipation is given by

(2.19)

The dual bit type characterization method can be generalized to most types of architecture-level components such as multipliers, memories, etc.

Fig. 2.7

Given the modules characterized by their effective capacitance values, a key step in power analysis is to find the boundary between sign and noise bits. This could be done by simulation or probabilistic analysis. After the "bit type" of each input or output is determined, Equation (2.17) or (2.19) is applied to obtain the power dissipation.

The analysis method is a bottom-up approach meaning that the modules have to be pre-designed and pre-characterized before analysis. This bottom-up approach is acceptable for cell libraries because a cell is small enough to be pre-designed and reused. In general, architecture-level components such as adders, multipliers and register files are seldom pre-designed because the reusability of these components is very limited. Most components have to be designed to fit a specific performance and power constraint. However, once they are characterized, we can analyse the power dissipation of the components under various data correlation and dynamic ranges. Characterization of newly designed components is an inevitable process in any bottom-up approach.

2.6 Monte Carlo Simulation

We have seen various approaches to power analysis based on the dynamic simulation of circuits at different levels of abstraction. All simulation-based power analysis systems require a set of *simulation* vectors at the primary inputs of the circuit to trigger the circuit activities. To obtain the power dissipation of the circuit, the switching activity information is collected and applied to the appropriate power models after simulation. The final power dissipation is subsequently reported from the simulation results. The simulation vectors applied to the circuit have a substantial impact on the final power number reported because power dissipation of a digital circuit heavily depends on its switching activities. Each simulation vector causes some energy to be dissipated and the total power dissipation is derived by summing up the energy of each vector and dividing over the simulation time.

It is crucial that the vectors represent the *typical* conditions at which power estimate is sought. For example, to estimate the typical power dissipation of a CPU chip, the vector set has to be chosen such that it reflects the normal instruction mix of the CPU. We would expect that the CPU consumes more power executing an arithmetic instruction than the "no-operation" instruction. If the simulation vectors do not contain the proper instruction mix, the power analysis result will be skewed. For a CPU, the simulation vectors can be obtained from the instruction trace of executing standard benchmark software.

We will not address the problem of selecting simulation vectors since this depends on the chip and its intended application. Like functional simulation and verification, the chip designer is responsible for providing the proper simulation vectors. Sometimes randomly generated vectors are sufficient for power analysis but for others a proper vector sequence is required.

Regardless of how the simulation vectors are generated, if we simulate a circuit with only several vectors, we would expect that the power dissipation result obtained is not truthful because the vector length is too short. Most part of the circuit is probably not exercised enough to obtain the actual toggling activities. On the other hand, we can simulate the circuit for millions and millions of vectors to obtain a very accurate measure of the power dissipation. But is it necessary wasting computer resources to simulate that many vectors ? How much extra accuracy can we achieve by simulating a million vectors versus only a thousand vectors ? How do we know that we have simulated enough vector length ? All these lead to an important question of the *stopping criteria* of the simulation : when do we stop simulation so that the result is accurate enough for our purpose ?

2.6.1 Statistical Estimation of Mean

The stopping criteria of simulation-based power analysis were first studied by Burch, Najm, Yang and Trick [2.14]. To formulate the problem, we define a basic *sample period* *T* in which a single power dissipation value is observed. For example, T may be several vectors or several clock cycles. After a particular simulation period, T_i, the power dissipation P_i of the circuit during the period T_i is computed. As such, we obtained a series of *power samples* P_0, p_1,..., P_N. The estimated power dissipation P of the circuit under simulation is given by the average value of the sample, i.e.,

(2.20)

This is the classical mean estimation problem in statistics in which we draw *N* samples from a large population and try to determine the mean of the population. With a small value of *N*, we would expect that the result *P* is not truthful. If we set *N* to be too large, unnecessary computation will be performed without gaining meaningful accuracy. So, there is a trade-off between sample size (related to computing efficiency) and accuracy. Our questions of stopping criteria now become the determination of the sample size *N*.

The power samples P_i are random variables following some unknown probability density function. The distribution of P_i depends on the circuit, simulation vectors and the sample interval. Let μ and σ_2 be the mean and variance of P_i, respectively. The value is the true power dissipation of the circuit we are trying to estimate from the N samples of pi taken. The *sample mean* $P = (P_o + P_1 + ... + P_N) / N$ is undoubtedly the best estimate of μ we know. The question is : how accurate is P in estimating μ with N samples ?

According to the well-known *central limit theorem* in statistics, the sample mean P approaches the normal distribution for large N regardless of the distribution of P_i. For theoretical development, le us assume that the samples Pi have normal distribution. Basic statistical theory states that the average of normally distributed random variables also has normal distribution. The mean of P is exactly μ and its variance is

(2.21)

As we increase the sample size N, the variance σ^2_p diminishes so that we obtain more accurate measures of the true mean μ. The normal distribution curve for P is shown in Figure 2.8.

Fig. 2.8

For the sake of analysis, we assume that μ. is positive. To quantify the accuracy of the sample mean P, we prescribe a maximum error tolerance ε, typically with values less than 10%. Given ε, we ask what is the probability that P is within the ε error range of the true mean μ.? In other words, what is the probability for the condition $0 \leq |P - \mu| \leq \varepsilon$? If this probability is high, we will trust the estimate P, otherwise, we must increase the sample size N to gain more confidence. We can obtain this probability by integrating the normal distribution curve p(x) in Figure 2.8. For historical reason, the probability is more conveniently expressed by a confidence variable α. The *confidence level* is defined as 100 $(1-\alpha)$%. A confidence level of 100% ($\alpha = 0$) , means that P is absolutely within the error tolerance of ε. [♣]Typically, the confidence level is set to more than 90% to be meaningful, i.e., $\alpha \leq 0.1$.

Next, we want to explore the relationships among ε, α and N. *We define a variable* $z_{a/2}$ *such that the area between* $\mu - z_{a/2}\ \sigma_P$ and $\mu + z_{a/2}\ \sigma_P$ under the normal distribution curve p(x) is $(1 - \alpha)$. From Figure 2.8, this is also the confidence level of the condition $|P - \mu | \leq z_{a/2}\sigma_P$. To ensure the condition $|P - \mu | / \mu \leq \varepsilon$, we require that

(2.22)

Substituting Equation (2.21) into the second inequality of (2.22), we have

(2.23)

[♣] For any non-zero ε, infinite number of samples are required to achieve 100% confidence.

Since α and ε are fixed constants prescribed by the experimenter, the equation can be rewritten as

(2.24)

From Page 53 to 55.

Several precautions must be taken in Monte Carlo power simulation. First the samples P_i have to be statistically independent. This is always a fundamental assumption of the statistical analysis. Let us consider a logic-level Monte Carlo simulation. If the sample period *T* is shorter than the typical gate delay, we could violate the independence assumption. This is because when the input of a gate is switching, the output is very likely to switch after the gate delay. Thus, subsequent samples are highly correlated if the sample period is shorter than the gate delay. Of course, we can always set *T* to be very large to assure independence but this prolongs the simulation time. As a rule of thumb, we want to set *T* such that a large number of switching activities occur during the period *T* to ensure randomness; for examples, several clock cycles. If the signals in the circuit are largely correlated, which is observed to be true for most circuits, the samples P_i should be uncorrelated. We also need to make sure that the simulation vectors do not cause the power samples to become correlated. One way to do this is to pause for a fixed or random amount of time between subsequent samples. As a last resort, if we suspect that the independence assumption is violated, we can always increase the simulation time as a safeguard.

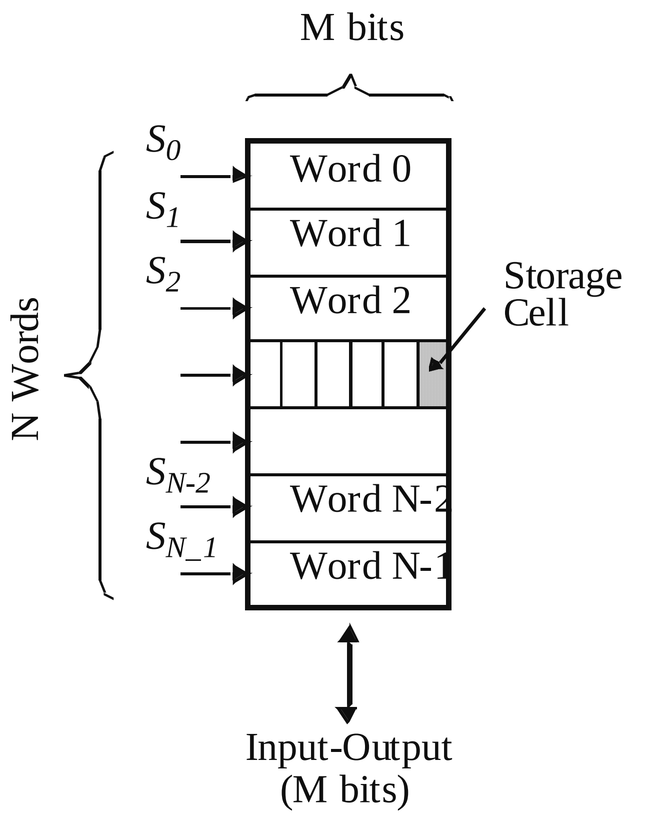
In our theoretical development in the last section, we have assumed that the samples P_i have normal distribution. In general this is difficult to prove for all circuits but there are good reasons to believe that this is true. Note that the power dissipation during a period *T* is the result of a lot of signal switching in the circuit. If we observe an individual signal of a circuit, its contribution to power dissipation certainly does not follow the normal distribution. However, a power sample P_i is a weighted sum of a large number of signal switches throughout the circuit. We can invoke the central limit theorem to claim that P_i has a normal distribution because it is the sum of a large number of random variables.

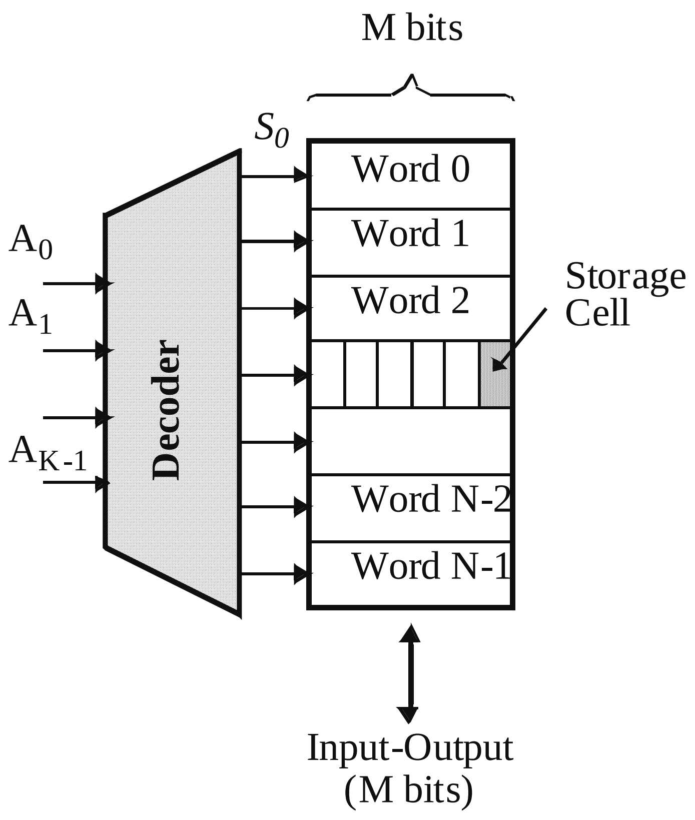
Note that the simulation time (related to sample size *N*) depends on the ratio of the sample variance and sample mean S^2 / P^2. One interesting observation is that this ratio typically decreases when the circuit size increases because S^2 is the average of many random variables. When the circuit size increases, the variance of the samples P_i decreases. This reduces *N* for large circuits due to more uncorrelated nodes. Since it takes more simulation time to compute a power sample P_i, this has an interesting effect that the simulation time is similar for large circuits and small circuits. Intuitively, we can reason that large circuits have more randomness such that subsequent power samples do not vary much. The readers are referred to [2.14] for further theoretical development and experiment results supporting the Monte Carlo simulation approach. A technique for deciding the stopping criteria a priori, i.e., without observing the power samples, has been reported [2.15].

Semiconductor Memory Classification

RWM		NVRWM	ROM
Random Access	**Non-Random Access**	EPROM E^2PROM FLASH	Mask-Programmed Programmable (PROM)
SRAM DRAM	FIFO LIFO Shift Register CAM		

Memory Architecture: Decoders



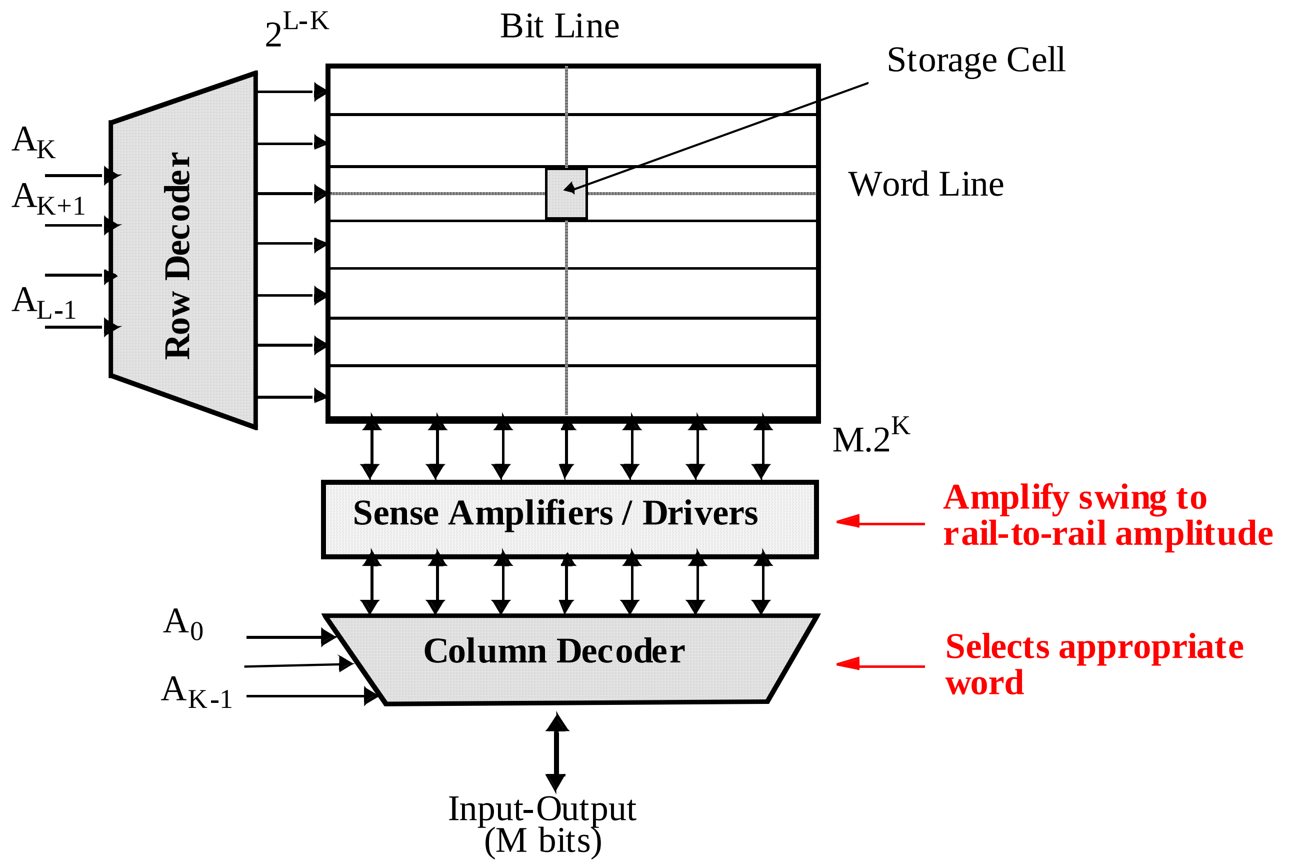


N words => N select signals
Too many select signals

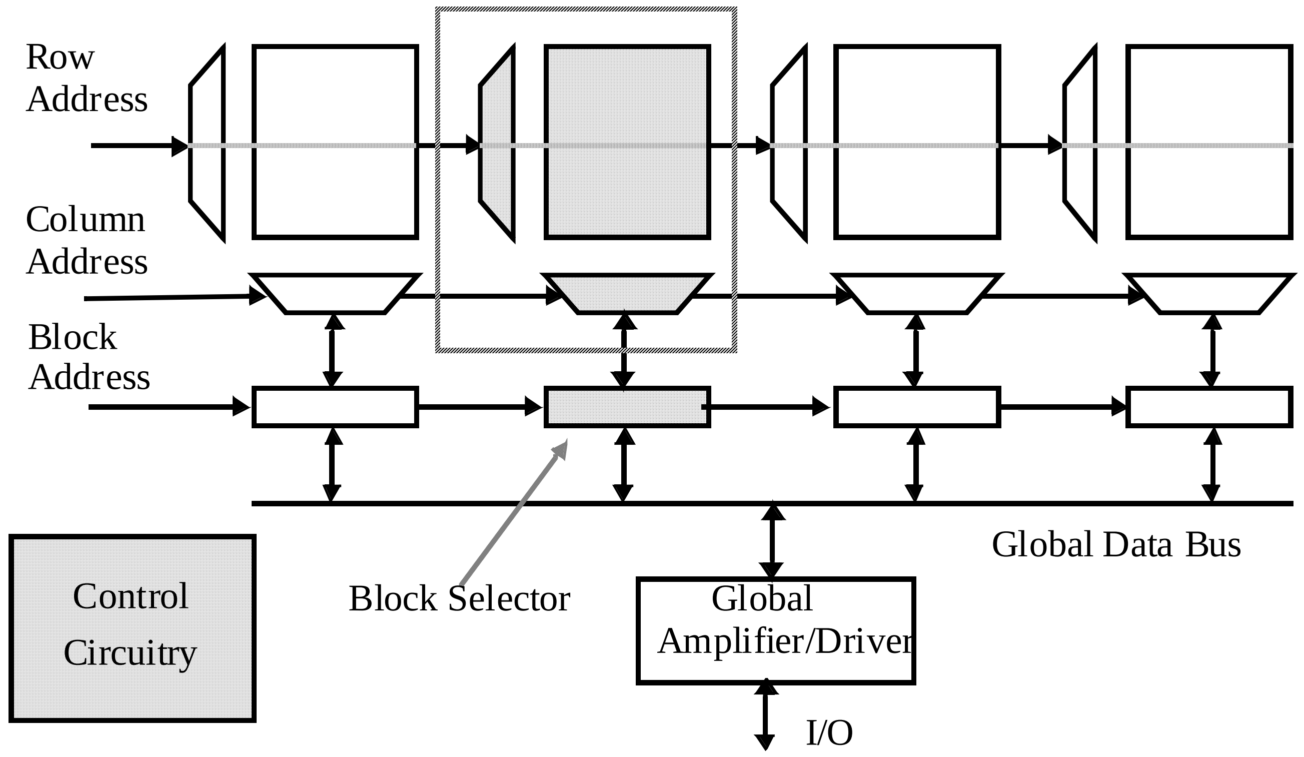
Decoder reduces # of select signals
$K = \log_2 N$

Array-Structured Memory Architecture

Problem: ASPECT RATIO or HEIGHT >> WIDTH

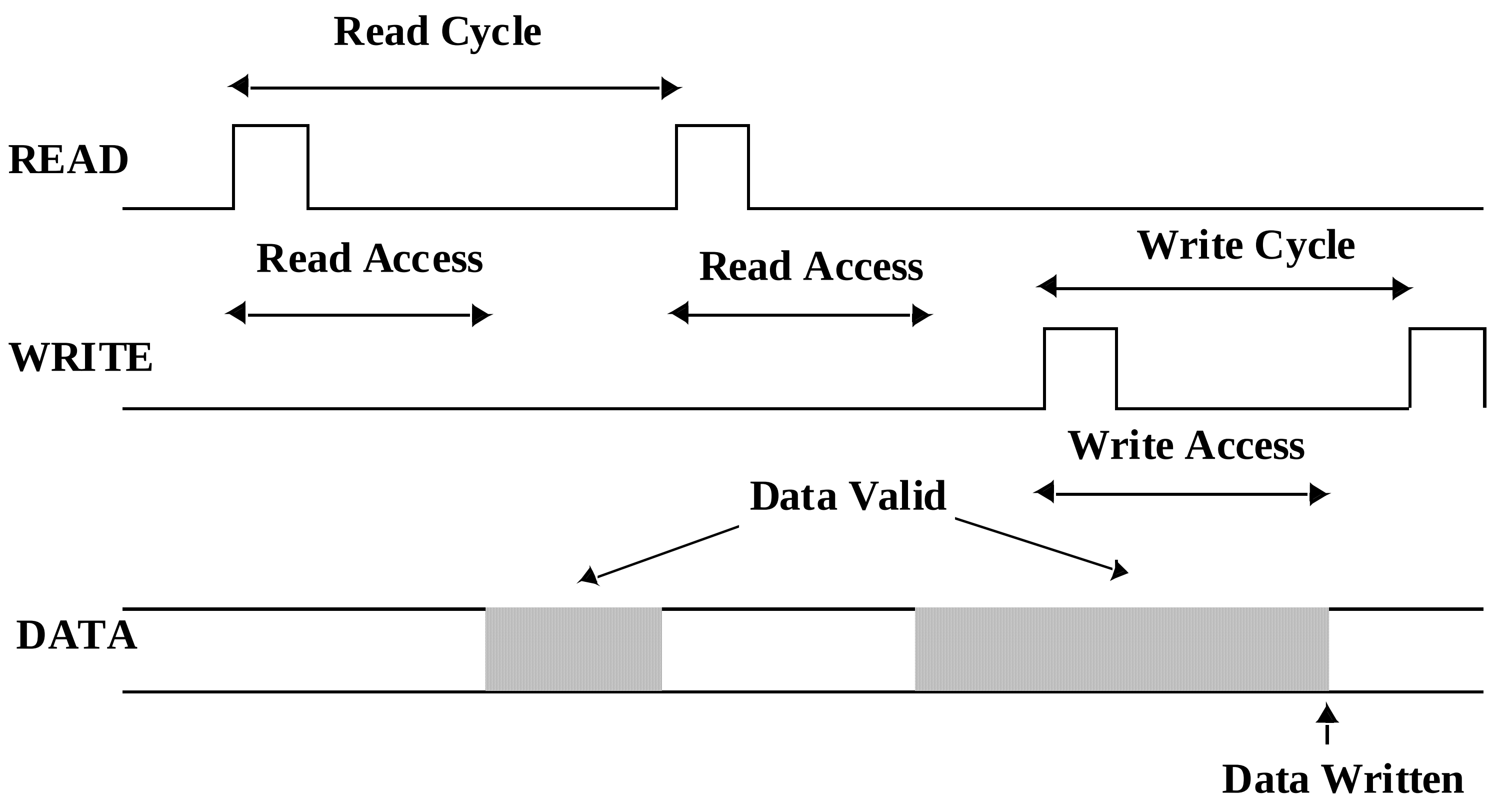


Hierarchical Memory Architecture

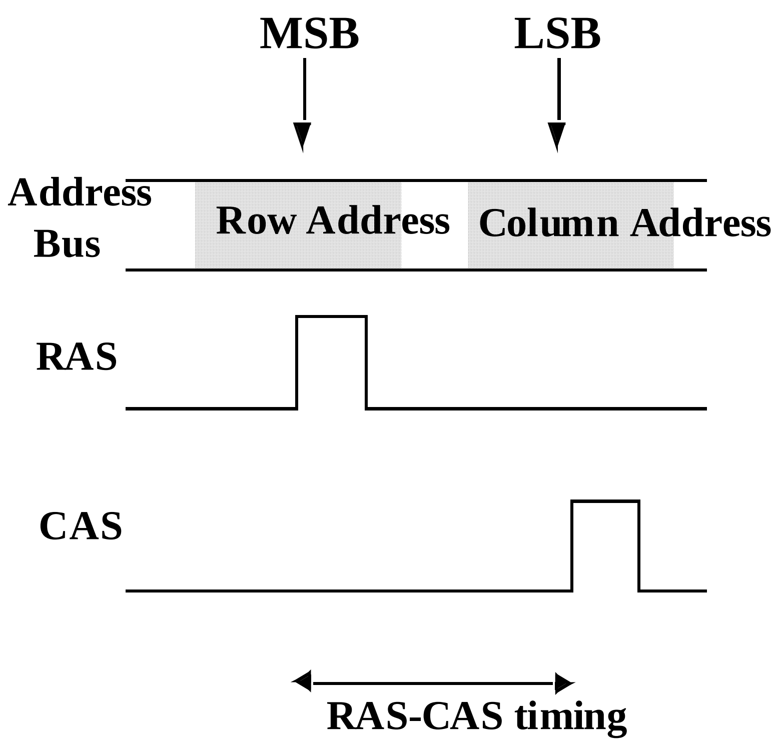


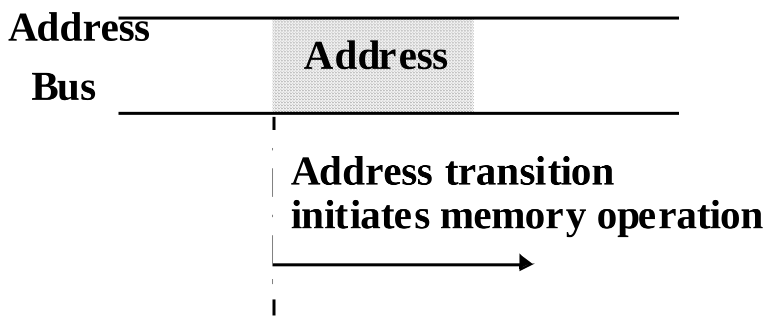
Advantages:
1. Shorter wires within blocks
2. Block address activates only 1 block => power savings

Memory Timing: Definitions



Memory Timing: Approaches

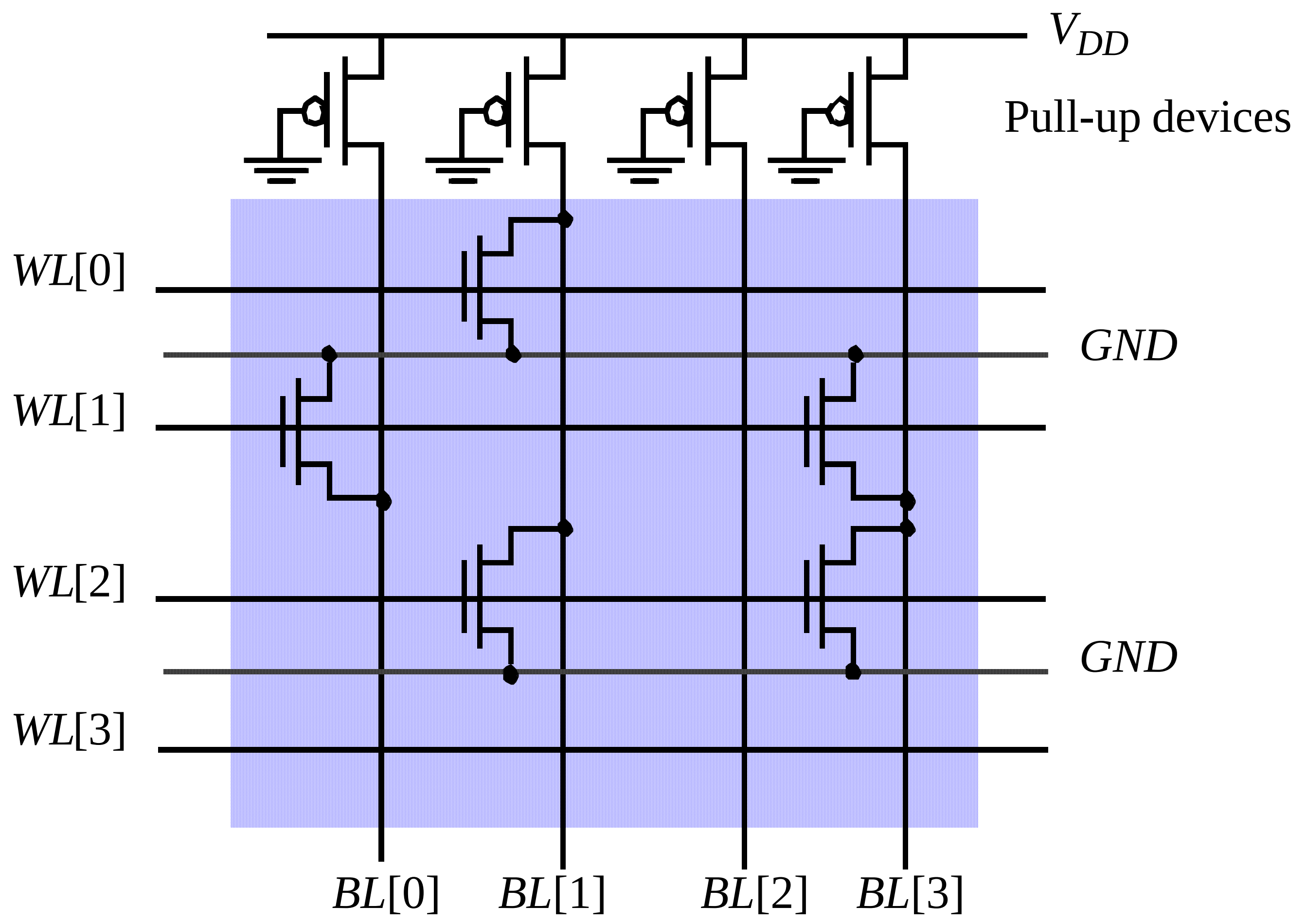




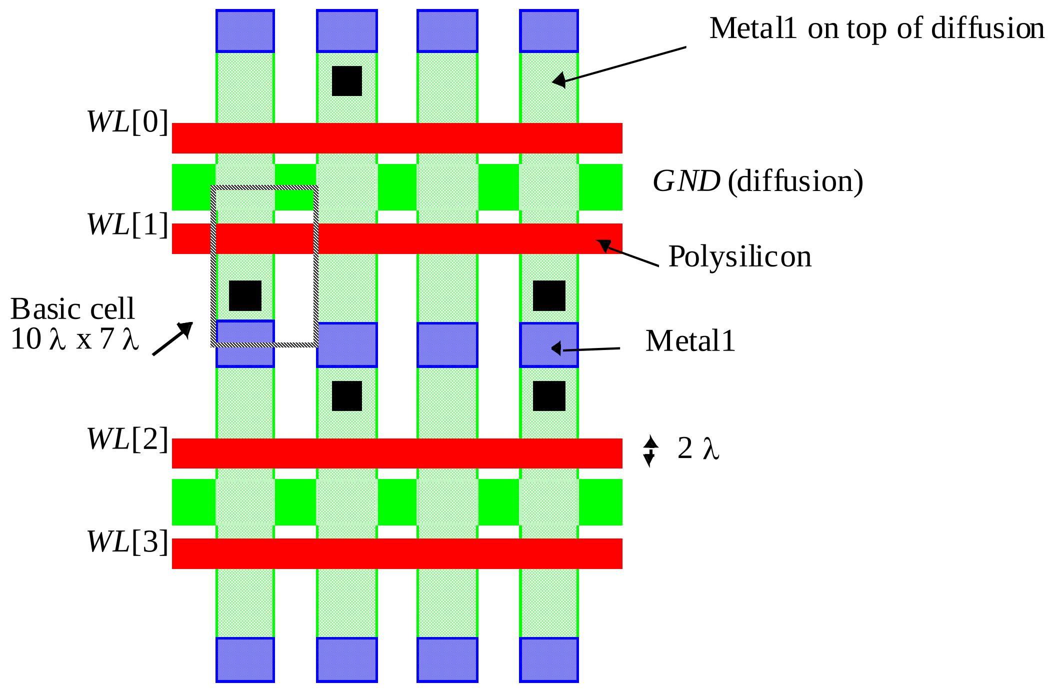
DRAM Timing
Multiplexed Adressing

SRAM Timing
Self-timed

MOS NOR ROM

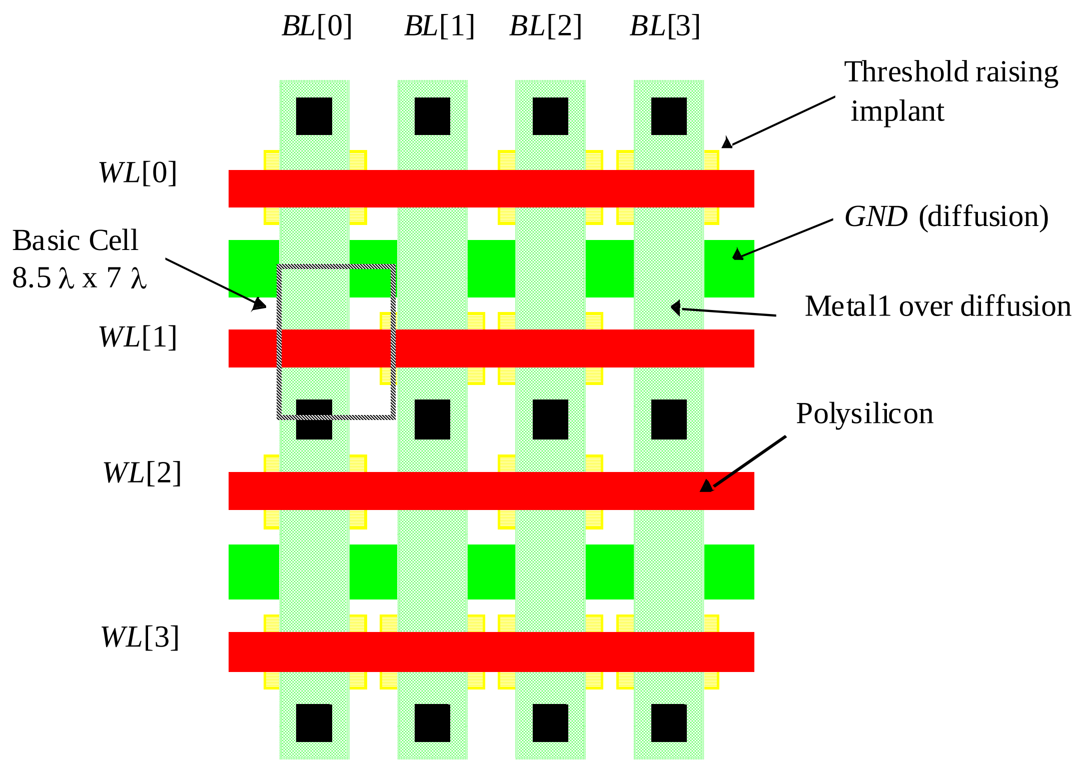


MOS NOR ROM Layout



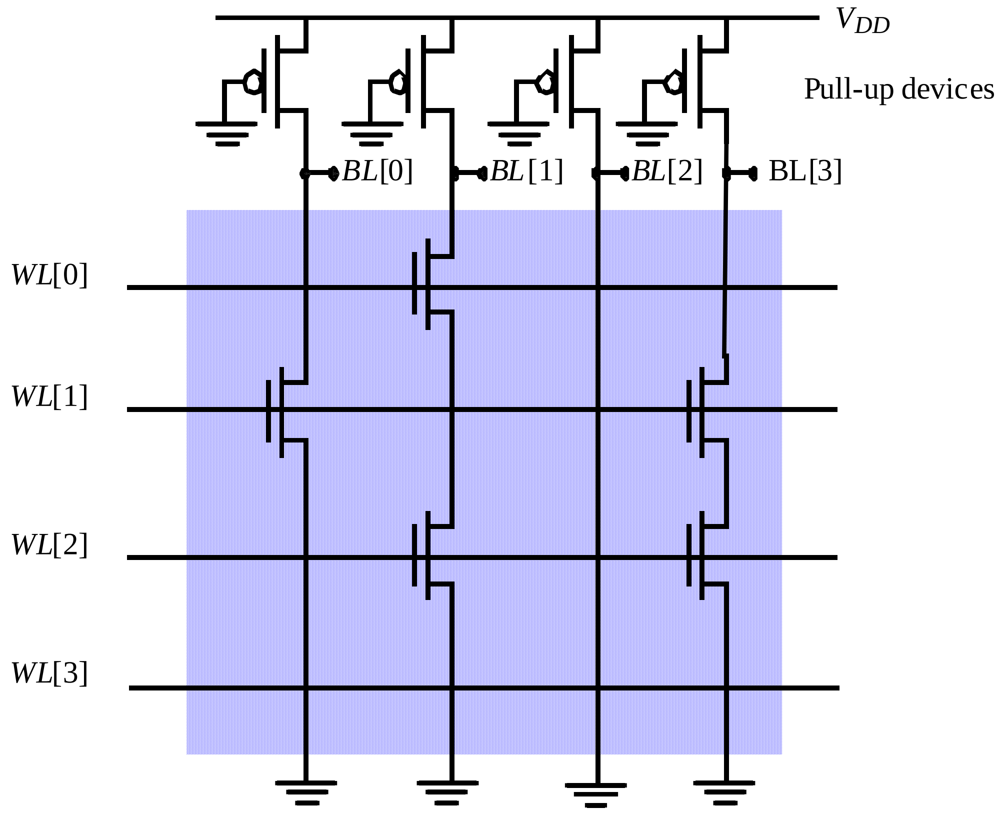
Only 1 layer (contact mask) is used to program memory array
Programming of the memory can be delayed to one of last process steps

MOS NOR ROM Layout



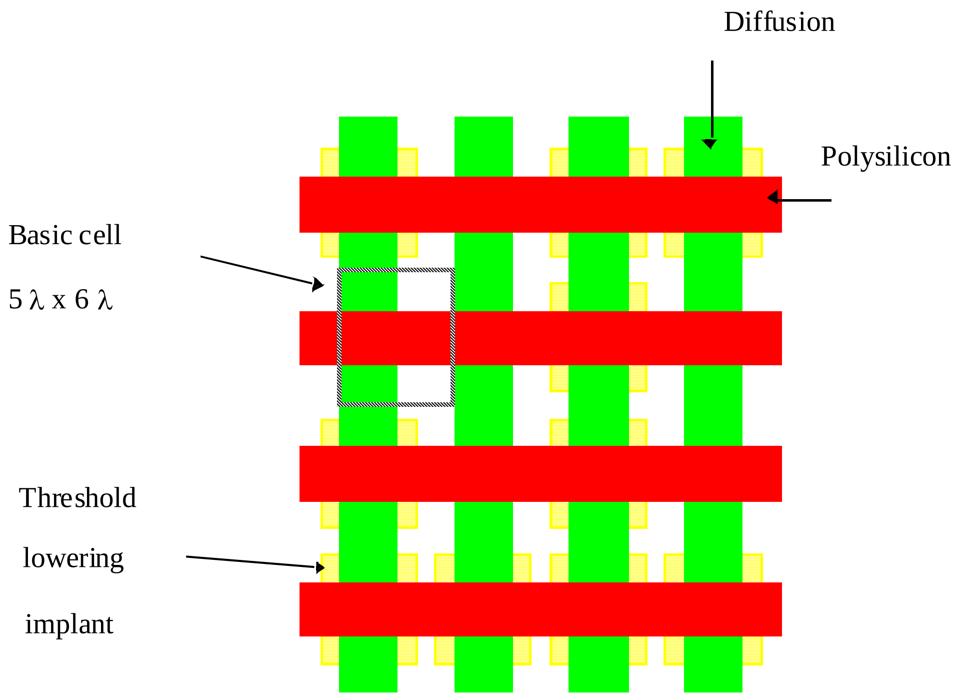
Threshold raising implants disable transistors

MOS NAND ROM



All word lines high by default with exception of selected row

MOS NAND ROM Layout

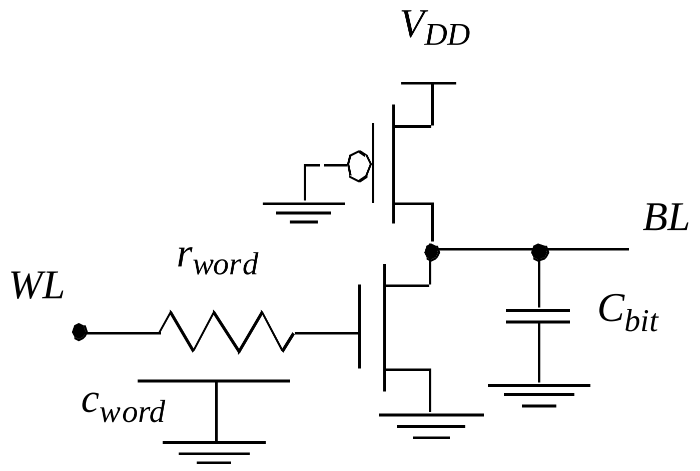


No contact to VDD or GND necessary; drastically reduced cell size

Loss in performance compared to NOR ROM

Equivalent Transient Model for MOS NOR ROM

Model for NOR ROM



Word line parasitics

Resistance/cell: (7/2) x 10 Ω/q = 35 Ω

Wire capacitance/cell: $(7\lambda \times 2\lambda)\ (0.6)^2$ 0.058 + 2 × $(7\lambda \times 0.6) \times$ 0.043 = 0.65 fF

Gate Capacitance/cell: $(4\lambda \times 2\lambda)\ (0.6)^2$ 1.76 = 5.1 fF.

Bit line parasitics:

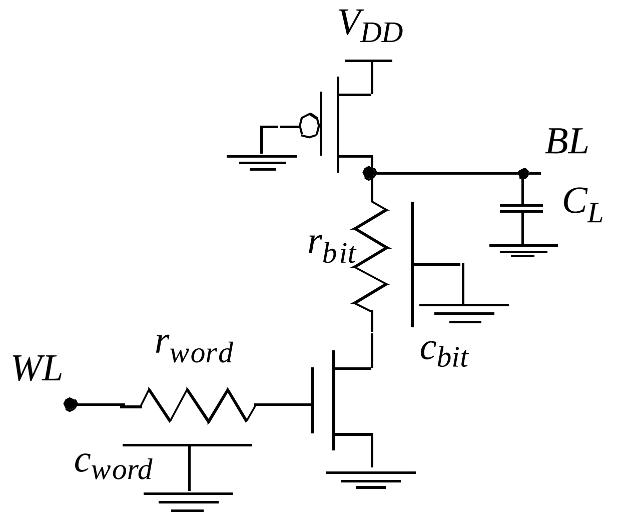
Resistance/cell: (8.5/4) x 0.07 Ω/q = 0.15 Ω (which is negligible)

Wire capacitance/cell: $(8.5\lambda \times 4\lambda)\ (0.6)^2$ 0.031 + 2 × $(8.5\lambda \times 0.6) \times$ 0.044 = 0.83 fF

Drain capacitance/cell: $((3\lambda \times 4\lambda)\ (0.6)^2 \times 0.3 + 2 \times 3\lambda \times 0.6 \times 0.8) \times 0.375 +$ $4\lambda \times 0.6 \times 0.43$ = 2.6 fF

Equivalent Transient Model for MOS NAND ROM

Model for NAND ROM



Word line parasitics:

Resistance/cell: (6/2) x 10 Ω/q = 30 Ω

Wire capacitance/cell: $(6\lambda \times 2\lambda)\,(0.6)^2$ 0.058 + 2 × (6λ × 0.6) × 0.043 = 0.56 fF

Gate Capacitance/cell: $(3\lambda \times 2\lambda)\,(0.6)^2$ 1.76 = 3.8 fF.

Bit line parasitics:

Resistance/cell: ~ 10 kΩ, the average transistor resistance over the range of interest.

Wire capacitance/cell: Included in diffusion capacitance

Source/Drain capacitance/cell: $((3\lambda \times 3\lambda)\,(0.6)^2 \times 0.3 + 2 \times 3\lambda \times 0.6 \times 0.8) \times 0.375 + (3\lambda \times 2\lambda)\,(0.6)^2 \times 1.76 = 5.2$ fF

Propagation Delay of NOR ROM

Word line delay

Consider the 512×512 case. The delay of the distributed *rc*-line containing *M* cells can be approximated using the expressions derived in Chapter 8.

$$t_{word} = 0.38\,(r_{word} \times c_{word})\,M^2 = 0.38\,(35\ \Omega \times (0.65 + 5.1)\ \text{fF})\ 512^2 = 20\ \text{nsec}$$

Bit line delay

Assume a (2.4/1.2) pull-down device and a (8/1.2) pull-up transistor. The bit line switches between 5 V and 2.5 V.

$$C_{bit} = 512 \times (2.6 + 0.8)\ \text{fF} = 1.7\ \text{pF}$$

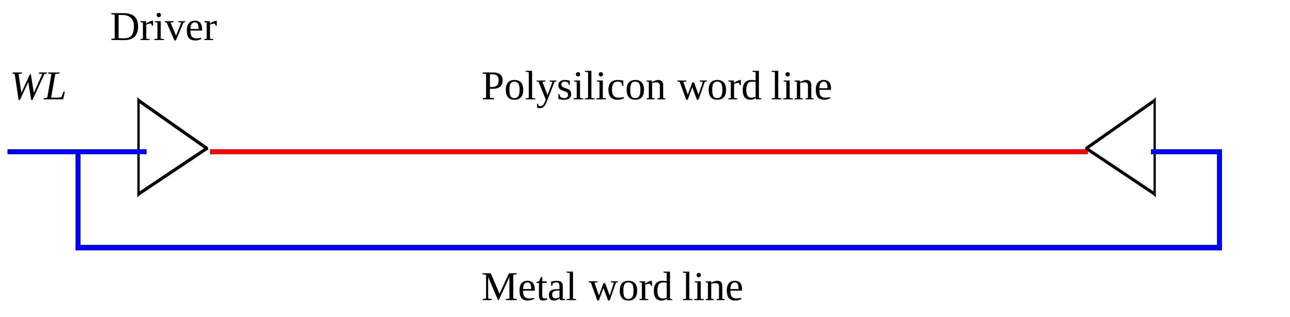
$$I_{avHL} = 1/2\ (2.4/0.9)\ (19.6\ 10^{-6})((4.25)^2/2 + (4.25 \times 3.75 - (3.75)^2/2)) -$$
$$1/2\ (8/0.9)\ (5.3\ 10^{-6})\ (4.25 \times 1.25 - (1.25)^2/2) = 0.36\ \text{mA}$$

$$t_{HL} = (1.7\ \text{pF} \times 1.25\ \text{V}) / 0.36\ \text{mA} = 5.9\ \text{nsec}$$

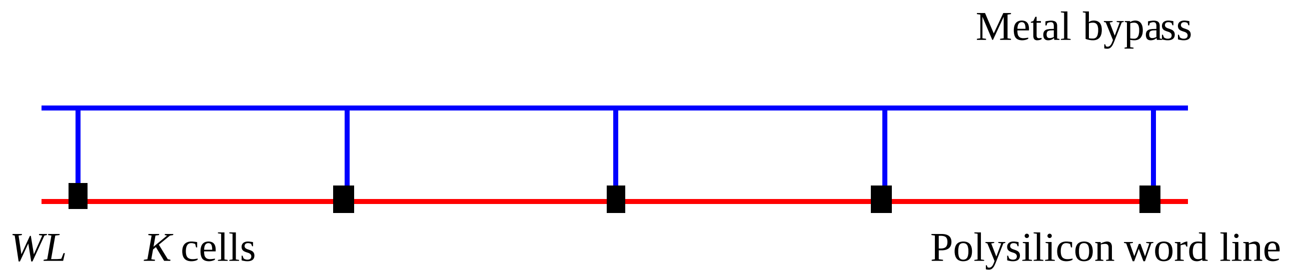
The low-to-high response time can be computed using a similar approach.

$$t_{LH} = (1.7\ \text{pF} \times 1.25\ \text{V}) / 0.36\ \text{mA} = 5.9\ \text{nsec}$$

Decreasing Word Line Delay



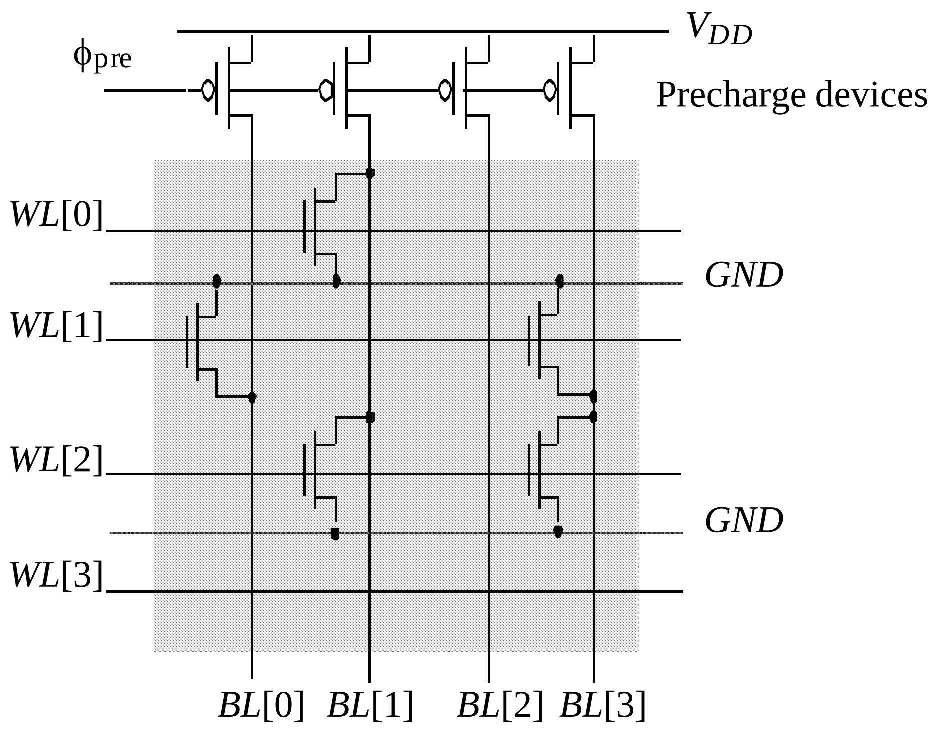
(a) Driving the word line from both sides



(b) Using a metal bypass

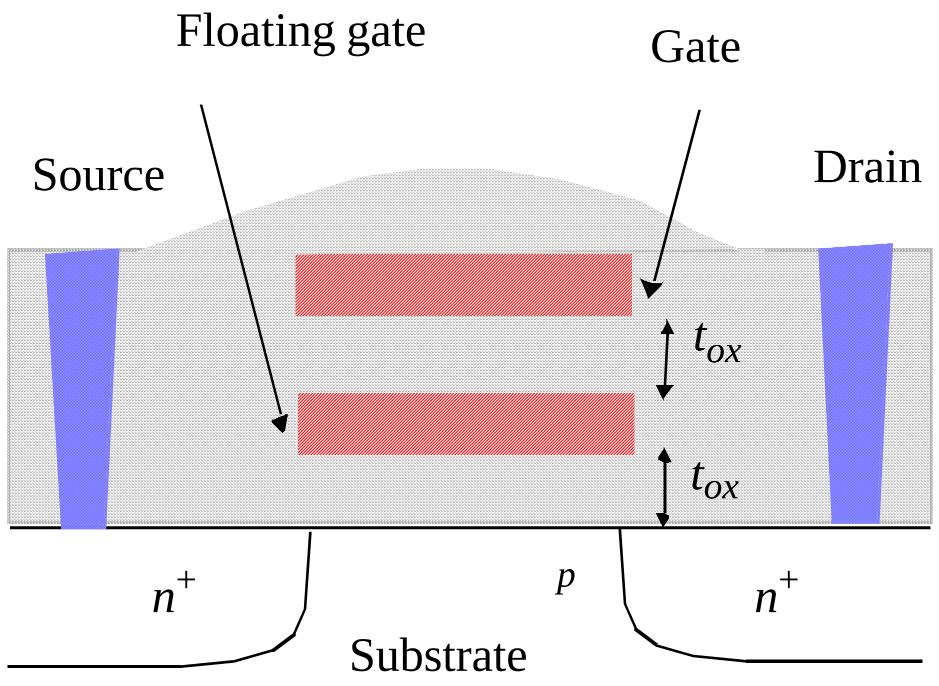
(c) Use silicides

Precharged MOS NOR ROM

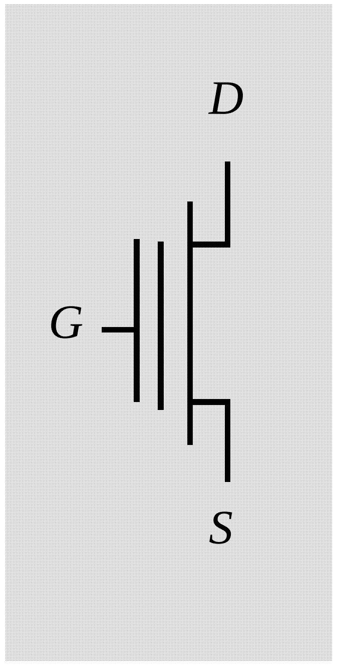


PMOS precharge device can be made as large as necessary, but clock driver becomes harder to design.

Floating-gate transistor (FAMOS)

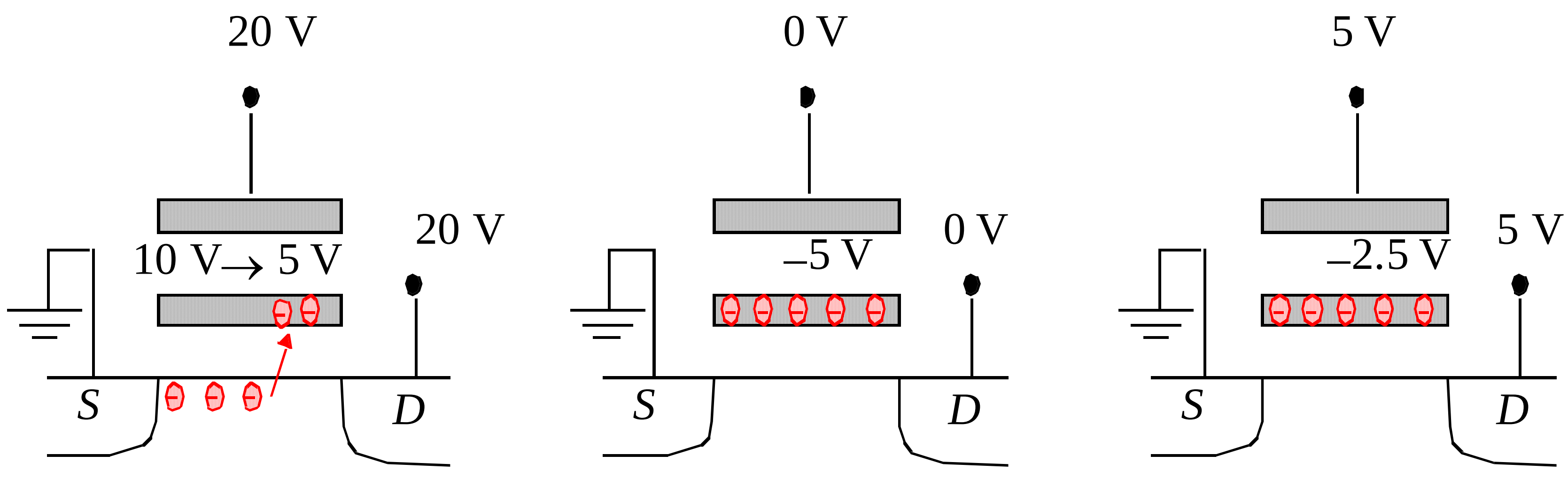


(a) Device cross-section



(b) Schematic symbol

Floating-Gate Transistor Programming

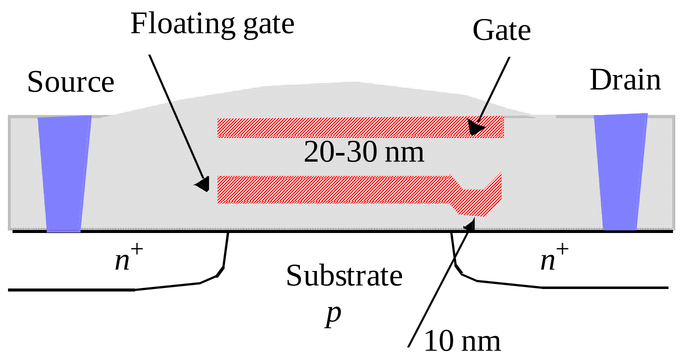


Avalanche injection.

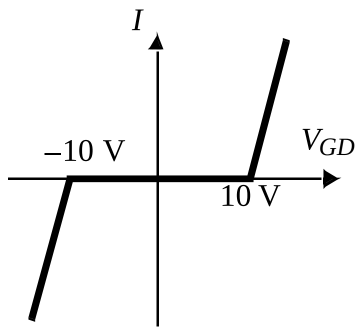
Removing programming voltage leaves charge trapped.

Programming results in higher V_T.

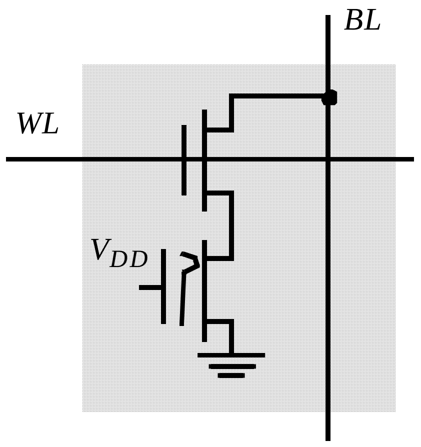
FLOTOX EEPROM



(a) Flotox transistor

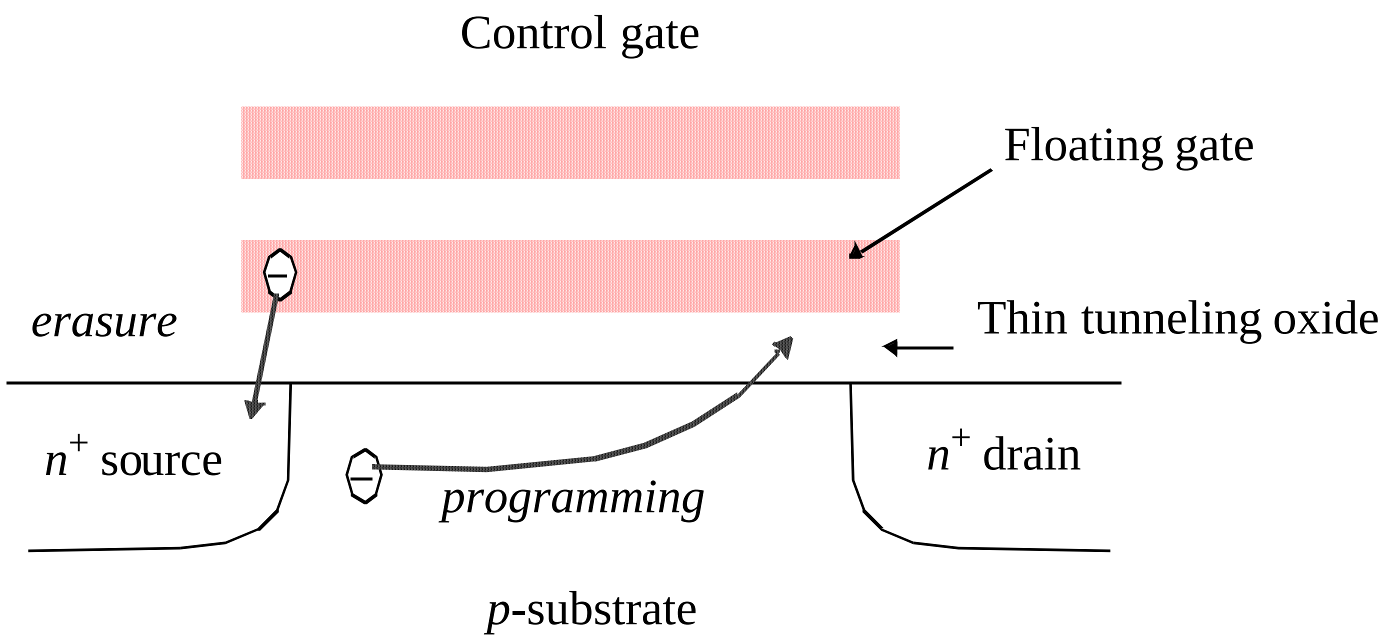


(b) Fowler-Nordheim *I-V* characteristic

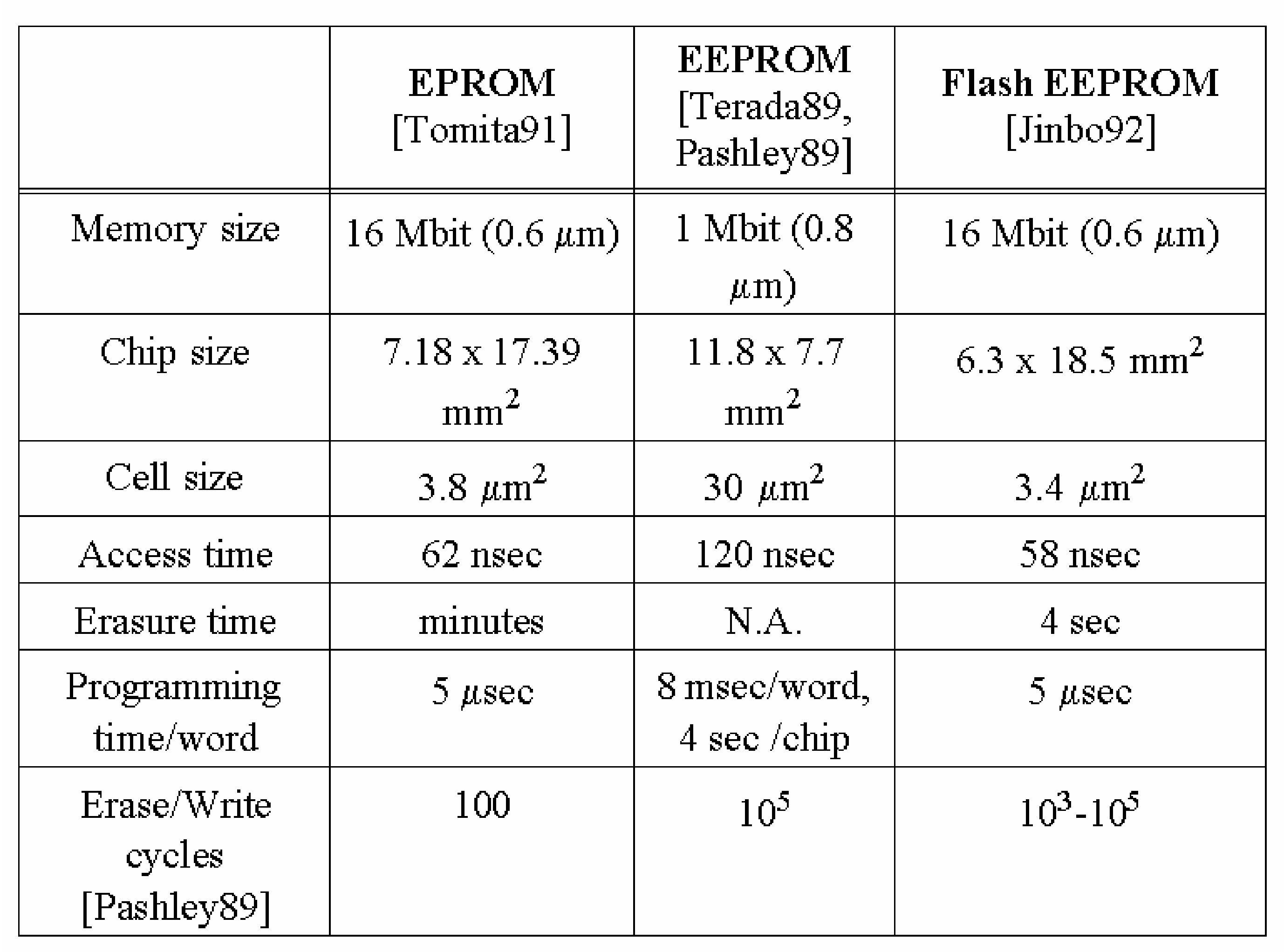


(c) EEPROM cell during a read operation

Flash EEPROM



Characteristics of State-of-the-art NVM



	EPROM [Tomita91]	**EEPROM** [Terada89, Pashley89]	**Flash EEPROM** [Jinbo92]
Memory size	16 Mbit (0.6 μm)	1 Mbit (0.8 μm)	16 Mbit (0.6 μm)
Chip size	7.18 x 17.39 mm^2	11.8 x 7.7 mm^2	6.3 x 18.5 mm^2
Cell size	3.8 μm^2	30 μm^2	3.4 μm^2
Access time	62 nsec	120 nsec	58 nsec
Erasure time	minutes	N.A.	4 sec
Programming time/word	5 μsec	8 msec/word, 4 sec /chip	5 μsec
Erase/Write cycles [Pashley89]	100	10^5	10^3-10^5

Read-Write Memories (RAM)

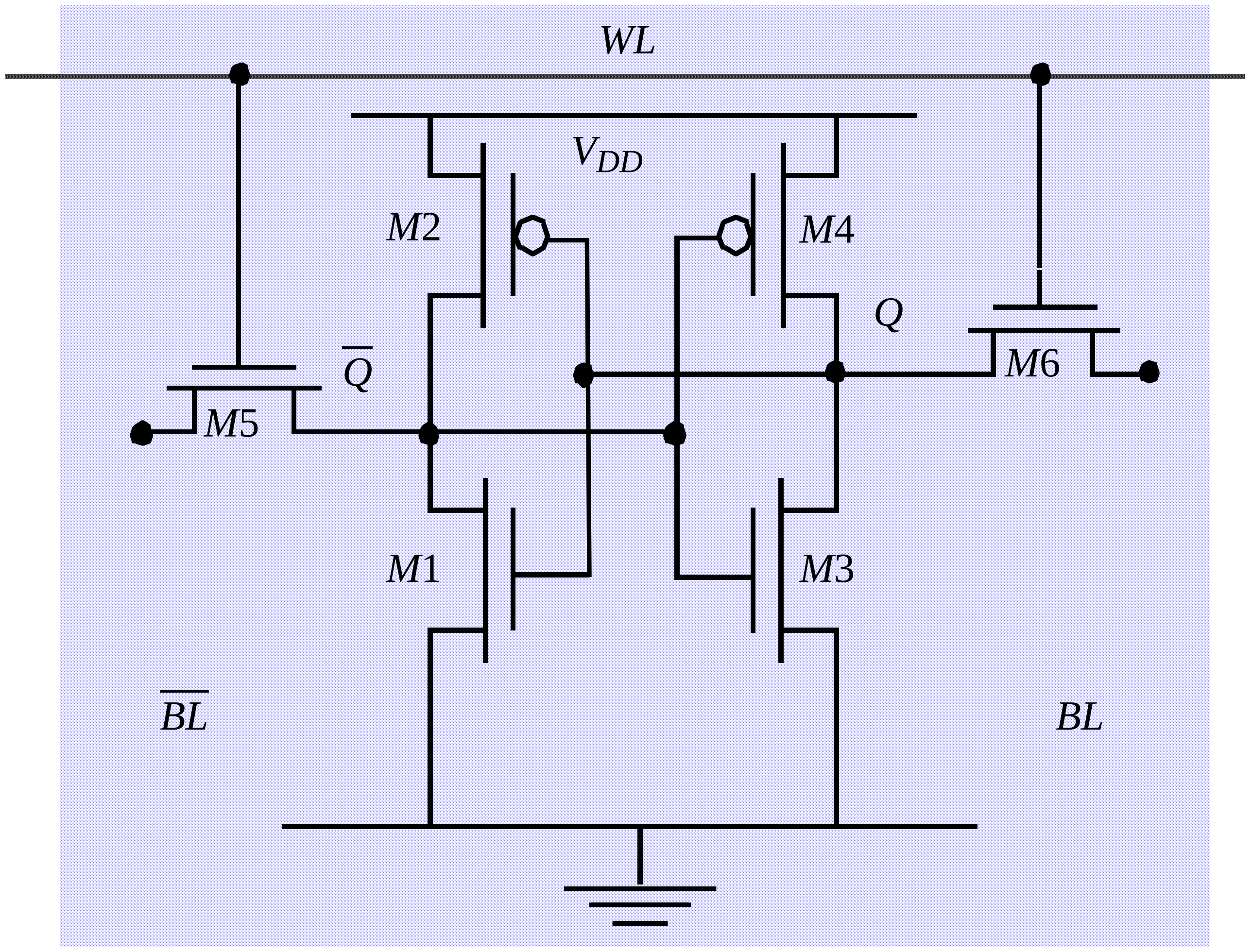
- **STATIC (SRAM)**

  Data stored as long as supply is applied
  Large (6 transistors/cell)
  Fast
  Differential

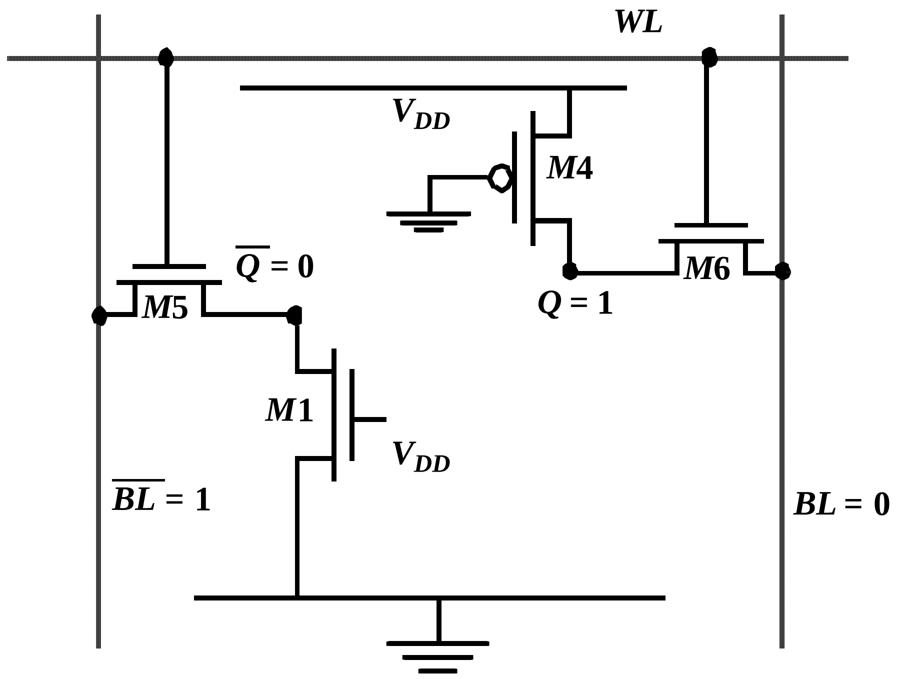
- **DYNAMIC (DRAM)**

  Periodic refresh required
  Small (1-3 transistors/cell)
  Slower
  Single Ended

6-transistor CMOS SRAM Cell



CMOS SRAM Analysis (Write)



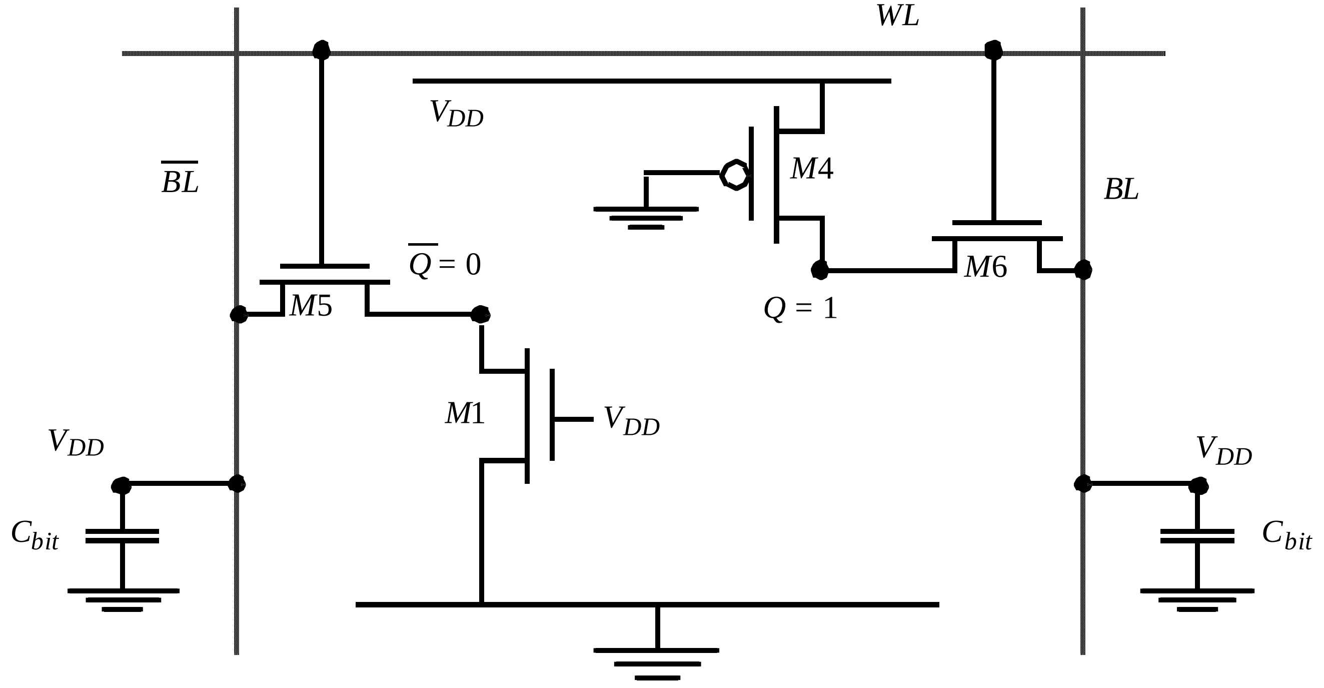
$$k_{n,\,M6}\left((V_{DD} - V_{Tn})\frac{V_{DD}}{2} - \frac{V_{DD}^2}{8}\right) = k_{p,\,M4}\left((V_{DD} - |V_{Tp}|)\frac{V_{DD}}{2} - \frac{V_{DD}^2}{8}\right)$$

$$\boxed{(W/L)_{n,M6} \geq 0.33\,(W/L)_{p,M4}}$$

$$\frac{k_{n,\,M5}}{2}\left(\frac{V_{DD}}{2} - V_{Tn}\left(\frac{V_{DD}}{2}\right)\right)^2 = k_{n,\,M1}\left((V_{DD} - |V_{Tn}|)\frac{V_{DD}}{2} - \frac{V_{DD}^2}{8}\right)$$

$$(W/L)_{n,M5} \geq 10\,(W/L)_{n,M1}$$

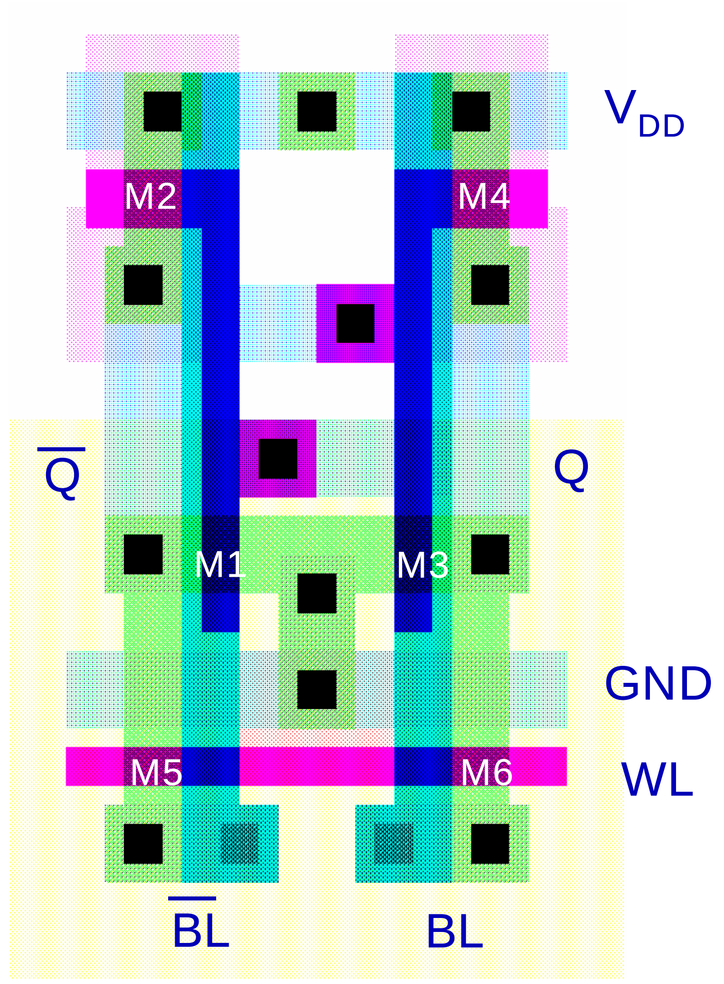
CMOS SRAM Analysis (Read)



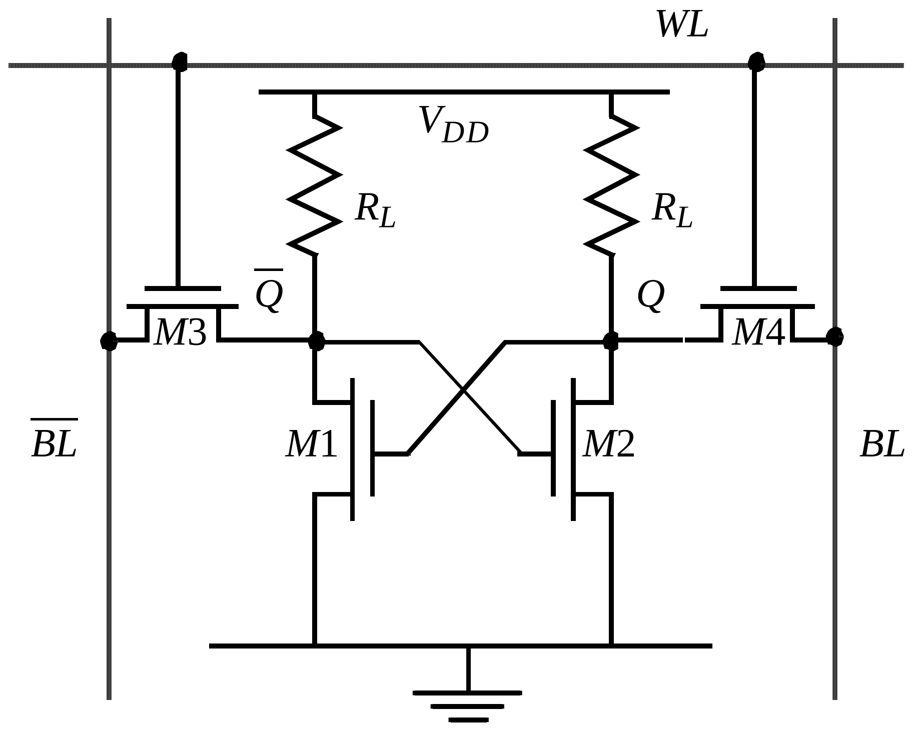
$$\frac{k_{n,\,M5}}{2}\left(\frac{V_{DD}}{2} - V_{Tn}\left(\frac{V_{DD}}{2}\right)\right)^2 = k_{n,\,M1}\left((V_{DD} - |V_{Tn}|)\frac{V_{DD}}{2} - \frac{V_{DD}^2}{8}\right)$$

$(W/L)_{n,M5} \leq 10\,(W/L)_{n,M1}$ **(supercedes read constraint)**

6T-SRAM — Layout

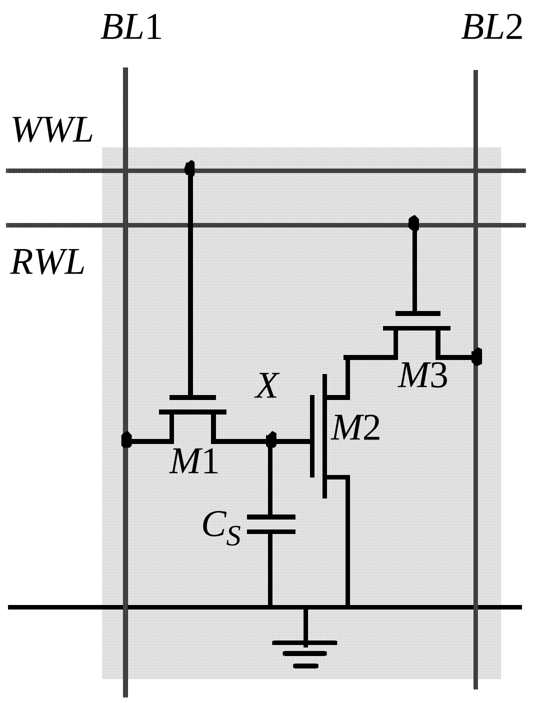


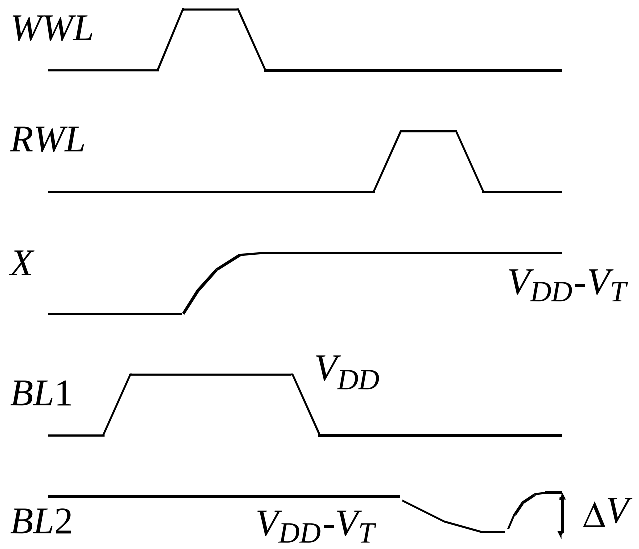
Resistance-load SRAM Cell



Static power dissipation -- Want R_L large
Bit lines precharged to V_{DD} to address t_p problem

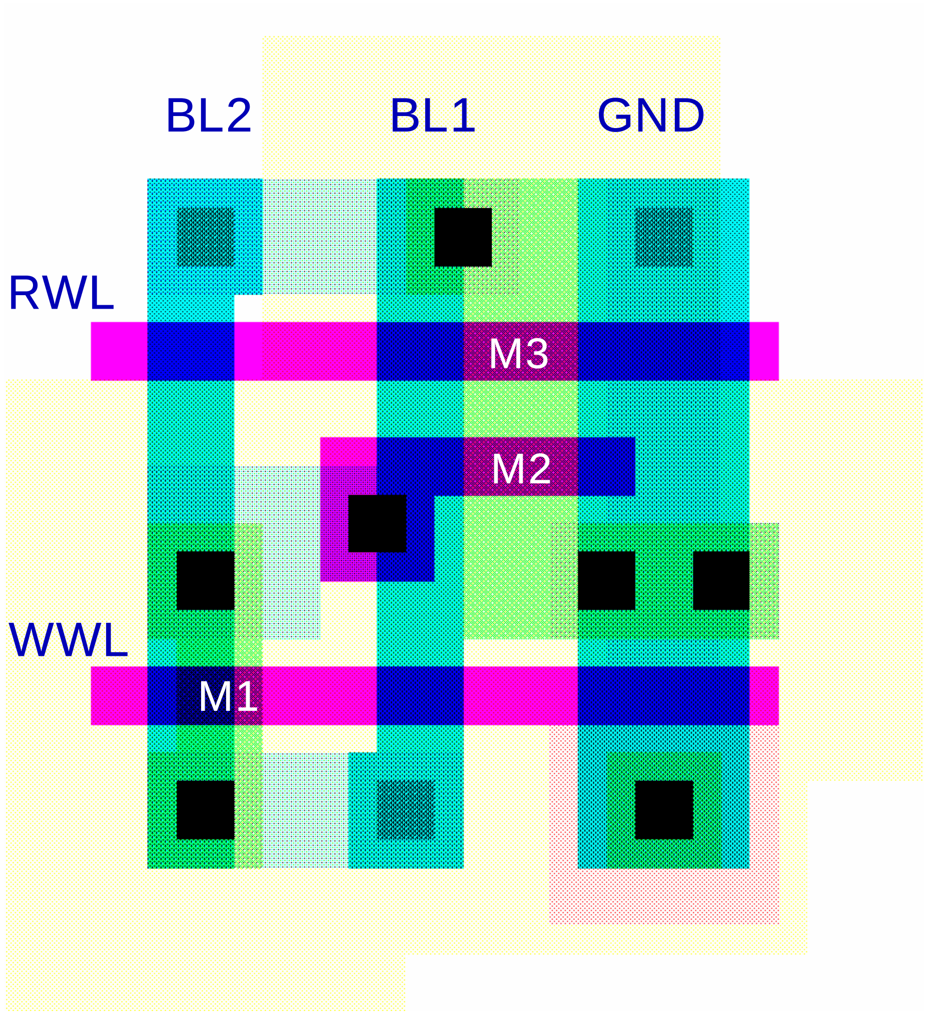
3-Transistor DRAM Cell



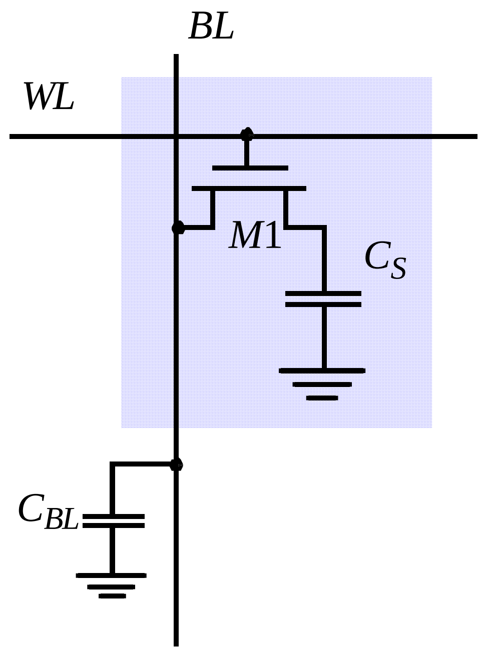


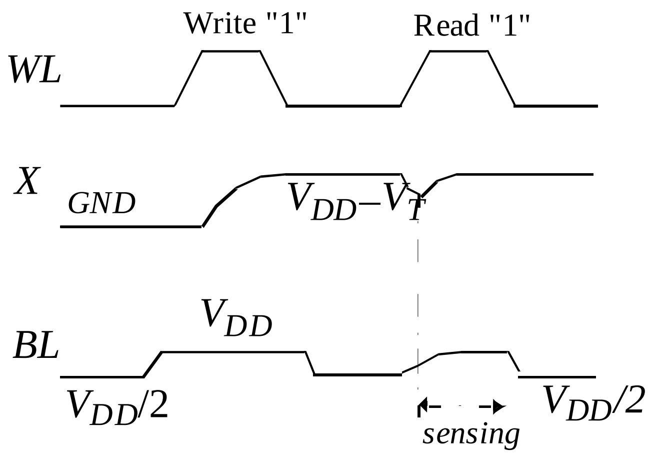
No constraints on device ratios
Reads are non-destructive
Value stored at node X when writing a "1" = V_{WWL}-V_{Tn}

3T-DRAM — Layout



1-Transistor DRAM Cell





Write: C_S is charged or discharged by asserting WL and BL.
Read: Charge redistribution takes places between bit line and storage capacitance

$$\Delta V = V_{BL} - V_{PRE} = (V_{BIT} - V_{PRE})\frac{C_S}{C_S + C_{BL}}$$

Voltage swing is small; typically around 250 mV.

DRAM Cell Observations

1T DRAM requires a sense amplifier for each bit line, due to charge redistribution read-out.

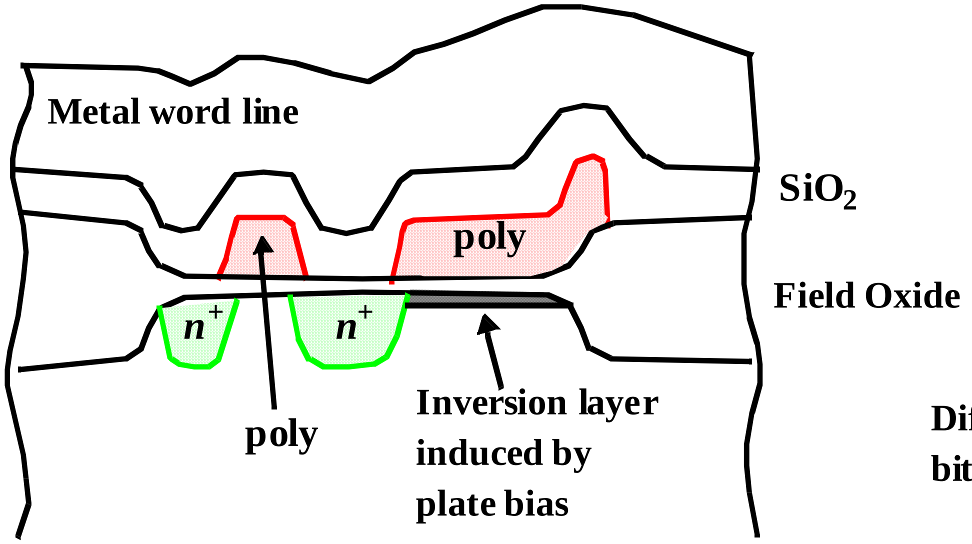
DRAM memory cells are single ended in contrast to SRAM cells.

The read-out of the 1T DRAM cell is destructive; read and refresh operations are necessary for correct operation.

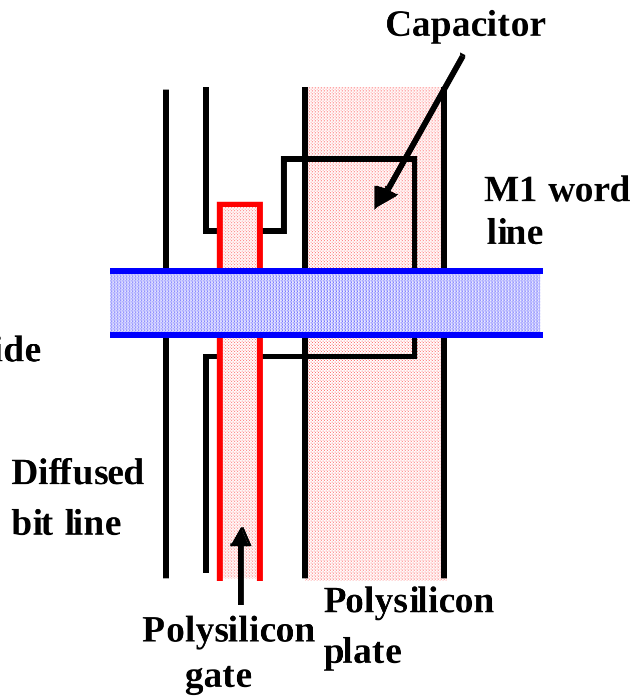
Unlike 3T cell, 1T cell requires presence of an extra capacitance that must be explicitly included in the design.

When writing a "1" into a DRAM cell, a threshold voltage is lost. This charge loss can be circumvented by bootstrapping the word lines to a higher value than V_{DD}.

1-T DRAM Cell



(a) Cross-section



(b) Layout

Used Polysilicon-Diffusion Capacitance

Expensive in Area

Periphery

- **Decoders**
- **Sense Amplifiers**
- **Input/Output Buffers**
- **Control / Timing Circuitry**

Row Decoders

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

(N)AND Decoder

$$WL_0 = A_0A_1A_2A_3A_4A_5A_6A_7A_8A_9$$

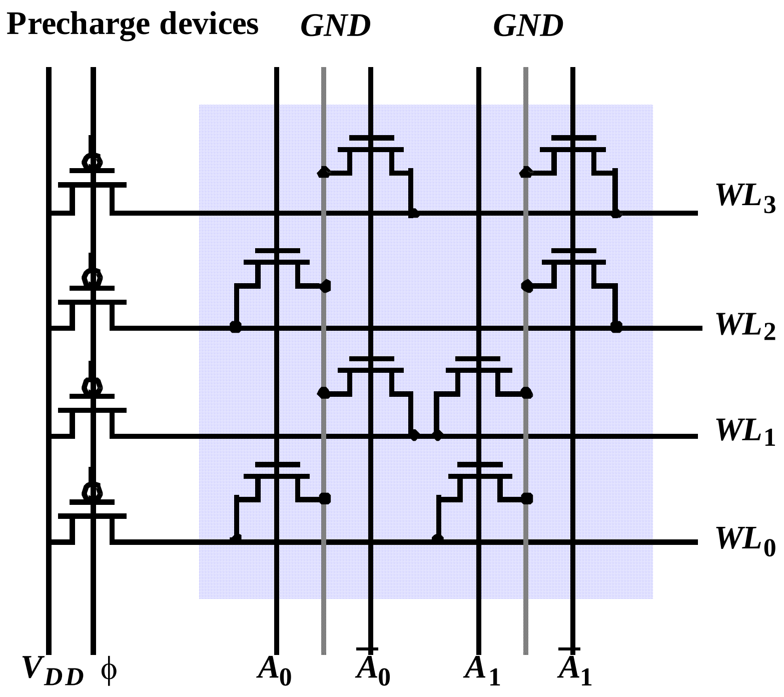
$$WL_{511} = \bar{A}_0A_1A_2A_3A_4A_5A_6A_7A_8A_9$$

NOR Decoder

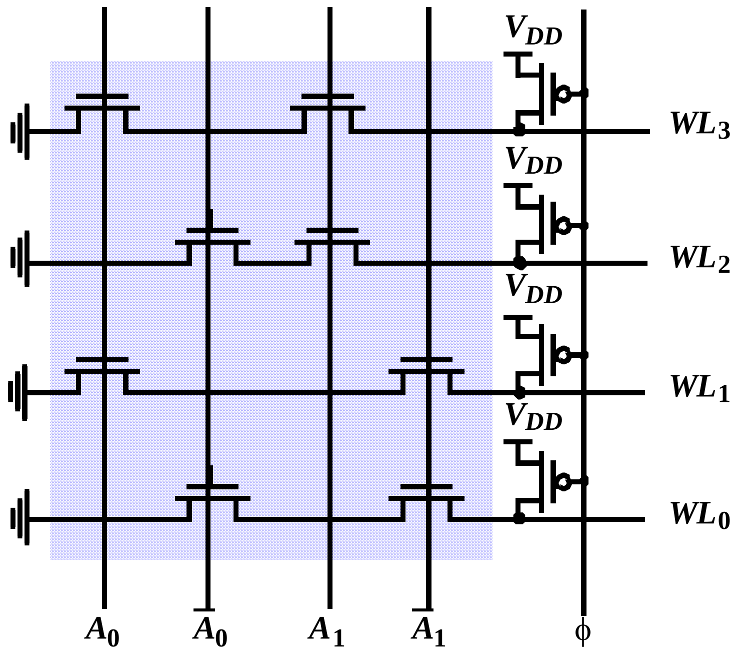
$$WL_0 = \overline{A_0+A_1+A_2+A_3+A_4+A_5+A_6+A_7+A_8+A_9}$$

$$WL_{511} = \overline{A_0+\bar{A}_1+\bar{A}_2+\bar{A}_3+\bar{A}_4+\bar{A}_5+\bar{A}_6+\bar{A}_7+\bar{A}_8+\bar{A}_9}$$

Dynamic Decoders



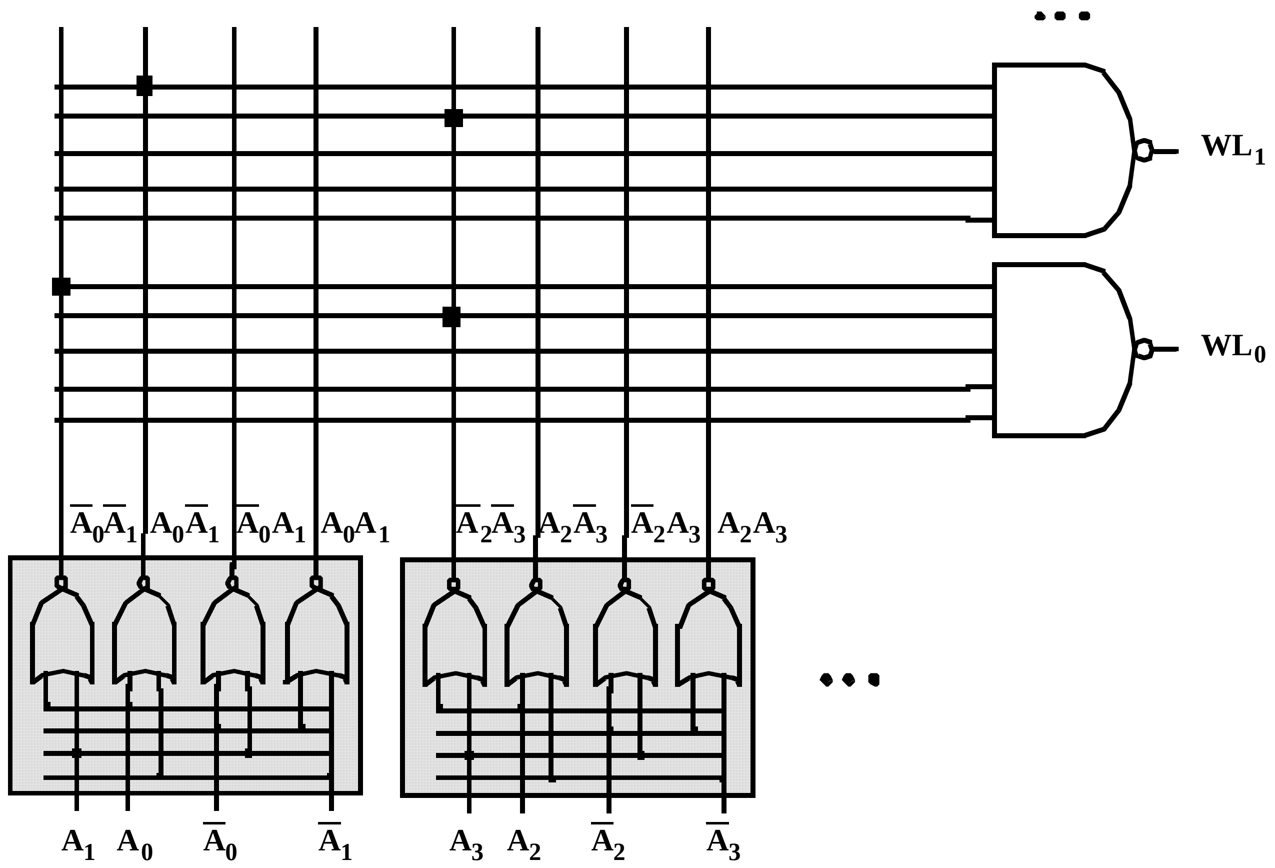
Dynamic 2-to-4 NOR decoder



2-to-4 MOS dynamic NAND Decoder

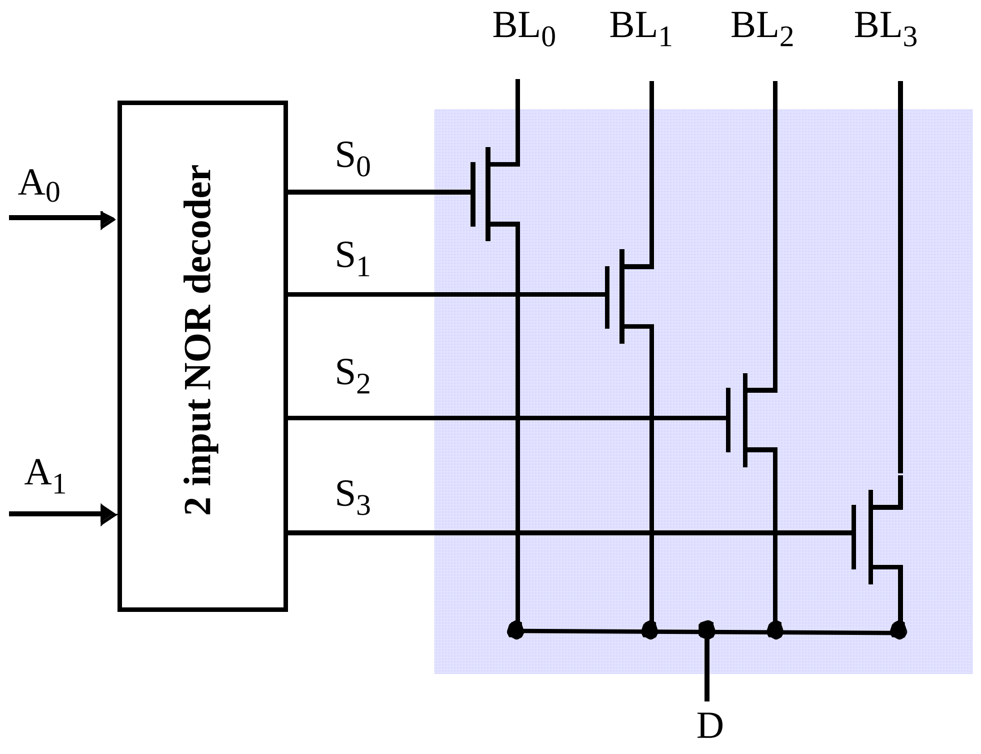
Propagation delay is primary concern

A NAND decoder using 2-input pre-decoders



Splitting decoder into two or more logic layers produces a faster and cheaper implementation

4 input pass-transistor based column decoder

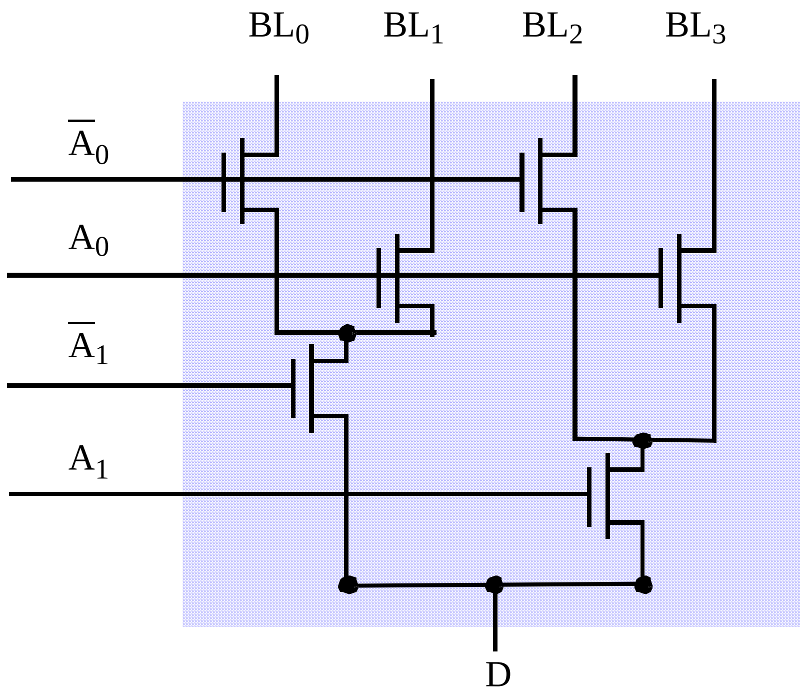


lvantage: speed (t_{pd} does not add to overall memory access time)

only 1 extra transistor in signal path

sadvantage: large transistor count

4-to-1 tree based column decoder

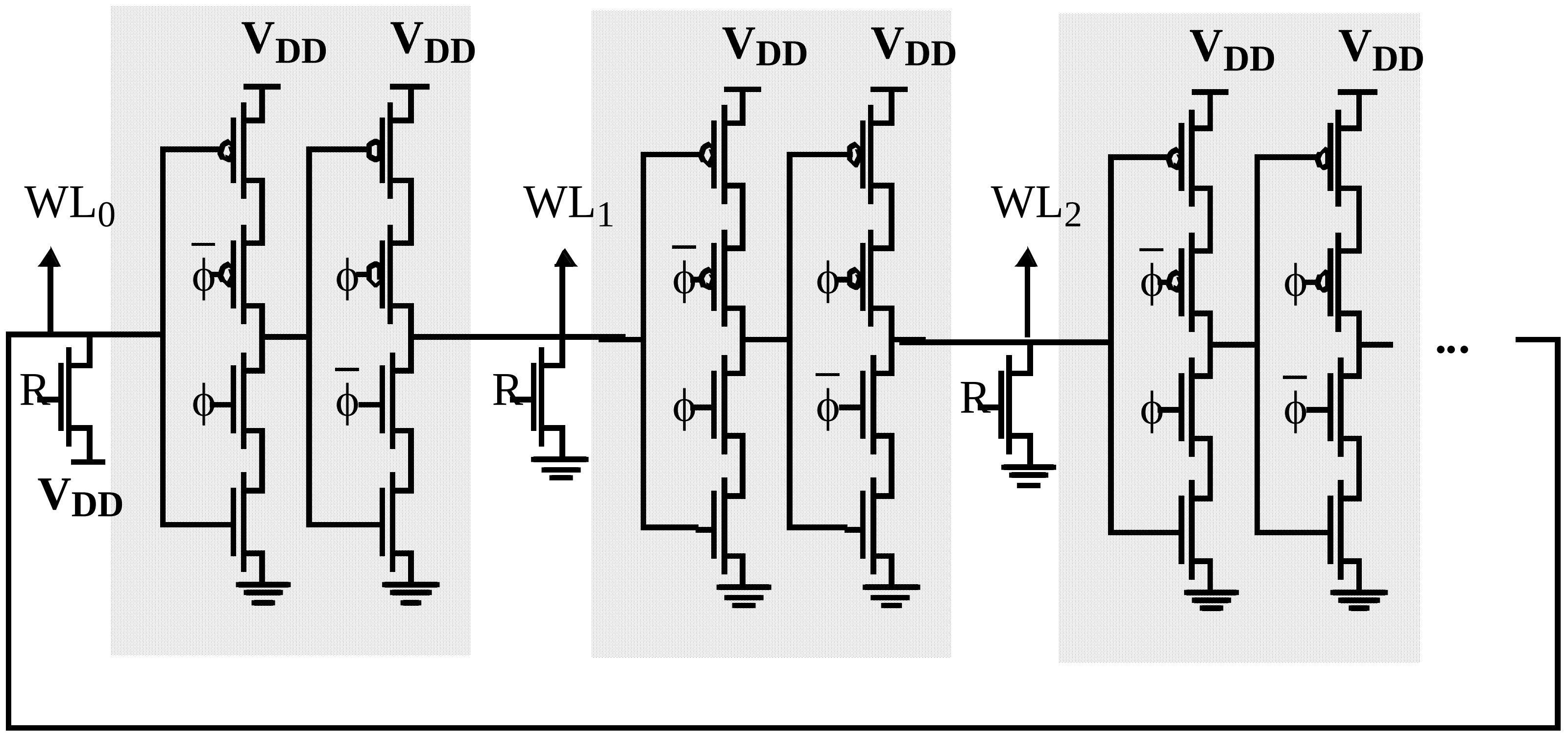


Number of devices drastically reduced

Delay increases quadratically with # of sections; prohibitive for large decoders

Solutions: **buffers**
progressive sizing
combination of tree and pass transistor approaches

Decoder for circular shift-register



Sense Amplifiers

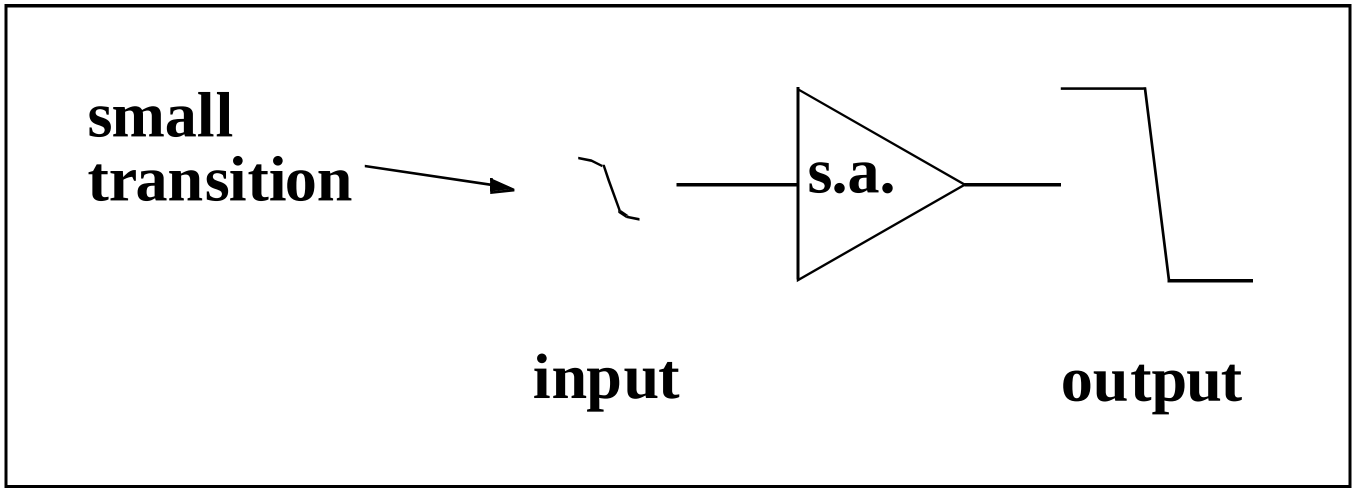
$$t_p = \frac{C \cdot \Delta V}{I_{av}}$$

make ΔV as small as possible

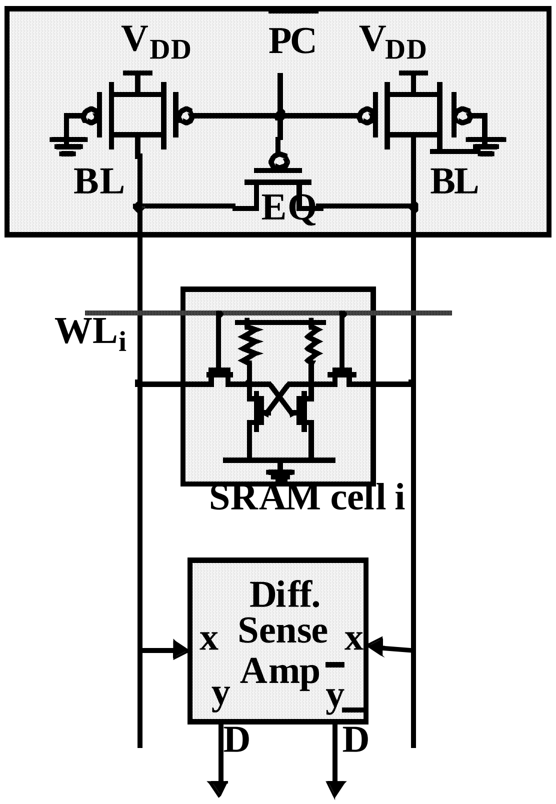
large

small

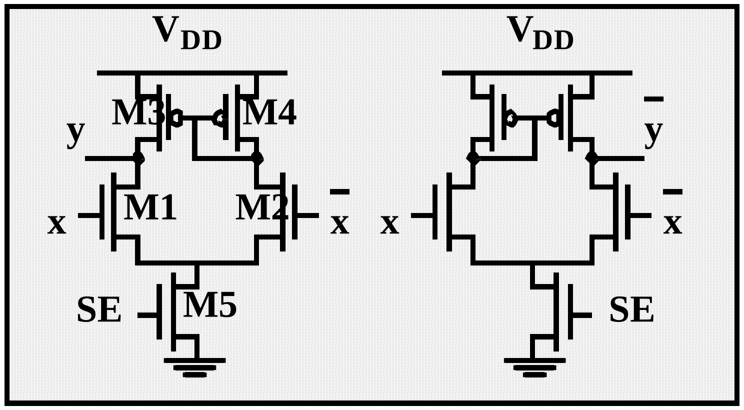
Idea: Use Sense Amplifer



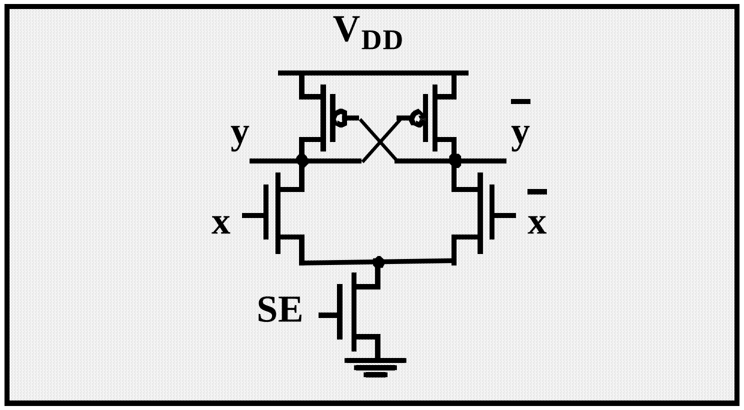
Differential Sensing - SRAM



(a) SRAM sensing scheme.

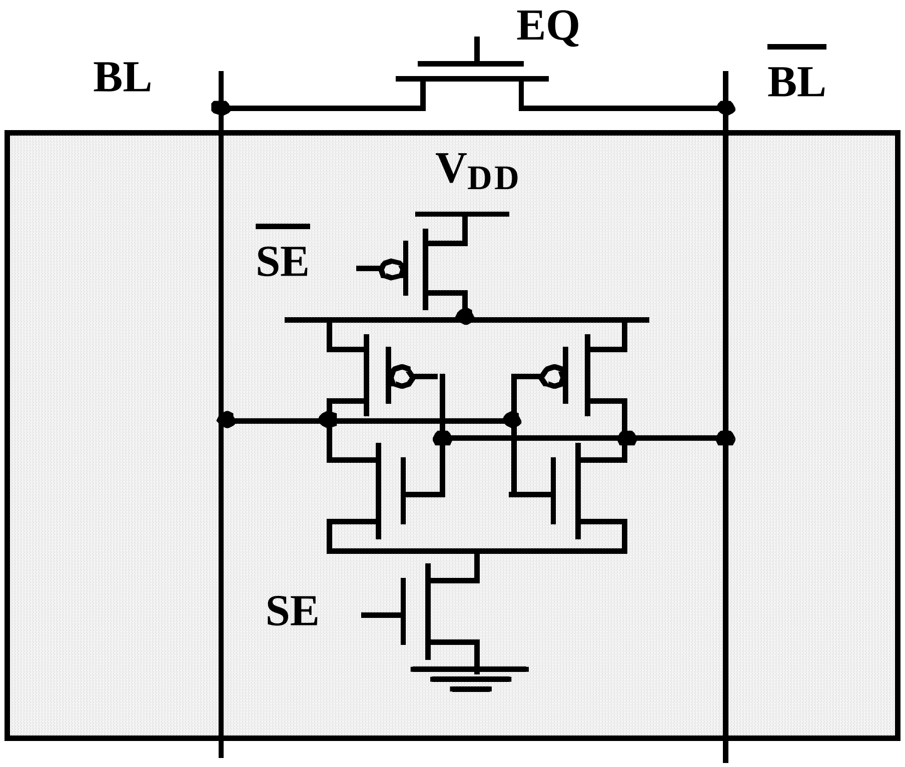


(b) Doubled-ended Current Mirror Amplifier



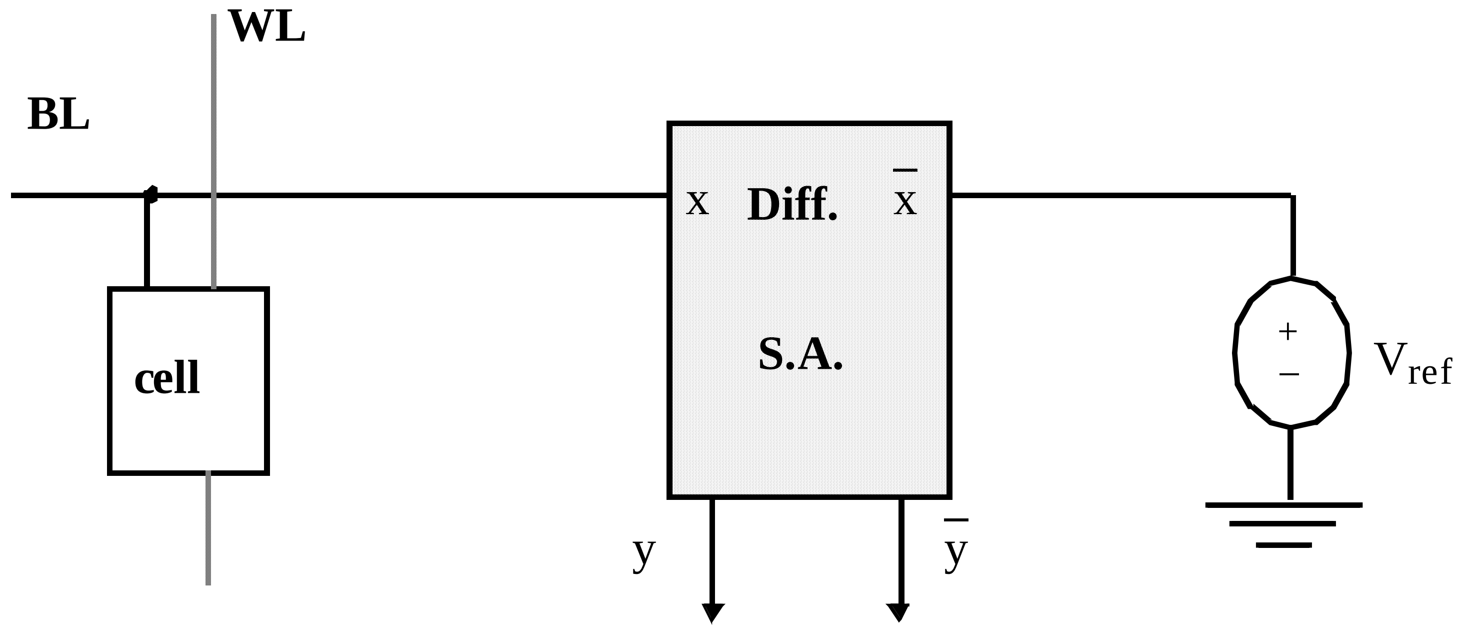
(c) Cross-Coupled Amplifier

Latch-Based Sense Amplifier



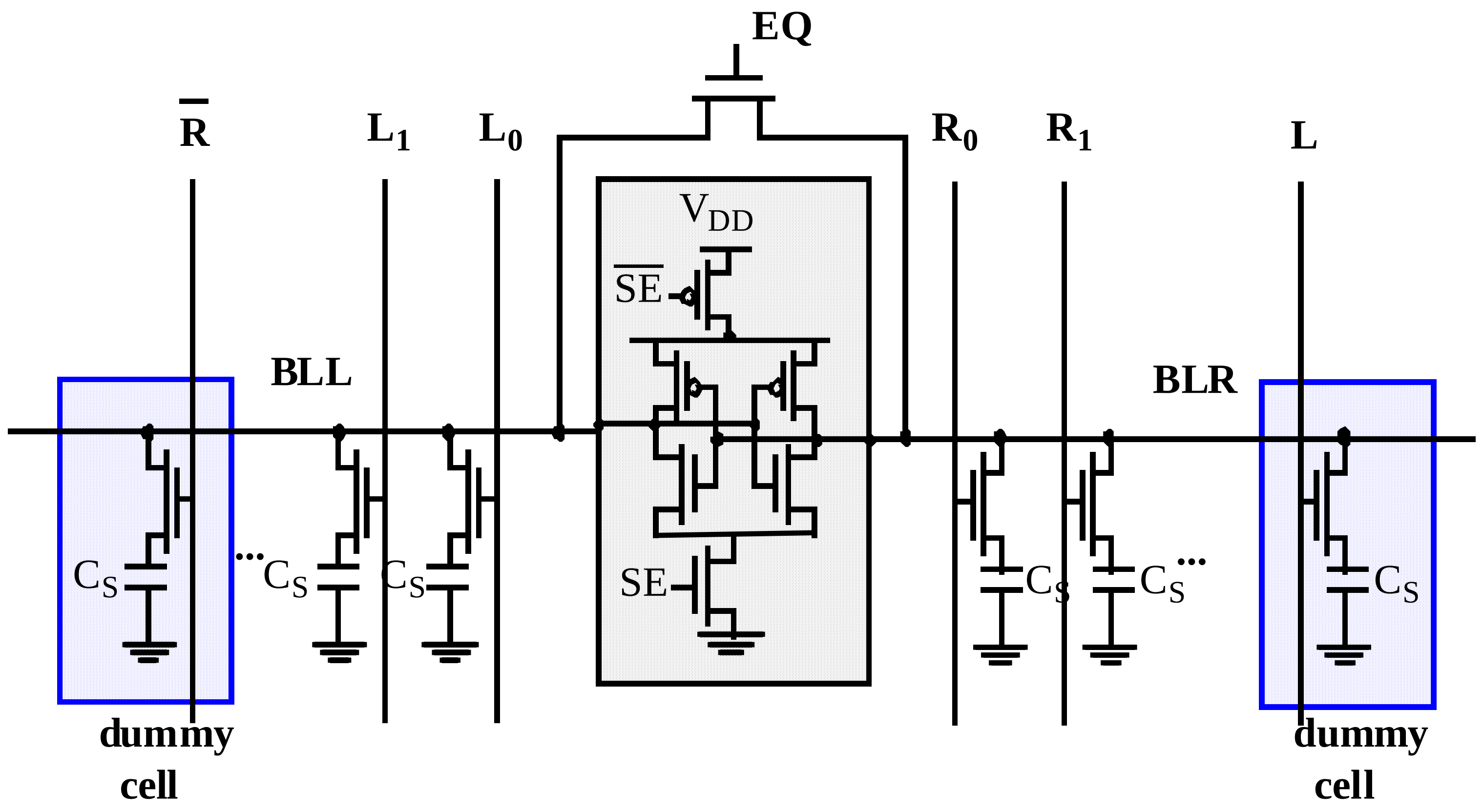
Initialized in its meta-stable point with EQ
Once adequate voltage gap created, sense amp enabled with SE
Positive feedback quickly forces output to a stable operating point.

Single-to-Differential Conversion

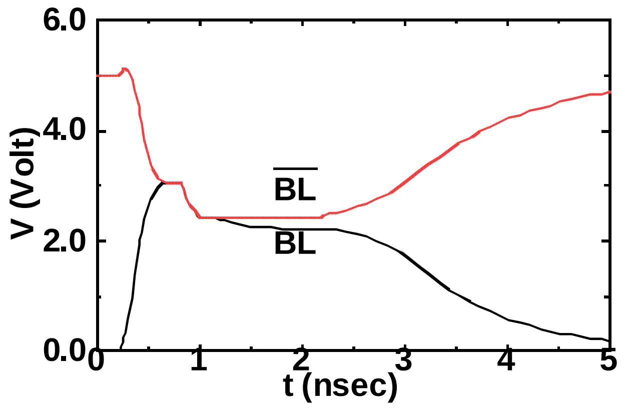


How to make good V_{ref}?

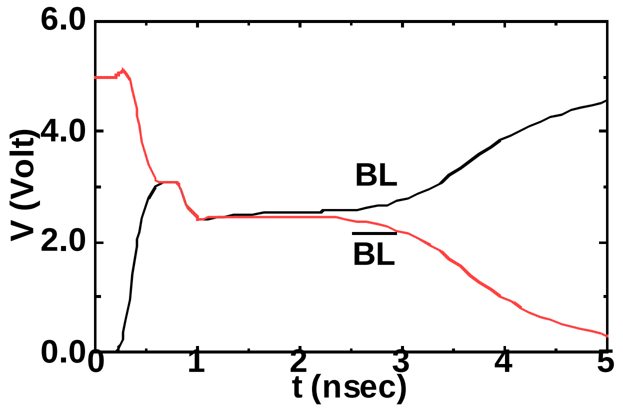
Open bitline architecture



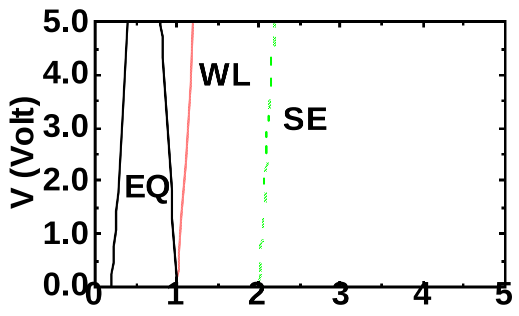
DRAM Read Process with Dummy Cell



(a) reading a zero

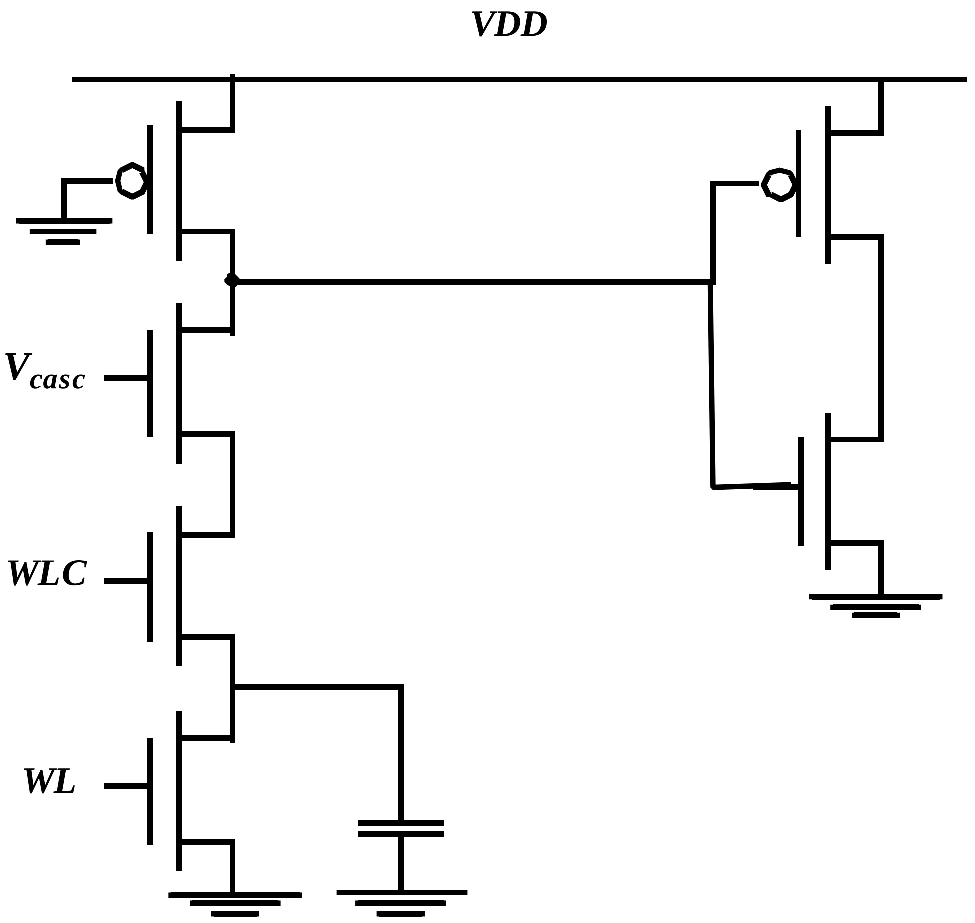


(b) reading a one

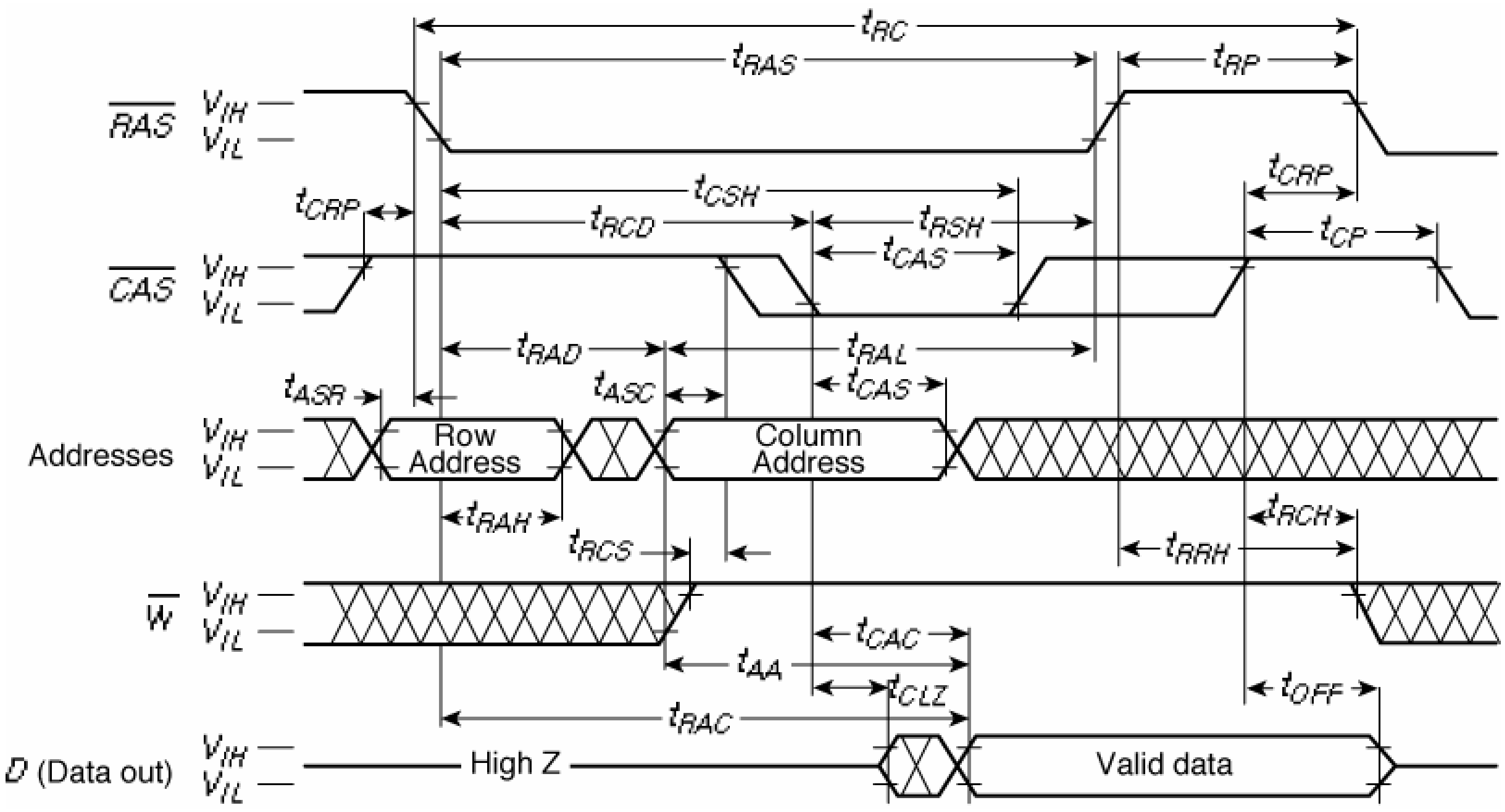


(c) control signals

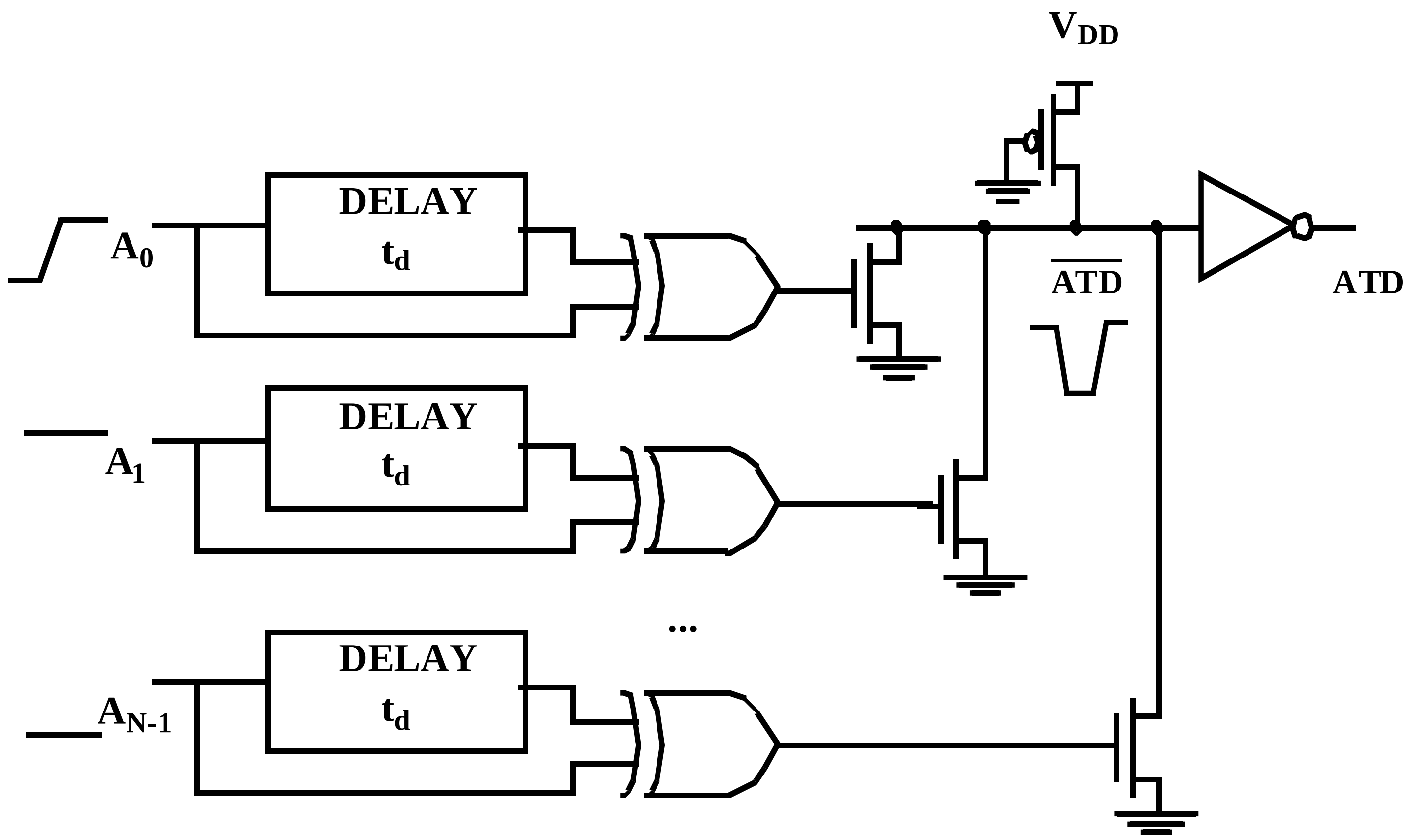
Single-Ended Cascode Amplifier



DRAM Timing


t_{RC}
t_{RAS}
t_{RP}
$\overline{RAS}$
V_{IH}
V_{IL}
t_{CRP}
t_{CSH}
t_{RCD}
t_{RSH}
t_{CAS}
t_{CP}
$\overline{CAS}$
t_{RAD}
t_{RAL}
t_{ASR}
t_{ASC}
Addresses
Row Address
Column Address
t_{RAH}
t_{RCS}
t_{RCH}
t_{RRH}
$\overline{W}$
t_{CAC}
t_{AA}
t_{CLZ}
t_{OFF}
t_{RAC}
D (Data out)
High Z
Valid data

Address Transition Detection



Reliability and Yield

- Semiconductor memories trade off noise-margin for density and performance

Highly Sensitive to Noise (Crosstalk, Supply Noise)

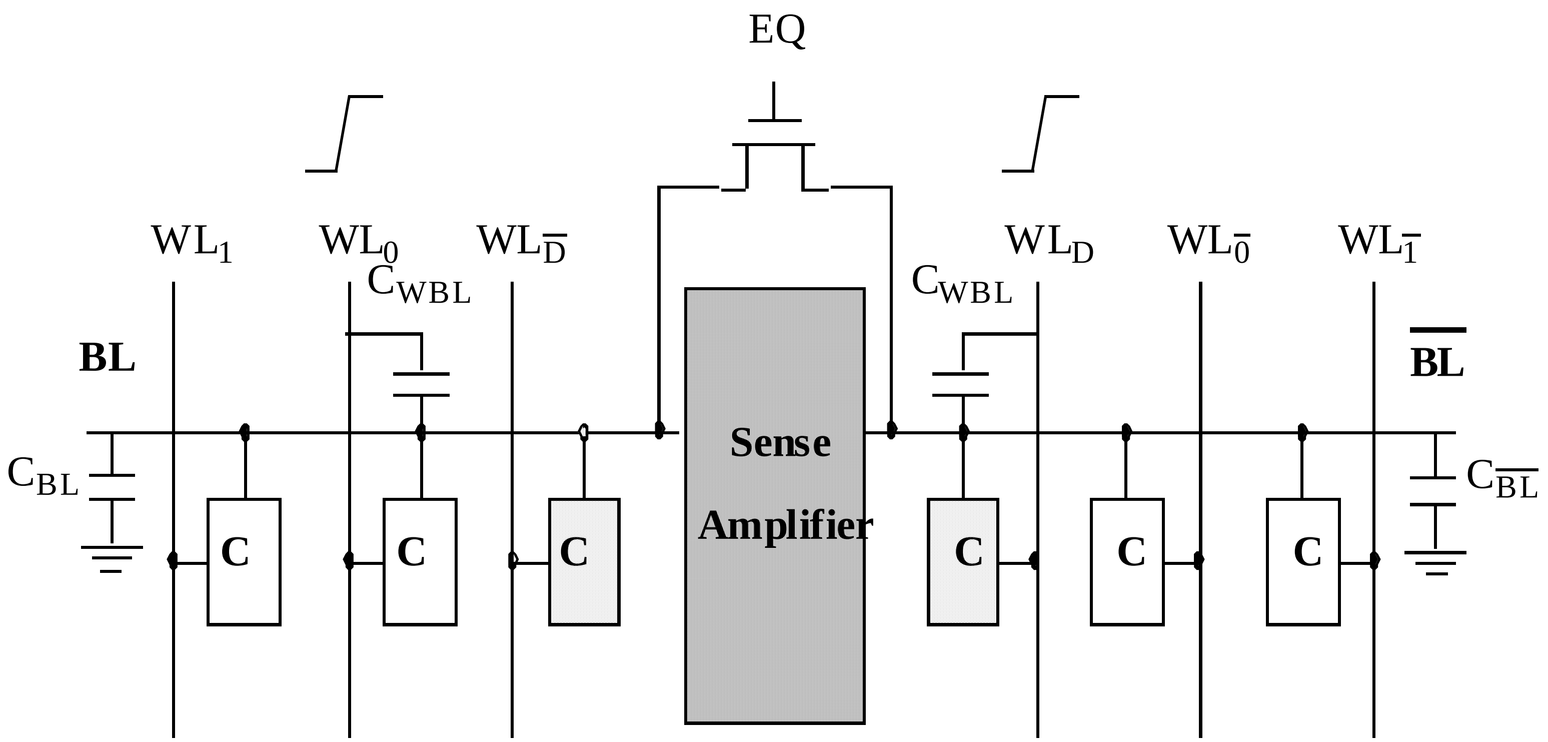
- High Density and Large Die size cause Yield Problems

$$Y = 100\frac{Number''\ of''\ ''Good''\ Chips''\ on''\ Wafer}{Number''\ of''\ Chips''\ on''\ Wafer}$$

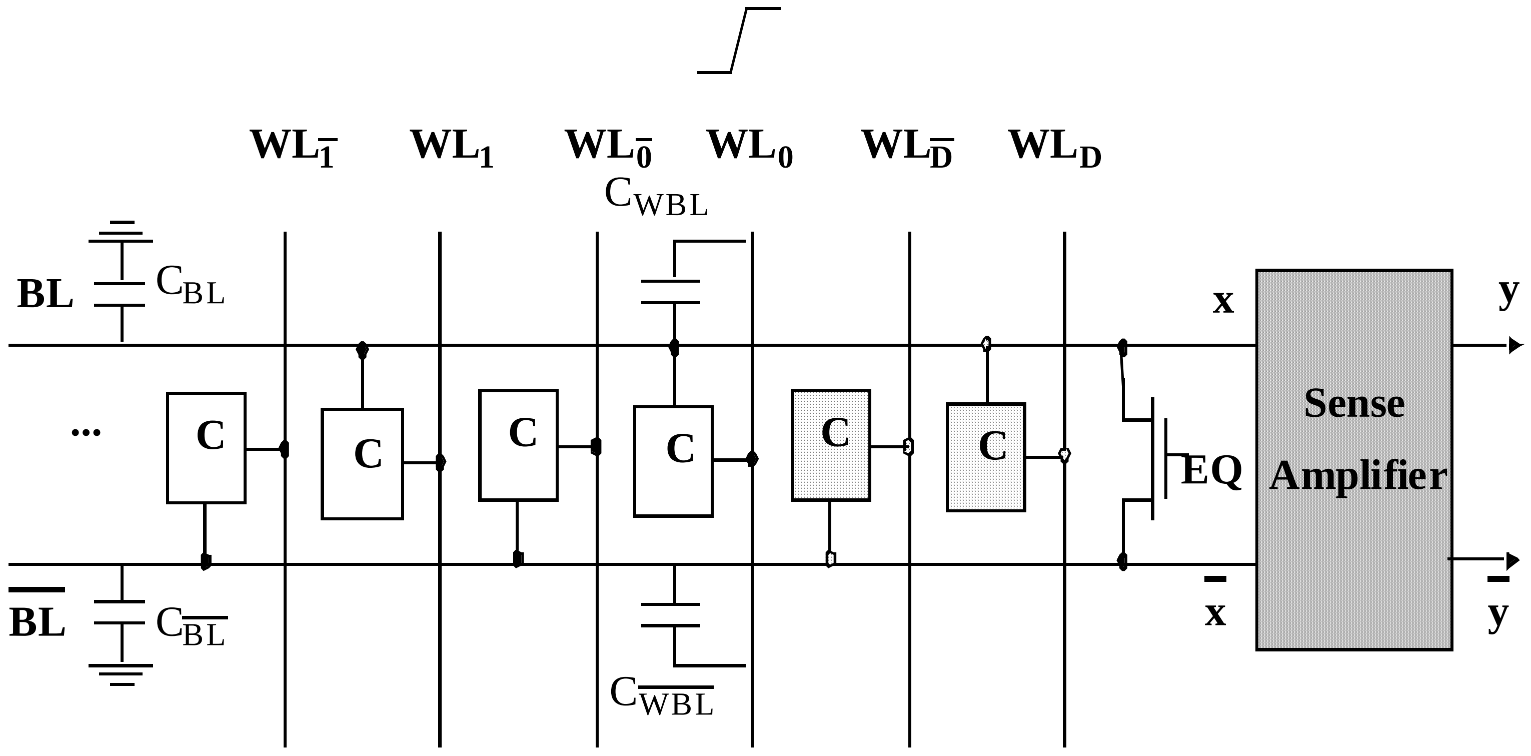
$$Y = \left[\frac{1 - e^{-AD}}{AD}\right]^2$$

Increase Yield using Error Correction and Redundancy

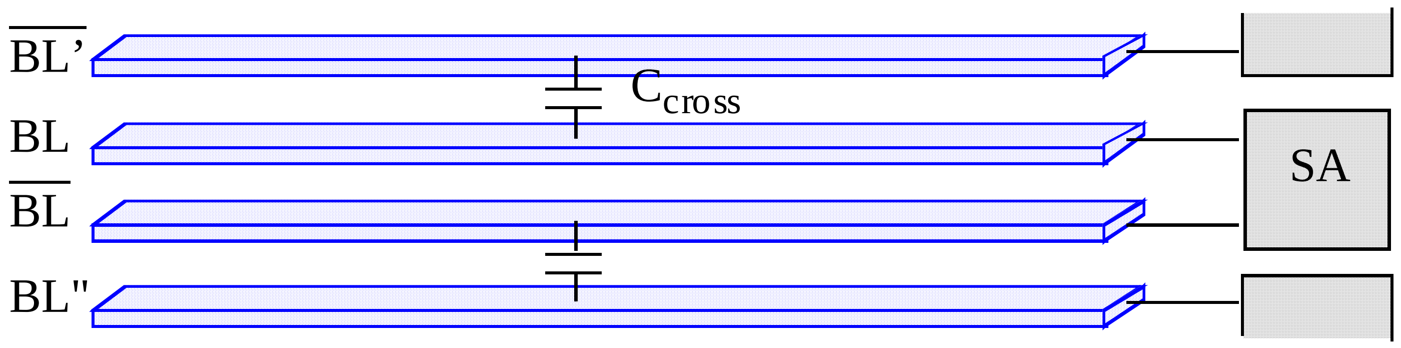
Open Bit-line Architecture —Cross Coupling



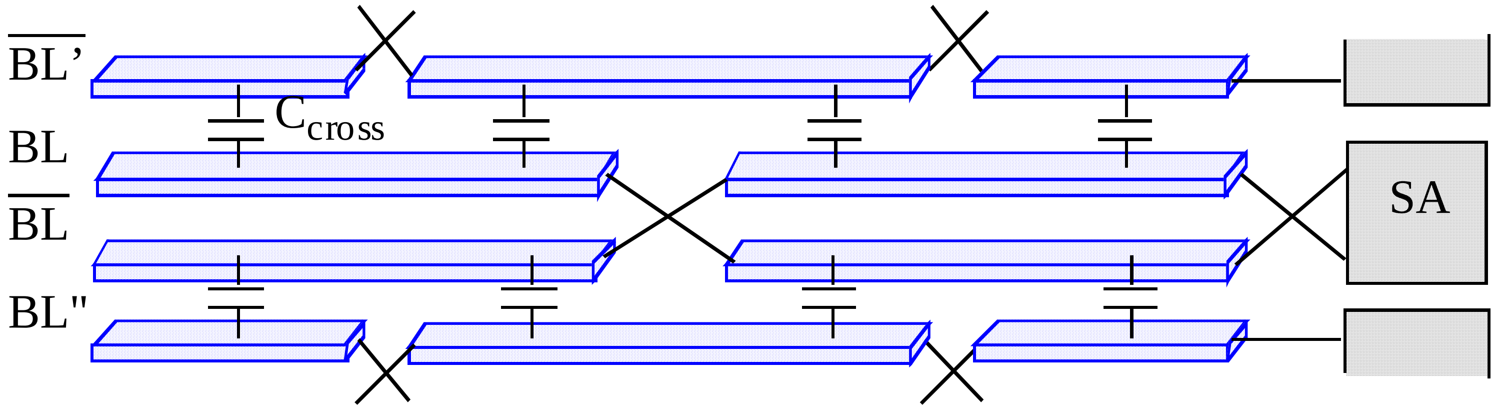
Folded-Bitline Architecture



Transposed-Bitline Architecture

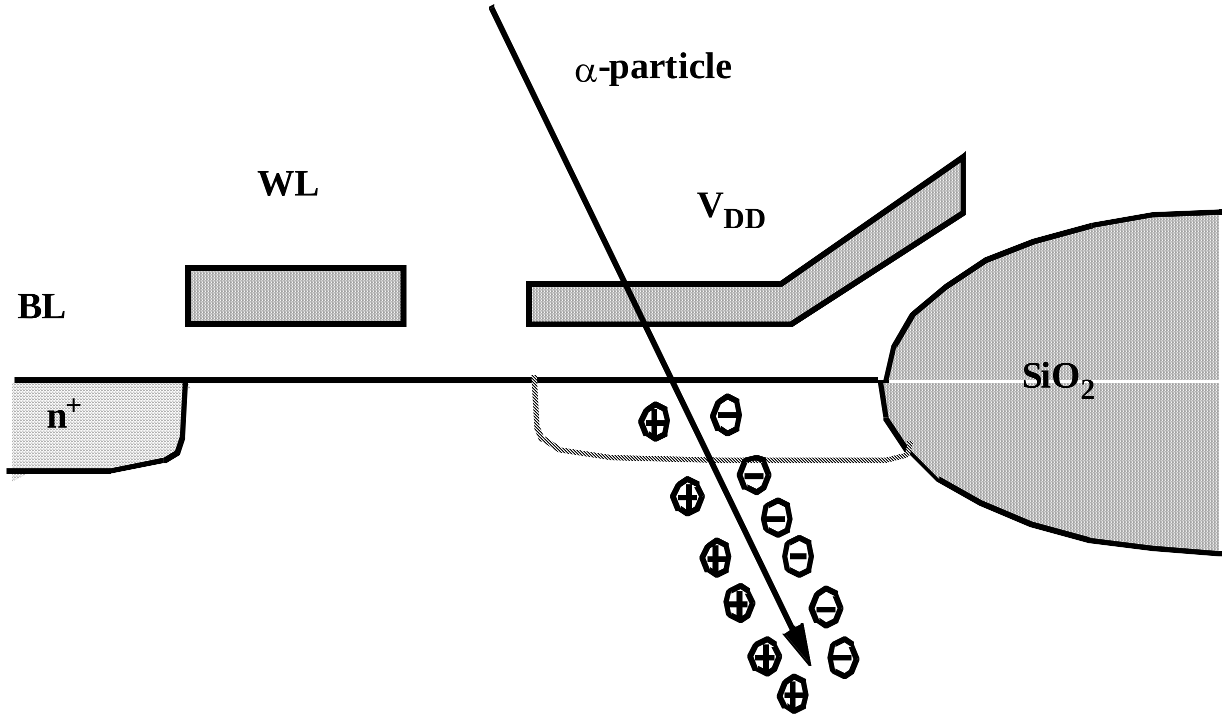


(a) Straightforward bitline routing.



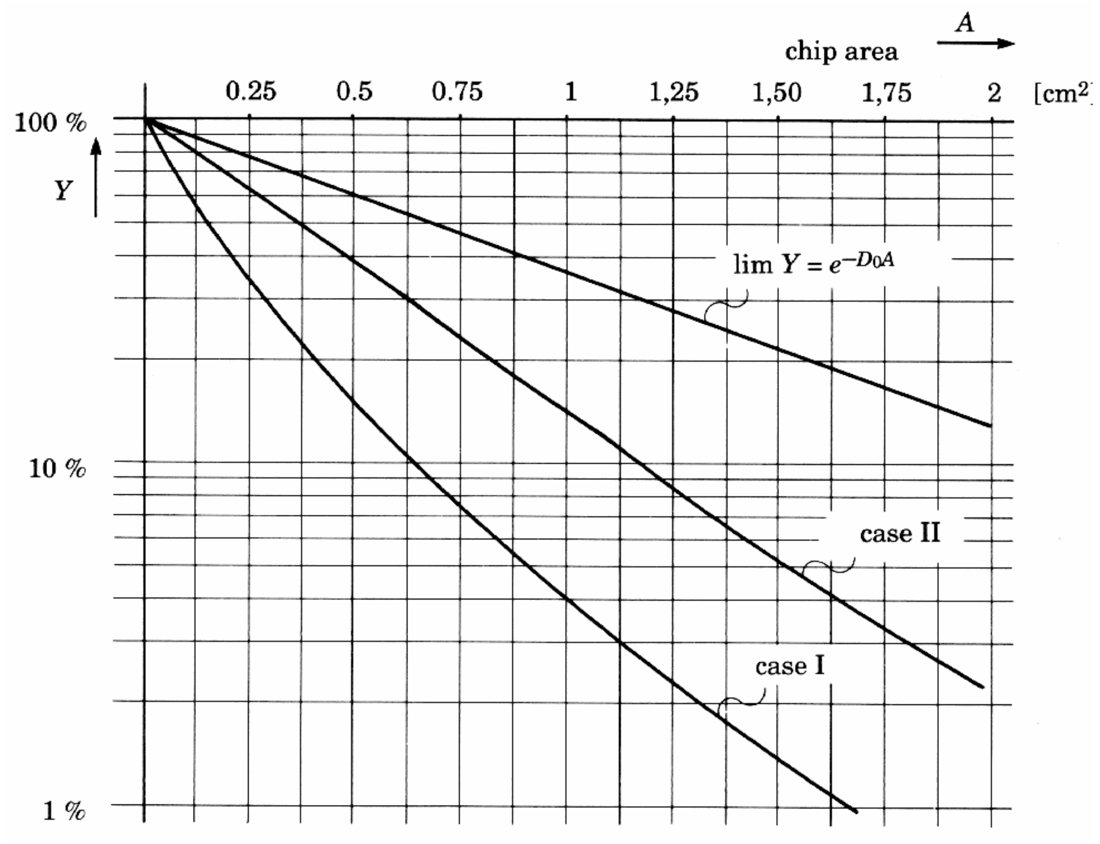
(b) Transposed bitline architecture.

Alpha-particles



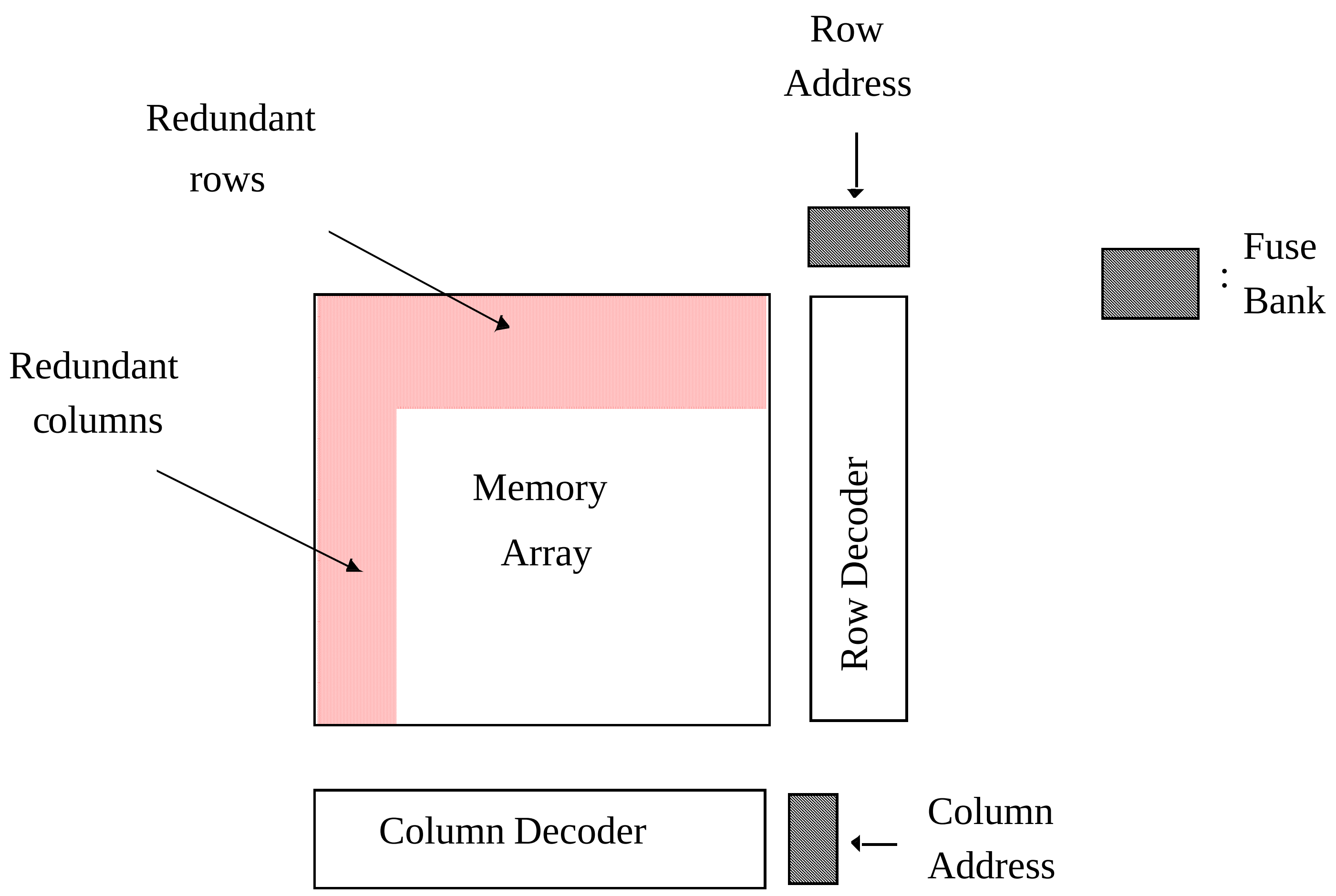
1 particle ~ 1 million carriers

Yield

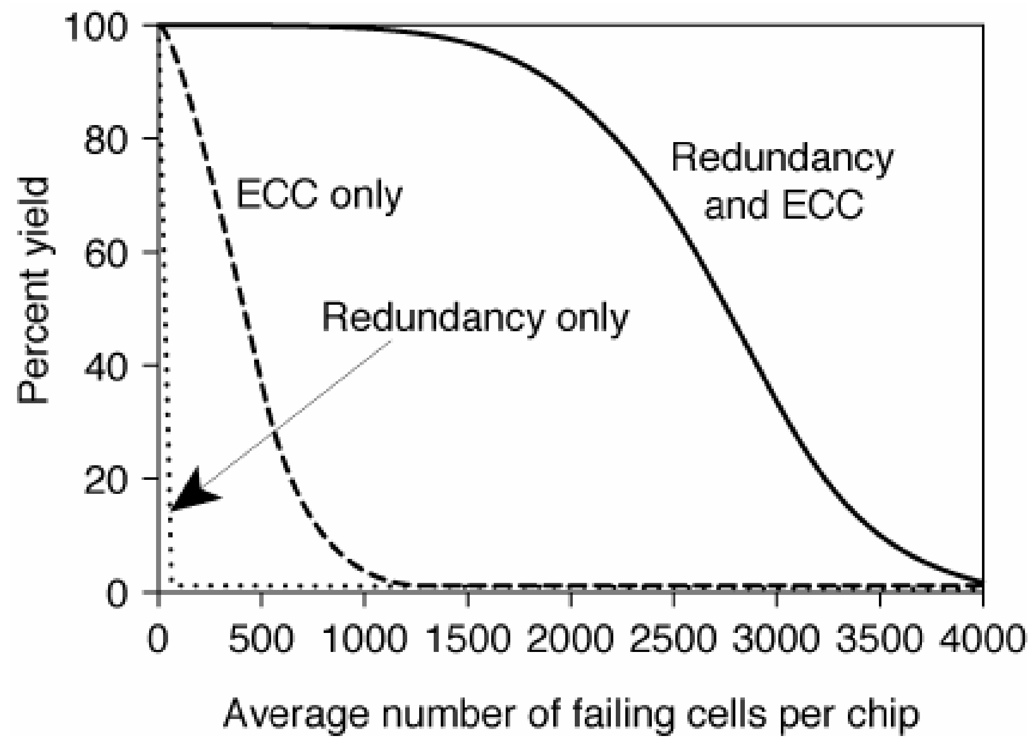


Yield curves at different stages of process maturity (from [Veendrick92])

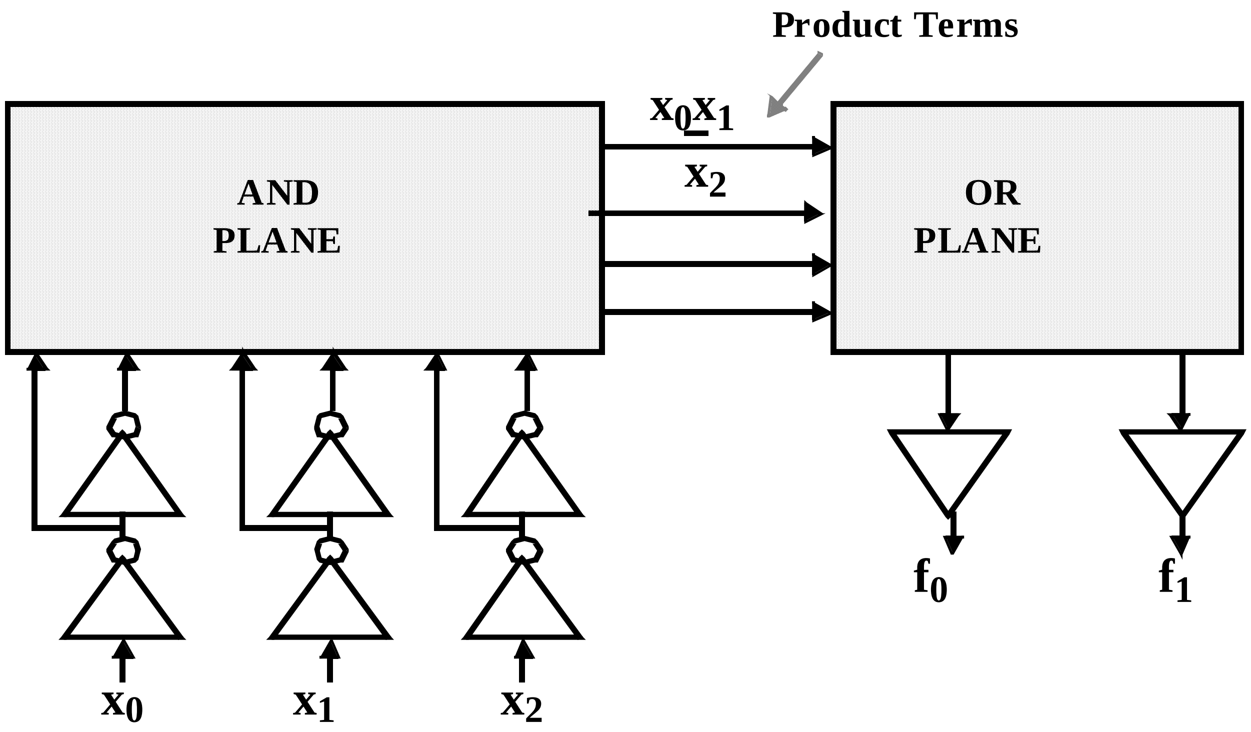
Redundancy


Row
Address
Redundant
rows
Fuse
:
Bank
Redundant
columns
Memory
Array
Row Decoder
Column Decoder
Column
Address

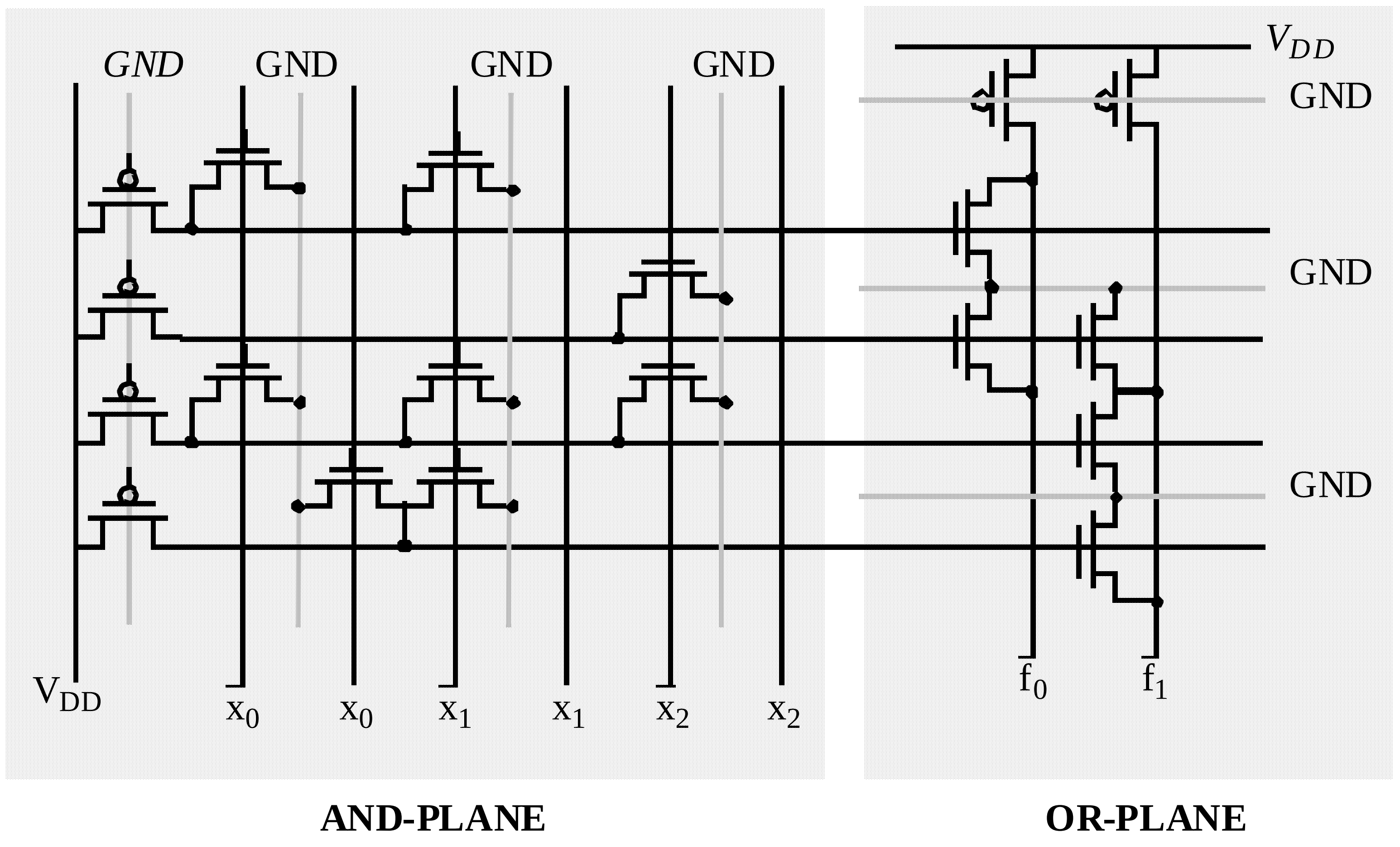
Redundancy and Error Correction



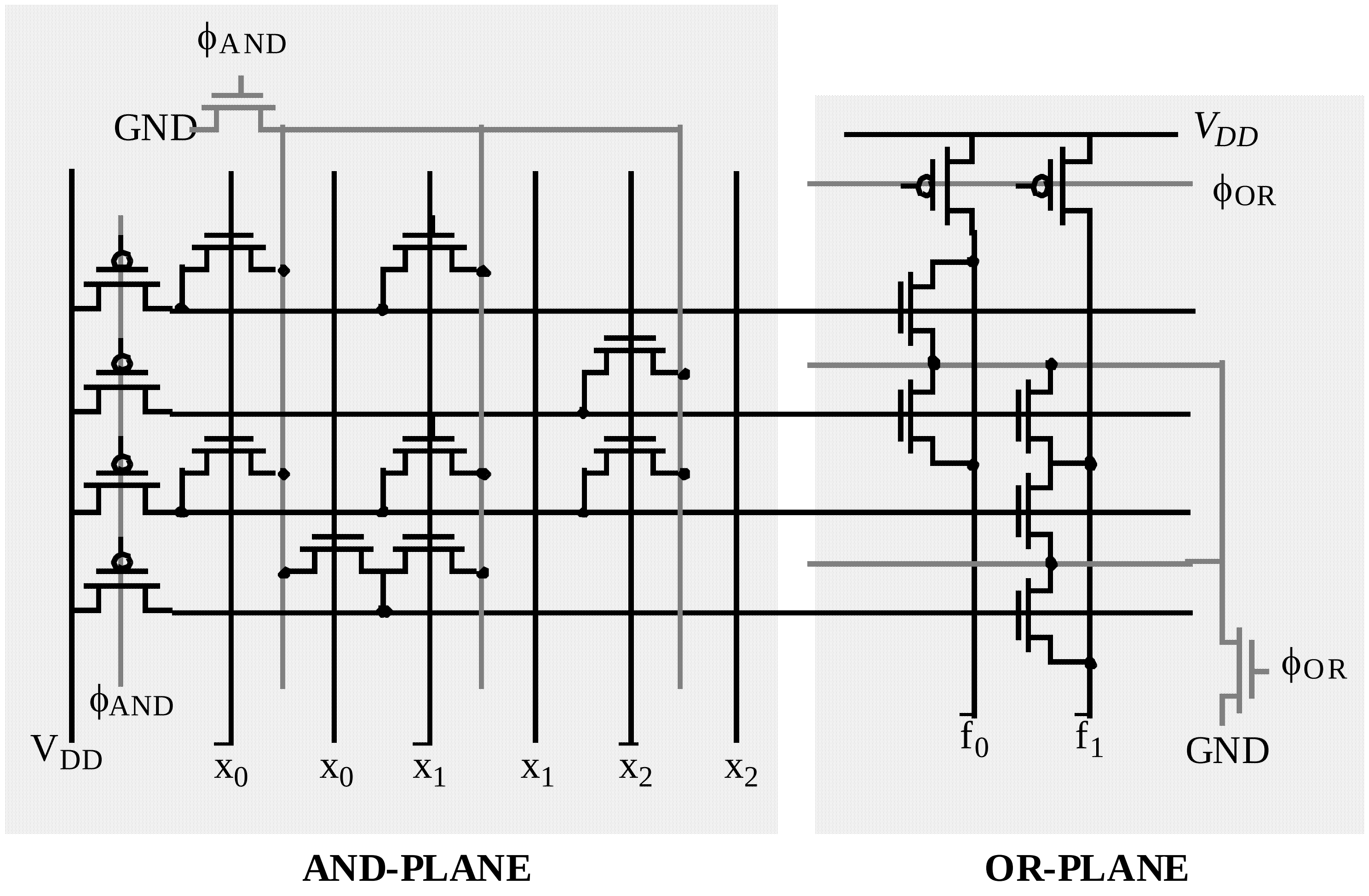
Programmable Logic Array



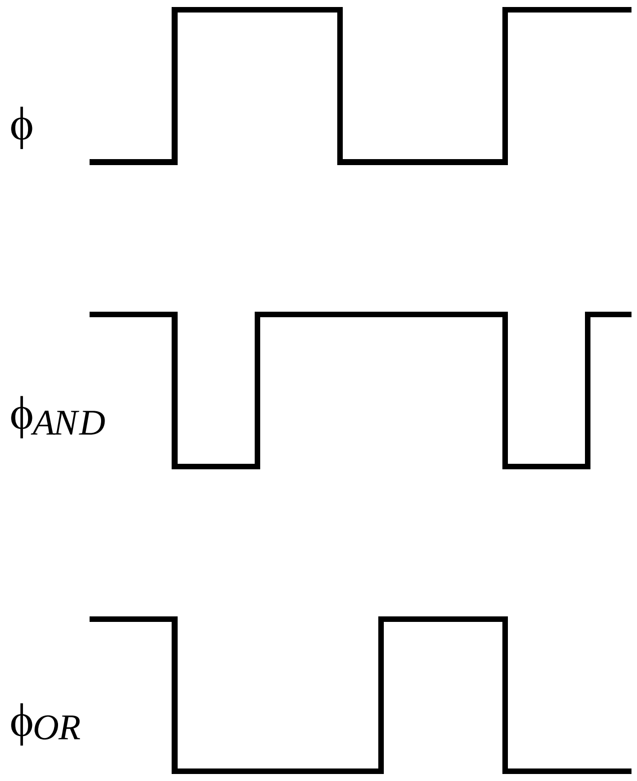
Pseudo-Static PLA



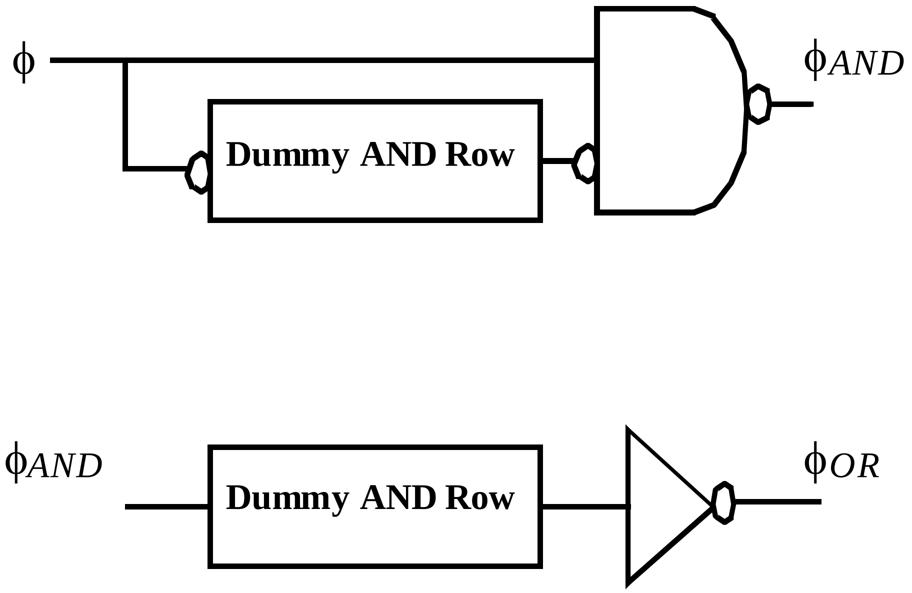
Dynamic PLA


ϕ_{AND}
GND
V_{DD}
ϕ_{OR}
ϕ_{AND}
V_{DD}
x_0
x_0
x_1
x_1
x_2
x_2
f_0
f_1
ϕ_{OR}
GND
AND-PLANE
OR-PLANE

Clock Signal Generation for self-timed dynamic PLA

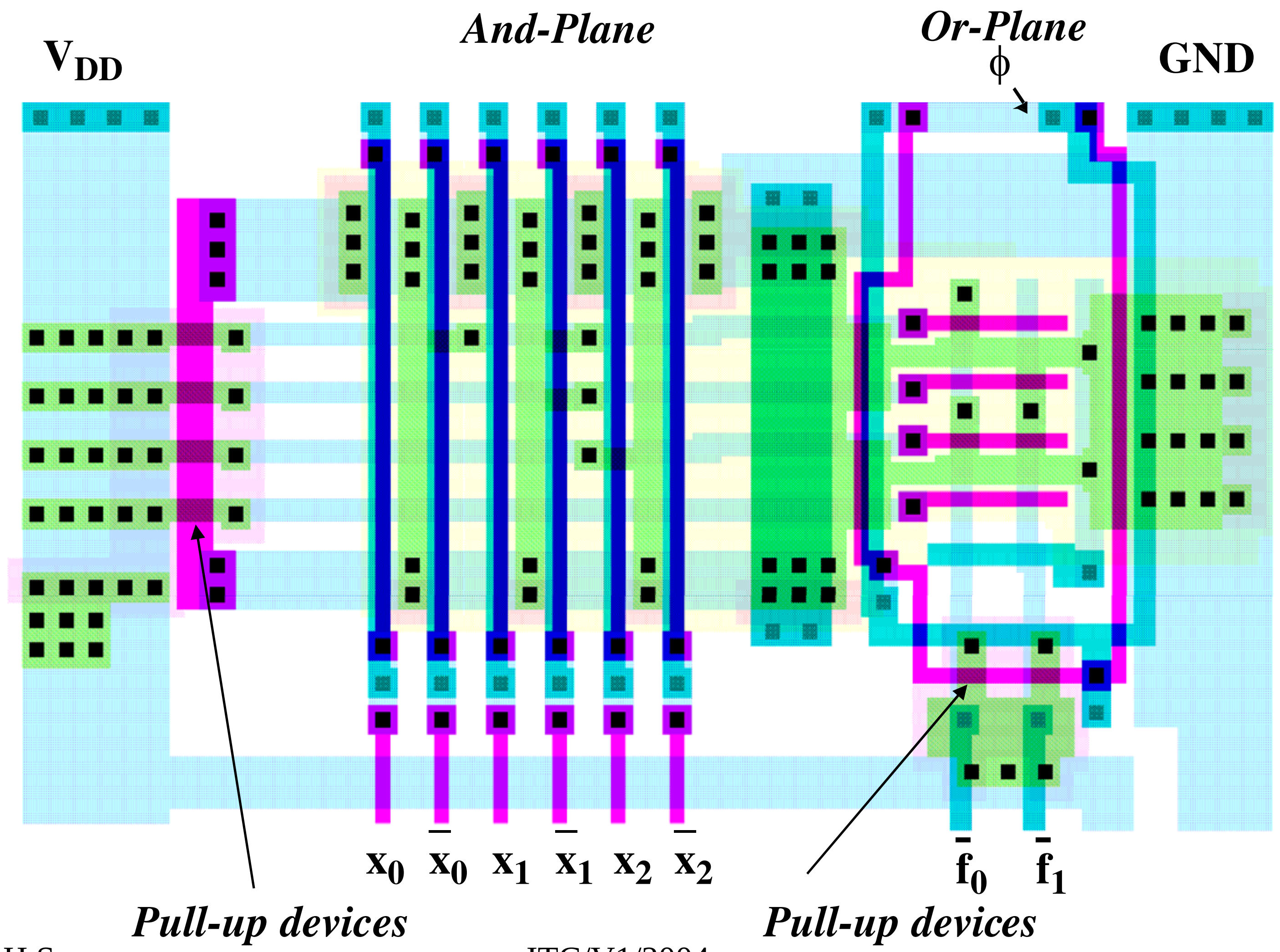


(a) Clock signals



(b) Timing generation circuitry.

PLA Layout



V_{DD}
And-Plane
Or-Plane
ϕ
GND
x_0
$\overline{x_0}$
x_1
$\overline{x_1}$
x_2
$\overline{x_2}$
$\overline{f_0}$
$\overline{f_1}$
Pull-up devices
Pull-up devices

PLA versus ROM

Programmable Logic Array

- **structured approach to random logic**
- **“two level logic implementation”**
  - **NOR-NOR (product of sums)**
  - **NAND-NAND (sum of products)**

IDENTICAL TO ROM!

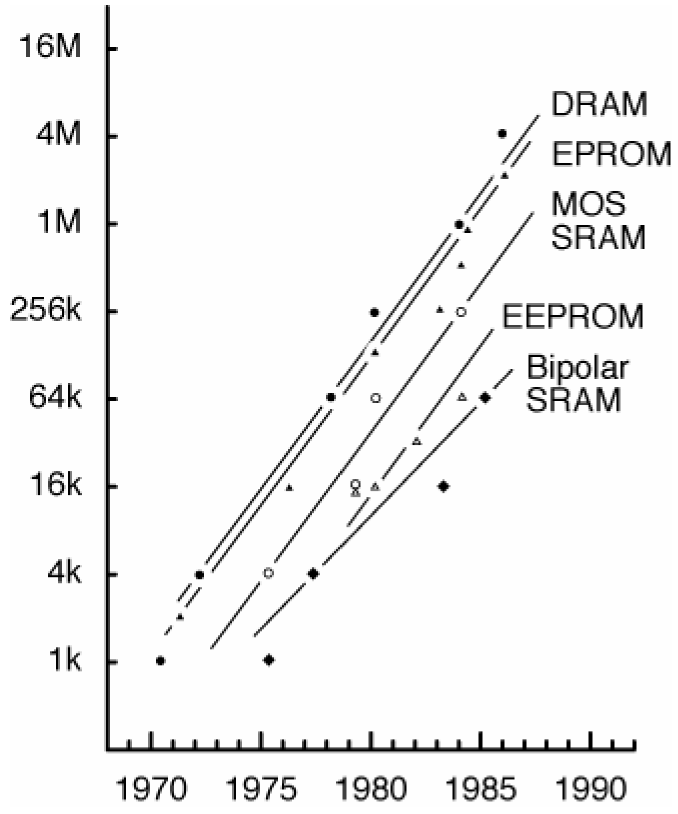
Main difference

- **ROM: fully populated**
- **PLA: one element per minterm**

Note: Importance of PLA’s has drastically reduced

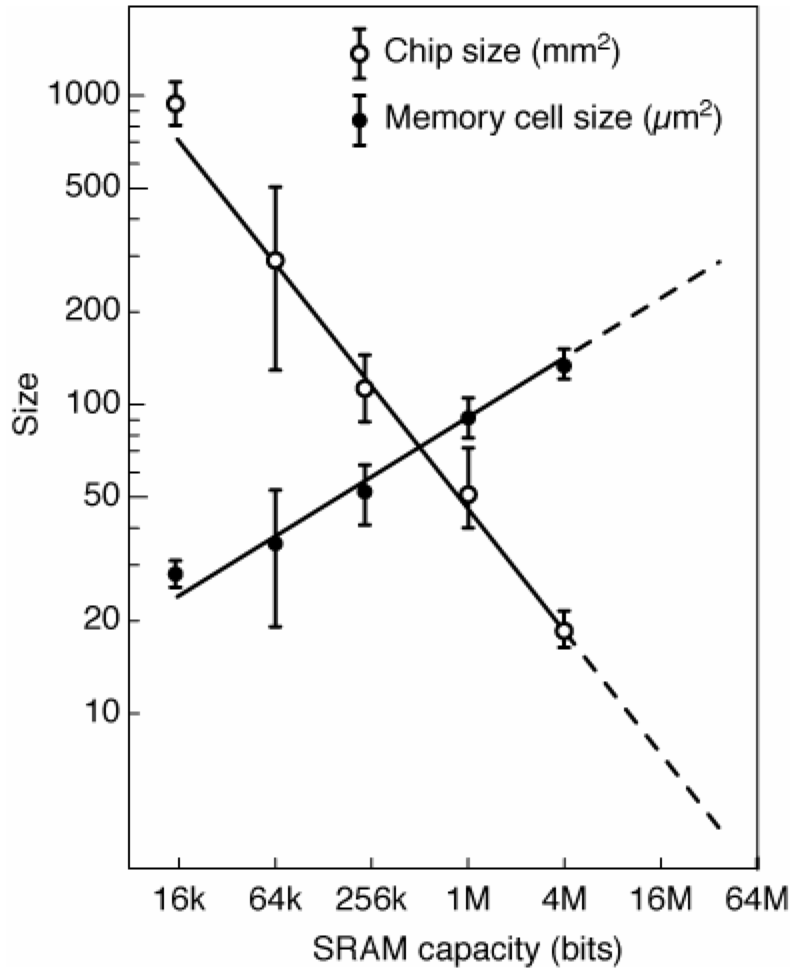
1. **slow**
2. **better software techniques (mutli-level logic synthesis)**

Semiconductor Memory Trends



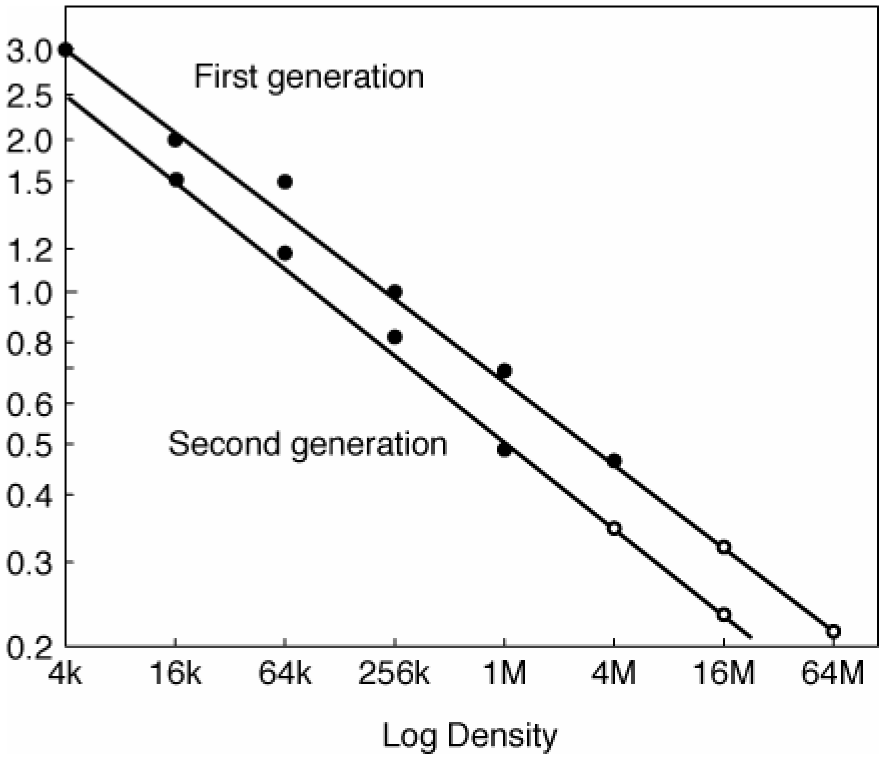
Memory Size as a function of time: x 4 every three years

Semiconductor Memory Trends



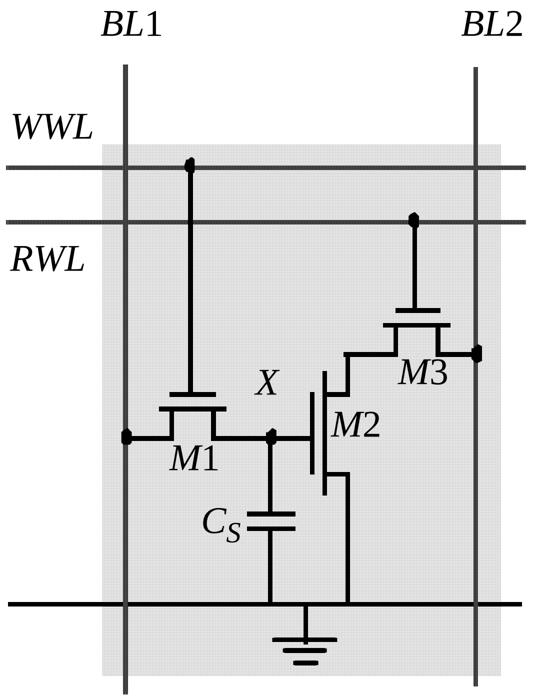
Increasing die size
factor 1.5 per generation
Combined with reducing cell size
factor 2.6 per generation

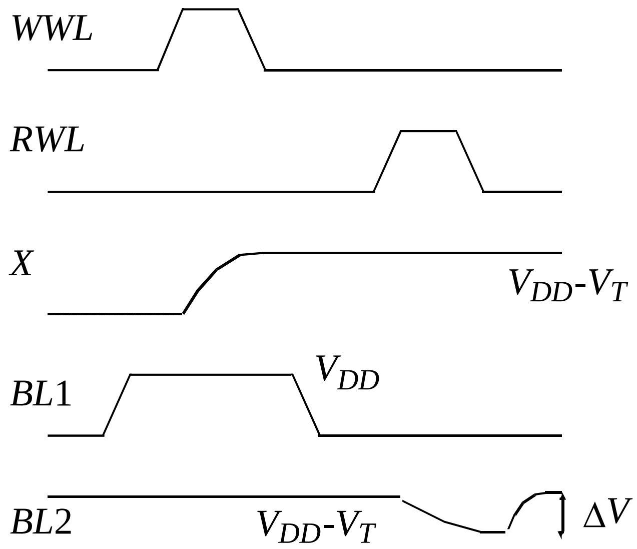
Semiconductor Memory Trends



Technology feature size for different SRAM generations

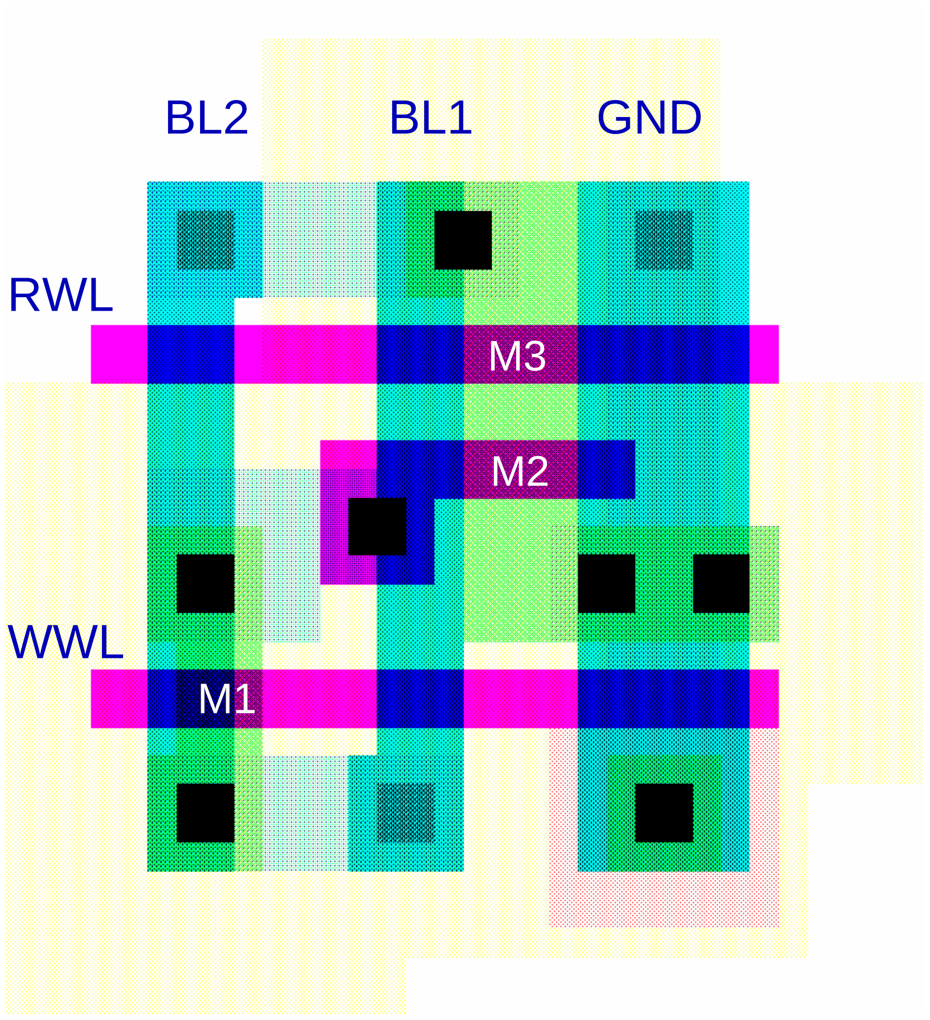
3-Transistor DRAM Cell



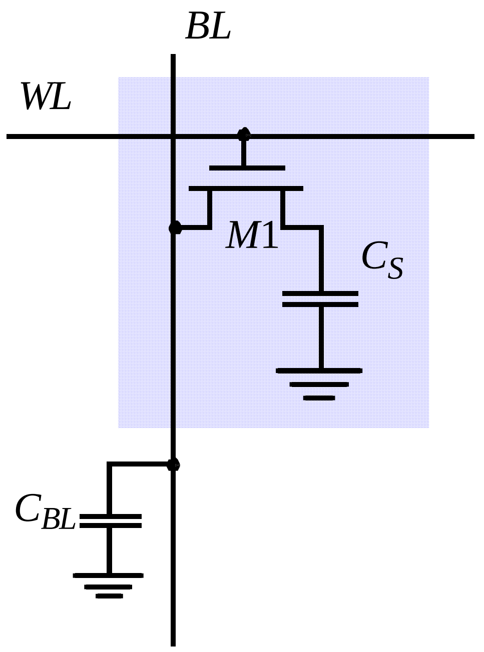


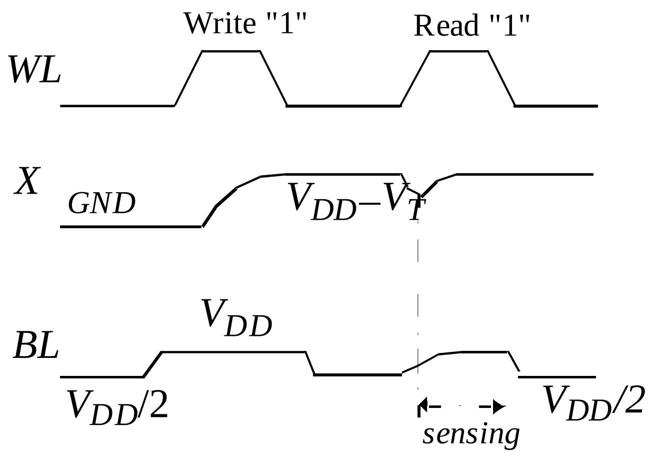
No constraints on device ratios
Reads are non-destructive
Value stored at node X when writing a "1" = $V_{WWL}-V_{Tn}$

3T-DRAM — Layout



1-Transistor DRAM Cell





Write: C_S is charged or discharged by asserting WL and BL.
Read: Charge redistribution takes places between bit line and storage capacitance

$$\Delta V = V_{BL} - V_{PRE} = (V_{BIT} - V_{PRE})\frac{C_S}{C_S + C_{BL}}$$

Voltage swing is small; typically around 250 mV.

DRAM Cell Observations

1T DRAM requires a sense amplifier for each bit line, due to charge redistribution read-out.

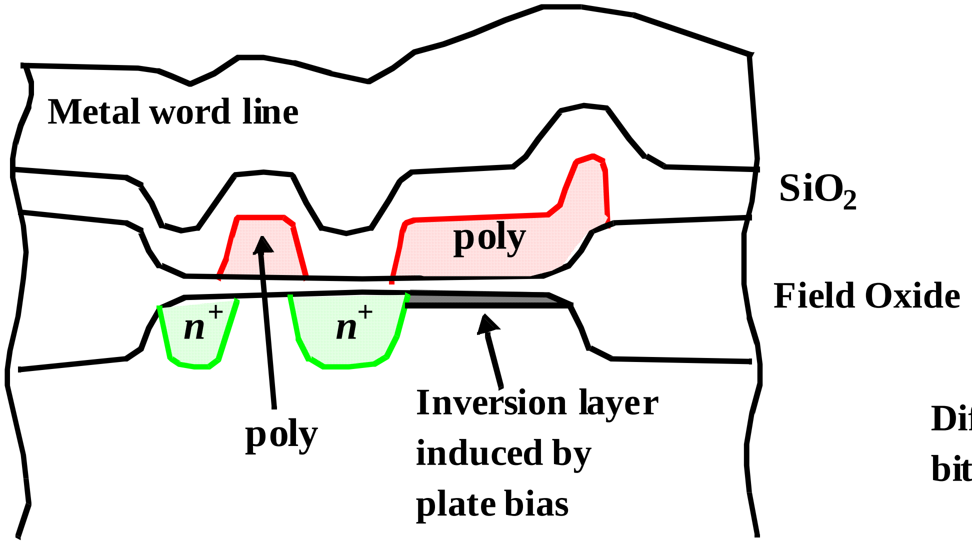
DRAM memory cells are single ended in contrast to SRAM cells.

The read-out of the 1T DRAM cell is destructive; read and refresh operations are necessary for correct operation.

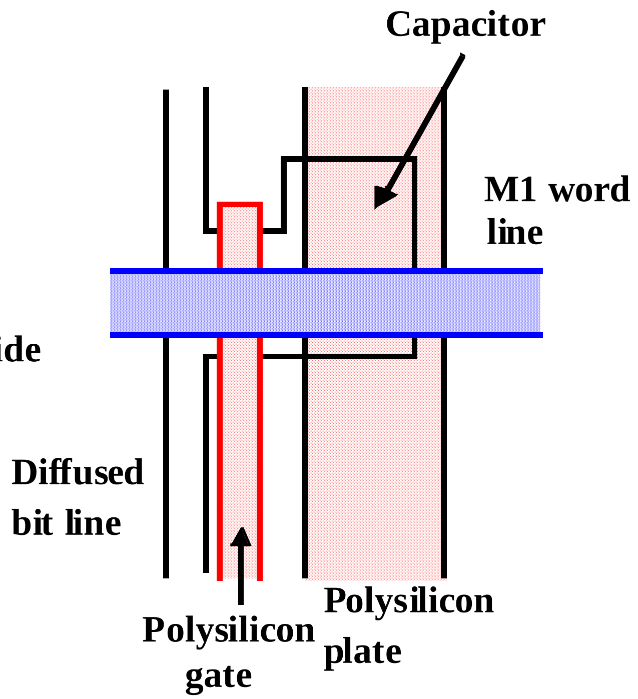
Unlike 3T cell, 1T cell requires presence of an extra capacitance that must be explicitly included in the design.

When writing a “1” into a DRAM cell, a threshold voltage is lost. This charge loss can be circumvented by bootstrapping the word lines to a higher value than V_{DD}.

1-T DRAM Cell



(a) Cross-section



(b) Layout

Used Polysilicon-Diffusion Capacitance

Expensive in Area

Periphery

- **Decoders**
- **Sense Amplifiers**
- **Input/Output Buffers**
- **Control / Timing Circuitry**

Row Decoders

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

(N)AND Decoder

$$WL_0 = A_0A_1A_2A_3A_4A_5A_6A_7A_8A_9$$

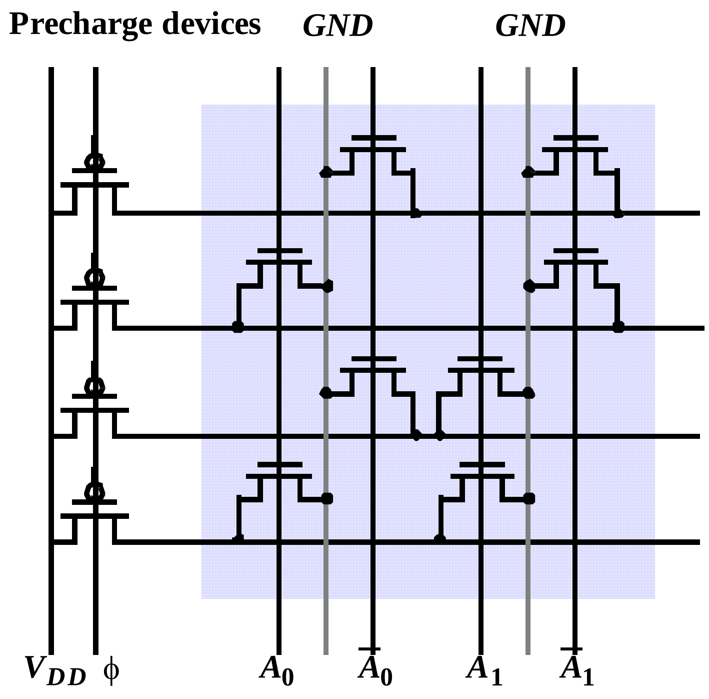
$$WL_{511} = \bar{A}_0A_1A_2A_3A_4A_5A_6A_7A_8A_9$$

NOR Decoder

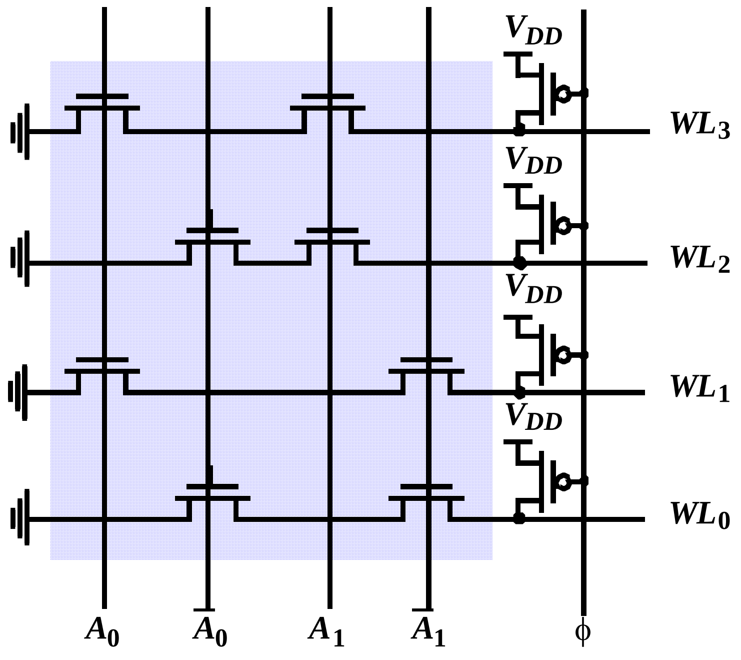
$$WL_0 = \overline{A_0+A_1+A_2+A_3+A_4+A_5+A_6+A_7+A_8+A_9}$$

$$WL_{511} = \overline{A_0+\bar{A}_1+\bar{A}_2+\bar{A}_3+\bar{A}_4+\bar{A}_5+\bar{A}_6+\bar{A}_7+\bar{A}_8+\bar{A}_9}$$

Dynamic Decoders



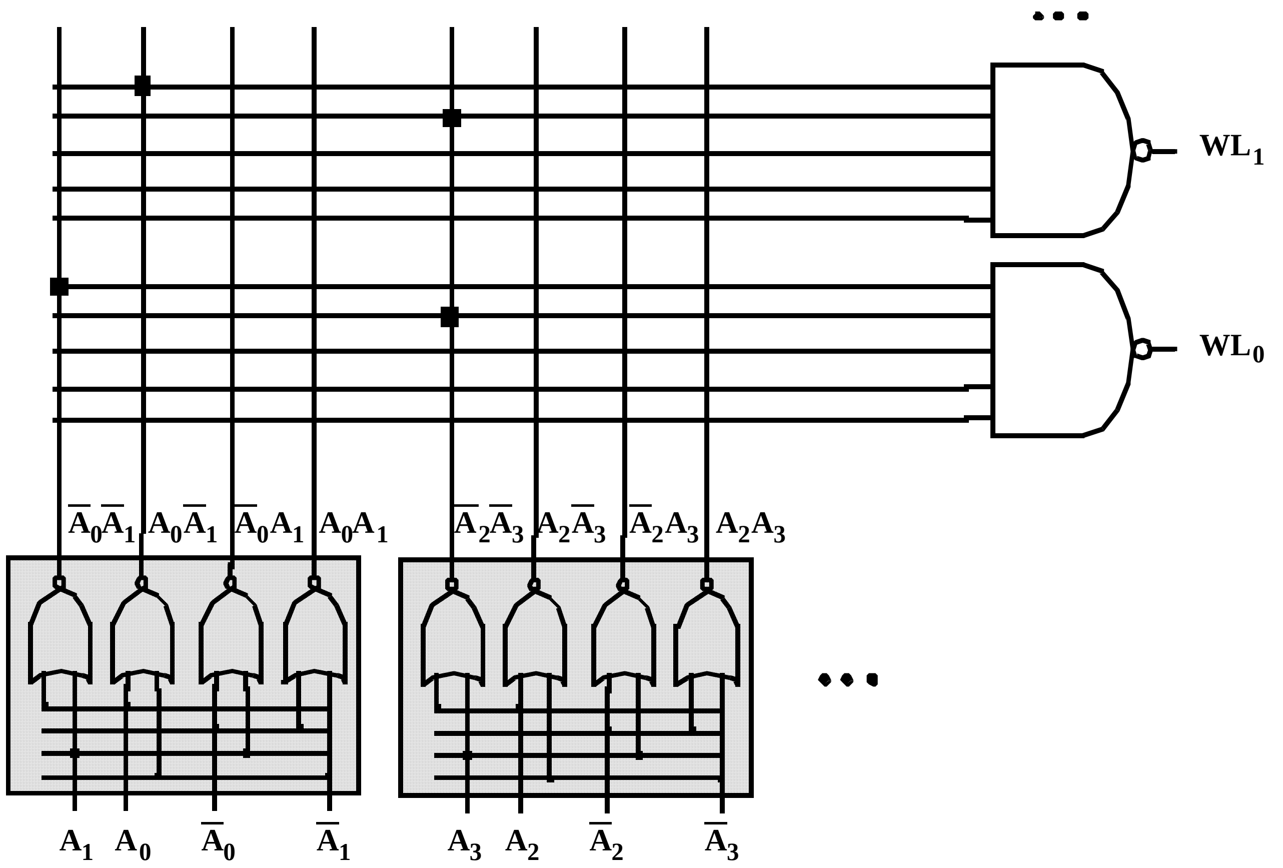
Dynamic 2-to-4 NOR decoder



2-to-4 MOS dynamic NAND Decoder

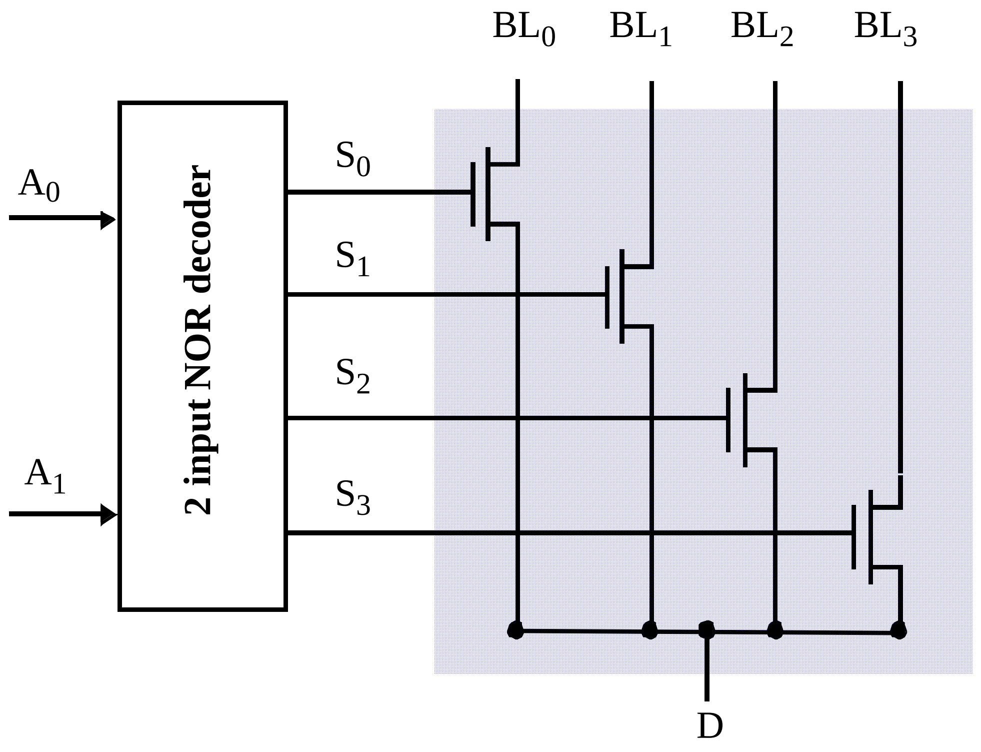
Propagation delay is primary concern

A NAND decoder using 2-input pre-decoders



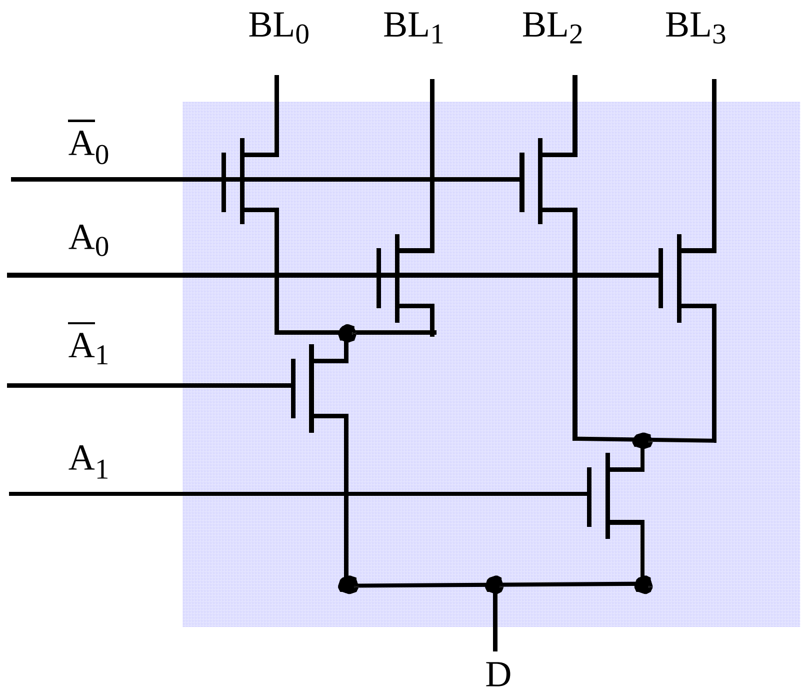
Splitting decoder into two or more logic layers produces a faster and cheaper implementation

4 input pass-transistor based column decoder



lvantage: speed (t_{pd} does not add to overall memory access time)
only 1 extra transistor in signal path
sadvantage: large transistor count

4-to-1 tree based column decoder



Number of devices drastically reduced

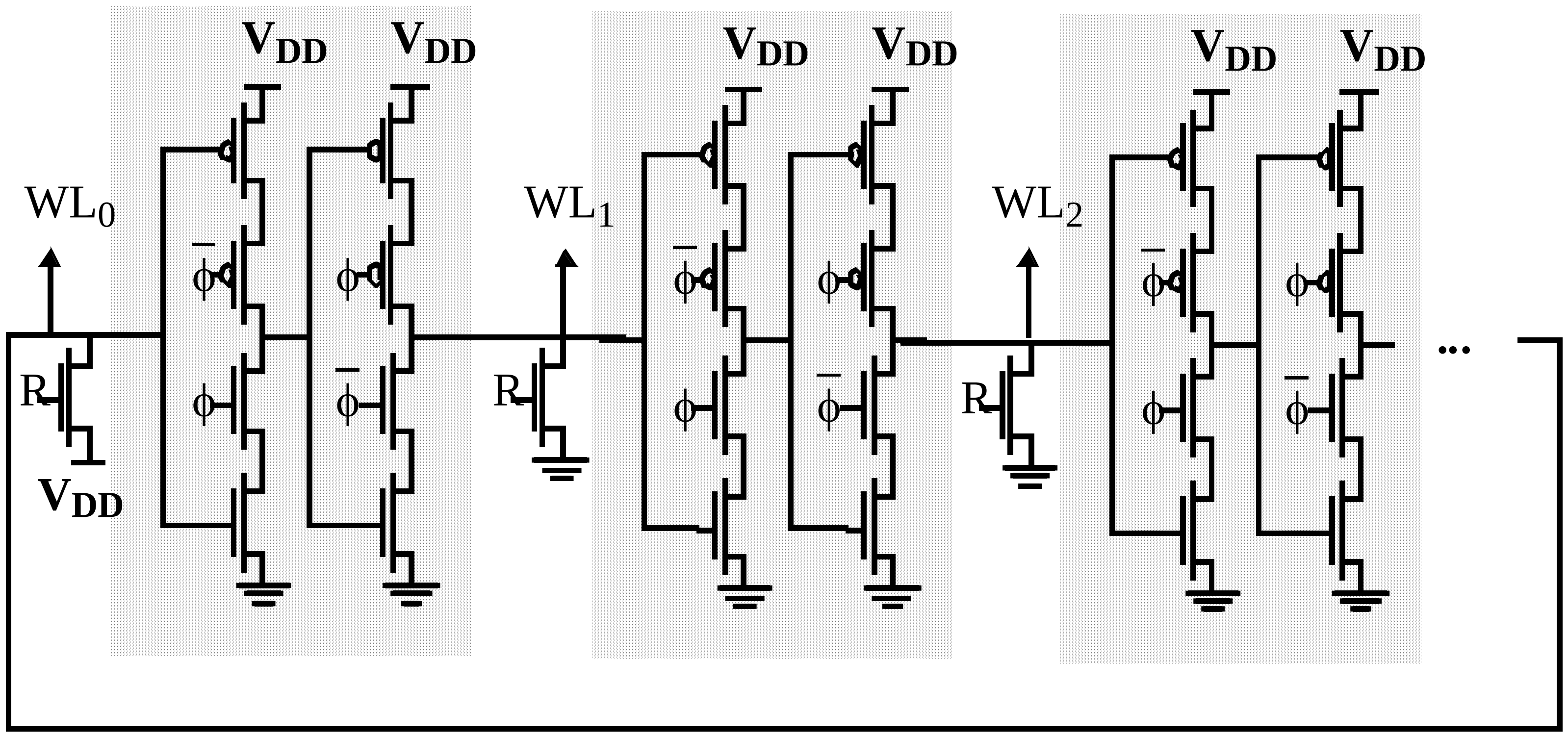
Delay increases quadratically with # of sections; prohibitive for large decoders

Solutions: **buffers**

progressive sizing

combination of tree and pass transistor approaches

Decoder for circular shift-register



Sense Amplifiers

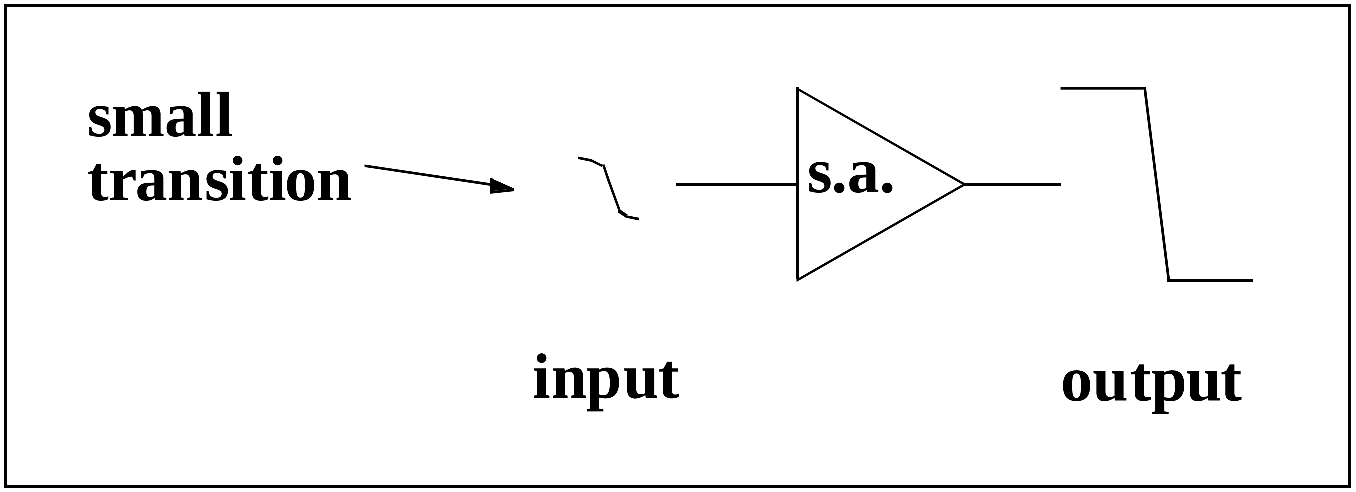
$$t_p = \frac{C \cdot \Delta V}{I_{av}}$$

make ΔV as small as possible

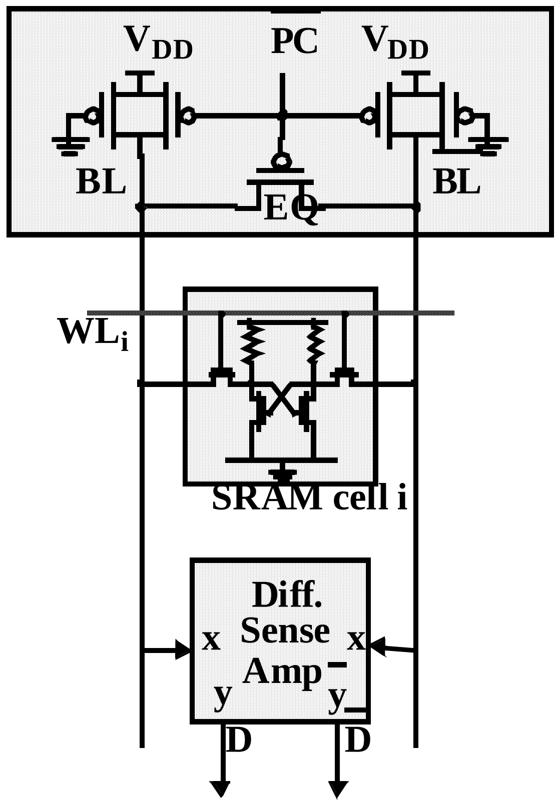
large

small

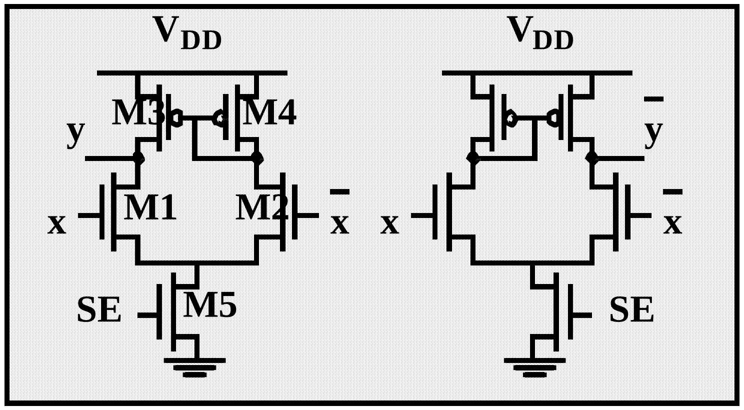
Idea: Use Sense Amplifer



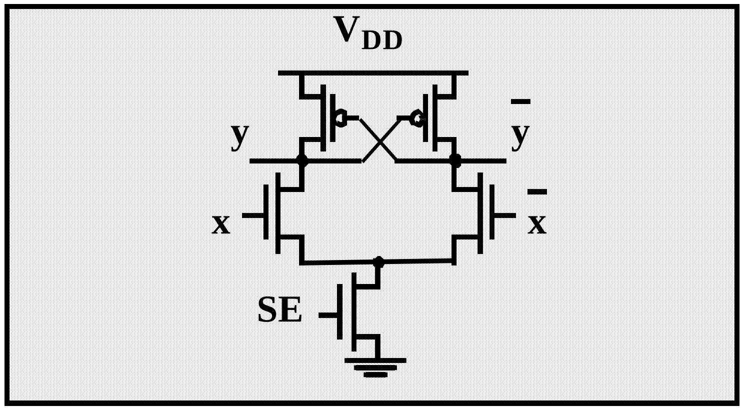
Differential Sensing - SRAM



(a) SRAM sensing scheme.

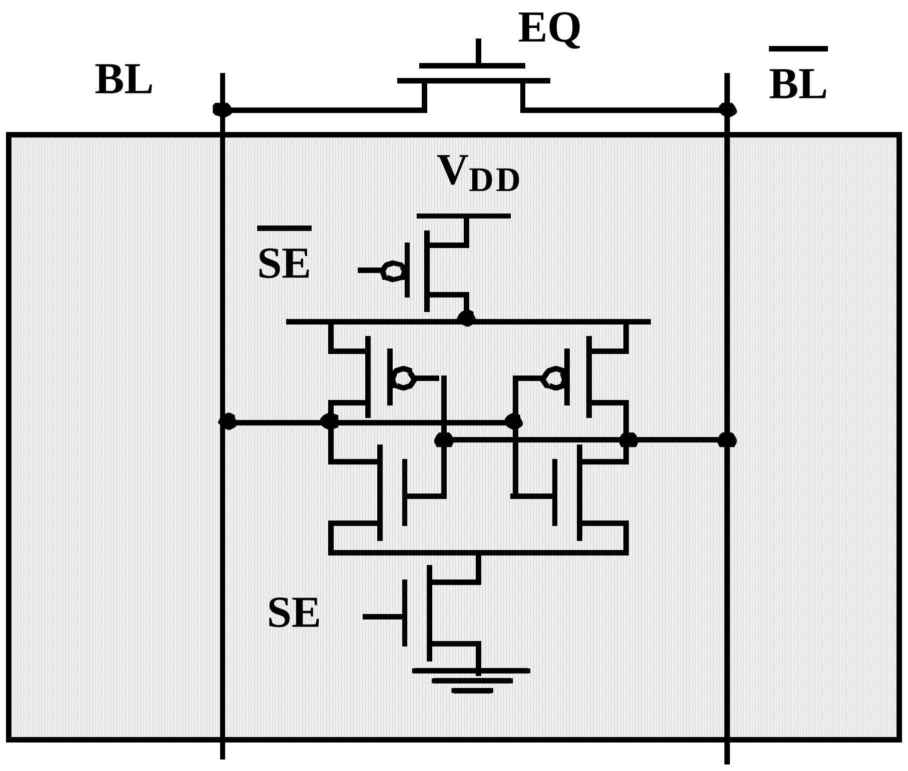


(b) Doubled-ended Current Mirror Amplifier



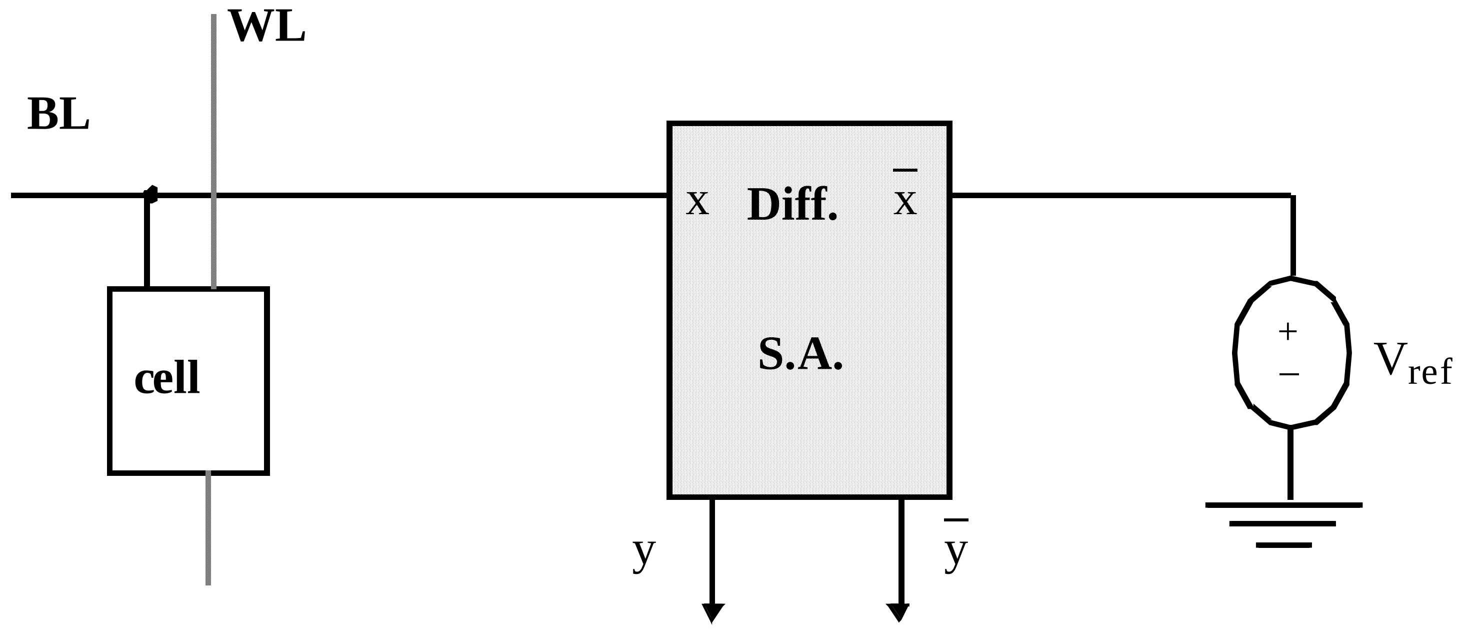
(c) Cross-Coupled Amplifier

Latch-Based Sense Amplifier



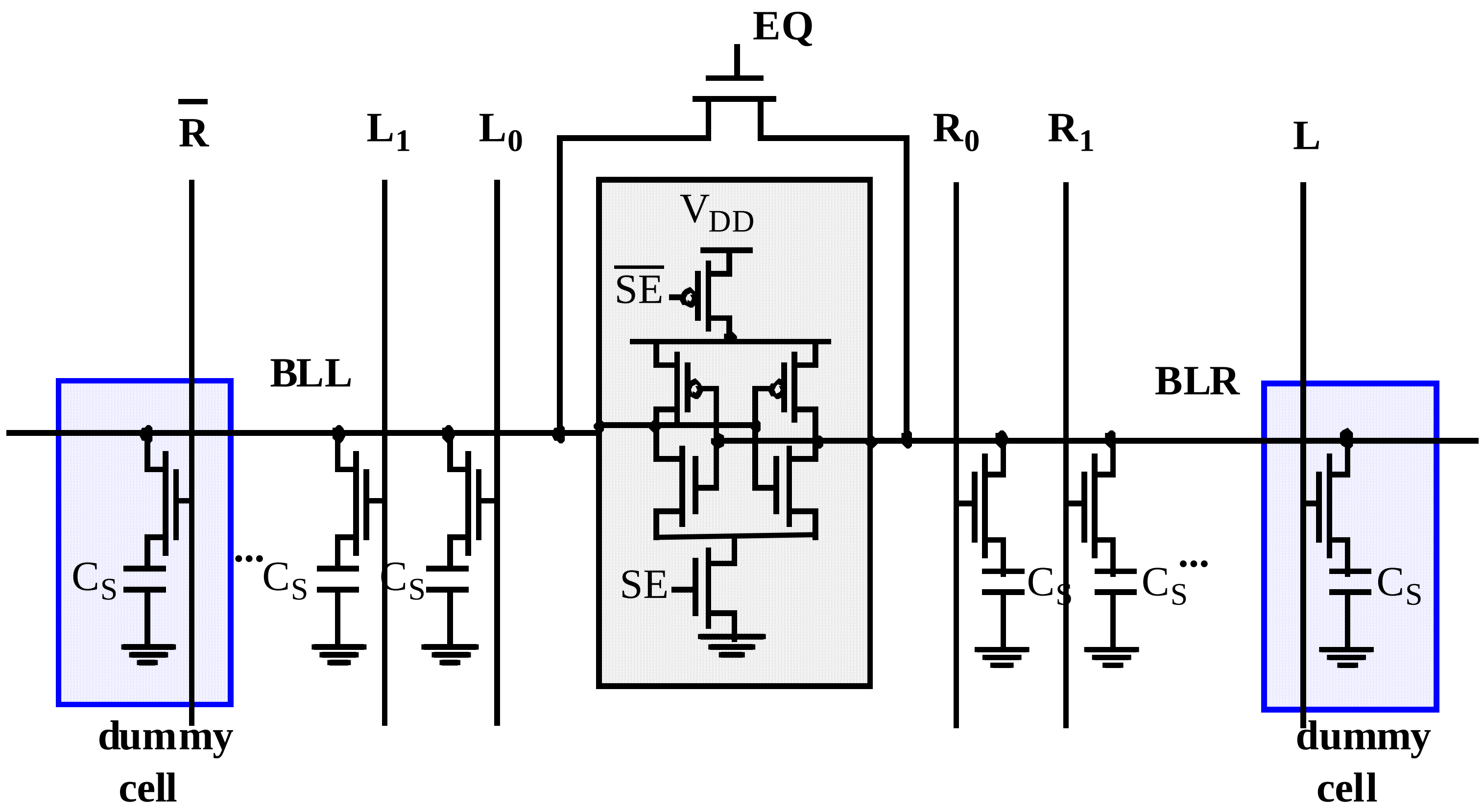
Initialized in its meta-stable point with EQ
Once adequate voltage gap created, sense amp enabled with SE
Positive feedback quickly forces output to a stable operating point.

Single-to-Differential Conversion

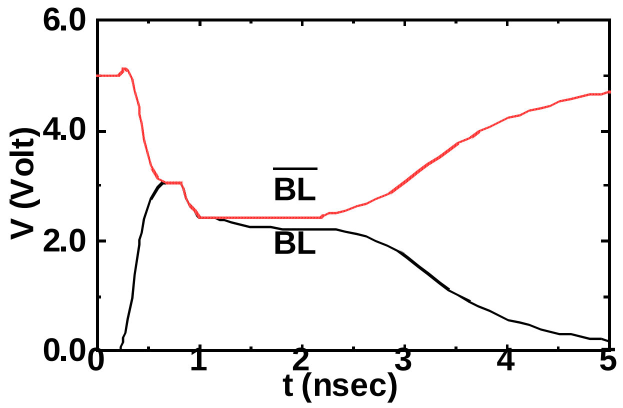


How to make good V_{ref}?

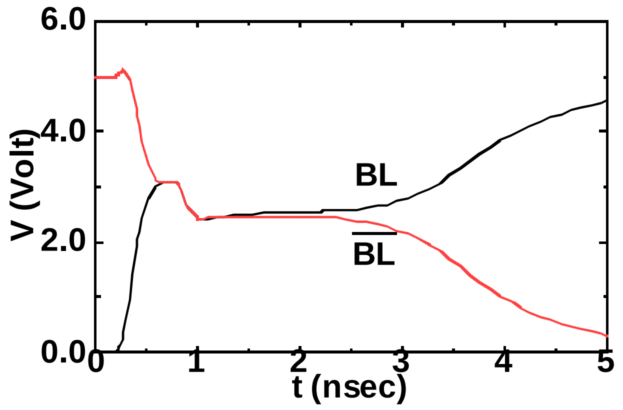
Open bitline architecture



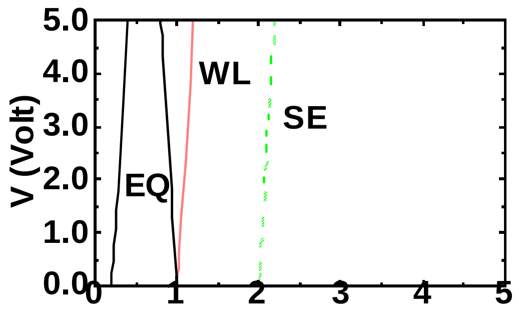
DRAM Read Process with Dummy Cell



(a) reading a zero

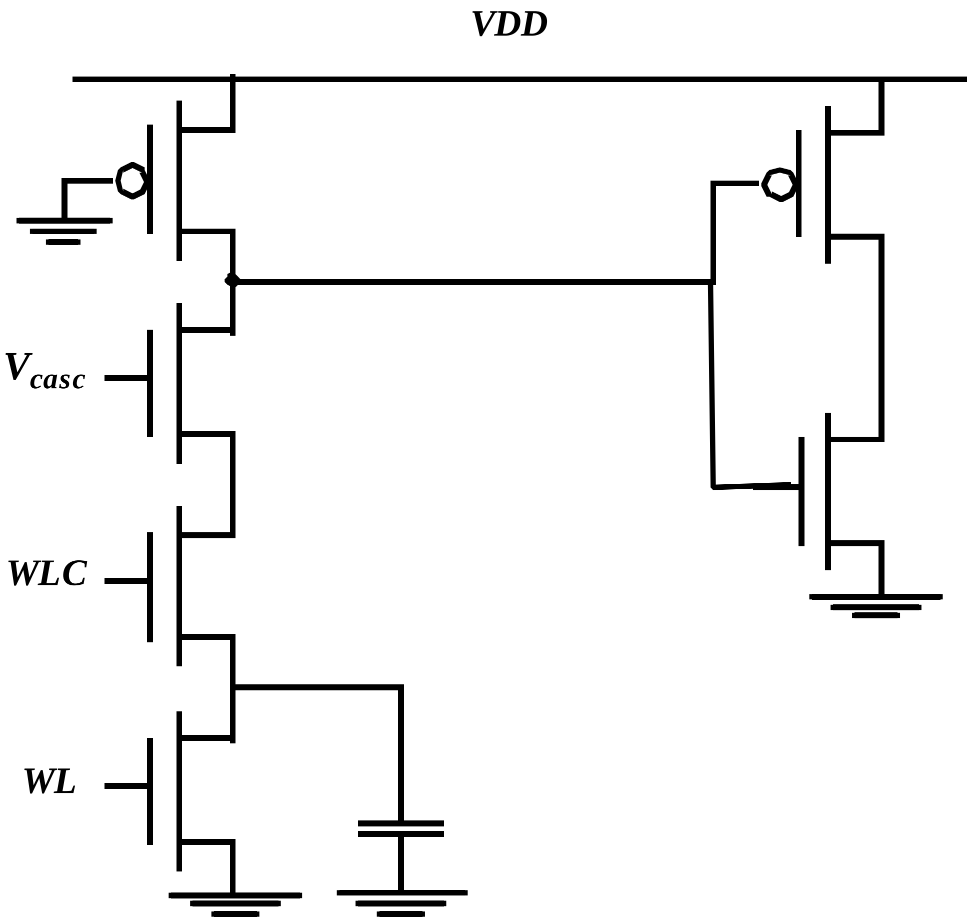


(b) reading a one

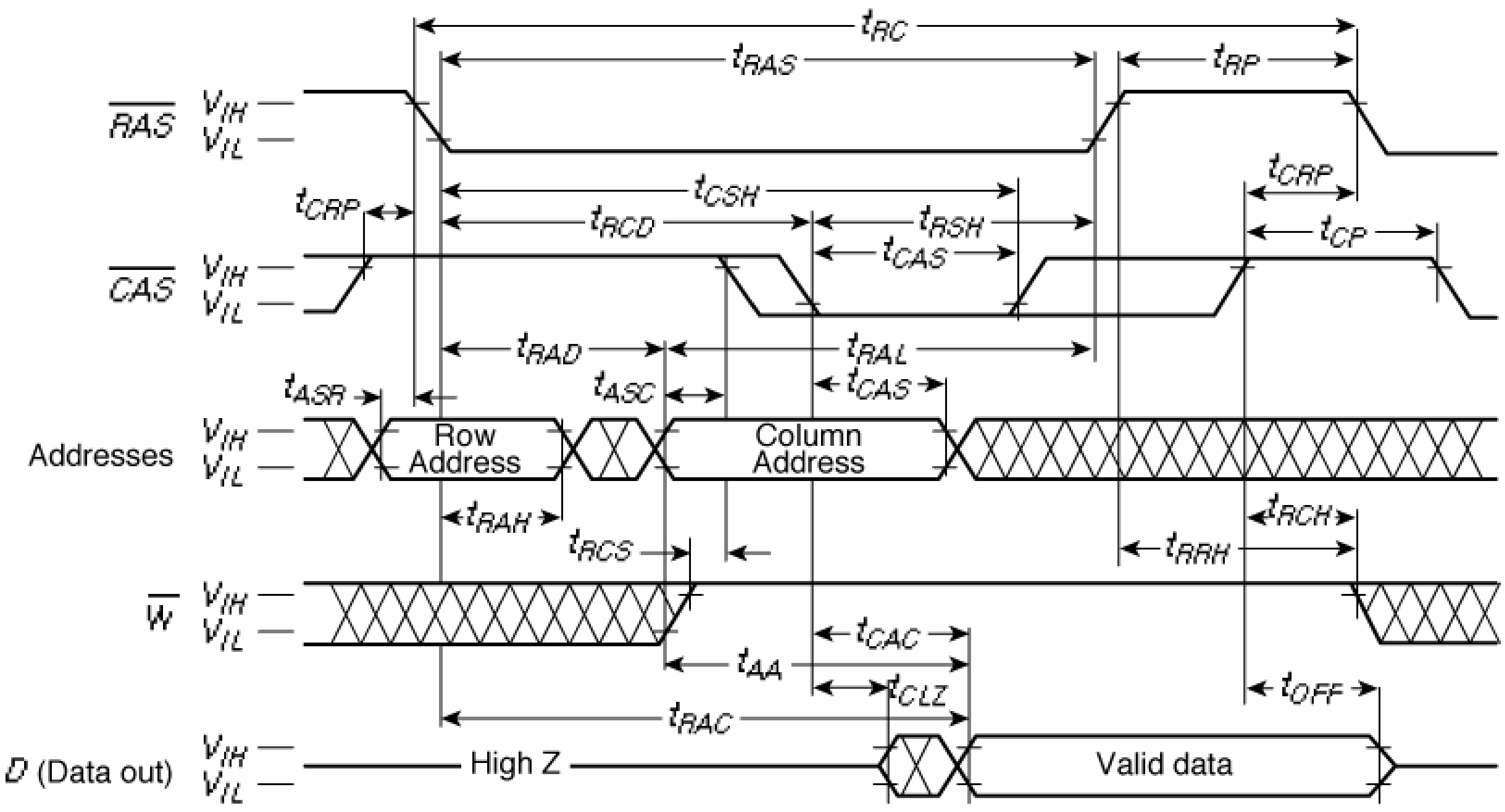


(c) control signals

Single-Ended Cascode Amplifier

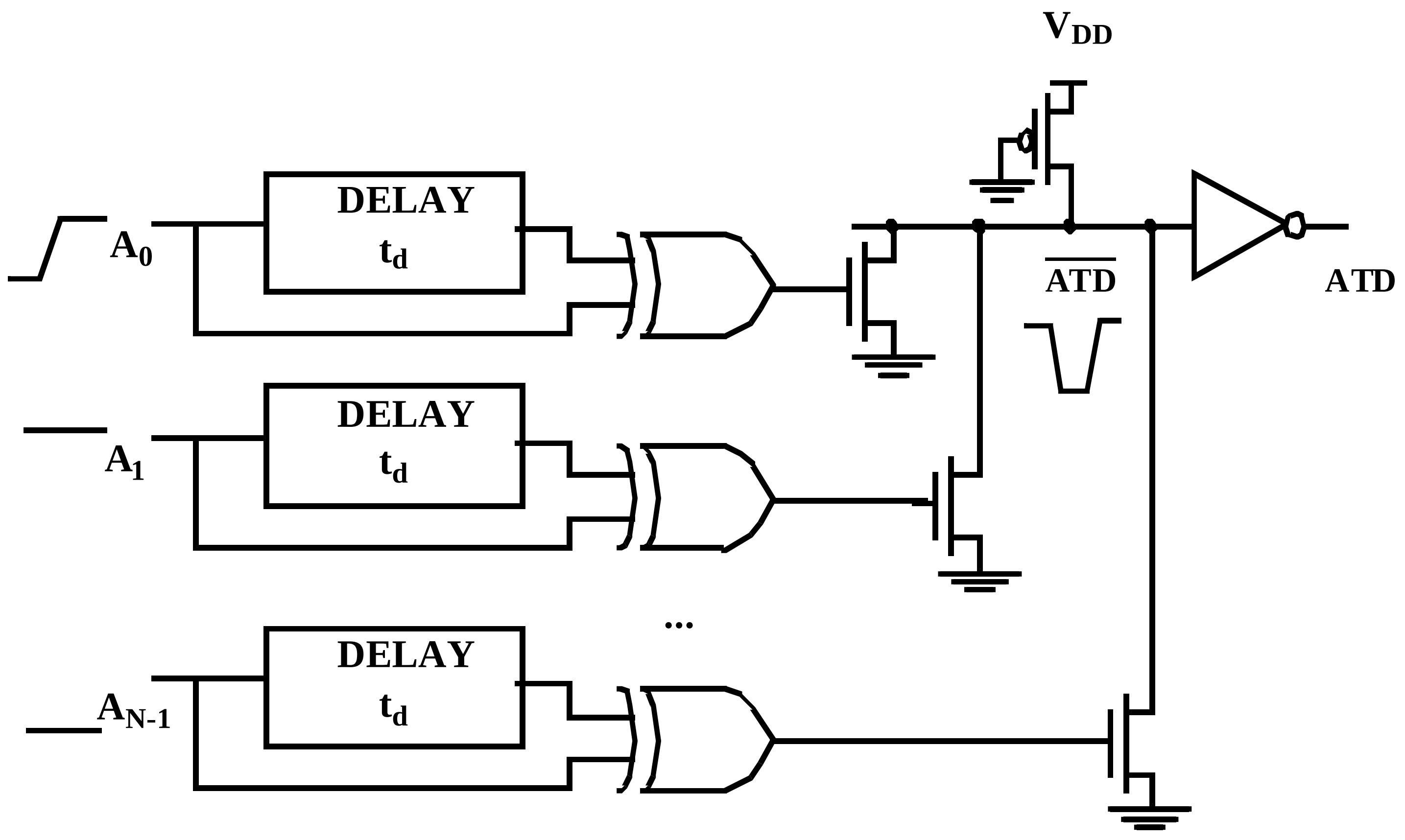


DRAM Timing

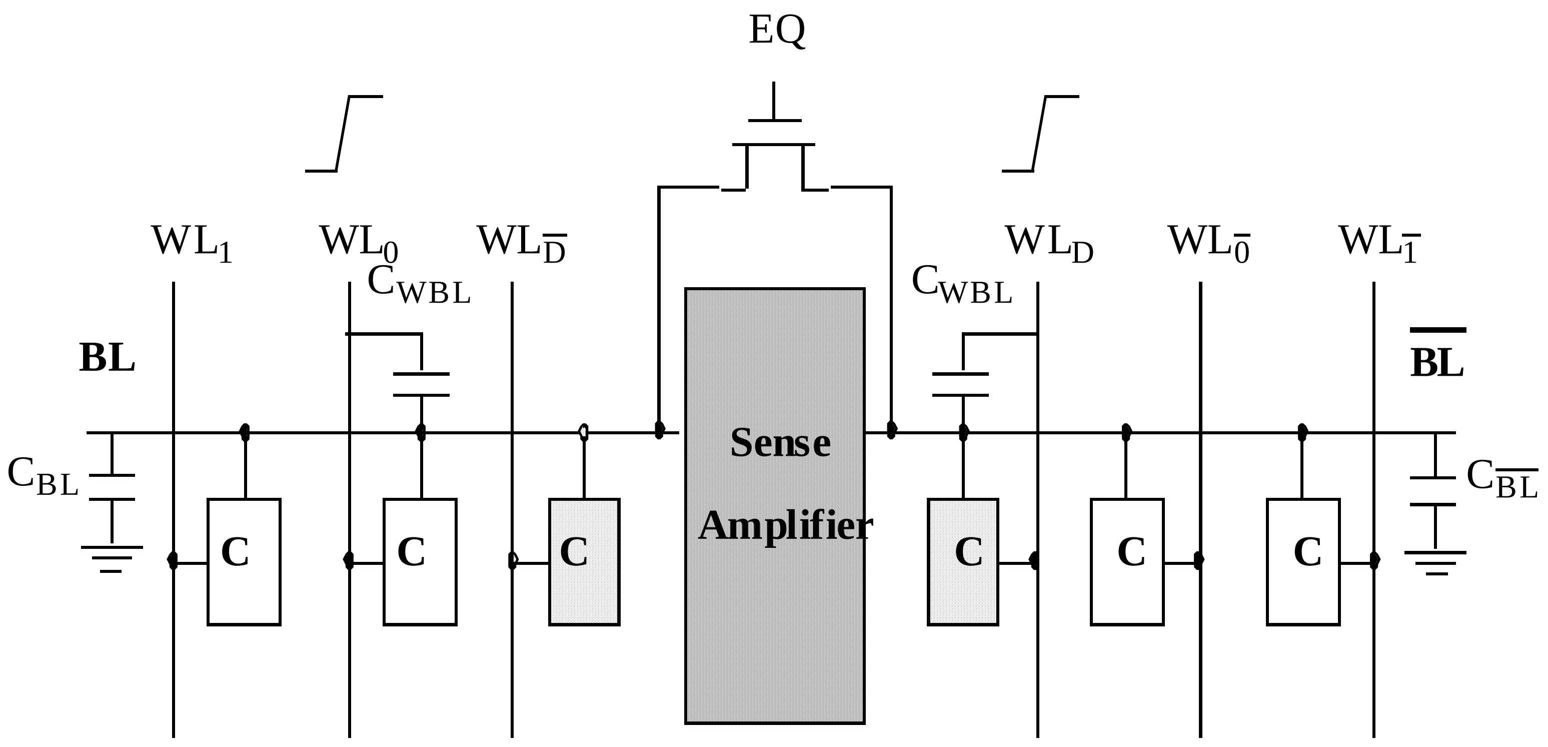


t_{RC}
t_{RAS}
t_{RP}
$\overline{RAS}$
V_{IH}
V_{IL}
t_{CSH}
t_{CRP}
t_{RCD}
t_{RSH}
t_{CAS}
t_{CP}
$\overline{CAS}$
t_{RAD}
t_{RAL}
t_{ASR}
t_{ASC}
Addresses
Row Address
Column Address
t_{RAH}
t_{RCS}
t_{RCH}
t_{RRH}
$\overline{W}$
t_{CAC}
t_{AA}
t_{CLZ}
t_{OFF}
t_{RAC}
D (Data out)
High Z
Valid data

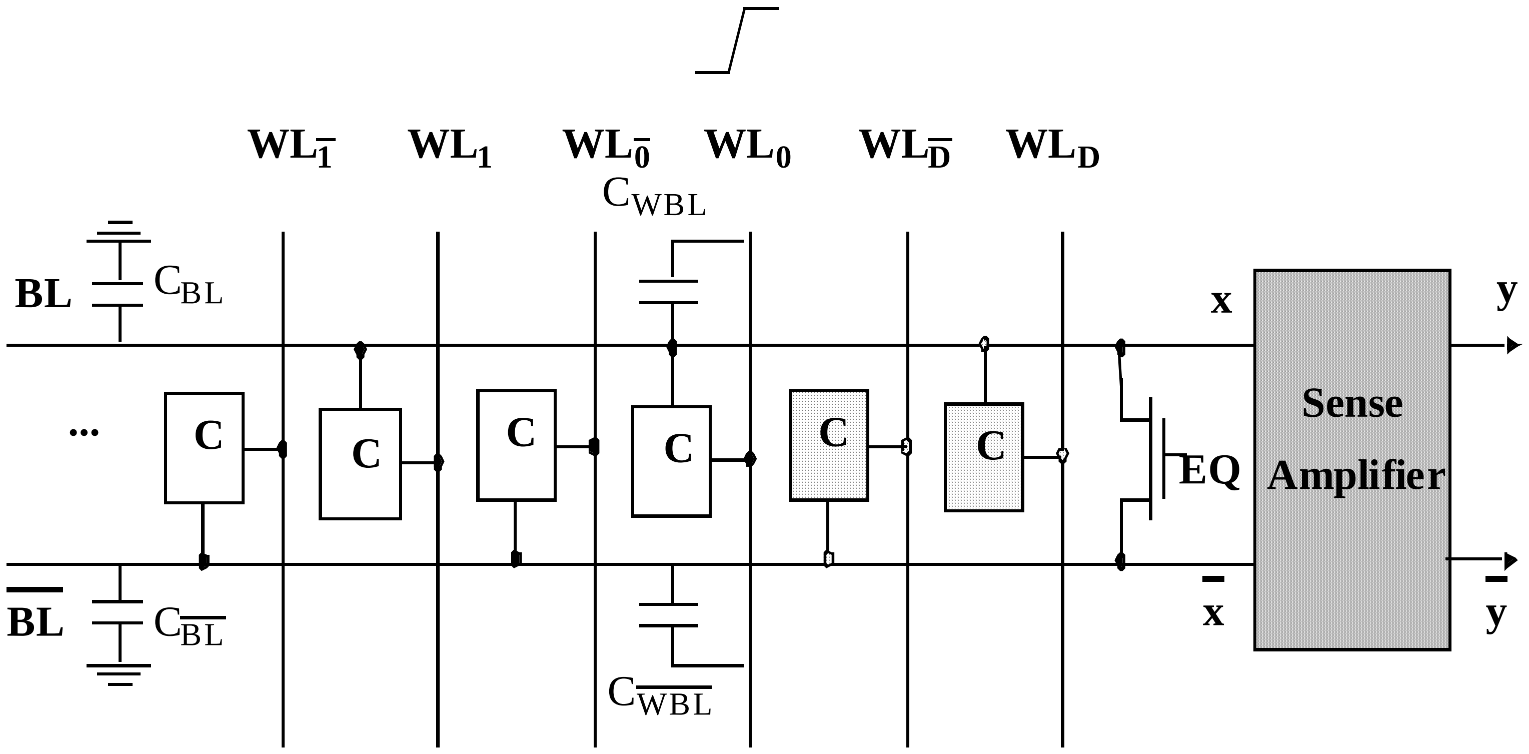
Address Transition Detection



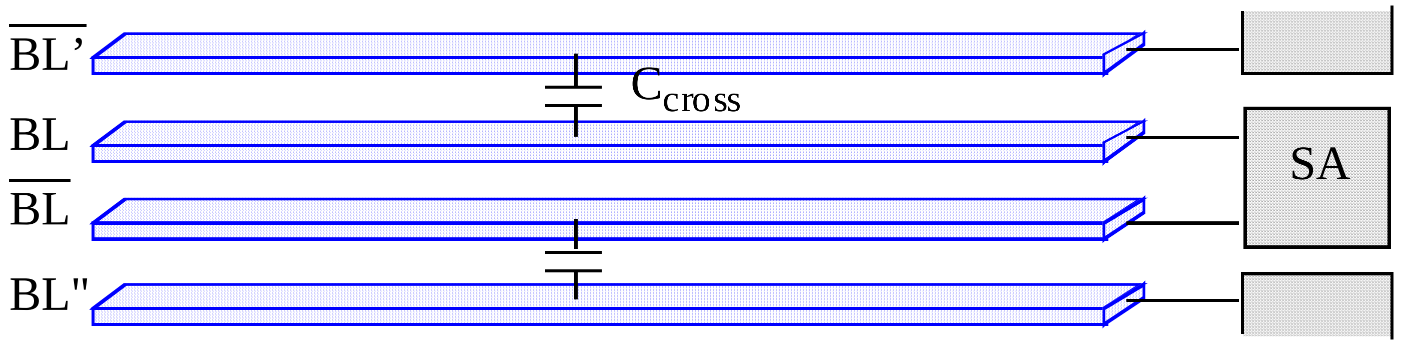
Open Bit-line Architecture —Cross Coupling



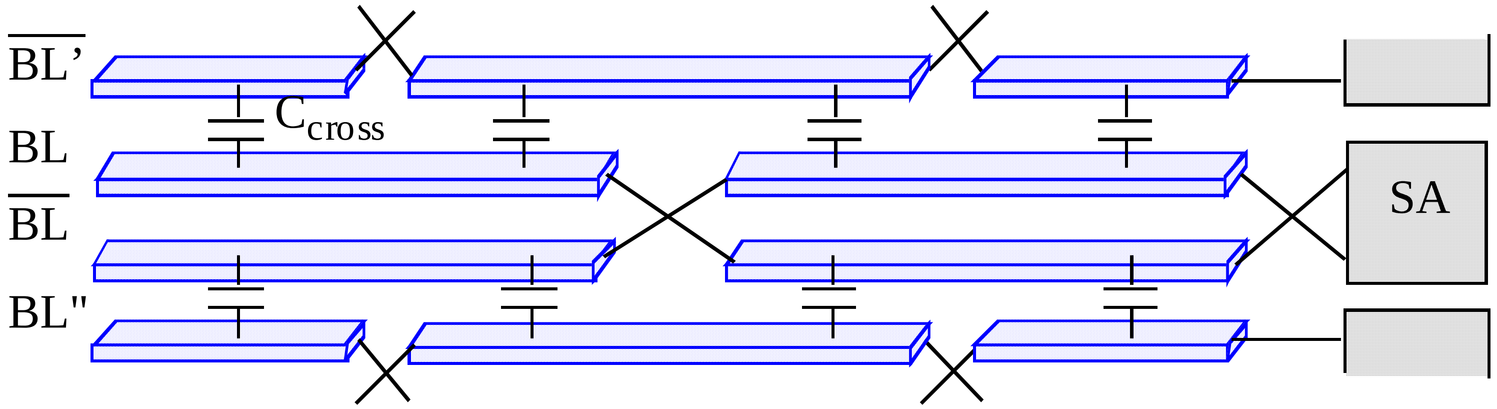
Folded-Bitline Architecture



Transposed-Bitline Architecture



(a) Straightforward bitline routing.



(b) Transposed bitline architecture.