EE582
Physical Design Automation of VLSI Circuits and Systems

Prof. Dae Hyun Kim
School of Electrical Engineering and Computer Science
Washington State University

Preliminaries

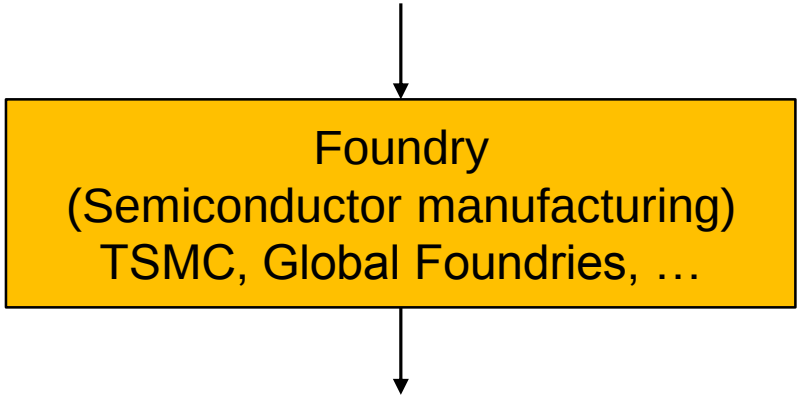
WASHINGTON STATE UNIVERSITY

Table of Contents

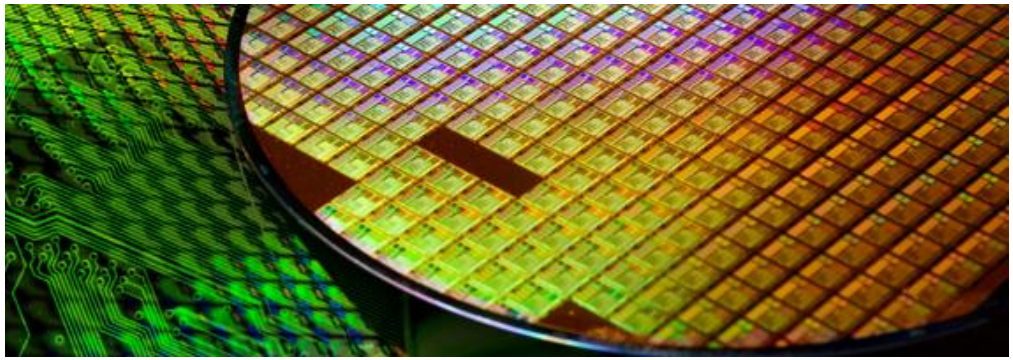
- Semiconductor manufacturing
- Problems to solve
- Algorithm complexity analysis

Semiconductor Manufacturing

Layout (GDSII stream format)

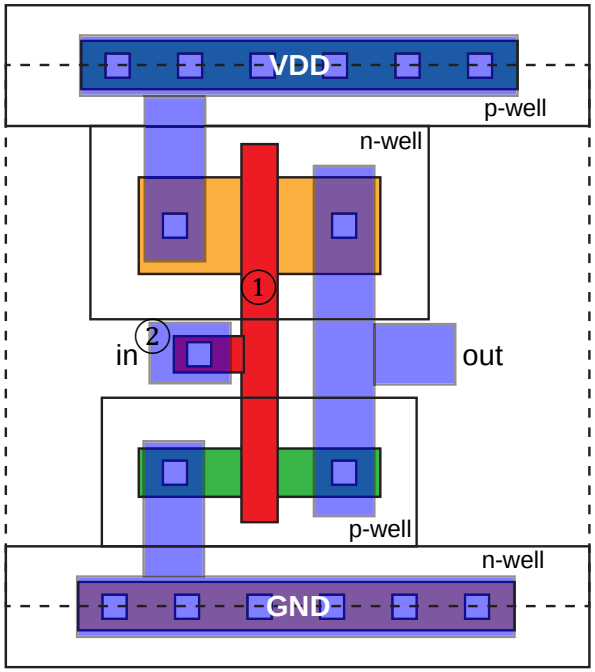


Bare dies



Semiconductor Manufacturing

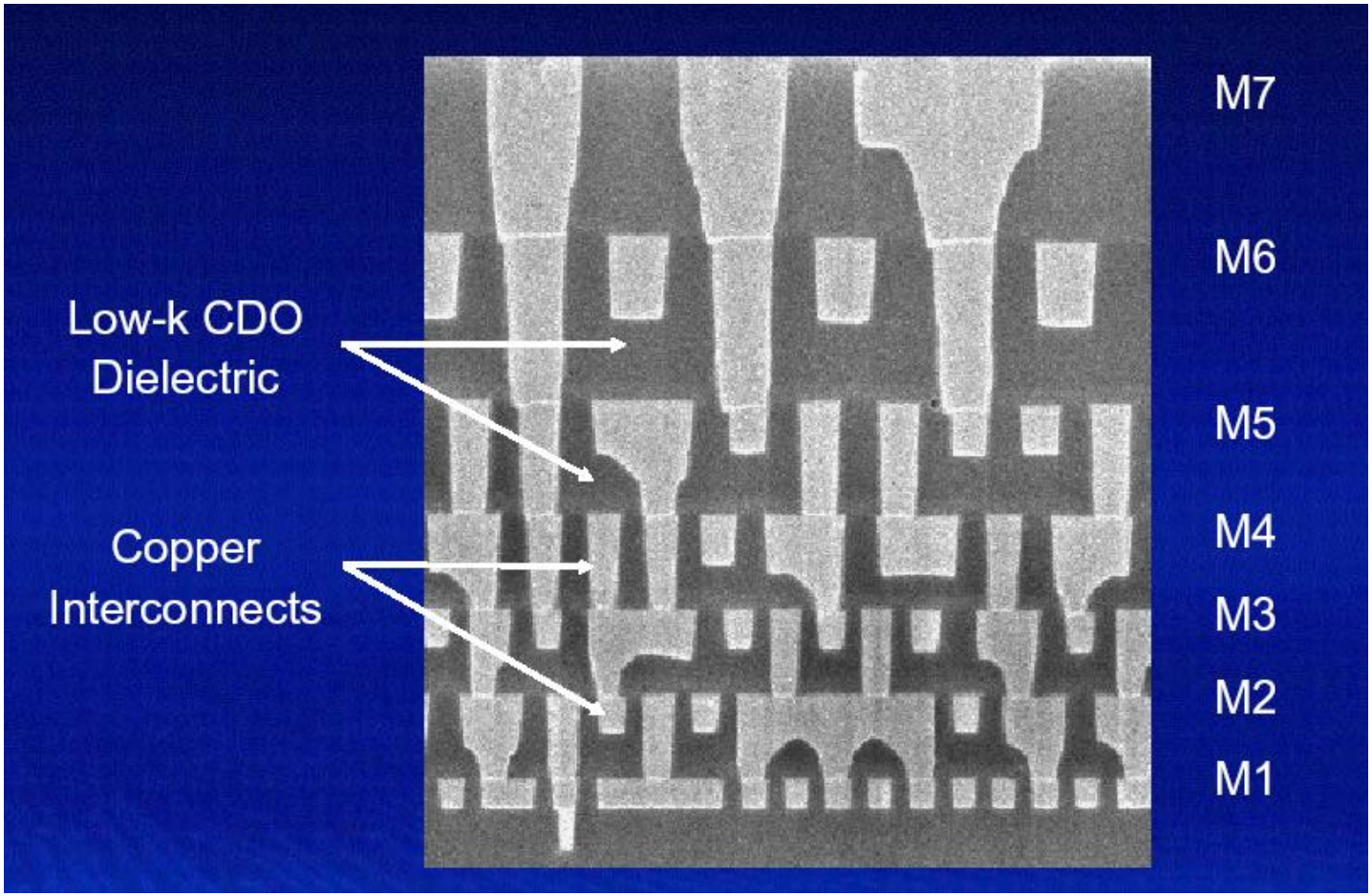
- Input
  - Layout (GDSII stream format)
    - A set of geometric objects



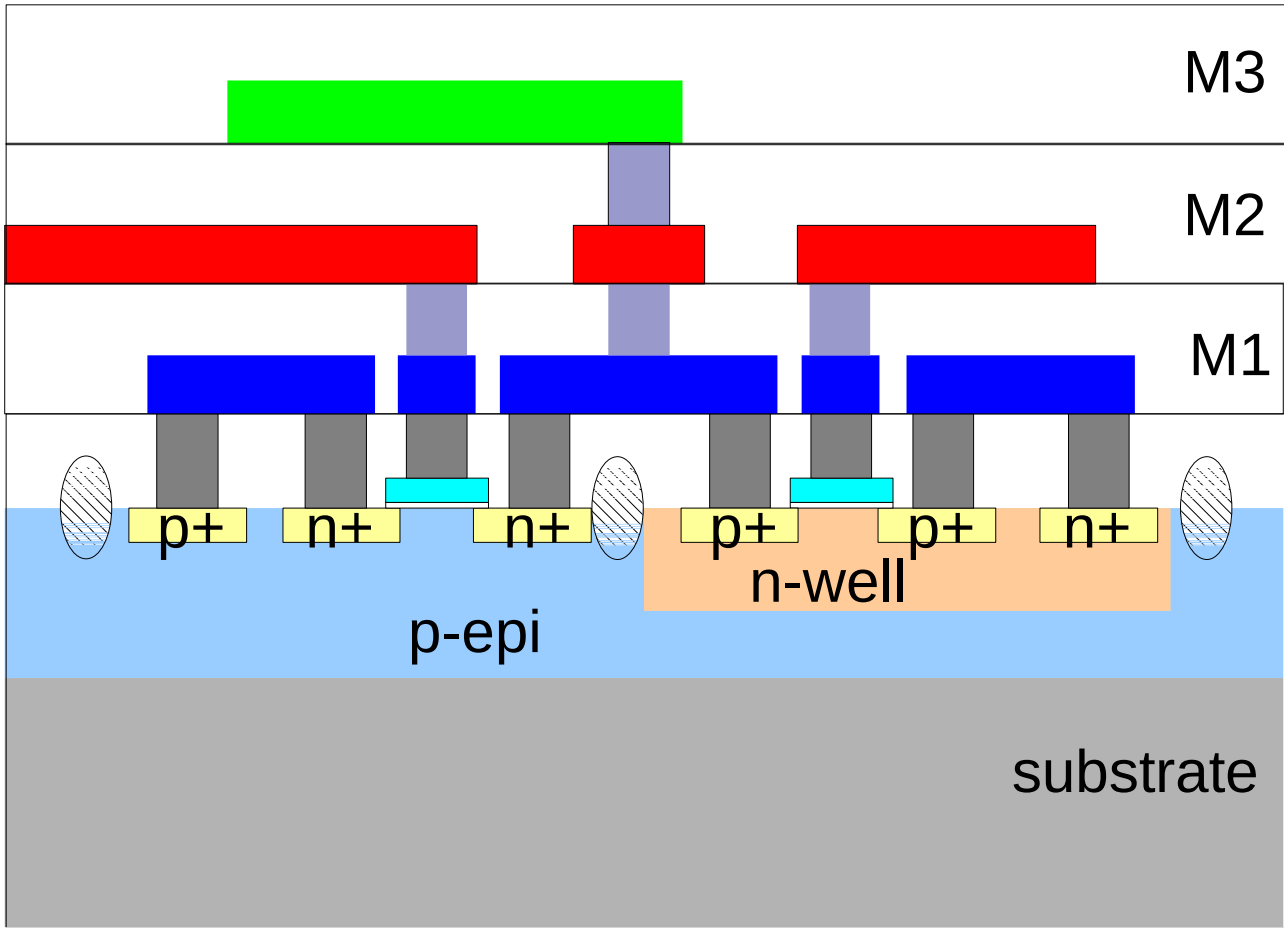
①: Layer id 3, polygon { 50, 40, 70, 40, 70, 220, 50, 220, 50, 140, 20, 140, 20, 110, 50, 110, 50, 40 }

②: Layer id 7, rectangle { 10, 105, 40, 150 }

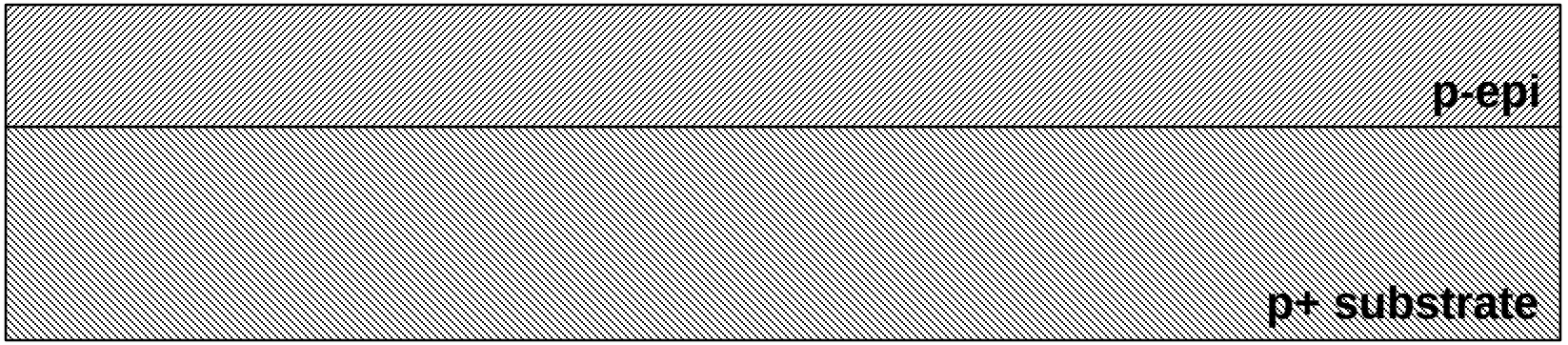
Semiconductor Manufacturing



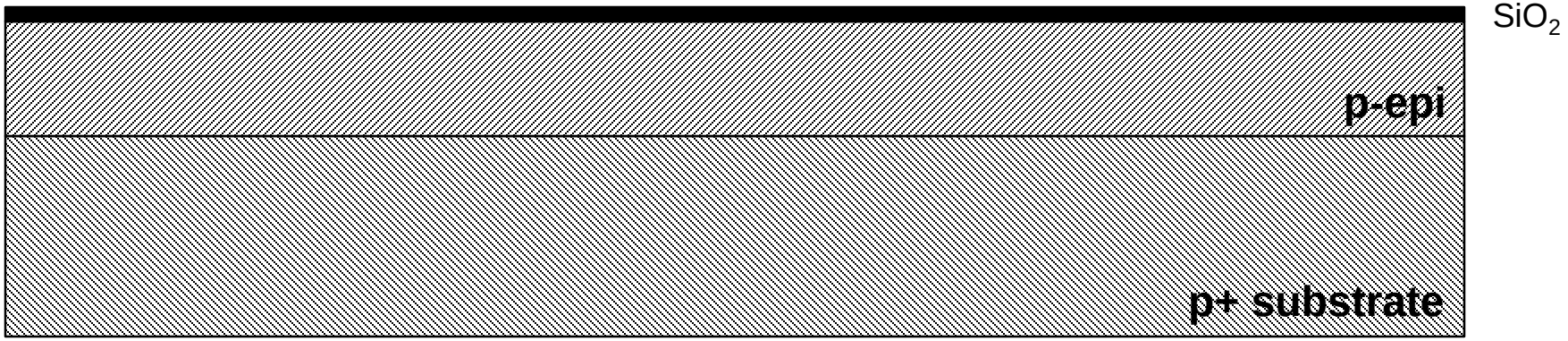
Semiconductor Manufacturing



Semiconductor Manufacturing

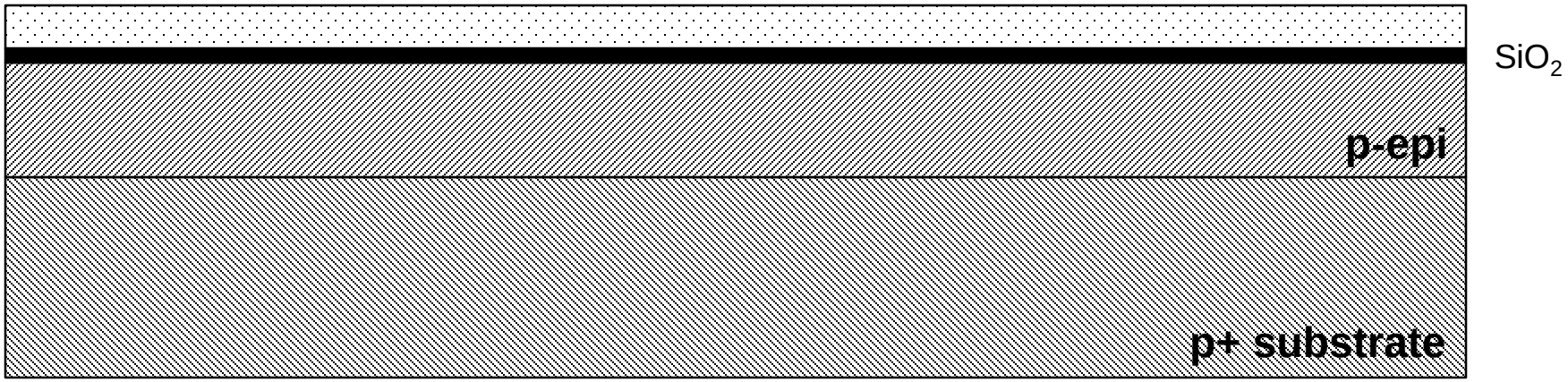


Semiconductor Manufacturing

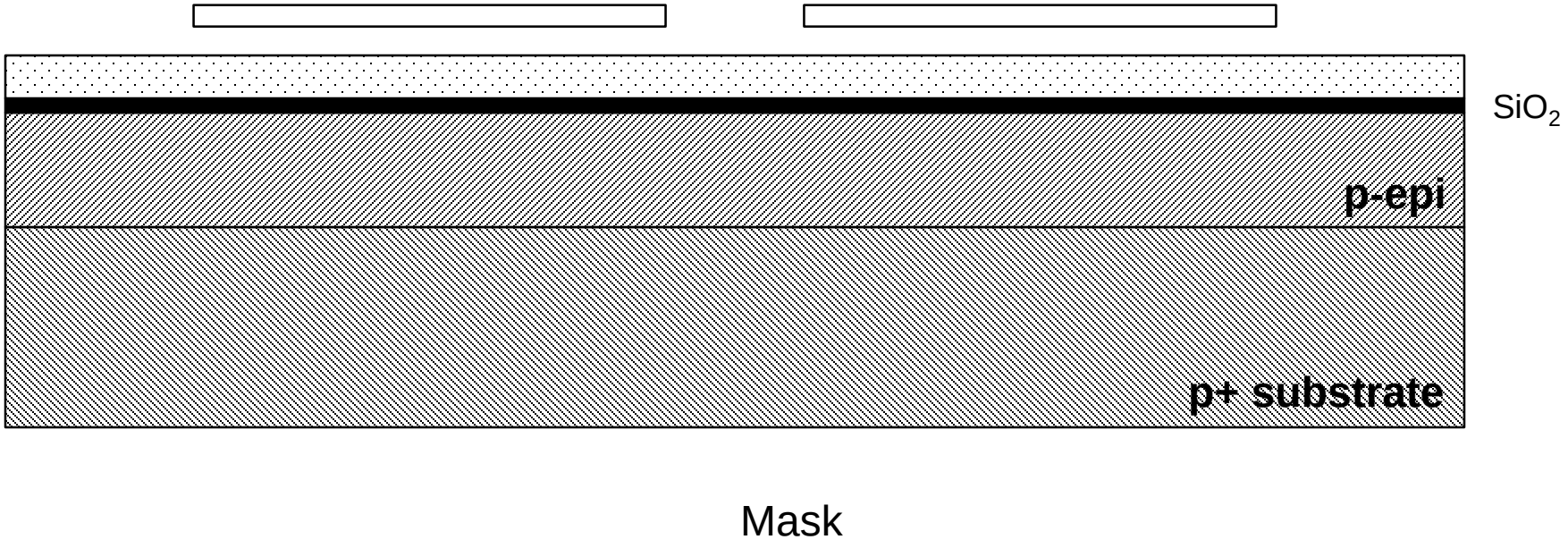


Gate-oxide deposition

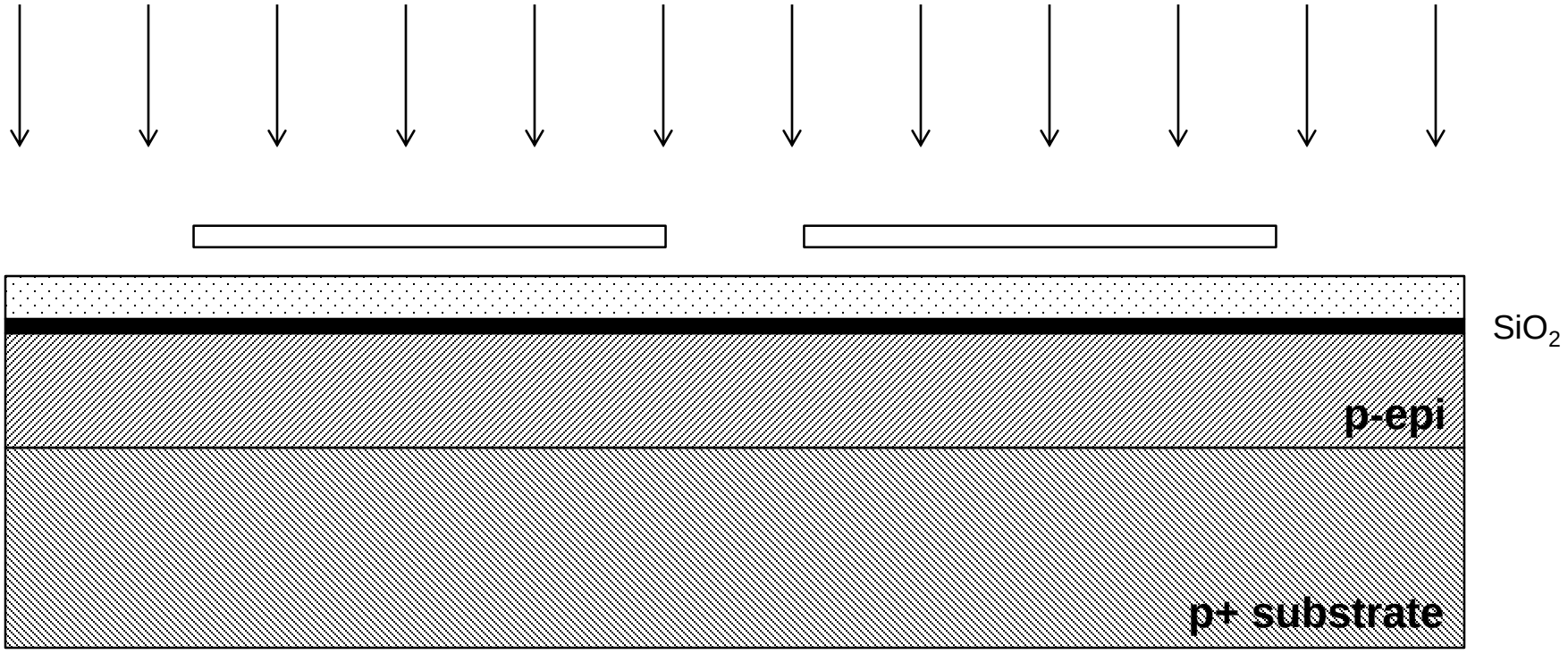
Semiconductor Manufacturing



Semiconductor Manufacturing

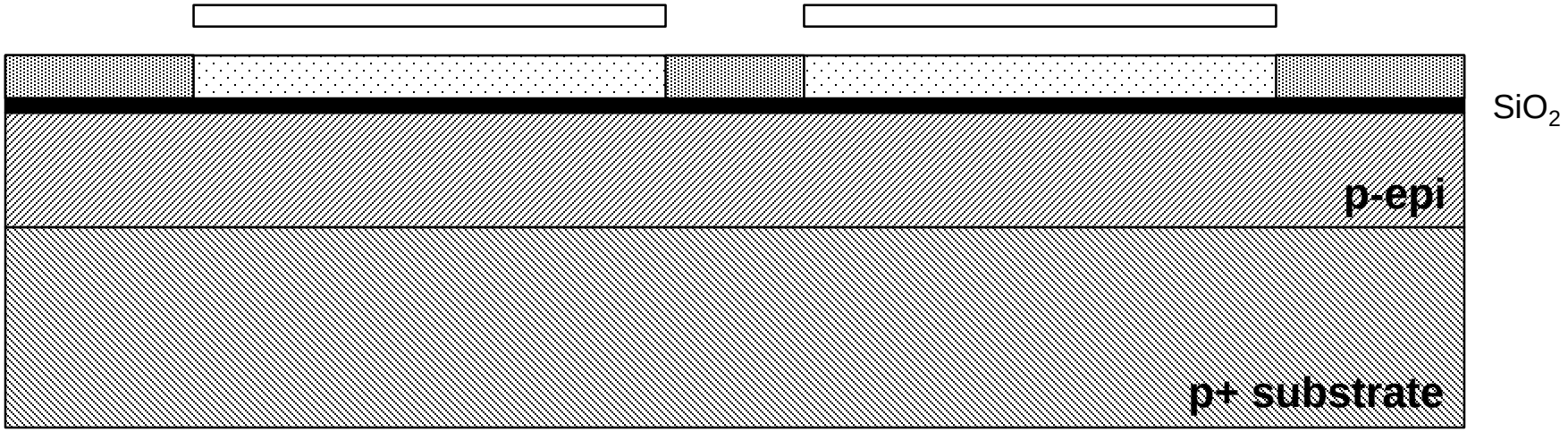


Semiconductor Manufacturing



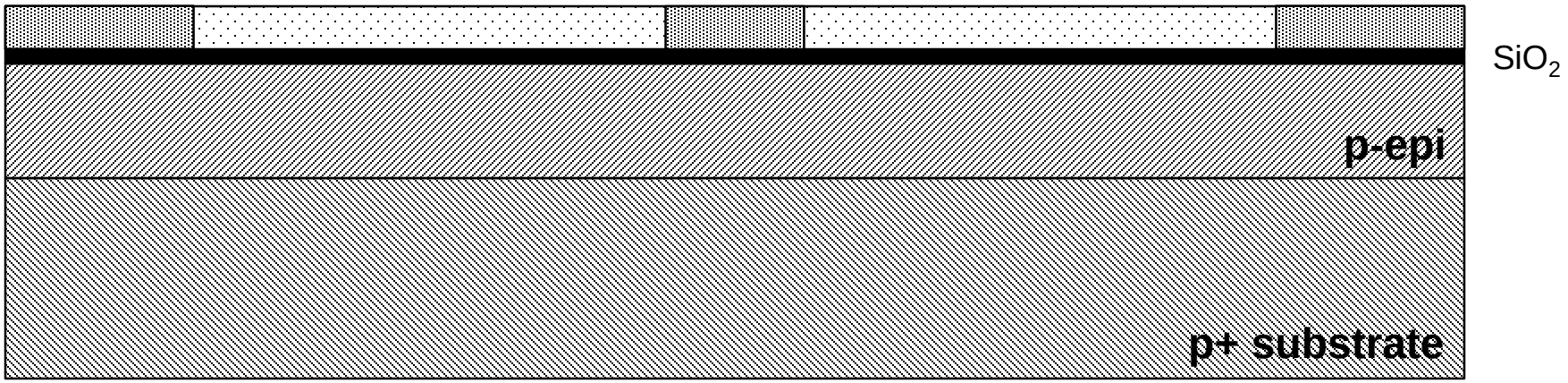
Expose (photolithography)

Semiconductor Manufacturing



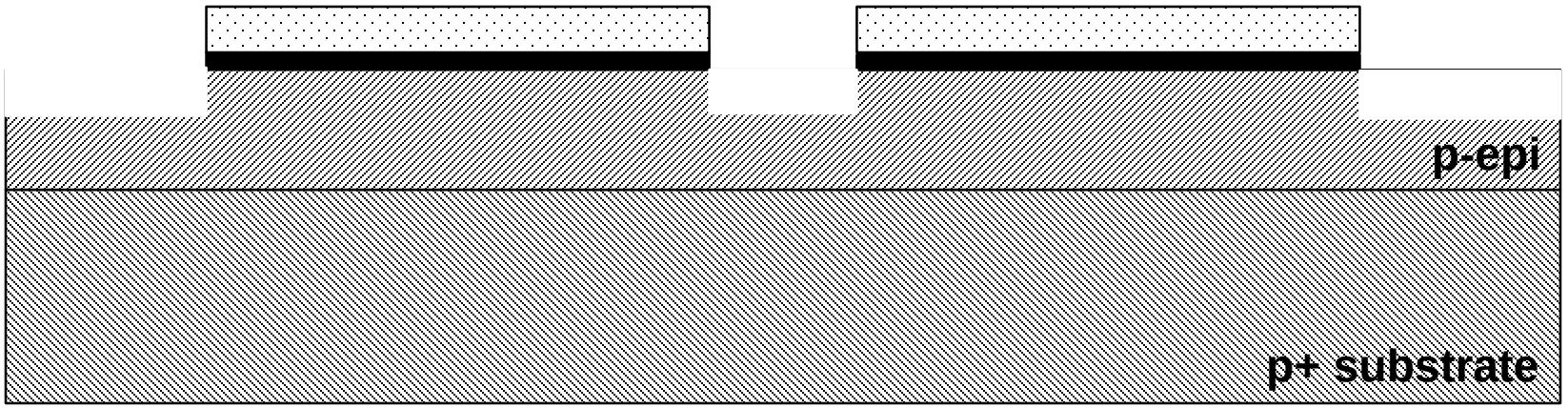
After photolithography

Semiconductor Manufacturing



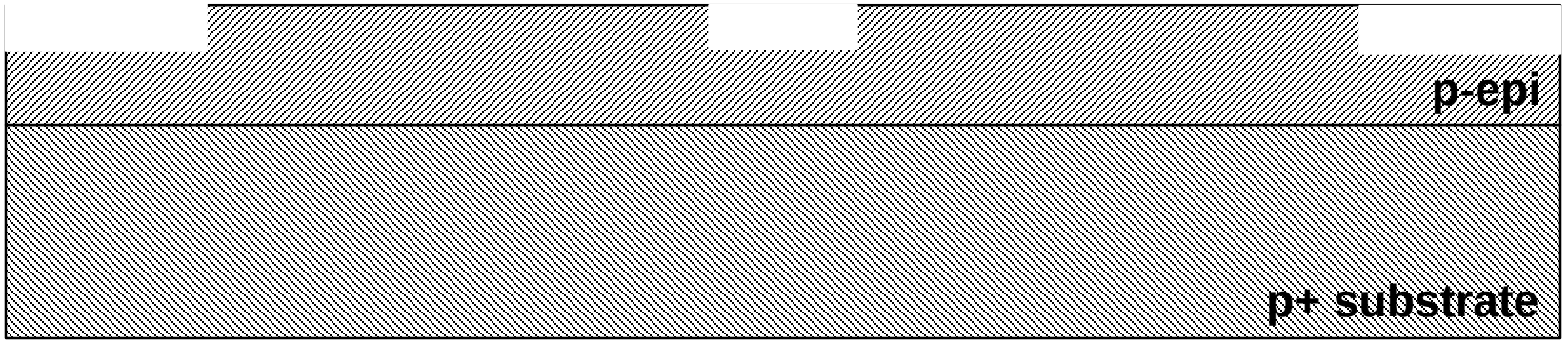
Remove mask

Semiconductor Manufacturing



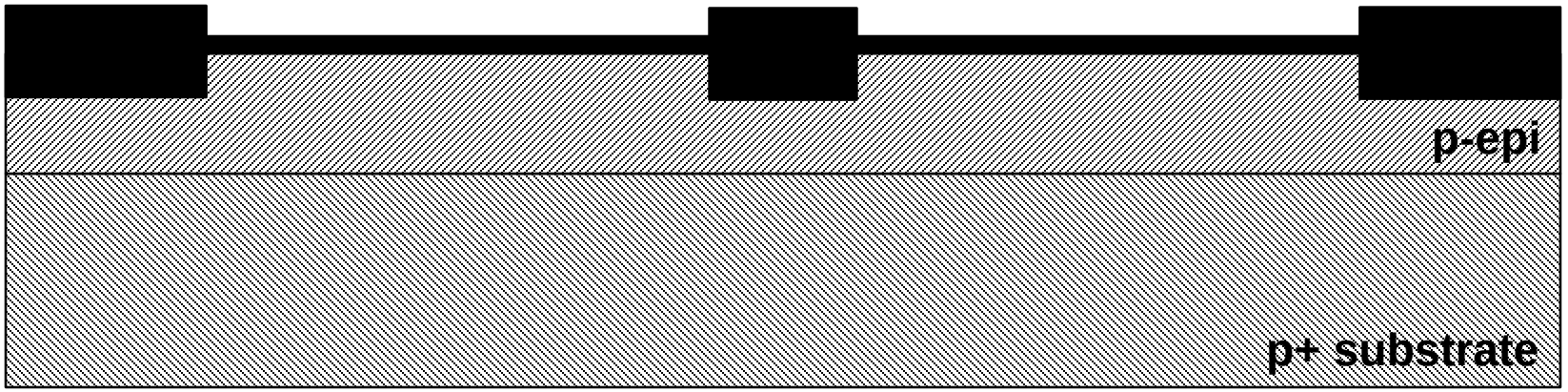
Etching

Semiconductor Manufacturing



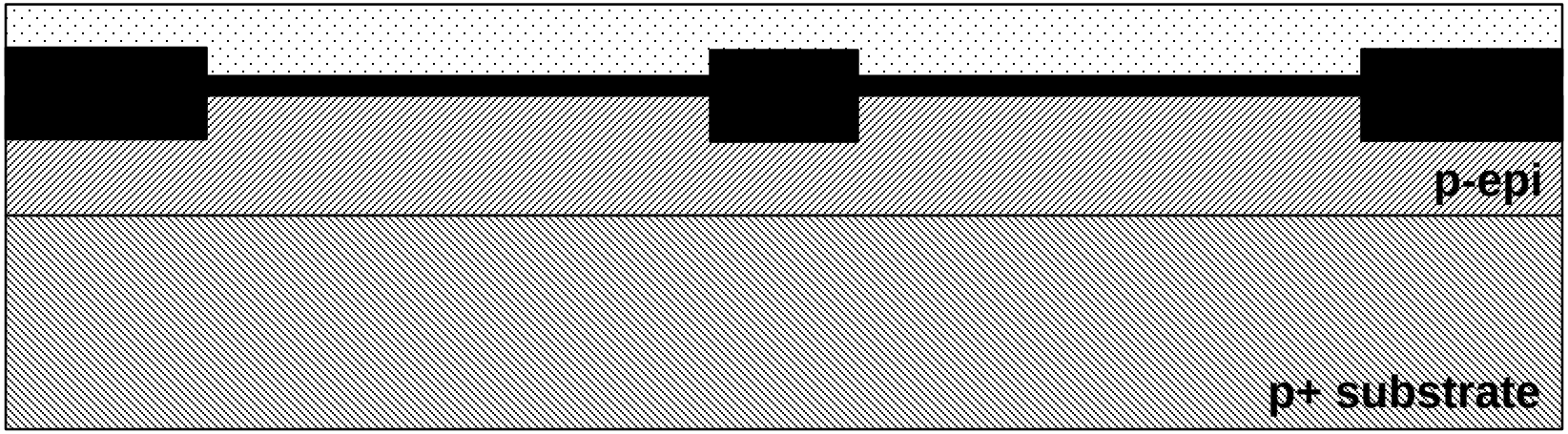
Etching

Semiconductor Manufacturing



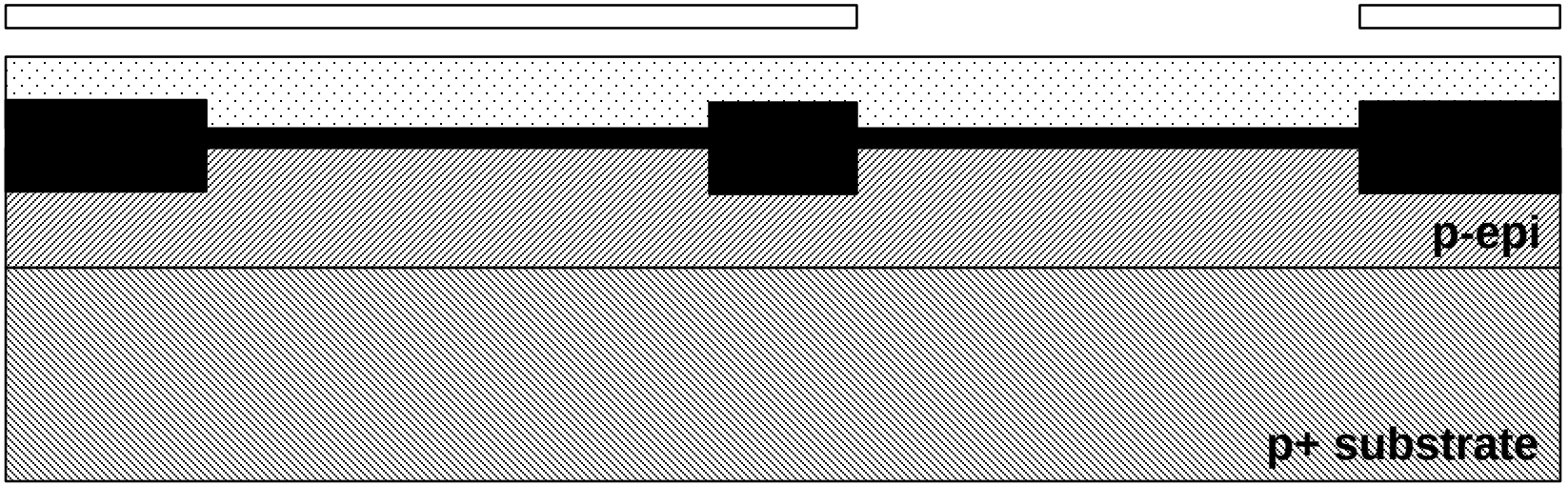
Oxide deposition

Semiconductor Manufacturing



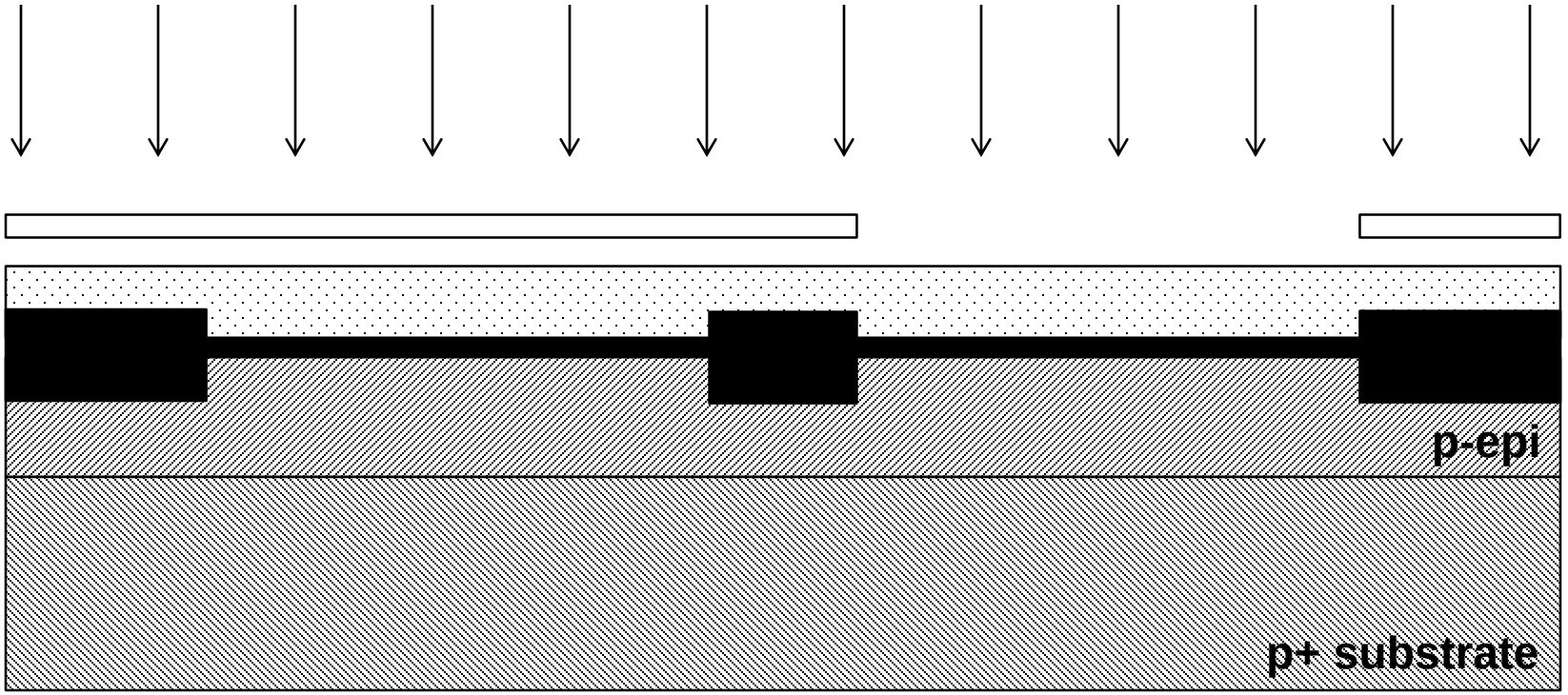
Photoresist

Semiconductor Manufacturing



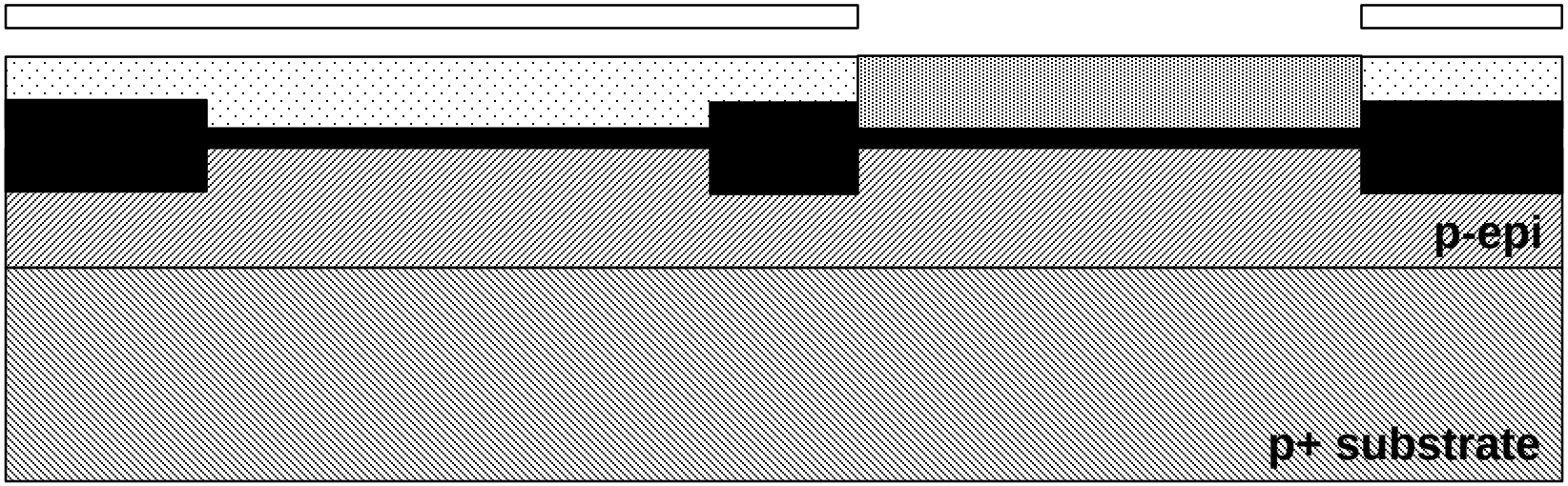
Mask

Semiconductor Manufacturing



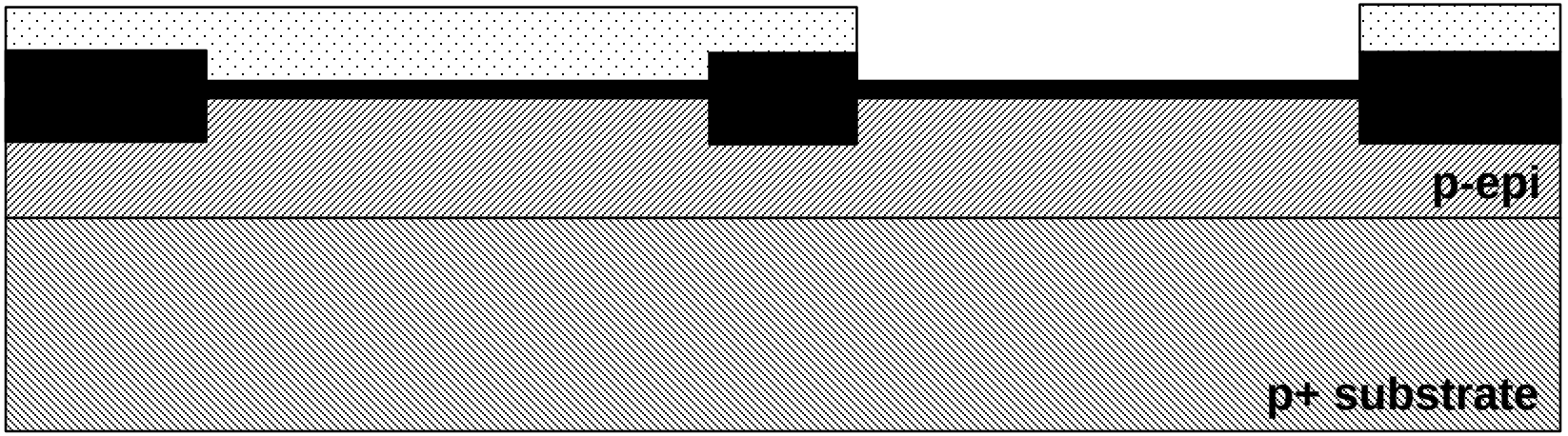
Photolithography

Semiconductor Manufacturing



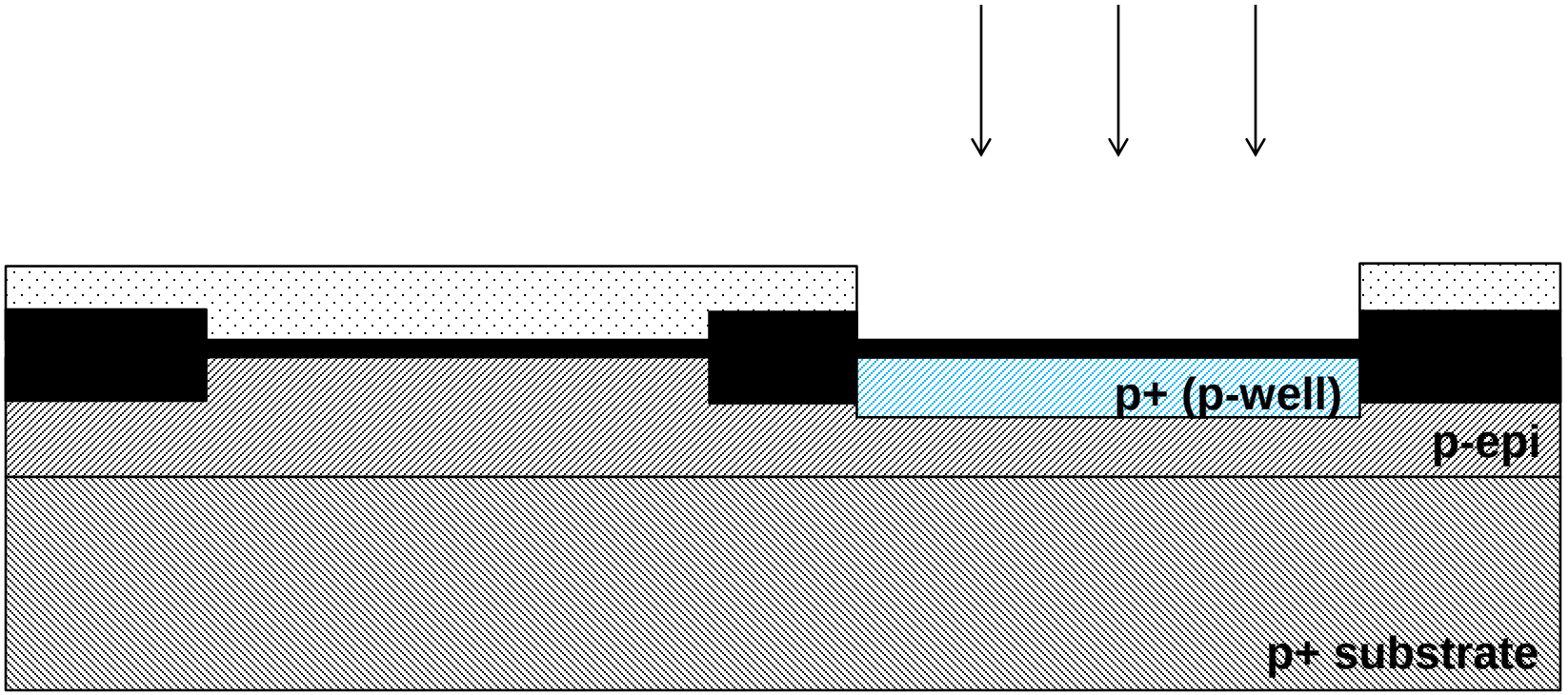
After photolithography

Semiconductor Manufacturing



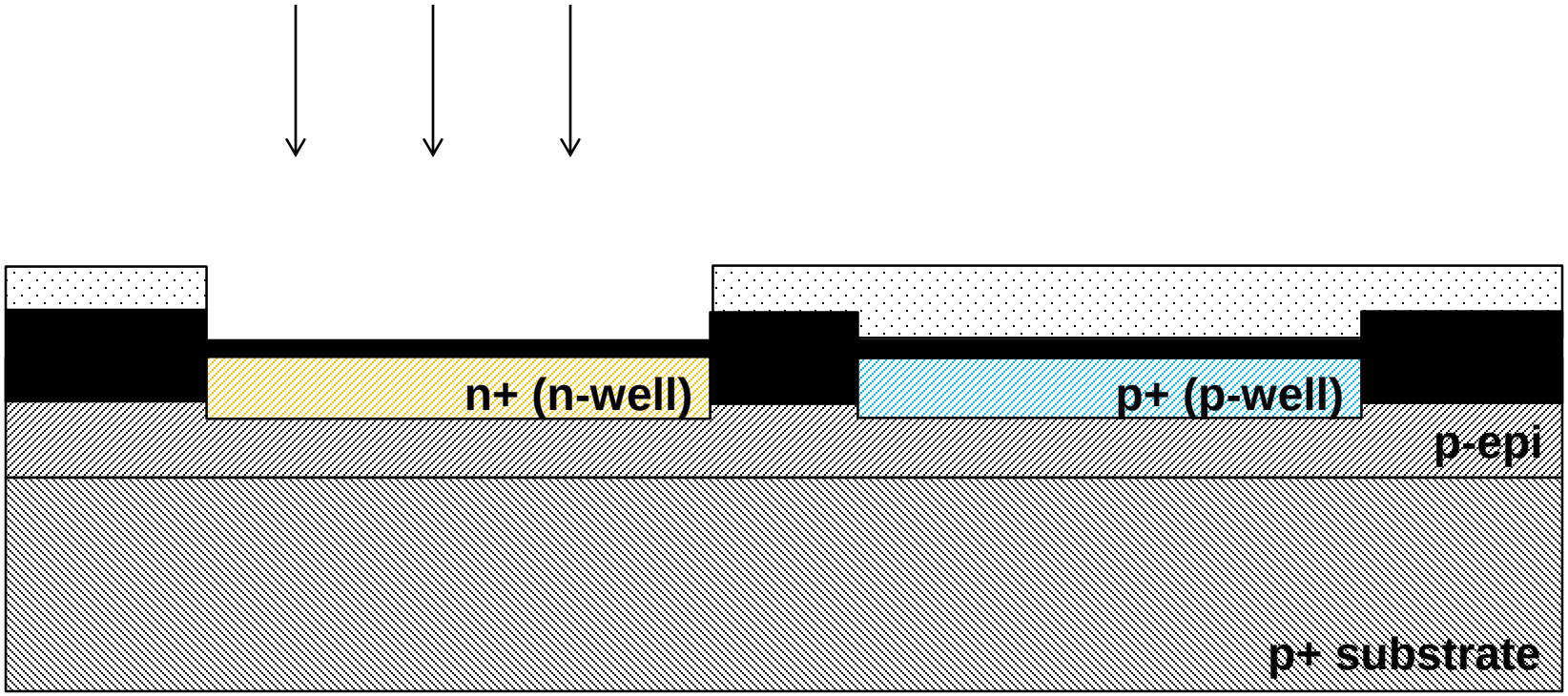
Etch

Semiconductor Manufacturing



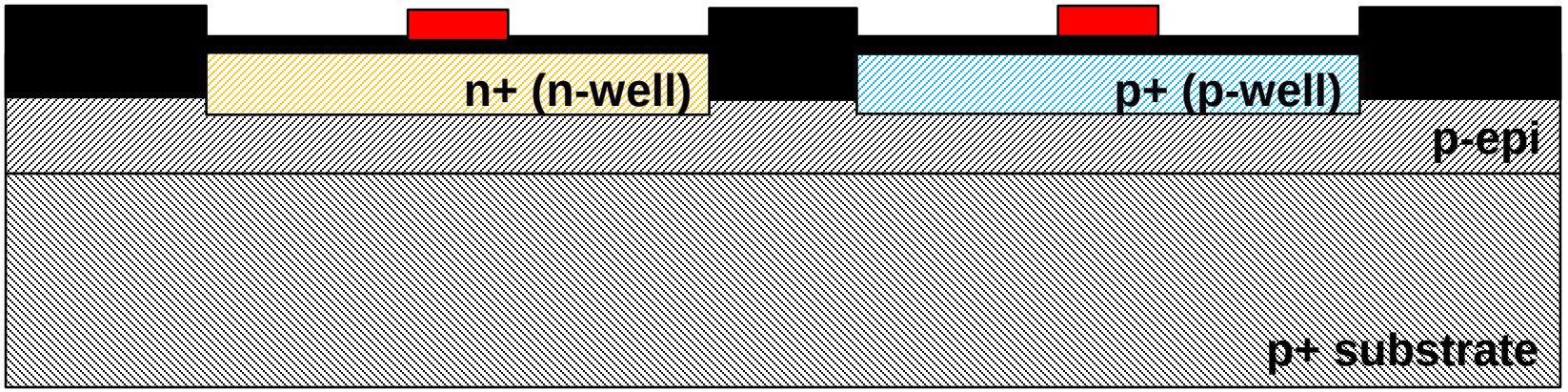
Doping

Semiconductor Manufacturing



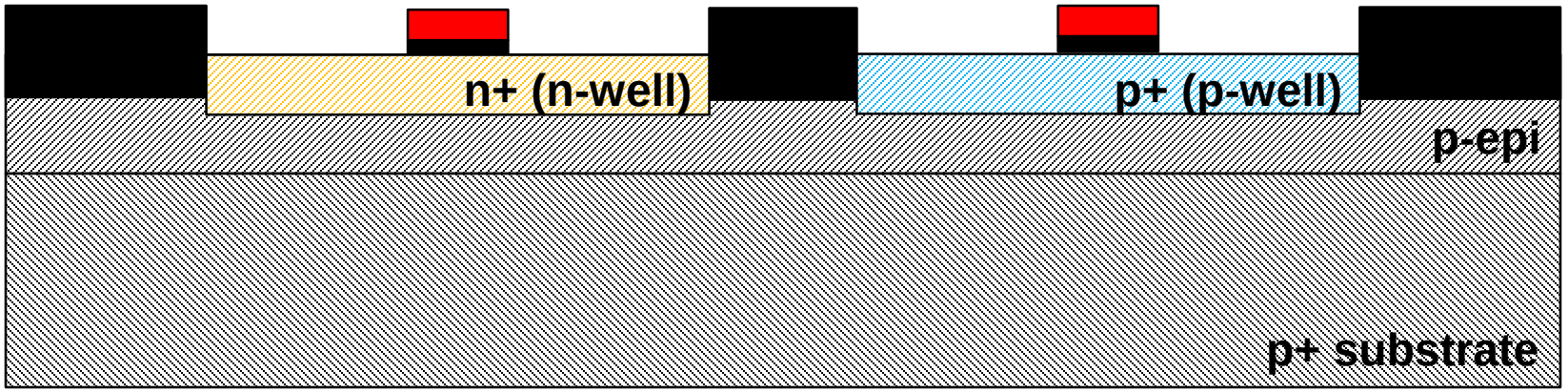
Doping

Semiconductor Manufacturing



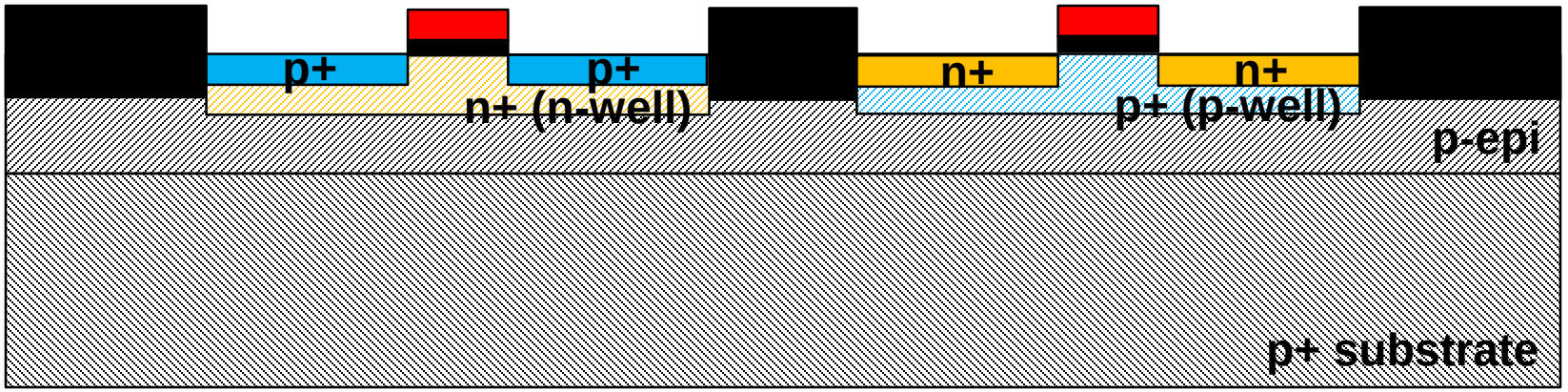
Poly

Semiconductor Manufacturing



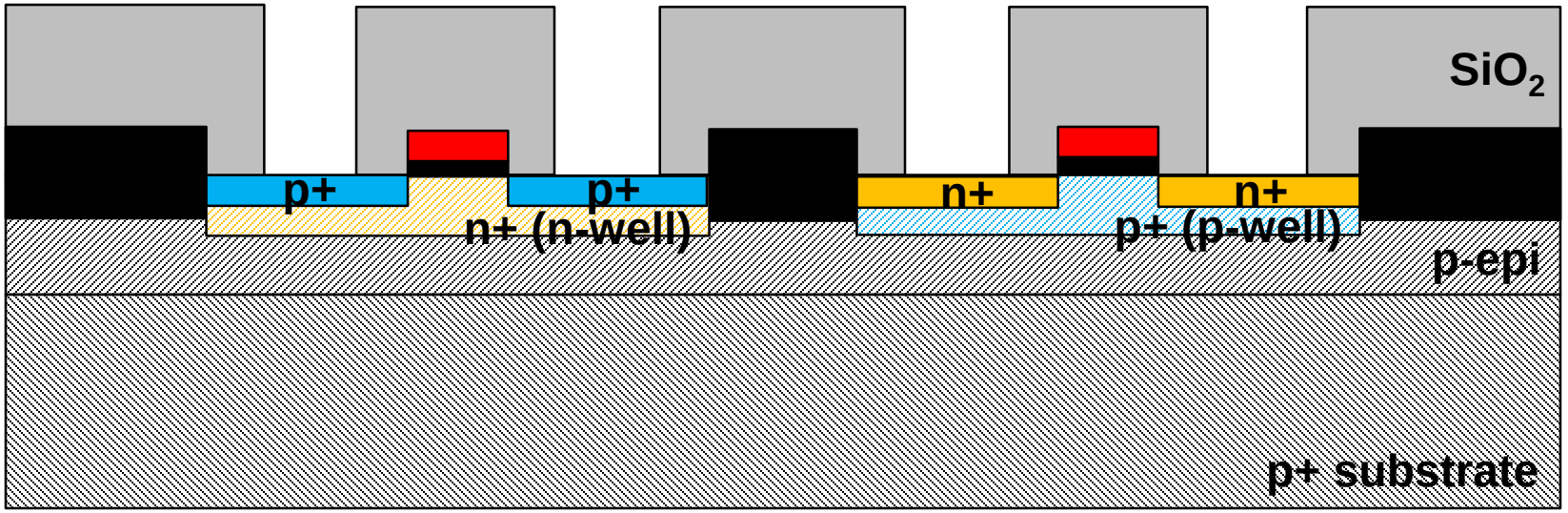
Etch

Semiconductor Manufacturing



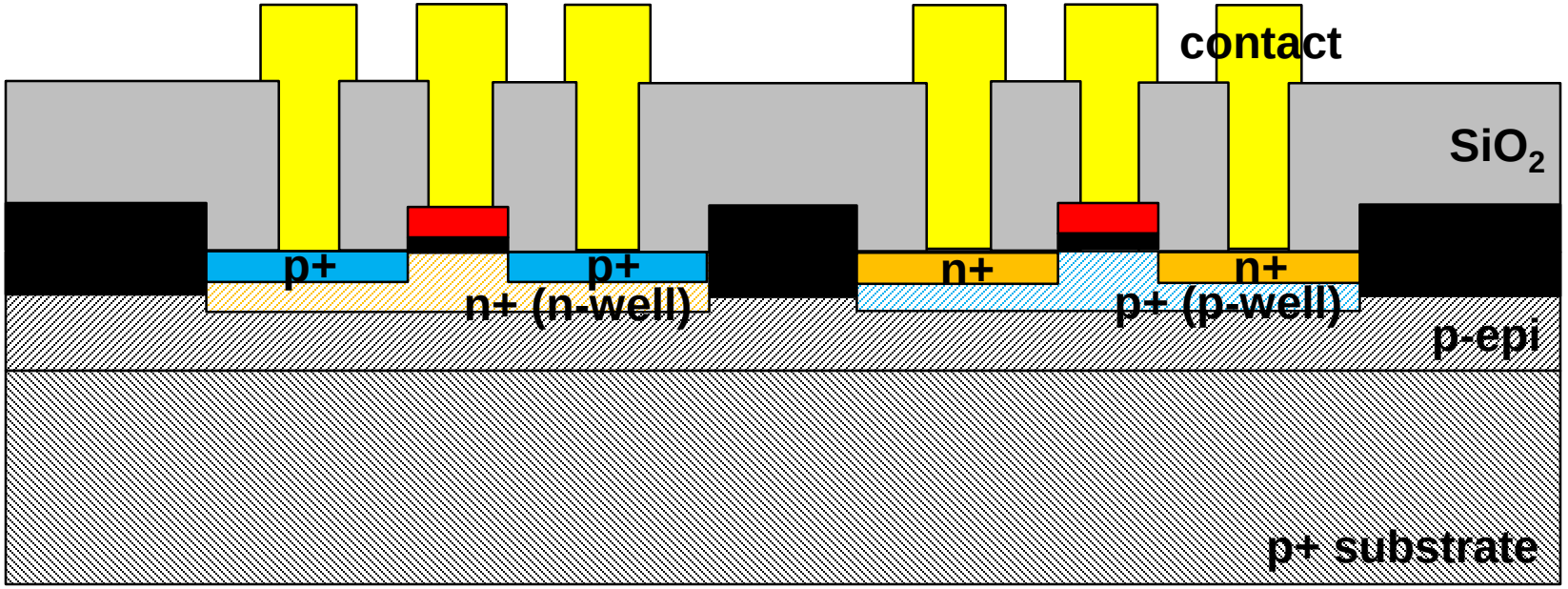
Doping

Semiconductor Manufacturing



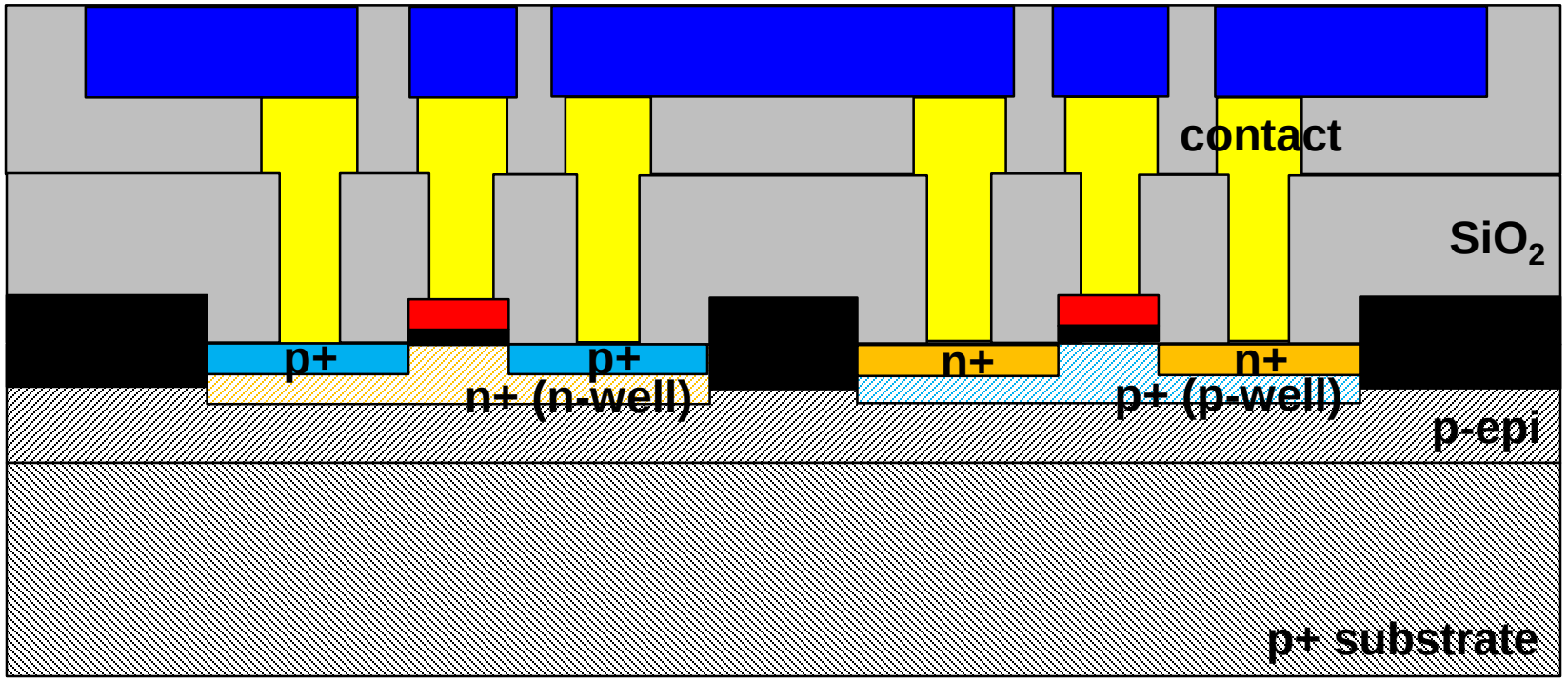
Oxide deposition

Semiconductor Manufacturing



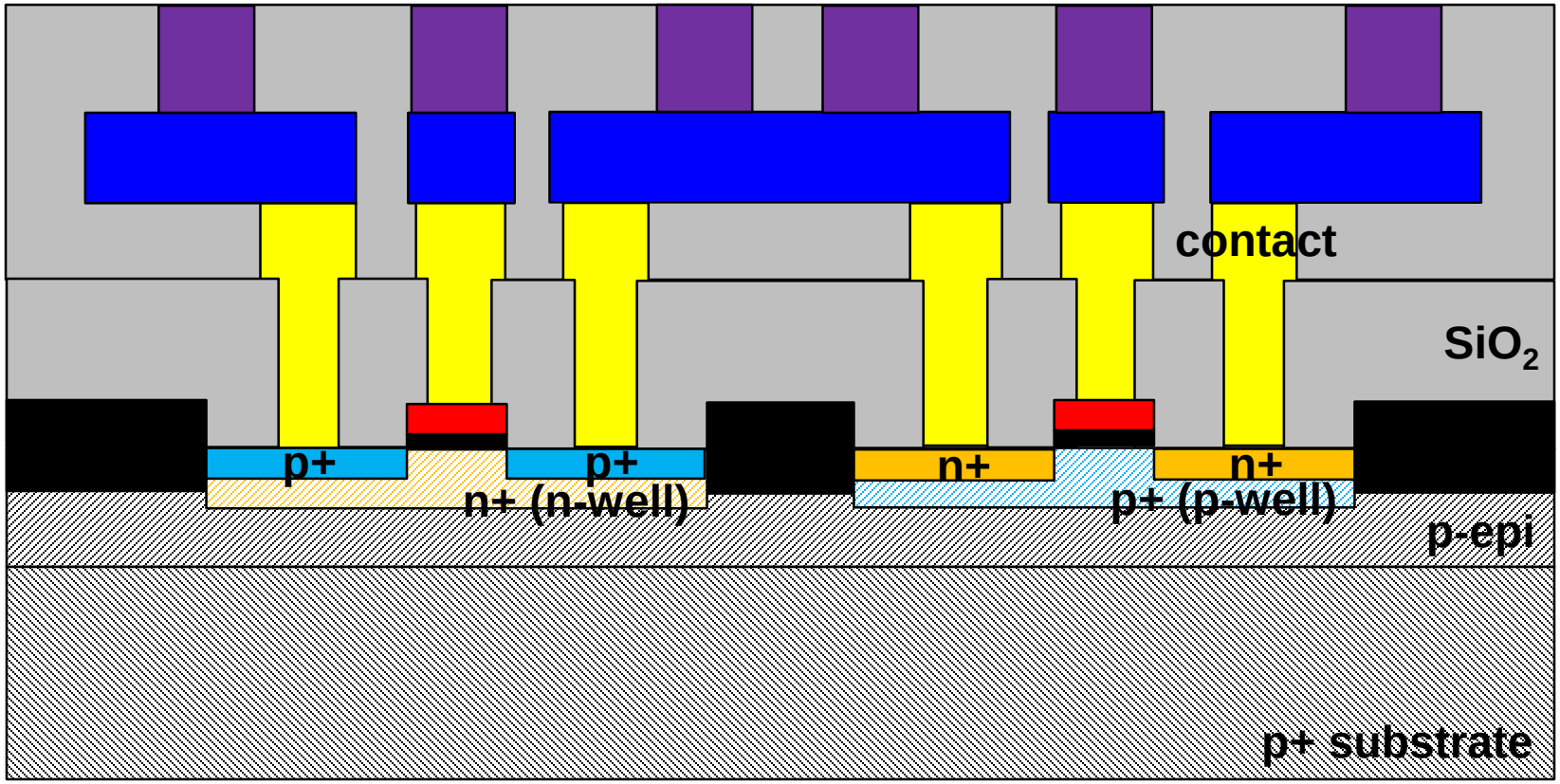
Contact

Semiconductor Manufacturing



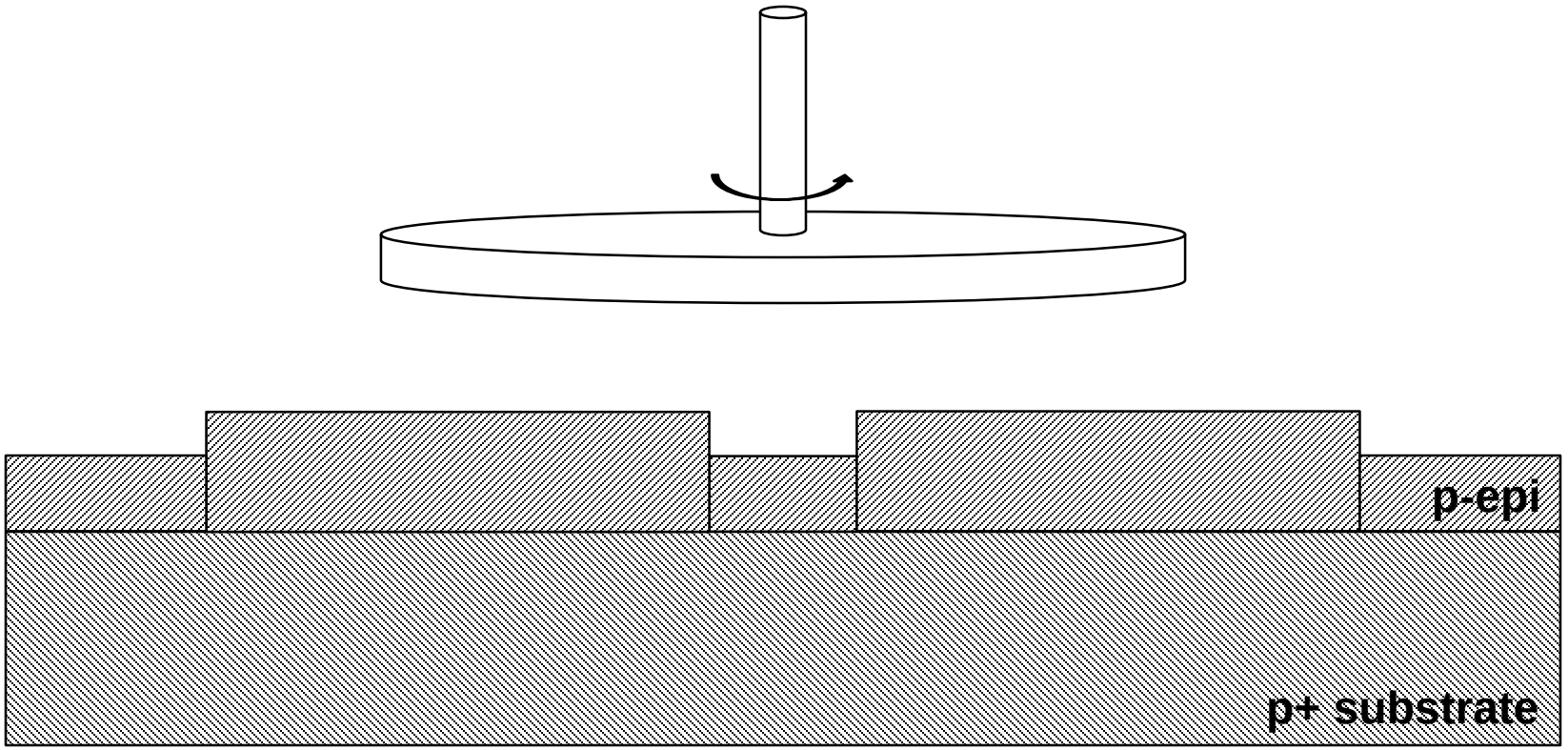
Metal 1

Semiconductor Manufacturing



Via12

Semiconductor Manufacturing

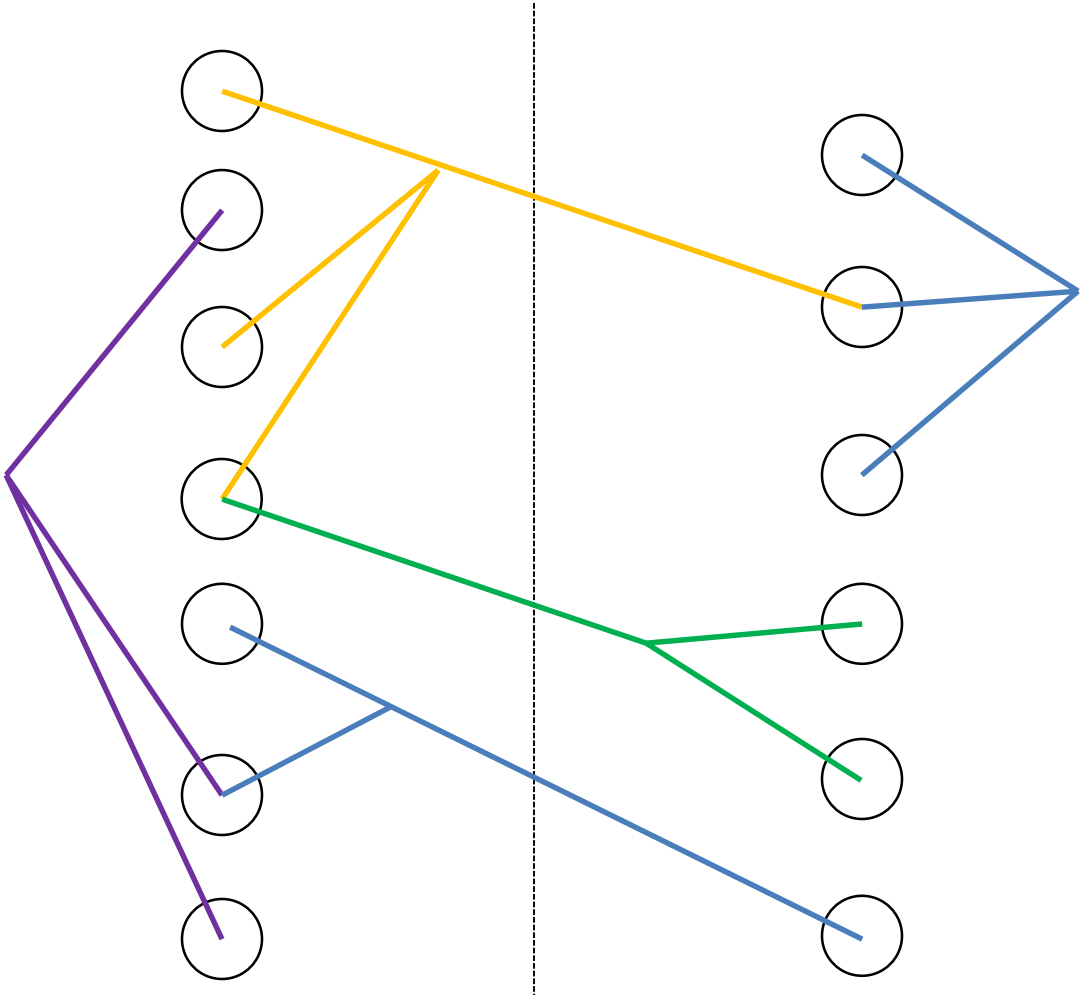


Chemical-mechanical-polishing (CMP)

Problems – Partitioning

- You are given
  - A set of modules
    - M1 (area: 10), M2 (area:20), …
  - Netlist
    - N1 (M1, M2), N2 (M2, M3, M4), …
- Find *k* partitions
  - Satisfy the following constraints:
    - $S_{min} \leq Size(P_k) \leq S_{max}$
  - Minimize
    - Cutsize

Problems – Partitioning

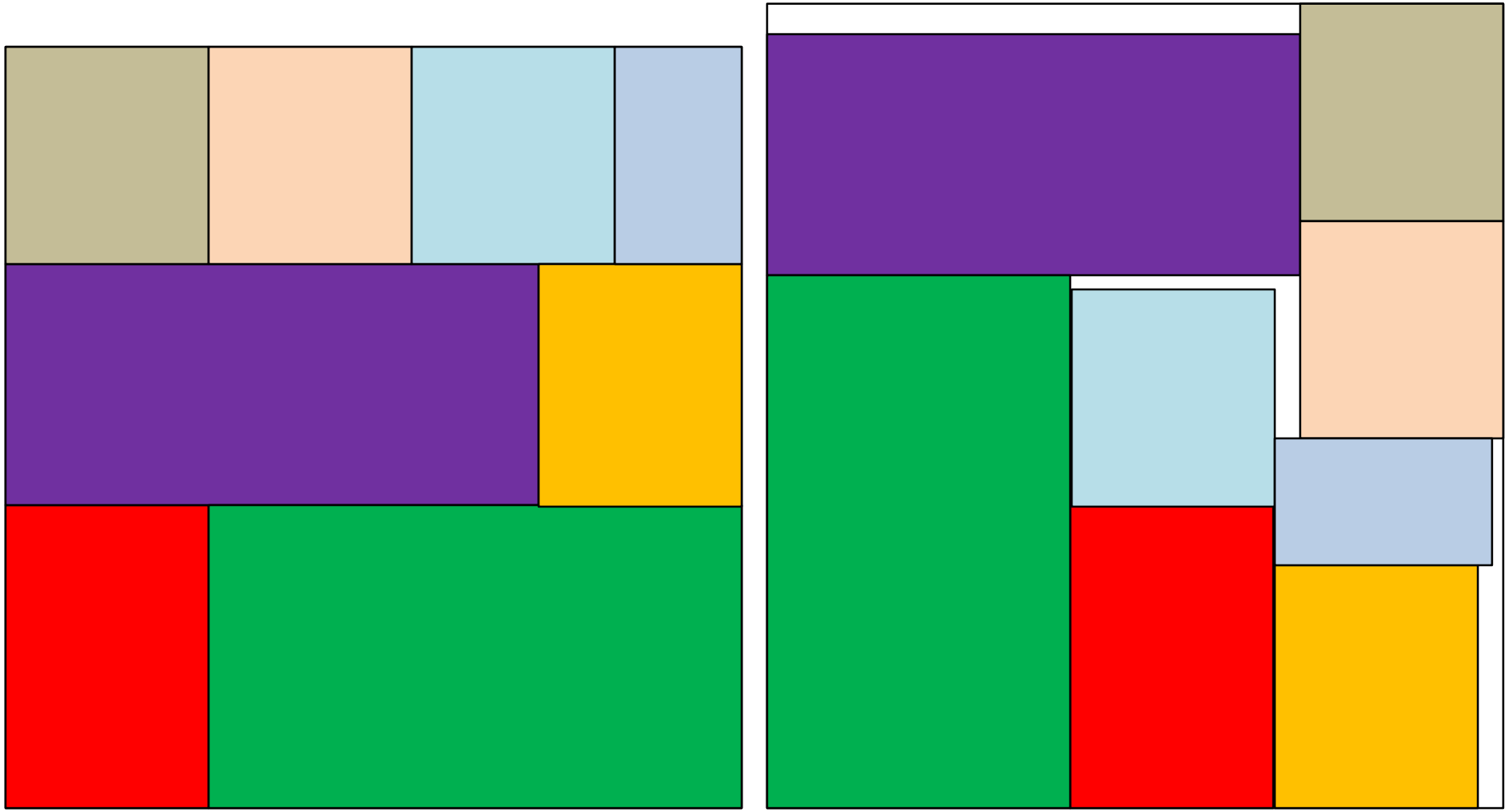


Problems – Floorplanning

- You are given
  - A set of modules
    - M1 (w: 10, h: 20), M2 (w: 30, h: 10), …
  - Netlist
    - N1 (M1, M2), N2 (M2, M3, M4), …

- Find a floorplan
  - Minimize
    - Area
    - Wirelength

Problems – Floorplanning



Problems – Floorplanning

- More complex floorplanning
  - Some modules are rotatable.
    - M1, M4, M6, …
  - Some modules are soft.
    - M2 (w_{min}: 10, w_{max}: 20, area: 150), …
  - Some modules have fixed locations.
  - The outline is fixed (fixed-outline floorplanning)
  - Find a floorplan
    - Minimize
      - Area
      - Wirelength

Problems – Floorplanning

- More complex floorplanning
  - Each module also has power profile.
  - Each net has access frequency profile.
  - Find a floorplan
    - Minimize
      - Area
      - Interconnect power
      - Worst-case temperature
    - Maximize
      - Performance
    - Achieve
      - 100% routability

Problems – Placement

- Global placement
  - You are given
    - A set of standard cells (and their dimensions)
      - C1 (NAND2_X4), C2 (FA_X4), …
    - Netlist
      - N1 (C1/Z, C3/A, C4/A, C5/B), N2 (C2/S, C6/A), …
    - Core area
      - Width: 1,000um, Height: 1,000um
  - Find the (non-legal) locations of the cells
    - Satisfy
      - Density (W_j) ≤ D_{max} (e.g., 70%)
    - Minimize
      - Total wirelength

Problems – Placement

- More complex global placement
  - You are also given
    - Timing libraries
  - Find (non-legal) locations of the cells
    - Satisfy
      - Density (W_j) ≤ D_{max} (e.g., 70%)
    - Minimize
      - Worst net delay
      - Worst path delay

Problems – Placement

- More complex global placement
  - You are also given
    - Routing resources
  - Find (non-legal) locations of the cells
    - Satisfy
      - Density (W_j) ≤ D_{max} (e.g., 70%)
    - Maximize
      - Routability

Problems – Placement

- Legalization
  - You are given
    - A set of standard cells (and their dimensions)
      - C1 (NAND2_X4), C2 (FA_X4), …
    - Netlist
      - N1 (C1/Z, C3/A, C4/A, C5/B), N2 (C2/S, C6/A), …
    - Core area
      - Width: 1,000um, Height: 1,000um
    - Initial (non-legal) locations of the cells
      - C1 (100, 100), C2 (101, 100), …
  - Find the legal locations of the cells

Problems – Placement

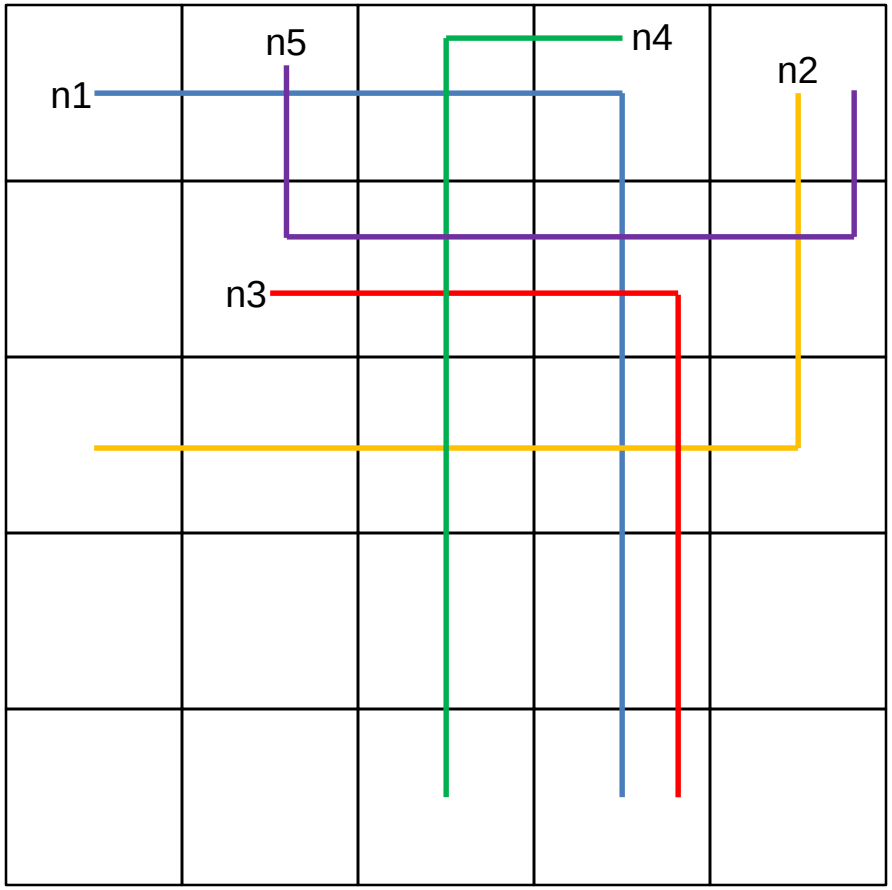
- Detailed placement
  - You are given
    - A set of standard cells (and their dimensions)
      - C1 (NAND2_X4), C2 (FA_X4), …
    - Netlist
      - N1 (C1/Z, C3/A, C4/A, C5/B), N2 (C2/S, C6/A), …
    - Core area
      - Width: 1,000um, Height: 1,000um
    - Initial (legal) locations of the cells
      - C1 (100, 100), C2 (105, 100), …
  - Optimize
    - Total wirelength
    - Routability
    - Design-rule violations

Problems – Routing

- Global routing
  - You are given
    - A set of pin locations
      - P1 (50, 100), P2 (55, 150), …
    - Netlist
      - N1 (P1, P4, P6), N2 (P2, P3), …
    - Core area
      - Width: 1,000um, Height: 1,000um
    - Detailed placement result
  - Find a global routing solution
    - Minimize
      - Total wirelength
    - Minimize runtime

Problems – Routing

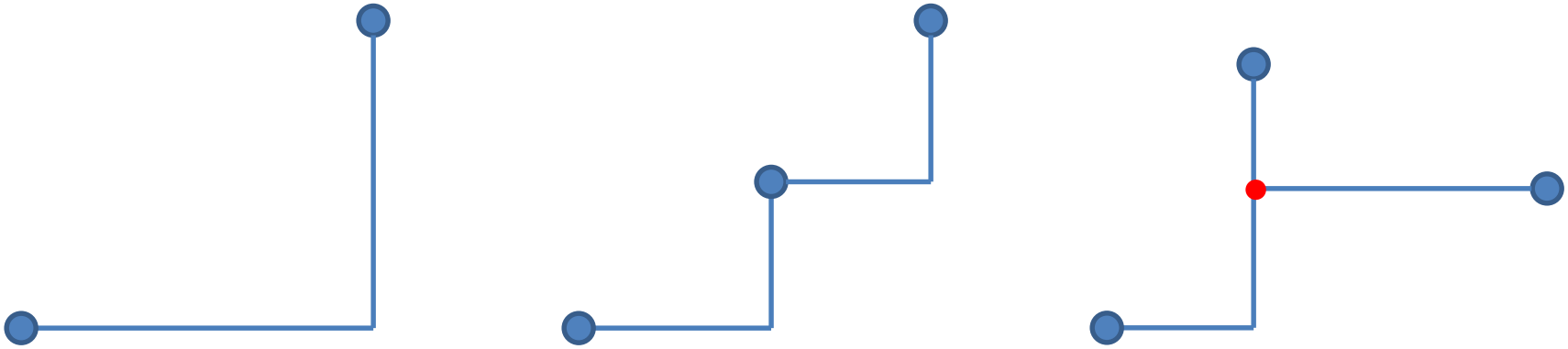
- Global routing



Each boundary can accommodate two nets.

Problems – Routing

- Global routing
  - Steiner routing (for multi-fanout nets)

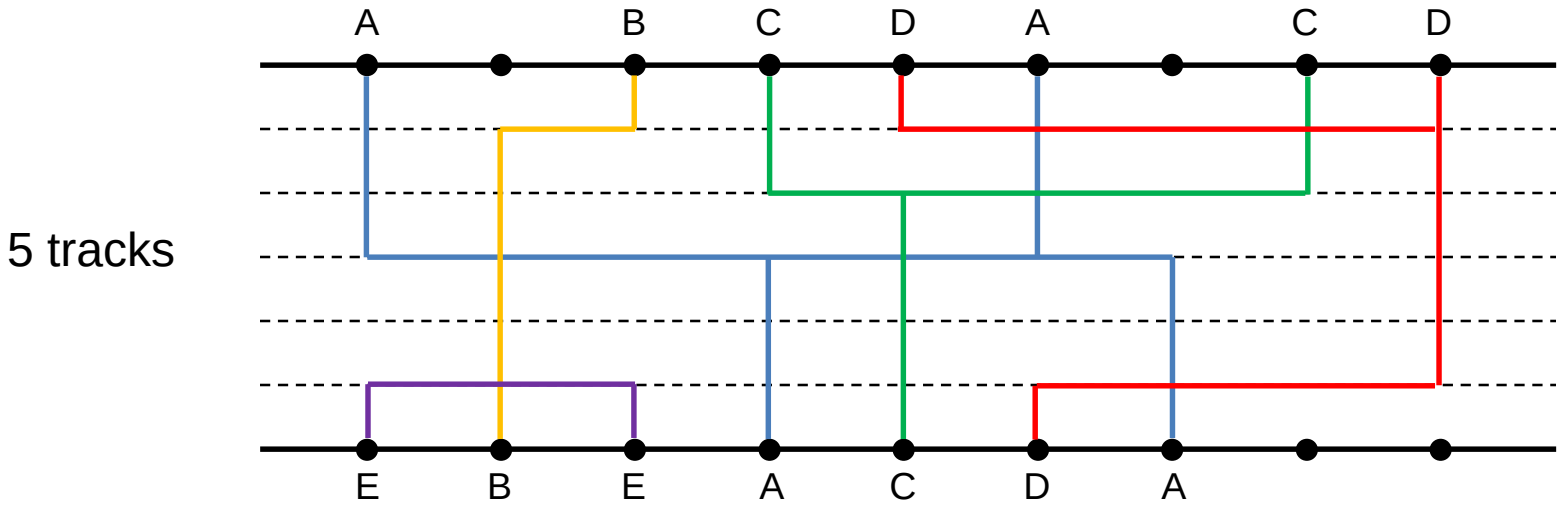


Problems – Routing

- Detailed routing
  - You are given
    - A set of pin locations
      - P1 (50, 100), P2 (55, 150), …
    - Netlist
      - N1 (P1, P4, P6), N2 (P2, P3), …
    - Core area
      - Width: 1,000um, Height: 1,000um
    - Global routing result
    - Detailed routing resources
  - Find a detailed routing solution
    - Minimize
      - Total wirelength
    - Maximize
      - Routability

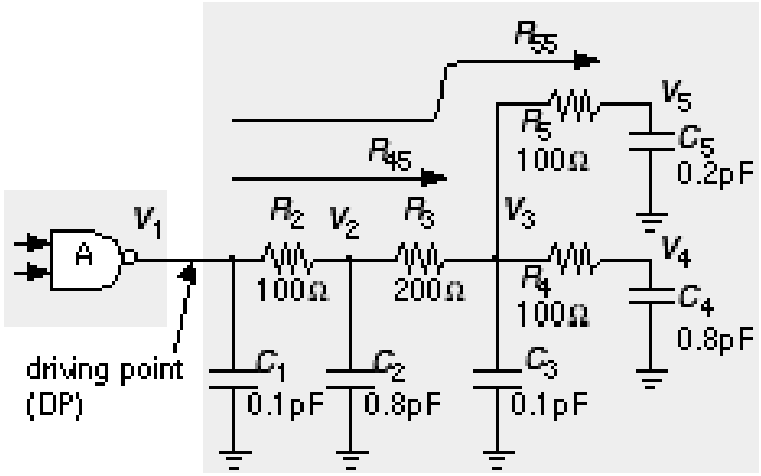
Problems – Routing

- Detailed routing
  - Channel routing
    - Two metal layers

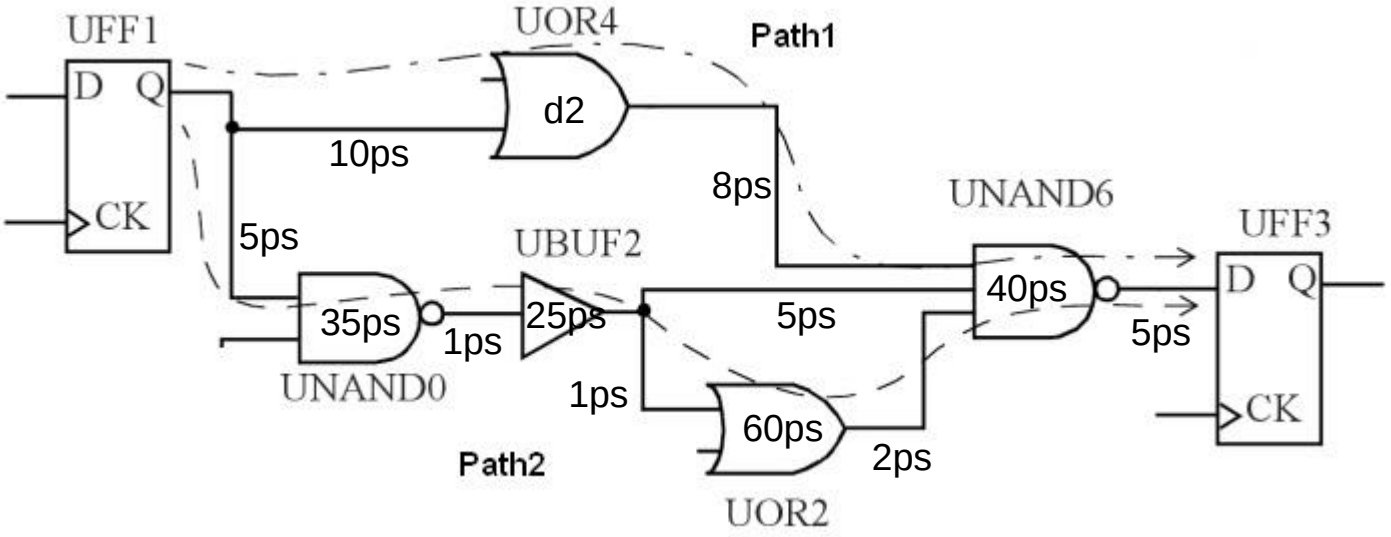


Problems – Interconnect

- Delay calculation

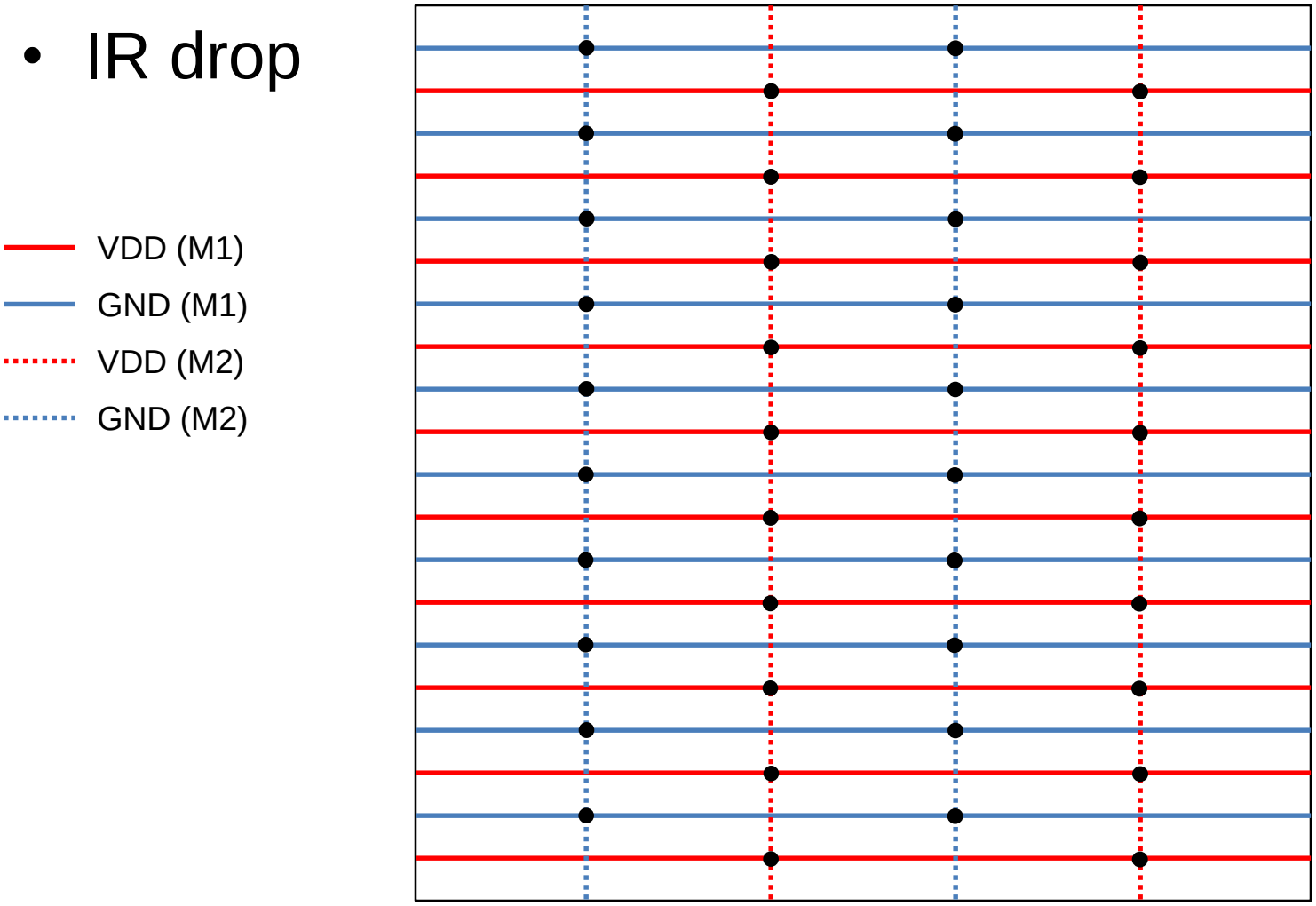


Problems – Timing Analysis



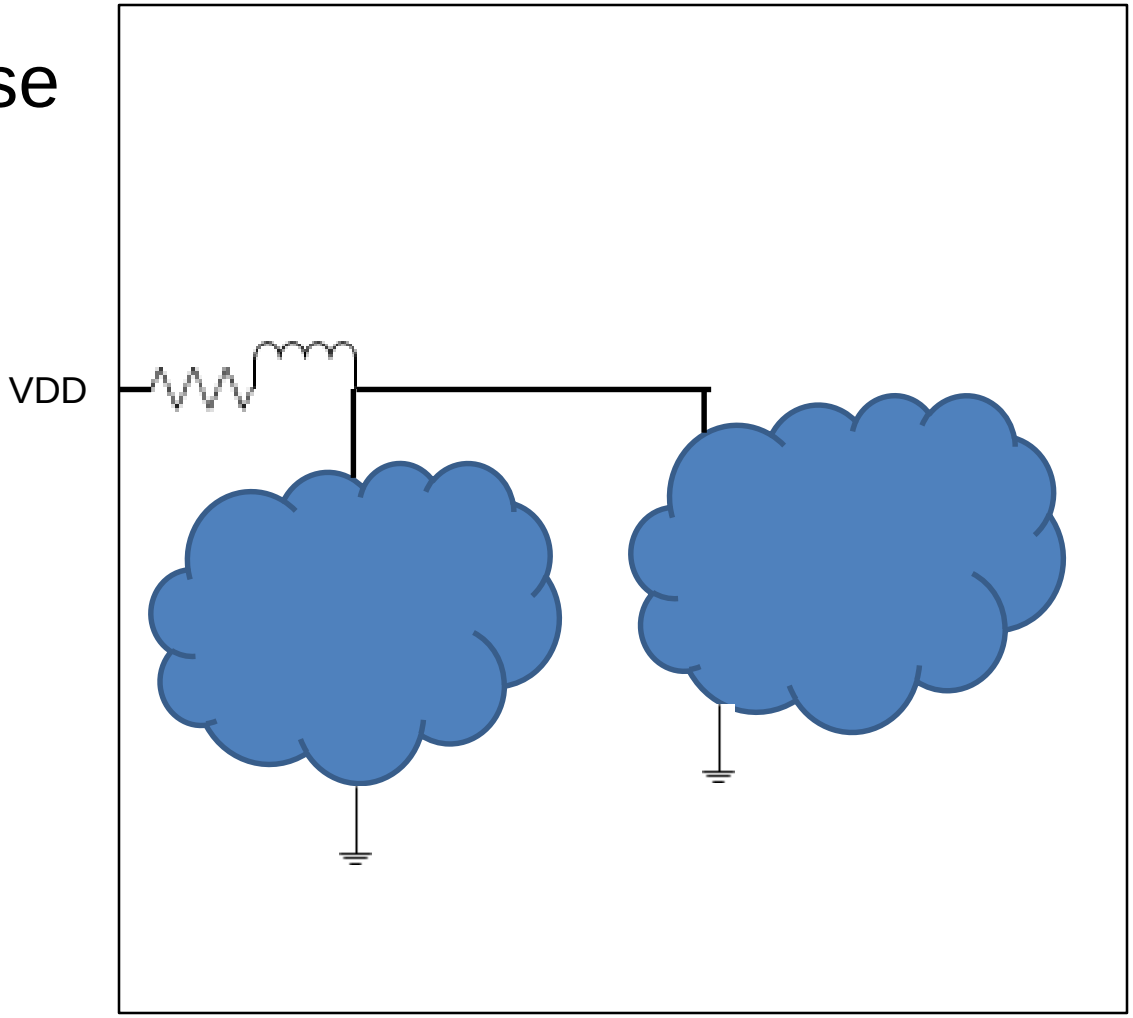
Problems – Power Analysis

- IR drop

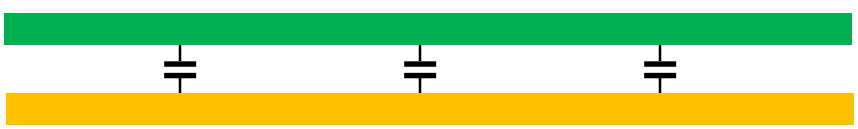


Problems – Power Analysis

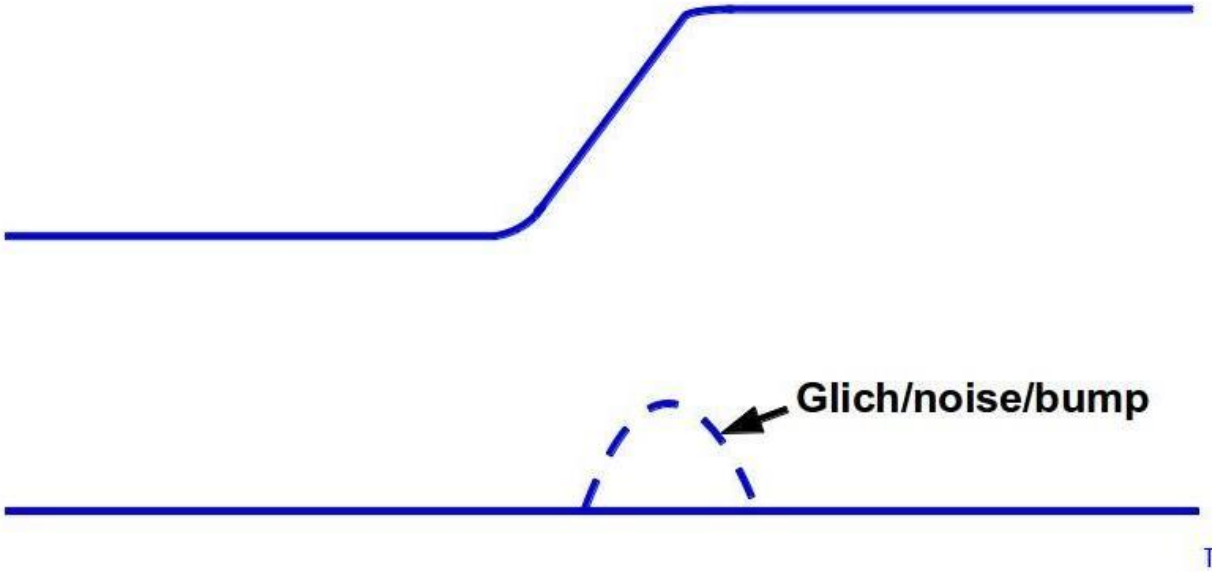
- $L\frac{di}{dt}$ noise



Problems – Cross-talk



- Delay
- Signal inversion



Problems – Interconnect Optimization

- Buffer insertion
  - You are given
    - An RC tree
    - A set of available buffers
    - Bufferable locations
  - Minimize by buffer insertion
    - Net delay

Problems – Interconnect Optimization

- Buffer insertion
  - What to consider
    - Slew computation / estimation / propagation
    - Delay calculation
    - Bufferable locations
    - Routing topology generation
    - Power consumption
    - …

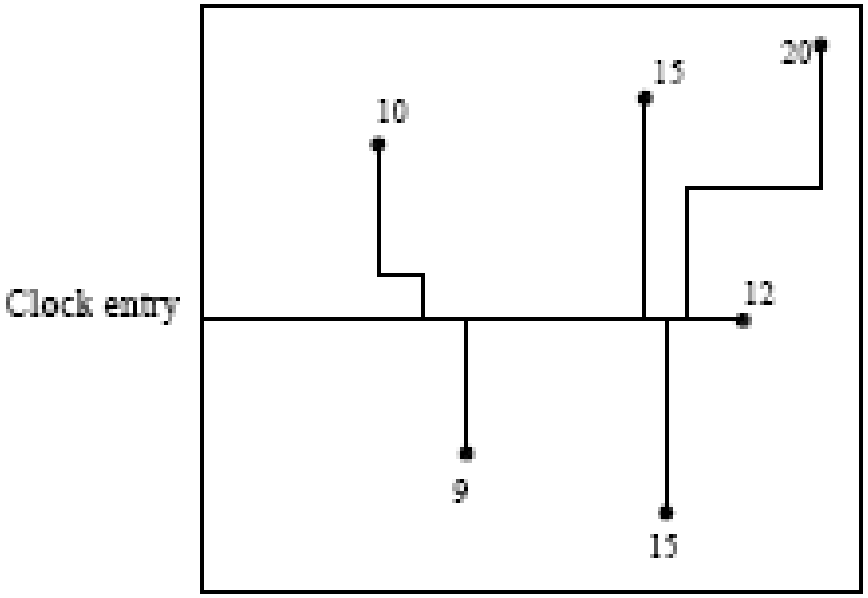
Problems – Interconnect Optimization

- Gate sizing
  - You are given
    - An RC tree
    - A set of available buffers
  - Minimize by gate sizing
    - Net delay
    - Path delay

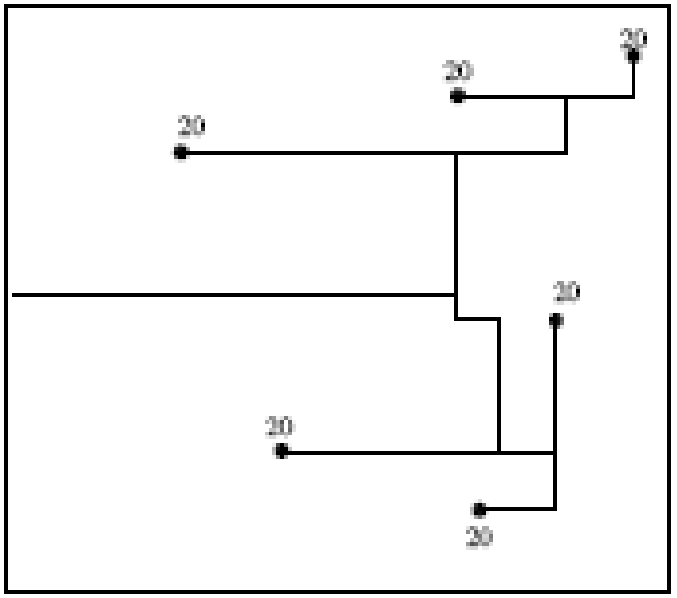
Problems – Clock Tree Synthesis

- CTS
  - You are given
    - F/Fs and their locations
    - A set of available buffers
    - RC characteristics of the interconnect
  - Minimize
    - Clock skew
    - Power consumption
    - Noise
    - Slew

Problems – Clock Tree Synthesis



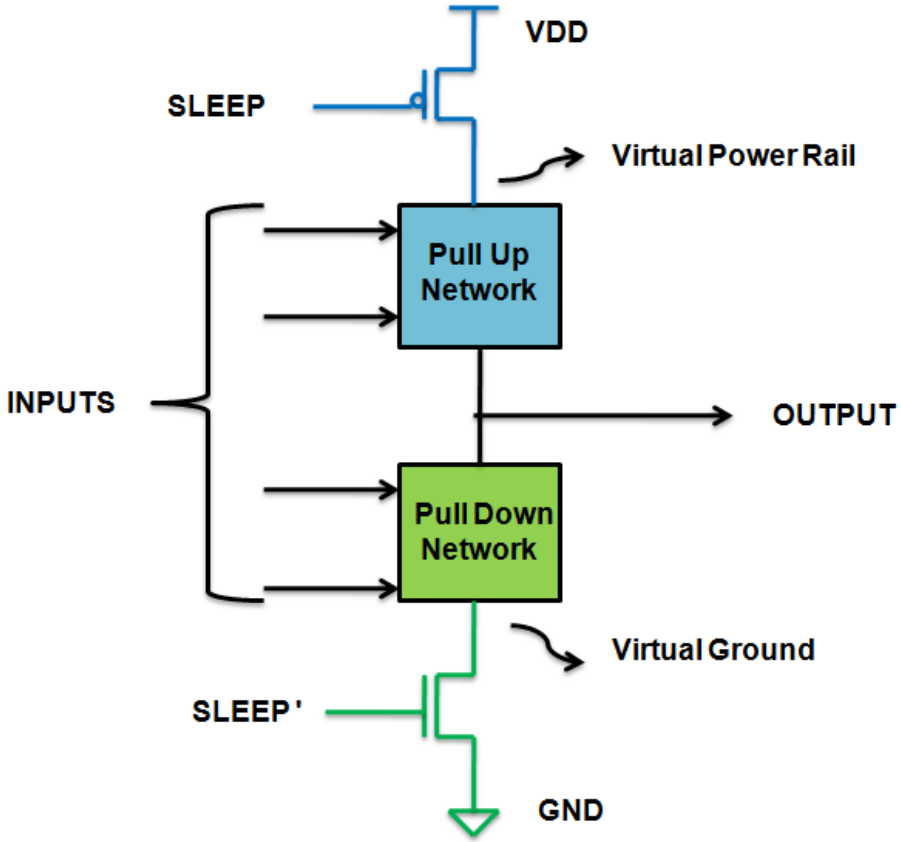
Clock skew = 20 - 9 = 11 units



Clock skew = 0

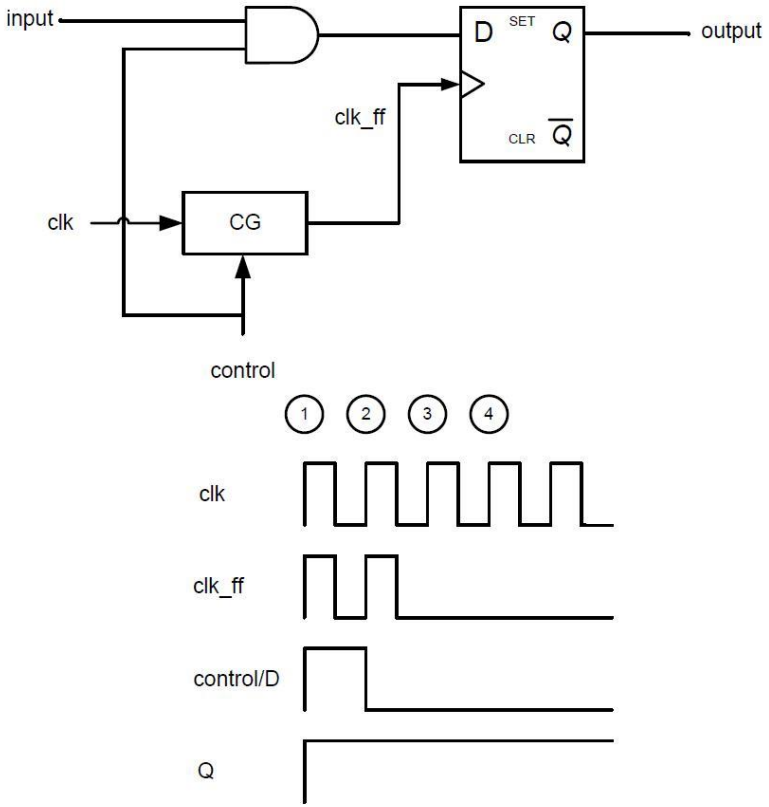
Problems – Low-Power Design

- Power gating



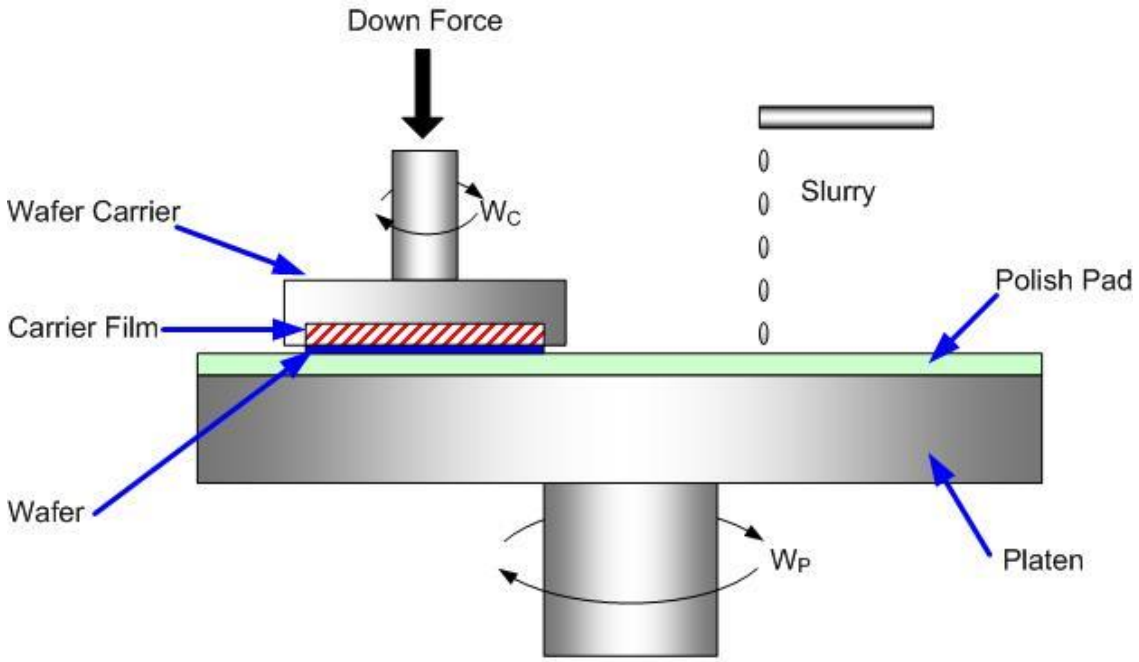
Problems – Low-Power Design

- Clock gating



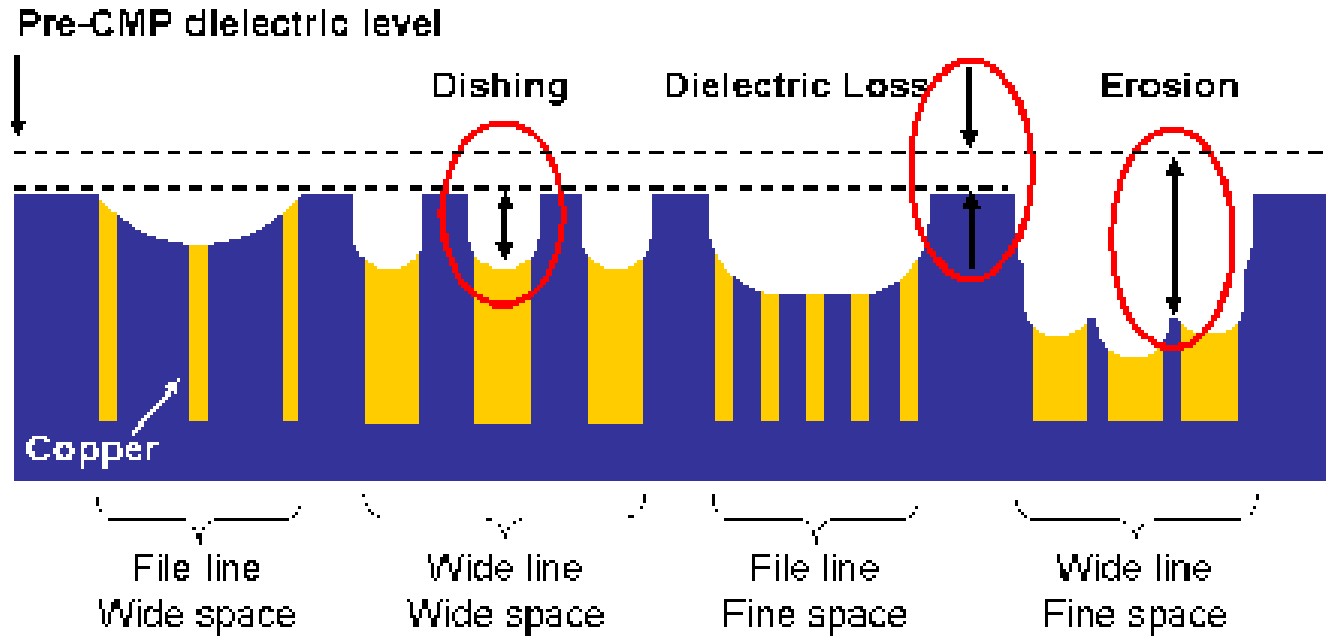
Problems – DFM

- Design for Manufacturability (DFM)
  - Chemical-Mechanical Polishing (CMP)



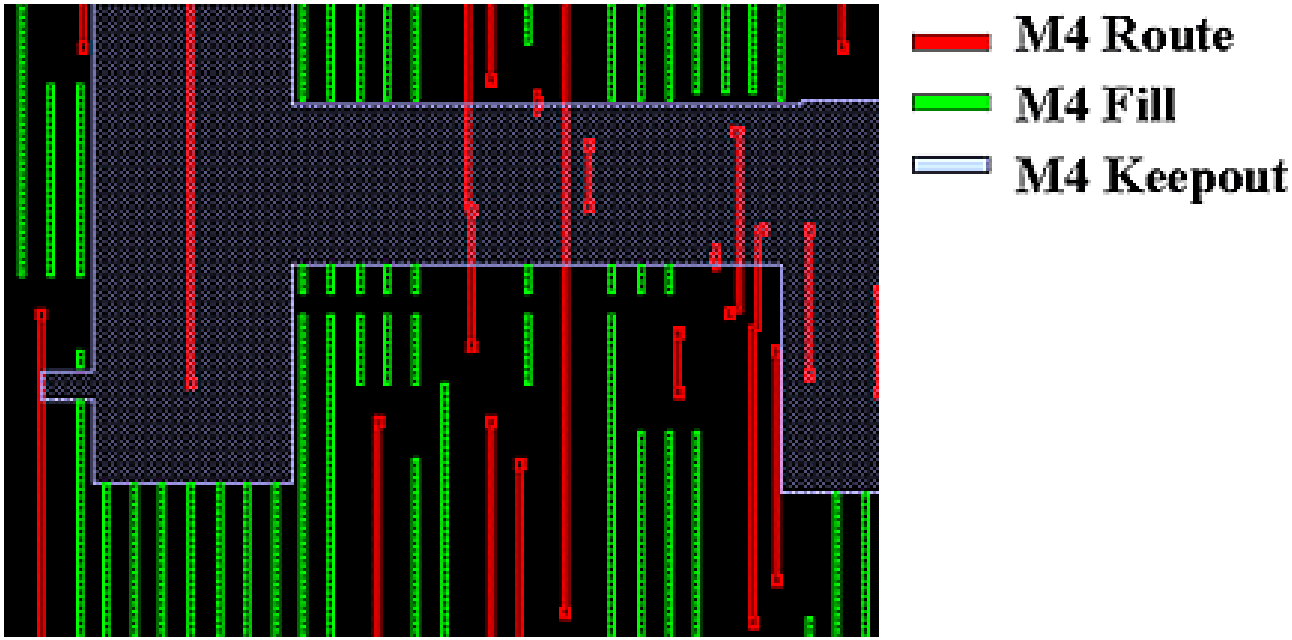
Problems – DFM

- Metal fill insertion



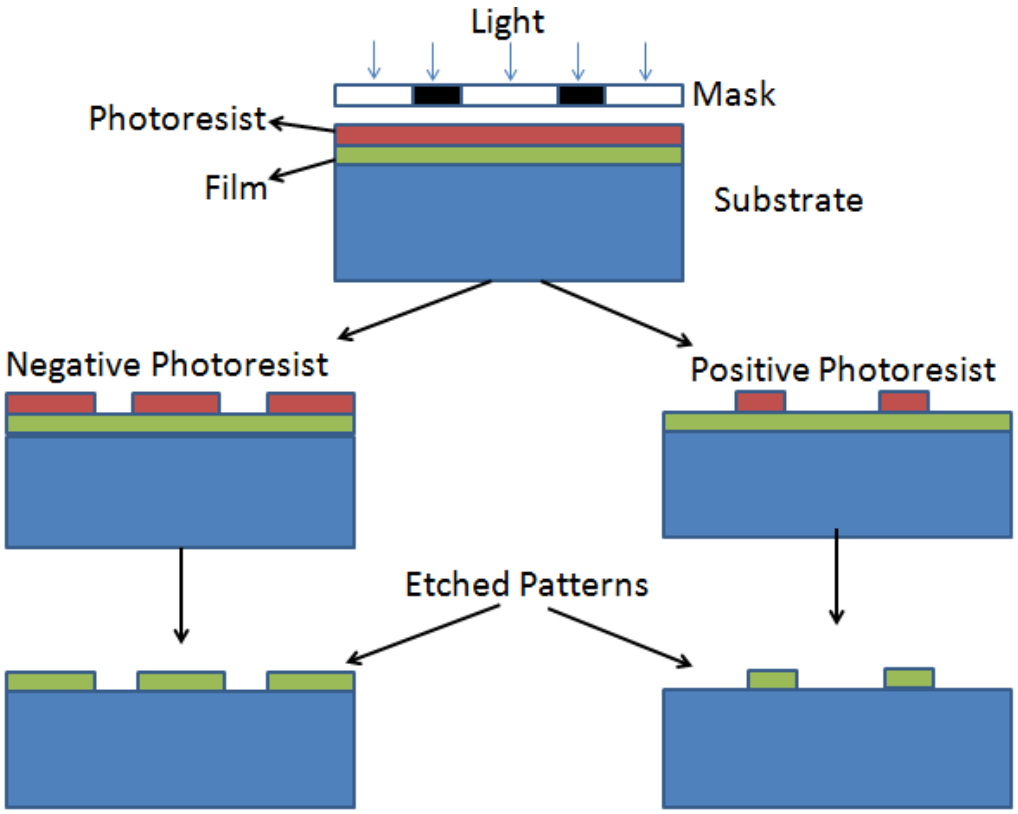
Problems – DFM

- Metal fill insertion



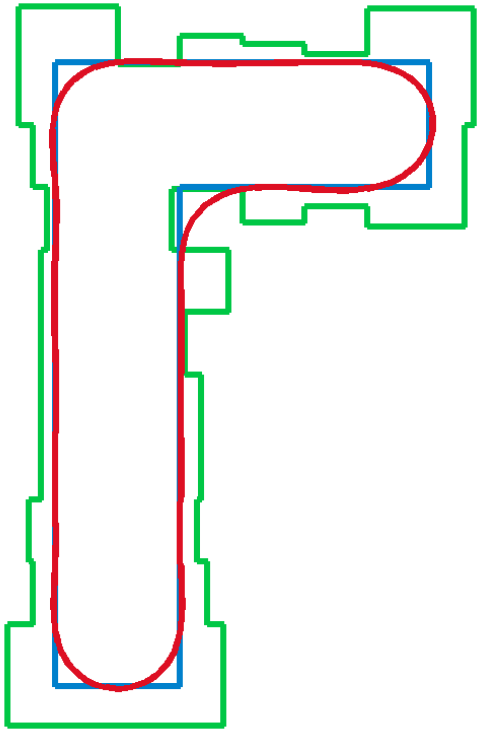
Problems – DFM

- Lithography



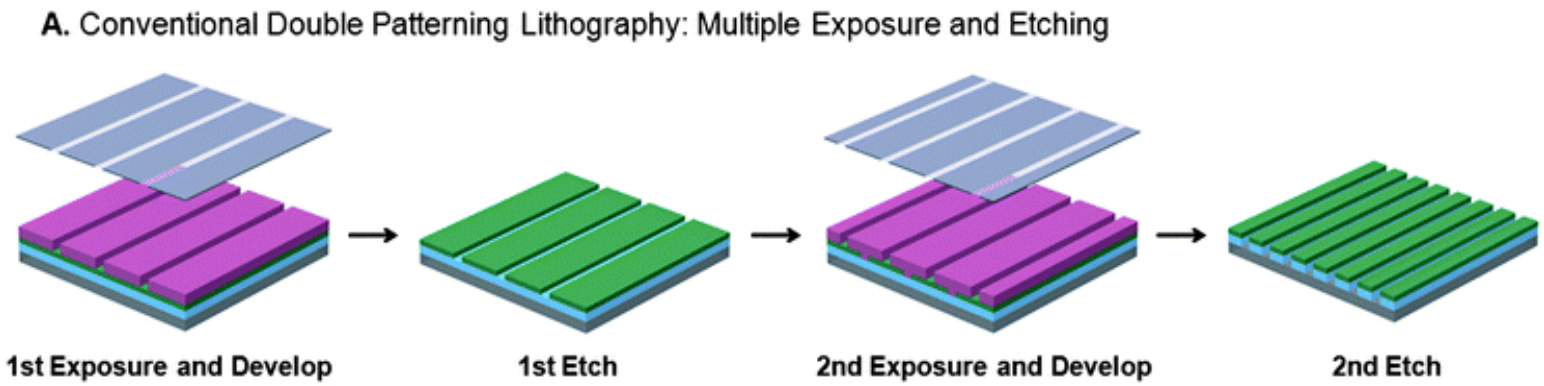
Problems – DFM

- Lithography
  - Optical Proximity Correction (OPC)



Problems – DFM

- Lithography
  - Multiple patterning



Complexity Analysis

- Sorting algorithms
  - Bubble sort
  - Merge sort
  - Bucket sort

Complexity Analysis – Bubble Sort

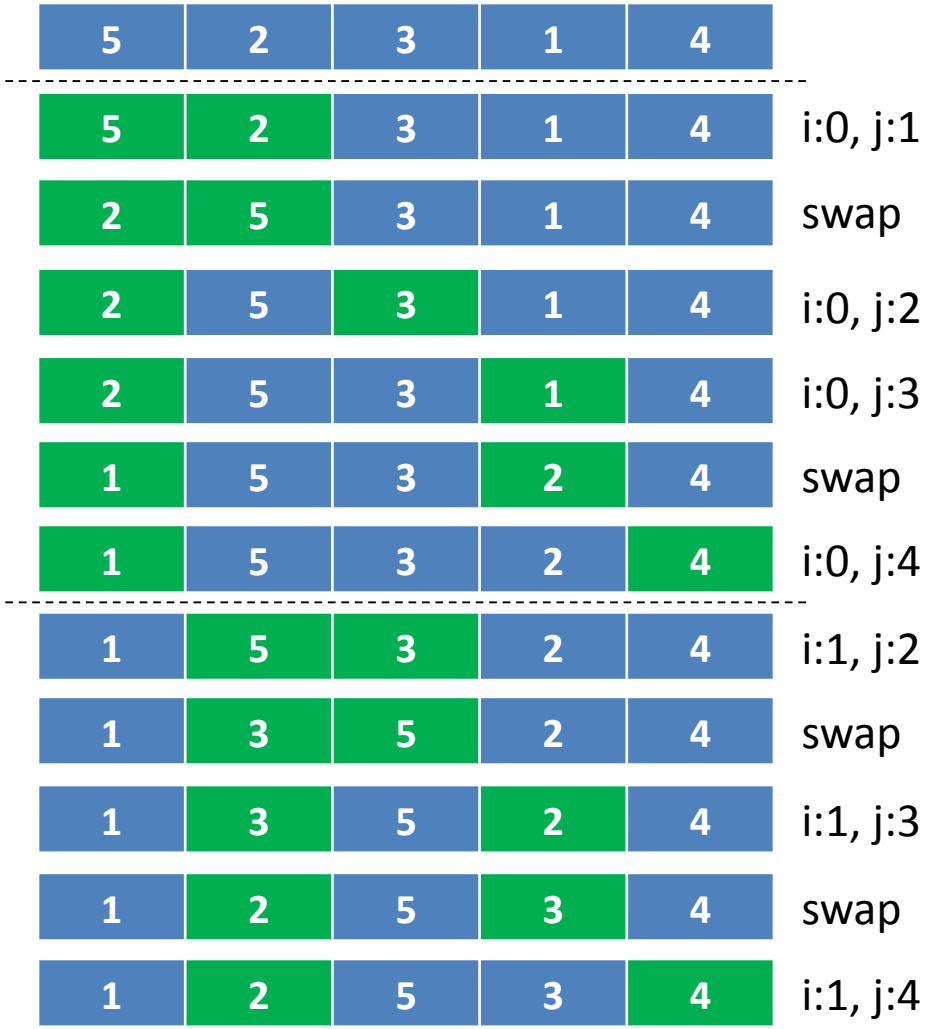
- Pseudo code

```
for ( int i=0 ; i<5 ; i++ ) {
    for ( int j=i+1 ; j<5 ; j++ ) {
        if ( array[j] < array[i] )
            swap (array[i], array[j]);
    }
}
```

Complexity Analysis – Bubble Sort

- Pseudo code

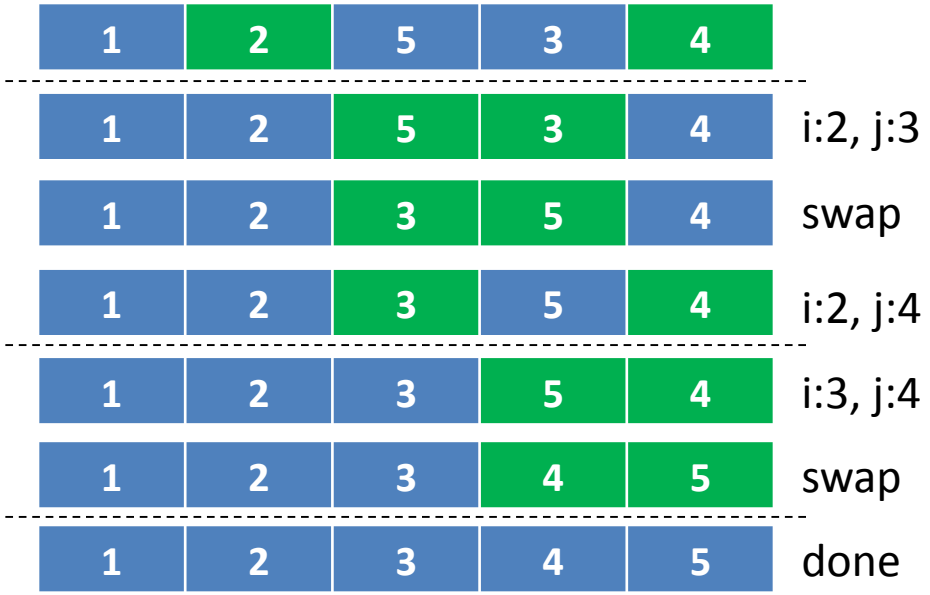
```
for ( int i=0 ; i<5 ; i++ ) {
    for ( int j=i+1 ; j<5 ; j++ ) {
        if ( array[j] < array[i] )
            swap (array[i], array[j]);
    }
}
```



Complexity Analysis – Bubble Sort

- Pseudo code

```
for ( int i=0 ; i<5 ; i++ ) {
    for ( int j=i+1 ; j<5 ; j++ ) {
        if ( array[j] < array[i] )
            swap (array[i], array[j]);
    }
}
```



- # iterations (for five elements): 4+3+2+1+0
- # iterations (for *n* elements): (n-1) + (n-2) + … + 0 = $\frac{(n-1)n}{2}$ = $O(n^2)$
- Complexity of the if statement: O(1) = constant
- Final: $O(n^2)$

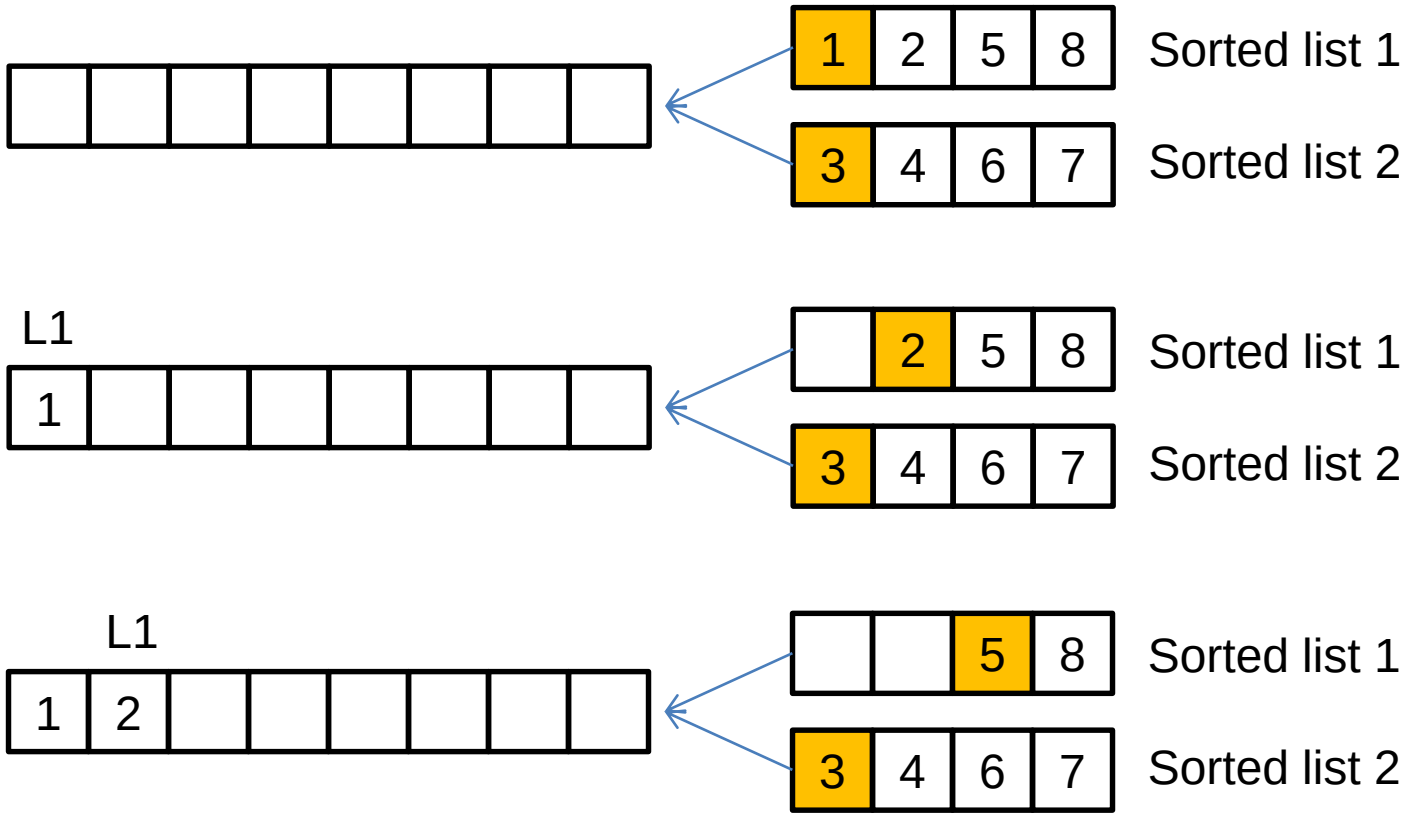
Complexity Analysis – Merge Sort

- Pseudo code

```
Split (left, right) {
    if ( (right – left) ≥ 2 ) {
        Split (left, (left+right)/2);
        Split ((left+right)/2, right);
        Merge (left, (left+right)/2, right);
    }
}
```

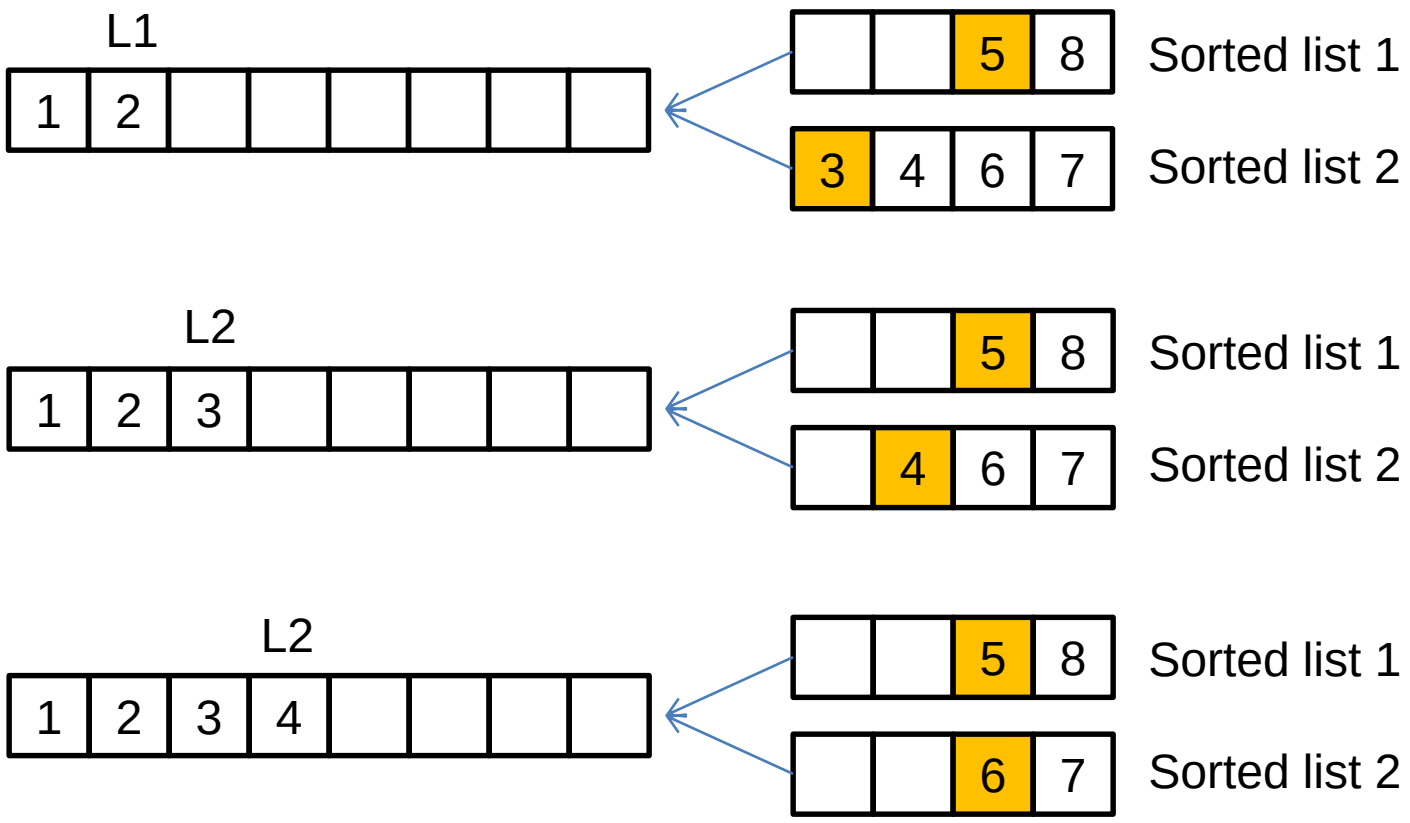
Complexity Analysis – Merge Sort

- Idea



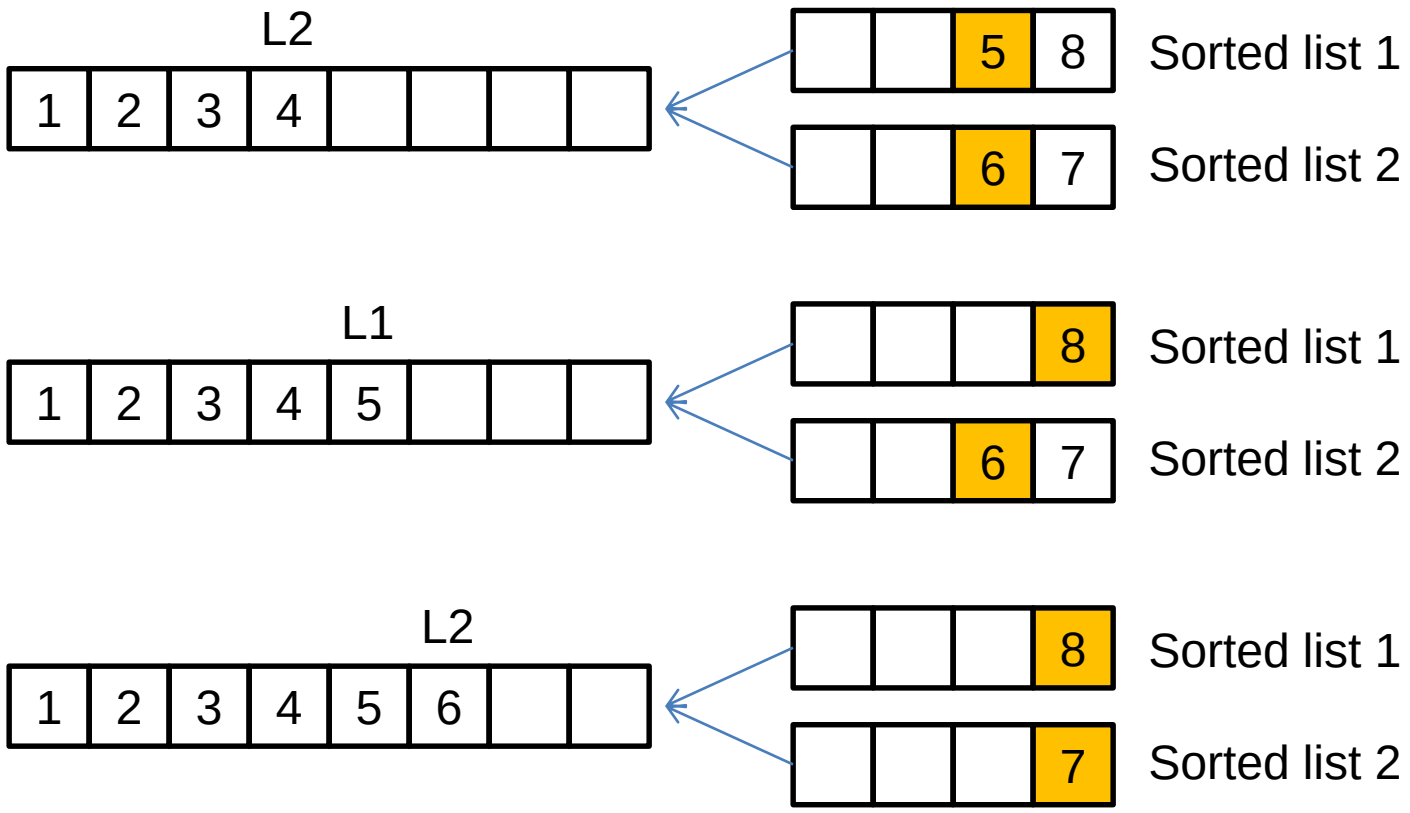
Complexity Analysis – Merge Sort

- Idea



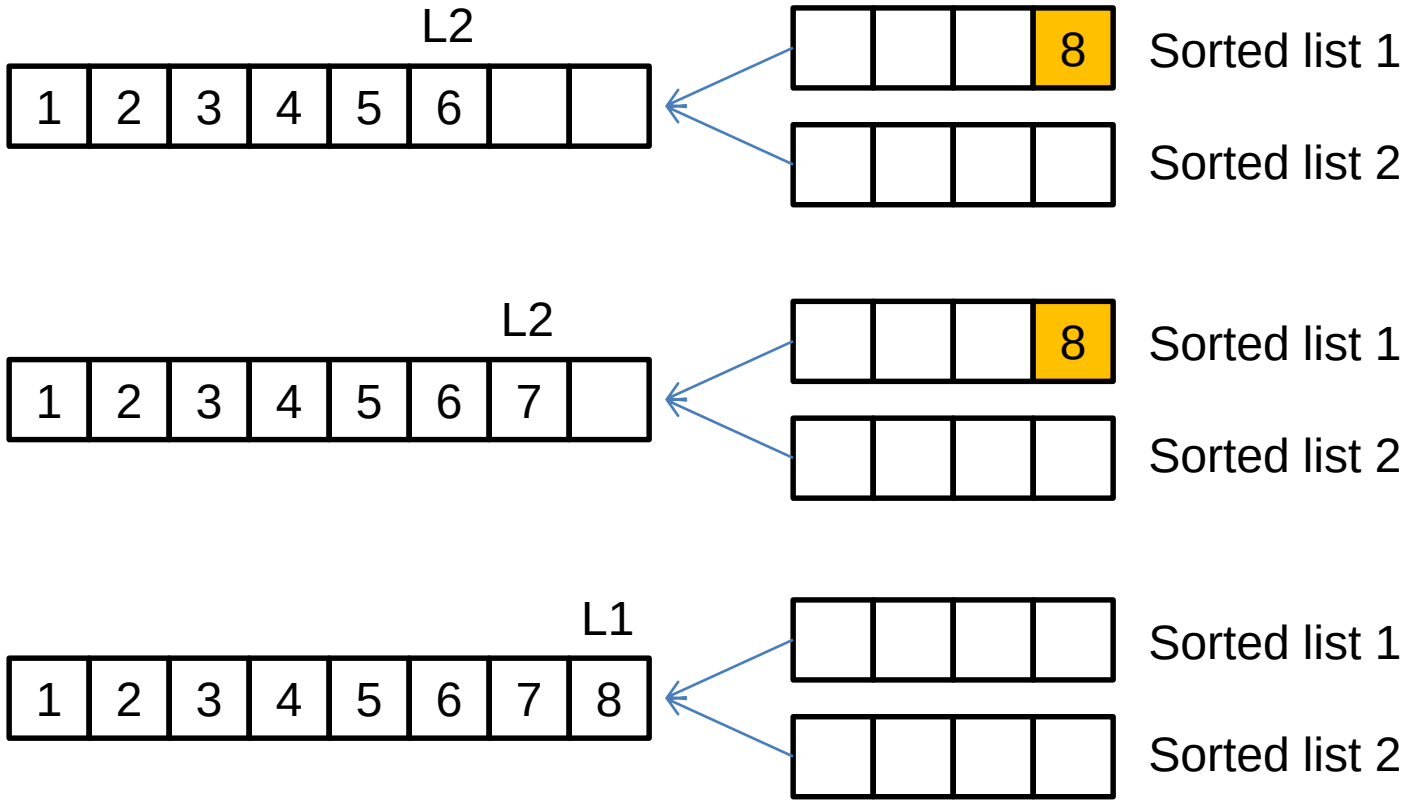
Complexity Analysis – Merge Sort

- Idea

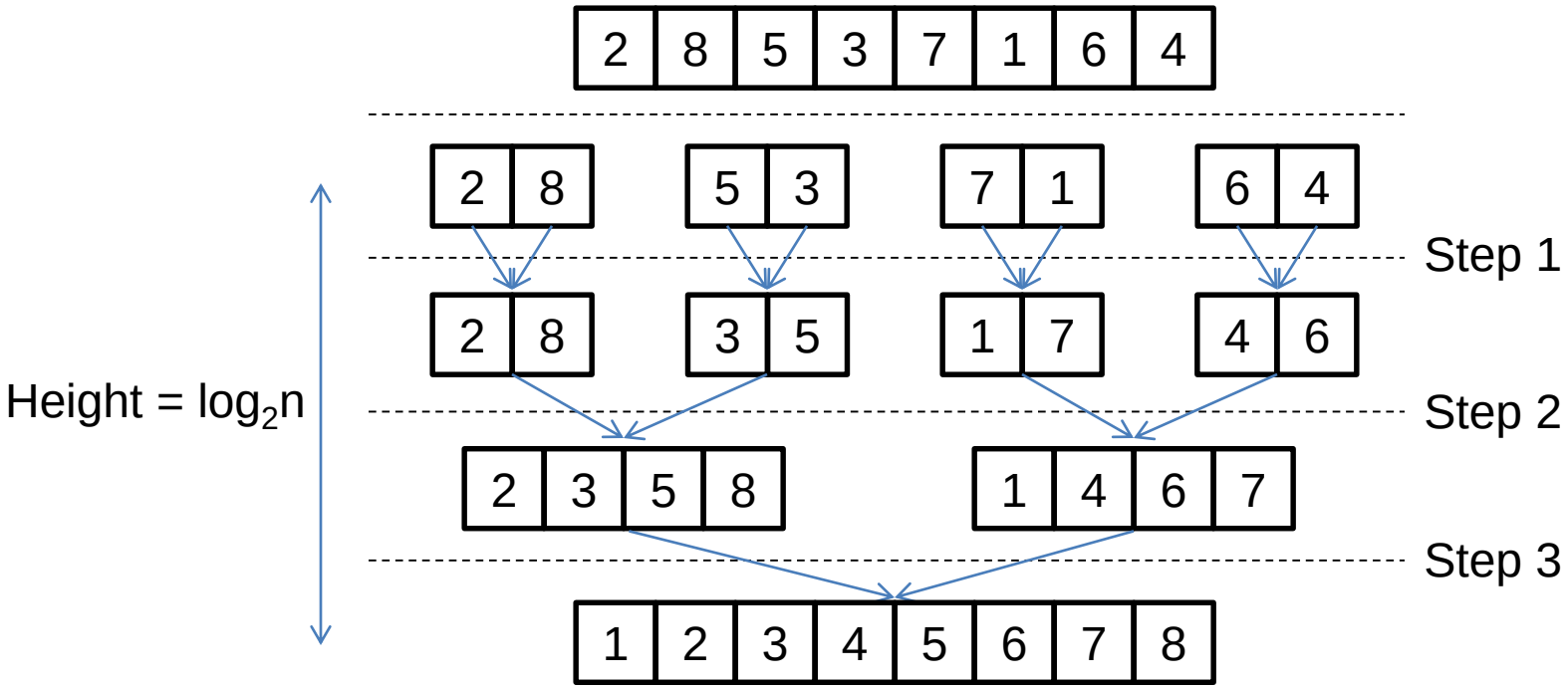


Complexity Analysis – Merge Sort

- Idea



Complexity Analysis – Merge Sort



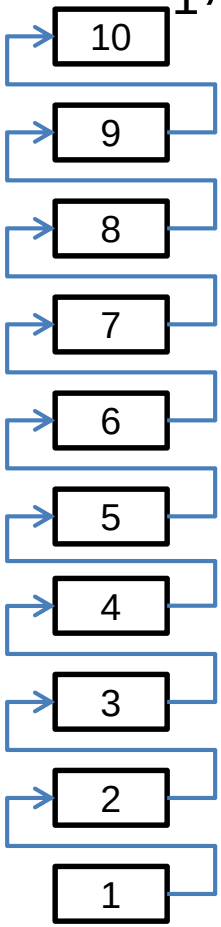
Complexity: $O(n * \log_2 n)$

Complexity Analysis – Bucket Sort

- We know the range of the values.
  - [1, 10]
  - 7, 3, 3, 1, 6, 8, 3, 6, 8, 6, 4, 2, 2, 7, 8, 3

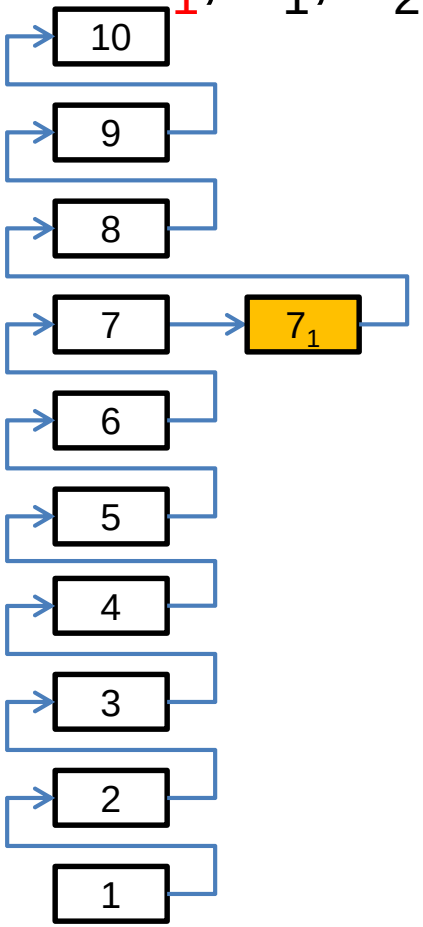
Complexity Analysis – Bucket Sort

- $7_1, 3_1, 3_2, 1_1, 6_1, 8_1, 3_3, 6_2, 8_2, 6_3, 4_1, 2_1, 2_2, 7_2, 8_3$



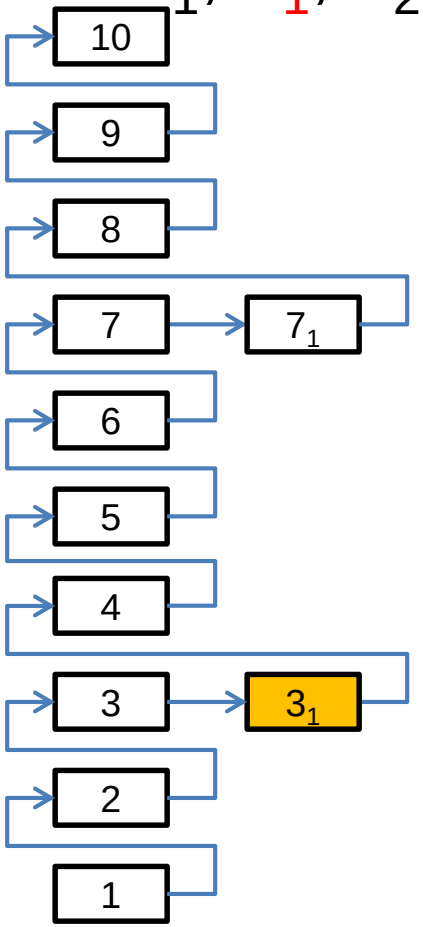
Complexity Analysis – Bucket Sort

- 7_1, 3_1, 3_2, 1_1, 6_1, 8_1, 3_3, 6_2, 8_2, 6_3, 4_1, 2_1, 2_2, 7_2, 8_3



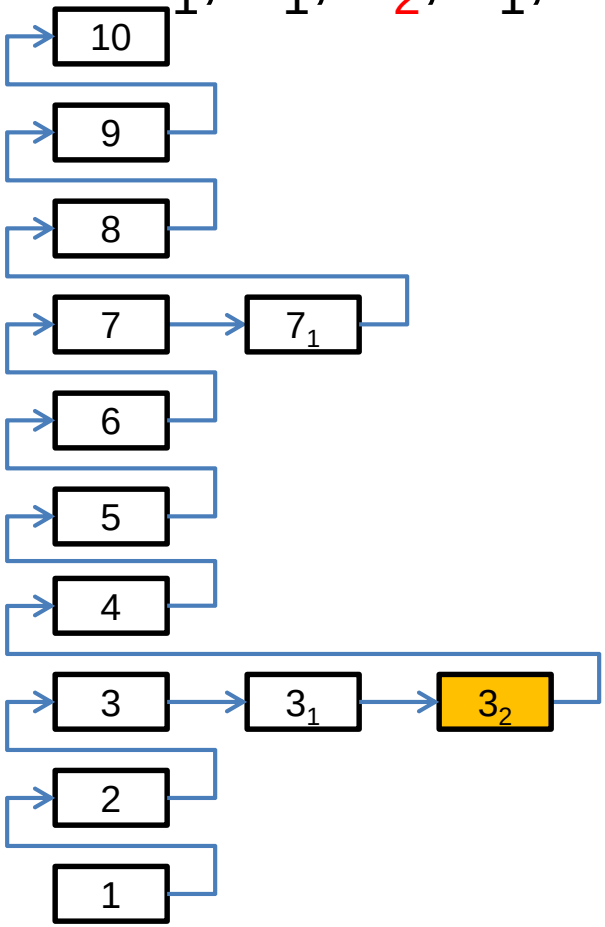
Complexity Analysis – Bucket Sort

- $7_1, 3_1, 3_2, 1_1, 6_1, 8_1, 3_3, 6_2, 8_2, 6_3, 4_1, 2_1, 2_2, 7_2, 8_3$



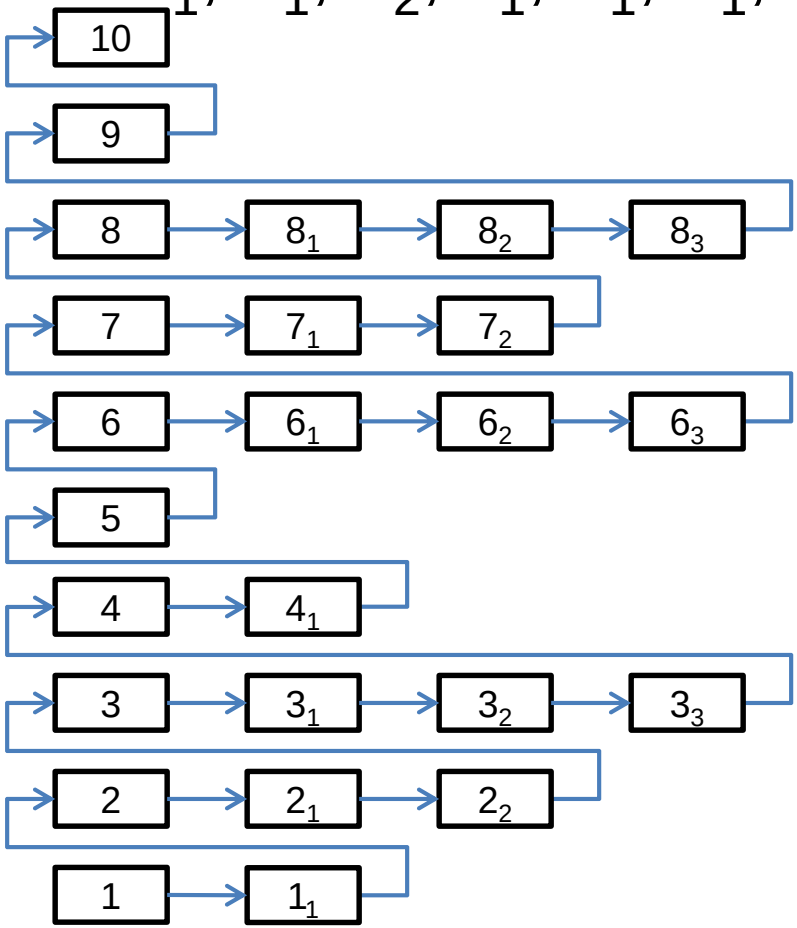
Complexity Analysis – Bucket Sort

- $7_1, 3_1, 3_2, 1_1, 6_1, 8_1, 3_3, 6_2, 8_2, 6_3, 4_1, 2_1, 2_2, 7_2, 8_3$



Complexity Analysis – Bucket Sort

- $7_1, 3_1, 3_2, 1_1, 6_1, 8_1, 3_3, 6_2, 8_2, 6_3, 4_1, 2_1, 2_2, 7_2, 8_3$



Complexity: O(n)