EE244: Design Technology for Integrated Circuits and Systems

Outline

Lecture 1.2

- **Representing the Physical Design.**

Simplified Model of Design

Behavior
Functions the system must implement
Time, area, power, etc. constraints
Implementation-independent description

Register
Components and their interconnections
Std. components & ROM, ASIC, PLD
Timing constraints for components

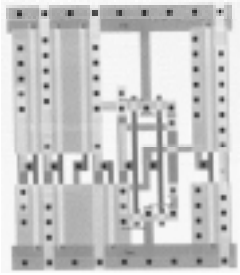
Gate
Low-level components & nets
In terms of ASIC library
EDIF well-suited to this description

Mask
Physical layout of IC or board
EDIF well-suited to this description

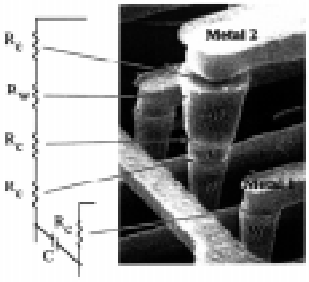
Symbolic Layout
Combination of relative physical information plus electrical connectivity

Copyright © 1997, A. Richard Newton

Mask-Level Layout



- Collection of shapes, on different *mask layers*, that represent the circuit to be constructed.
- Almost exclusively rectangles
- 1B gates: O(100B) rectangles/mask layer; O(10^{12}) rectangles/chip
- What are the best ways to store and manipulate such data?



Storing and Manipulating Mask Data

- <u>Precision</u>: Dimensions relative, so introduce the concept of a minimum resolvable distance, λ (lambda), and then represent all geometry in terms of it. If λ = 0.1μm, and chip edge, Λ, is 10cm (worst case), then we need a coordinate space of range 10^6.
- 2^{16} ~ O(10^5), 2^{24} ~ O(10^7), 2^{32} ~ O(10^9)
- Optimally 3bytes/int, practically 4bytes/int
  - (X_{LL}, Y_{LL}, X_{UR}, Y_{UR}, mask) or (X_{LL}, Y_{LL}, L, W, mask)
  - 4+4+4+4+1=17 bytes/rectangle ~ O(10^{12}) bytes/mask layer
  - Certainly impractical today: Must exploit *hierarchy* and *regularity,* and *multiprocessing* in the design methodology

Copyright © 1997, A. Richard Newton

Manipulating Geometry: Common Operations

- Full Chip Analysis
  - *foreach*(rectangle on a mask layer, R)
  - generate representation of "empty space" ("white space")
- Neighborhood Search: same mask or inter-mask
- *foreach*(rectangle "near" R)
  - within an area/volume, "visible", "to the right of", "touching", etc.
- Incremental Operations
  - Insert rectangle, delete "named" rectangle
  - Window: return all shapes present in a coordinate range
- What are the relative complexities of these operations?

EE244 Fall 97 1.2.5

Storing Geometry: Different Approaches

- Linked list of rectangles (sorted? how?)
- Sorted in X, sorted in Y
- Bin-sort, tree sort, logs and poles
- *Quad CIF Tree*
- Corner-stitched data structure
- Raster-based

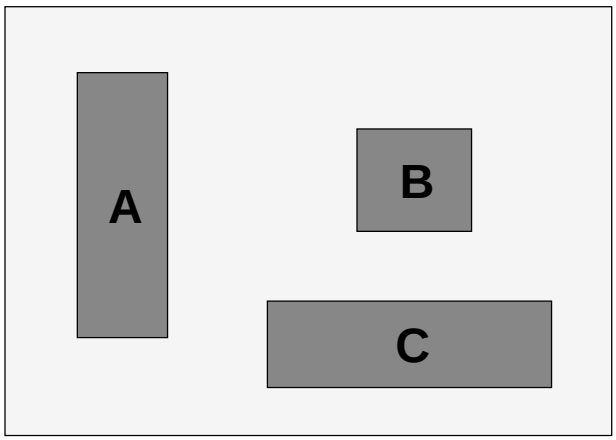
Issues:
How to convert geometric "data base" into our run-time "data structure"?
How to most efficiently manipulate our run-time data structure

Taking into account multiprocessing options (e.g. NOW)

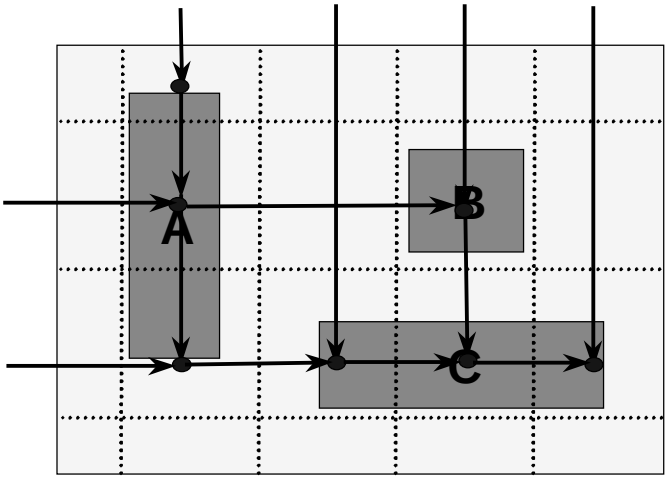
EE244 Fall 97 1.2.6

Copyright © 1997, A. Richard Newton

Storing Geometry: Different Approaches



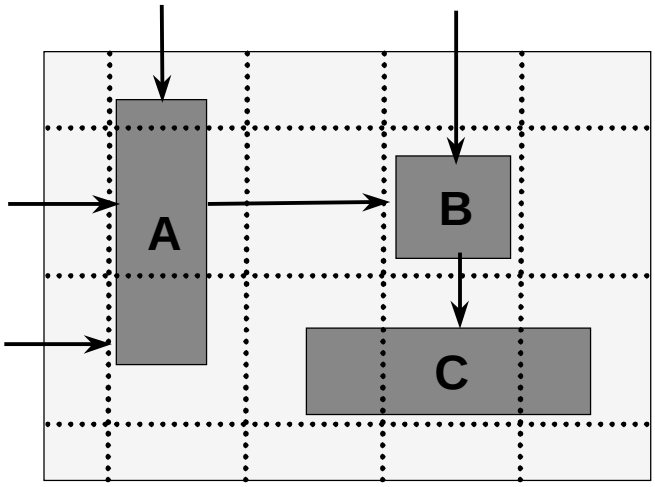
Storing Geometry: Different Approaches



X-Y Bins

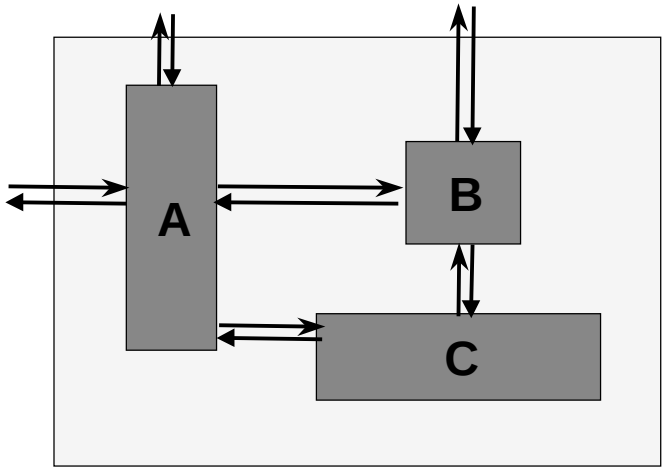
Copyright © 1997, A. Richard Newton

Storing Geometry: Different Approaches



Logs and Poles

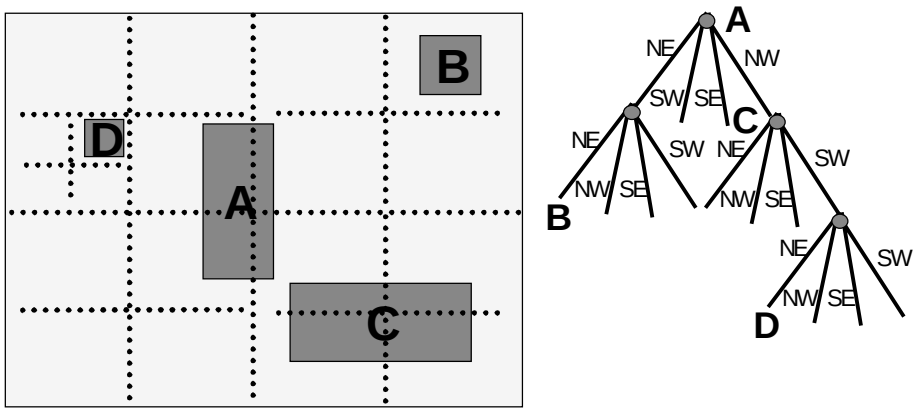
Storing Geometry: Different Approaches



Neighbor Pointers

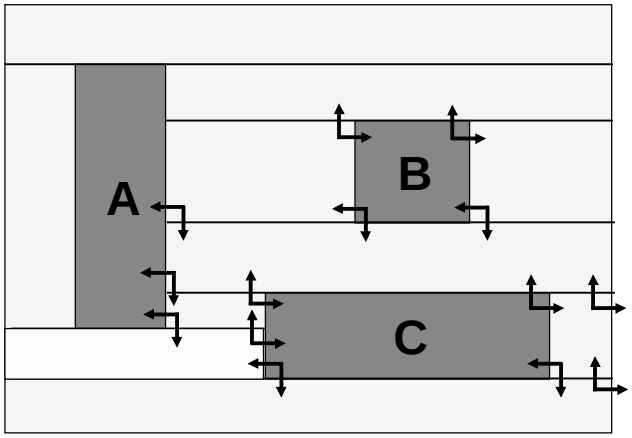
Copyright © 1997, A. Richard Newton

Storing Geometry: Different Approaches



Trees: Binary, Quad, KD

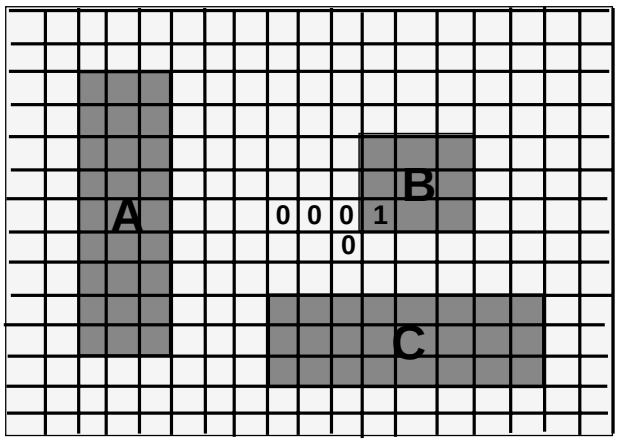
Storing Geometry: Different Approaches



Corner Stitching

Copyright © 1997, A. Richard Newton

Storing Geometry: Different Approaches



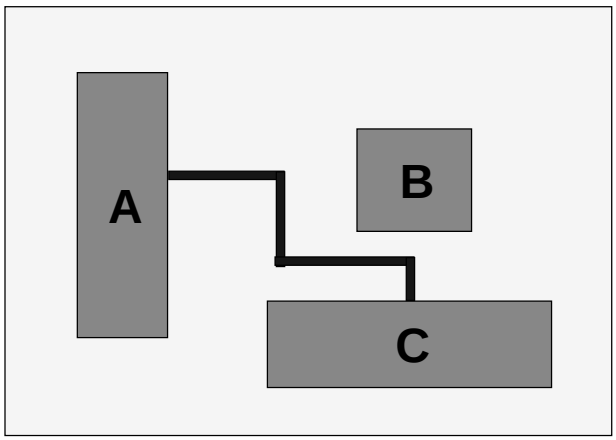
Bitmap

Summary of Approaches

Data Structure	Initialize	Insert R	Point_Find	Window(X,Y)	Memory
Linked-list (sorted)					
X-Y Bins					
Logs & Poles					
Neighbor Point's					
Binary Tree					
Quad Tree					
Corner Stitch					
Bitmap					

Copyright © 1997, A. Richard Newton

Storing Geometry: Different Approaches



Complexity of Operations

- A strong function of the geometrical decomposition!
- Memory/storage requirements
- Processing time for common operations
- Ability to perform in parallel and how that effects efficiency
- Worst-case and expected-case
- Great project idea!

Copyright © 1997, A. Richard Newton