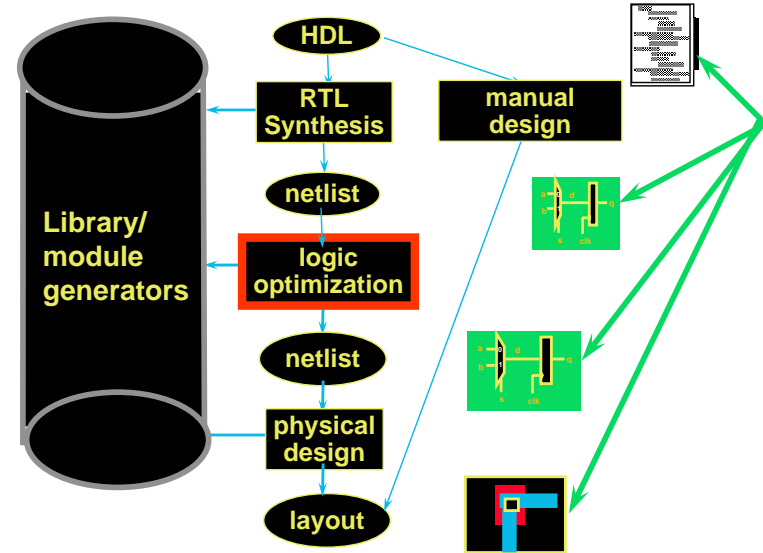


# VLSI CAD Flow: Logic Synthesis, Placement and Routing

## 6.375 Lecture 5

Guest Lecture by Sriniv Devadas

# RTL Design Flow



## Two-Level Logic Minimization

Can realize an arbitrary logic function in sum-of-products or two-level form

$$F_1 = \bar{A}\bar{B} + \bar{A}BD + \bar{A}B\bar{C}\bar{D} + ABC\bar{D} + A\bar{B} + ABD$$

$$F_1 = \bar{B} + D + \bar{A}\bar{C} + AC$$

Of great interest to find a minimum sum-of-products representation

– Solved problem even for functions with 100's of inputs (variants of Quine-McCluskey)

## Two-Level versus Multilevel

2-Level:

$$f_1 = AB + AC + AD$$

$$f_2 = \bar{A}B + \bar{A}C + \bar{A}E$$

6 product terms which cannot be shared.  
24 transistors in static CMOS

Multi-level:

Note that  $B + C$  is a common term in  $f_1$  and  $f_2$

$$K = B + C$$

$$f_1 = AK + AD$$

$$f_2 = \bar{A}K + \bar{A}E$$

3 Levels

20 transistors in static CMOS  
not counting inverters

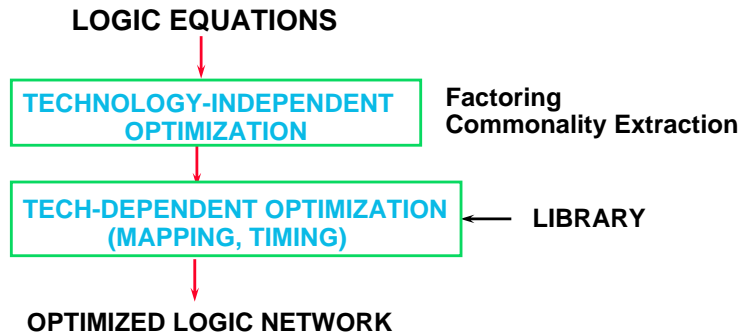
## Technologies

“Closed book”:

gate-array  
standard-cell

“Open book”:

CMOS Domino,  
complex gate static CMOS



5

## Tech.-Independent Optimization

Involves:

Minimizing two-level logic functions.  
Finding common subexpressions.  
Substituting one expression into another.  
Factoring single functions.

Factored versus Disjunctive forms

$$f = ac + ad + bc + bd + a\bar{e}$$

sum-of-products or disjunctive form

$$f = (a + b)(c + d) + a\bar{e}$$

factored form

multi-level or complex gate

6

## Optimizations

$$F = \begin{cases} f_1 = AB + AC + AD + AE + \bar{A}\bar{B}\bar{C}\bar{D}\bar{E} \\ f_2 = \bar{A}\bar{B} + \bar{A}\bar{C} + \bar{A}\bar{D} + \bar{A}\bar{F} + \bar{A}\bar{B}\bar{C}\bar{D}\bar{F} \end{cases}$$

Factor  $F$

$$F = \begin{cases} f_1 = A(B + C + D + E) + \bar{A}\bar{B}\bar{C}\bar{D}\bar{E} \\ f_2 = \bar{A}(B + C + D + F) + \bar{A}\bar{B}\bar{C}\bar{D}\bar{F} \end{cases}$$

Extract common expression

$$G = \begin{cases} g_1 = B + C + D \\ f_1 = A(g_1 + E) + \bar{A}\bar{E}\bar{g}_1 \\ f_2 = \bar{A}(g_1 + F) + \bar{A}\bar{F}\bar{g}_1 \end{cases}$$

7

## What Does “Best” Mean?

Transistor count  $\longrightarrow$  AREA  
Number of circuits  $\longrightarrow$  POWER  
Number of levels  $\longrightarrow$  DELAY (Speed)

Need quick estimators of area, delay and power which are also accurate

8

## Algebraic vs. Boolean Methods

Algebraic techniques view equations as polynomials and attempt to factor equations or “divide” them

Do not exploit Boolean identities e.g.,  $a\bar{a} = 0$

In algebraic substitution (or division) if a function  $f = f(a, b, c)$  is divided by  $g = g(a, b)$ ,  $a$  and  $b$  will not appear in  $f/g$

Algebraic division:  $O(n \log n)$  time

Boolean division: 2-level minimization required

9

## Comparison

$$f = a\bar{b} + a\bar{c} + b\bar{a} + b\bar{c} + c\bar{a} + c\bar{b}$$

Algebraic factorization procedures

$$f = a(\bar{b} + \bar{c}) + \bar{a}(b + c) + b\bar{c} + c\bar{b}$$

Boolean factorization produces

$$f = (a + b + c)(\bar{a} + \bar{b} + \bar{c})$$

$$l = (b\bar{f} + \bar{b}f)(a + e) + \bar{a}\bar{e}(b\bar{f} + bf)$$

$$r = (b\bar{f} + \bar{b}f)(\bar{a} + \bar{e}) + ae(b\bar{f} + bf)$$

Algebraic substitution of  $l$  into  $r$  fails

Boolean substitution

$$r = a(\bar{e}l + el) + \bar{a}(\bar{e}l + e\bar{l})$$

$$l = a(er + \bar{e}\bar{r}) + \bar{a}(\bar{e}r + e\bar{r})$$

10

## Strong (or Boolean) Division

Given a function  $f$  to be strong divided by  $g$

Add an extra input to  $f$  corresponding to  $g$ , namely  $G$  and obtain function  $h$  as follows

$$h_{DC} = G\bar{g} + \bar{G}g$$

$$h_{ON} = f_{ON} - h_{DC}$$

Minimize  $h$  using two-level minimizer

11

## Strong Division Example

$$f = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$

$$g = a\bar{b} + \bar{a}b$$

$$h_{DC} = G(ab + \bar{a}\bar{b}) + \bar{G}(a\bar{b} + \bar{a}b)$$

$$h_{ON} = f_{ON} - h_{DC}$$

		Ga				
		00	01	11	10	
bc	00		x	1	x	Function $h$
	01	1	x		x	
	11	x	1	x		
	10	x		x	1	

Minimization gives  $h = \bar{G}c + G\bar{c}$

12

## Weak (or Algebraic) Division

Definition: support of  $f$  as  $sup(f) = \{ \text{set of all variables } v \text{ that occur in } f \text{ as } v \text{ or } \bar{v} \}$

Example:  $f = A\bar{B} + C$   
 $sup(f) = \{A, B, C\}$

Definition: we say that  $f$  is orthogonal to  $g$ ,  $f \perp g$ , if  $sup(f) \cap sup(g) = \phi$

Example:  $f = A + B$        $g = C + D$   
 $\therefore f \perp g$  since  $\{A, B\} \cap \{C, D\} = \phi$

13

## Weak Division - 2

We say that  $g$  divides  $f$  weakly if there exist  $h, r$  such that  $f = gh + r$  where  $h \neq \phi$  and  $g \perp h$

Example:  $f = ab + ac + d$   
 $g = b + c$   
 $f = a(b + c) + d$        $h = a$        $r = d$

We say that  $g$  divides  $f$  evenly if  $r = \phi$

The quotient  $f/g$  is the largest  $h$  such that  $f = gh + r$  i.e.,  $f = (f/g)g + r$

14

## Weak Division Example

$$f = abc + abde + abh + bcd$$

$$g = c + de + h$$

Theorem:  $f/g = f/c \cap f/de \cap f/h$

$$f/c = ab + bd$$

$$f/de = ab$$

$$f/h = ab$$

$$f/g = (ab + bd) \cap ab \cap ab = ab$$

$$f = ab(c + de + h) + bcd$$

Time complexity:  $O(|f|/|g|)$

15

## How to Find Good Divisors?

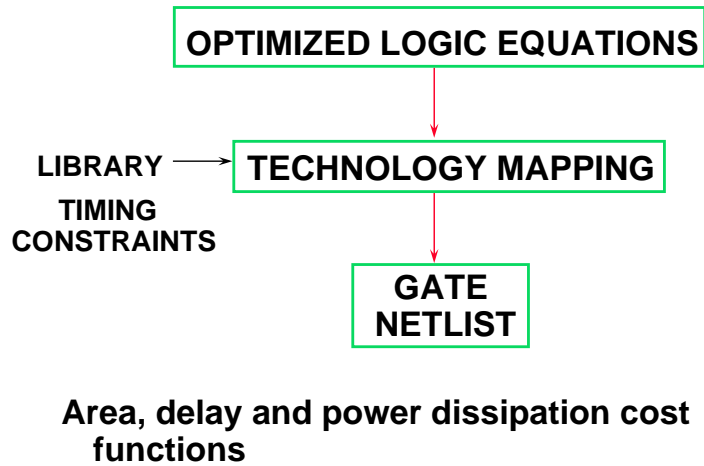
\$64K question

**Strong division:** Use existing nodes in the multilevel network to simplify other nodes

**Weak division:** Generate good algebraic divisors using algorithms based on “kernels” of an algebraic expression

16

## Tech.-Dependent Optimization

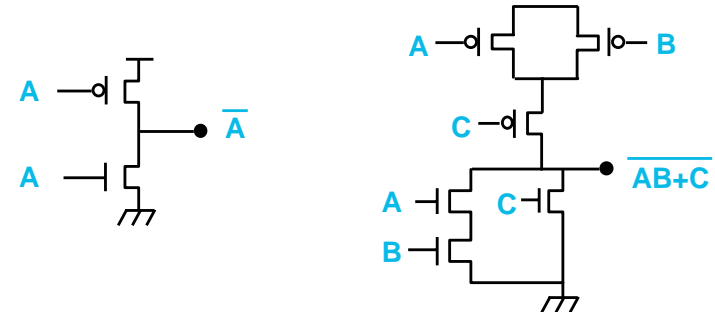


17

## “Closed Book” Technologies

A standard cell technology or library is typically restricted to a few tens of gates e.g., MSU library: 31 cells

Gates may be NAND, NOR, NOT, AOIs.



18

## Mapping via DAG Covering

Represent network in canonical form  
 $\Rightarrow$  subject DAG

Represent each library gate with canonical forms for the logic function  
 $\Rightarrow$  primitive DAGs

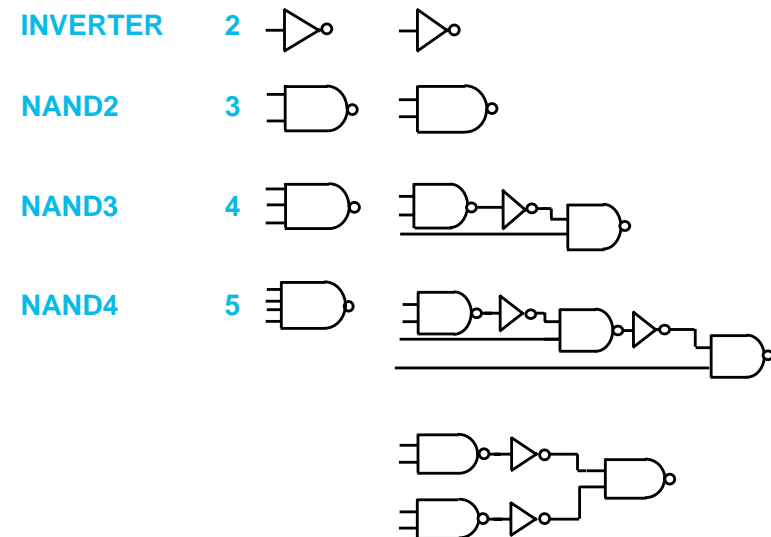
Each primitive DAG has a cost

Goal: Find a minimum cost covering of the subject DAG by the primitive DAGs

Canonical form: 2-input NAND gates and inverters

19

## Sample Library

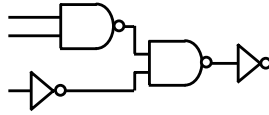


20

## Sample Library - 2

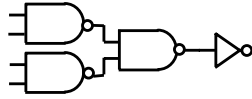
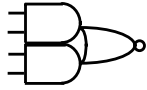
AOI21

4



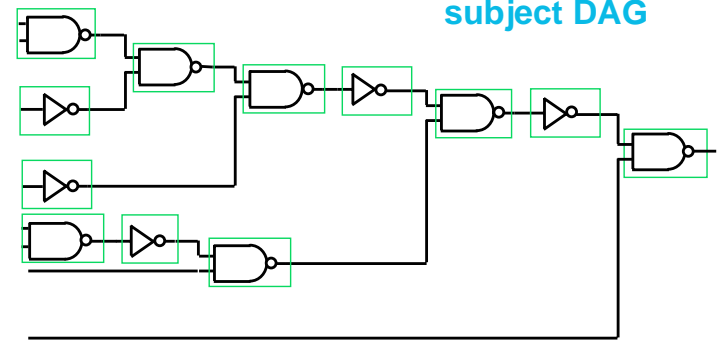
AOI22

5



21

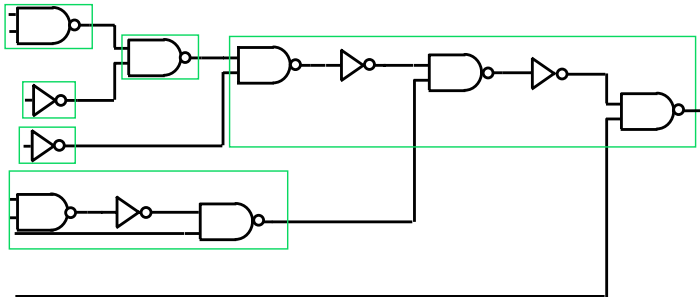
## Trivial Covering



$$\begin{array}{r}
 7 \\
 5 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 \text{NAND2} = 21 \\
 \text{INV} = 10 \\
 \hline
 31
 \end{array}$$

22

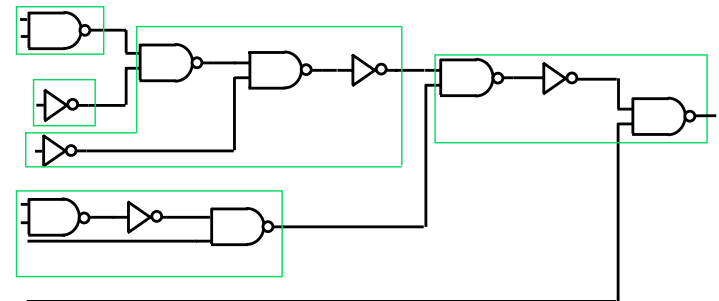
## Covering #1



$$\begin{array}{r}
 2 \text{ INV} \\
 2 \text{ NAND2} \\
 1 \text{ NAND3} \\
 1 \text{ NAND4} \\
 \hline
 19
 \end{array}
 \quad
 \begin{array}{r}
 = 4 \\
 = 6 \\
 = 4 \\
 = 5 \\
 \hline
 19
 \end{array}$$

23

## Covering #2

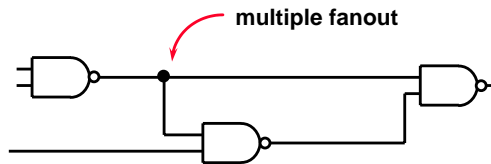


$$\begin{array}{r}
 1 \text{ INV} \\
 1 \text{ NAND2} \\
 2 \text{ NAND3} \\
 1 \text{ AOI21} \\
 \hline
 17
 \end{array}
 \quad
 \begin{array}{r}
 = 2 \\
 = 3 \\
 = 8 \\
 = 4 \\
 \hline
 17
 \end{array}$$

24

## DAG Covering

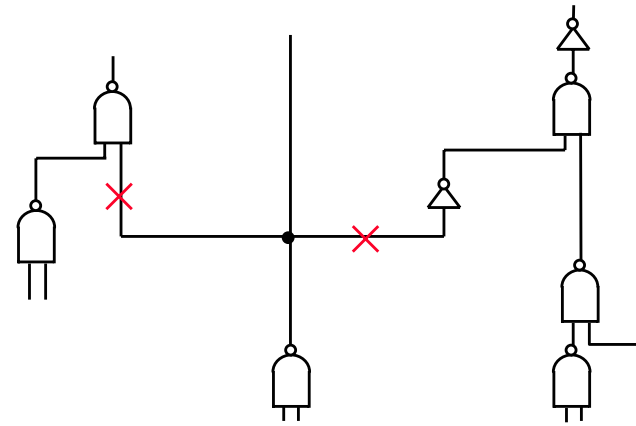
Sound Algorithmic approach  
NP-hard optimization problem



Tree covering heuristic: If subject and primitive DAGs are trees, efficient algorithm can find optimum cover in linear time  
⇒ dynamic programming formulation

25

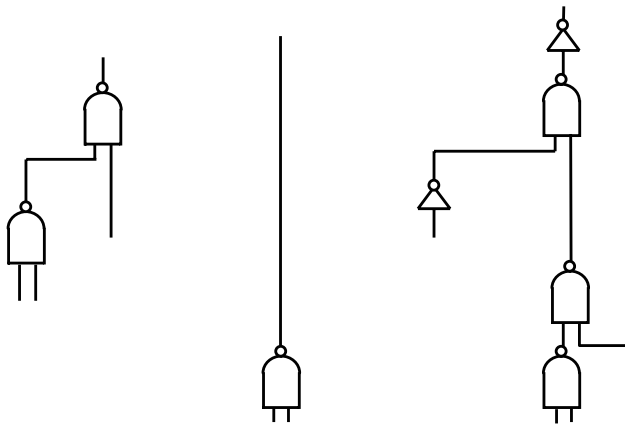
## Partitioning a Graph



26

## Resulting Trees

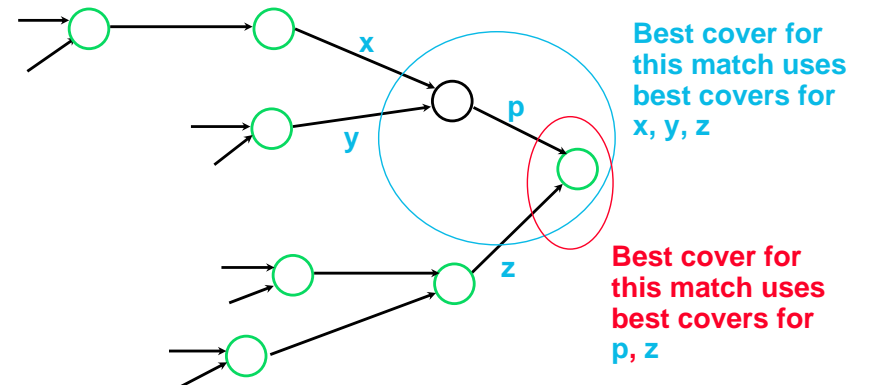
Break at multiple fanout points



27

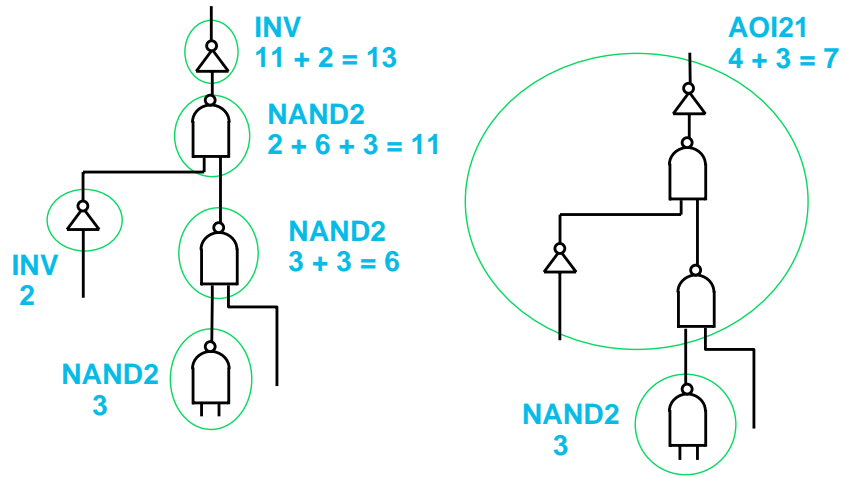
## Dynamic Programming

Principle of optimality: Optimal cover for a tree consists of a match at the root of the tree plus the optimal cover for the sub-trees starting at each input of the match



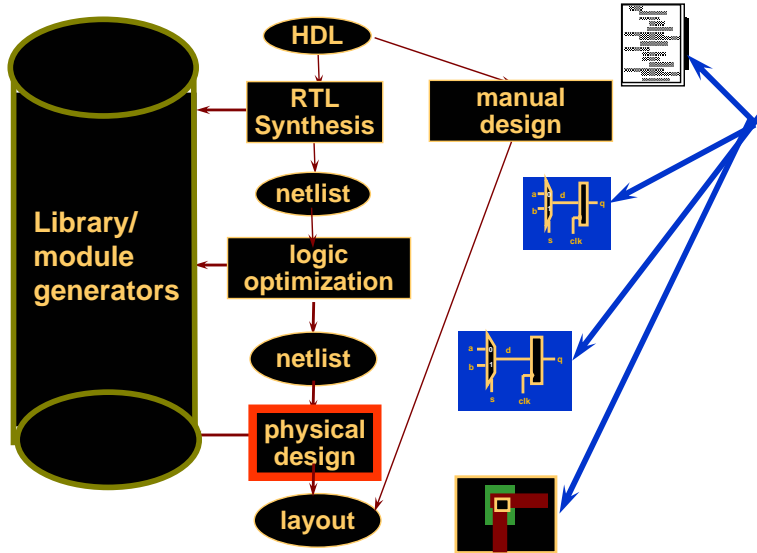
28

# Optimum Tree Covering

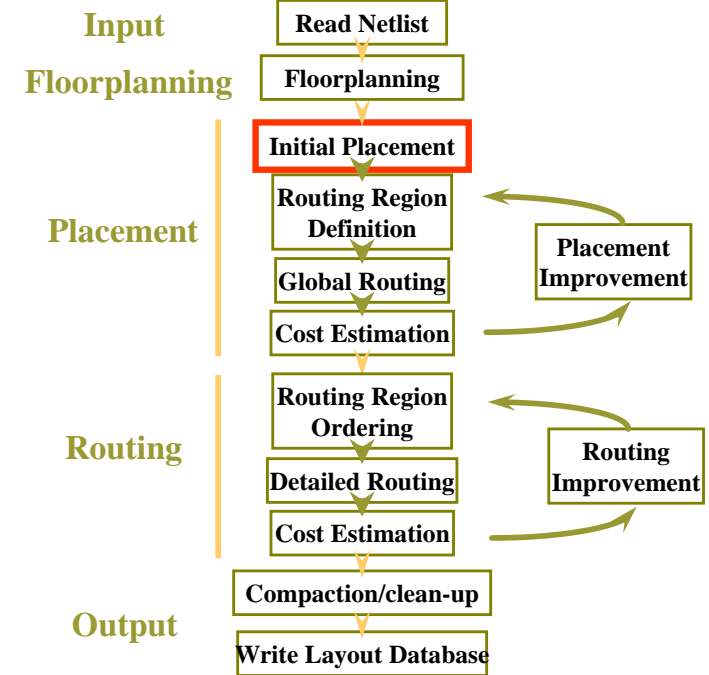




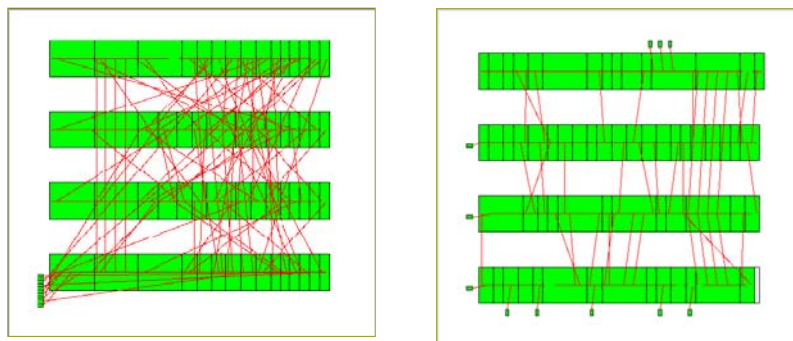
# RTL Design Flow



# Physical Design: Overall Conceptual Flow



# Results of Placement



A bad placement                      A good placement

What's good about a good placement?  
 What's bad about a bad placement?

# Results of Placement



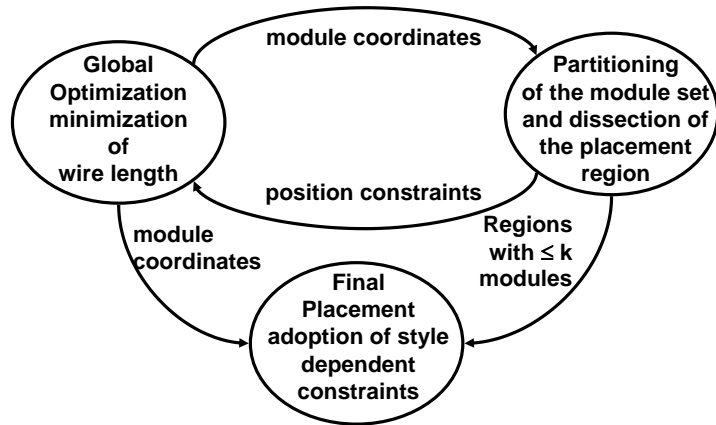
Bad placement causes routing congestion resulting in:

- Increases in circuit area (cost) and wiring
- Longer wires → more capacitance
  - Longer delay
  - Higher dynamic power dissipation

Good placement

- Circuit area (cost) and wiring decreases
- Shorter wires → less capacitance
  - Shorter delay
  - Less dynamic power dissipation

## Gordian Placement Flow



Data flow in the placement procedure GORDIAN

### Complexity

space:  $O(m)$  time:  $Q(m^{1.5} \log^2 m)$

### Final placement

•standard cell      •macro-cell & SOG

## Gordian: A Quadratic Placement Approach

- **Global optimization:** solves a sequence of quadratic programming problems
- **Partitioning:** enforces the non-overlap constraints

## Intuitive formulation

Given a series of points  $x_1, x_2, x_3, \dots, x_n$

and a connectivity matrix  $C$  describing the connections between them

(If  $c_{ij} = 1$  there is a connection between  $x_i$  and  $x_j$ )

Find a location for each  $x_j$  that minimizes the total sum of all spring tensions between each pair  $\langle x_i, x_j \rangle$



Problem has an obvious (trivial) solution – what is it?

## Improving the intuitive formulation

To avoid the trivial solution add constraints:  $Hx=b$

- These may be very natural - e.g. endpoints (pads)



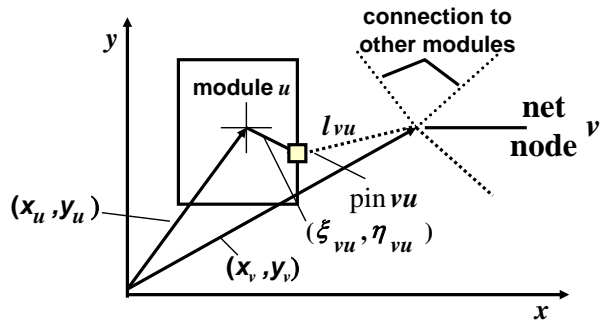
To integrate the notion of "critical nets"

- Add weights  $w_{ij}$  to nets



$w_{ij}$  - some springs have more tension should pull associated vertices closer

# Modeling the Net's Wire Length



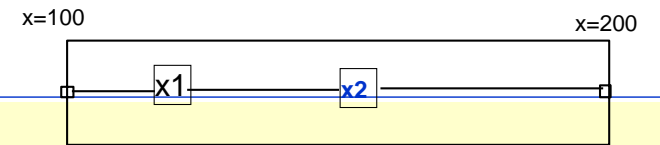
The length  $L_v$  of a net  $v$  is measured by the squared distances from its points to the net's center

$$L_v = \sum_{u \leftarrow M_v} [(x_{uv} - x_v)^2 + (y_{uv} - y_v)^2]$$

$$(x_{uv} = x_u + \xi_{uv} ; y_{uv} = y_u + \eta_{vu})$$

# Toy

## Example:



$$Cost = (x_1 - 100)^2 + (x_1 - x_2)^2 + (x_2 - 200)^2$$

$$\frac{\partial Cost}{\partial x_1} = 2(x_1 - 100) + 2(x_1 - x_2)$$

$$\frac{\partial Cost}{\partial x_2} = -2(x_1 - x_2) + 2(x_2 - 200)$$

setting the partial derivatives = 0 we solve for the minimum Cost:

$$Ax + B = 0$$

$$\begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -200 \\ -400 \end{bmatrix} = 0$$

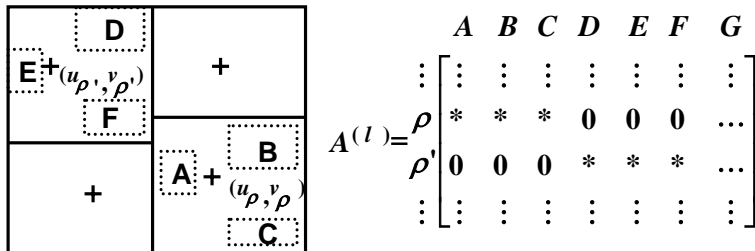
$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -100 \\ -200 \end{bmatrix} = 0$$

$x_1 = 400/3$     $x_2 = 500/3$

Kurt Keutzer

D. Pan

# Quadratic Optimization Problem



- Linearly constrained quadratic programming problem

$$\min_{x \in R^m} \{ \Phi(x) = x^T C x + d^T x \}$$

Accounts for fixed modules

Wire-length for movable modules

$$\text{s.t. } A^{(l)} x = u^{(l)}$$

Center-of-gravity constraints

Problem is computationally tractable, and well behaved

Commercial solvers available: mostek

# Global Optimization Using Quadratic Placement

Quadratic placement clumps cells in center

Partitioning divides cells into two regions

- Placement region is also divided into two regions

New center-of-gravity constraints are added to the constraint matrix to be used on the next level of global optimization

- Global connectivity is still conserved

## Setting up Global Optimization

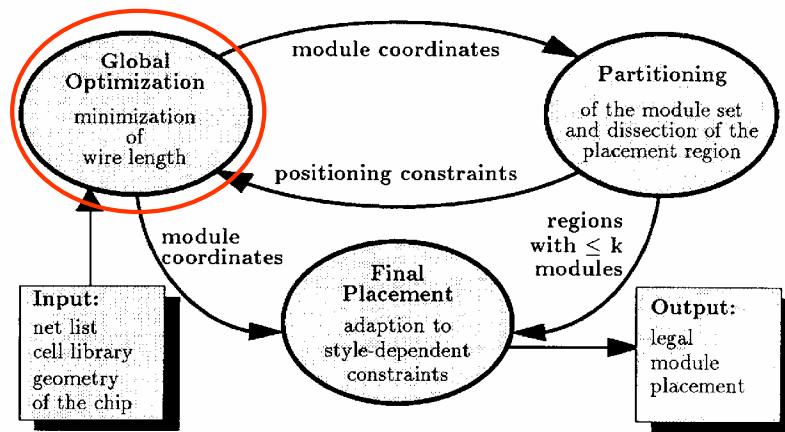
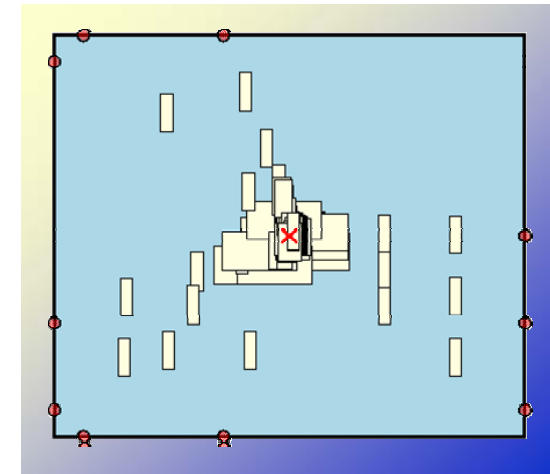


Fig. 1. Data flow in the placement procedure GORDIAN.

## Layout After Global Optimization



A. Kahng

## Partitioning

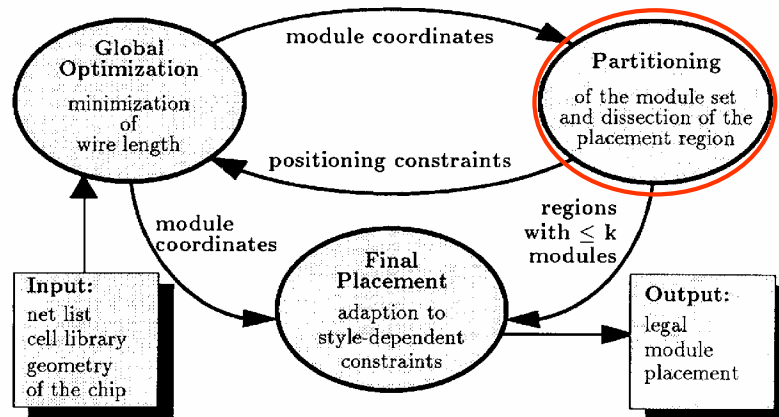


Fig. 1. Data flow in the placement procedure GORDIAN.

## Partitioning

In GORDIAN, partitioning is used to constrain the movement of modules rather than reduce problem size

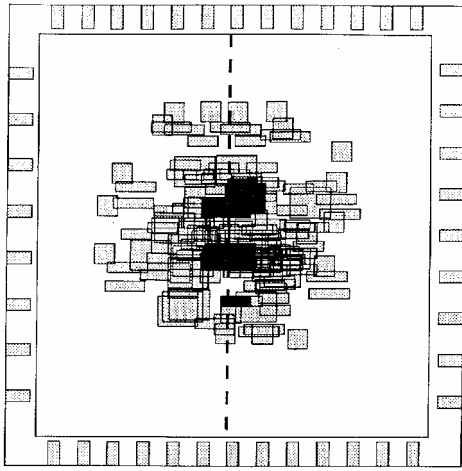
By performing partitioning, we can iteratively impose a new set of constraints on the global optimization problem

- Assign modules to a particular block

Partitioning is determined by

- Results of global placement – initial starting point
  - Spatial (x,y) distribution of modules
- Partitioning cost
  - Want a min-cut partition

## Layout after Min-cut



Now global placement problem will be solved again with two additional center\_of\_gravity constraints

## Adding Positioning Constraints

- Partitioning gives us two new “center of gravity” constraints

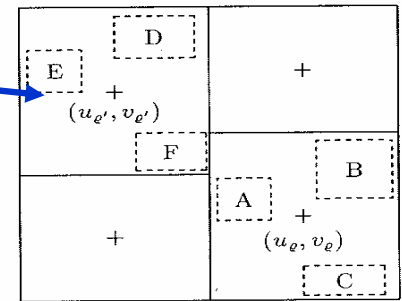
- Simply update constraint matrix

- Still a single global optimization problem

- Partitioning is not “absolute”

- modules can migrate back during optimization

- may need to re-partition



$$A^{(t)} = \begin{matrix} & A & B & C & D & E & F & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \varrho & * & * & * & 0 & 0 & 0 & \dots \\ \varrho' & 0 & 0 & 0 & * & * & * & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{matrix}$$

Fig. 4. The constraints for global placement.

## Continue to Iterate

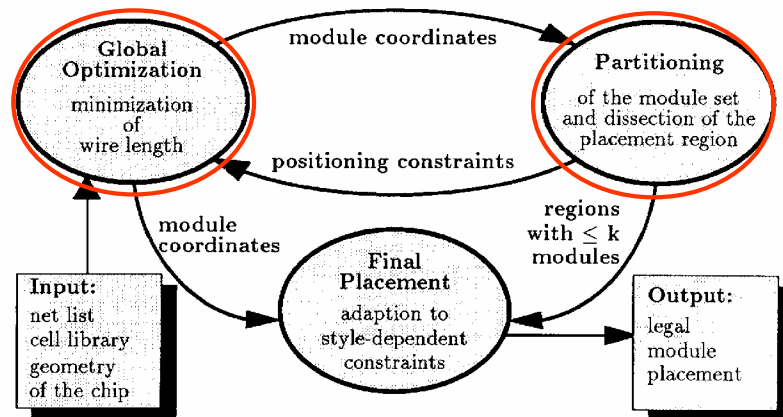
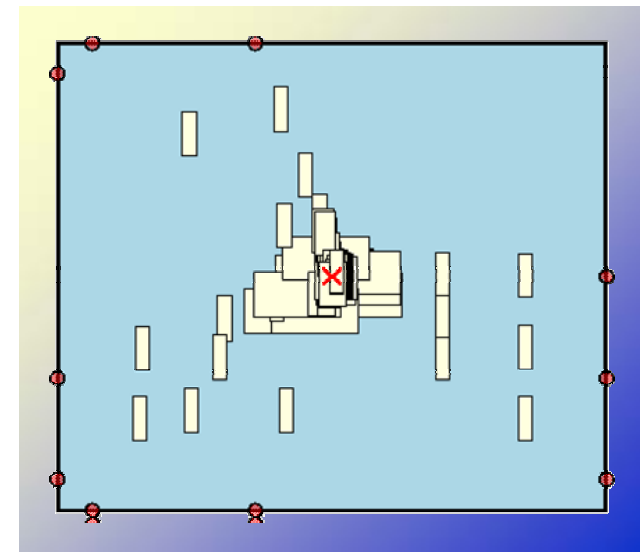
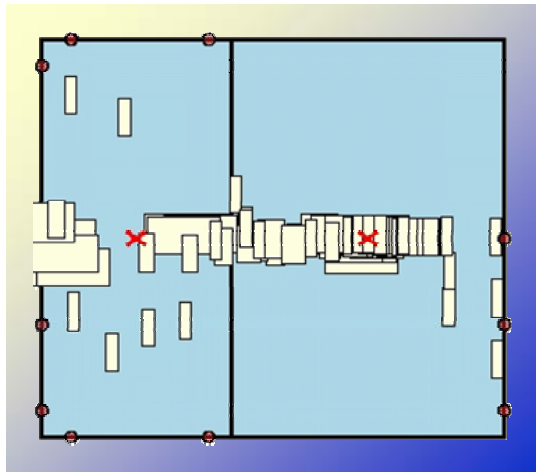


Fig. 1. Data flow in the placement procedure GORDIAN.

## First Iteration



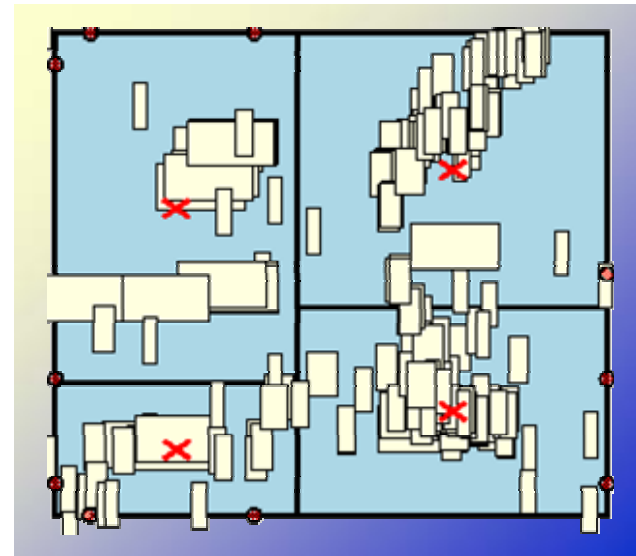
## Second Iteration



A. Kahng<sub>21</sub>

Kurt Keutzer

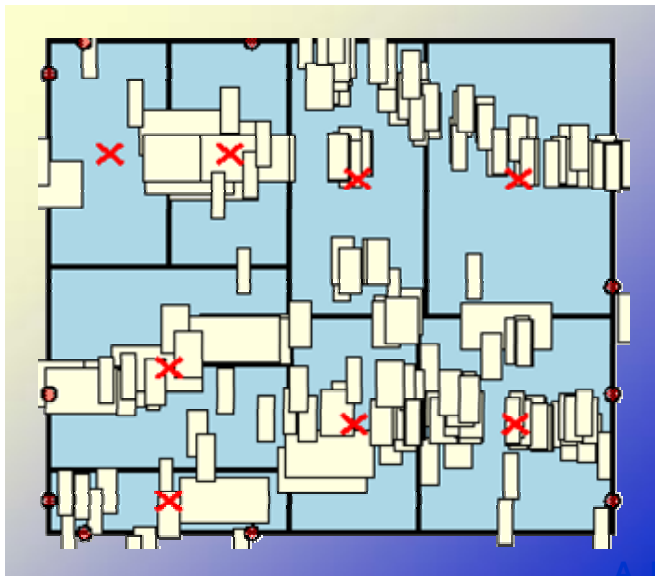
## Third Iteration



A. Kahng<sub>22</sub>

Kurt Keutzer

## Fourth Iteration



A. Kahng<sub>23</sub>

Kurt Keutzer

## Final Placement

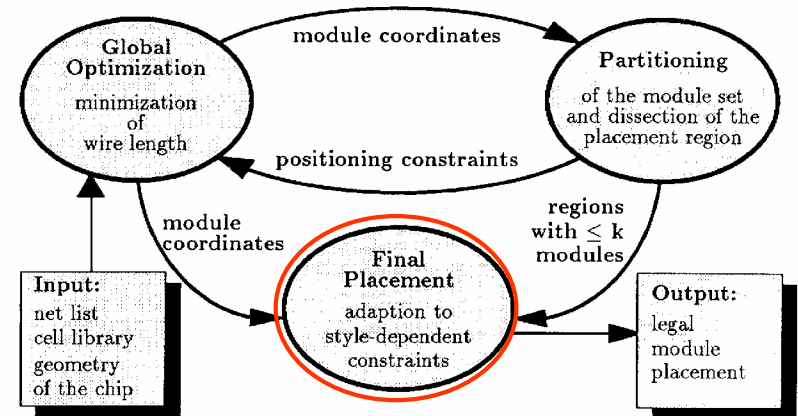


Fig. 1. Data flow in the placement procedure GORDIAN.

## Final Placement - 1

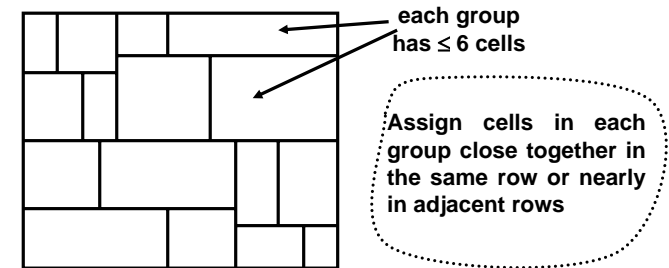
Earlier steps have broken down the problem into a manageable number of objects

Two approaches:

- Final placement for standard cells/gate array – row assignment
- Final placement for large, irregularly sized macro-blocks – slicing – won't talk about this

## Final Placement – Standard Cell Designs

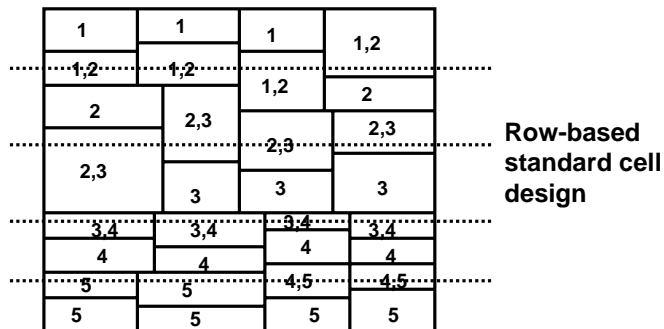
This process continues until there are only a few cells in each group ( $\approx 6$ )



group: smallest partition

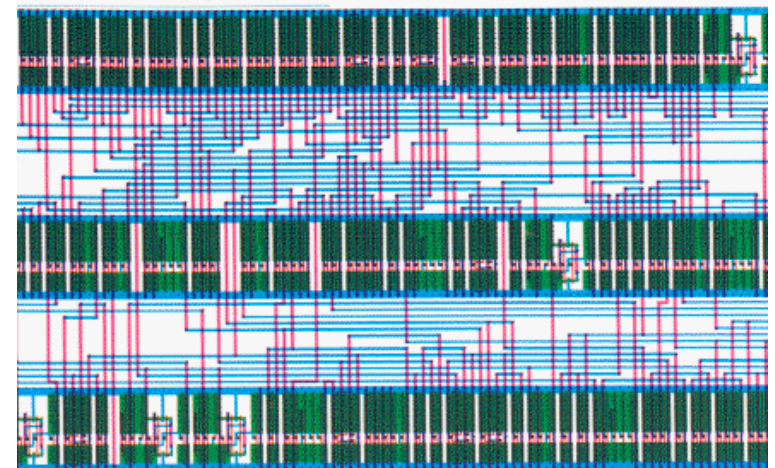
A. E. Dunlop, B. W. Kernighan,  
*A procedure for placement of standard-cell VLSI circuits*, IEEE Trans. on CAD, Vol. CAD-4, Jan , 1985, pp. 92- 98

## Final Placement – Creating Rows



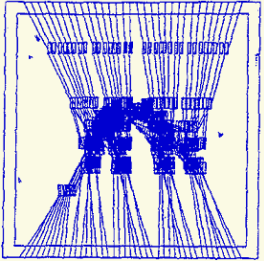
Partitioning of circuit into 32 groups. Each group is either assigned to a single row or divided into 2 rows

## Standard Cell Layout

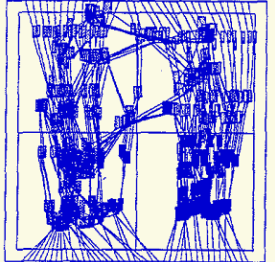


## Another Series of Gordian

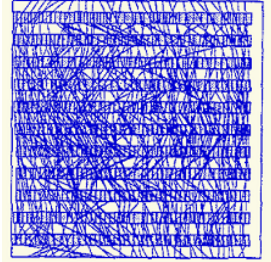
---



(a) Global placement with 1 region



(b) Global placement with 4 region

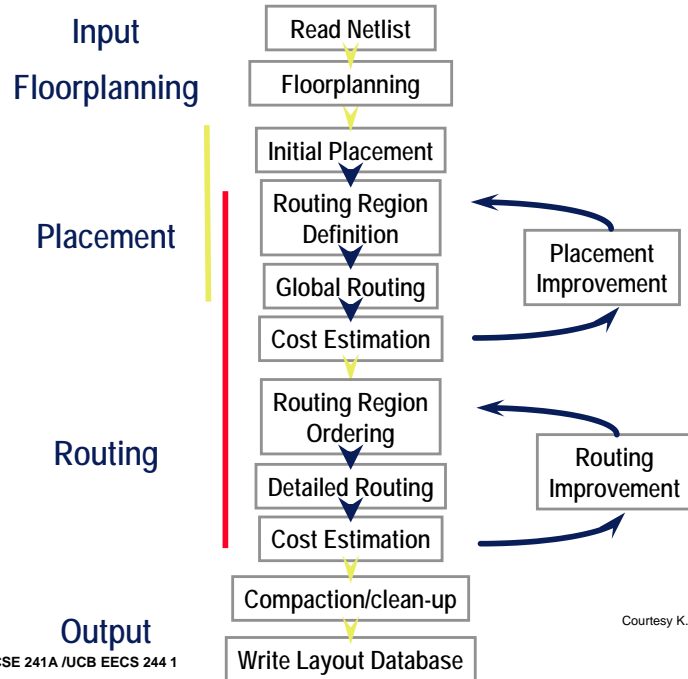


(c) Final placements

D. Pan – U of Texas



# Physical Design Flow



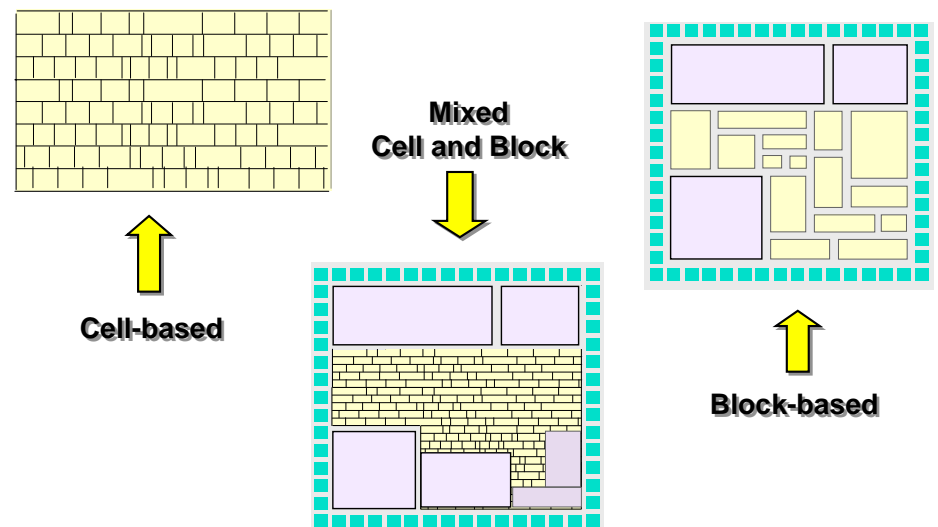
# Imagine ...

- You have to plan transportation (i.e. roads and highways) for a *new* city the size of Chicago
- Many dwellings need direct roads that can't be used by anyone else
- You can affect the layout of houses and neighborhoods but the architects and planners will complain
- And ... you're told that the time along any path can't be longer than a fixed amount
- What are some of your considerations?

# What are some of your considerations?

- How many levels do my roads need to go? Remember: Higher is more expensive.
- How do I avoid congestion?
- What basic structure do I want for my roads?
  - Manhattan?
  - Chicago?
  - Boston?
- Automated route tools have to solve problems of comparable complexity on every leading edge chip

# Routing Applications



## Routing Algorithms

Hard to tackle high-level issues like congestion and wire-planning and low level details of pin-connection at the same time

### Global routing

- Identify routing resources to be used
- Identify layers (and tracks) to be used
- Assign particular nets to these resources
- Also used in floorplanning and placement

### Detail routing

- Actually define pin-to-pin connections
- Must understand most or all design rules
- May use a compactor to optimize result
- Necessary in all applications

Kahng/Keutzer/Newton

ECE 260B – CSE 241A /UCB EECS 244 5

## Basic Rules of Routing - 1

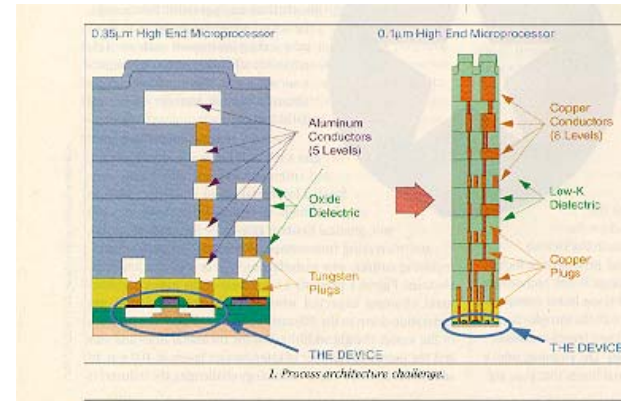


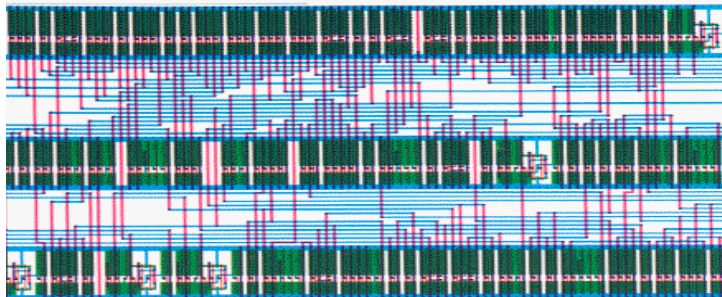
Photo courtesy:  
Jan M. Rabaey  
Anantha Chandrakasan  
Borivoje Nikolic

ECE 260B – CSE 241A /UCB EECS 244 6

- Wiring/routing performed in layers – 5-9 (-11), typically only in “Manhattan” N/S E/W directions
  - E.g. layer 1 – N/S
  - Layer 2 – E/W
- A segment cannot cross another segment on the same wiring layer
- Wire segments *can* cross wires on other layers
- Power and ground may have their own layers

Kahng/Keutzer/Newton

## Basic Rules of Routing – Part 2

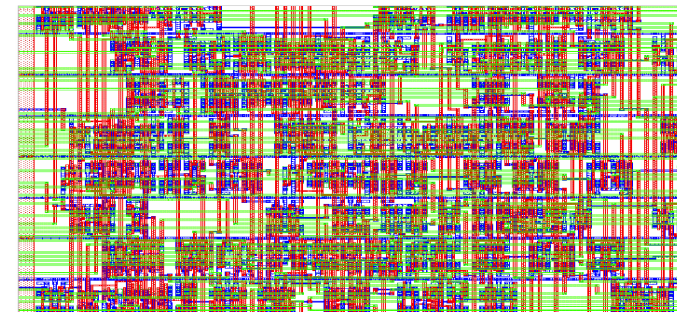


- Routing can be on a fixed grid –
- Case 1: Detailed routing only in channels
  - Wiring can only go over a row of cells when there is a free track – can be inserted with a “feedthrough”
  - Design may use of metal-1, metal-2
  - Cells *must* bring signals (i.e. inputs, outputs) out to the channel through “ports” or “pins”

Kahng/Keutzer/Newton

ECE 260B – CSE 241A /UCB EECS 244 7

## Basic Rules of Routing – Part 3

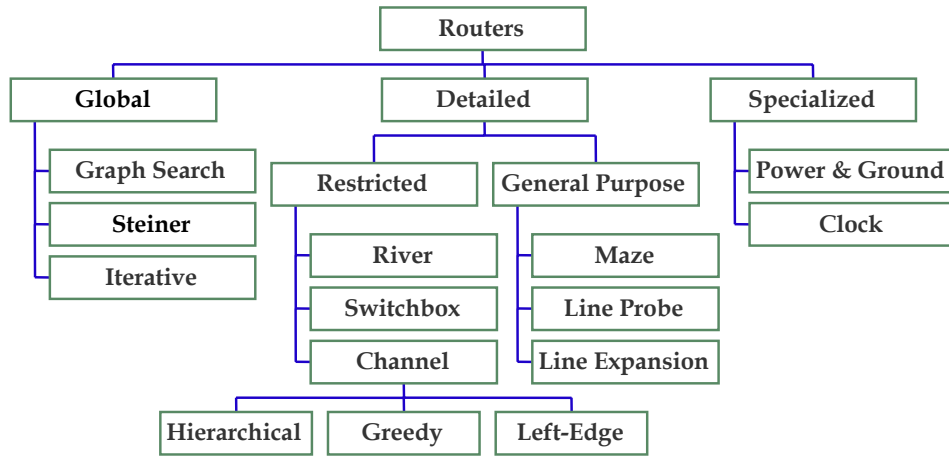


- Routing can be on a fixed or gridless (aka area routing)
- Case 1: Detailed routing over cells
  - Wiring can go over cells
  - Design of cells must try to minimize obstacles to routing – i.e. minimize use of metal-1, metal-2
  - Cells *do not* need to bring signals (i.e. inputs, outputs) out to the channel – the route will come to them

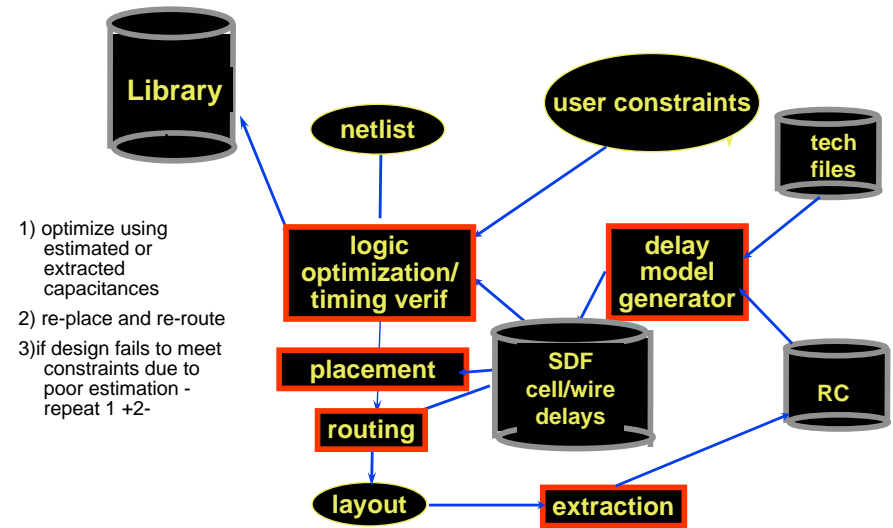
Kahng/Keutzer/Newton

ECE 260B – CSE 241A /UCB EECS 244 8

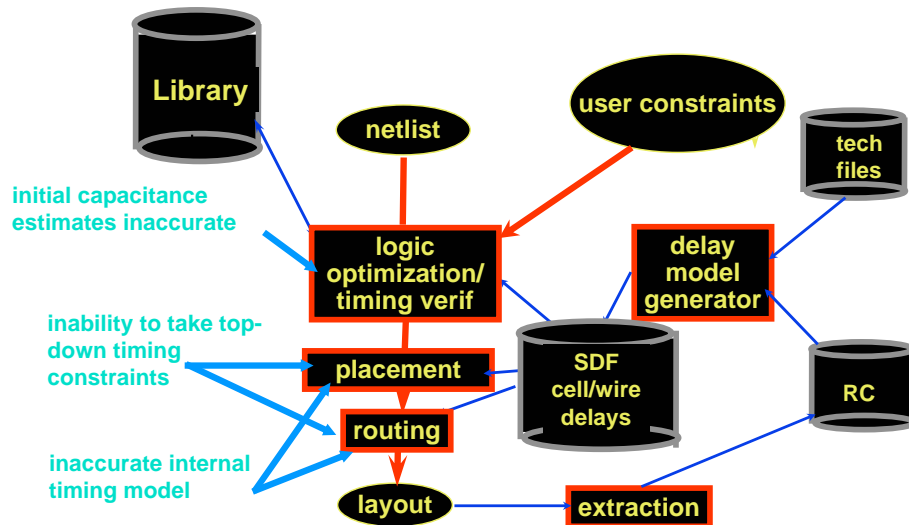
# Taxonomy of VLSI Routers



# Today's high-perf logical/physical flow



# Top-down problems in the flow



# Iteration problems in the flow

