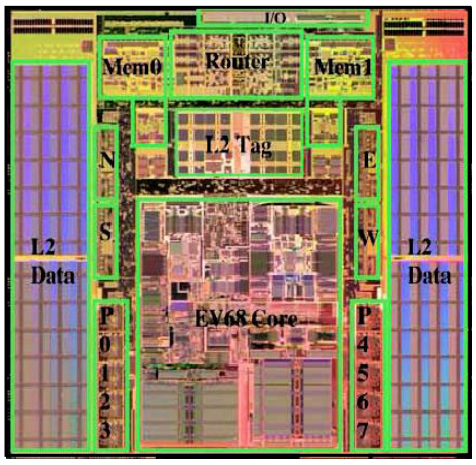
Floorplanning

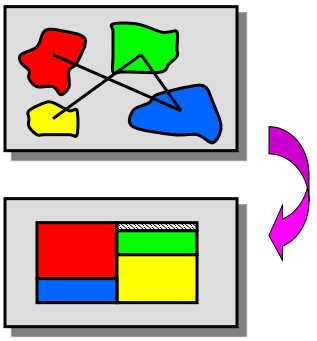
An Example Floorplan

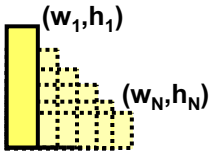
- Alpha 21364



Floorplanning

- Problem
  - Given circuit modules (or cells) and their connections, determine the *approximate* location of circuit elements
  - Consistent with a hierarchical / building block design methodology
  - Modules (result of partitioning):
    - Fixed area, generally rectangular
    - Fixed aspect ratio → hard macro (aka fixed-shaped blocks)
      fixed / floating terminals (pins)
      Rotation might be allowed / denied
    - Flexible shape → soft macro (aka soft modules)





[Bazargan]

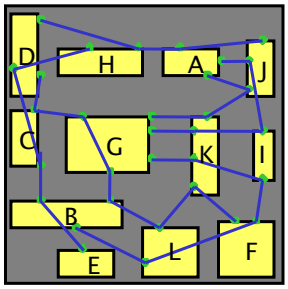
Floorplanning (cont.)

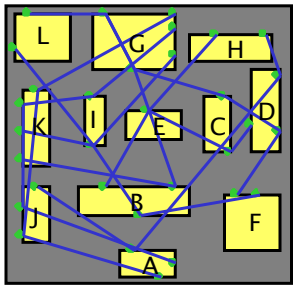
- Objectives:
  - Minimize area
  - Determine best shape of soft modules
  - Minimize total wire length
    - to make subsequent routing phase easy
      (short wire length roughly translates into routability)
  - Additional cost components:
    - Wire congestion (exact routability measure)
    - Wire delays
    - Power consumption
    - System throughput (e.g., CPI of a processor)
- Possible additional constraints:
  - Fixed location for some modules
  - Fixed die, or range of die aspect ratio

[Bazargan]

Floorplanning: Why Important?

- Early stage of physical design
  - Determines the location of large blocks
    ➔ detailed placement easier (divide and conquer!)
  - Estimates of area, delay, power
    ➔ important design decisions
  - Impact on subsequent design steps (e.g., routing, heat dissipation analysis and optimization)



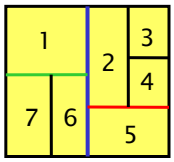


Figs: [©Sherwani]

[Bazargan]

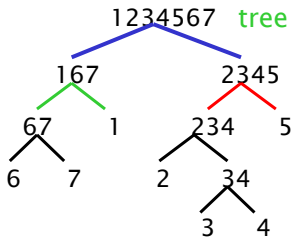
Floorplan Classes

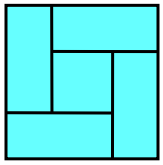
- Slicing, recursively defined as:
  - A module OR
  - A floorplan that can be partitioned into two slicing floorplans with a horizontal or vertical cut line
- Non-slicing
  - Superset of slicing floorplans
  - Contains the "wheel" shape too.



Slicing floorplan

Corresp. Slicing tree





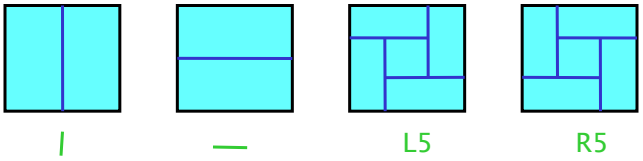
Non-Slicing floorplan

[©Sarrafzadeh]

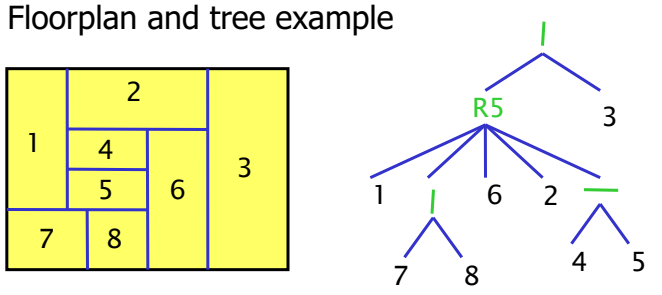
[Bazargan]

Non-slicing Floorplan Example

- Hierarchical floorplan of order 5
  - Templates



  - Floorplan and tree example



[©Sarrafzadeh]

[Bazargan]

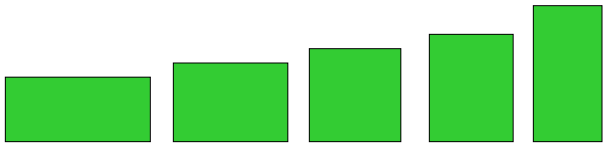
Floorplanning Algorithms

- Components
  - "Placeholder" representation
    - Usually in the form of a tree
    - Slicing class: Polish expression [Otten]
    - Non-slicing class: O-tree, Sequence Pair, BSG, etc.
    - Just defines the *relative position* of modules
  - Perturbation
    - Going from one floorplan to another
    - Usually done using Simulated Annealing
  - Floorplan sizing
    - Definition: Given a floorplan tree, choose the best shape for each module to minimize area
    - Slicing: polynomial, bottom-up algorithm
    - Non-slicing: NP! Use mathematical programming (exact solution)
  - Cost function
    - Area, wire-length, ...

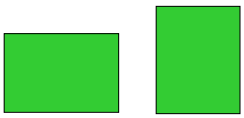
[Bazargan]

Bounds on Aspect Ratios

- We can also allow several shapes for each block:



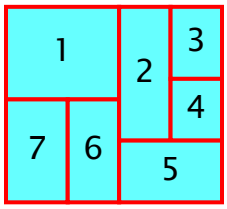
- For hard blocks, the orientations can be changed:

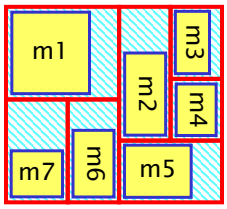


[Pan]

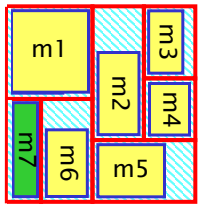
Area Utilization, Hard and Soft Modules

- The hierarchy tree and floorplan define "place holders" for modules
- Area utilization
  - Depends on how nicely the rigid modules' shapes are matched
  - Soft modules can take different shapes to "fill in" empty slots ➔ floorplan sizing





Area = 20x22 = 440



Area = 20x19 = 380

[Bazargan]

Bounds on Aspect Ratios

If there is no bound on the aspect ratios, can we pack everything tightly?

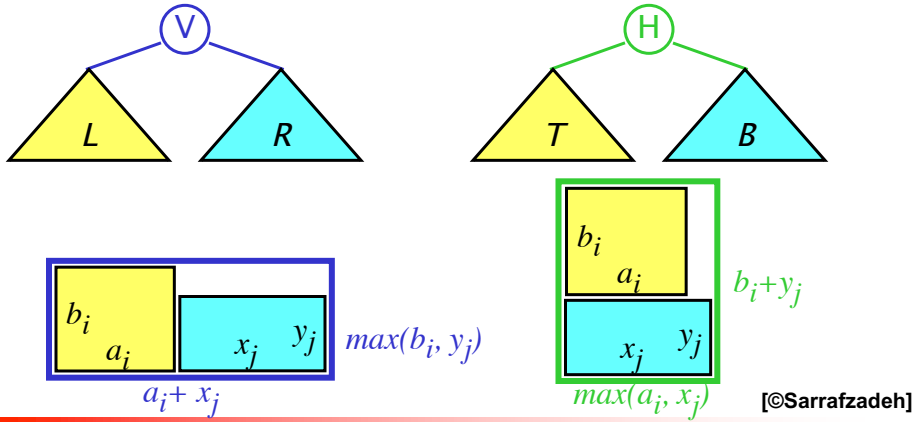
- Sure!

But we don't want to layout blocks as long strips, so we require $r_i \leq h_i/w_i \leq s_i$ for each i.

[Pan]

Floorplan Sizing for Slicing Floorplans

- Bottom-up process
- Has to be done per floorplan perturbation
- Requires O(n) time.
  - n is the total number of shapes of all the modules

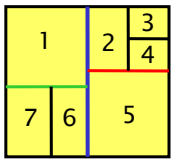


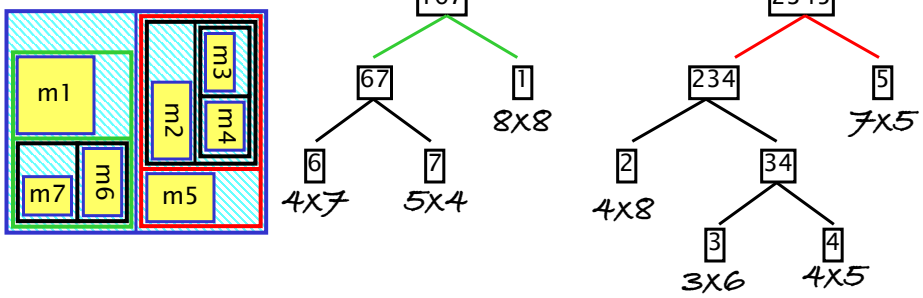
[©Sarrafzadeh]

[Bazargan]

Sizing Slicing Floorplans

- Simple case:
  - All modules are hard macros
  - No rotation allowed
    ➔ one shape only

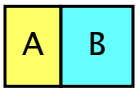




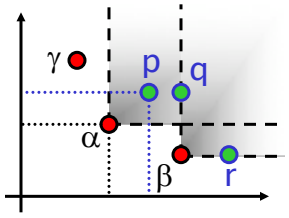
[Bazargan]

Sizing Slicing Floorplans (cont.)

- What if modules have more than one shape?
- If area only concern:
  - Module A has shapes 4x6, 7x8, 5x6, 6x4, 7x4, which ones should we pick?
  - Module A has shapes 4x6, 5x5, 6x4, which ones should we pick?

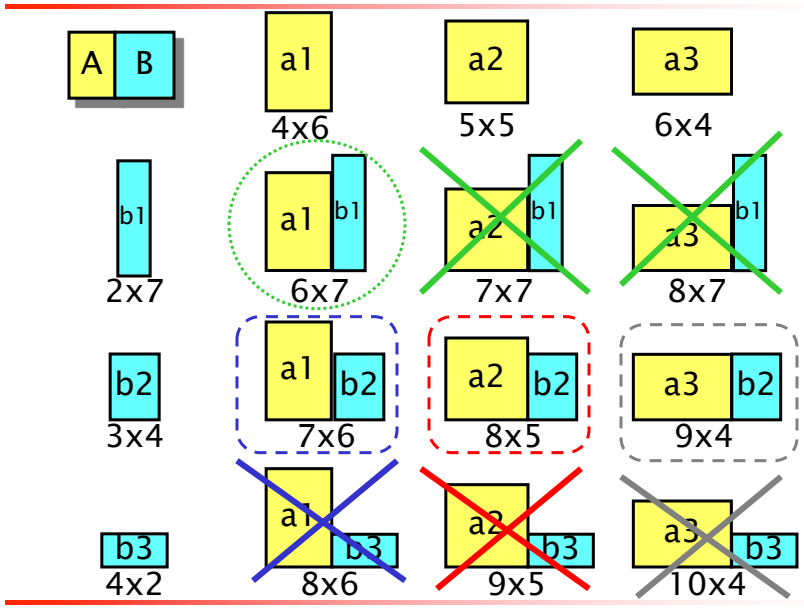


- Dominant points
  - Shape (x_1, y_1) dominates (x_2, y_2) if $x_1 \le x_2$ and $y_1 \le y_2$.



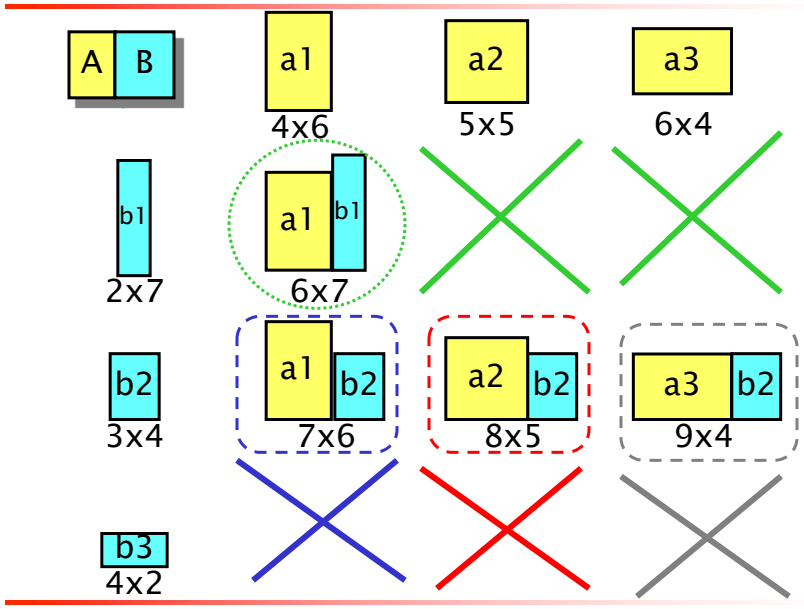
[Bazargan]

Sizing Slicing Floorplans: Poor Implementation



[Bazargan]

Sizing Slicing Floorplans: REAL computation



[Bazargan]

Slicing Floorplan Sizing Algorithm

Procedure Vertical_Node_Sizing

Input: Two sorted lists L = { $(a_1, b_1), \ldots, (a_s, b_s)$ },
R = { $(x_1, y_1), \ldots, (x_t, y_t)$ }
where $a_i < a_j$, $b_i > b_j$, for all $i < j$; $x_i < x_j$, $y_i > y_j$ for all $i < j$
Output: A sorted list H = { $(c_1, d_1), \ldots, (c_u, d_u)$ }
where $u \leq s + t - 1$, $c_i < c_j$, $d_i > d_j$ for all $i < j$

```
begin
        H := ϕ
        i := 1, j := 1, k = 1
        while (i ≤ s) and (j ≤ t) do
        begin
                (c_k, d_k) := (a_i + x_j, max(b_i, y_j))
                H := H ∪ { (c_k, d_k) }
                k := k + 1
                if  max(b_i, y_j) = b_i then  i := i + 1
                if  max(b_i, y_j) = y_j then  j := j + 1
        end
end
```

[©Sarrafzadeh]

[Bazargan]

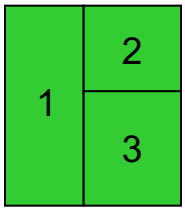
Slicing Floorplan Sizing

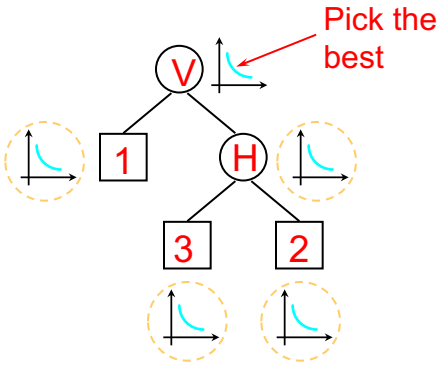
- Input: floorplan tree, modules shapes
- Start with sorted shapes lists of modules
- In a bottom-up fashion, perform:
  - Vertical_Node_Sizing
    AND
    Horizontal_Node_Sizing
- When get to the root node, we have a list of shapes. Select the one that is best in terms of area
- In a top-down fashion, traverse the floorplan tree and set module locations

[Bazargan]

Find the Best Area

- Recursively combining shape curves.

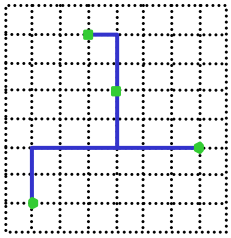




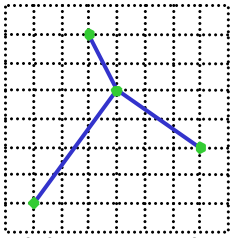
[Pan]

Wire Length

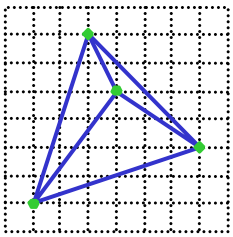
- For hyperedges:
  - Either of complete graph, MST, or Steiner tree



(a) Steiner tree (length = 13)



(b) minimum spanning tree (length = 11)



(c) complete graph (length = 32)

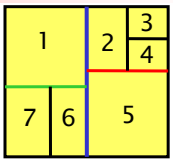
- For each edge:
  - Euclidian distance sqrt($(x_1-x_2)^2 + (y_1-y_2)^2$).
    - Direct lines
  - Manhattan distance $|x_1 - x_2| + |y_1 - y_2|$
    - Manhattan: Only horizontal / vertical lines

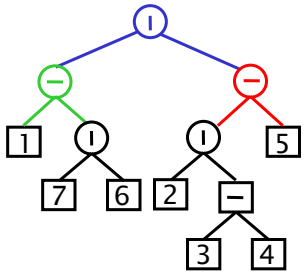
[©Sherwani]

[Bazargan]

Polish Expression

- Tree representation of the floorplan
  - Left child of a V-cut in the tree represents the left slice in the floorplan
  - Left child of an H-cut in the tree represents the top slice in the floorplan
- Polish expression representation
  - A string of symbols obtained by traversing a binary tree in post-order.



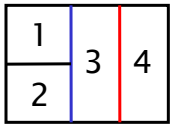


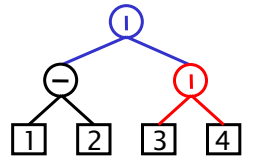
1 7 6 | – 2 3 4 – | 5 – |

[Bazargan]

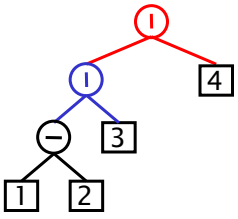
Normalized Polish Expression

- Problem with Polish expressions?
  - Multiple representations for some slicing trees
    - When more than one cut in one direction cut a floorplan
  - Larger solution space
  - A stochastic algorithm (e.g., Simulated Annealing) will be more biased towards floorplans with multiple representations
    - (More likely to be visited)





1 2 – 3 4 | |



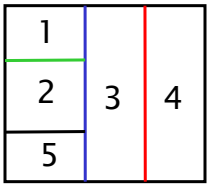
1 2 – 3 | 4 |

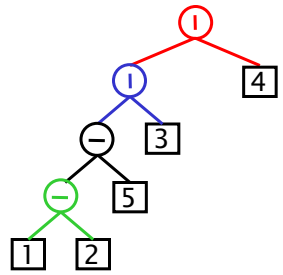
[©Sarrafzadeh]

[Bazargan]

Normalized Polish Expression (cont.)

- Solution?
  - Assign priorities to the cuts
  - In a top-down tree construction,
    - Pick the right-most cut
    - Pick the lowest cut
  - Result: no two same operators adjacent in the Polish expression (i.e., no "| |" or "— —")





1 2 – 5 – 3 | 4 |

[Bazargan]

Simulated Annealing

- Idea originated from observations of crystal formations (e.g., in lava)
  - A crystal is in a low energy state
  - Materials tend to form crystals (global minimum)
  - If at the right temperature (i.e., right speed), a molecule will adhere to a crystal formation
- Very slowly decrease temperature
  - When very hot, molecules move freely
    - When a molecule gets to a chunk of crystal, it *might* move away due to its high speed
  - When colder, molecules slow down
    - The probability of moving away from a local optimum decreases
  - When the material "freezes", all molecules are fixed and the material is in minimum energy state

[Bazargan]

Simulated Annealing Algorithm

- Components:
  - Solution space (e.g., slicing floorplans)
  - Cost function (e.g., the area of a floorplan)
    - Determines how "good" a particular solution is
  - Perturbation rules
    (e.g., transforming a floorplan to a new one)
  - Simulated annealing engine
    - A variable T, analogous to temperature
    - An initial temperature T_0 (e.g., T_0 = 40,000)
    - A freezing temperature T_{freez} (e.g., T_{freez}=0.1)
    - A cooling schedule (e.g., T = 0.95 * T)

[Bazargan]

Simulated Annealing Algorithm

```
Procedure SimulatedAnnealing
    curSolution = random initial solution
    T = T0                          // initial temperature
    while (T > Tfreez) do
        for i=1 to NUM_MOVES_PER_TEMP_STEP do
            nextSol = perturb (curSolution)
            Δcost = cost(nextSol) – cost(curSolution)
            if acceptMove (Δcost, T) then
                curSolution = nextSol      // accept the move
        T = coolDown (T )
```

```
Procedure acceptMove (Δcost, T)
    if Δcost < 0 then return TRUE      // always accept a good move
    else
        boltz = e^(−Δcost / k T)       // Boltzmann probability function
        r = random(0,1)                // uniform rand # between 0&1
        if r < boltz then return TRUE
        else return FALSE
```

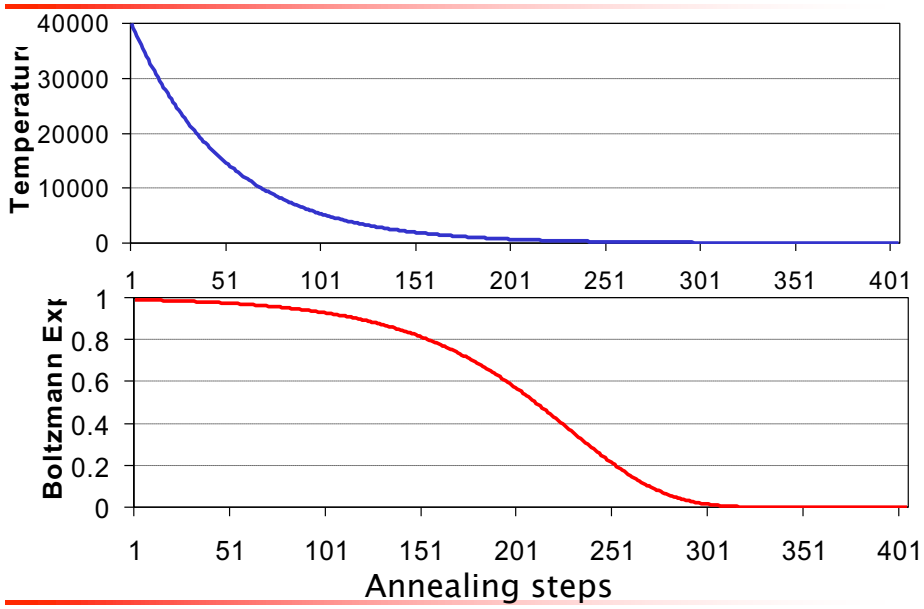
[Bazargan]

Simulated Annealing: Move Acceptance

- Good moves are always accepted
- Accepting bad moves:
  - When T = T_0, bad move acceptance probability ≈ 1
  - When T = T_{freez}, Bad move acceptance probability = 0
- Boltzmann probability function?!?
  - boltz = $e^{-\Delta cost / k T}$.
  - k is the Boltzmann constant, chosen so that all moves at the initial temperature are accepted

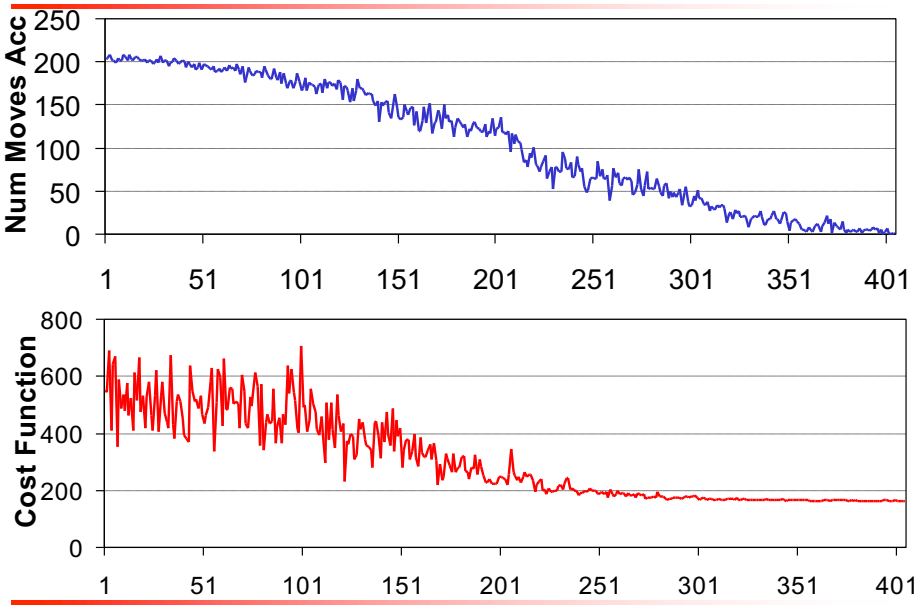
[Bazargan]

Simulated Annealing: More Insight...



[Bazargan]

Simulated Annealing: More Insight...



[Bazargan]

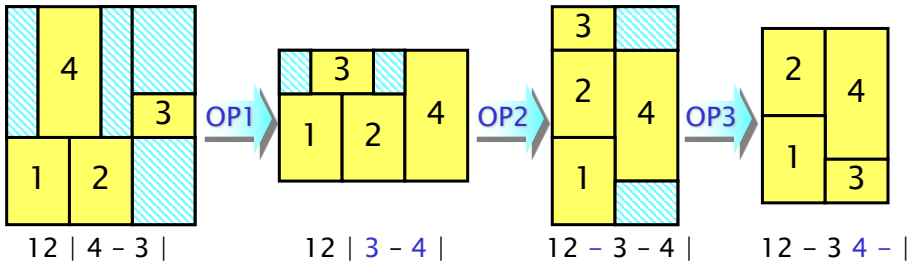
Wong-Liu Floorplanning Algorithm

- Uses simulated annealing
- Normalized Polish expressions represent floorplans
- Cost function:
  - cost = area + λ totalWireLength
  - Floorplan sizing is used to determine area
  - After floorplan sizing, the exact location of each module is known, hence wire-length can be calculated

[Bazargan]

Wong-Liu Floorplanning Algorithm (cont.)

- Moves:
  - OP1: Exchange two operands that have no other operands in between
  - OP2: Complement a series of operators between two operands
  - OP3: Exchange adjacent operand and operator if the resulting expression still a normalized Polish exp.



[Bazargan]

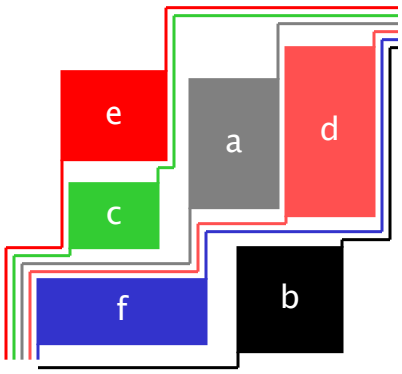
[©Sarrafzadeh]

The Sequence Pair Algorithm

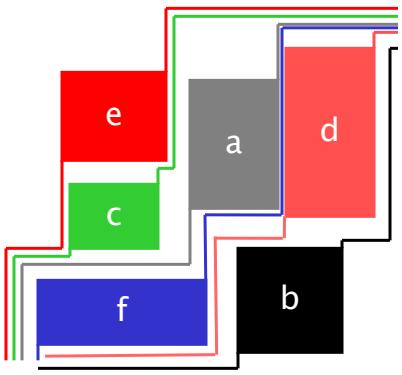
- Sequence-Pair is a succinct representation of non-slicing floorplans of rectangles
  - Just like Polish Expression for slicing floorplans
- Represent a non-slicing floorplan by a pair of sequences of blocks
- Using Simulated Annealing to find a good sequence-pair
- Can only handle hard blocks
  - i.e., cannot do things like shape-curve computation
- Essentially macro placement
- Techniques for soft block shaping exist (e.g., using Lagrangian Relaxation) but are very slow

[Pan]

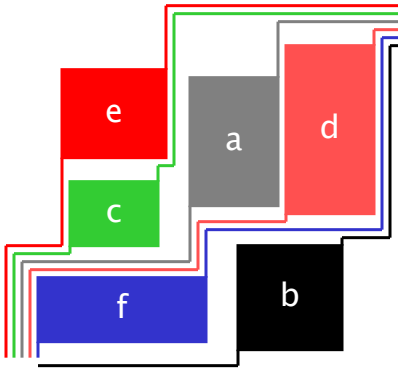
Positive step lines



Is this unique?



Not the same as…



Sequence Pair

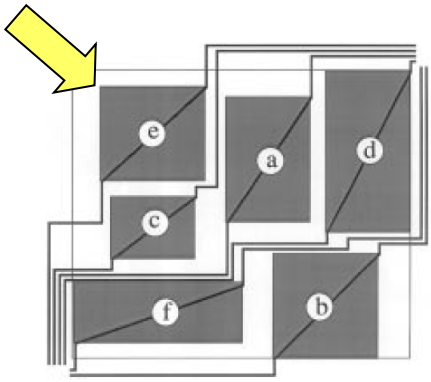
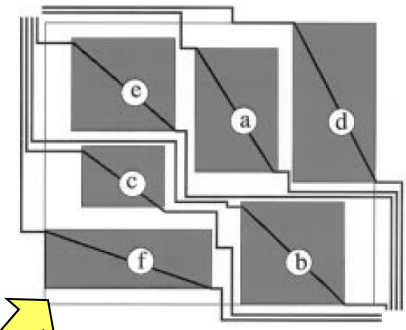


Fig. 2. Positive step-lines.

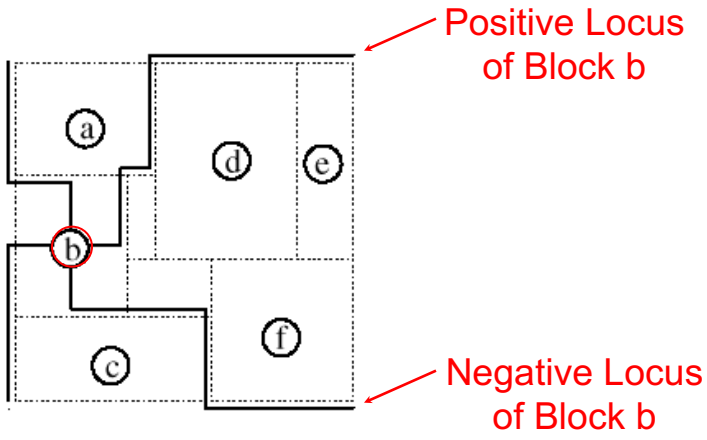


Negative step-lines.

- Positive step line sequence: ecadfb
  [or ecafdb in the alternative version]
- Negative step line sequence: fcbead

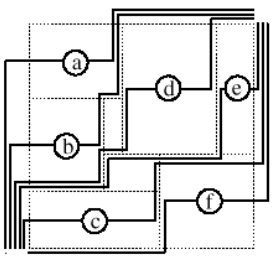
[Pan]

Positive Locus and Negative Locus

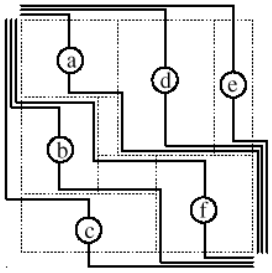


[Pan]

Sequence-Pair



Positive Loci



Negative Loci

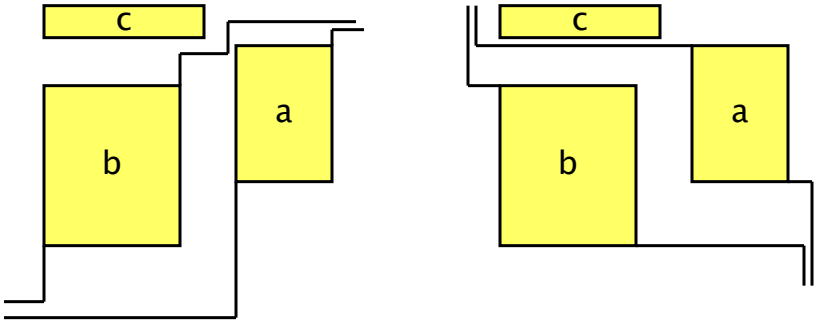
Sequence-Pair = (abdecf, cbfade)

[Pan]

Geometric Info of Sequence-Pair

Given a placement and the corresponding sequence-pair (P, N):

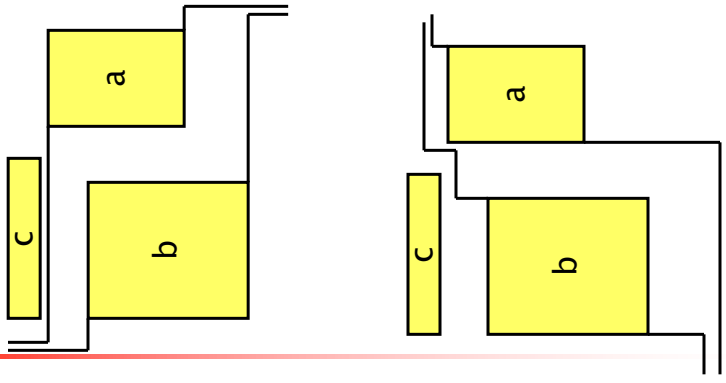
- a right of b ⇔ a is after b in both P and N.



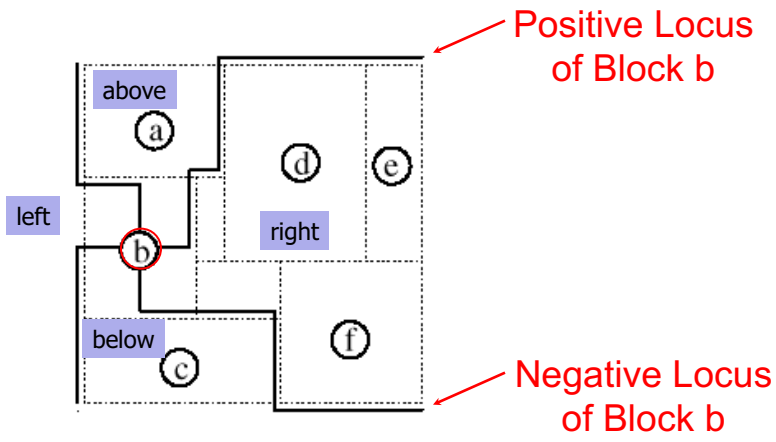
Geometric Info of Sequence-Pair

Given a placement and the corresponding sequence-pair (P, N):

- a above b ⇔ a is before b in P and after b in N



Positive Locus and Negative Locus



[Pan]

Geometric Info of Sequence-Pair

Given a placement and the corresponding sequence-pair (P, N):

- a right of b ⇔ a is after b in both P and N.
- a left of b ⇔ a is before b in both P and N.
- a above b ⇔ a is before b in P and after b in N.
- a below b ⇔ a is after b in P and before b in N.

[Pan]

Sequence Pair

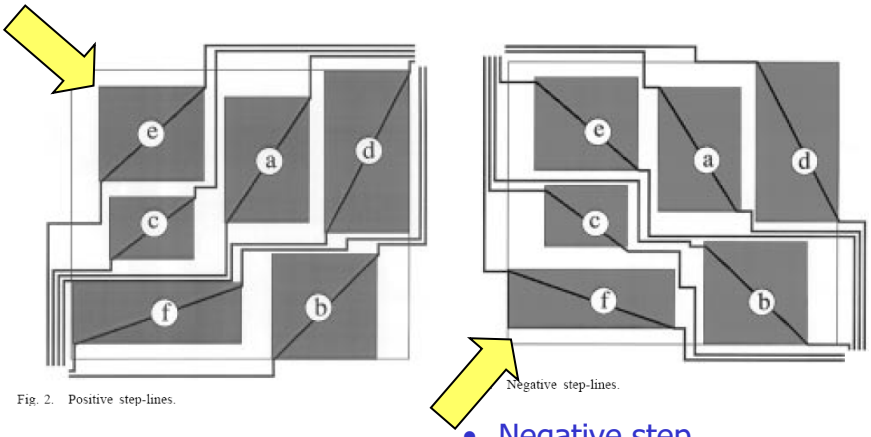


Fig. 2. Positive step-lines.

Negative step-lines.

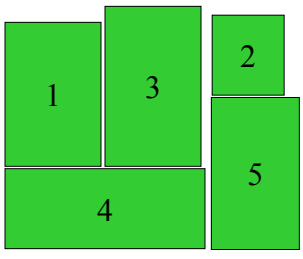
- Positive step line sequence: ecadfb
- Negative step line sequence: fcbead

[Pan]

Example

Consider the sequence pair:

(13245,41352)

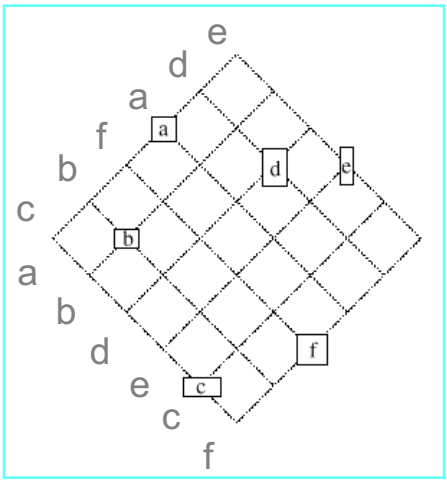


Any other SP that is also valid for this packing?

[Pan]

From Sequence-Pair to a Floorplan

Labeled grid for (abdecf, cbfade)

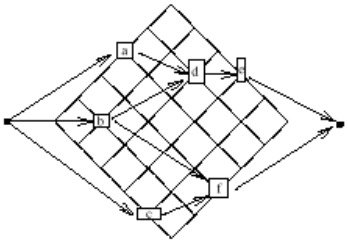


- Given a sequence-pair, the floorplan with smallest area can be found in $O(n^2)$ time.
- Algorithms of time O(n log log n) or O(n log n) exist. But faster than $O(n^2)$ algorithm only when n is quite large.

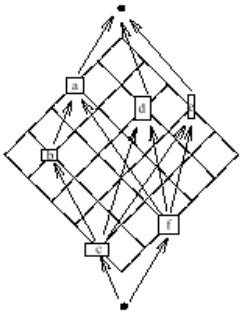
[Pan]

From Sequence-Pair to Placement

- Distance from left (bottom) edge can be found using the longest path algorithm on the horizontal (vertical) constraint graph.



Horizontal Constraint Graph

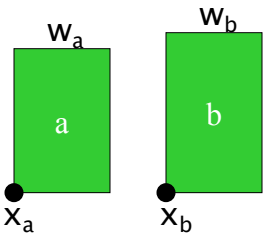


Vertical Constraint Graph

[Pan]

The longest path problem

- Layout compaction
  - Given a set of blocks, place them in the most compact way possible
  - The 2D compaction problem is often solved as a sequence of 1D problems, in the x and y directions

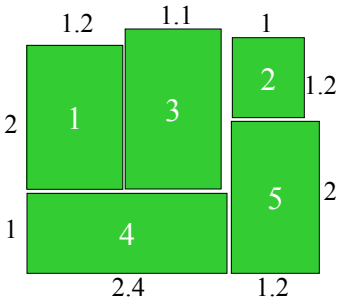


$$x_b - x_a \geq w_a$$

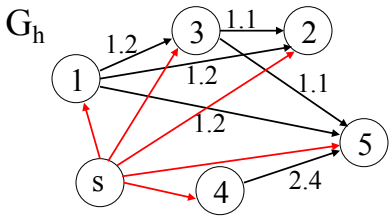
A set of such "≥" constraints represents a longest path problem

47

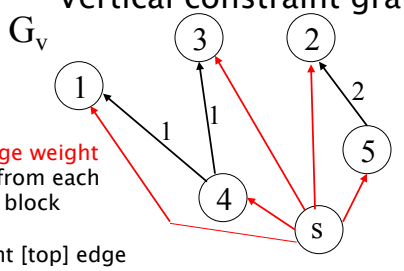
Example



Horizontal constraint graph



Vertical constraint graph

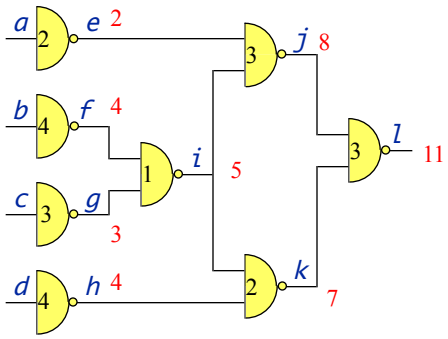


All red edges from the source have zero edge weight
Can also create a sink vertex t, with edges from each vertex to t, with weight = dimension of the block represented by the vertex
S represents the left [bottom] edge, t = right [top] edge

48

Adapted from [Pan]

Solving the longest path problem



- Place inputs (with no predecessors) on a queue
- Process vertex from head of queue
- Add vertex to queue if all inputs processed
- Complexity *O(V+E)*

Floorplan Realization

- Floorplan realization is the step to construct a floorplan from its representation.
- How to construct a floorplan from a sequence pair?
- We can make use of the horizontal and vertical constraint graphs (G_h and G_v).

[Pan]

Floorplan Realization

- Whenever we see (…A…B…, …A…B…), add an edge from A to B in G_h with weight w_A.
- Whenever we see (…A…B…, …B…A…), add an edge from B to A in G_v with weight h_B.
- Add a source vertex s to G_h and G_v pointing, with weight 0, to all vertices without incoming edges.
- Finally, find the longest paths from s to every vertex in G_h and G_v (how?), which are the coordinates of the lower left corner of the module in the packing.

[Pan]

Moves

- Three kinds of moves in the annealing process:
  - M1: Rotate a module, or change the shape of a module
  - M2: Interchange 2 modules in both sequences
  - M3: Interchange 2 modules in the first sequence
- Does this set of move operations ensure reachability? Why?

[Pan]

Pros and Cons of SP

- Advantages:
  - Simple representation
  - All floorplans can be represented.
  - The solution space is finite. (How big?)
- Disadvantages:
  - Redundant representation. The representation is not 1-to-1.
  - The size of the constraint graphs, and thus the runtime to construct the floorplan is quadratic

[Pan]

*-Tree Methods

- Various methods and representations for nonslicing floorplans
  - Bounded slicing grid (BSG) (1996)
  - O-tree (1999)
  - B*-tree (2000)
  - Corner block list (CBL) (2000)
  - Transitive closure graph (TCG) (2001)
- These represent nonslicing floorplans by strings and use simulated annealing to optimize the layout.

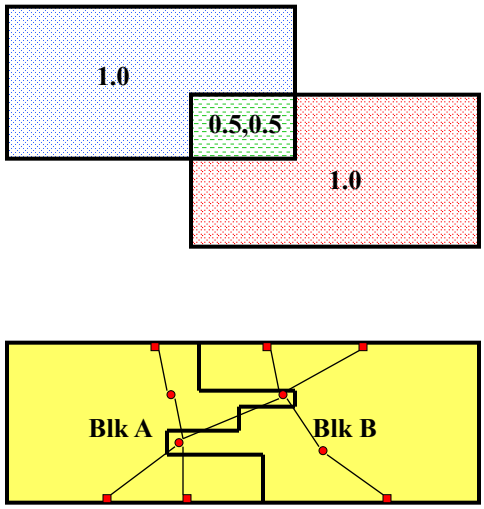
Reexamining the assumptions of floorplanning

Overconstrained Shaping

- Why rectangles, L's, T's ?
  - available granularity is by site spacing, row height
  - placers can handle arbitrarily complex region constraints
  - hard IP reuse, generated modules benefit from shape freedom
- Why non-overlapping ?
  - only requirement: total assigned cell area $\leq$ total resource area
- Roundness and shape simplicity are mythical needs
  - constructive pin assignment $\rightarrow$ don't need roundness
  - path timing optimization $\rightarrow$ may even want disconnected shapes

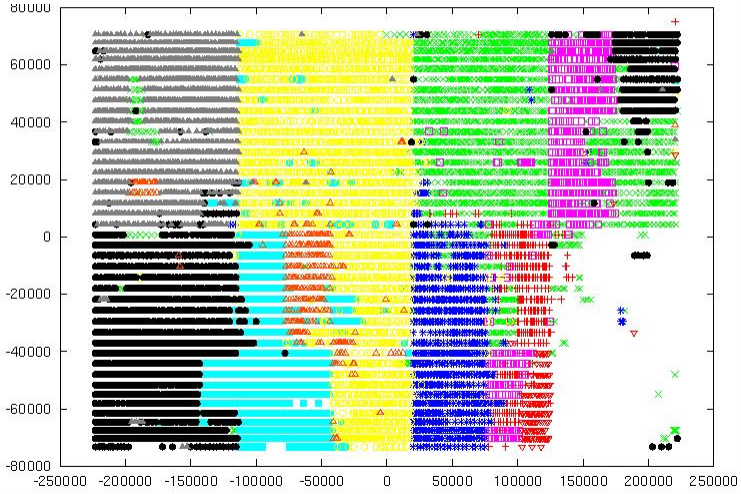
[Kahng]

This is Okay, Really... (Trust Me)



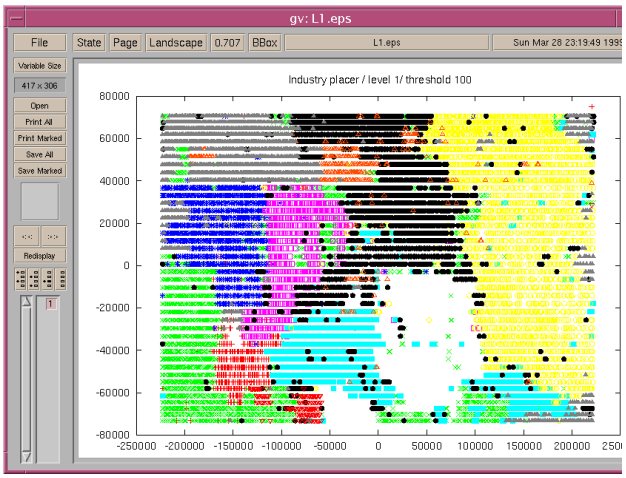
[Kahng]

...The Cells Won't Mind



[Kahng]

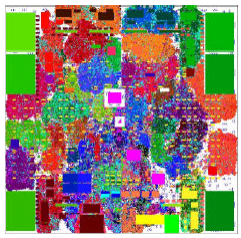
Using Floorplan Information: A Typical "Fluid" Placement



[I. Markov]

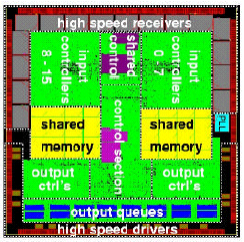
Flat vs. hierarchical placement

Flat



- Works well for highly interconnected networks

Hierarchical



- Good choice for SoC

Can hybridize the two to get best of both worlds

[Lackey *et al.*, IBM, DAC 03]

Other Objective Functions

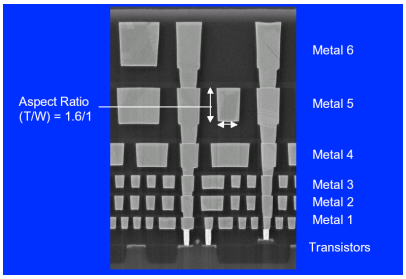
Wire delays

- Critical length as a function of technology
  - Wire length at which delay = clock period

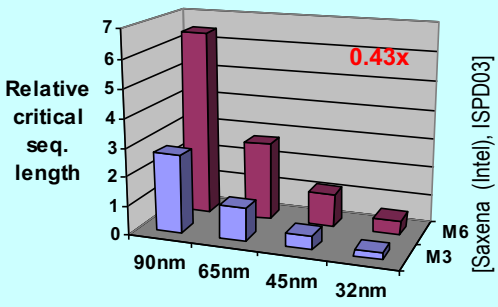
    Across-chip wire delays > clock period

    ⇒ Multicycle global communication is essential

Chip cross-section

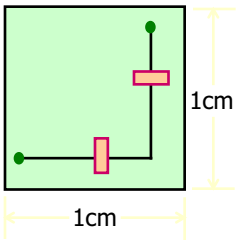


[Intel]



Wire-pipelining

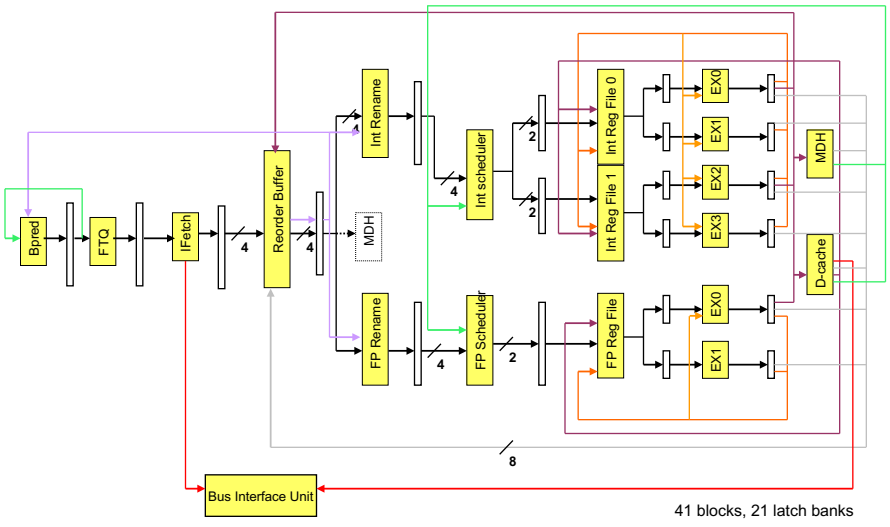
- Interconnect delay is distributed among several clock cycles by inserting flip-flops
- Adds area/power overhead



- Delay = 0.67ns (70nm)
  - [Cong, Proc. IEEE 2001]
- Target Frequency : 3GHz
  (clock period : 0.33ns)

- Widely used, e.g., Intel's Itanium processor

An Example Microarchitectue



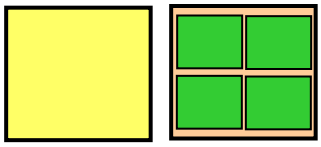
41 blocks, 21 latch banks

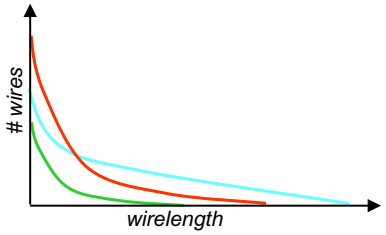
• Numbers below the lines indicate the # of instructions flowing across the line (not bit width)

MDH = Memory Disambiguation Hardware

An Architectural Solution to Interconnect Tyranny

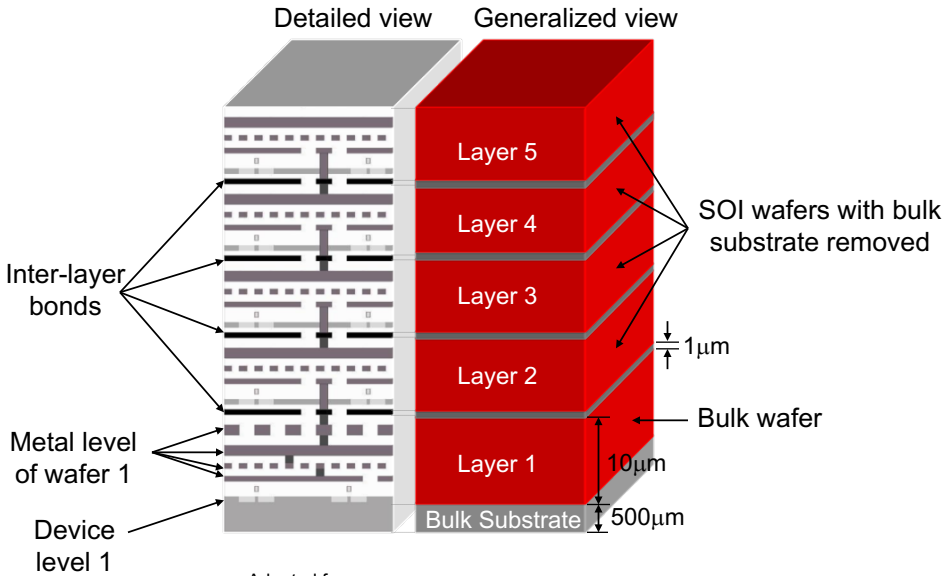
- As seen earlier, alternate scaling scenarios also face interconnect tyranny (albeit to differing degrees)
- Most promising approach: simplify interconnection complexity architecturally
  - Modify wiring histogram shape (i.e. Rent's parameters) of design
- An example: multi-core microprocessors
  - Goes counter to traditional approach of increased integration through block size scaling





[Saxena]

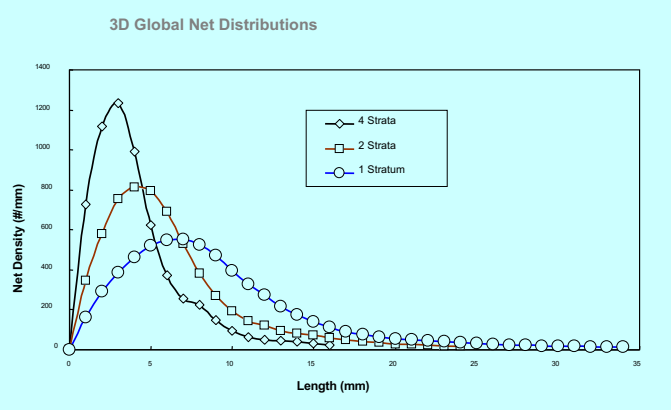
3D integrated circuits



Adapted from
[Das et al., *ISVLSI*, 2003]

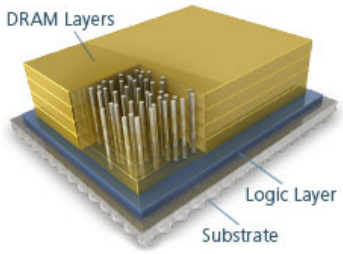
Global Net Length Distribution

- Histogram of net length, for various numbers of 3D layers



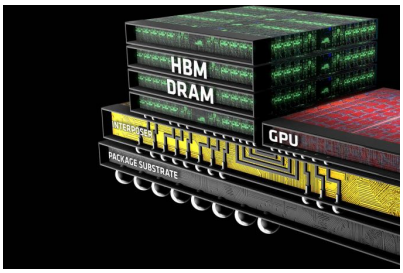
Examples

Micron HMC



https://www.micron.com/products/hybrid-memory-cube/all-about-hmc

AMD HBM



https://www.anandtech.com/show/9266/amd-hbm-deep-dive