Floorplanning

• The final portion of the course covers
  – Scheduling algorithms
  – Resource sharing algorithms
  – Module selection
  – Retiming
  – Floorplanning
  – Function approximation
  – Perspectives for the future

• This lecture covers
  – The floorplanning problem
  – Slicing and non-slicing floorplans and representations
  – Heuristic and ILP solutions

Motivation

• In recent years, we have moved to deep sub-micron design.
• Wiring delays have started to compete with (and sometimes overtake) logic delay.
  – it is important to be able to estimate wiring delay early in the design process.
• We need an early idea of geometrical layout on silicon
  – a floorplan.
• Floorplanning becomes part of architectural synthesis.

Slicing Floorplans

• Floorplans are typically categorised into
  – slicing floorplans or non-slicing floorplans

• Slicing floorplan
  – obtainable by repeated bisection of rectangular cells
  – simplifies representation and optimization

Slicing Tree Representation

• A slicing tree is a binary tree representation of a slicing floorplan
  – a leaf is a resource to be floorplanned
  – other nodes indicate how to compose their children: vertically, or horizontally.
Skewed Slicing Trees

- Unfortunately, slicing trees are not unique representations of the floorplan.

Both slicing trees are valid representations

Skewed Slicing Trees

- A skewed slicing tree has the following property
  - no node and its right-child have the same type
- Every slicing floorplan has a unique skewed slicing tree.
- How to represent the trees in a floorplanning algorithm?
  - we can represent it as a string, called a *Polish expression*.

Polish Expressions

- A skewed slicing tree corresponds to a Polish expression where
  - no two consecutive operators (H/V) are of the same type.

- Polish expression for:
  
  \[
  \text{Polish}(Y) + \text{Polish}(Z) + "X"
  \]

- Polish expression for leaf is leaf value.
- For tree on the left:
  
  "712H3H645HVHV"

Floorplan Optimization

- We have a compact and unique representation of a slicing floorplan. How to optimize for smallest area?
- A common approach:
  - start with a random floorplan
  - improve it based on certain well-defined "moves"
- What moves?°
  - Swap two adjacent operands (leaf nodes) in the Polish expression.
  - Take a chain of consecutive operators, e.g. "HVHV", and complement it, e.g. "VHVH".
  - Swap an adjacent operator and operand. (But make sure still a skewed tree!)

\[1\] Moves from Prof. Hai Zhou
**Floorplan Optimization**

- 12H3H → 21H3H
- 21V3H

**Area Computation**

- How to tell whether a move improves area?
  - Height(XYH) = max(Height(X), Height(Y))
  - Width(XYH) = Width(X) + Width(Y)
  - Height(XYV) = Height(X) + Height(Y)
  - Width(XYV) = max(Width(X), Width(Y))

- Height(21V3H) = max(Height(21V), Height(3))
  = max(Height(2) + Height(1), Height(3))

- Width(21V3H) = Width(21V) + Width(3)
  = max(Width(2), Width(1)) + Width(3)

**Simulated Annealing**

- In our example, not all moves improved area
  - not good enough to just “pick the best move” each time

- Simulated annealing is often used
  - pick a move at random.
  - if it improves area, do it.
  - if it doesn’t improve area, maybe do it.

- Probability of selecting a move that does not improve area
  - reduces with area penalty for move
  - decreases (for a fixed area penalty) with iteration number

**An ILP Approach**

- We can also take an ILP approach to the floorplanning problem
  - guaranteed optimal solutions
  - slicing and non-slicing floorplans within a single framework
  - poor execution-time scaling
An ILP Approach

- Resources cannot overlap

\[
\begin{align*}
 x_i &\geq x_j + w_j \quad (1) \\
 x_i &\geq x_j + w_i \quad (2) \\
 y_i &\geq y_j + h_j \quad (3) \\
 y_i &\geq y_j + h_i \quad (4)
\end{align*}
\]

- We need to ensure that at least one of (1)-(4) holds

Good Floorplanning

- Some floorplans are better than others
  - Place resources that communicate close to each other.

- Given a maximum wire-length \( W_{ij} \) for each pair \((i,j) \in R^2\) of connected resources, (5)-(9) must hold.

\[
\begin{align*}
 x_i + 0.5w_i - x_j - 0.5w_j &\leq W_{ij}^h \quad (5) \\
 -x_i - 0.5w_i + x_j + 0.5w_j &\leq W_{ij}^h \quad (6) \\
 y_i + 0.5h_i - y_j - 0.5h_j &\leq W_{ij}^v \quad (7) \\
 -y_i - 0.5h_i + y_j + 0.5h_j &\leq W_{ij}^v \quad (8) \\
 W_{ij} = W_{ij}^h + W_{ij}^v \quad (9)
\end{align*}
\]
Design Area

- We must ensure that the design fits in chip dimensions $X$ by $Y$.
  - For all resources $i \in R$, (10) and (11) must hold.
    
    $$x_i + w_i \leq X \quad (10)$$
    $$y_i + h_i \leq Y \quad (11)$$

- If the chip aspect ratio is given, $Y = kX$ (12).
  - Objective is then min: $X$
- If aspect ratio is not given, we have min: $XY$
  - problem: nonlinear objective

Linearization

- Two standard approaches
  - iterate: solve “min: $X” with $Y$ fixed, many times for different values of $Y$.
  - approximate:
    
    $$XY \approx X' Y' + (X - X')Y' + (Y - Y')X'$$

- (or some combination of the two).

- More recently, convex (nonlinear) optimization techniques have started to appear.

ILP Approaches

- The approach has a (very) large execution time: $O(n^2)$ integer variables.
  - techniques have been proposed to break down into sub-problems\(^1\).
  - sub-problems can be stitched into suboptimal solutions.

Summary

- This lecture has introduced floorplanning
  - motivation: deep-submicron era
  - slicing vs non-slicing floorplans
  - Polish expressions
  - optimizing moves
  - an ILP approach

- The next lecture will look at function approximation.

---

Suggested Problems

• Draw the floorplan represented by the following slicing tree:

```
  2
 / \
V   V

  H
  1
```

• Convert this tree into a skewed slicing tree.
• Write the Polish expression for the skewed tree.
• Identify one of the three moves proposed in this lecture that could be applied to obtain an optimal area floorplan for the given resource dimensions.
  – Resource 1: Height = 2, Width = 2
  – Resource 2: Height = 2, Width = 1
  – Resource 3: Height = 1, Width = 1
  – Resource 4: Height = 1, Width = 1