Floorplanning: Overview

**Introduction:** With ever larger designs, it is increasingly important to plan a design at an early stage. This early plan helps constrain later design decisions in terms of area, wire usage, ports, and port locations. The early stage plan, a.k.a. a floorplan, is fleshed out with increasing details with the design flow. The issue is a chicken-and-egg problem in that an accurate floorplan is difficult without knowing the details, and yet, building the details is greatly facilitated with a floorplan. So such a plan is a first guess. We use a lot of estimates for area to arrive at a reasonable plan. The plan discussed in this lecture includes area for blocks, ports and their locations, routing channels, metal layer usage, power and ground routing, clock routing, and I/O pins. The result is a diagram of the chip...
Floorplanning

- This is a plan of the chip,
  - Shows the module/blocks
  - The space needed for wires
    - In a cell it is called the “color plan”; at the chip level - “floorplan”
- How $V_{DD}$, $V_{GND}$, and Clock are distributed + wire width
- The area is estimated by the type of block (dpath or ctrl)
- The routing is based on position of I/O pins of each block
  - Floorplanning tools will help position large blocks, rotating, flipping to minimize the routing between blocks
  - Helps predict wiring loads and area of chip
  - Makes sure you have enough pins, and space
- Early in design
  - Floorplan budgets area, wire area/delay. Negotiate tradeoffs
- Late in design
  - Make sure the pieces fit together as planned
  - Implement the global layout
Floorplanning

**Input**
- Required
  - Design netlist
  - Area requirements
  - Power requirements
  - Timing constraints
  - Physical partitioning information
  - Die size vs. performance tradeoff
- Optional
  - I/O placement
  - Macro placement information

**Output (design ready for standard cell placement)**
- Die/block area
- I/Os placed
- Macros placed
- Power placed
- Power pre-routing
- Standard cell placement areas
A Conceptual Floorplan

- **Blocks** inside a pad frame
- **Routing** may be inside blocks but need to interconnect them
- **Brick and Mortar**
  - Random sized blocks that are more difficult than standard cell rows to route
- **Layout hierarchy not deep**
  - Tough to do this too many times

Example Chip Layout
More Complex Chip

Design Flow and Physical Design Stage

- **Definitions:**
  - **Cell:** a circuit component to be placed on the chip area. In placement, the functionality of the component is ignored.
  - **Net:** specifying a subset of terminals to connect several cells.
  - **Netlist:** a set of nets which contains the connectivity information of the circuit.

*Courtesy: Andrew Kahng, UCSD*
Placement

- **Input**
  - A set of cells and their complete information (a cell library)
  - Connectivity information between cells (netlist information)
- **Output**
  - A set of locations on the chip, one location for each cell
- **Goal**
  - The cells are placed to produce a routable chip that meets timing and other constraints (e.g. low-power, noise, etc.)
- **Challenge**
  - The number of cells in a design is very large (> 1M)
  - The timing constraints are very tight

Placement: Ordering

- **Opt relative order**
  - To spread...
  - ... or not to spread
- **Place to the left**
  - ... or to the right
- **Opt relative order**

Without “free” space, the placement problem is dominated by order

Courtesy: Andrew Kahng, UCSD
Global and Detailed Placement

- **Global placement**
  - Decode approximate locations for cells by placing cells in global bins

- **Detailed placement**
  - Make some local adjustments to obtain final non-overlapping placement

Courtesy: Andrew Kahng, UCSD
Placement Footprints

- Standard cell
- Data path
- IP – floorplanning

Courtesy: Andrew Kahng, UCSD

Placement Footprints

Reserved areas

Core

IO

Control

Mixed Data Path & sea of gates:

Courtesy: Andrew Kahng, UCSD
Unconstrained Placement

Floorplanned Placement
Placement: Blocks and Wires

- **Placement**
  - Simple: min length of wires
  - Timing-driven: cost of wires weighted to min critical-path delays

- **Blocks**
  - Blocks have area and aspect ratio
  - Different rotations and reflections
  - Uniform size blocks are easier to interchange

- **Wires**
  - Cannot ignore wiring during block placement
    - Large wiring areas may force rearrangement of blocks
  - Wiring plan must consider area and delay of critical signals
  - Blocks divide wiring area into routing channels

- **Next step: routing**
  - Design saved in standard design exchange format (DEF) for routing

Types of Routing

- **Two levels of routing**
  - Global: notional set of abutting channels
  - Local: actual geometry required to complete signal connections

- **Two types of routing**
  - Channel routing
    - Channel may grow in one dimension to accommodate wires
    - Pins generally on only two sides
  - Switchbox routing
    - Cannot grow in any dimension
    - Pins are on all four sides, fixing dimensions of the box
Channel Definition

- Channels end at block boundaries

- Several alternate channel definitions are possible:
  - Depends on position of the blocks

- Changing spacing changes relationship between block edges

Channel Graph

- Nodes are channels
  - Edges placed between two channels that touch

- Channel graph
  - Shows paths between channels
  - Can be used to guide global routing
Routing Channels in Order

- Wire out of end of one channel creates pin on side of next channel:

- Can create an unroutable combination of channels with circular constraints (Windmills)

Slicable Floorplan

- Can be recursively cut in two without cutting any blocks

- Guaranteed no windmills
  - Therefore guaranteed to have a feasible order of routing channels

- Slicability is a desirable property for floorplans

- A binary tree representation
  - Nodes: slices
  - Leaves: blocks
Global Routing and Detailed Routing

**Global Routing**
- Goal: assign wires to paths through channels
- Don’t worry about exact routing of wires within channel
  - Final step is the detailed routing
- Channel utilization
  - Can estimate channel height from global routing using congestion
  - Keep all channels about equally full to minimize wasted area

**Detailed Routing**
- Picking exact route of the wires
- Pay attention to the actual timing constraints
  - Route time-critical signals first
  - Shortest path may not be best for global wiring
- May need to rip-up wires and reroute to improve the global routing

Line-Probe Routing

**Heuristic method for finding a short route**
- From one channel to the next

**Works with arbitrary combination of obstacles**

**Does not explore all possible paths – not optimal**

**Method**
- Draw probe line from each source/destination port
- Extend line until hitting an obstacle (a block, a previous connection) or hit the probe line of destination
- May need to iterate
Switchbox Routing

- Can’t expand a switchbox to make room for more wiring
- Switchbox may be defined by intersection of channels

Routing Order
- Switchboxes frequently need more experimentation with wiring order because nets may block other nets

O-Tree

- Partial ordering based on projection overlapping (with given physical locations)
- Transforming into binary trees by pivoting, etc.
- Coded in a node sequence given a tree traversal algorithm
  - E.g., OACBDEF for DFS
- Condensed solution space

Courtesy: Andrew Kahng, UCSD
**Sequence Pair**

- Based on layout partitions by non-overlapping ascending/descending staircases
- Coded in two node sequences
  - E.g., CEDFAB for descending staircases and
  - ABCDEF for ascending staircases
- Larger solution space, finer representation

*Courtesy: Andrew Kahng, UCSD*

---

**Next Step: Partitioning**

- Circuit netlist represented by hypergraph

*Courtesy: Andrew Kahng, UCSD*
Hypergraph Partitioning in VLSI

- Circuit netlist represented by hypergraph
- Variants
  - Directed/undirected hypergraphs
  - Weighted/unweighted vertices, edges
  - Constraints, objectives...
- Human-designed instances
- Benchmarks
  - Up to 4,000,000 vertices
  - Sparse (vertex degree ~ 4, superedge size ~ 4)
  - Small number of very large hyperedges

Example: Partitioning of a Circuit

Input size: 48

Cut 1=4
Size 1=15

Cut 2=4
Size 2=16

Size 3=17

Courtesy: Andrew Kahng, UCSD
Hierarchical Partitioning

- **Levels of partitioning**
  - System-level
    - Each sub-system can be designed as a single PCB
  - Board-level
    - Circuit assigned to a PCB is partitioned into sub-circuits each fabricated as a VLSI chip
  - Chip-level
    - Circuit assigned to the chip is divided into manageable sub-circuits

Delay at Different Levels of Partitions

![Diagram showing delay at different levels of partitions](image-url)

*Courtesy: Andrew Kahng, UCSD*
Delay at Different Levels of Partitions

Multilevel Partitioning

Courtesy: Andrew Kahng, UCSD
Floorplanning Implications on Verilog

- Top-level partition should be well thought out
  - Create chip-plan early
- Need to match physical partition
  - Signals between blocks will be real wires
  - Blocks have to fit together in the layout
- Keep units with different implementation methods separate
  - Automatic synthesis versus custom
- Estimate long wires between blocks
- Don’t forget about testability
  - Generates tests (vectors that are sometimes read from file)
  - Checks the answers (or make self-checking tests)
  - Often check internal nodes to be sure stuff is working
  - To test chip - ensure that the chip is the same as the design (no faults)
    - All tests must be from the pins of the chip. No access to internal nodes

Floorplanning Summary

- Develop a wiring plan
  - Think about how layers will be used to distribute important wires
- Sweep small components into larger blocks
  - A floorplan with a single NAND gate in the middle: hard to work with
- Design wiring that looks simple
  - If it looks complicated, it is complicated
- Design planar wiring
  - Planarity is the essence of simplicity. It isn’t always possible, but do it where feasible (and where it doesn’t introduce unacceptable delay)
- Draw separate wiring plans for power and clocking
  - These are important design tasks which should be tackled early
- Chip finishing
  - Pads
    - Library components which require careful electrical design.
  - Package
    - Can introduce significant inductance