PHYSICAL SYNTHESIS
Outline

- Physical Synthesis overview
- Four main steps
  - Partitioning
  - Floor planning
  - Placement
  - Routing
Objectives

After studying this lecture, you should be able to understand the concepts, styles, algorithms, and design flow involved in physical design.

Outline

- Physical design
  - Input: A netlist of gates (or blocks) and their inter-connections
  - Output: A geometrical layout of the netlist within an area constraint
  - Other goals: Minimize signal delays, inter-connection area, number of layers, power, and cross-talk.

- Physical design styles
  - Full custom, Standard cells
  - Gate array, FPGA

- Physical design flow
  - Partitioning
  - Floor planning
  - Placement
  - Routing
    - ASIC
    - FPGA
  - Circuit extraction (not covered)
  - Post-simulation (not covered)
Classical ASIC design flow (w/ behavioral)
Recall: Classical ASIC design flow (cont.)

- Behavioral level of abstract

- Classical ASIC design flow
  1. Design entry:
     - Enter the design into an ASIC design system, traditionally using schematic entry, later HDLs, and recently, higher level of abstractions such as algorithms (SystemC).
  2. Logic synthesis:
     - Use an HDL and a logic synthesis tool to produce a netlist, i.e. a description of the logic cells and their connections.
  3. System partitioning:
     - Divide a complex system to subsystem in a hierarchical manner (sizes manageable by current ASIC technology).
  4. Pre-layout simulation:
     - Check to see if the design functions correctly.
  5. Floor planning
     - Map out the topology of a complete chip with i/o pads, modules, local and global routings.
  6. Placement:
     - Decide the locations of cells in a block.
  7. Routing:
     - Make the connections among cells and blocks.
  8. Circuit extraction
     - Determine the resistance and capacitance of the interconnect.
  9. Post-layout simulation
     - Check to see if the design still works with the added loads of the interconnects (works with refined constraints).
Example of a netlist: input to physical design

```
timescale 1ns / 10ps
module comp_mux_o (a, b, outp);
input [2:0] a; input [2:0] b;
output [2:0] outp;
supply1 VDD, supply0 VSS;
assign
  in01d0.B1_11 = .I(a[2]),
  .ZN(B1_11_EN);
assign
  in01d0.B1_12 = .I(b[1]),
  .ZN(B1_12_EN);
assign
  ao31di.B1_13 = .A1(a[0]),
  .S2(B1_13_EN), .B1(B1_12_EN),
  .B2(a[1]), .ZN(B1_13_E);
assign
  .B1(b[1]), .ZN(B1_14_EN);
assign
  fn52d1.B1_15 = .A(B1_13_EN),
  .B(b[1]), .ZN(B1_15_EN);
assign
  nx21d1.B1_16 = .I0(a[0]),
  .S1(b[0]), .S(B1_15_EN),
  .t(output[0]);
assign
  nx21d1.B1_17 = .I0(a[1]),
  .S1(b[1]), .S(B1_15_EN),
  .t(output[1]);
assign
  nx21d1.B1_18 = .I0(a[2]),
  .S1(b[2]), .S(B1_15_EN),
  .t(output[2]);
endmodule
```

**FIGURE 12.3** The comparator/MUX after logic synthesis and logic optimization with the default settings. This figure shows the structural netlist, comp_mux_o.v, and its derived schematic.
Example of layout: partial output
Examples of layout: output of phy. design
Physical design styles

- **Full Custom**
  - Utilized for large production volume chips such as microprocessors.
  - No restriction on the placement of functional blocks and their interconnections.
  - Highly optimized, but labour intensive.

- **Standard Cell**
  - Utilized for smaller production ASICs that are generated by synthesis tools.
  - Layout arranged in row of cells that perform computation.
  - Routing done on “channels” between the rows.

- **Gate Arrays**
  - Pre-fabricated array of gates (could be NAND).
  - Design is mapped onto the gates, and the inter-connections are routed.

- **Field Programmable Gate Arrays**
  - Pre-fabricated array of programmable logic and interconnections.
  - No fabrication step required.
Full custom layout

Full- Custom Layout of 8- bit Multiplier and BIST Logic in SCAL- D

© Akash Kumar
Standard cell layout
Standard cell layout
**FPGA Layout**

- **Island FPGAs**
  - Array of Configurable Logic Blocks (CLB)
  - Horizontal and vertical routing channels connecting the functional units
  - Programmable Switch Matrices (PSM)
  - Example: Xilinx, Altera FPGAs

- **Row-based FPGAs**
  - Like standard cell design
  - Rows of CLBs
  - Routing channels (fixed width) between rows of logic
  - Example: Actel FPGAs
Typical steps in physical design

- **Partitioning**: Divide the net-list into sub-sets.

- **Floor-planning**: Determine the dimensions of the various units and their placements.

- **Placement**: Place the cells to minimize area and wire length.

- **Routing**:
  - **Global routing**: Determine the regions through the chip that the wires or net would be routed.
  - **Detailed routing**: Determine the actual layout of the nets within each routing region.
The partitioning concept

- **What**
  A physical design step in which a large circuit entity is segmented into many sub-entities...

- **Why**
  ... so that they can be designed independently, and fitted into many area-constrained boards or chips...

- **How**
  ... by partitioning and minimizing the interconnects among sub-entities.
Hierarchical partitioning

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

PCB1

10x

PCB2

20x or more

© Sherwani
© Akash Kumar
Example: Partitioning of a circuit

Input size: 48

Cut 1 = 4
Size 1 = 15

Cut 2 = 4
Size 2 = 16

Size 3 = 17
Subgraph replication to reduce cutsize

- Vertices are replicated to improve cutsize
- Good results if limited number of components replicated
The *floor planning* concept

- **What**
  A physical design step in which a topology of a complete chip is planned on a (usually) rectangular area …

- **Why**
  … so that the final area, interconnects, and possibly power consumption can be minimized…

- **How**
  … by strategically budgeting areas for functional modules and their physical locations along with the I/O pads, clock, and power/ground rails.
Floor planning: why is it important?

- Early stage of physical design:
  - To generate the coarse floor plan
  - Determines the location of large blocks (why large blocks first?)
    - detailed placement easier (divide and conquer!)
  - Estimates of area, delay (timing), and power
    - important design decisions
  - Impact on subsequent design steps (e.g., routing, thermal analysis and management, and optimization)
Floor planning (cont.)

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

© Akash Kumar
Floor planning techniques

- 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
Example floor planning using non-slicing

- Hierarchical floorplan of order 5
  - Templates

- Floorplan and tree example
The placement concept

- **What**
  A fundamental physical design step in which various functional modules, after being planned, will be placed to achieve routability …

- **Why**
  … so that the final area, interconnects, timing/congestions and possibly power consumption can be minimized.
Problems with placement:

- Given a netlist and cells, find the exact location of the cells to minimize area and wire-length. The provided cells/modules:
  - Usually fixed, equal height (exception: double height cells)
  - Some fixed (I/O pads)
  - Connected by edges or hyper-edges

- Placement must be consistent with the standard-cell design methodology
  - Row-based, no hard-macros

- Serious interconnect issues (delay, routability, noise) in deep-submicron design must be considered.
  - Need placement information even in early design stages (e.g., logic synthesis)
Placement Footprints

Standard cells

IP blocks

Data Path
Placement Footprints

Perimeter IO

Area IO
Examples: VLSI global placement

bad placement

good placement
Placement algorithms

- **Top-Down**
  - Partitioning-based placement
  - Recursive bi-partitioning or quadrisection
    - Cut direction?
    - Partition vs. physical location

- **Iterative**
  - Start with an initial placement, iteratively improve wire-length / area

- **Constructive**
  - Start with a few cells in the center, and place highly connected adjacent modules around them
The *routing* concept

- **What**
  
  A physical design step in which various functional modules, after being placed, are connected …

- **Why**
  
  … so that the final area, delay, number of layers, and vias can be minimized…

- **How**
  
  …by performing global (coarse) routing followed by detailed (fine) routing of modules according to some micro-electronic governing rules.
Routing anatomy

Top view

Symbolic Layout

Metal layer 1

Via

Metal layer 2

Metal layer 3

Note: Colors used in this slide are not standard
Global vs. Detailed routing

- **Global routing**
  - Input: detailed placement, with exact terminal locations
  - Determine “channel” (routing region) for each net
  - Objective: minimize area (congestion), and timing (approximate)

- **Detailed routing**
  - Input: channels and approximate routing from the global routing phase
  - Determine the exact route and layers for each net
  - Objective: valid routing, minimize area (congestion), meet timing constraints
  - Additional objectives: min via, power

Figs. [©Sherwani]
Routing environment

- Chip Architecture
  - Full-custom
  - Standard cell
  - FPGA
Routing environment – Full Custom

- No constraint on routing regions
- Objective functions are the objective functions of general routing problem.
  - Routability
  - Area Minimization
  - Total wire length minimization
  - Maximum wire length minimization
- Very labour intensive
Routing environment – Standard Cell

- Variable channel height?

- Feed-through cells connect channels

- Channels do not have predetermined capacity

- Feedthroughs have predetermined capacity

- Objectives
  - Area minimization
  - Minimization to total wire length
  - Minimization of maximum wire length
Routing environment – FPGA

- Fixed channel height
- Limited switchbox connections
- Prefabricated wire segments have different weights
- Objectives
  - Routability
  - Minimize total wire length
  - Minimize maximum wire length

Tracks

Failed connection
FPGA programmable switch elements

- Used in connecting:
  - The I/O of CLBs to the wires
  - A horizontal wire to a vertical wire
  - Two wire segments to form a longer wire segment
FPGA routing channels architecture

- Note: fixed channel widths (tracks)
- Should “predict” all possible connectivity requirements when designing the FPGA chip
- Channel -> track -> segment

- Segment length?
  - Long: carry the signal longer, less “concatenation” switches, but might waste track
  - Short: local connections, slow for longer connections
FPGA switch boxes

- Ideally, provide switches for all possible connections

- Trade-off:
  - Too many switches:
    - Large area
    - Complex to program
  - Too few switches:
    - Cannot route signals

One possible solution

Xilinx 4000
FPGA Routing

- Routing resources pre-fabricated
  - 100% routability using existing channels
  - If fail to route all nets, redo placement

- FPGA architectural issues
  - Careful balance between number of logic blocks and routing resources (100% logic area utilization?)
  - Designing flexible switchboxes and channels (conflicts with high clock speeds)
Summary

- Four main steps in physical synthesis
  - Partitioning
  - Floor planning
  - Placement
  - Routing