Routing is Everywhere in Your Daily Life
BAD Routing: Waste of Time + Gas!

BAD Routing: Slow Chip + Waste of Energy!

From Synthesis to Place-and-Route (PnR)

```vhdl
module ADD(DO, DI_A, DI_B);
    output [31:0] DO;
    input [31:0] DI_A;
    input [31:0] DI_B;
    wire ... ;
    ...
    FADDX1 U31 ( ... );
    FADDX1 U30 ( ... );
    ...
    XOR2X1 U1 ( ... );
    XOR2X1 U2 ( ... );
    INVX0 U3 ( ... );
    NAND2X0 U4 ( ... );
    OA21X1 U5 ( ... );
endmodule
```

Visualization of Netlist
From Synthesis to Place-and-Route (PnR)

Wait! Of Course, Not That Easy...

- IO assignment
- Floor-planning & power-planning
- Cell placement
- Optimization, clock-tree syn. & detailed routing
- Add core filler
- Add IO filler
- DRC, LVS, tapeout
- Bonding (for chip measurements)
Chip Layout vs. Micrograph

How Tool Handles Large-Scale PnR (1)

- Today we’re talking about billion-transistor designs
- Continuous design variables for circuit opt.: infeasible
- PnR technology: constrained, discrete optimization
  - Library of standard cells
  - Site, Row-based placement
  - Fixed routing track & pitch
  - Layout exchange format (LEF)
**PnR Technology (1)**

**Standard-Cell Library**
- Same height
- Same power rail
- Discrete sizing
  \((1\times, 2\times, 4\times, \ldots)\)

**Site, Row-Based PnR**
- Placement grid
- Easy VDD/VSS connection

On-grid placement of standard cells

**PnR Technology (2)**

**Fixed routing pitch**
- On-grid routing
- Lowering complexity of routing algorithm

**Layout Ex. Format**
- Process info. (layer, design rule, par. RC)
- Simplified view of std. cells & macros
LEF: Technology Information

LEF: Simplified Cell View (1)

PnR tool doesn’t care about detailed layout within a cell, it only cares about the shapes of IO pins and VDD/VSS. Simplifying the layout of cells: improved tool runtime
like building a road in front of your home
You don’t provide internal layout of your home to workers.
You only tell them where the doors are.

**LEF: Simplified Cell View (2)**

- **cell name**
- **placed in the core region**
- can be flipped both vertically & horizontally
- **placed on the grid defined by SITE core**
- **RECT x1 y1 x2 y2** to describe the shape
- Antenna information for design-rule check (DRC) & SI analysis (discuss later)
- **USE POWER/GROUND** tells the tool where the power pins are
Timing & Noise Analysis

- **Static timing analysis (propagation delay)**
  - Cell timing library (.lib)

- **Parasitic extraction (wire delay)**
  - Capacitance table (.captab)
  - QX technology file (.cl)

- **Signal integrity (crosstalk, antenna effect)**
  - CeltIC library (.cdb)

Crosstalk

- **Getting serious in deep-submicron era**
  - Smaller pitches
  - Greater height/width ratio
  - Higher clock frequency

![Crosstalk Diagram]

- **Delay Problem**
  - Aggressor
  - Orig. Signal
  - Impacted Signal

- **Noise Problem**
  - Aggressor
  - Orig. Signal
  - Impacted Signal
Crosstalk Prevention

- Placement solution
  - Insert buffer in lines
  - Upsize driver

- Routing solution
  - Congestion optimization
  - Limit length of parallel nets
  - Wider wire spacing
  - Shield special nets (e.g. clock)

Antenna Effect

During chip fabrication, metals are initially deposited, covering the entire chip.

Unneeded portions of the metal are removed by etching, typically using plasma (charged particles).

Exposed metal collects charges, forming voltage potential. Might be large enough to damage the gate oxide.
Fixing Antenna Problem

Antenna Diode → Add Jumper
Add Buffer

Commercial PnR Tool: SoC Encounter

Hybrid GUI-Script Interface

Menu → Tool Bar → Design Views → Switch Bar → Display Ctrl. → Cursor Coordinate

Your Layout
**Tool Bar**

- Import Design
- Zoom Fit
- Clear All Ruler
- Undo/Redo
- Zoom In/Out
- Selected Zoom (down/up)
- Move/Resize/Reshape
- Add VIA
- View Hier.
- Selected View
- Hier.

**Design Views**

- **Floorplan View**
  - Show hierarchical module block guides, connection lines & floorplan objects

- **Amoeba View**
  - Display the outline of modules after placement

- **Placement View**
  - Display the detailed placement of cells and macro blocks
Display Control

Color Display: Floorplan Objects & Physical Layers
(can be customized by yourself)

- Switch Bar b/w F-plan & Physical

Useful Bindkeys

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>Edit Attribute</td>
</tr>
<tr>
<td>f</td>
<td>Zoom Fit</td>
</tr>
<tr>
<td>z</td>
<td>Zoom In</td>
</tr>
<tr>
<td>Z</td>
<td>Zoom Out</td>
</tr>
<tr>
<td>k</td>
<td>Create Ruler</td>
</tr>
<tr>
<td>Shift + k</td>
<td>Delete All Ruler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>Select Next</td>
</tr>
<tr>
<td>e</td>
<td>Popup Edit</td>
</tr>
<tr>
<td>T</td>
<td>Edit Trim</td>
</tr>
<tr>
<td>0-9</td>
<td>Toggle layer [0-9]</td>
</tr>
<tr>
<td>h/H</td>
<td>View Hier. (Up/Down)</td>
</tr>
<tr>
<td>x</td>
<td>Clear DRC</td>
</tr>
</tbody>
</table>

More bindkeys:
Options → Set Preference → Binding Key
SoC Encounter: General PnR Steps

- File preparation & initial setup
- Floor-planning
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells’ VDD/VSS to power ring)
- Placement & pre-CTS optimization
- Clock-tree synthesis (CTS) & post-CTS optimization
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- Detailed routing & post-route opt. (timing & SI driven)
- Layout verification & stream out
File Preparation

You Generate
- Gate-level netlist (.vg) from synthesis result
- Design constraints (.sdc) from synthesis result
- Multi-Mode Multi-Corner (MMMC) view (.view) let SoCE generate template (only the 1st time), then manually edit
- IO assignment file (.io) Same ways as above
- Configuration file (.conf) Same ways as above
- Clock specs (.clocktch) Same ways as above
- Timing library (.lib)
- Layout exchange format (.lef)
- Capacitance table (.captab)
- Parasitic extraction tech file (.cl)
- CeltlC noise analysis model (.cdb)
- Stream-out layer map (.map)

You Find from Foundry Design Kit
- Timing library (.lib)
- Layout exchange format (.lef)
- Capacitance table (.captab)
- Parasitic extraction tech file (.cl)
- CeltlC noise analysis model (.cdb)
- Stream-out layer map (.map)

Slight Modification of .vg

• When we’re at the final chip-level integration stage, we need to connect top-level module IO pins to the chip IO pads to communicate with outside environment
  - Create a new module (e.g. CHIP)
  - Instantiate your top-level module in module CHIP
  - Find names of IO pad modules in process verilog model
  - Manually instantiate the IO pad modules, connect it with top-level modules IO

```verilog
module FullAdd(S, A, B);
  output S;
  input A;
  input B;
  FADDXI U00 (S, A, B);
endmodule

module CHIP (CHIP_S, CHIP_A, CHIP_B);
  output CHIP_S;
  input CHIP_A;
  input CHIP_B;
  wire A, B;
  FullAdd TopLvl (S, A, B);
  PDO08CDG O1 (.Z(CHIP_S), .I(S));
  PDIDGZ ID (Z(A), .I(CHIP_A));
  PDIDGZ ID (Z(B), .I(CHIP_B));
endmodule
```
Slight Modification of .sdc

- Synthesis only considers propagation delay of cells. It doesn’t include accurate wire delay information.

- We can’t use same clock period as in synthesis for PnR.
  - Suggest: increase 0.2~0.4ns to accommodate parasitic

- I/O delay and the three major design rules (max-fanout, max-transition, max-cap.) might need modification too.

IO Assignment File

**Method 1:** Follow the format below, store as .io file, then do File → Load → I/O File

**Method 2:** Do Edit → Pin Editor to assign pins after your design is completely loaded. Save it (File → Save → I/O File) for future use
Getting Started!

Login `eeapps.seas.ucla.edu`

Enter `source /usr/apps/cadence/SETUP.EDI10.11`

Enter `encounter` (32-bit version) to open (note: don’t enter `encounter &`)

---

**MMMC View Files (1)**

**Method 1:** Follow the format below, store as `.view` file, then source it in `.conf` file

```plaintext
# Version:1.0 MMMC View Definition File
# Do Not Remove Above Line
create_rcc_corner -name rcc_corner_setup -cap_table {.captab_file} -qx_tech_file {.cl_file}
create_rcc_corner -name rcc_corner_hold -cap_table {.captab_file} -qx_tech_file {.cl_file}
create_op_cond -name op_cond_setup -library_file {.lib_file}
create_op_cond -name op_cond_hold -library_file {.lib_file}
create_library_set -name library_setup -timing {.lib_file} -si {.cdb_file}
create_library_set -name library_hold -timing {.lib_file} -si {.cdb_file}
create_constraint_mode -name SDC_setup -sdc_files " .sdc_file name"
create_constraint_mode -name SDC_hold -sdc_files " .sdc_file name"
create_delay_corner -name setup -library_set (library_setup) -rc_corner (rcc_corner_setup)
create_delay_corner -name hold -library_set (library_hold) -rc_corner (rcc_corner_hold)
create_analysis_view -name setup -constraint_mode {SDC_setup} -delay_corner {setup}
create_analysis_view -name hold -constraint_mode {SDC_hold } -delay_corner {hold}
set_analysis_view -setup (setup) -hold (hold)
```
Method 2: Do File → Import Design → Create Analysis Configuration and follow the wizard. Save it as .view file for future use.

---

Initial Setup (1)

File → Import Design

- For the first time, press **Save** to create an empty .conf file.
- Edit .conf to include required files (.vg, .sdc, .lef, .view).
- Also specify core utilization and CTS buffer list in .conf.
- Save it, then press **Load** to import the modified .conf.
Initial Setup (2)

global rda_input
set cwd your_working_directory
set rda_input(import_mode)
    {-treatUndefinedCellAsBox 0 -keepEmptyModule 1}
set rda_input(ui_netlist) ".vg file"
set rda_input(ui_netlisttype) {Verilog}
...
set rda_input(ui_topcell) "top_module_of_your_design"
...
set rda_input(ui_view_definition_file) ".view file"
set rda_input(ui_leffile) ".lef file"
set rda_input(ui_ccts_cell_list) 
    (check_standard_cell_library_manual_to_list_the_buffers 
    you_want_for_PnR_tool_to_do_clock_tree_synthesis)
set rda_input(ui_aspect_ratio) {aspect_ratio_you_want}
set rda_input(ui_core_util) {utilization_you_want}
...
set rda_input(ui_row_height) 
    (please_check_lef_file_for_the_height_of_std_cell)
set rda_input(ui_isHorTrackHalfPitch) {0}
set rda_input(ui_isVerTrackHalfPitch) {1}
set rda_input(ui_ioOri) {R0}
set rda_input(ui_isOrigCenter) {0}
set rda_input(ui_isVerticalRow) {0}
set rda_input(ui_delay_limit) {1000}
set rda_input(ui_net_load) {0.5pf}
set rda_input(ui_in_tran_delay) {0.0ps}
...
set rda_input(ui_preRoute_cap) {1}
set rda_input(ui_postRoute_cap) {1}
set rda_input(ui_preRoute_res) {1}
set rda_input(ui_postRoute_res) {1}
set rda_input(ui_shr_scale) {1.0}
set rda_input(ui_cmptol_c_thresh) {5.0}
set rda_input(ui_cpl_c_thresh) {3.0}
set rda_input(ui_tightness) {0.03}
set rda_input(ui_time_unit) {none}
...
set rda_input(ui_pwrnet) {VDD}
set rda_input(ui_gndnet) {VSS}
set rda_input(flip_first) {1}
set rda_input(double_back) {1}
set rda_input(assign_buffer) {1}
set rda_input(use_io_row_flow) {0}
set rda_input(ui_gen_footprint) {0}

SoC Encounter: General PnR Steps

- File preparation & initial setup
- Floor-planning
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells’ VDD/VSS to power ring)
- Placement & pre-CTS optimization
- Clock-tree synthesis (CTS) & post-CTS optimization
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- Detailed routing & post-route opt. (timing & SI driven)
- Layout verification & stream out
Floor-planing

Floorplan → Specify Floorplan

Specify core size indirectly using aspect ratio (height by width) & utilization or specify core size directly using dimension (H & W).

Reserve margin (core to IO boundary) for power ring.

Mapping Cell Power Pins to Global Rail

Floorplan → Connect Global Nets

Connect VDD pin to global VDD net. Do same thing to connect VSS. You can see the message about # of VDD/VSS pins connect to global VDD/VSS in the terminal window.
Mapping Power Nets to Global Rail

**Floorplan \(\rightarrow\) Connect Global Nets**

- Connect VDD net to global VDD net
- Do same thing to connect VSS
- You can see the message about # of VDD/VSS nets connect to global VDD/VSS in the terminal window

**Manually Place/Edit Hard Macro (if wanted)**

- Use to move objects to desired locations
- **Edit HALO:**
  - Floorplan \(\rightarrow\) Edit Floorplan \(\rightarrow\) Edit HALO
  - Prevent the placement of blocks & std. cells within specified HALO region in order to reduce congestion around a block.
Add Power Ring (1)

Power → Power Planning → Add Rings

Specify the names of global nets to deliver core power

Specify layer, metal width & spacing of power ring

(note: after setting up a new value, press Update to have the tool adjust the values to the nearest legal ones that can pass DRC for you)

Add Power Ring (2)

Power → Power Planning → Add Rings

using wire group w/o interleaving, number of bits = 2

using wire group with interleaving, number of bits = 2

we suggest this option

Press OK, or Apply→Cancel to add ring
How to Decide the Width of Power Ring?

Formula for metal width

\[
\text{Width} = \frac{\text{Total Power}}{\text{Supply Voltage}} \times \frac{1}{4} \times \frac{1}{\text{Metal Current Density}}
\]

- Rule-of-thumb current density \(\sim 4 \text{ mA}/\mu\text{m}\) (if \(>0.5 \mu\text{m}\) & below 100°C).
  For exact value please check process design-rule manual.
- The \(\frac{1}{4}\) is by assuming each edge delivers same amount of current.
  (Note: Above is only for \(V_{DD}\) or \(V_{SS}\), should times 2 to get both. Also need to plus metal spacing to pass DRC)

**Example:** A post-layout core consumes 80 mW@1 Volt. It uses M3 (top & bottom) and M4 (left & right) for power ring, each having density of 4mA/\mu m. Design rule asks the spacing b/w metals to be \(>0.5 \mu\text{m}\). We want a wire-group interleaved by 2.

**Solution:** Total width of \(V_{DD}\) or \(V_{SS}\) = 0.25\times(80\text{mW}/1\text{Volt})/(4\text{mA}/\mu\text{m}) = 5\mu\text{m}. Total power ring width = 5\mu m\times2\text{(for } V_{DD}+V_{SS}) + 3\times0.5\mu\text{m}\text{(spacing inside ring, as shown in previous page)} + 2\times0.5\mu\text{m}\text{(spacing b/w ring & core+IO pads)} = 12.5\mu\text{m}. This also tells you should specify 12.5\mu m spacing on each side of the core during floorplan. (Note: Due to 2-bit wire group, the width for each \(V_{DD}\) or \(V_{SS}\) is in fact 5\mu m/2=2.5\mu m)

Add Stripe to Enhance Power Delivery

Power → Power Planning → Add Stripes

Specify the stripe information (metal layer it uses, direction, width and spacing). Update to get DRC-legal values.

Tip: We usually only put vertical strips using M2 or M4 because std cells’ VDD/VSS rail naturally serve as horizontal stripes.

Start point of the first stripe

set-to-set distance

Click the two “…” options, in Via Generation page Switch to Advanced page to check the two “…” options. Can also setup wire group & interleaving option as we did for power ring.
Special Route: Connect Std. Cell Power Pins

Put VDD VSS in Net, check Follow Pins, then press OK. (change SRoute options to connect Pad Pins, Block Pins and Pad Rings)

SoC Encounter: General PnR Steps

- File preparation & initial setup
- Floor-planning
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells’ VDD/VSS to power ring)
- Placement & pre-CTS optimization
- Clock-tree synthesis (CTS) & post-CTS optimization
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
  - Detailed routing & post-route opt. (timing & SI driven)
  - Layout verification & stream out
Placement (1)

Placement: highly dependent on IO locations & floorplan, and can affect circuit performance

A bad placement

A good placement

Courtesy: Andrew Kahng, UCSD

Placement (2)

Place → Specify → Placement Blockage

Don’t place standard cells with specified metal layers under stripes

Place → Physical Cell → Add End Cap

Press Select
Placement (3)

Place ➔ Place Standard Cells and Blocks

**Pre-place opt. (PPO):** tool will delete all the buffers in gate-level netlist first, then do placement

**In-place opt. (IPO):** tool won’t delete any buffer in .vg. It does placement & pre-CTS optimization

Note: IPO will soon become obsolete, replaced by pre-CTS optimization. Therefore we don’t suggest to check that option. Only check PPO if needed.

Pre-CTS Optimization

**Timing ➔ Report Timing**
(check setup or hold to see timing info.)

**Optimize ➔ Opt. Design**
(check Pre-CTS, setup and DRV)

Tip: We suggest to fix all DRV during pre-CTS opt. stage, and check only Max Cap and Max Tran in post-CTS and post-Route. Also we can change to Incremental option if the timing violation is almost fixed.

Can try opt. for multiple times to pass the timing
SoC Encounter: General PnR Steps

- File preparation & initial setup
- Floor-planning
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells’ VDD/VSS to power ring)
- Placement & pre-CTS optimization
- Clock-tree synthesis (CTS) & post-CTS optimization
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- Detailed routing & post-route opt. (timing & SI driven)
- Layout verification & stream out

Clock Tree Synthesis (CTS) (1)

Clock tree propagate clock signals, through clock buffers, to each of the registers

Clock problem
- Heavy net loading
- Long insertion delay
- Skew
- Coupling
- Power consumption
- Electromigration
Clock Tree Synthesis (CTS) (2)

Clock → Synthesize Clock Tree

Press **Gen Spec** for the first time. Select all **clock buffers** and **clock inverters** for CTS.

If already have **Clock.ctstch**, just put that file in the **Clock Specification Files** and press **OK** to synthesize clock tree.

Press **OK** to get **Clock.ctstch**

Display Clock Tree to See Clock Distribution

Clock → Display → Display Clock Tree

Click to select single object, Shift-Ctrl to deselect...
Post-CTS Optimization

Optimize → Optimize Design
(check Post-CTS, setup and DRV to fix setup time. Change to hold to fix hold time)

press Mode to enable advanced opt. features

Set setup slack threshold to be >0 helps hold-time fixing

Orig. max. utilization=0.95, can change to higher value

SoC Encounter: General PnR Steps

• File preparation & initial setup
• Floor-planning
  ▪ Specify layout size & power rail names
  ▪ Manually place hard macros (e.g. SRAM blocks, if any)
  ▪ Add power ring & power stripes
  ▪ Special route (connect cells’ VDD/VSS to power ring)
• Placement & pre-CTS optimization
• Clock-tree synthesis (CTS) & post-CTS optimization
  ▪ Generate clock specs. & synthesize clock tree
  ▪ Display clock and optimize design
• Detailed routing & post-route opt. (timing & SI driven)
• Layout verification & stream out
Detailed Route

Route → Nano Route → Route

Step 1: Press Mode, go to DFM, check Spread Wire

Step 2: Check following options (can set multi-CPU to speed up)

For optimized timing, set Effort=10; if design highly congested, set Effort=1

Post-Route Optimization

Optimize → Optimize Design

(check Post-Route, setup and DRV to fix setup time. Change to hold to fix hold time)

Note: Must fix setup/hold time first, then fix timing again by checking the Include SI option for further noise-aware optimization.
Adding Fillers to Fill Unplaced Core Area

Place → Physical Cell → Add Filler

Do Select → Add → Close to list the fillers you want to use. Remember: add from wider filler to narrower filler. Tip: use left click + drag to select multiple fillers at once.

SoC Encounter: General PnR Steps

- File preparation & initial setup
- Floor-planning
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells' VDD/VSS to power ring)
- Placement & pre-CTS optimization
- Clock-tree synthesis (CTS) & post-CTS optimization
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- Detailed routing & post-route opt. (timing & SI driven)
- Layout verification & stream out
Verify Layout

- **DRC:** Verify ➔ Verify Geometry (should get zero error, otherwise gotta fix them)

  ```
  Begin Summary ...
  Cells : 0
  SameNet : 0
  Wiring : 0
  Antenna : 0
  Short : 0
  Overlap : 0
  End Summary
  Verification Complete : 0 Viol. 0 Wrngs.
  ```

- **Antenna violation:** Verify ➔ Verify Process Antenna

  ![Image]

- **LVS:** Verify ➔ Verify Connectivity

  ```
  Begin Summary
  Found no problems or warnings.
  End Summary
  ```

Stream Out GDS File

**Design ➔ Save ➔ GDS/OASIS**

- Decide the name of `.gds` & library it’s gonna be belong to. **Source the .map from foundry design kit**
- Check Structure Name, press **OK** to generate

![Image]
Post-Layout Parasitic/LEF/Netlist/SDF/LIB

Parasitic: **Timing → Extract RC**

Check **Save SPEF** to and name the `.spef` file. Don’t forget to generate `.spef` for both best- and worse-case RC corners. We use `.spef` in post-layout simulation.

LEF, Netlist, SDF: **Using commands**
- `saveNetlist XXX_sim.vg` (for post-sim.)
- `saveNetlist -flattenBus -includePowerGround XXX_lvs.vg` (for Calibre LVS check)
- `lefOut XXX.lef` - stripePin
- `write_sdf XXX.sdf`

LIB: **Using commands**
- `setAnalysisMode -checkType setup`
- `do_extract_model XXX_setup.lib`
- `setAnalysisMode -checkType hold`
- `do_extract_mode XXX_hold.lib`

Post-Layout Simulation & Bottom-Up Design

- **Post-layout simulation**
  - Prepare `.vg`, and `.sdf` from SoC Encounter
  - Do the same thing as in pre-layout gate-level simulation

- **Bottom-up hierarchical design flow**
  - Once having `.lef` and `.lib`, your design can be treated as a “**big** standard cell”. We call it as a **macro block**.
  - At higher-level integration, just source that `.lef` and `.lib` and do the same thing as in bottom-level PnR.
Summary

- Physical synthesis tool processes the design from gate to transistor level
- Slightly modify .vg and .sdc if necessary
- Take some time to carefully think about your floorplan & I/O locations to get optimized circuit performance
  - Can affect wiring congestion
  - Suggested core dimension: as square as possible
- Optimize timing in each design stage
  - pre-cts: setup
  - post-cts: setup + hold
  - post-route: setup + hold + SI
- Make sure to pass build-in DRC & LVS before finish