EE671: VLSI DESIGN
SPRING 2024/25

LAXMEESHA SOMAPPA

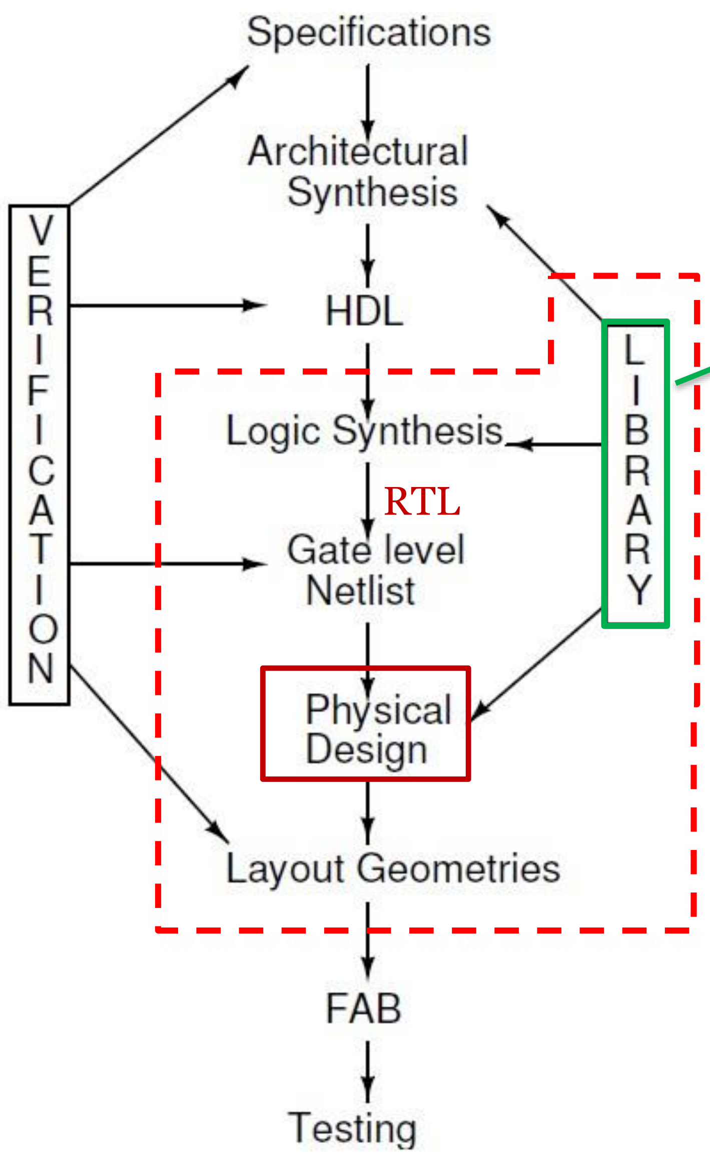
DEPARTMENT OF ELECTRICAL ENGINEERING

IIT BOMBAY

laxmeesha@ee.iitb.ac.in

Lecture – 23
Physical Design

Physical Design



Standard Cell Library:
.v, .lef. Lib, .sp, .gdsII/.mag

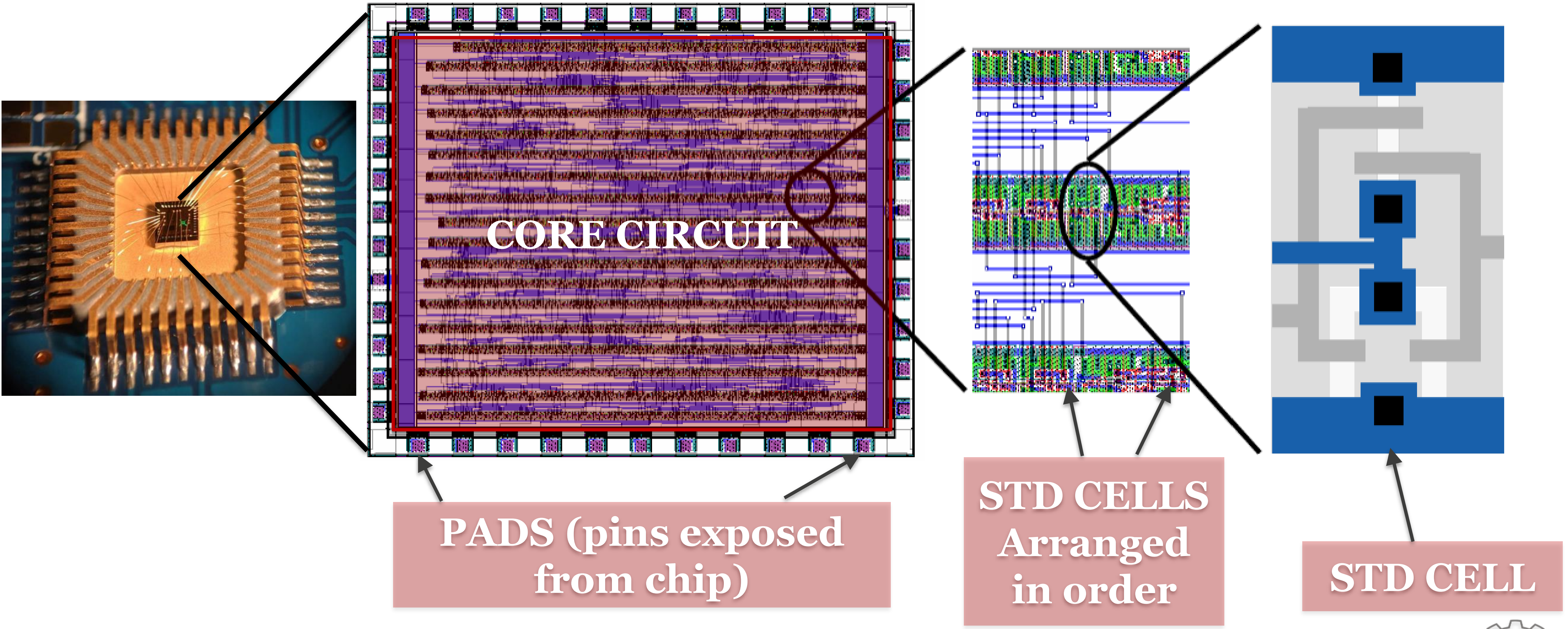
Physical Design:

- Input: RTL (mapped to std. cell library)
- Outputs:

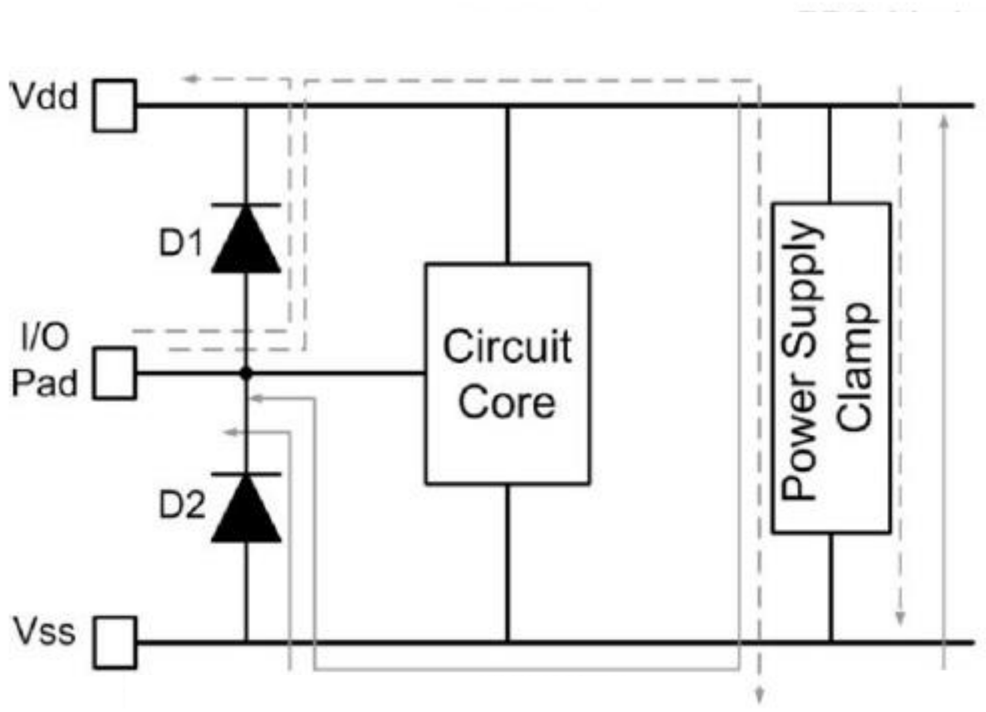
final chip layout (GDSII), final chip RTL, final chip LEF, final chip timing data (SDF)

- GDSII: will be sent out to fabrication foundry to produce the chip

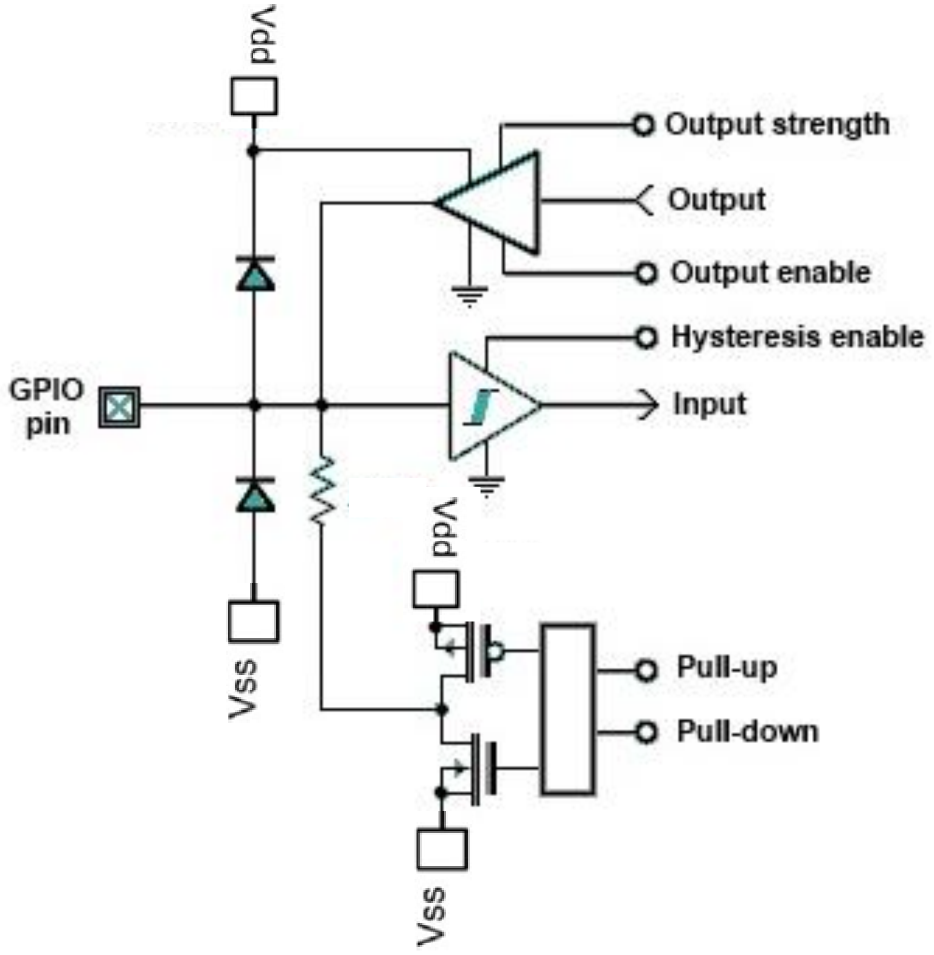
MOTIVATION


CORE CIRCUIT
PADS (pins exposed from chip)
STD CELLS Arranged in order
STD CELL

TYPICAL ESD PROTECTION CIRCUIT

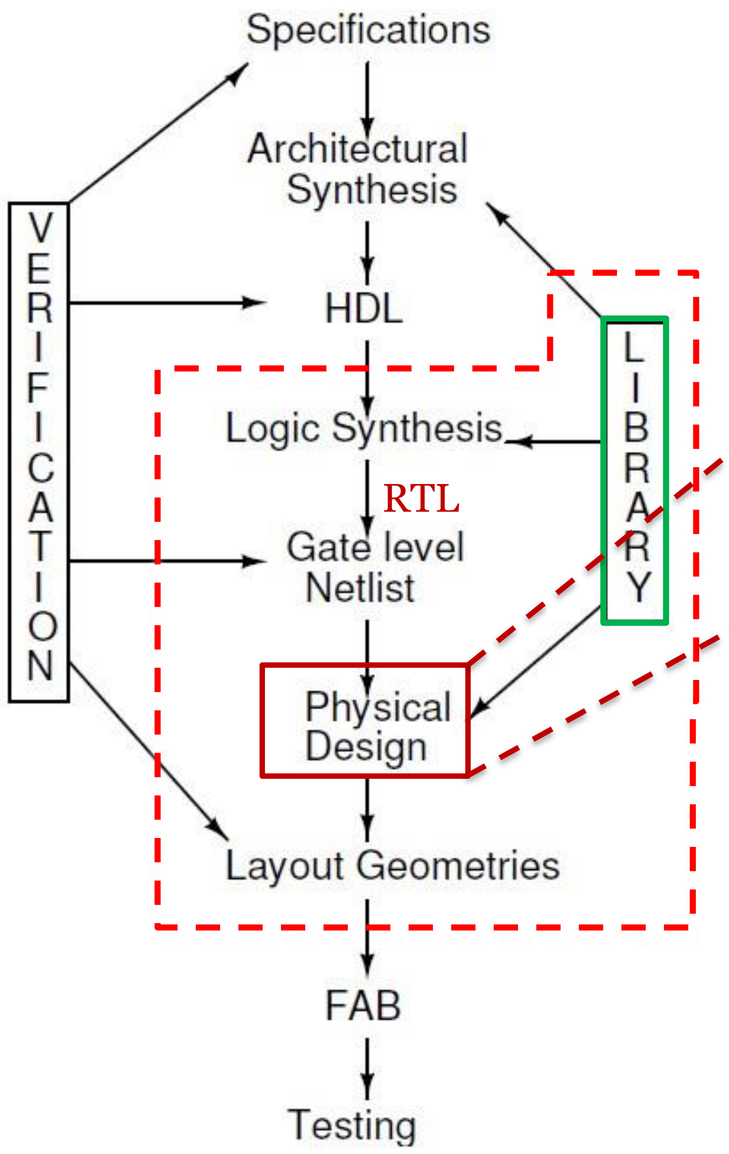


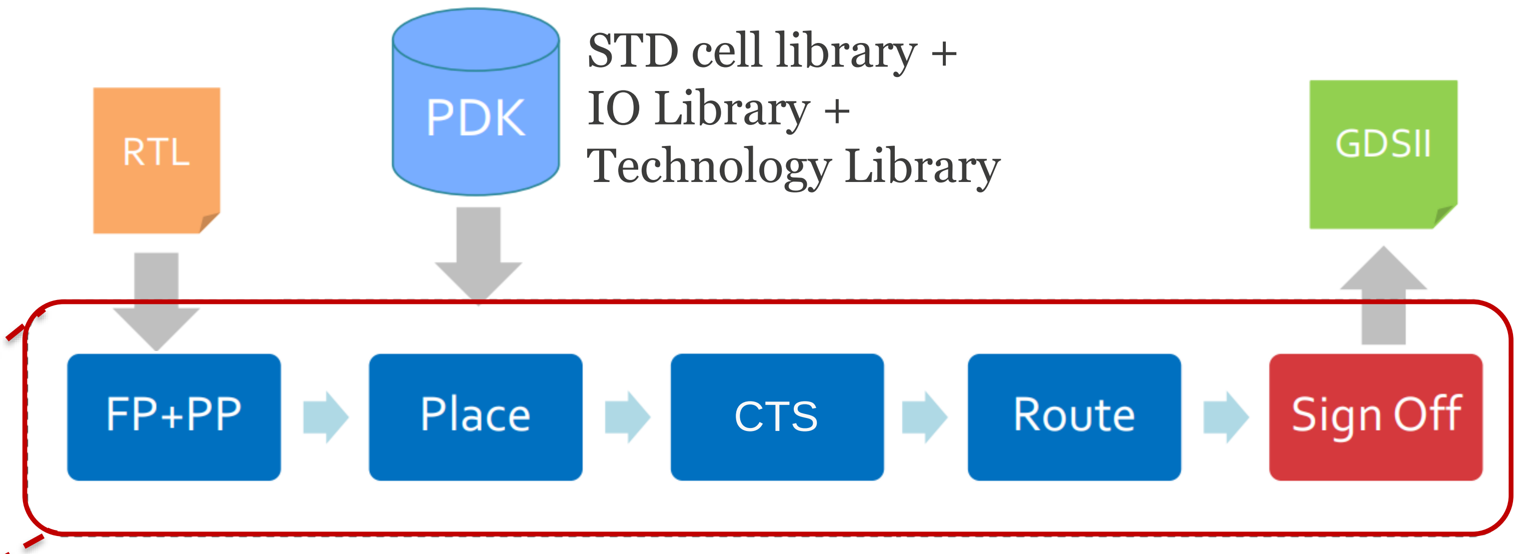
Typical Analog I/O PAD



Typical bi-directional, programmable digital I/O PAD

PHYSICAL DESIGN

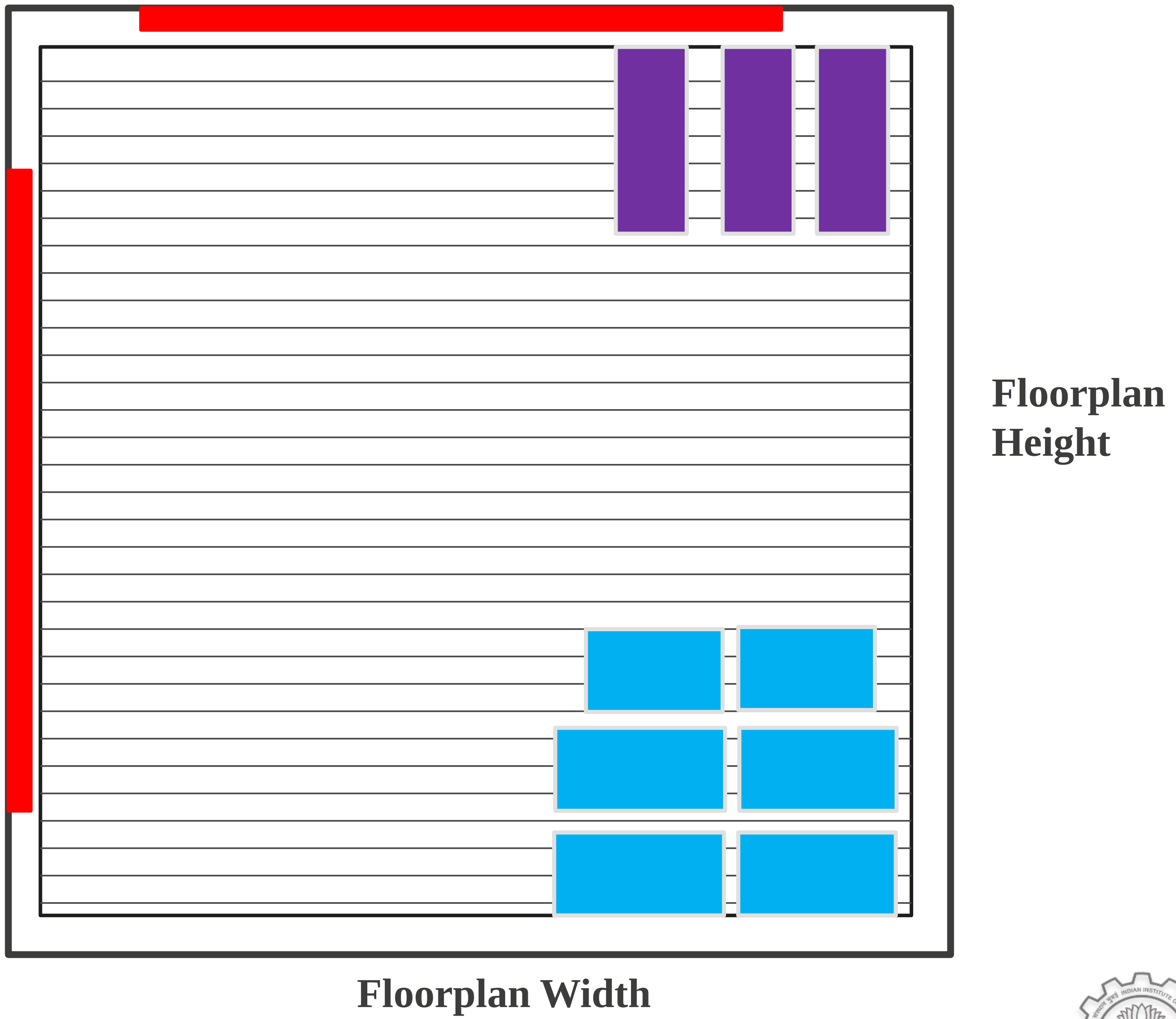




1. FP + PP: Floor planning + Power planning
2. Place: Placement
3. CTS: Clock tree synthesis
4. Route: Routing (Global and local)
5. Sign-off: final checks before gdsII submission
   a) DRC: Design rule check
   b) LVS: layout – vs - schematic
   c) STA: Static Timing Analysis

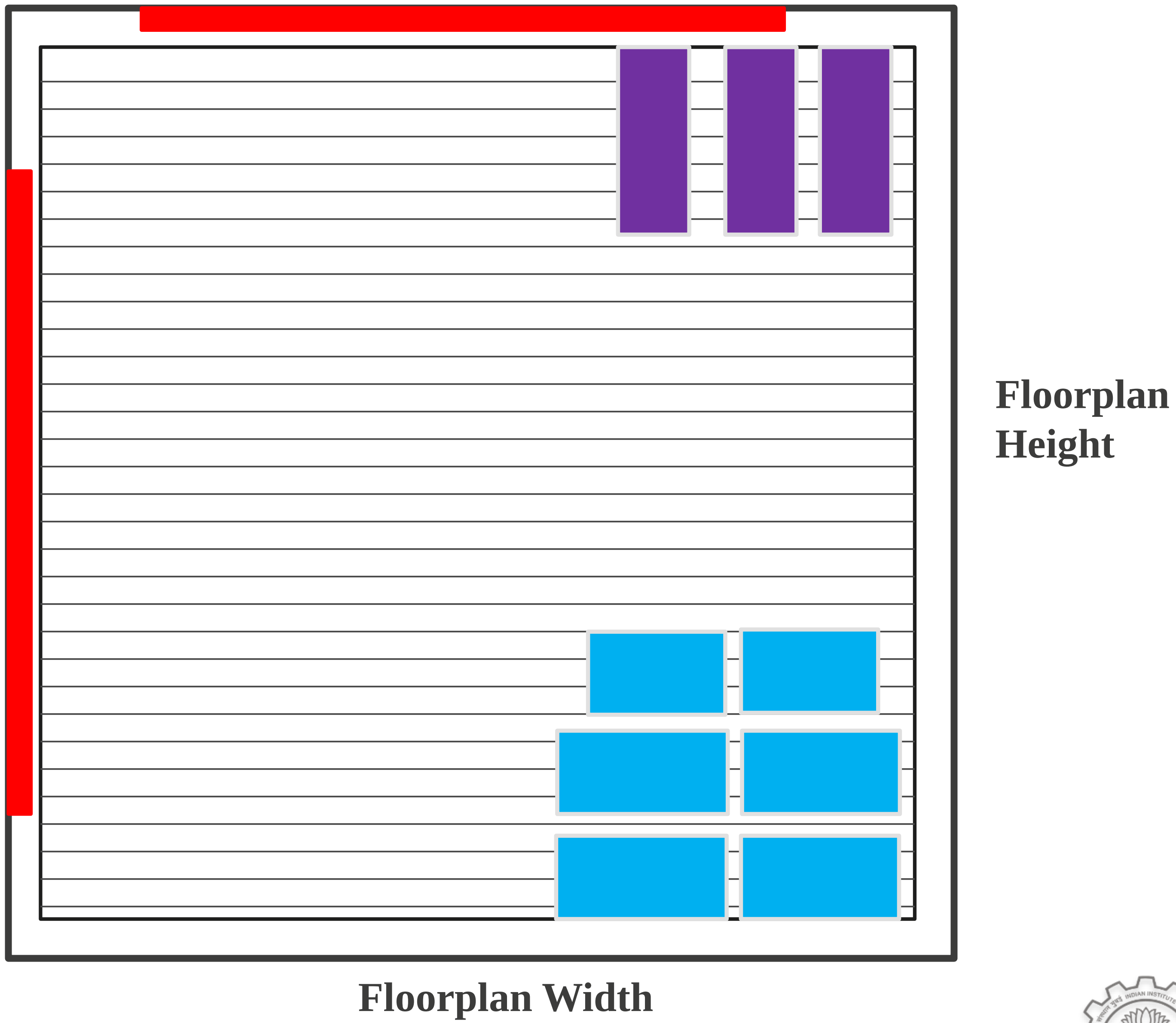
PHYSICAL DESIGN: FLOORPLAN

- ❑ Floor planning:
- ❑ Define core area size (Area: Width & Height)
- ❑ Create cell rows (std cell height)
- ❑ IO placement:
  - ❑ Define what kind of IO (analog/digital/VDD/VSS) should reside where
- ❑ Macro cell placement:
  - ❑ Macros are modules for which you already have a layout (Ex: third part memory, ARM core etc.)



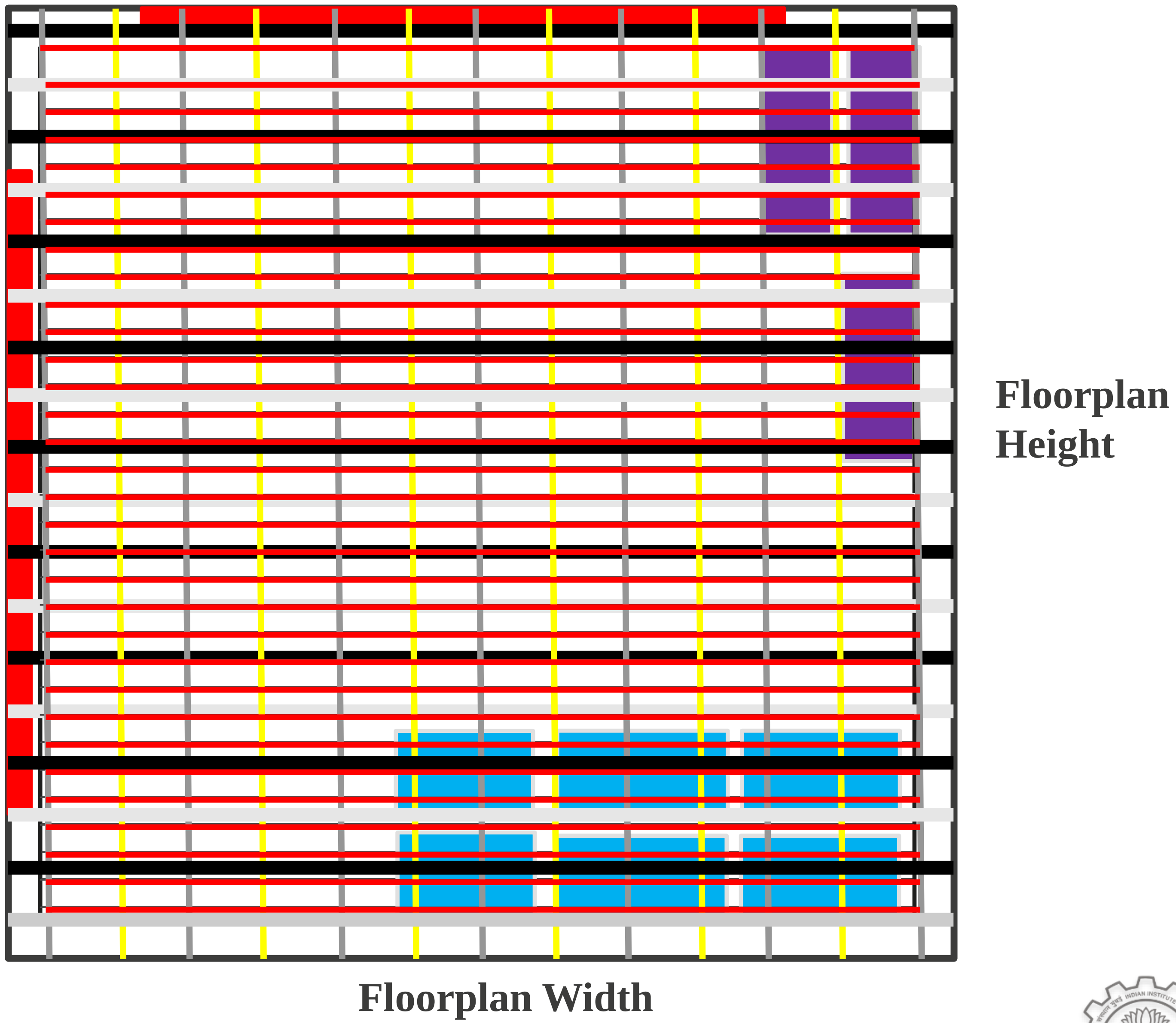
PHYSICAL DESIGN: FLOORPLAN

- ❑ Macro Placement:
- ❑ Make smart choices:
  - ❑ Timing: be aware of what block talk to what so that you can place them closer (for minimal routing)
  - ❑ Macro pins oriented towards nearest standard cells
  - ❑ Distance of memory macro from standard cells
  - ❑ Plan to avoid routing congestion (number of connections from macro)

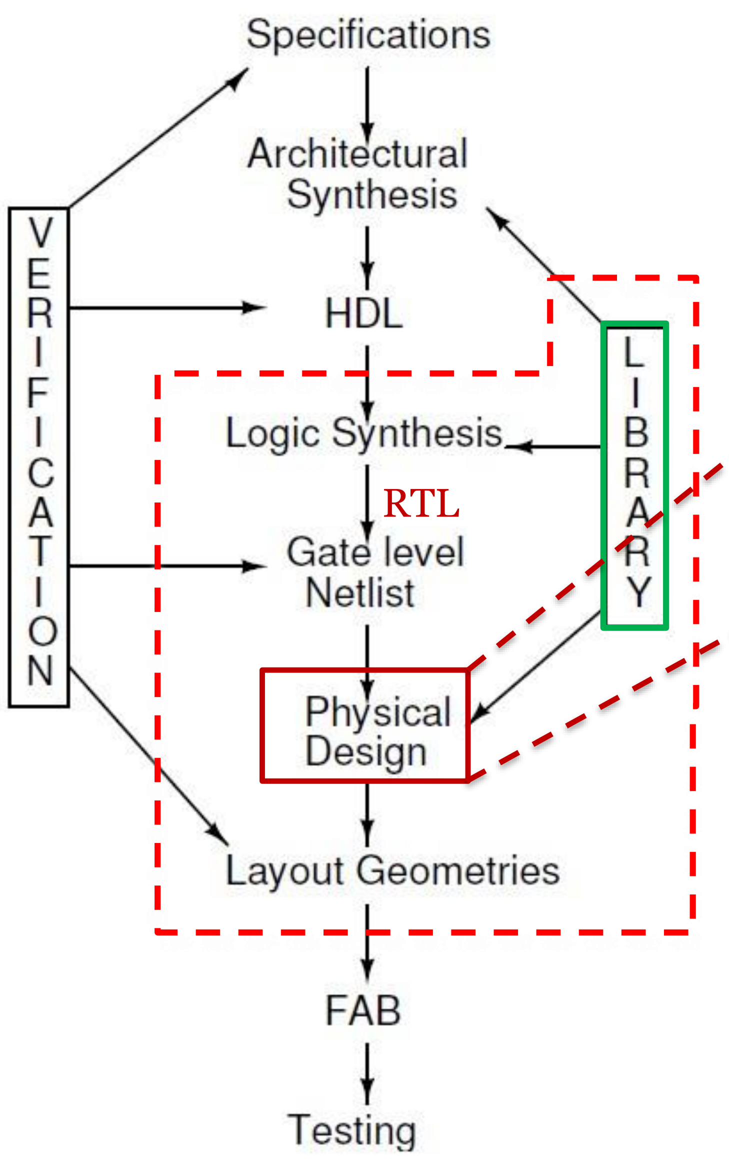


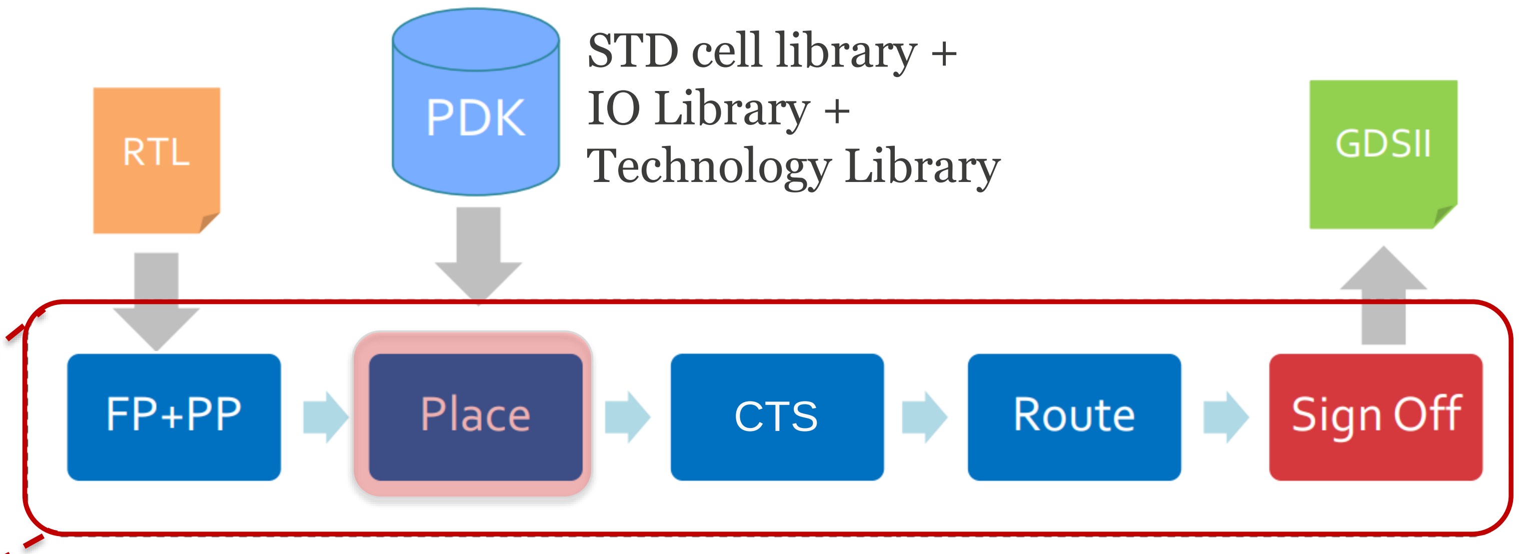
Physical Design: Floorplan

- ❑ Power Planning:
- ❑ Supply VDD/VSS to standard cells and Macros
- ❑ Complex designs:
  - ❑ Be aware of the power consumption of blocks – IR drop on the power rail/grid – degrades circuit performance



PHYSICAL DESIGN

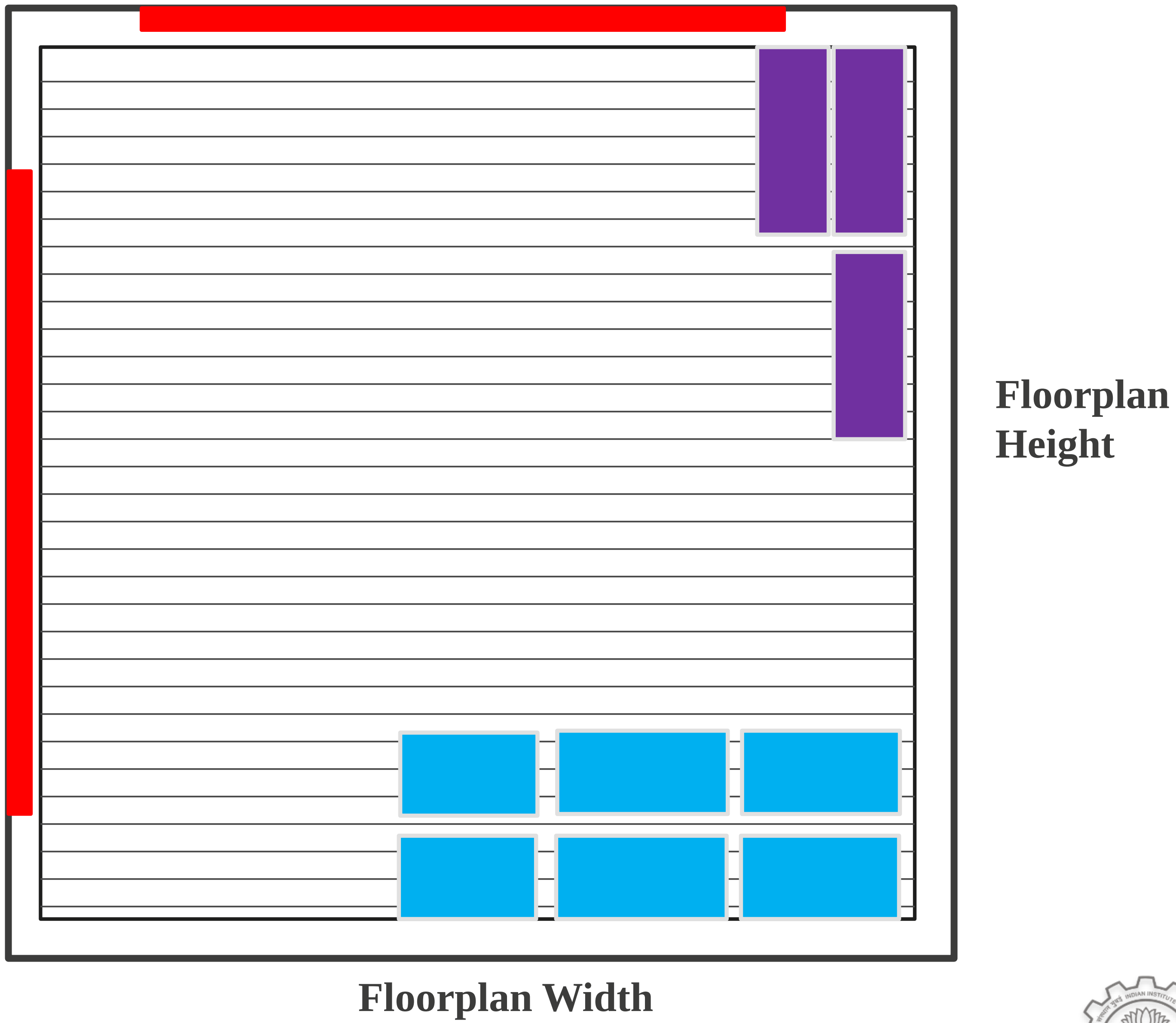




1. FP + PP: Floor planning + Power planning
2. Place: Placement
3. CTS: Clock tree synthesis
4. Route: Routing (Global and local)
5. Sign-off: final checks before gdsII submission
   a) DRC: Design rule check
   b) LVS: layout – vs - schematic
   c) STA: Static Timing Analysis

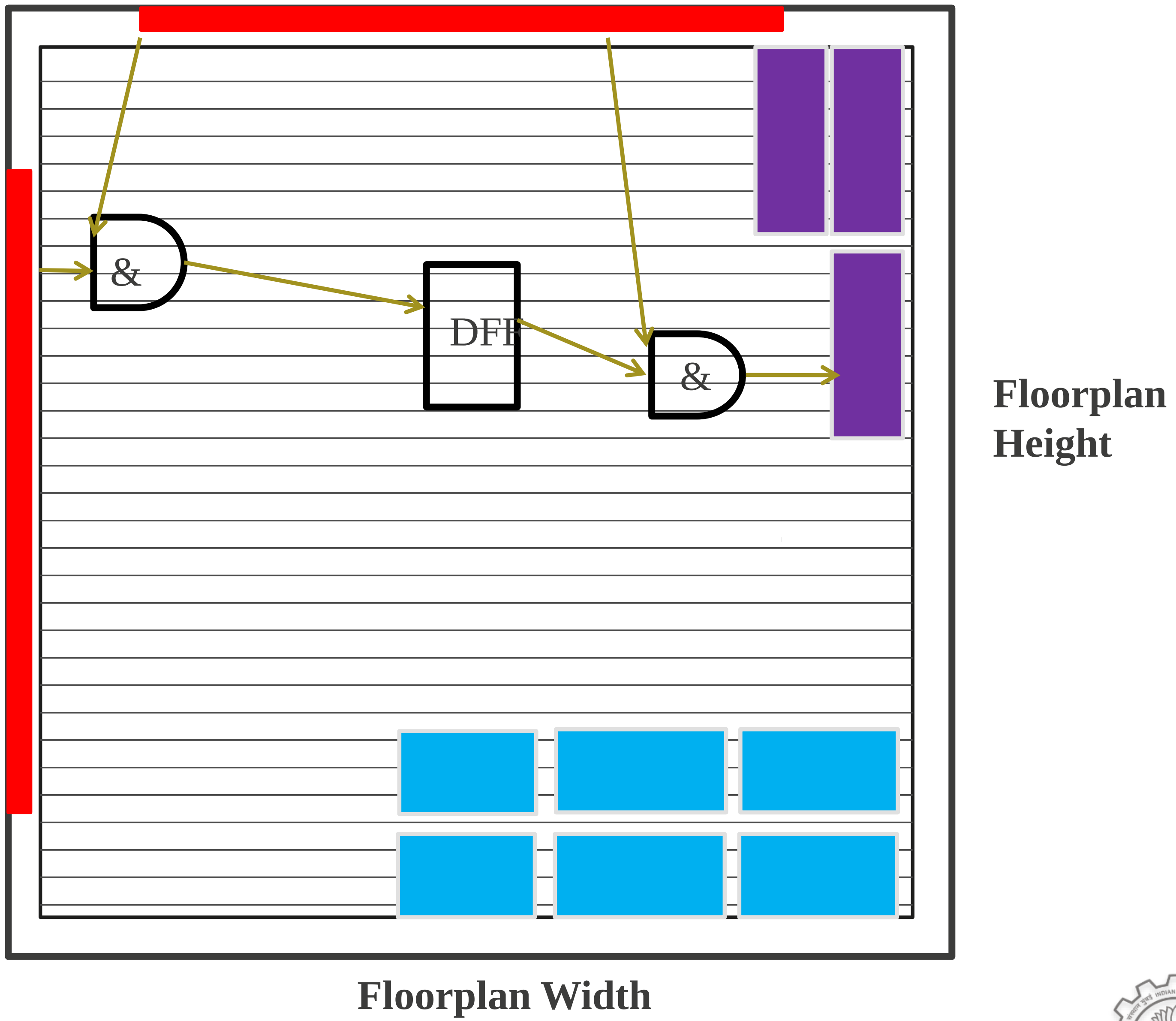
Physical Design: Placement

- ❑ Placement: mainly placement of std cells. Following steps:
  - ❑ Placement optimization
  - ❑ Timing optimization
  - ❑ Parasitic estimation based on global route (wire line model discussed earlier)
  - ❑ Congestion removal
  - ❑ Standard cell legalization

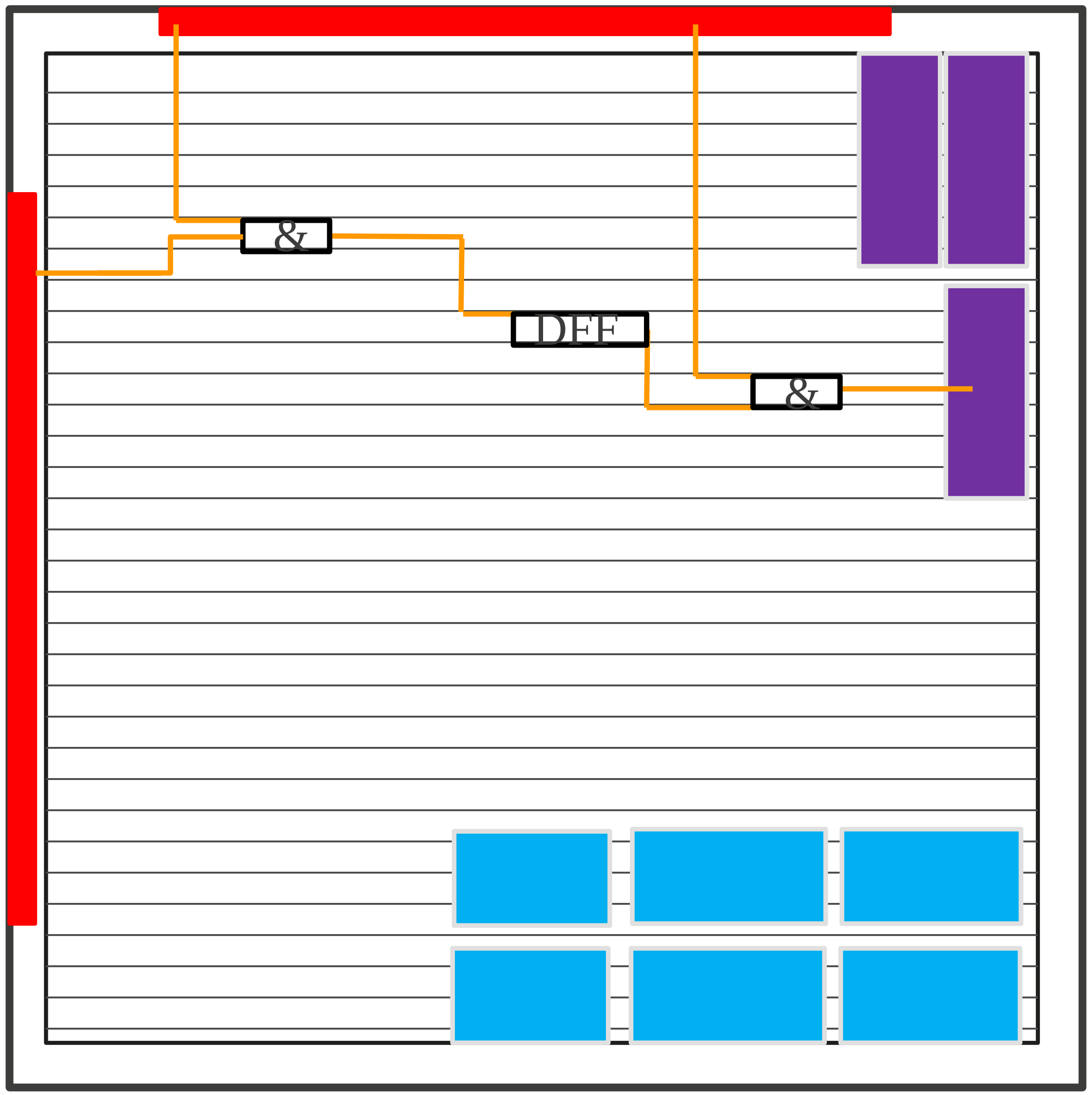


Physical Design: Placement

- Placement optimization
  - Placing cells in the vicinity
- Timing optimization
  - Size cells (x1, x2, x4 ..)
  - Buffer insertion
  - Based on wire delay and input capacitance



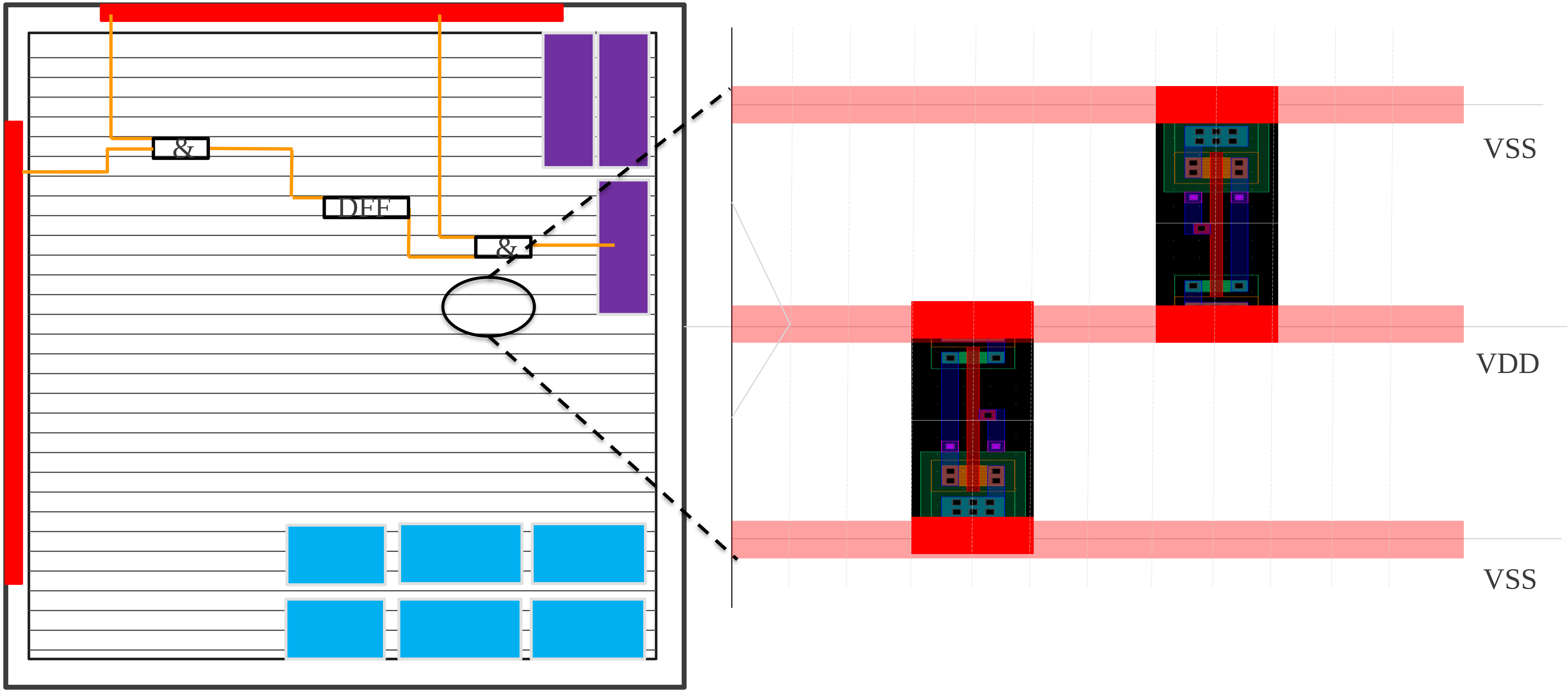
Physical Design: Placement



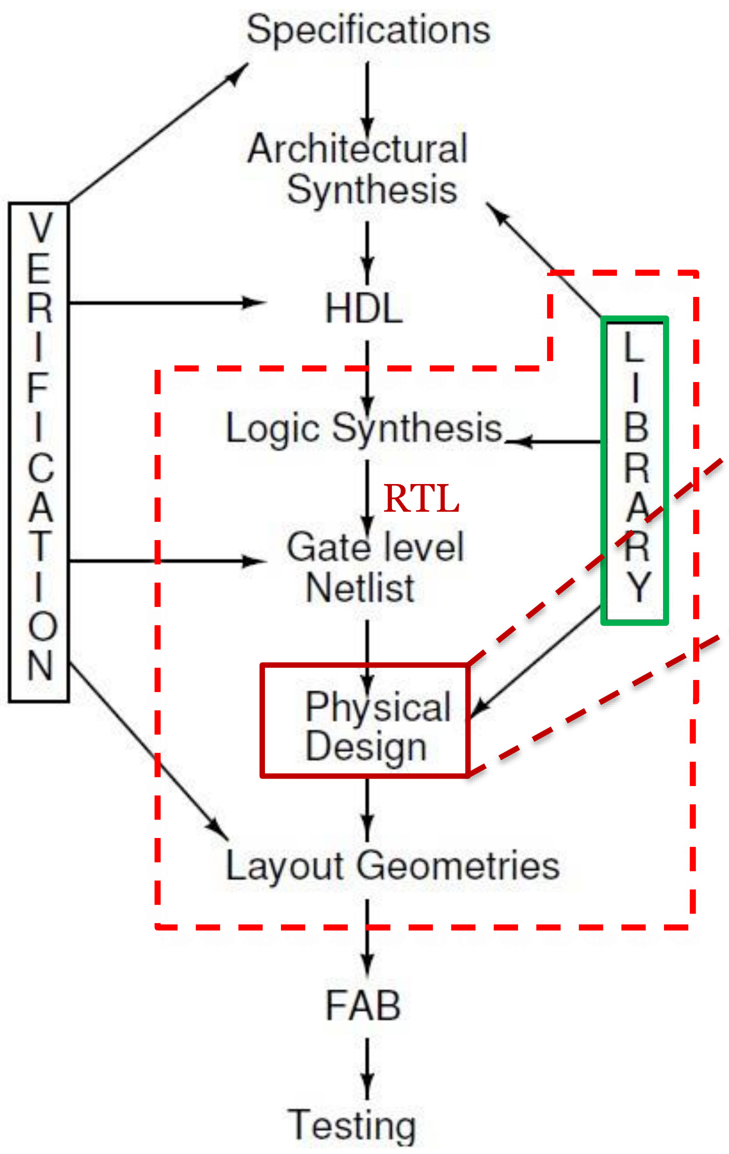
Floorplan Height

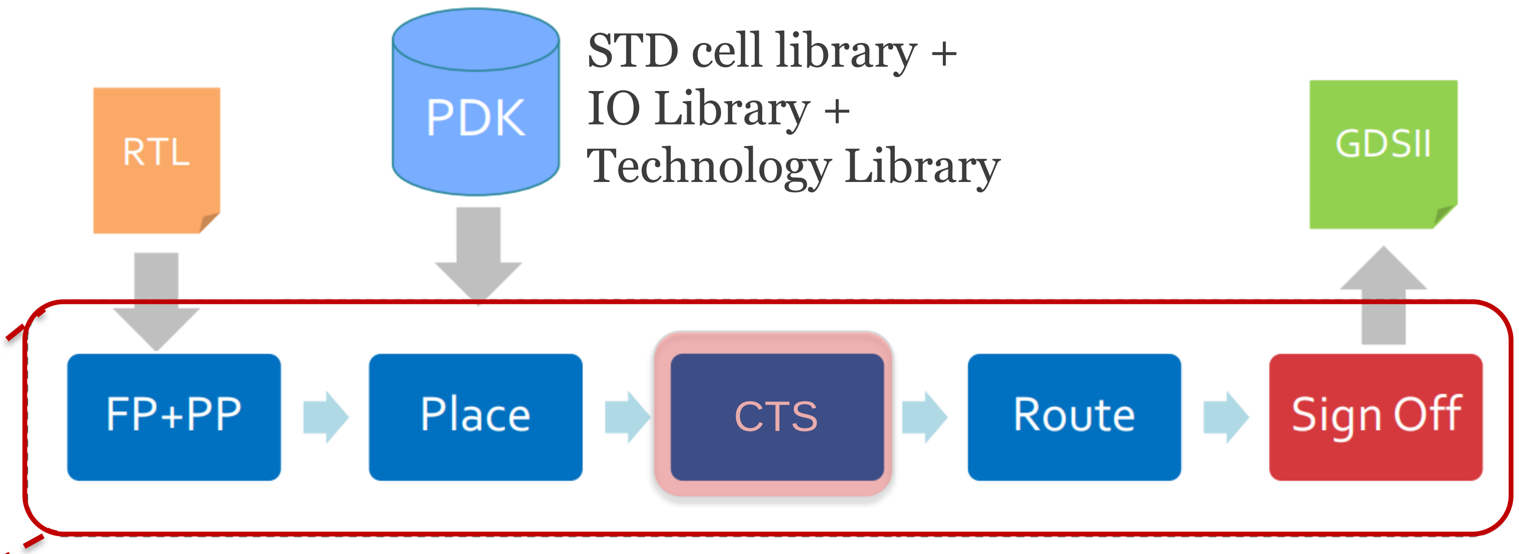
Floorplan Width

Physical Design: Placement legalization



Physical Design

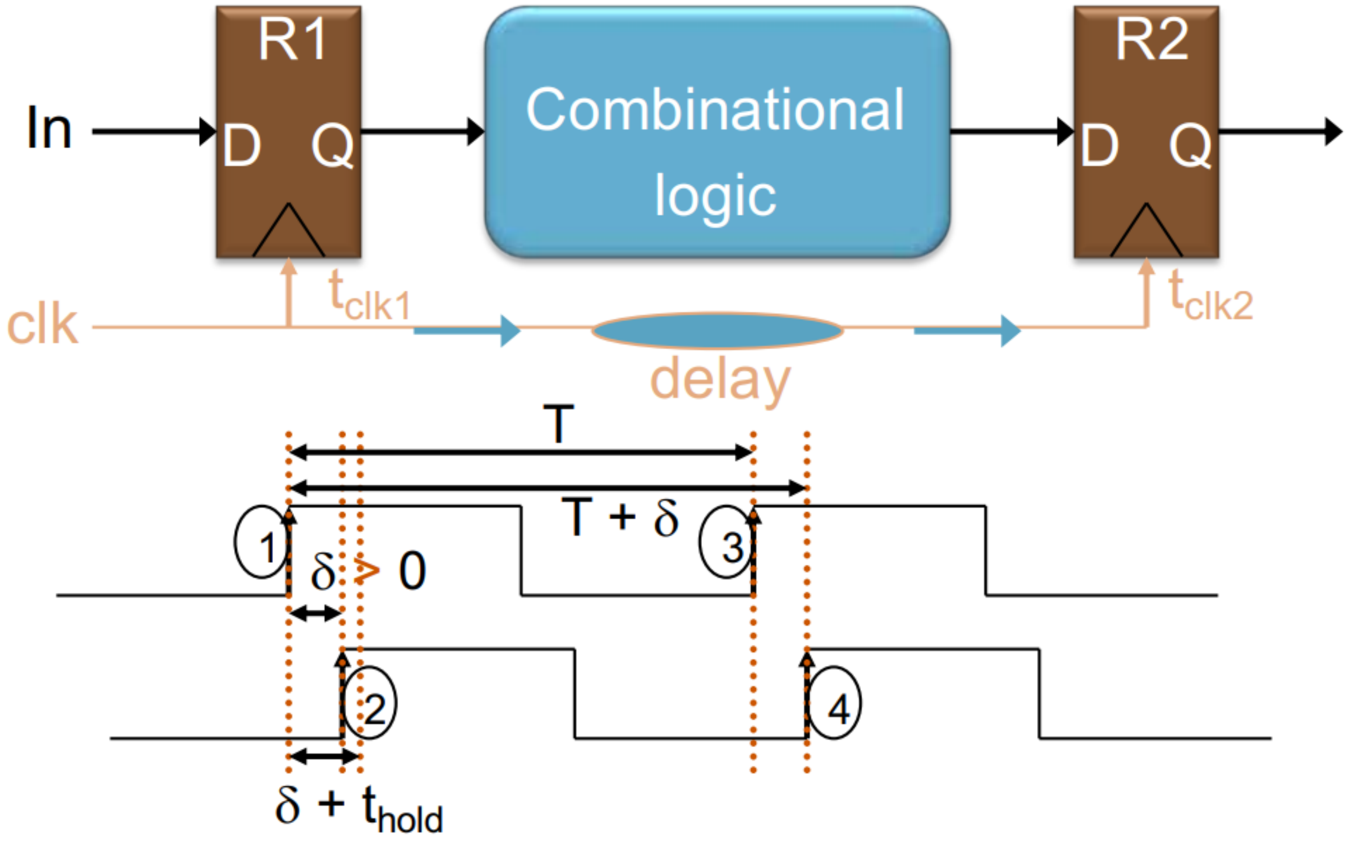




1. FP + PP: Floor planning + Power planning
2. Place: Placement
3. CTS: Clock tree synthesis
4. Route: Routing (Global and local)
5. Sign-off: final checks before gdsII submission
   a) DRC: Design rule check
   b) LVS: layout – vs - schematic
   c) STA: Static Timing Analysis

Physical Design: CTS Motivation

- Positive clock skew: δ > 0

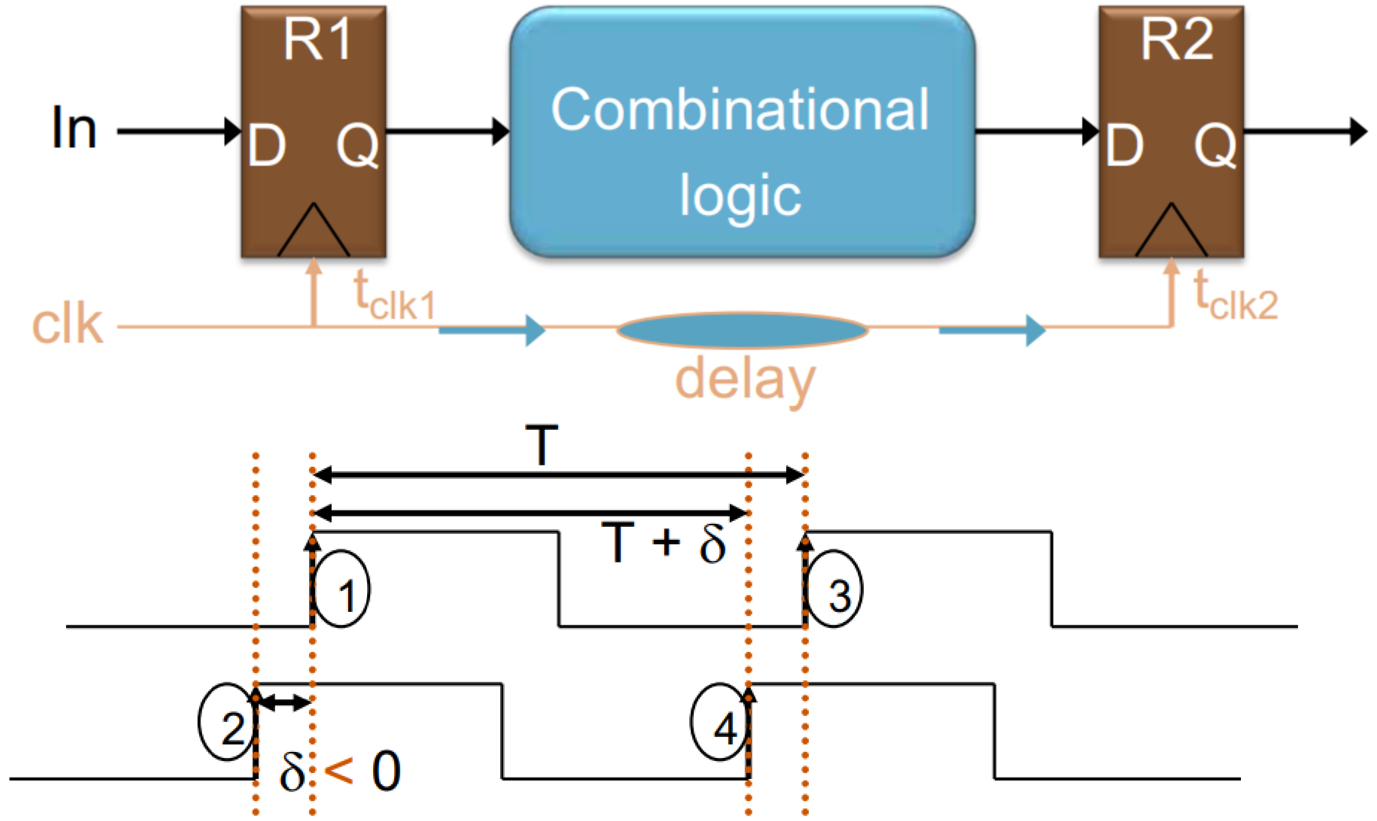


$$T + \delta \geq t_{clk\text{-}Q} + t_{p\text{-}logic(max)} + t_s \quad \rightarrow \quad T \geq t_{clk\text{-}Q} + t_{p\text{-}logic(max)} + t_s - \delta$$ 😃

$$t_h + \delta \leq t_{clk\text{-}Q} + t_{p\text{-}logic(min)} \quad \rightarrow \quad t_h \leq t_{clk\text{-}Q} + t_{p\text{-}logic(min)} - \delta$$ ☹

Physical Design: CTS Motivation

- Negative clock skew: $\delta < 0$

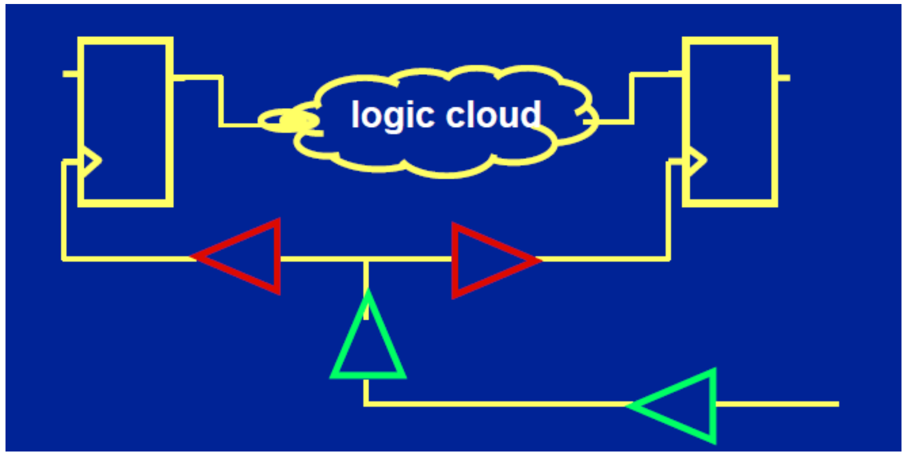


$T - |\delta| \geq t_{clk\text{-}Q} + t_{p\text{-}logic(max)} + t_s \rightarrow T \geq t_{clk\text{-}Q} + t_{p\text{-}logic(max)} + t_s + |\delta|$ ☹

$t_h - |\delta| \leq t_{clk\text{-}Q} + t_{p\text{-}logic(min)} \rightarrow t_h \leq t_{clk\text{-}Q} + t_{p\text{-}logic(min)} + |\delta|$ 😃

PHYSICAL DESIGN: CTS

- ❑ Need clock to reach all points ideally at the same time – to avoid timing violations (setup/hold) due to clock skew



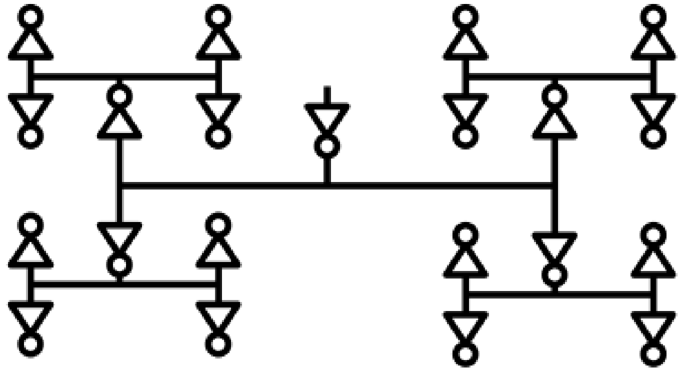
- ❑ Create a clock distribution network
  - ❑ Three broad categories:
    - ❑ Clock tree
    - ❑ Clock mesh
    - ❑ Clock spine

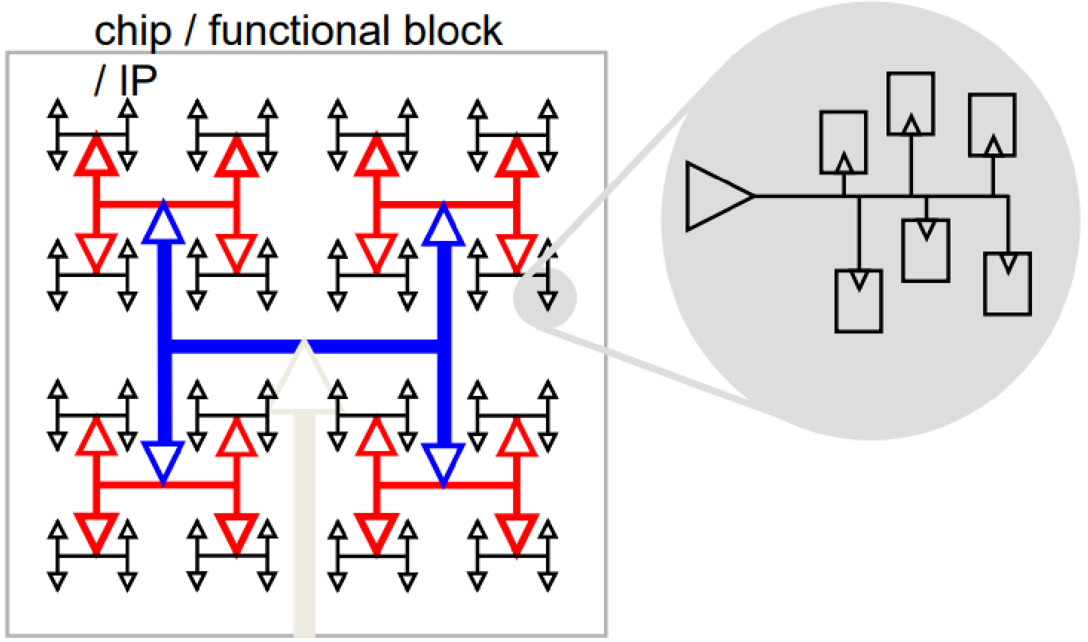
$\left.\begin{matrix} \\ \end{matrix}\right\}$ Will not be covered in this course

Physical Design: CTS

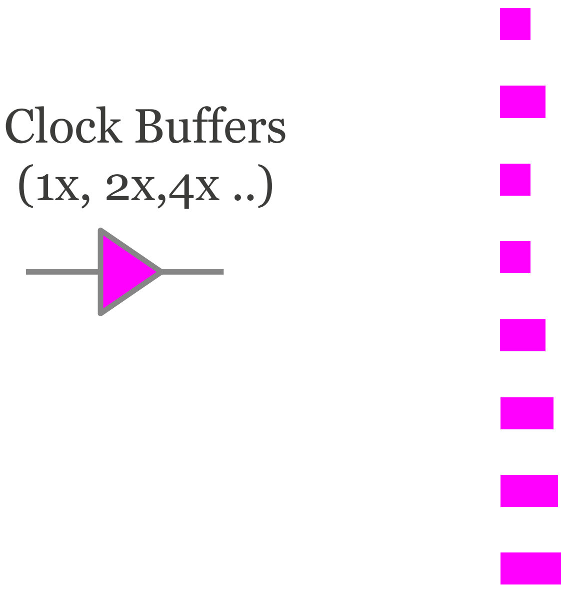
- Clock tree: popularly used version is H-tree

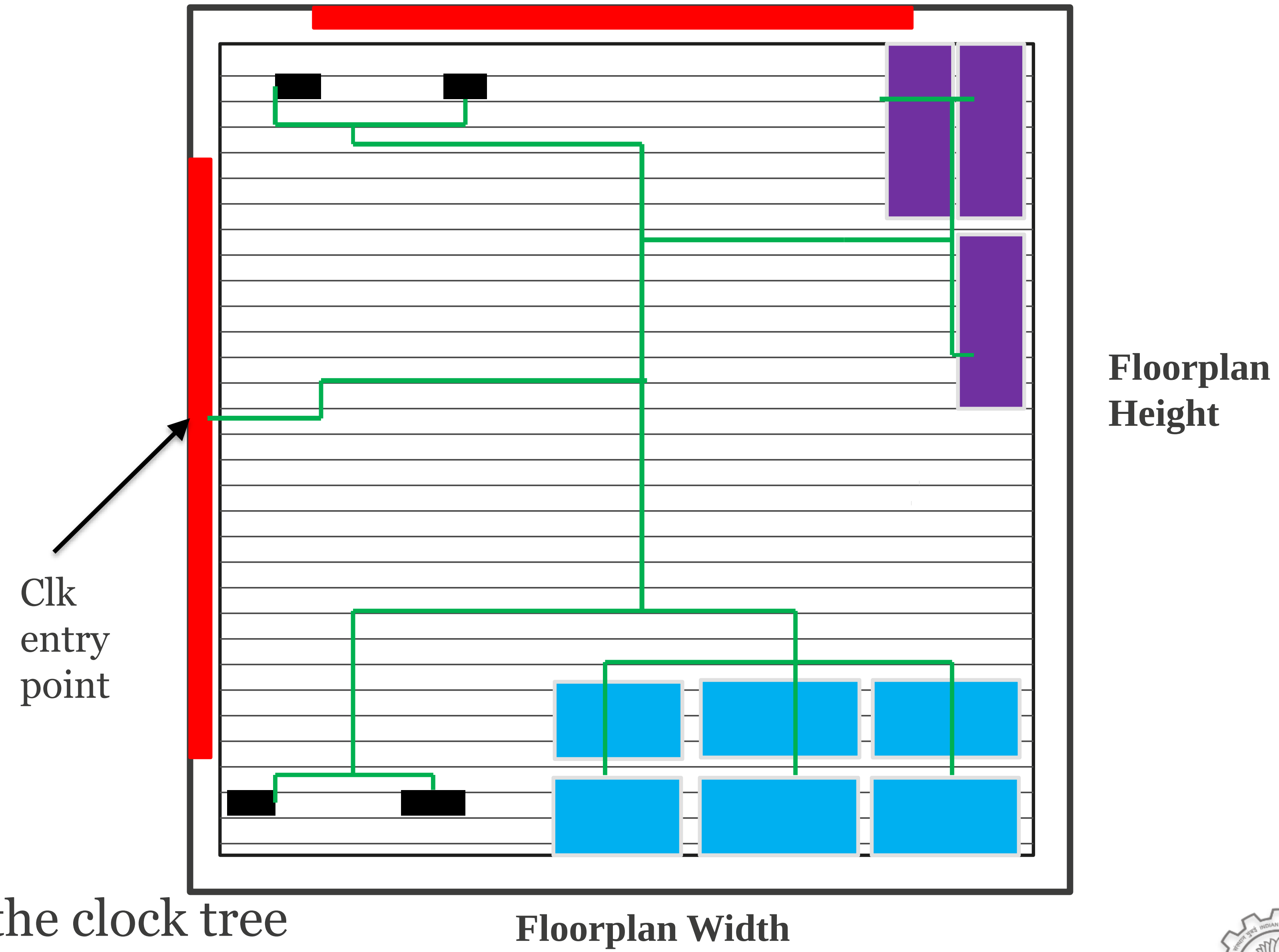
- Recursively build H-style structure to match wire length
- Inserting a buffer at branching point
- More realistic: tapered H-tree





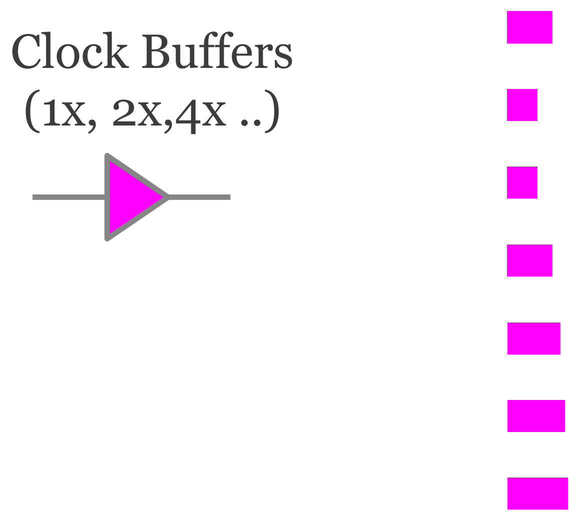
PHYSICAL DESIGN: CTS

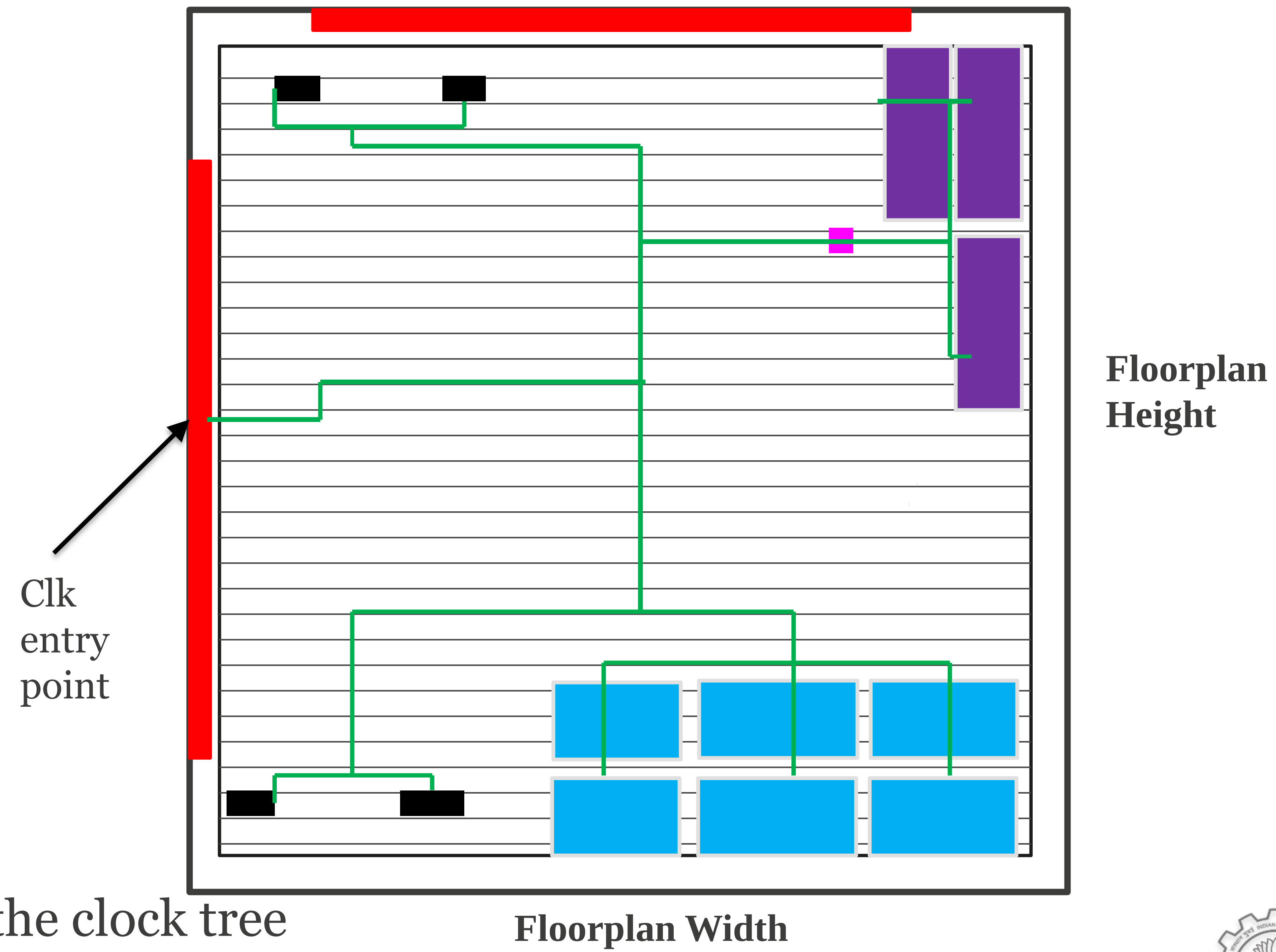




- ❑ CTS Involves two steps:
  - ❑ Create clock tree
  - ❑ Insert clock buffers in the clock tree

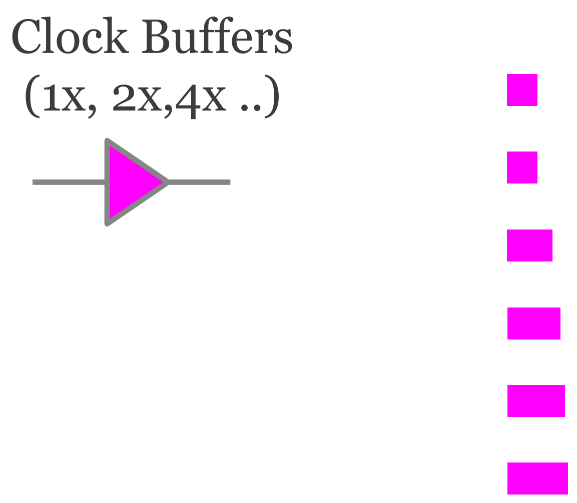
Physical Design: CTS

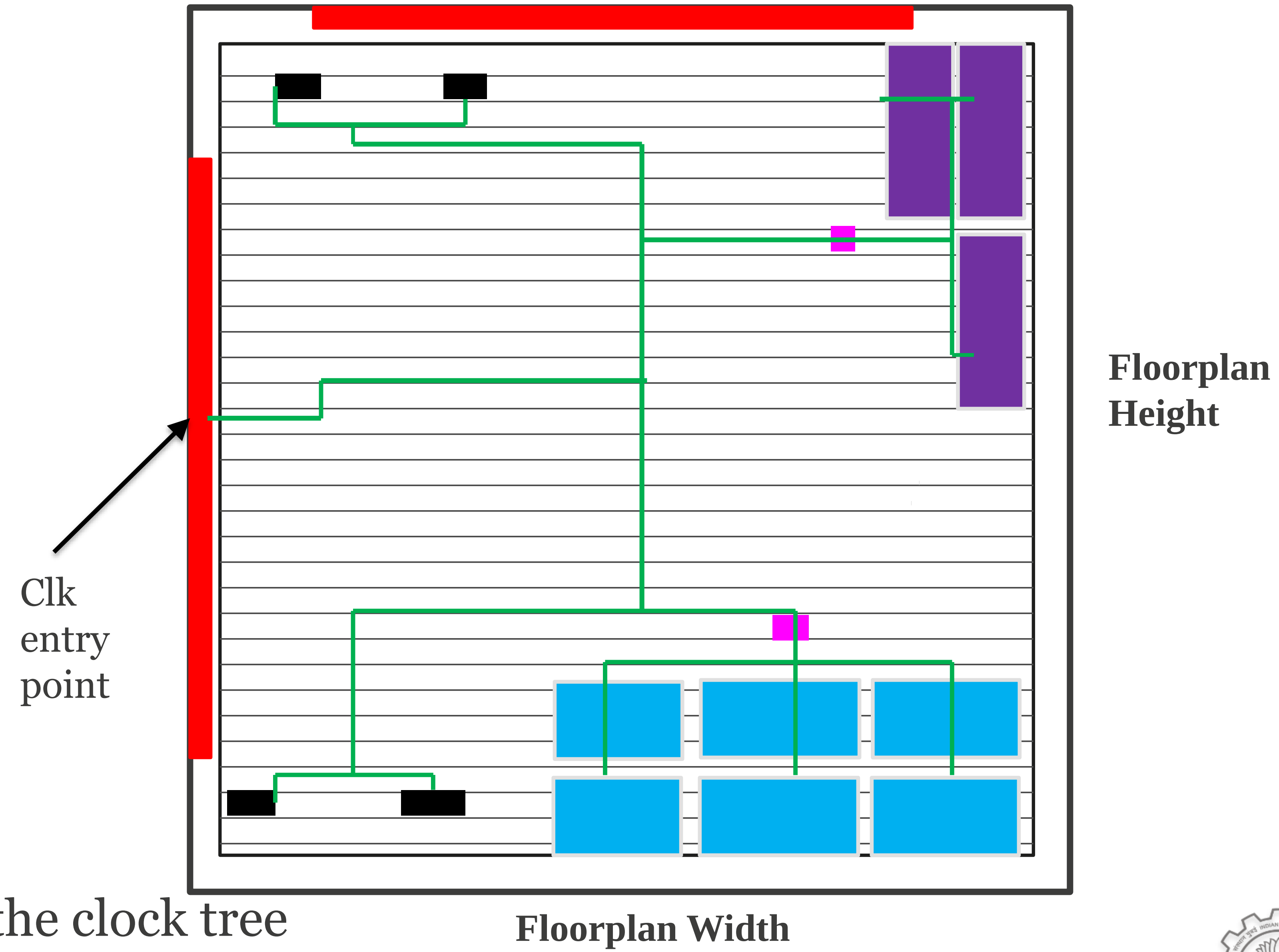




- ❑ CTS Involves two steps:
  - ❑ Create clock tree
  - ❑ Insert clock buffers in the clock tree

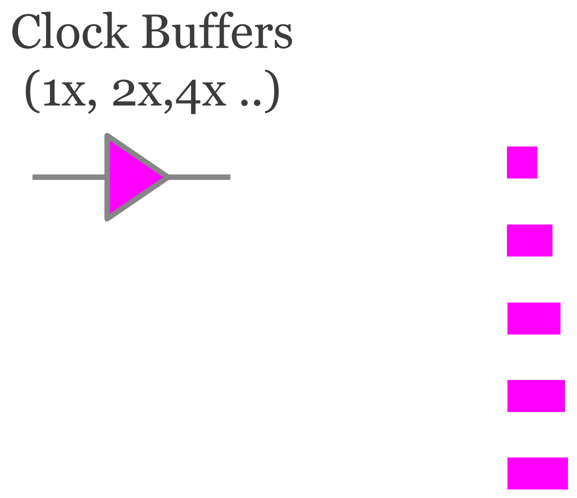
PHYSICAL DESIGN: CTS

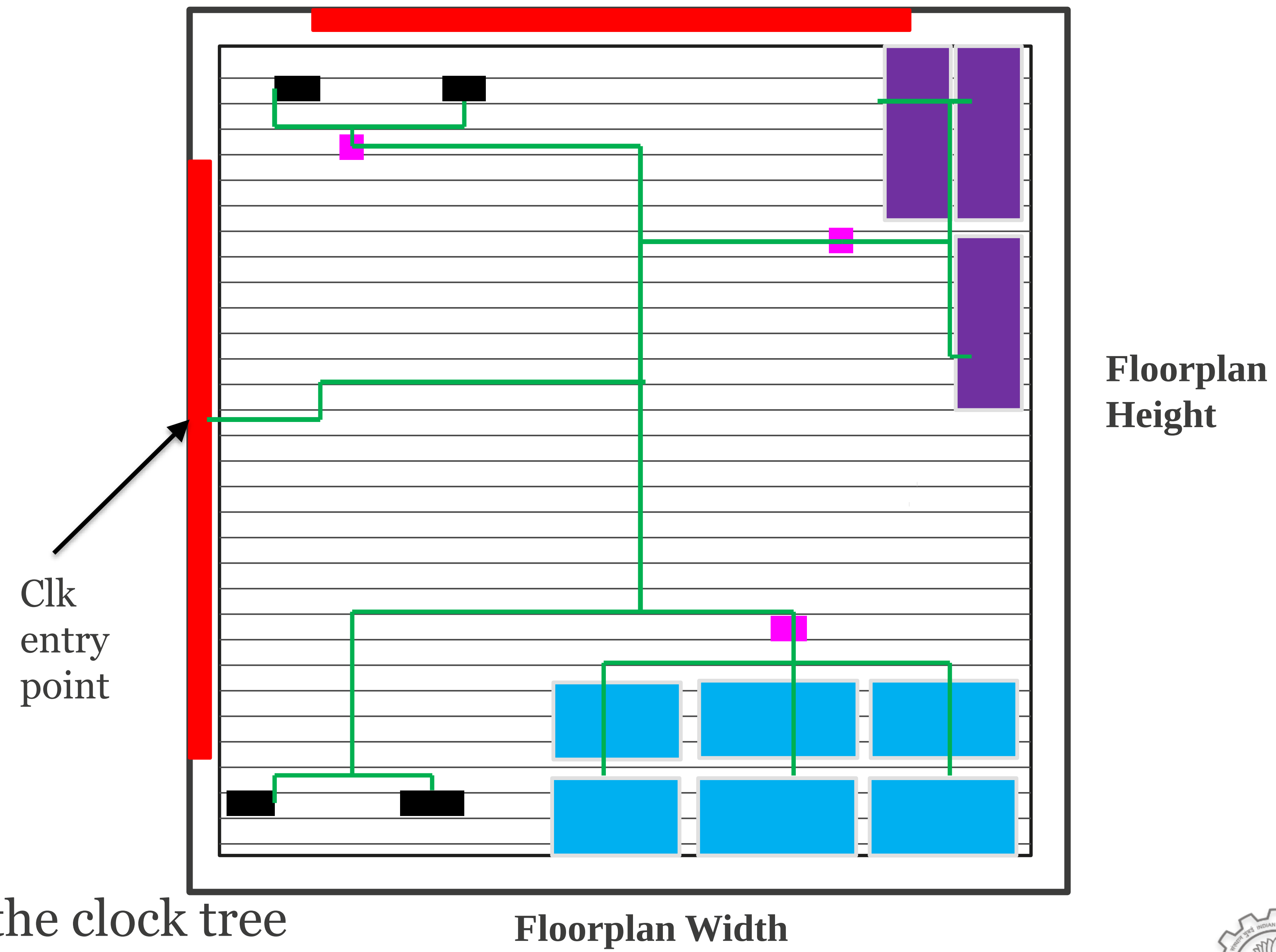




- CTS Involves two steps:
  - Create clock tree
  - Insert clock buffers in the clock tree

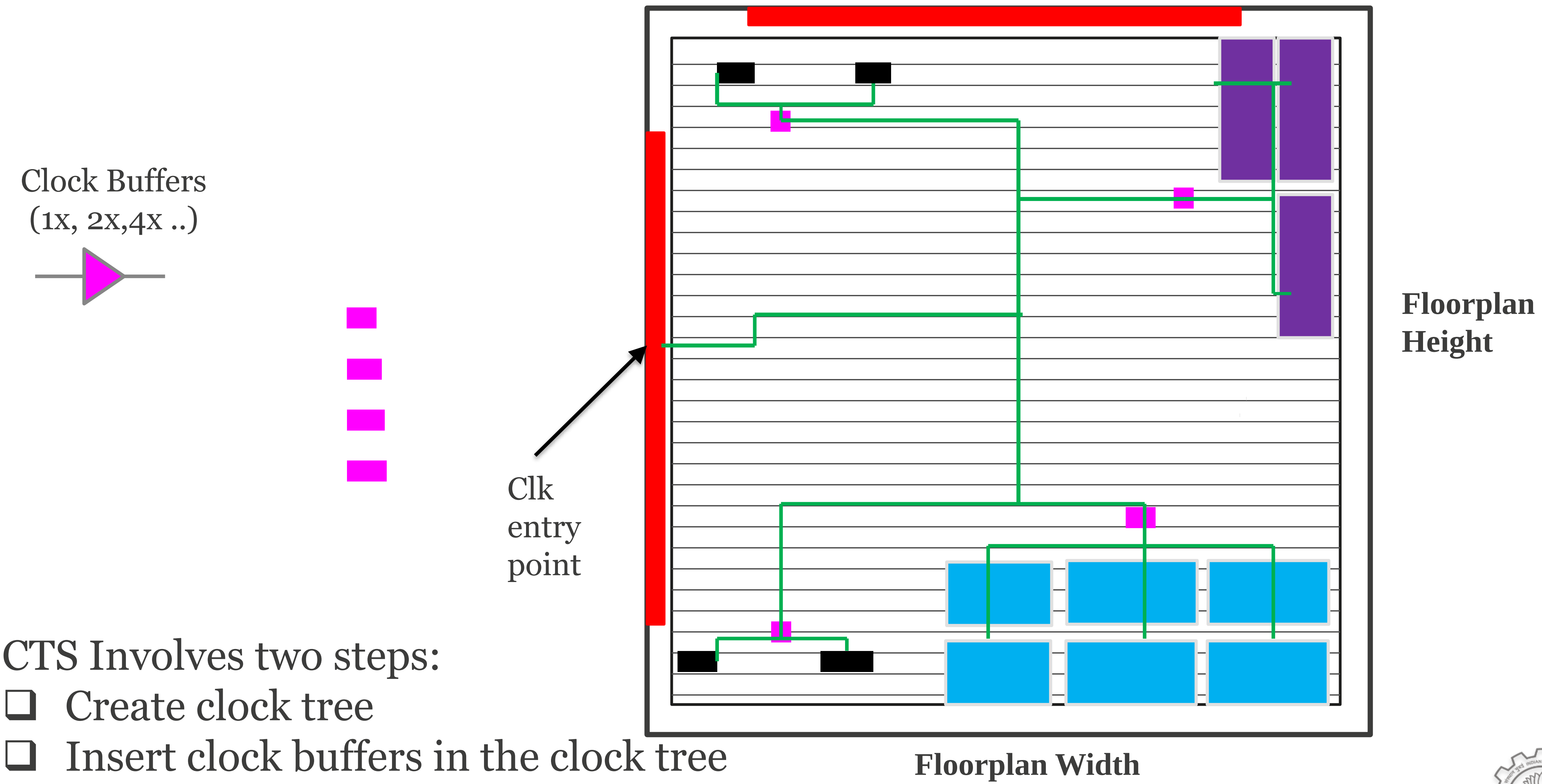
Physical Design: CTS





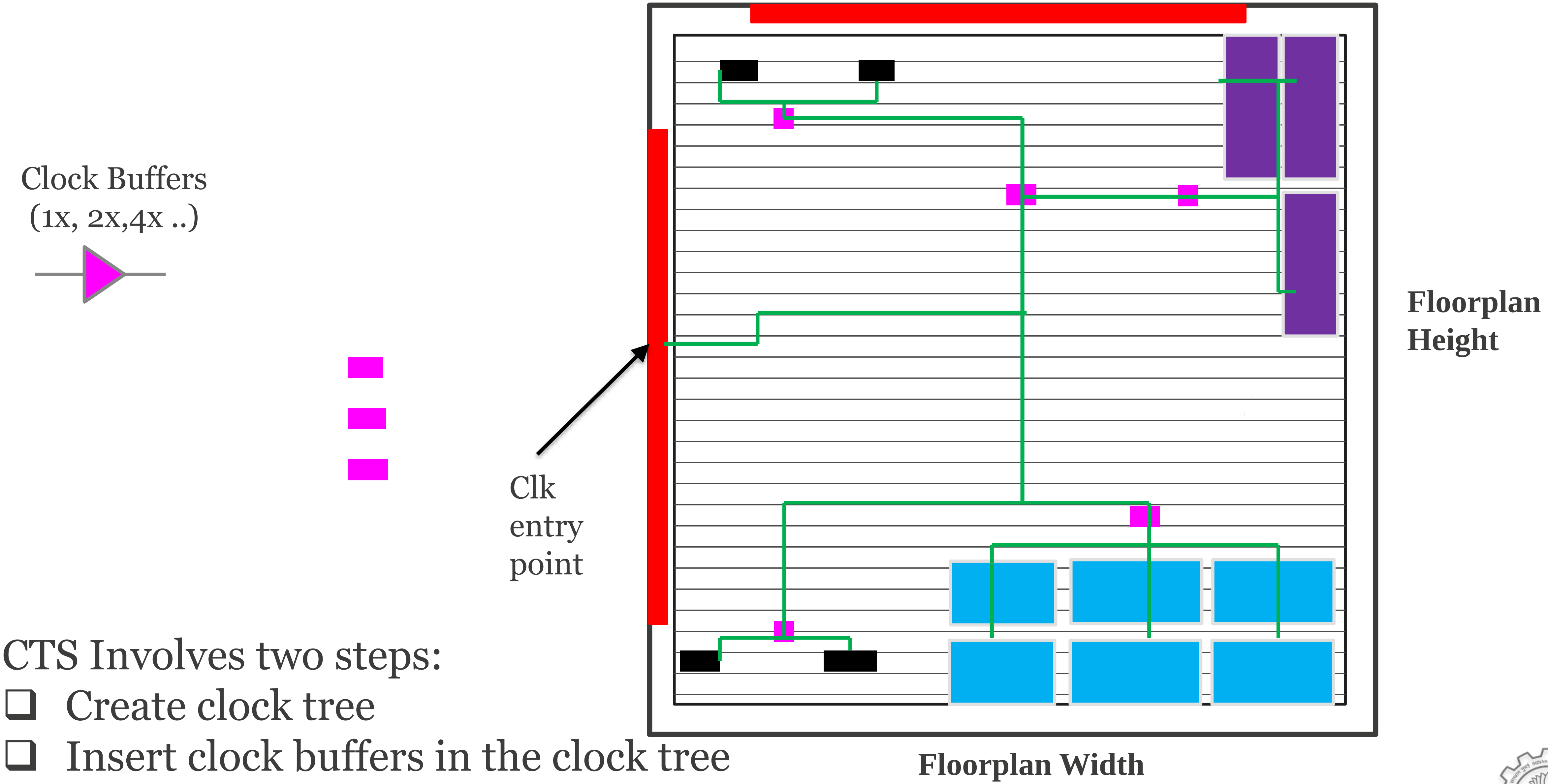
- ❑ CTS Involves two steps:
  - ❑ Create clock tree
  - ❑ Insert clock buffers in the clock tree

Physical Design: CTS



- ❑ CTS Involves two steps:
  - ❑ Create clock tree
  - ❑ Insert clock buffers in the clock tree

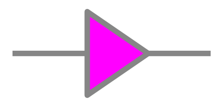
PHYSICAL DESIGN: CTS

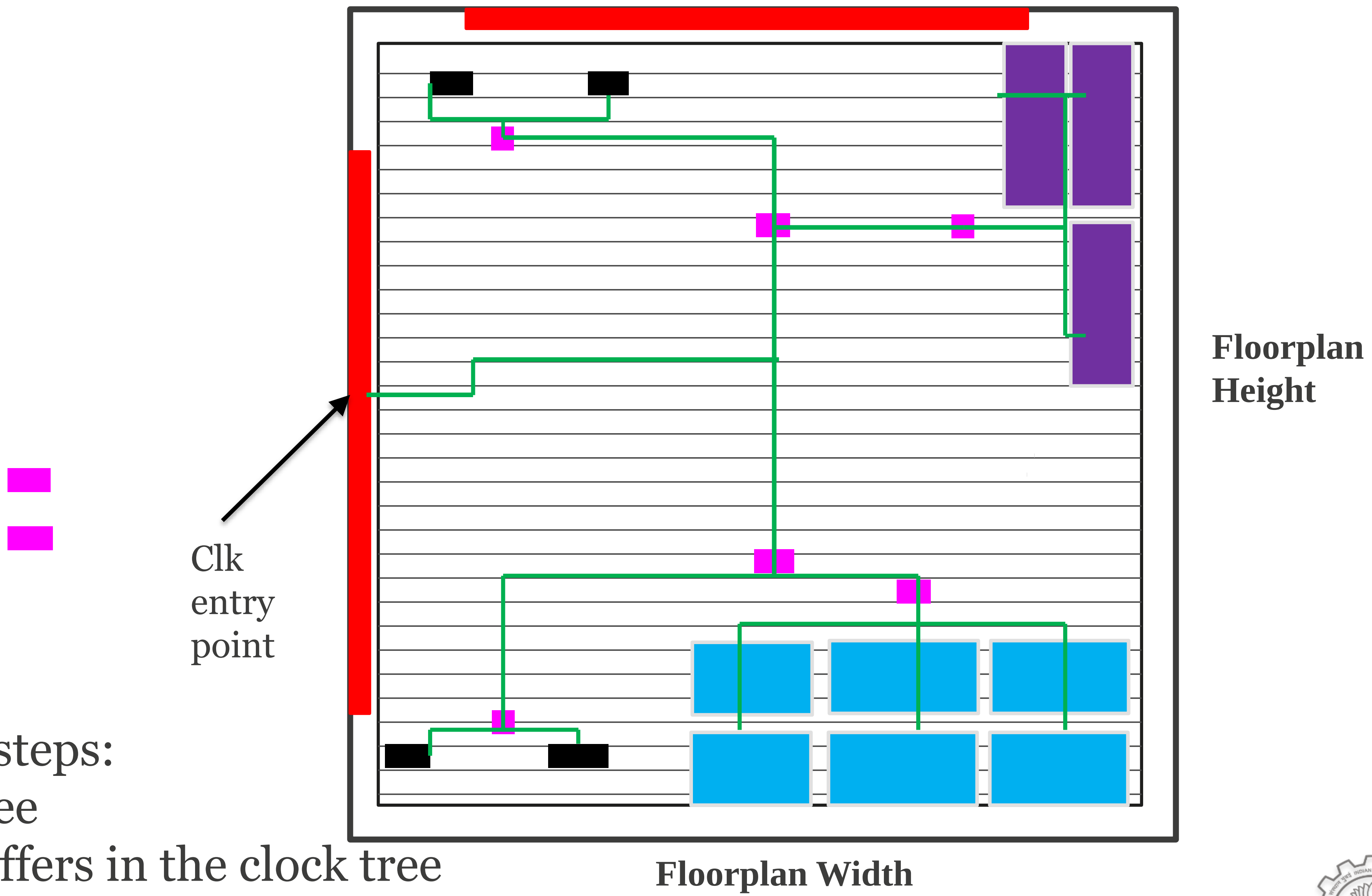


- ❑ CTS Involves two steps:
  - ❑ Create clock tree
  - ❑ Insert clock buffers in the clock tree

Physical Design: CTS

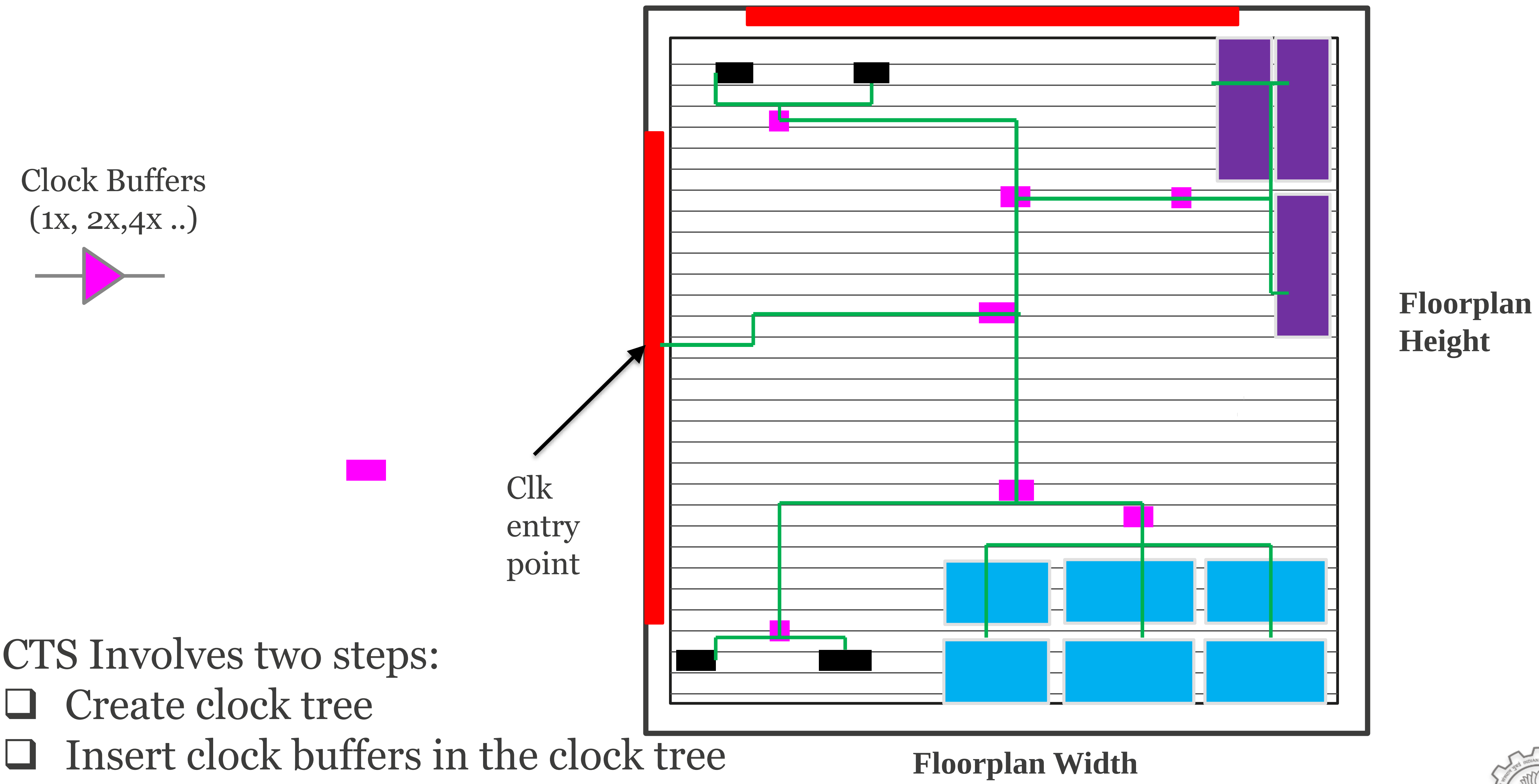
Clock Buffers
(1x, 2x,4x ..)





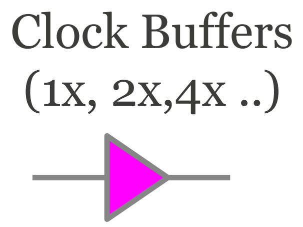
- ❑ CTS Involves two steps:
  - ❑ Create clock tree
  - ❑ Insert clock buffers in the clock tree

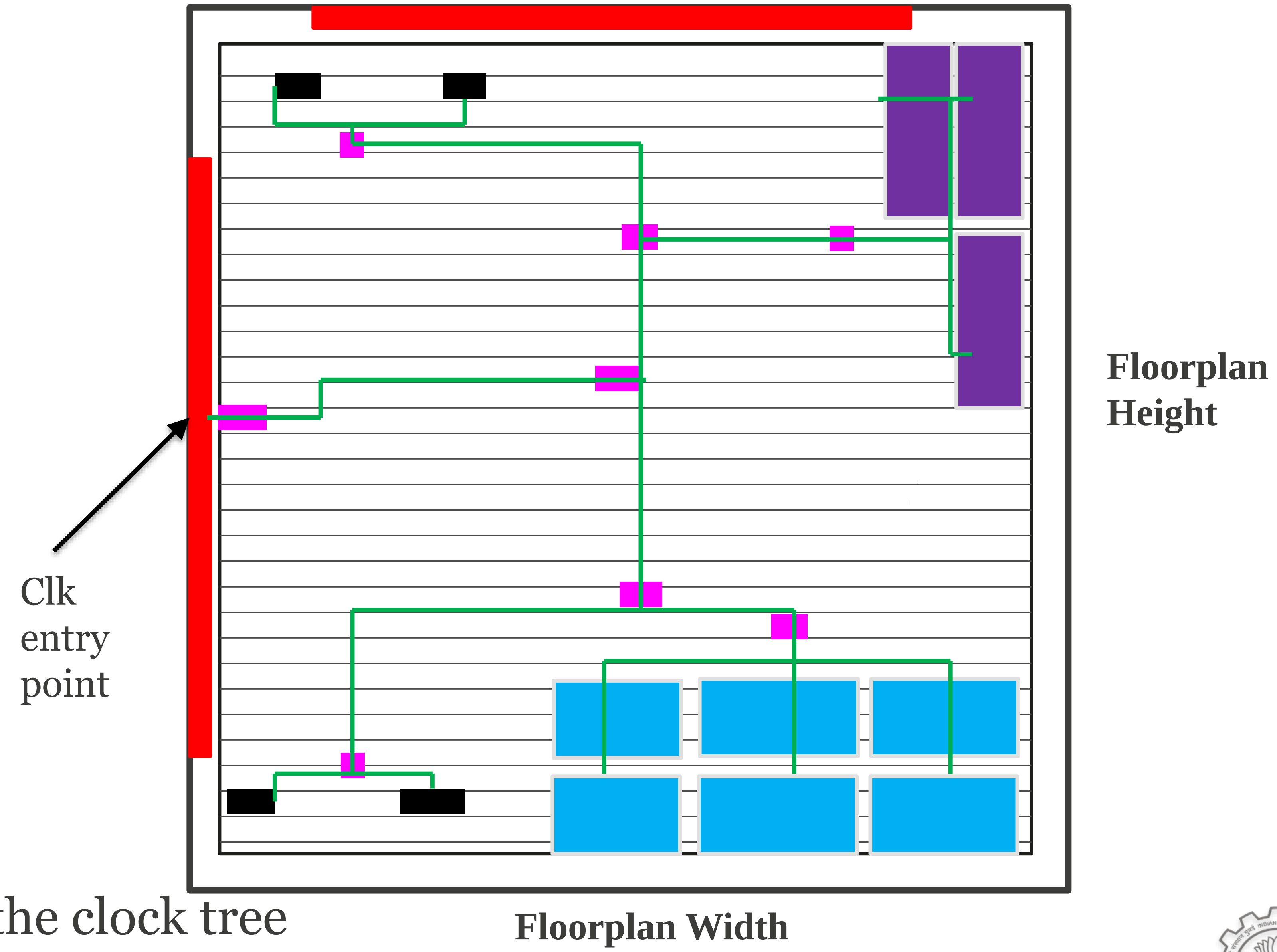
PHYSICAL DESIGN: CTS



- ❑ CTS Involves two steps:
  - ❑ Create clock tree
  - ❑ Insert clock buffers in the clock tree

Physical Design: CTS

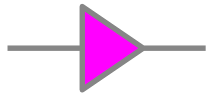


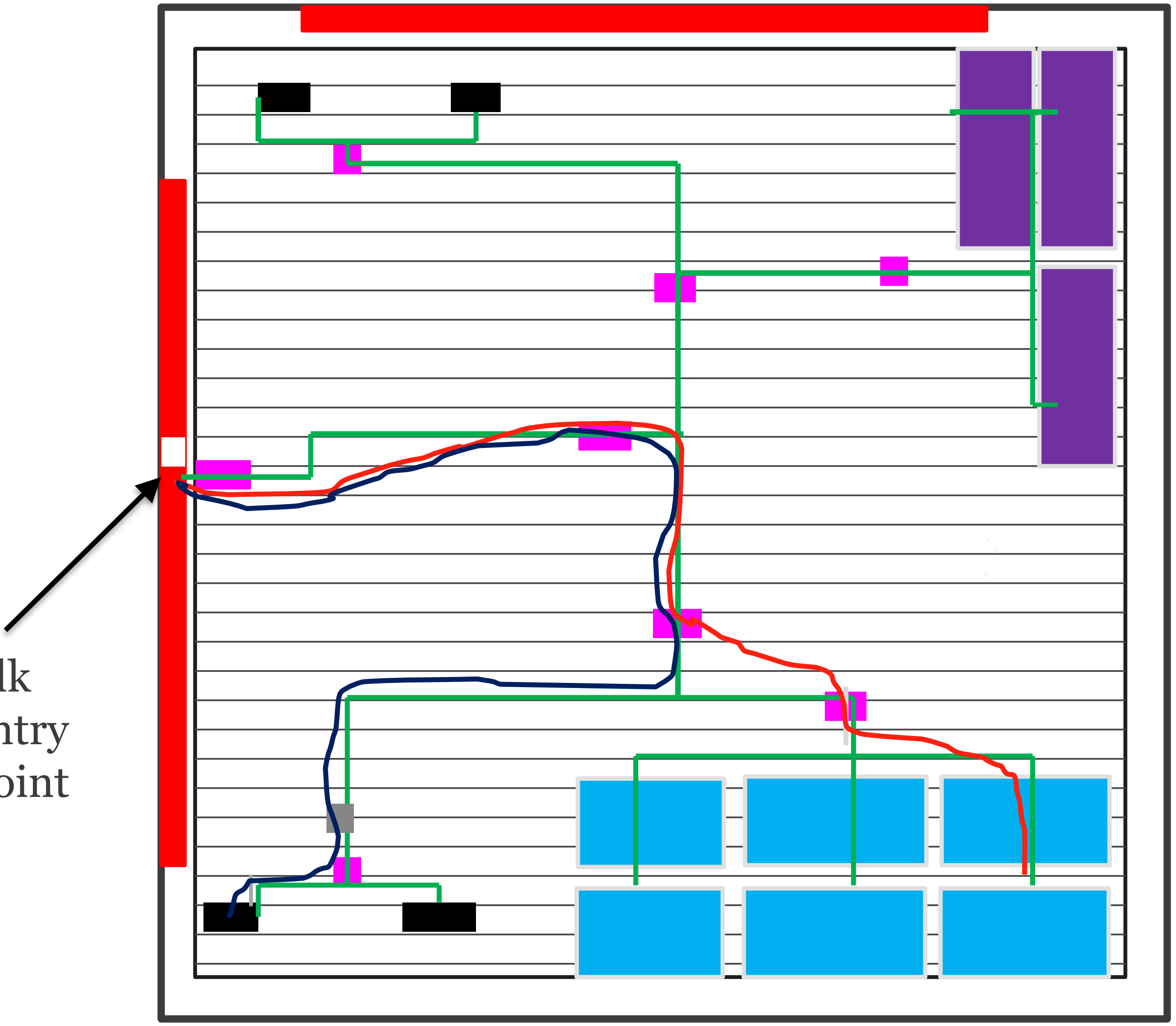


- ❑ CTS Involves two steps:
  - ❑ Create clock tree
  - ❑ Insert clock buffers in the clock tree

PHYSICAL DESIGN: CTS

Clock Buffers
(1x, 2x,4x ..)





Insertion Delay
Time taken by clock tree to transfer the clock signal to the sink

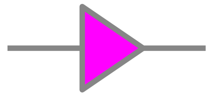
Skew

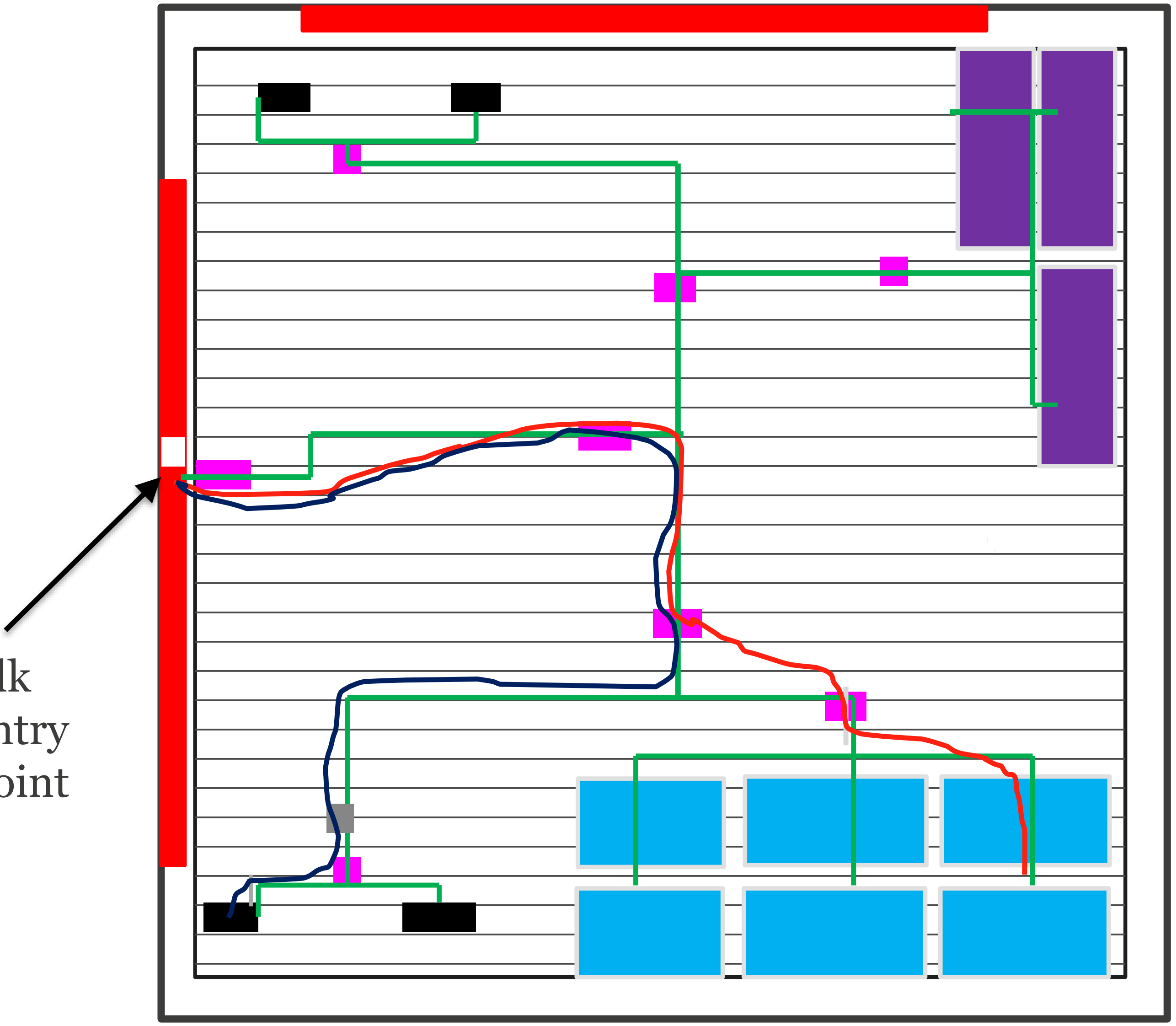
Difference between the insertion delays

Clock tree balancing
Achieve Identical insertion delay for all sinks
i.e. zero skew

Physical Design: CTS

Clock Buffers
(1x, 2x,4x ..)





Insertion Delay
Time taken by clock tree to transfer the clock signal to the sink

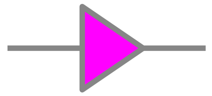
Skew

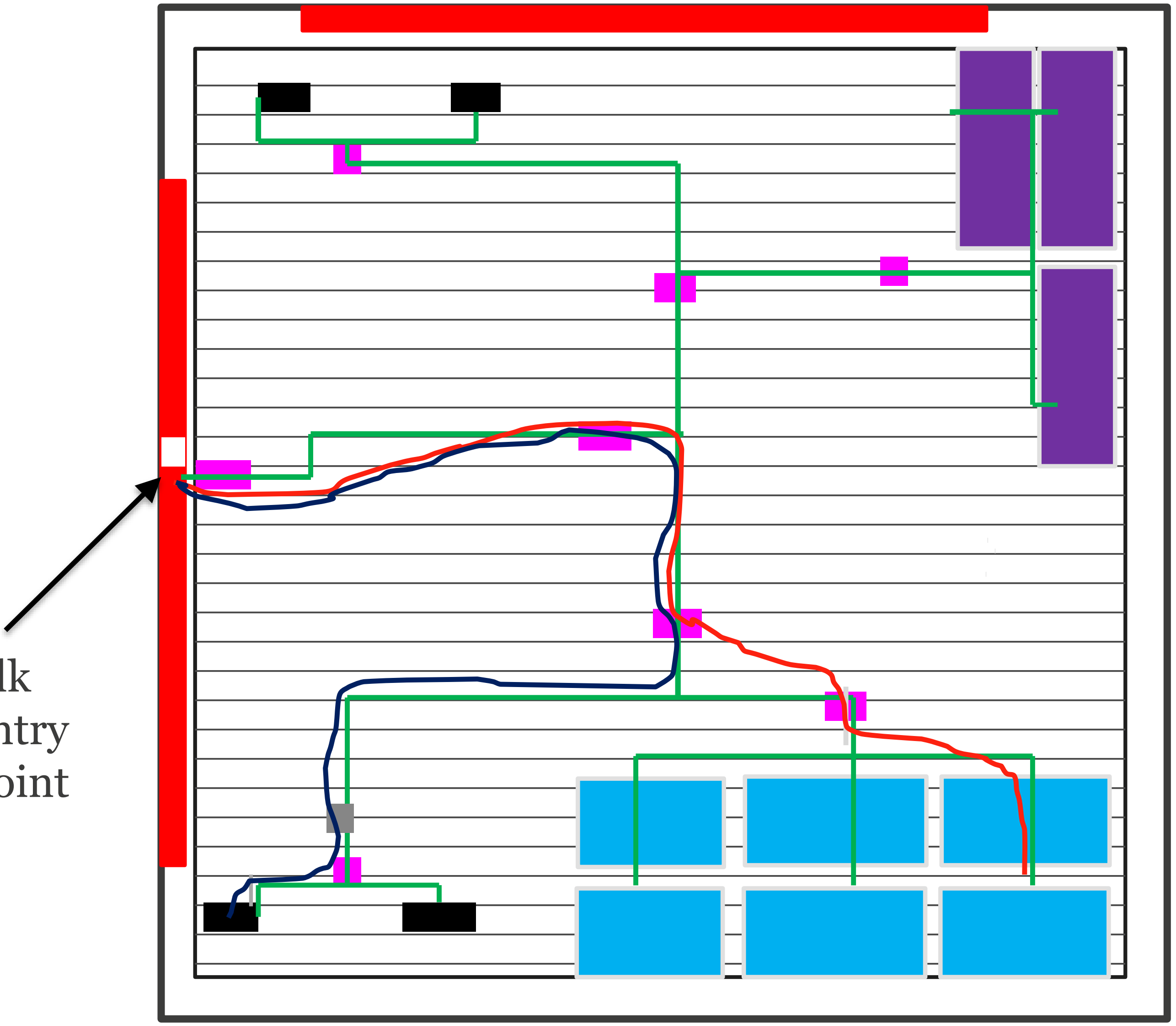
Difference between the insertion delays

Clock tree balancing
Achieve Identical insertion delay for all sinks
i.e. zero skew

Physical Design: CTS

Clock Buffers (1x, 2x,4x ..)





Insertion Delay
Time taken by clock tree to transfer the clock signal to the sink

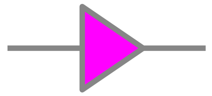
Skew

Difference between the insertion delays

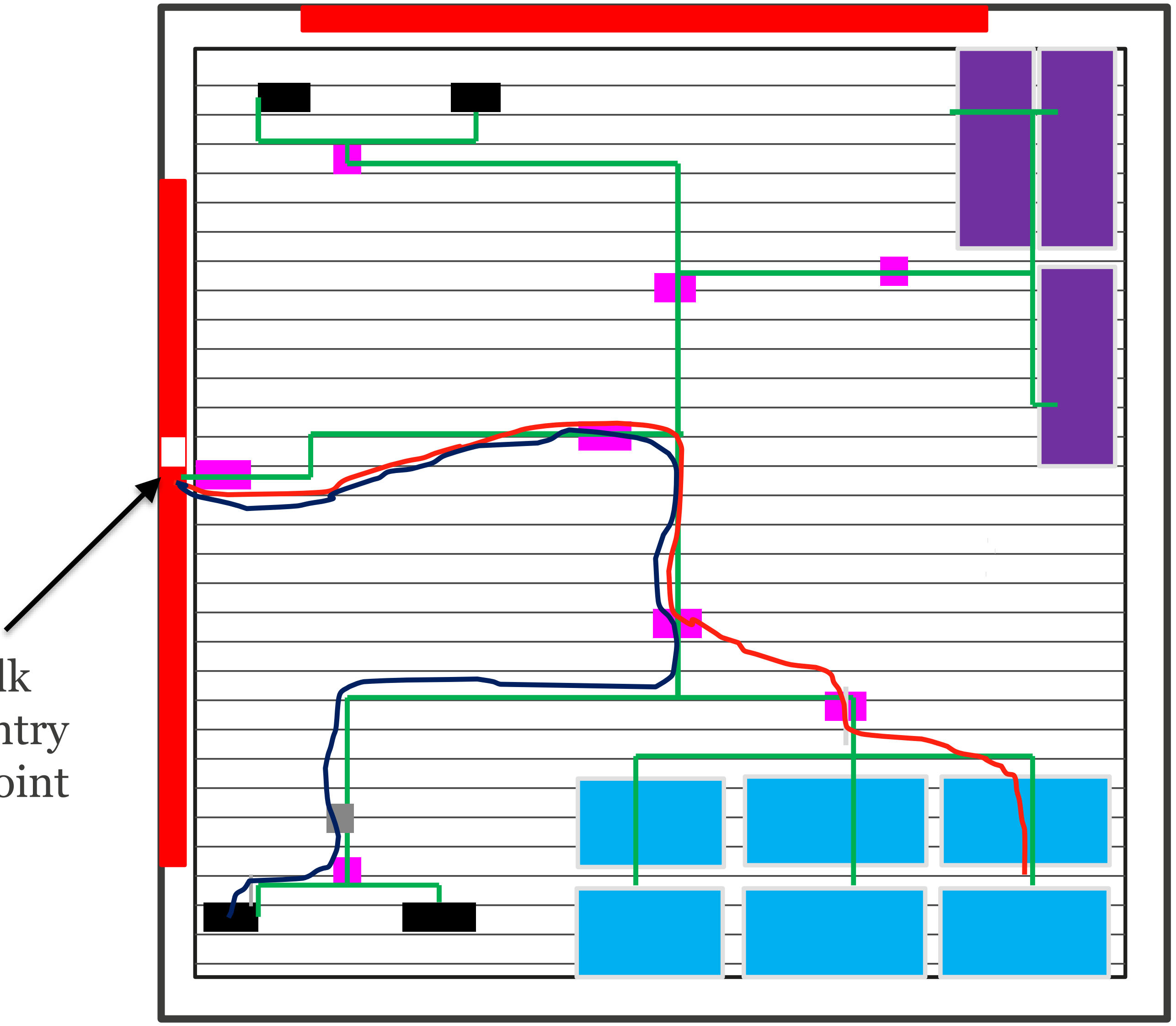
Clock tree balancing
Achieve Identical insertion delay for all sinks
i.e. zero skew

Physical Design: CTS

Clock Buffers
(1x, 2x,4x ..)



Clk
entry
point



Insertion Delay
Time taken by clock tree to transfer the clock signal to the sink

Skew

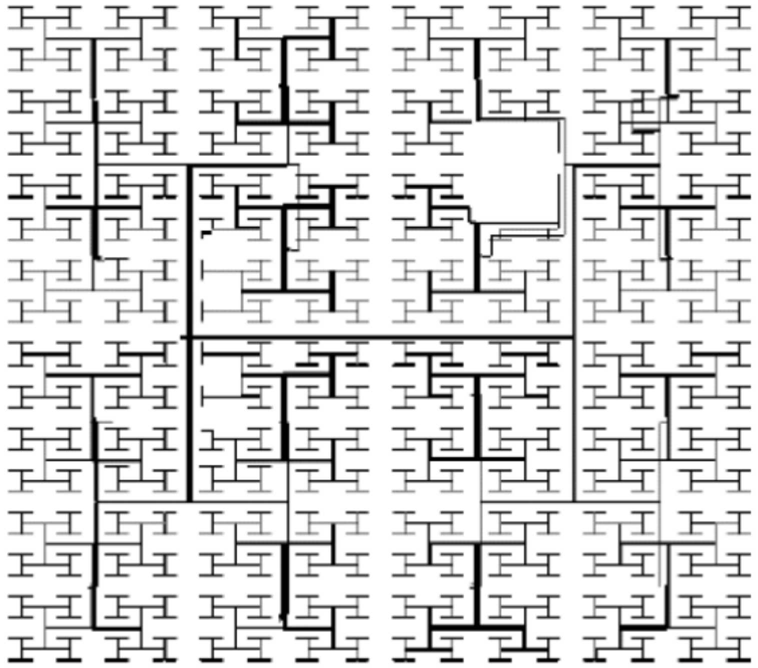
Difference between the insertion delays

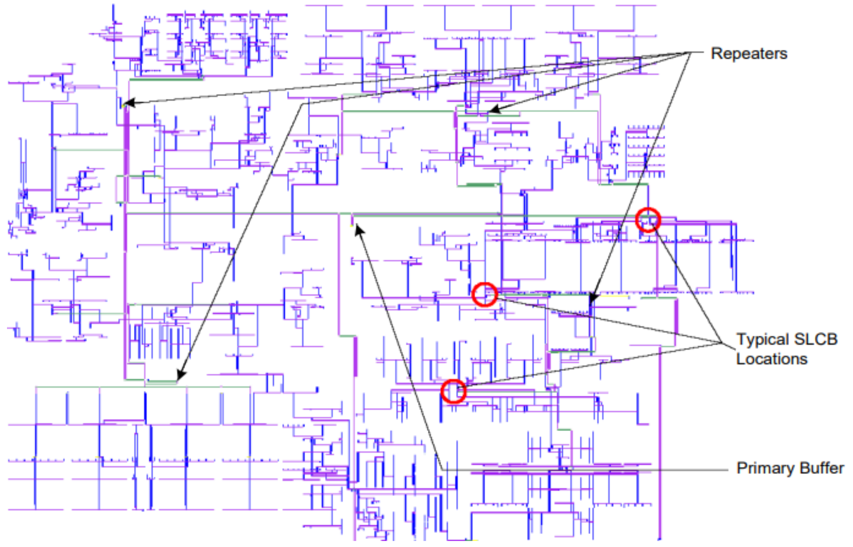
Clock tree balancing
Achieve Identical insertion delay for all sinks
i.e. zero skew

Physical Design: CTS

- ❑ Note that the dynamic power is $\alpha\, CV^2 f$. For clock, α = 1.
  - ❑ Insertion of clock tree → additional dynamic power!

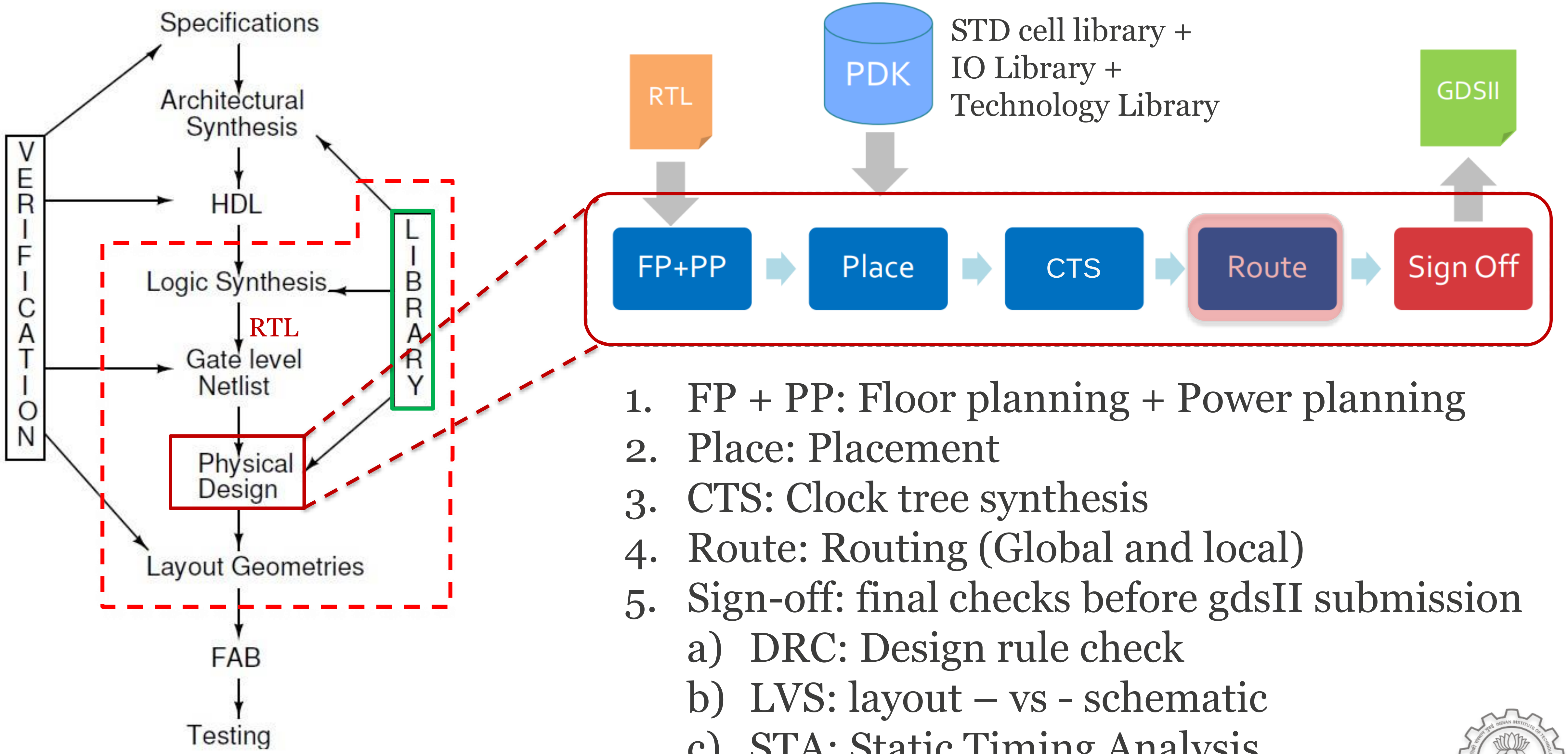
- ❑ In order to reduce skew → we are burning area and power and insertion delay!
- ❑ Modern tools do better → Clock concurrent optimization (CCopt) → advanced topic


IBM power PC 2002: H tree



Intel Itanium 2005: H tree

PHYSICAL DESIGN



1. FP + PP: Floor planning + Power planning
2. Place: Placement
3. CTS: Clock tree synthesis
4. Route: Routing (Global and local)
5. Sign-off: final checks before gdsII submission
   a) DRC: Design rule check
   b) LVS: layout – vs - schematic
   c) STA: Static Timing Analysis

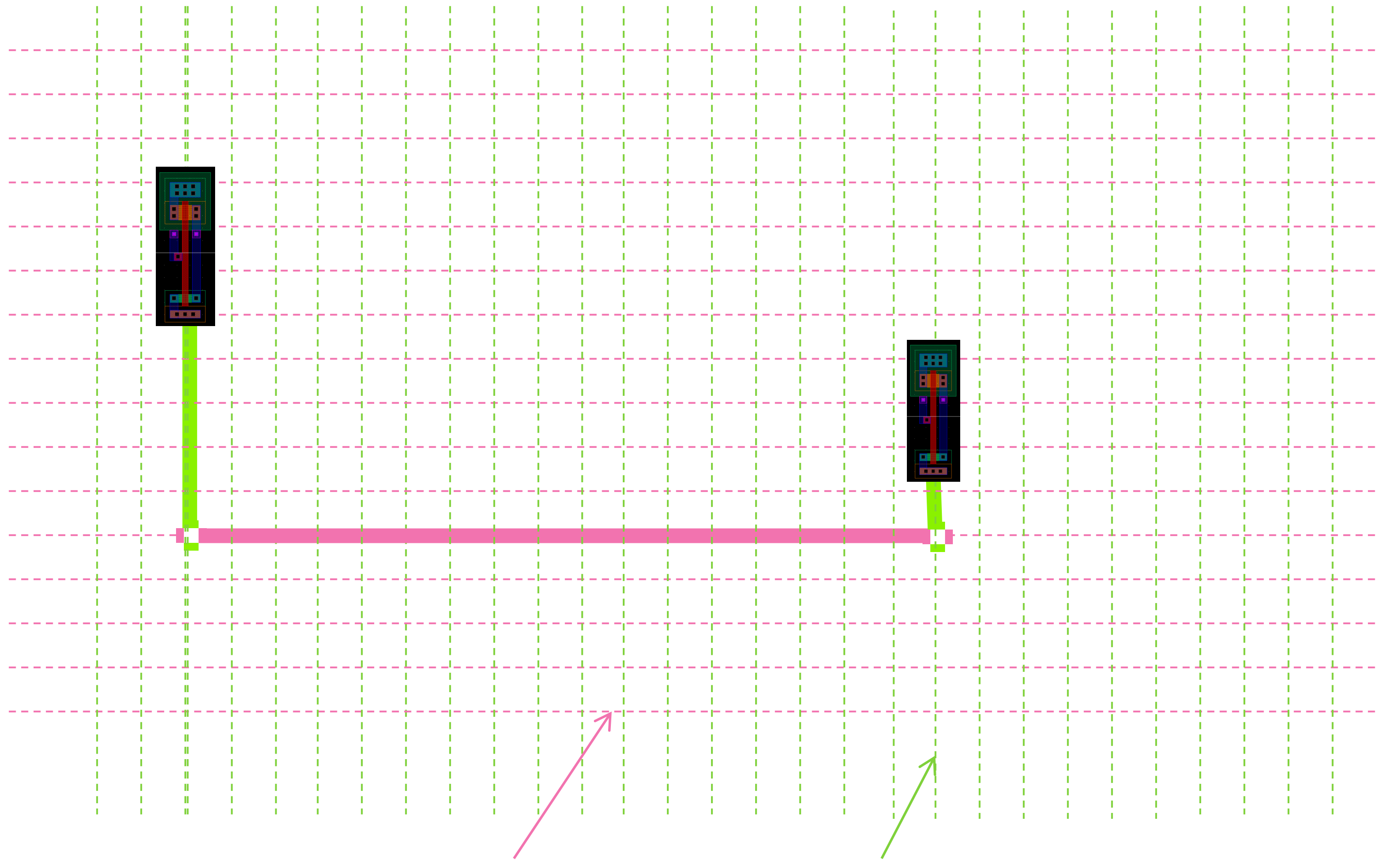
Physical Design: Routing

- ❑ Netlist connections (RTL) must be realized through routing
- ❑ Current chips can have up to 10 layers of metal routing (Sky130 has 6 routing layers)
- ❑ Routing of a metal is done only in one direction (H or V)
  - ❑ Ex: M1-H, M2-V, M3-H, M4-V …
- ❑ Connections between metal layers: VIA
- ❑ To make the job of the router easy, routing is done on specified grids (grids are defined in process technology file)
- ❑ For each metal layer:
  - ❑ Minimum width
  - ❑ Minimum spacing

Physical Design: Routing

- Example:

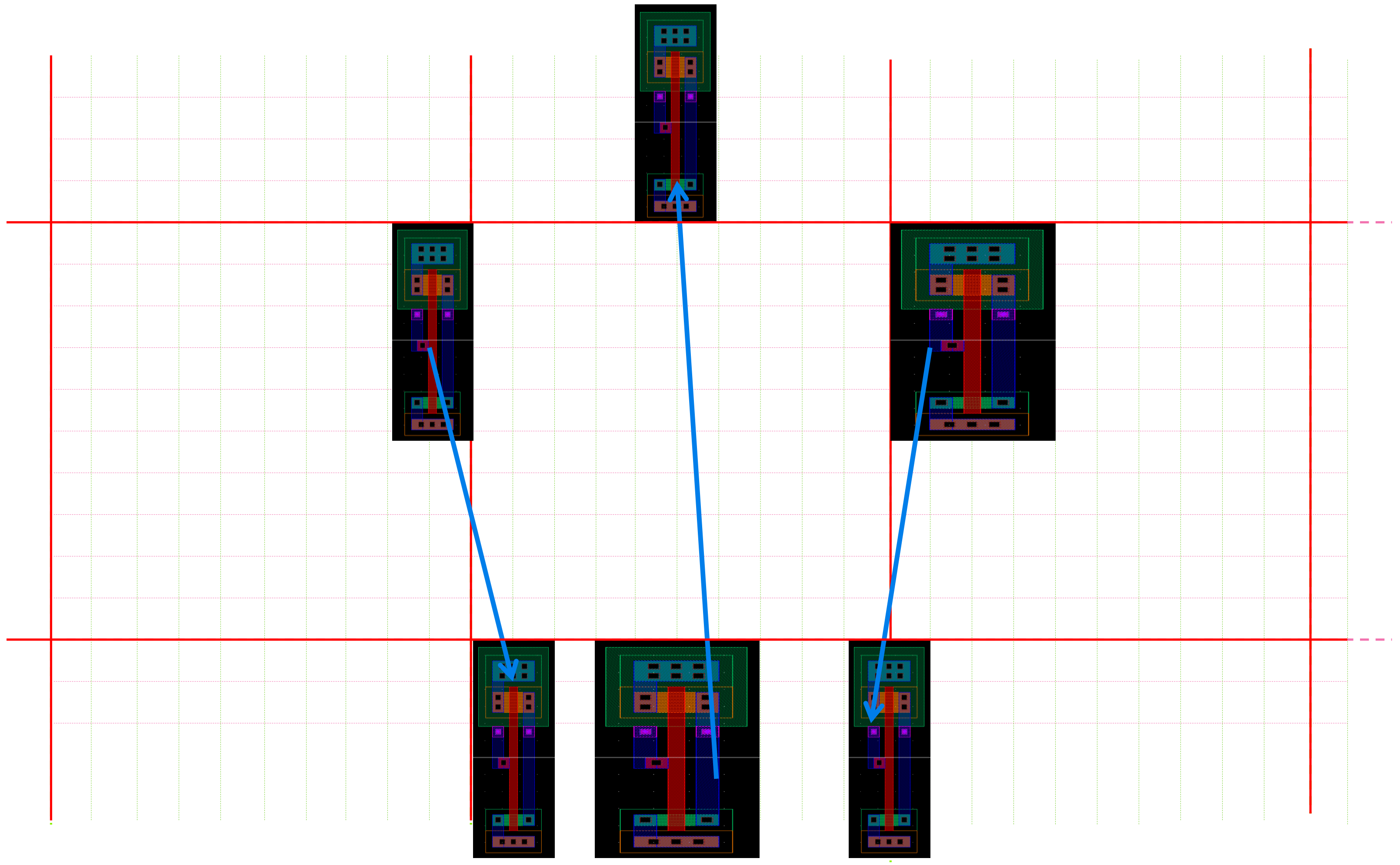
Metals, VIAs & tracks



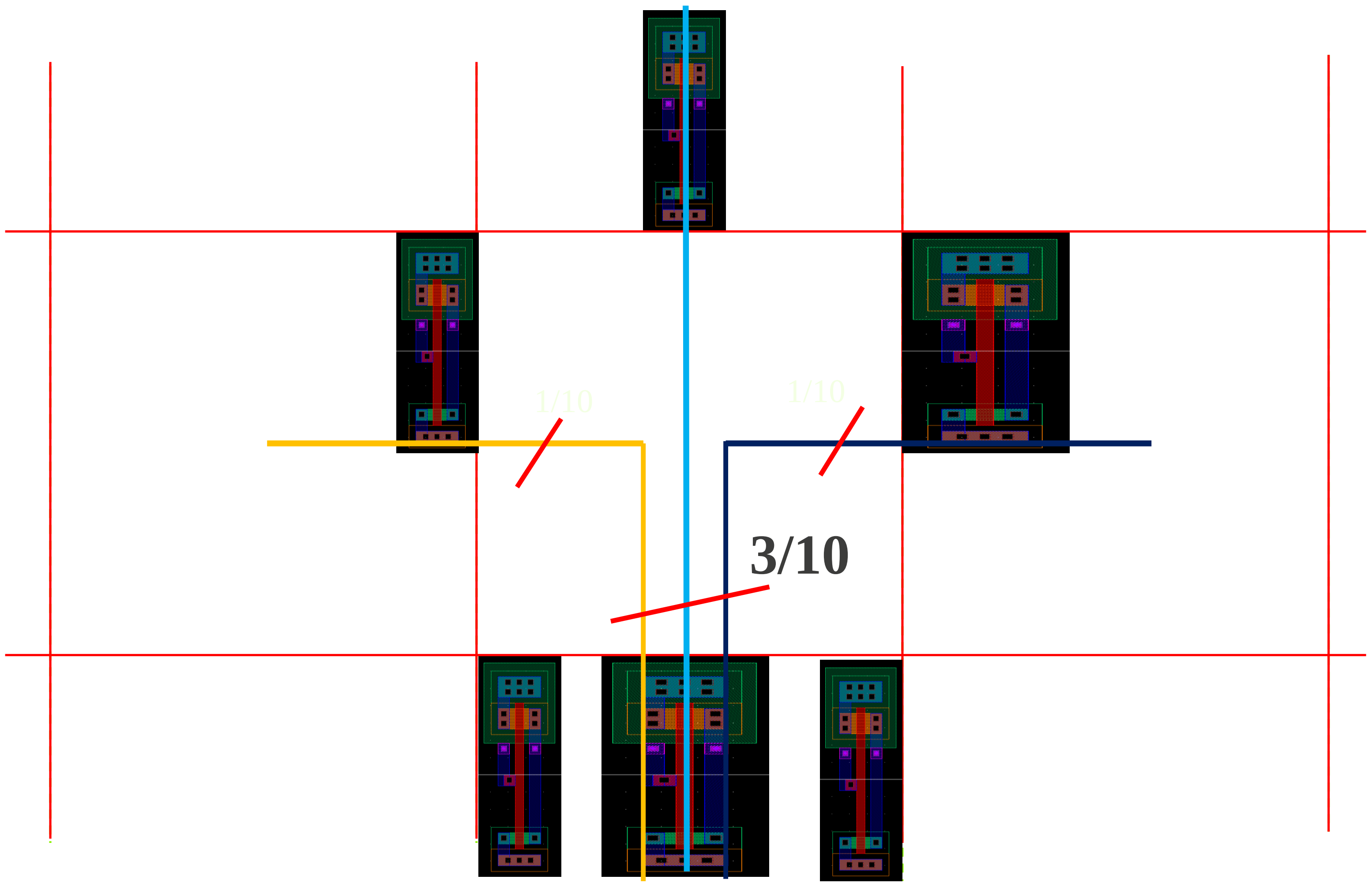
Physical Design: Routing

- Routing: millions of nets
- Infinite possibilities to route: tedious job for the tool
- To simplify, split into three steps:
  - Global route
    - Center to center route of std cells.
    - Find short route for timing critical nets.
    - Resolve routing congestion.
    - Predict crosstalk and correct.
  - Track route
    - Assign tracks to global routes
    - Crosstalk free track assignment.
    - More accurate than global route
  - Detail route
    - Most refined routing including VIAS.
    - DRC clean

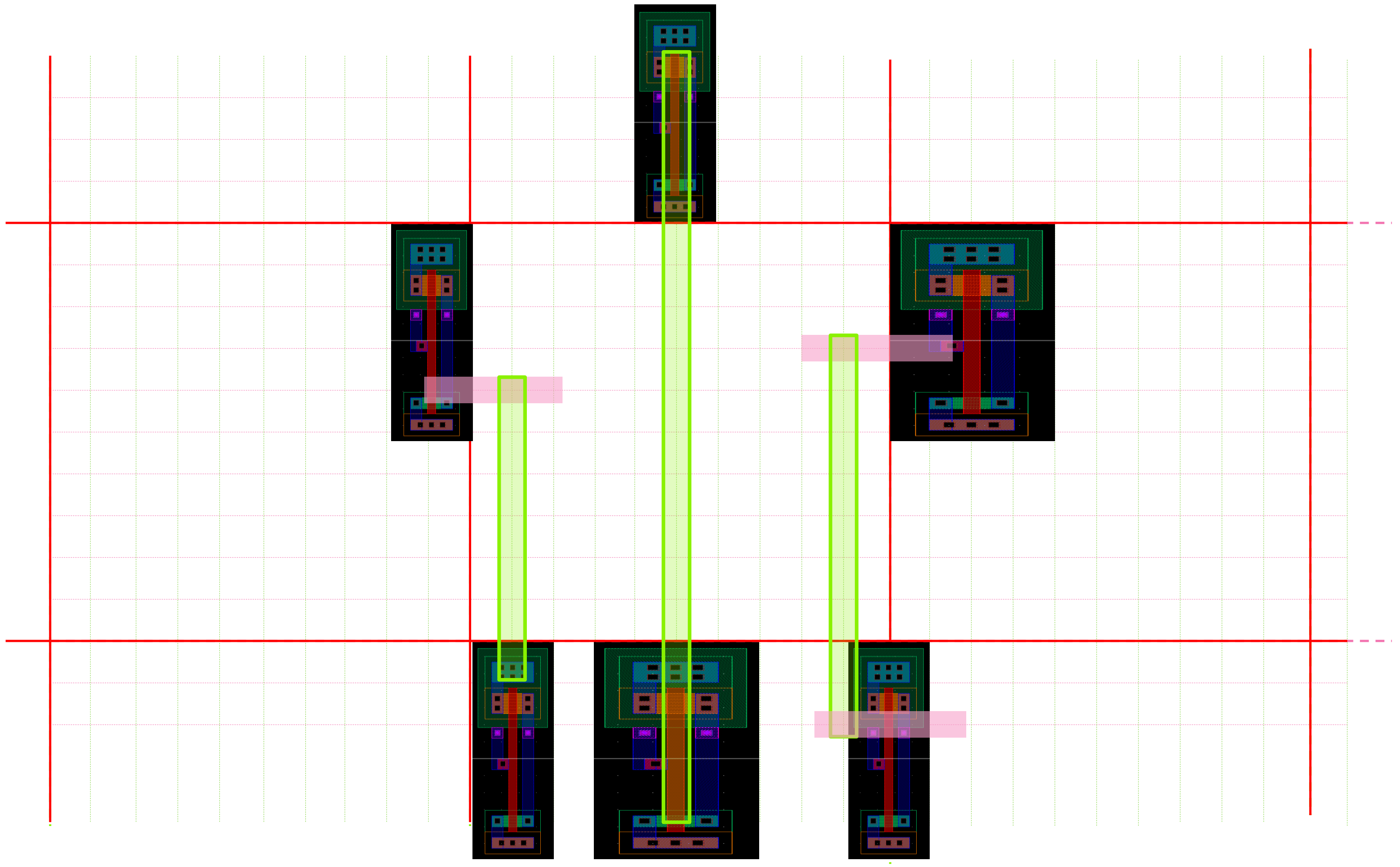
Physical Design: Routing (Global)



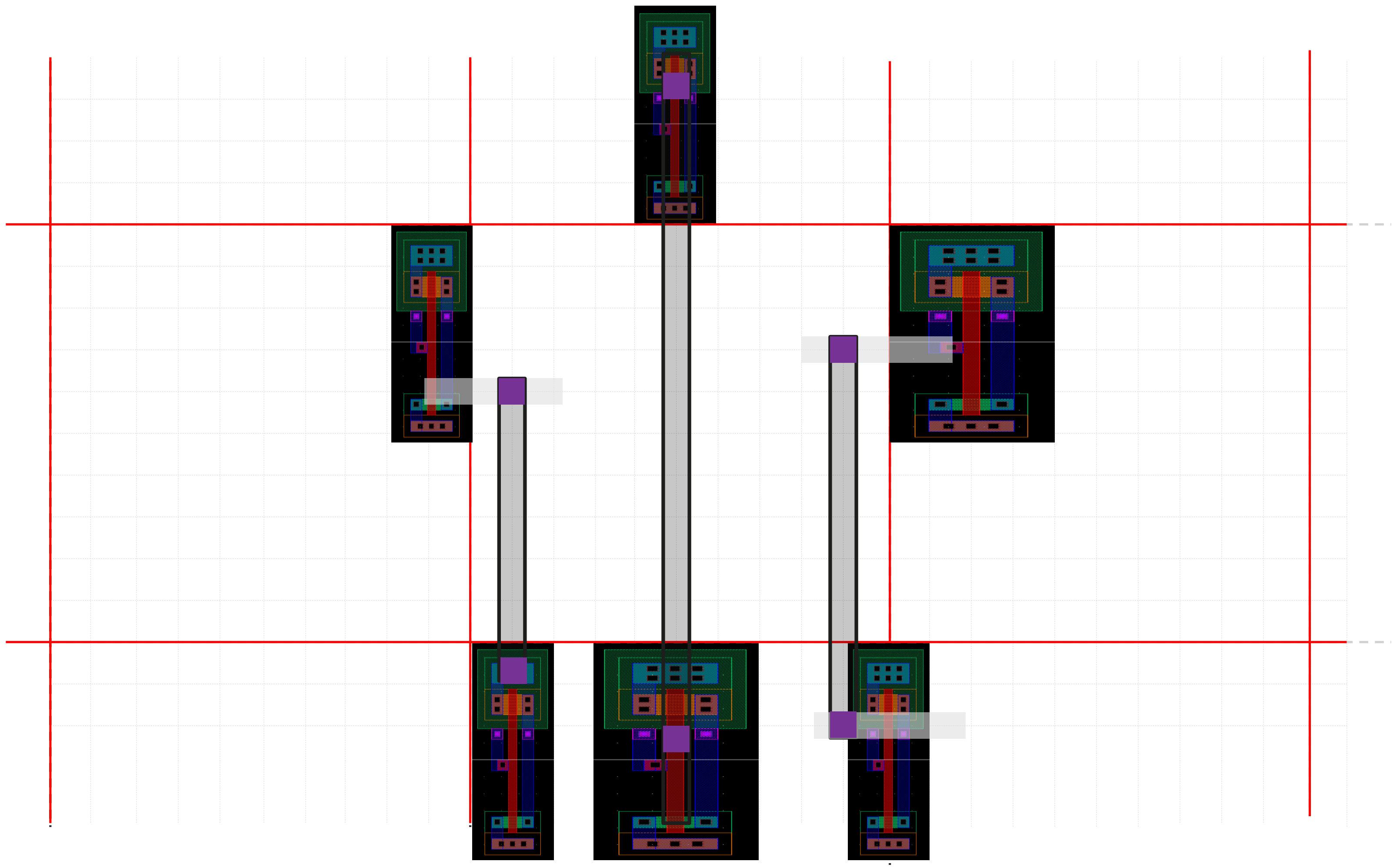
Physical Design: Routing (Global)



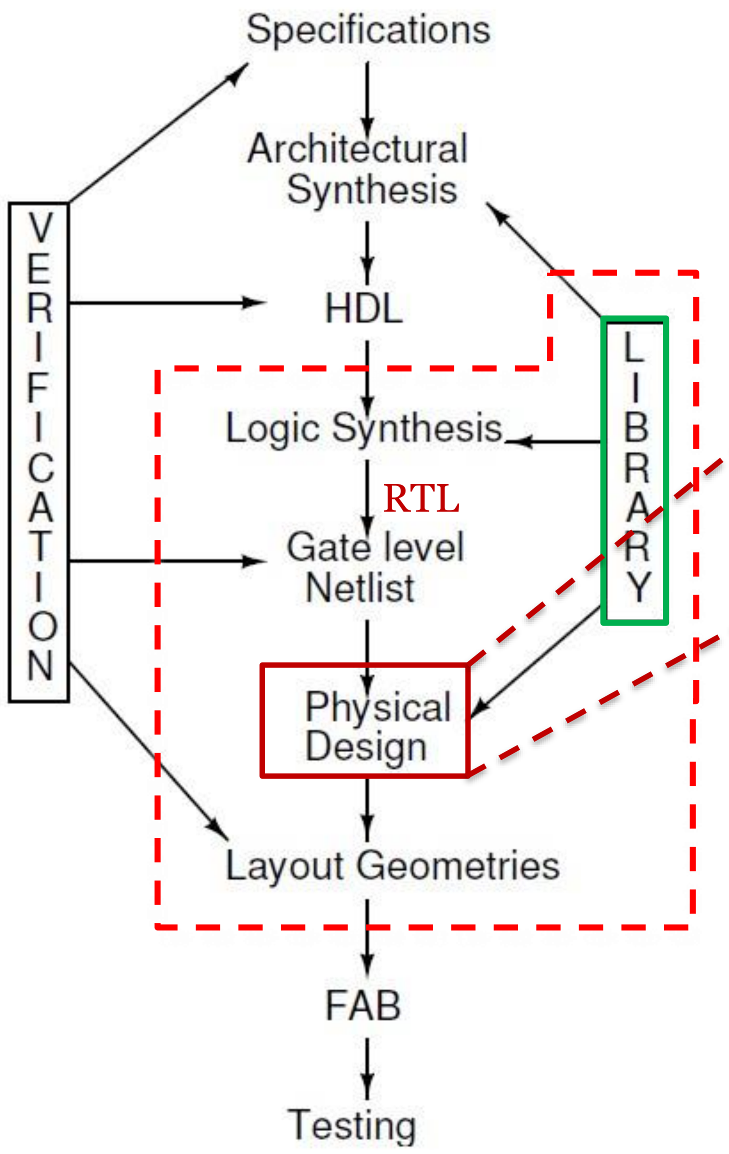
PHYSICAL DESIGN: ROUTING (TRACK)

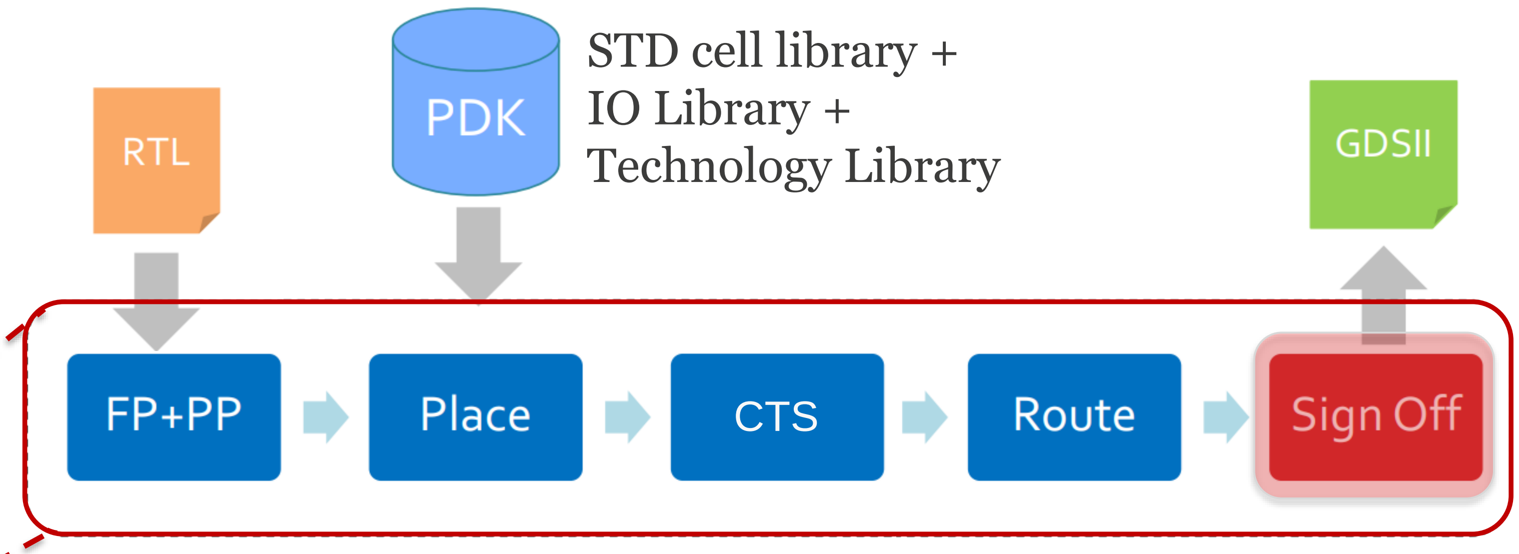


Physical Design: Routing (Detailed)



Physical Design





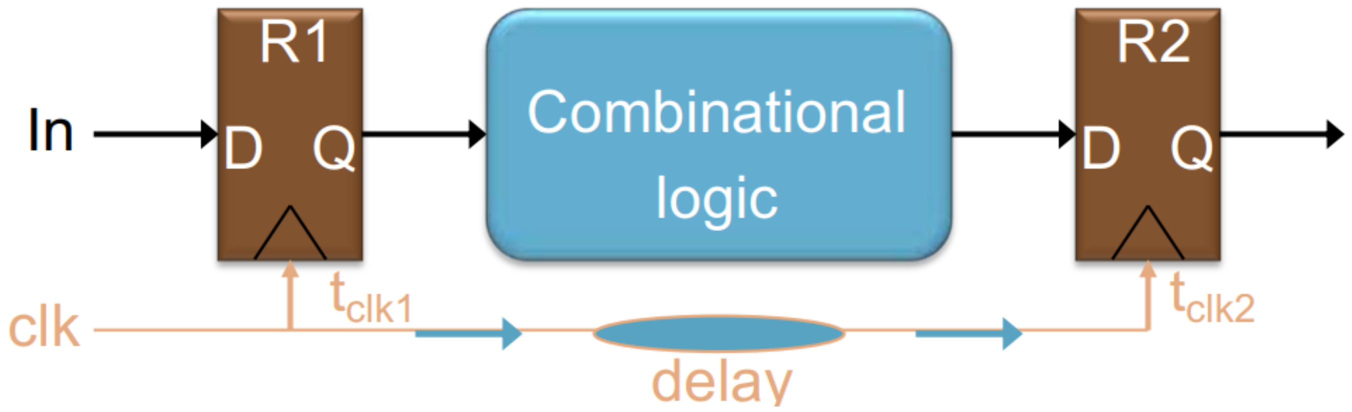
1. FP + PP: Floor planning + Power planning
2. Place: Placement
3. CTS: Clock tree synthesis
4. Route: Routing (Global and local)
5. Sign-off: final checks before gdsII submission
   a) DRC: Design rule check
   b) LVS: layout – vs - schematic
   c) STA: Static Timing Analysis

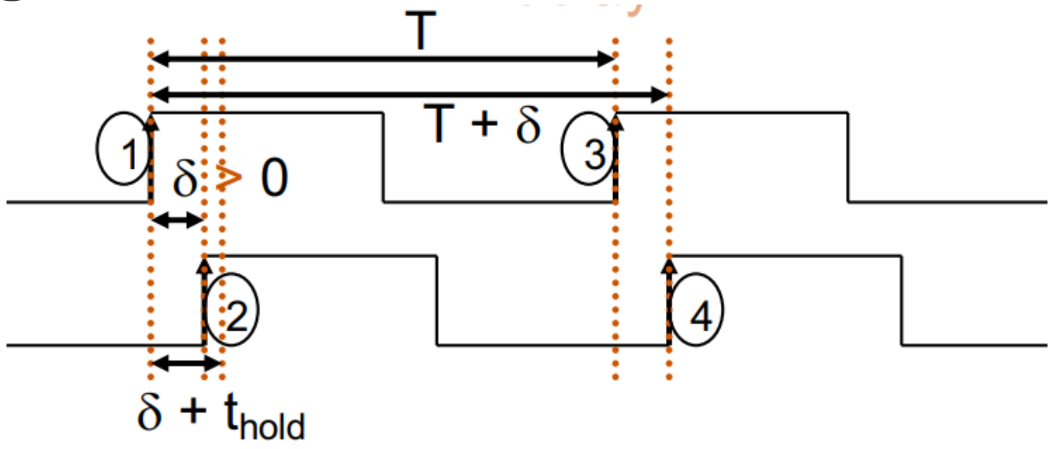
Physical Design: Signoff

- Signoff: physical verifications before gdsII submission to foundry
- Important signoff verifications:
  - STA: Static timing analysis
  - DRC: design rule check of the entire chip
  - LVS: Layout vs schematic of the entire chip
  - Antenna DRC

PHYSICAL DESIGN: STA

- For every path, the timing check has to be performed to check if we meet the timing constraints – not violating setup/hold
- Post physical design, we have added
  - routing – RC delay of interconnect
  - Clock network – clock skew minimization logic





$T + \delta \geq t_{clk\text{-}Q} + t_{p\text{-}logic(max)} + t_s \rightarrow T \geq t_{clk\text{-}Q} + t_{p\text{-}logic(max)} + t_s - \delta$ 😃

$t_h + \delta \leq t_{clk\text{-}Q} + t_{p\text{-}logic(min)} \rightarrow t_h \leq t_{clk\text{-}Q} + t_{p\text{-}logic(min)} - \delta$ ☹

PHYSICAL DESIGN: STA

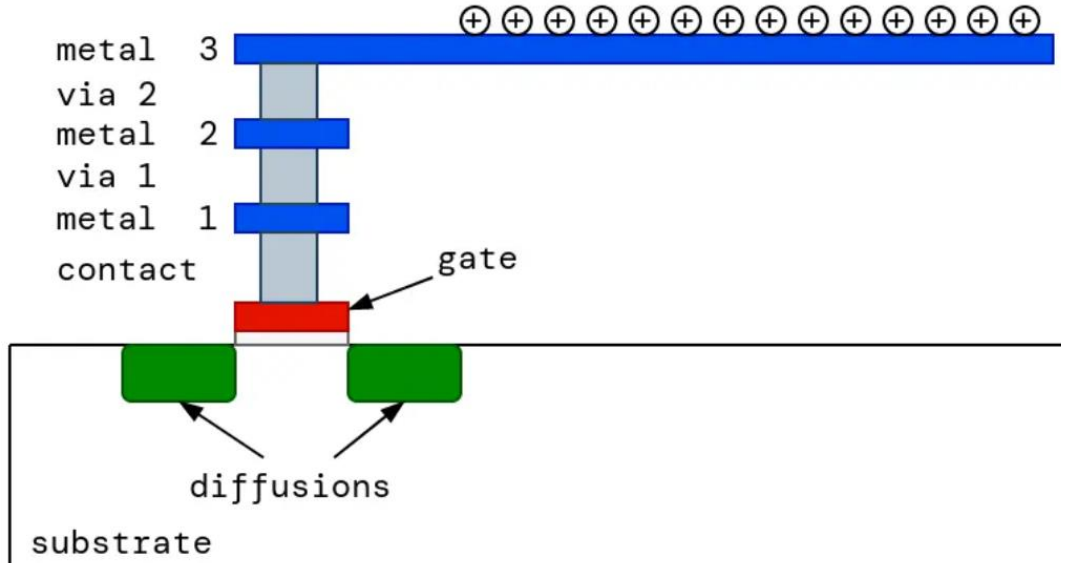
- STA: Analysis to check if a circuit meets the required timing
- Timing check against given user constraint
- Analysis is static in nature and not dynamic
  - Therefore not an actual simulation with clock cycles ! (that would be dynamic)
  - Functionality is not checked!
  - Faster than an actual dynamic simulation

- Design is broken into set of timing paths
- Delay of each path is calculated: all paths must meet the timing constraint
  - Information on cell delay is coming from .lib files
  - Parasitic extraction (PEX) is performed – delay of all interconnects

- At the end of STA, design should not violate setup or hold conditions in any path
- If setup violations are present, can be fixed by reducing clock (if it is okay). If not, try a different placement

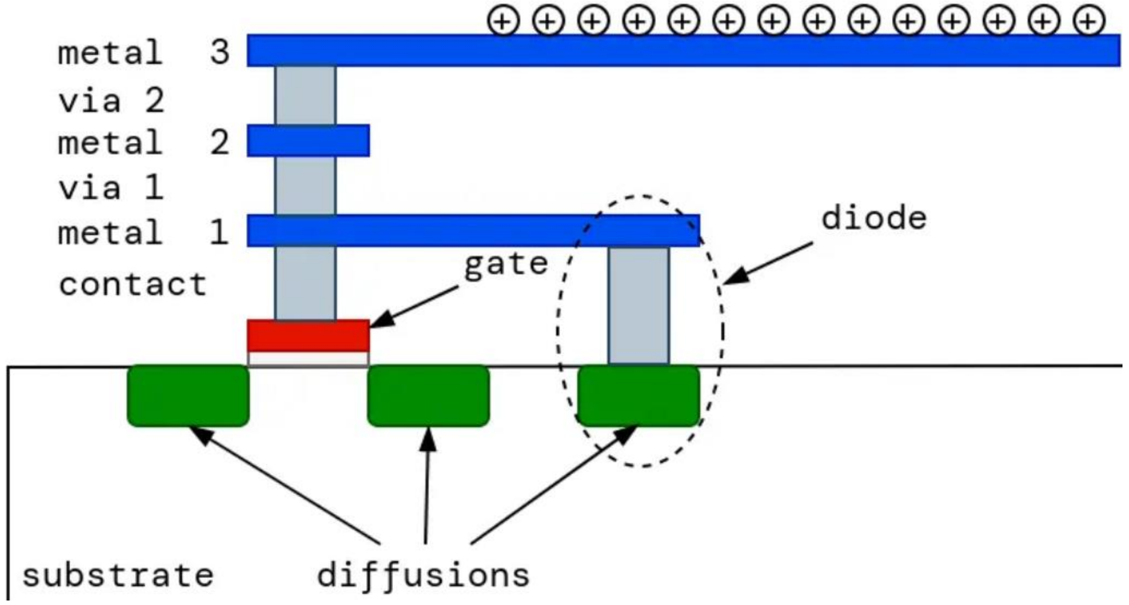
PHYSICAL DESIGN: DRC, LVS

- DRC: checks for process design rules (already did this at the standard cell level – now at the chip level)
- LVS: to ensure the layout and the spice netlist (of the entire design) is exactly the same
- Antenna check:
  - Issue: Long metal wire segments connected to MOS gate → collects charges during fabrication process →may damage the gate oxide during fabrication process → called the antenna effect



PHYSICAL DESIGN: ANTENNA

- Antenna DRC: checks if the a gate connected wire has length greater than a threshold to cause oxide damage (i.e ratio of gate surface area to the metal length)
- Fix:
  - Approach 1: Replace all long metals with short metals and jump to next metal (hopping) – effects the routing congestion
  - Approach 2: Insert reverse biased diodes close to the gate that suffers from antenna – done by the tool if the diode is specified (provided by foundry)

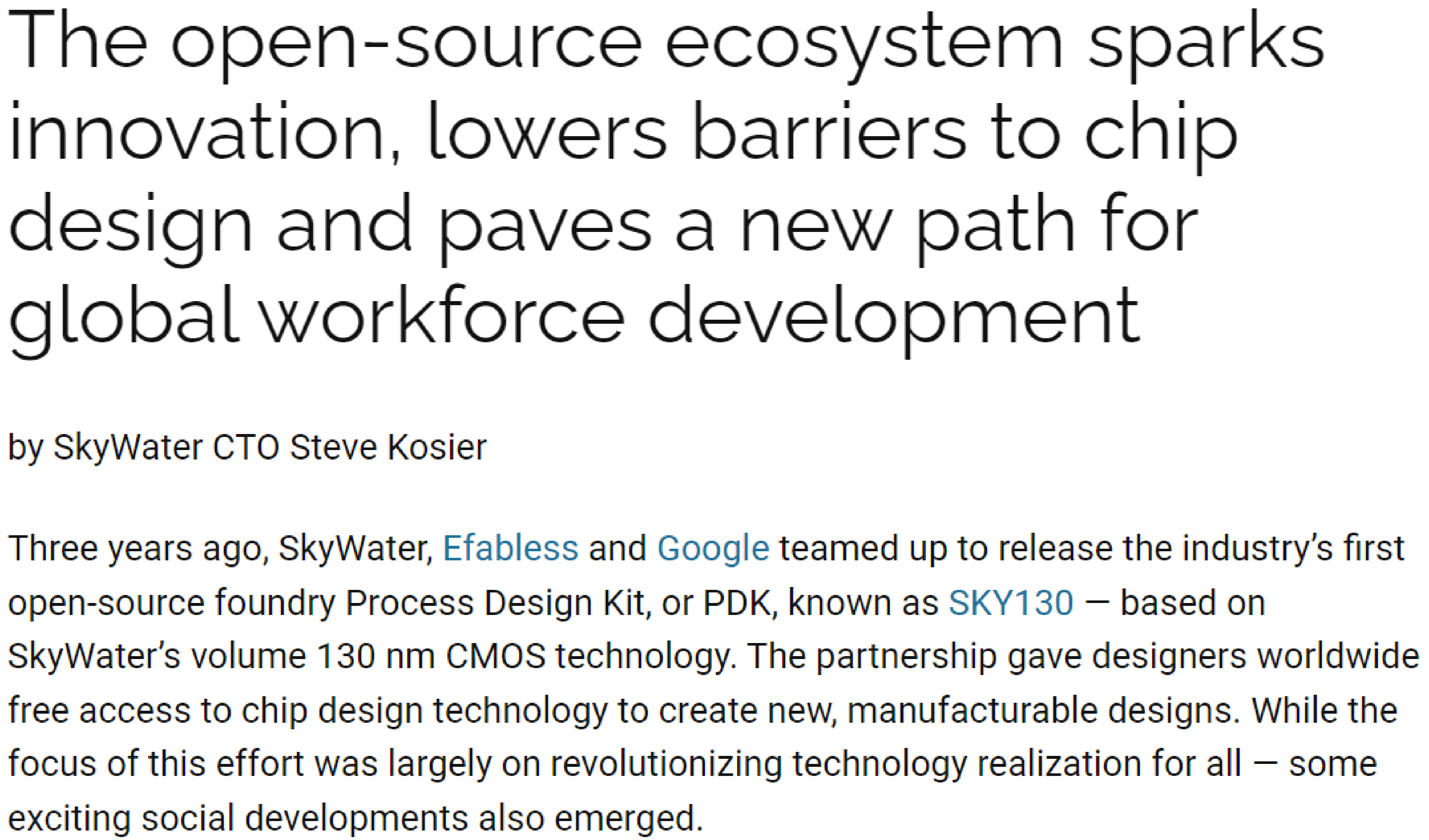


DESIGN TOOLS

- Popular logic synthesis tools:
  - Synopsys: Design Compiler (DC)
  - Cadence: Genus
  - Open Source: Yosys (will be used in this course)
- Popular RTL to gdsII tools:
  - Synopsys: IC Compiler (ICC)
  - Cadence: Innovus
  - Open Source: OpenLane (Google Skywater-130 PDK)

- tcl scripting is a must!

Open Source PDK (SkyWater 130)



The open-source ecosystem sparks innovation, lowers barriers to chip design and paves a new path for global workforce development

by SkyWater CTO Steve Kosier

Three years ago, SkyWater, Efabless and Google teamed up to release the industry's first open-source foundry Process Design Kit, or PDK, known as SKY130 — based on SkyWater's volume 130 nm CMOS technology. The partnership gave designers worldwide free access to chip design technology to create new, manufacturable designs. While the focus of this effort was largely on revolutionizing technology realization for all — some exciting social developments also emerged.

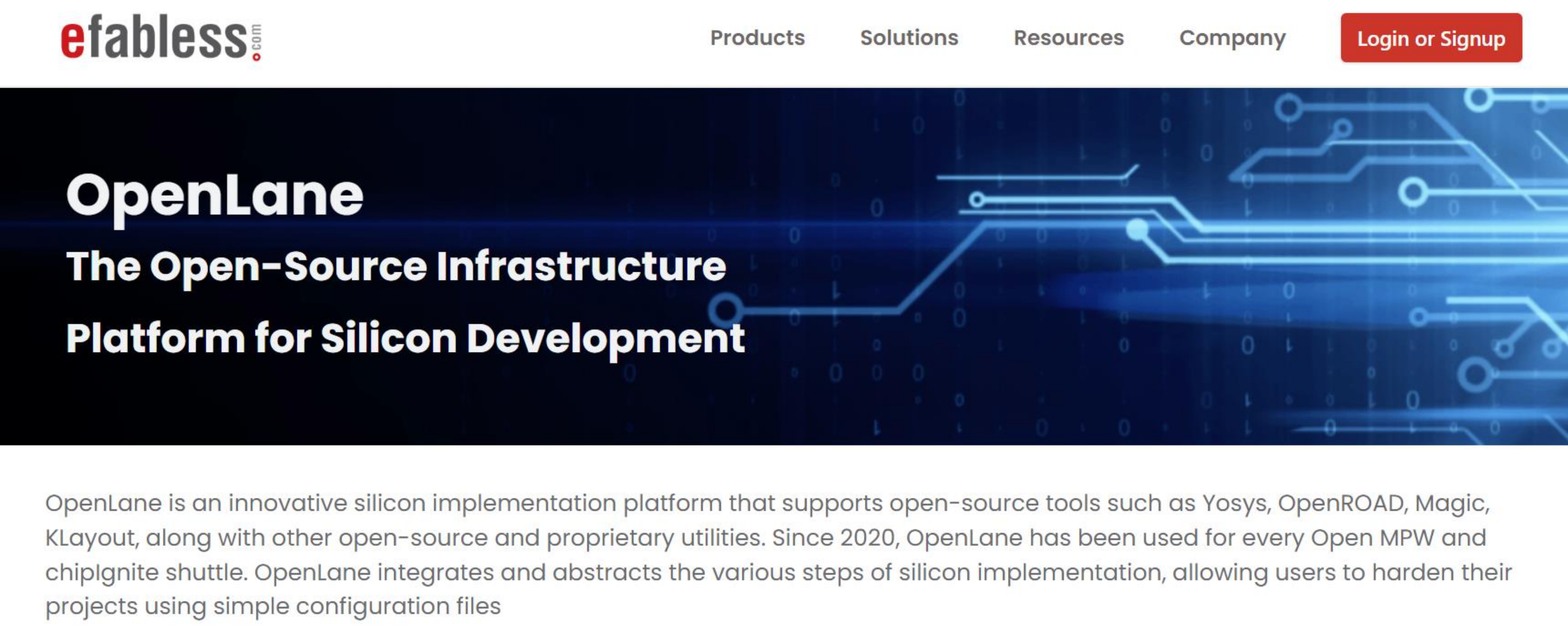
Open Source platform (efabless)

efabless.com

Products Solutions Resources Company

Efabless is the first creator platform for chips, empowering a global community of hardware and software developers, chip experts, students, and researchers to design, share, collaborate, fabricate, and commercialize their own semiconductor innovations. Our platform revolutionizes chip creation by providing the tools, resources, and network needed to bring ideas to life—from concept to production. We are more than just a platform; we are a collaborative ecosystem where innovation thrives and the boundaries of technology are constantly redefined.

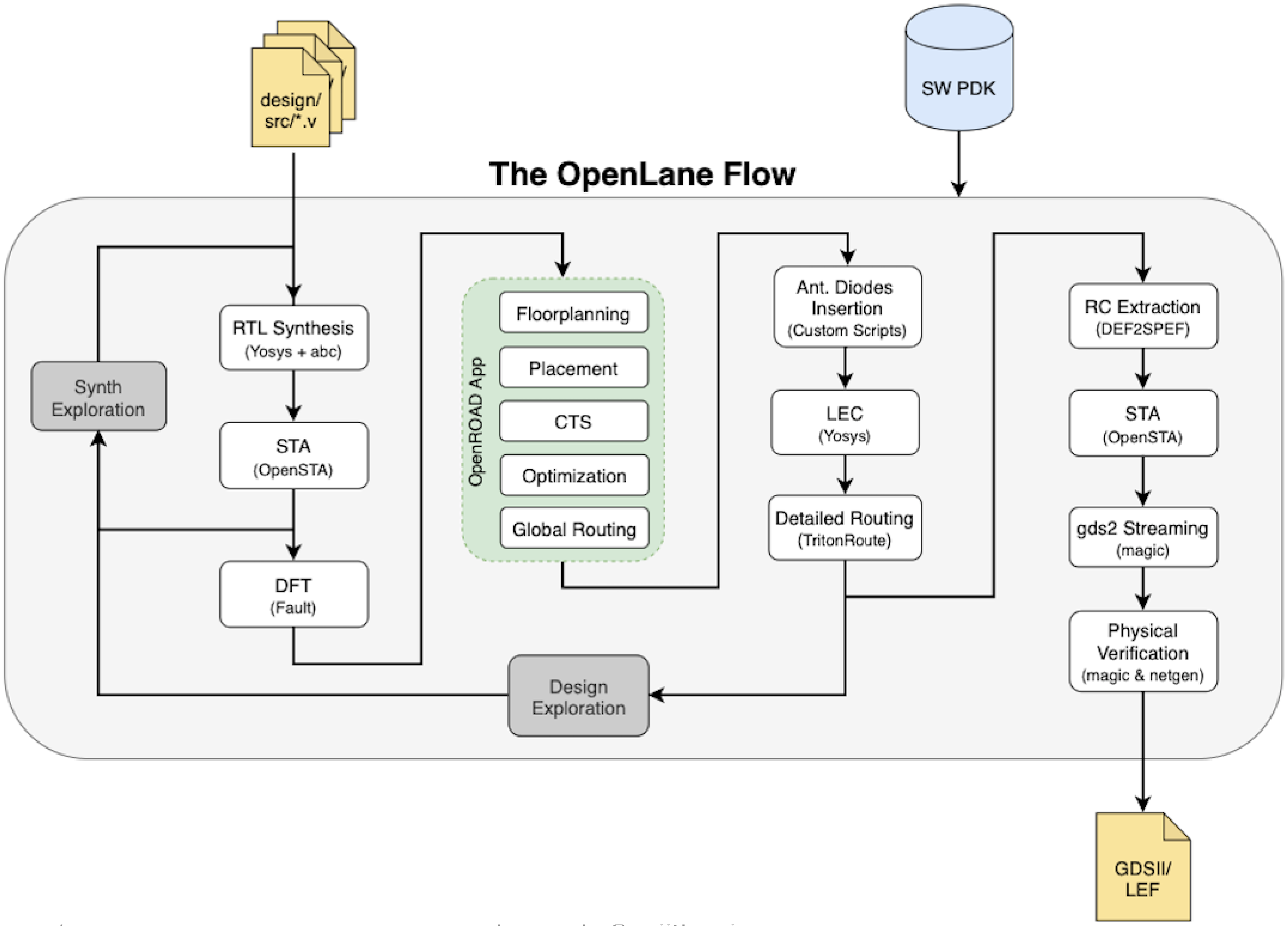
OPEN SOURCE PLATFORM (EFABLESS)



OpenLane
The Open-Source Infrastructure
Platform for Silicon Development

OpenLane is an innovative silicon implementation platform that supports open-source tools such as Yosys, OpenROAD, Magic, KLayout, along with other open-source and proprietary utilities. Since 2020, OpenLane has been used for every Open MPW and chipIgnite shuttle. OpenLane integrates and abstracts the various steps of silicon implementation, allowing users to harden their projects using simple configuration files

OpenLane: Steps & Tools used


design/
src/*.v
SW PDK
The OpenLane Flow
Synth Exploration
RTL Synthesis (Yosys + abc)
STA (OpenSTA)
DFT (Fault)
OpenROAD App
Floorplanning
Placement
CTS
Optimization
Global Routing
Ant. Diodes Insertion (Custom Scripts)
LEC (Yosys)
Detailed Routing (TritonRoute)
Design Exploration
RC Extraction (DEF2SPEF)
STA (OpenSTA)
gds2 Streaming (magic)
Physical Verification (magic & netgen)
GDSII/
LEF