

# Layout Synthesis Using Automatic Place-and-Route Tools - Brief Overview & Preparation

**Jaeha Kim, Sung-Joon Lee, and Eunseo Kim**  
**Mixed-Signal IC and System Group**  
**Seoul National University**  
**May 20. 2016**

# Chip Assembly

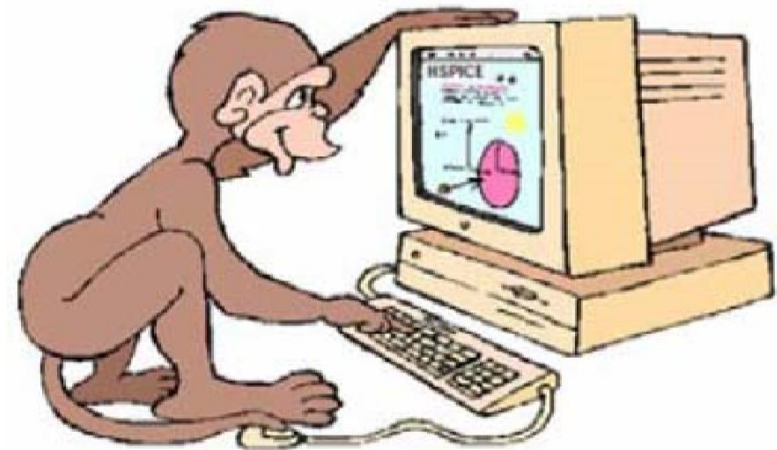
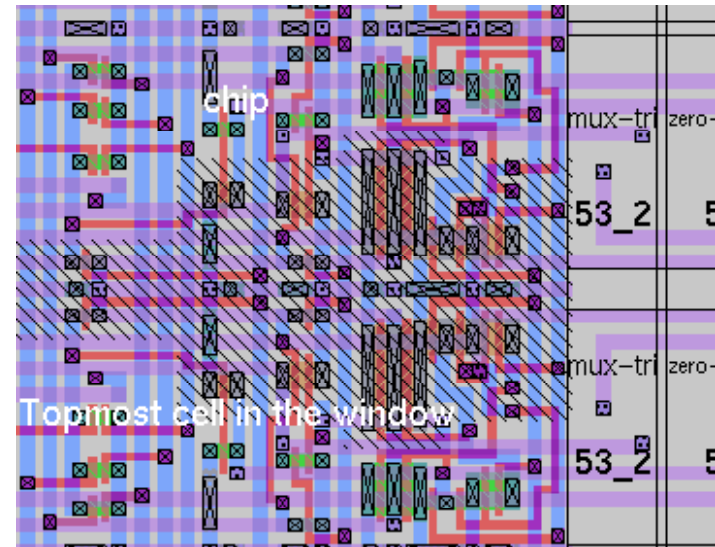
- ▶ Simply, it is process of integrating blocks
- ▶ Can resemble picture on right
  - ▶ Key is to have a early plan
  - ▶ And continue to update it
  - ▶ Need to have accurate floorplan
- ▶ Early floorplanning is critical
  - ▶ Sets the specs for the components
  - ▶ Functional, physical, timing



***Most chips are constructed by correction;  
not correct by construction!***

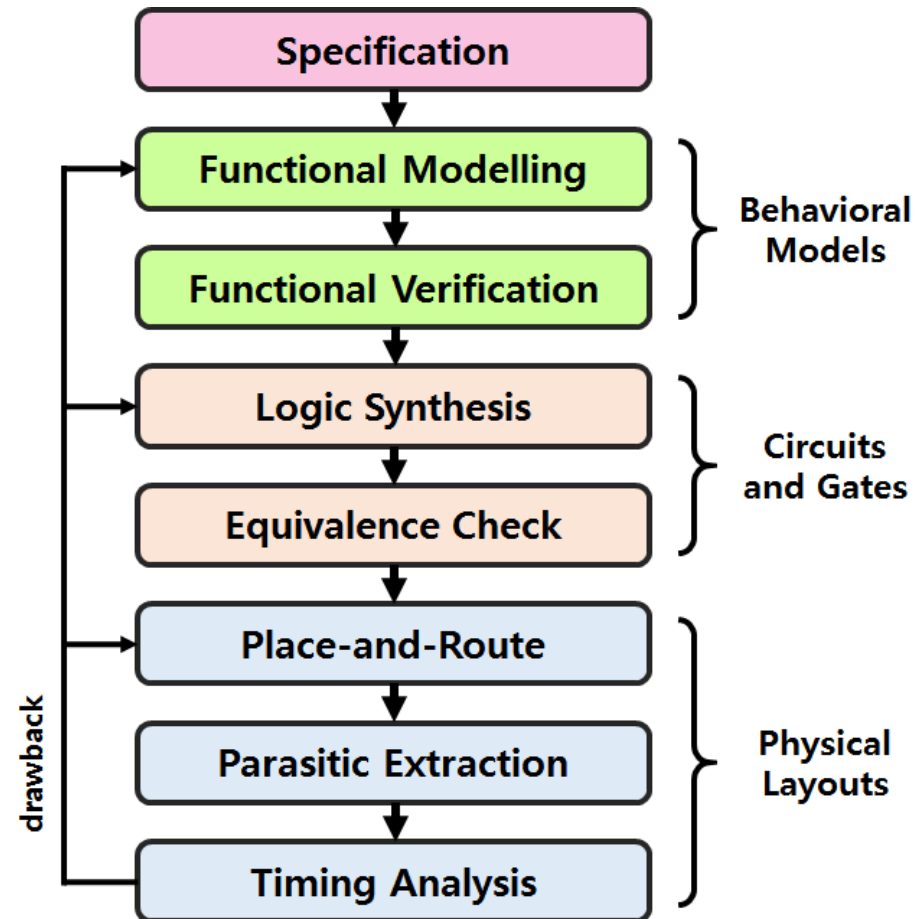
# Manual Design in AMS System

- ▶ “Bottom-Up” Design Flow
  - ▶ Design is a process of scaling the components to a system
- ▶ Chip assembly is achieved by human’s effort
- ▶ Correcting the error in the chip takes huge amount of time
  - ▶ Which is very annoying and painful process for the designer



# Design Automation in Digital System

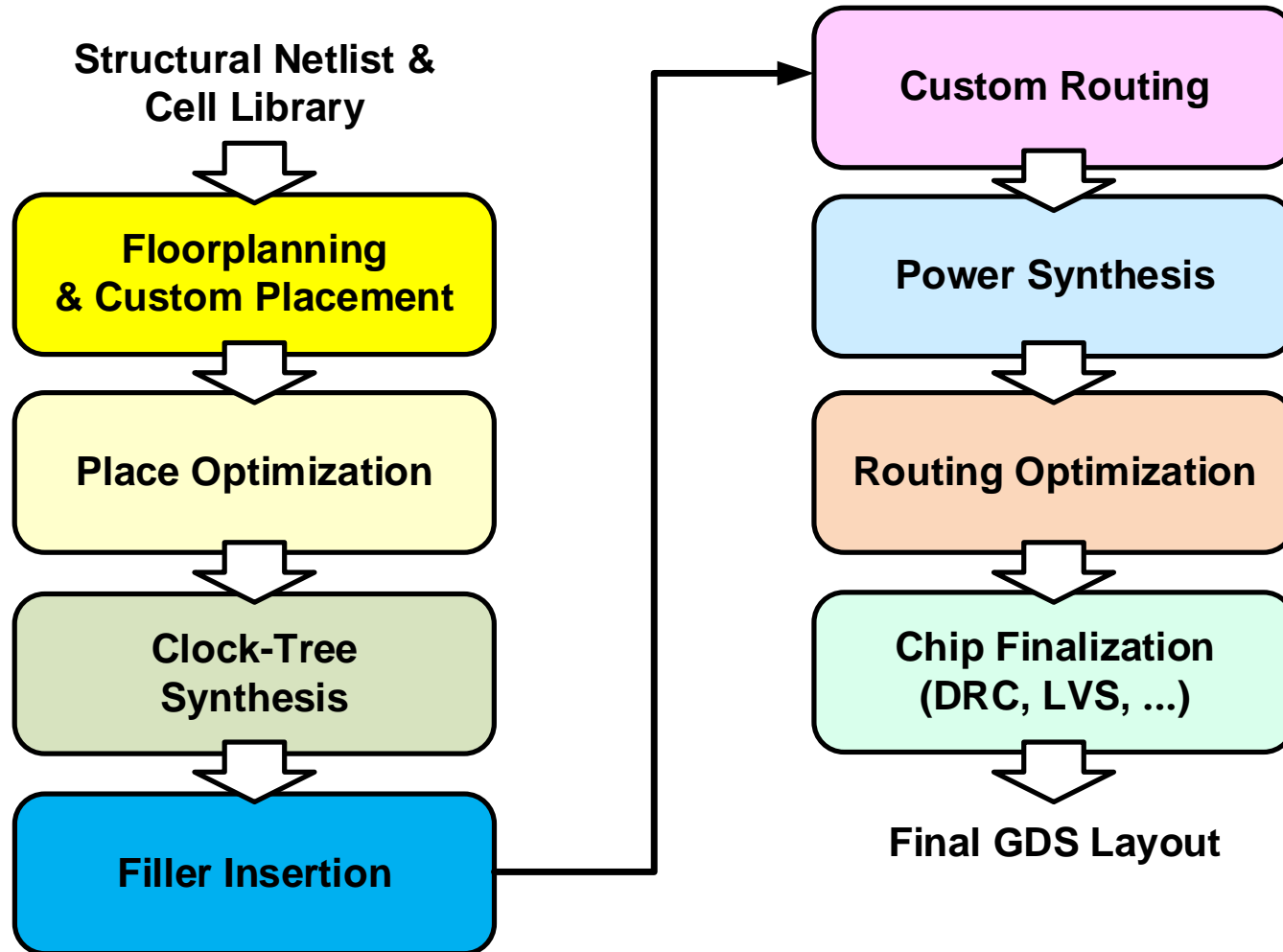
- ▶ “Top-Down” Design flow
  - ▶ a.k.a, Cell-Based Design flow
  - ▶ Designing the system of “intend” first and verifying the details later
- ▶ Chip assembly is achieved by automatic place and route tool
- ▶ Key is **iteration** :
  - ▶ enabling designer to do “construct by correction” easily



# Is it possible to use CBDF in AMS system?

- ▶ The answer is **YES**, if :
  - ▶ Each cell is well-defined with its functionality and abstraction
  - ▶ and therefore, we can define the pins of cell clearly

# Overview of Place & Route

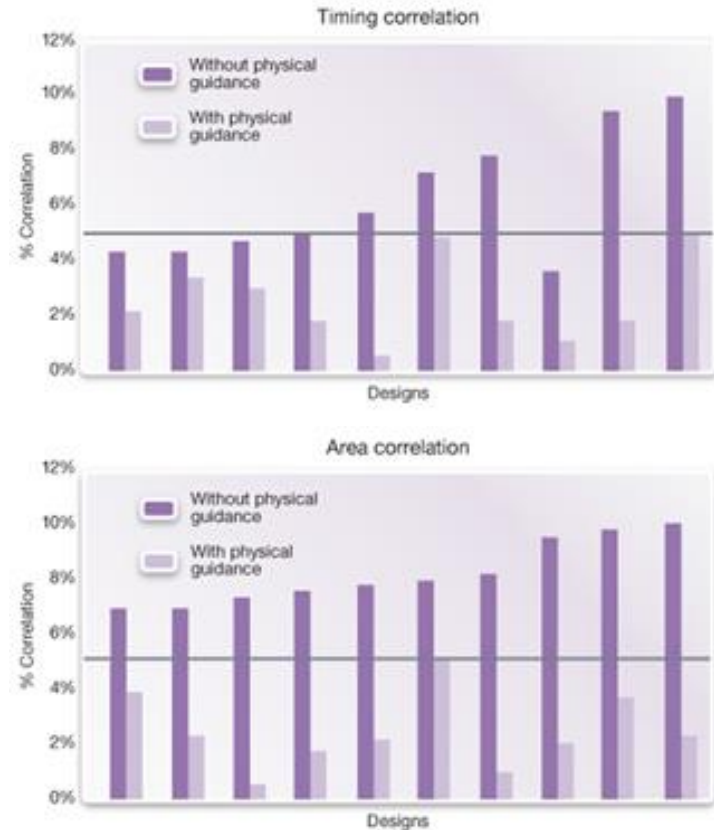


# About IC Compiler

- ▶ The Synopsys IC Compiler tool provides a complete netlist-to-GDS design solution, which combines :
  - ▶ proprietary design planning
  - ▶ physical synthesis
  - ▶ clock tree synthesis
  - ▶ and routing for logical and physical design implementations throughout the design flow

# About IC Compiler

- High-performance cell-based design tools



<DC's physical guidance to IC Compiler>



# About Custom Designer

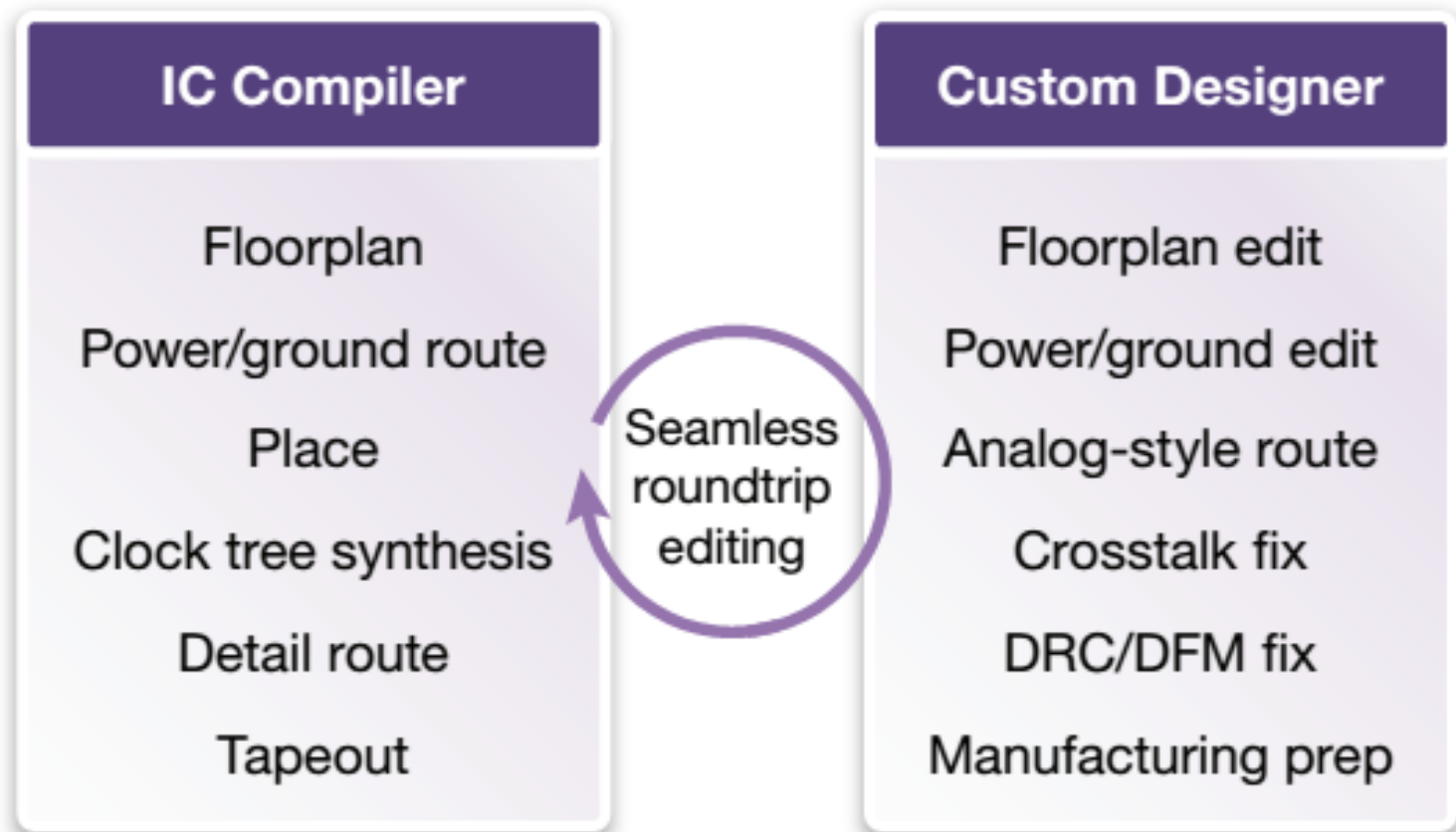
- ▶ Unified solution for custom and cell-based design and verification includes :
  - ▶ Full custom schematic and layout editing
  - ▶ Open architecture interoperable PDK (iPDK)
  - ▶ Unified custom and digital physical implementation
  - ▶ High-performance mixed-signal simulation
  - ▶ Integrated physical verification and parasitic extraction

# Places of CD in Design Flow

- ▶ Steps in the design flow covered by Custom Designer:
  - ▶ Schematic Design (Custom Designer - SE)
  - ▶ Symbol Creation (Custom Designer - SE)
  - ▶ Layout Design (Custom Designer - LE)
- ▶ Various tools for other steps are also integrated
  - ▶ Hercules, Calibre for physical verification
  - ▶ StarRC, Calibre-PEX for parasitic extraction
  - ▶ HSpice for simulation

# Why to Use Custom Designer?

## ► IC Compiler Custom Co-Design



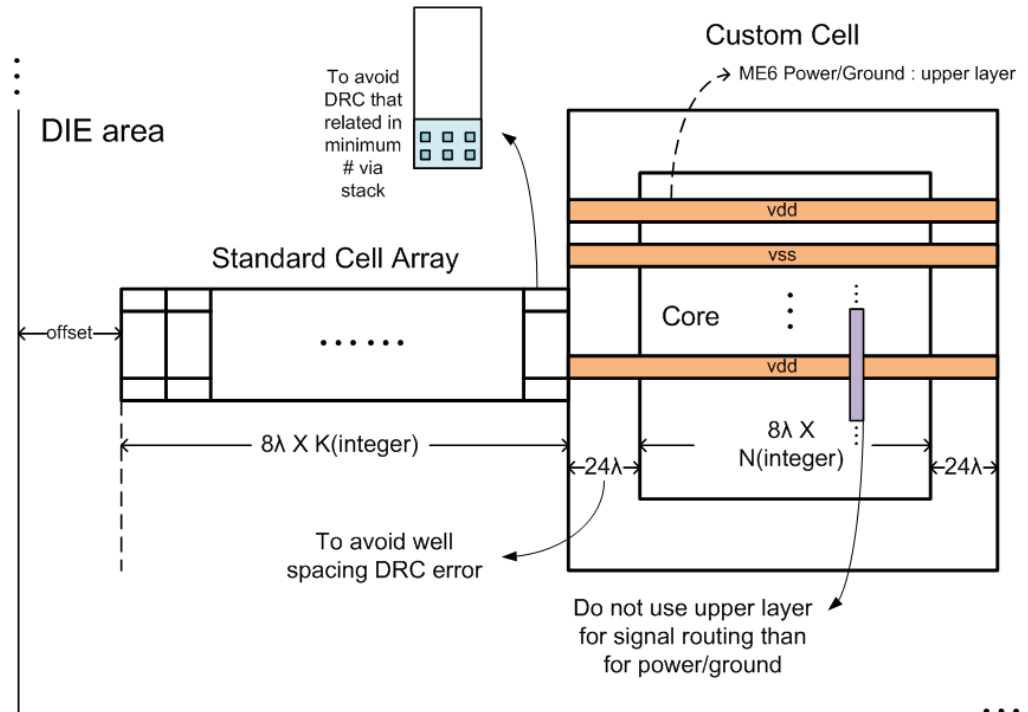
# Before doing Place & Route

# Chip-Level Model

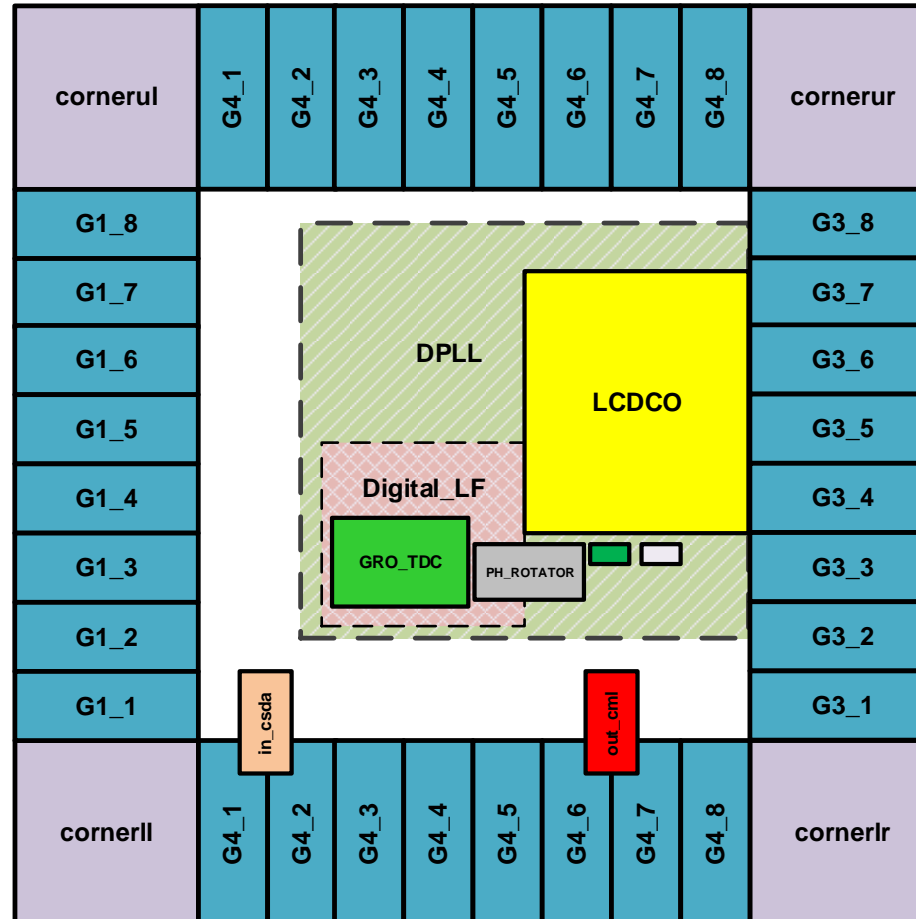
- ▶ Description of overall whole chip ,including your core circuit, drivers, I/O Pad
- ▶ IC Compiler use the chip-level model to get the connection information between the instances and to proceed P&R

# Floorplanning – $\lambda$ -Based Design

- ▶  $\lambda$  : half of minimum feature size of tech.
- ▶ For P&R, Cells must be placed in coarsely discretized grids
- ▶ Our P&R flow uses grid of  $8\lambda$  of x-axis and  $80\lambda$  of y-axis



# Chip-level Model Example



# Chip-level Model Example

► \$ vi ~/IDEC\_CBDF/xmodel\_dp11/top/top.sv

```
Module top(  
    input vdda,  
    input vss,  
    ... //Chip I/O Declaration  
);  
// PAD  
...  
padh_vss          G4_5 (.vdda_o(vdda),.vddt(vddt),.vss(vss));  
padh_empty G4_6 (.pad(clk_outn),.vdda_o(vdda),.vddt(vddt),.vss(vss));  
padh_empty G4_7 (.pad(clk_outp),.vdda_o(vdda),.vddt(vddt),.vss(vss));  
...  
// Core circuit  
dp11 dp11o ( .clk_ref(x_clk_ref), .clk_outp(x_clk_outp), .clk_outn(x_clk_outn));  
//driver circuit  
out_cml out_cml (.inp(x_clk_outp),.inn(x_clk_outn),.outp(clk_outp),.outn(clk_outn));  
in_csda in_csda (.inp(clk_ref_p),.inn(clk_ref_n),.out(x_clk_ref));
```



# Is Your Top Model Correct?

- ▶ When you are in a real chip design, connecting the inputs and outputs to the pad without any mistakes is really important
- ▶ Good way to examine the connection of **'top.sv'** is conducting simulation with the same testbench used in the previous DPLL simulation
- ▶ Simulates **'tb\_locking'** and checks the phase-lock

# Exercise: Top Model Simulation

- ▶ Testbench: 'tb\_locking'
- ▶ Change your directory to 'tb\_locking'

```
$ cd ~/IDEC_CBDF/xmodel_dpll/top/tb/tb_locking
```

- ▶ Run the simulation by typing 'make <mode>'

```
$ make tb_locking
```

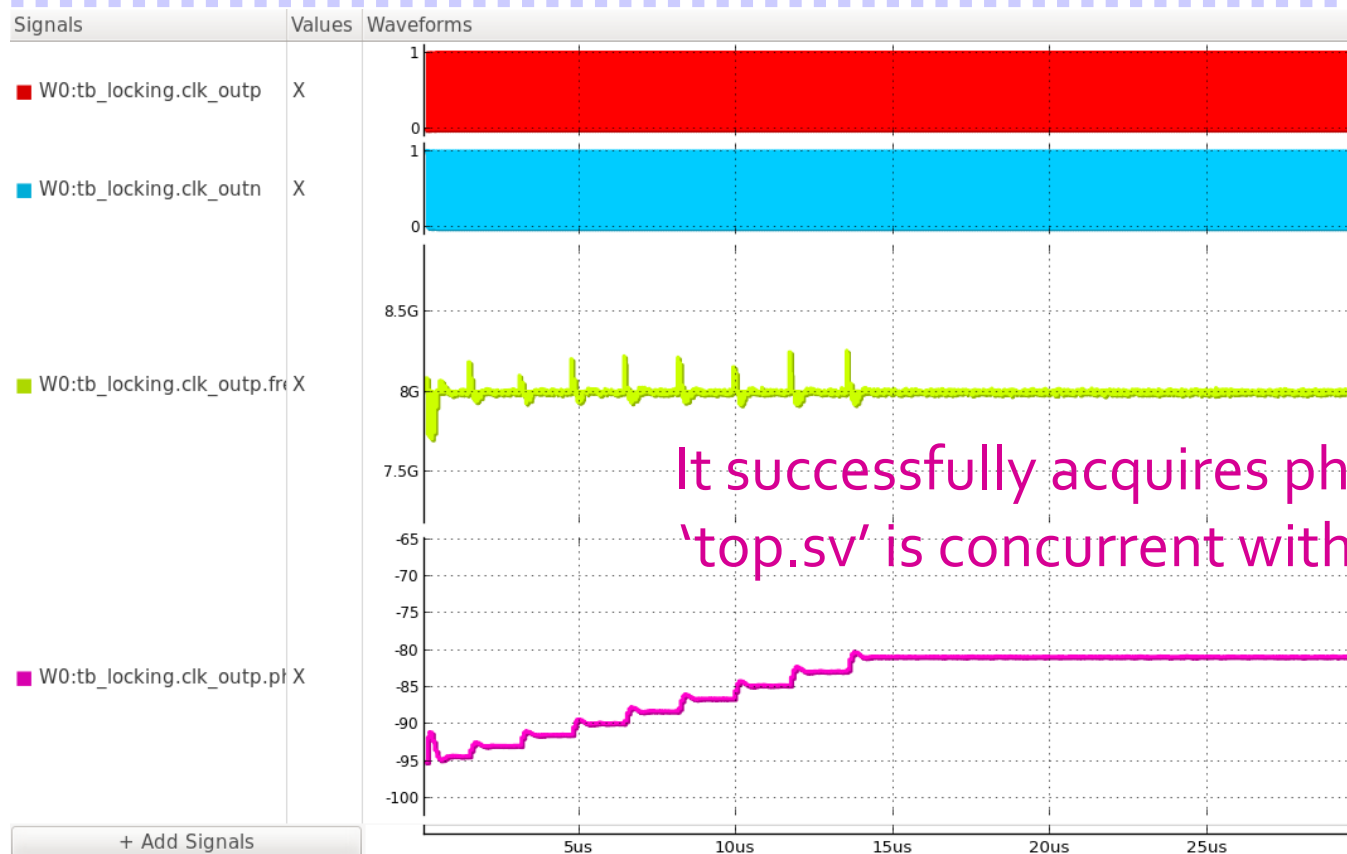
- ▶ (After you place-and-route the whole chip, you can run final simulation by typing 'make tb\_locking\_sdf')

- ▶ The simulation is now running

# Result - Locking

## ► Open XWAVE waveform viewer

```
$ xwave xmodel.jez
```



# Prerequisite Libraries for P&R

- ▶ **Logical library(.db)** – already used in Synthesis
- ▶ **Physical library** –MilkyWay Library to use in IC Compiler
- ▶ **Technology file(.tf)** – process technology file
- ▶ **TLUPLUS file(.tluplus)** - In general, IC Compiler imports the interconnect parasitics from TLUPlus files, rather than the technology file (ex-ASTRO)
- ▶ **ITF(.itf) file** – mapping file that has link between .tf and .tluplus

**Now, we are ready to do P&R!**