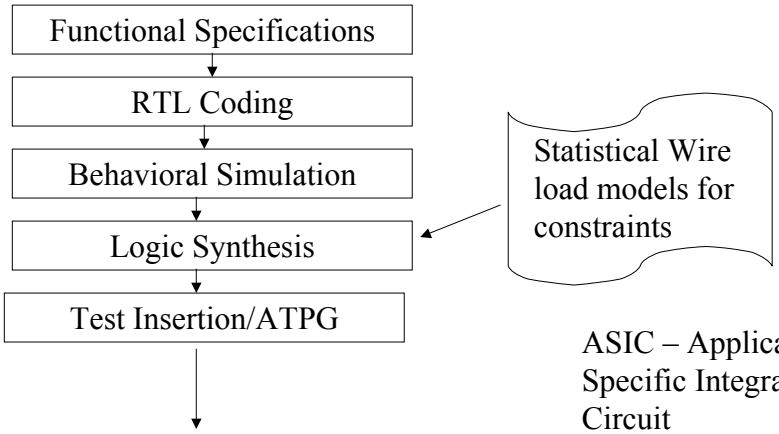
ECAD Tool Flows

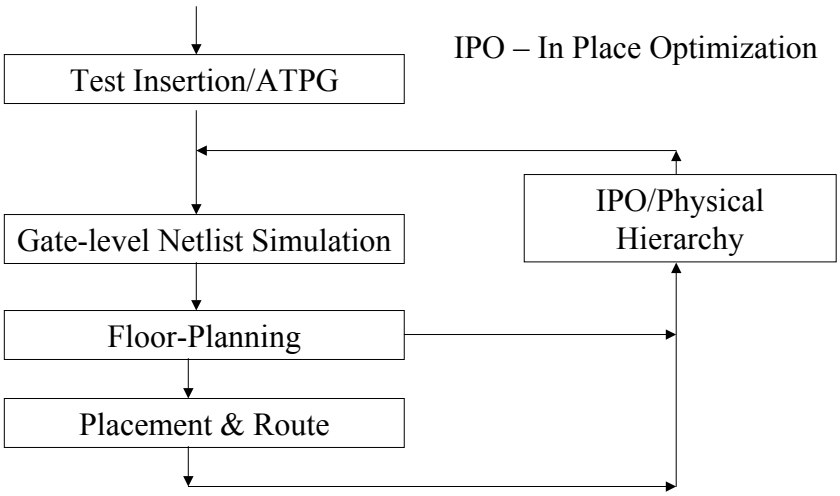
- These notes are taken from the book: *It's The Methodology, Stupid!* by Pran Kurup, Taher Abbasi, Ricky Bedi, Publisher ByteK Designs, (http://www.bytekinc.com (now a defunct link)
- A *tool flow* describes the method and order (the methodology) in tools are used produce a design
  - Different companies can take the same collection of tools and use a different methodology to produce an IC
  - Typically, companies will add their own in-house tools to the off-the-shelf tools to tailor the tool flow to their particular needs
  - An ECAD group is usually responsible for designing and maintaining the methodology – they are also responsible for training others in the use of this methodology.

An ASIC Flow



ASIC – Application Specific Integrated Circuit
ATPG – Automated Test Pattern Generation

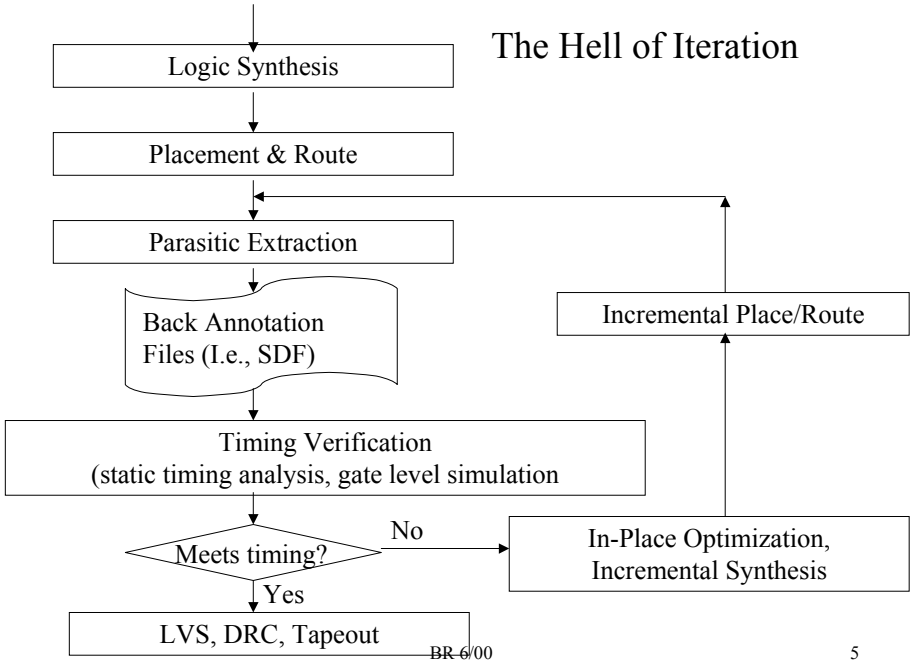
An ASIC Flow (cont)



"Timing Closure"

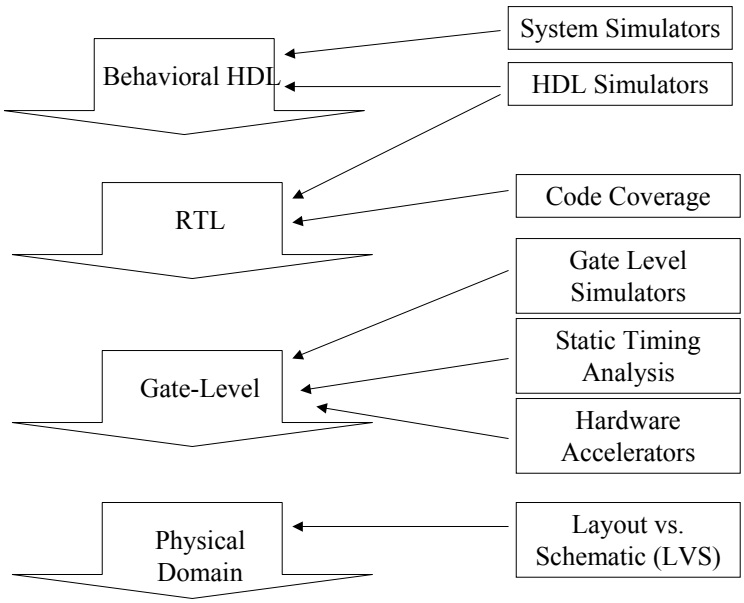
- "Timing Closure" refers to producing a design that meets timing specifications
  - Want to verify that your design has timing closure before you fabricate
- The problem is that with deep submicron, the parasitics (R,L,C) of the physical layout greatly affect timing
- This leads to the need to produce layout, extract parasitics from the layout, 'back-annotate' the parasitics into your timing verification methodology, and then modify logic/layout in order to meet timing specifications and achieve timing closure

The Hell of Iteration

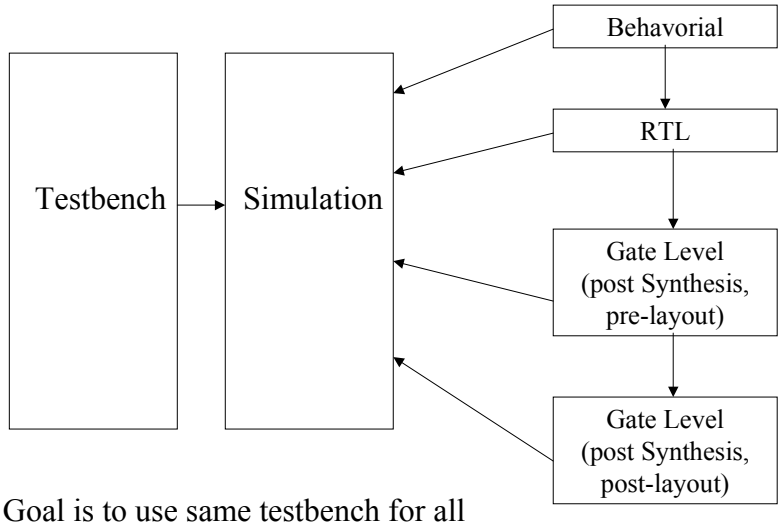


Loop through P&R is Time Consuming

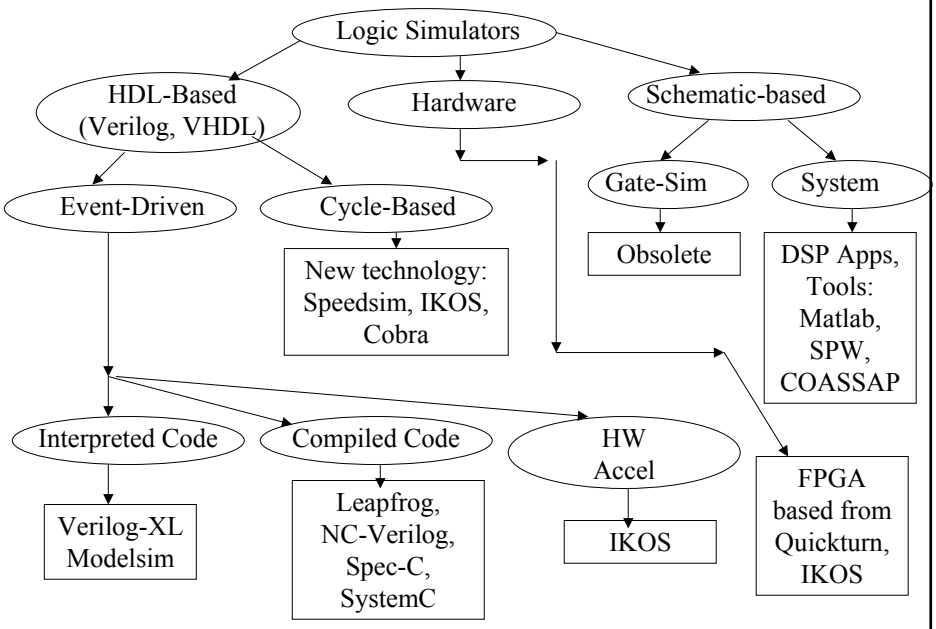
- In-Place Optimization essentially means to tweak transistor sizes without moving cells around
  - Moving cells around requires routing to be modifed
- If you have to move cells or change the netlist such that different cells are used, then would like to do it in a local area so that you do not have go through the entire Place/Route process again
  - Incremental Place/Route tries to accomplish this
- Incremental synthesis means to read in the gate level netlist, and modify the netlist incrementally in order to meet failed timing constraints
  - Does NOT START with RTL, starts with gate level netlist produced by first synthesis pass
  - Goal is to change as few as gates as possible, and then use incremental place/route to change the layout.



Verification Methodology



Goal is to use same testbench for all levels of simulation abstraction, and mix different levels



Comments on Logic Simulators

- Hardware Emulators usually FPGA-based, used to test functionality of design (can't simulate at speed)
  - Very expensive, used because 10x-100x faster than software simulation
- Hardware Acceleration usually means parallel execution of VHDL/Verilog/C/C++
- Push towards using C/C++ as simulation language
  - Compiled code can be faster than VHDL/Verilog simulators
  - VHDL/Verilog usually compiled to a 'byte code' form, then byte code is interpreted
  - Some simulators will convert Verilog/VHDL to C/C++, then compile for extra speed.
- Cycle-based simulations do not compute what is going on inside of a clock, just results from clock-to-clock
  - This is a higher level of abstraction, code can be written in either VHDL/Verilog or C/C++

Formal Verification

- *Formal Verification* means that mathematical techniques to *prove* that the hardware is correct as it progresses from one abstraction level to another (Behavioral to RTL to Gate-level to Physical), etc.
- Attraction is that circuit does not have to be simulated – no need for test vector generation
  - Generation of test vectors, simulation/checking of vectors time consuming
  - Test vectors may not cover all possible cases
- Formal Verification is difficult, very much a research area
- Is currently a 'hot' area for tool development

Formal Verification Categories

- Equivalence Checking – most widely used, easiest
  - Use a mathematical approach to compare a reference design to a revised design (do two netlists implement the same boolean function?)
  - Reference design must be correct
- Model Checking – a research area
  - Compare a design implementation against a set of properties (the model) that defines the behavior
  - Properties define the specifications of the design
  - Incorporates elements of Equivalence checking but goes beyond this
- Theorem Proving – most advanced
  - Formally prove two designs are correct
  - Designs must be represented in a 'formal' specification language that incorporates the specifications of the design in additon to the behavior
  - VHDL/Verilog does not include this though extensions have been proposed.

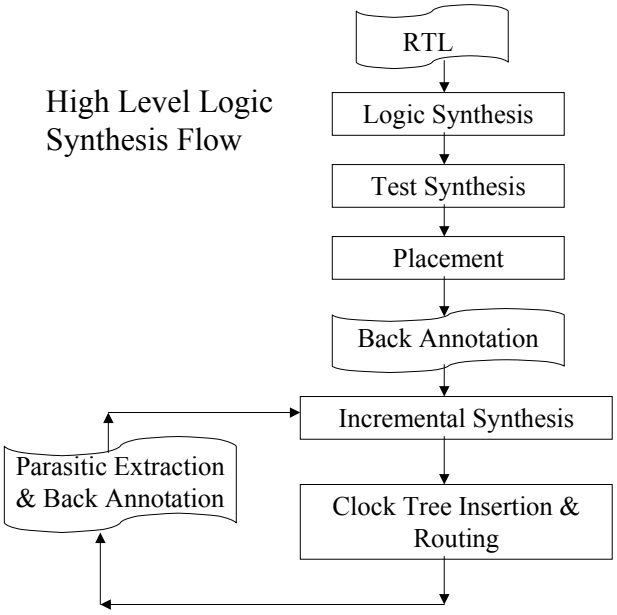
Static Timing vs. Full Simulation Timing

- Static timing traces paths in the design, computes delays along the paths, and checks if delay constraints met
  - No need for vector generation
  - Cannot detect glitches, timing failure due to dynamic behavior (such as charge sharing problems)
  - Static analysis used to direct synthesis optimization – synthesis tools computes delay paths and tries to produce netlists that meet constraints
  - Setup time violations checked for with 'slow corner' timing library
  - Hold time violations checked for with 'fast corner' timing library
- Full Simulation timing is time consuming, requires test vector generation, but only way to detect dynamic timing faults

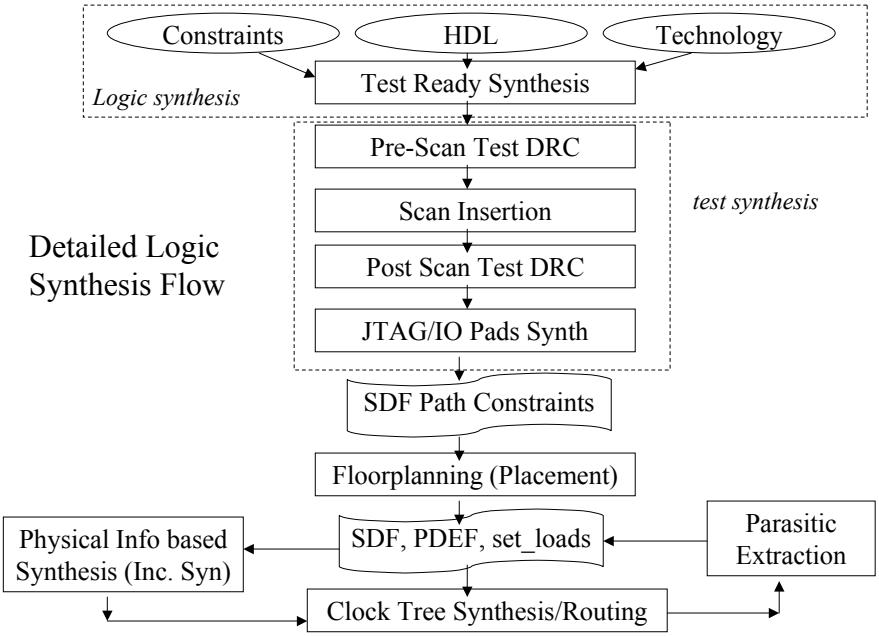
Timing-Driven Layout

- One way to reduce time spent in timing closure iteration is to have a Place/Route tool capable of timing-driven layout
- During Place/Route, tool uses timing constraints to driven generation of layout (primarily routing)
  - Must be able to accurately estimate wire delays during place/route procedure
  - Goal is to produce layouts that meet timing constraints so that iterations through physical layout are minimized
- Most modern P/R tools support timing driven layout

High Level Logic Synthesis Flow



Detailed Logic Synthesis Flow



Logic Synthesis Constraints

- Synthesis driven by constraints
- Timing Constraints
  - Clock period, setup time, hold time constraint
- Area constraints
- Power Constraints
  - Gate choices can definitely affect power consumption
  - Logic synthesis can generate gating that minimizes the number of transitions during operation
- Design Rule Constraints
  - Maximum loading on outputs
  - Maximum transition time on outputs

File Formats

File	Expansion	Description
LEF PLEF	Library Exchange Format Parametric LEF	Format developed by Cadence
DEF	Design Exchange Format	Format developed by Cadence
SPEF	Standard Parasitic Exchange Format	Industry standard format
SDF	Standard Delay Format	Industry standard format giving pin to pin delays
PDEF	Physical Design Exchange Format	Industry standard format for physical cluster and placement information (initiated by Synopsys)

SDF files can be read by Synopsys – contains both gate delays and interconnect delays. SDF can also be generated by Synopsys. VHDL/Verilog simulators can also use SDF.

Example 3-3 SDF v2.1 File

```
(DELAYFILE
(SDFVERSION "OVI 2.1")
(DESIGN "COUNTER4")
(DATE "Wed Jul 27 08:18:43 2000")
(VENDOR "class")
(PROGRAM "Acme EDA Co.")
(VERSION "2000.11")
(DIVIDER /)
(VOLTAGE 5.00:5.00:5.00)
(PROCESS "typical")
(TEMPERATURE 25.00:25.00:25.00)
(TIMESCALE 1ns)
(CELL
  (CELLTYPE "IVA")
  (INSTANCE U41)
  (DELAY
    (ABSOLUTE
    (IOPATH A Z (0.350:0.350:0.350) (0.340:0.340:0.340))
    )))
```

Header Information

Min:Typical:Max

Rising Delays

Falling Delays

Delay from input pin to output pin


```
(CELL
  (CELLTYPE "FD1")
  (INSTANCE REG_BLK1/F0)
  (DELAY
    (ABSOLUTE
    (IOPATH CP Q (3.199:3.199:3.199) (2.126:2.126:2.126))
    (IOPATH CP QN (1.959:1.959:1.959) (1.502:1.502:1.502))
    ))
  (TIMINGCHECK
    (SETUP D CP (0.800:0.800:0.800))
    (HOLD D CP (0.400:0.400:0.400))
  ))
(CELL
  (CELLTYPE "counter")
  (INSTANCE )
  (DELAY (ABSOLUTE
    (INTERCONNECT  mt_reg_18/QN U31/I (.006:.006:.006) (.005:.005:.005))
    (INTERCONNECT U8/ZN U95/A2 (.235:.235:.235)(.250:.250:.250))
    (INTERCONNECT U8/ZN U97/A2 (.229:.229:.229)(.244:.244:.244))
  ))))
```

Setup/Hold constraints

Interconnect Delays

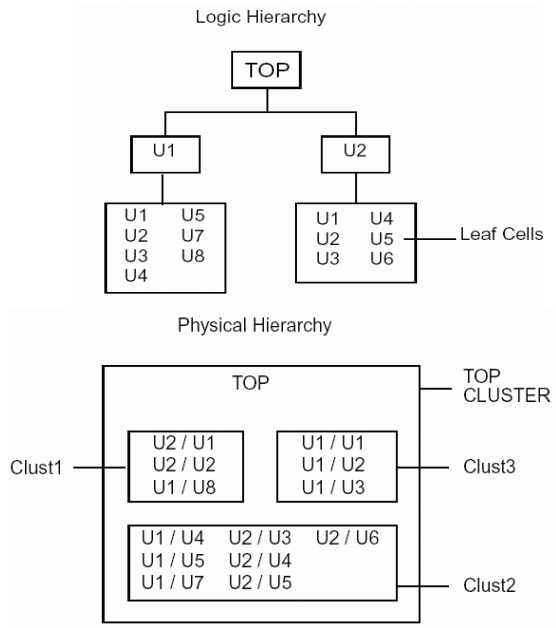
PDEF – Physical Design Exchange Format

- Would like to exchange clustering information between front end tools (logic synthesis - Design Compiler) and back end tools (physical layout)
  - 'clustering' means that the layout tool needs to place a group of cells (a cluster) close together because they are related
  - This will hopefully minimize routing delays between these cells

Example 3-7 PDEF v2.0 File

```
(CLUSTERFILE
(PDEFVERSION "2.0")
 (DESIGN "top")
 (DATE "May 6, 1996")
 (VENDOR "Synopsys, Inc.")
 (PROGRAM "Design Compiler")
 (VERSION "v3.5")
 (DIVIDER /)
 (CLUSTER
  (NAME "TOP_CLUSTER")
  (X_BOUNDS  0.0 50.0)
  (Y_BOUNDS  0.0 50.0)
  (CLUSTER
   (NAME "Clust1")
   (X_BOUNDS  0.0  20.0)
   (Y_BOUNDS  0.0  20.0)
   (CELL (NAME U2/U1) (LOC 10.0 10.0))
   (CELL (NAME U2/U2) (LOC  5.0  7.0))
   (CELL (NAME U1/U8) (LOC  6.3  7.0))
  )
```

Cluster Definition



Info in logic synthesis tool

Placement by floorplanner

Standard Parasitic Exchange Format (SPEF)

- Exchange parasitics between layout tools and delay calculators - delay calculators use parasitics to produce SDF files

```
*R_NET rgb_reg_data_outE_23 0.056
/*      *R_NET net_name total_cap */
*DRIVER RGB_REGISTERS/reg_out5_reg_23/Q
/*      *DRIVER pin_name */
*CELL DFFX1
/*      *CELL cell_name */
*C2_R1_C1 0.028 0.076 0.028
/*      *C2_R1_C1 C2_value R1_value C1_value*/
*LOADS
*RC SHIFT/U23/A 0.002/**RC load_pin load_value*/
*RC SHIFT/U14/A 0.001/**RC load_pin load_value*/
```

LEF files can contain the following statements:

```
[ VERSION number
    NAMESCASESENSITIVE statement
    NOWIREEXTENSIONATPIN statement
    BUSBITCHARS statement
    DIVIDERCHAR statement ]
[ UNITS statement ]
[ PROPERTYDEFINITIONS statement ]
{ LAYER (Nonrouting) statement
| LAYER (Routing) statement }...
{ VIA statement }
{ VIARULE statement
| VIARULE GENERATE statement }...
[ NONDEFAULTRULE statement ]
[ UNIVERSALNOISEMARGIN statement]
[ EDGERATETHRESHOLD1 statement]
[ EDGERATETHRESHOLD2 statement]
[ EDGERATESCALEFACTOR statement ]
[ NOISETABLE table ]
[ CORRECTIONTABLE table ]
[ SPACING statement ]
[ MINFEATURE statement ]
[ DIELECTRIC statement ]
[ IRDROP statement ]
{ SITE statement }...
[ ARRAY statement ]...
{ MACRO macroName
    Macro data
    { PIN statement }...
    { OBS statement }...
    [ TIMING statement ]
    END macroName }...
```

LEF files describe physical information for layout libraries – used by external place/route tools

Header contains information for technology (layers, spacing, etc).

Macro statements define each cell (pins and obstructions – timing info needed for timing driven layout)


```
VERSION statement
NAMESCASESENSITIVE statement
BUSBITCHARS statement
DIVIDERCHAR statement
DESIGN statement
[ TECHNOLOGY statement ]
[ ARRAY statement ]
[ FLOORPLAN statement ]
```

```
[ UNITS statement ]
[ HISTORY statement ]...
[ PROPERTYDEFINITIONS section]
[ DIEAREA statement ]
[ ROW statement] ...
[ TRACKS statement ]...
[ GCELLGRID statement ]...
[ DEFAULTCAP section ]
[ CANPLACE statement ] ...
[ CANNOTOCCUPY statement ] ...
[ VIAS statement ]
[ REGIONS statement ]
COMPONENTS section
[ PINS section ]
[ PINPROPERTIES section ]
[ SPECIALNETS section ]
NETS section
[ IOTIMINGS section ]
[ SCANCHAINS section]
[ CONSTRAINTS section ]
[ GROUPS section ]
[ BEGINEXT section]
END DESIGN statement
```

DEF files contains final placed/routed design

Produced by Silicon Ensemble after placement/routing, imported back into Cadence layout editor (Virtuoso).

Contains physical information for routes, pin placement, cell placement