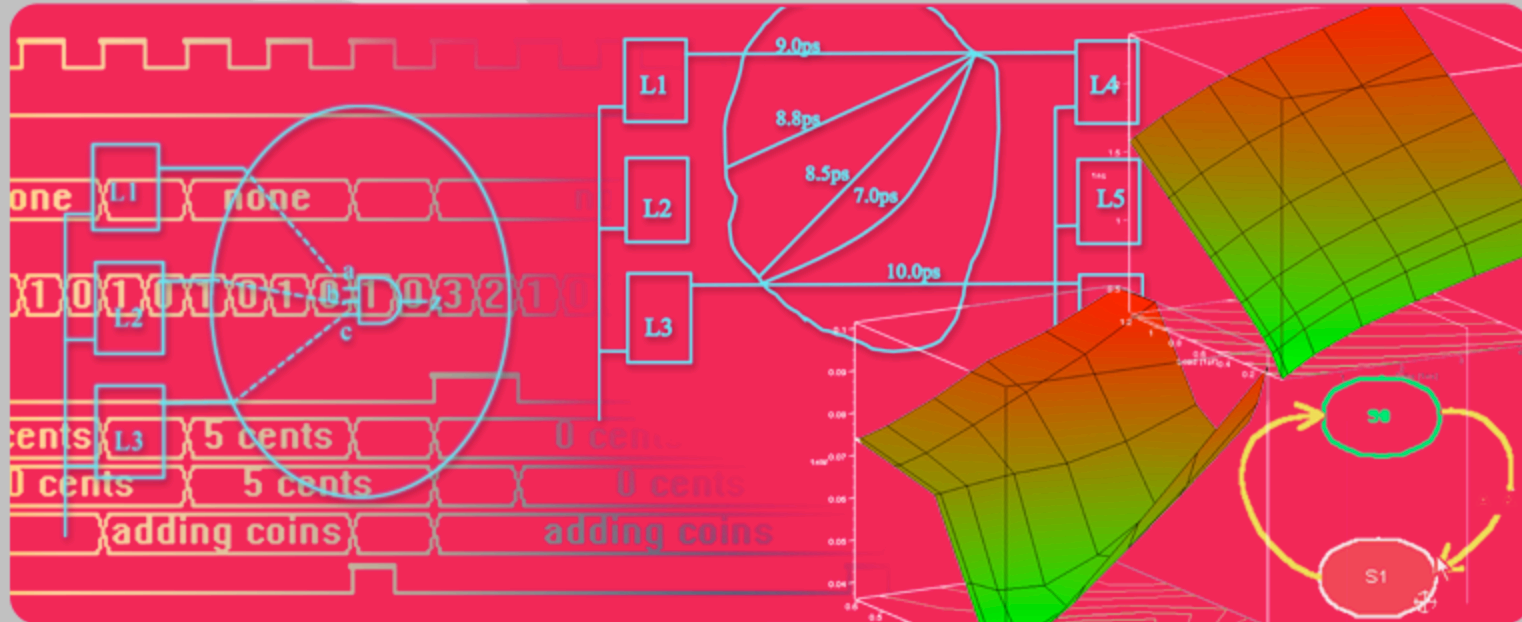
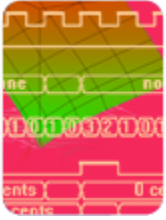


AccuCore –

SPICE Accurate Core Characterization with STA

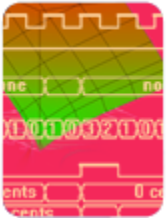


Silvaco Japan Technology Seminar
Spring 2007

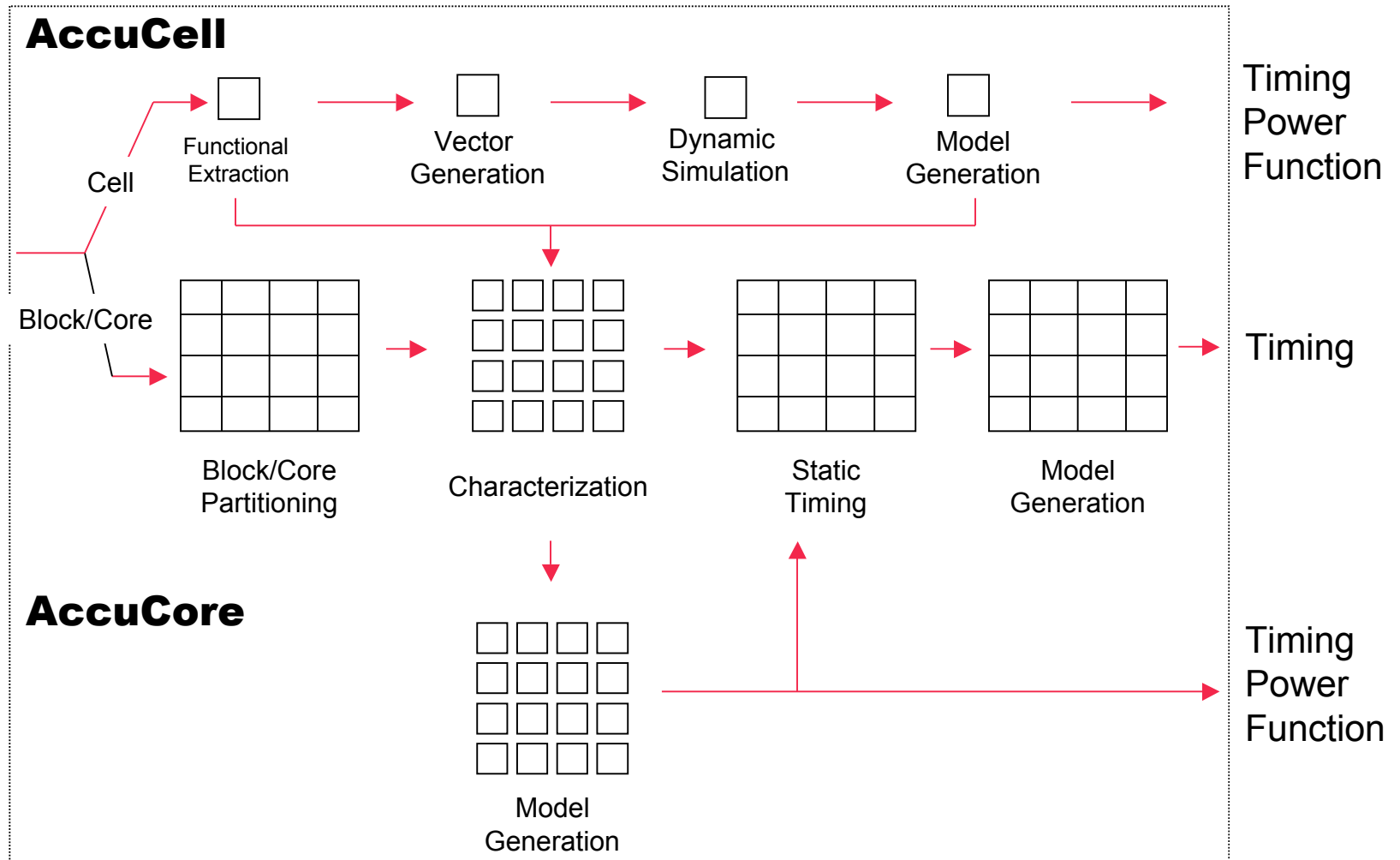


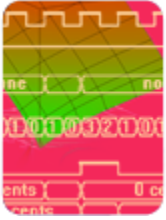
What is AccuCore? Why would I use it?

- AccuCore performs automatic block SPICE characterization and Static Timing Analysis
- Anyone who designs complex multi-million device ICs with various mixed static AND DYNAMIC design styles implemented in the latest VDSM technologies (90nm, 65nm and below) needs this product
- Adv. Static & Dynamic Designs
- Large Hier. or Flat RC designs
- Auto Partition / Function Extraction & Vector Generation
- State-Dependent Timing .lib & TLF
- Auto False Path Removal
- Export to SPICE of Critical Paths
- Compressed Model Generation
- Integrated SmartSpice API
- Firebird DB - incremental
- Supports ALL SPICE models
- Supports third-party SPICE



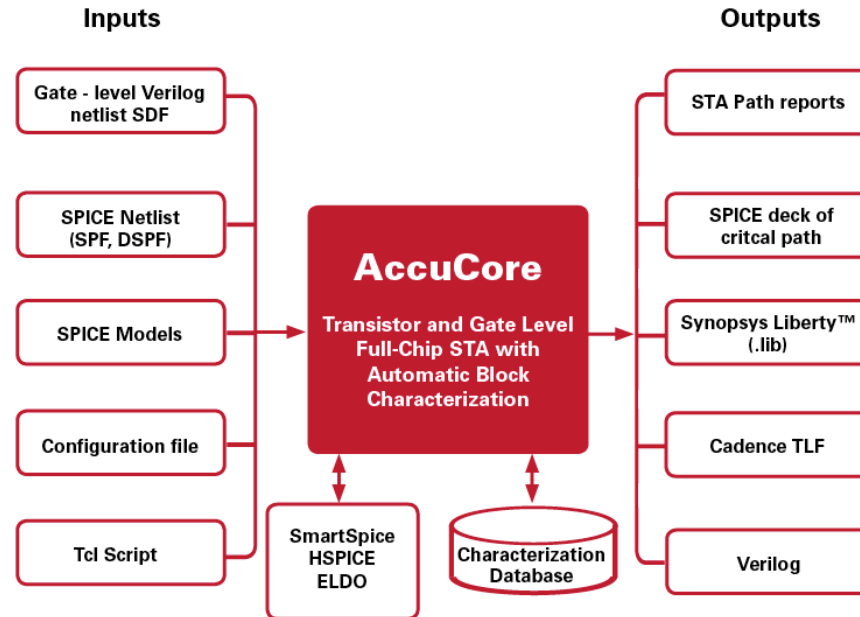
Simucad Cell and Core Characterization Flow





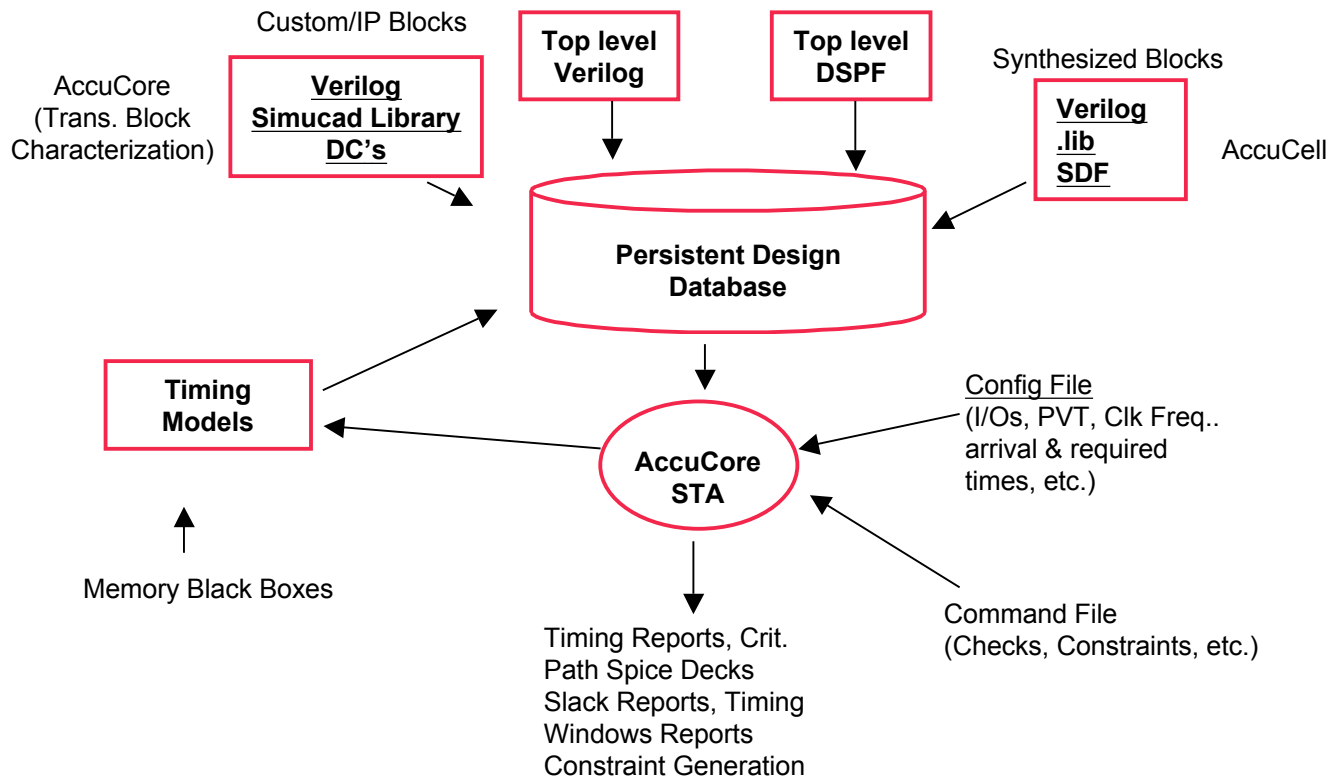
AccuCore – High Performance Timing Engine

- Support multiple input and output industry formats
- Incremental characterization with firebird database
- Gate-level timing analysis - fast & easy w/ support of ALL SPICE models
- Fast Hierarchical Full Chip timing analysis with automatic block level timing model options
- Support both static & dynamic logic and FULL .tcl automation scripting



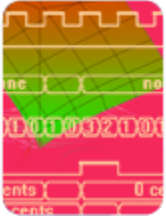


AccuCore STA Full Chip Flow



AccuCore Static Timing Analysis

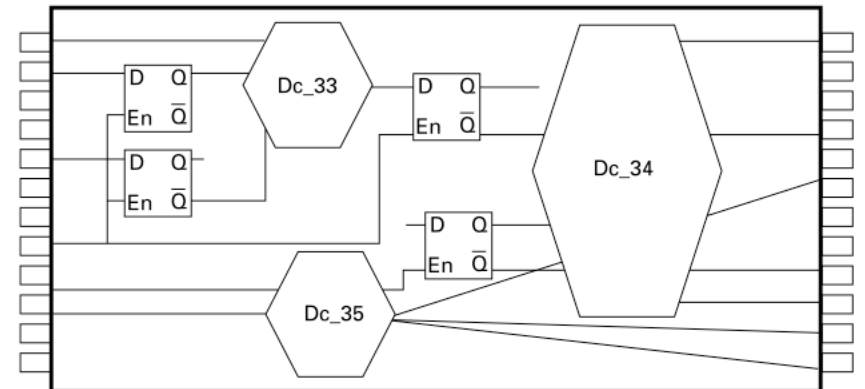
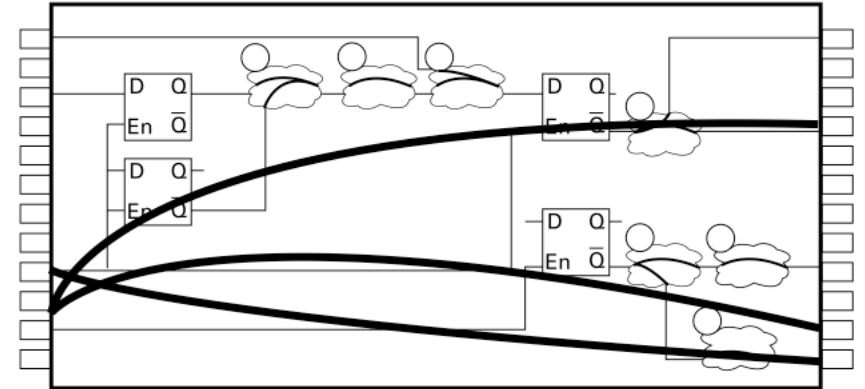
1. Reads in “All paths Models”
2. Performs Timing Checks on block
3. Performs Critical Path Analysis on block
4. Creates Spice decks for Critical Paths
5. Creates “Compressed Model” of block



AccuCore STA Timing Modeling

All-Paths Model

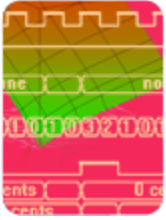
- A complete picture of the design
- **Compressed Model (N-level)**
 - Combinatorial devices collapsed
 - Debug path information
 - Support delay and slope tables
- **Black Model**
 - Debug path information
 - Support delay and slope tables
- **Interface Ring Model**
 - Leaves the interface logic intact
- **Interface Compressed Model**
 - Leaves interface devices intact, collapse the rest of combinatorial logic
- **Block Constraint Generation**
 - To block boundary for ASIC blocks
 - To individual pins for custom block
- **Clock Skew Analysis**





Description of the AccuCore Process

In the following 8 slides, after step 1, AccuCore processes the design automatically. The first step is the ONLY manual user step (creation of the .cfg file). The key part is partition control discussed in detail on the next slide.



Step 1: Create .cfg with KEEP_SUBCKT, FIND_SUBCKT

In some cases you may want to force the AccuCore partitioning algorithm to preserve certain circuit structures as a single partition. If the circuit structure resides within a space subckt (ie. a .subckt...) then the `KEEP_SUBCKT` config command can be used

The syntax for `KEEP_SUBCKT` is:

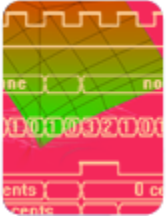
```
KEEP_SUBCKT <subckt_name> <inputs> <outputs> <bidirs> \  
<clocks> <optional_table_filename>
```

The `FIND_SUBCKT` config command is used in those cases where you are using a flat netlist (ie. Extracted) as input and you desire to force AccuCore's partitioning algorithm to preserve certain Circuit structures as a single partition. To use this command you must provide AccuCore with a Sample of the circuit structure in the form of a spice netlist.

The syntax for `FIND_SUBCKT` is:

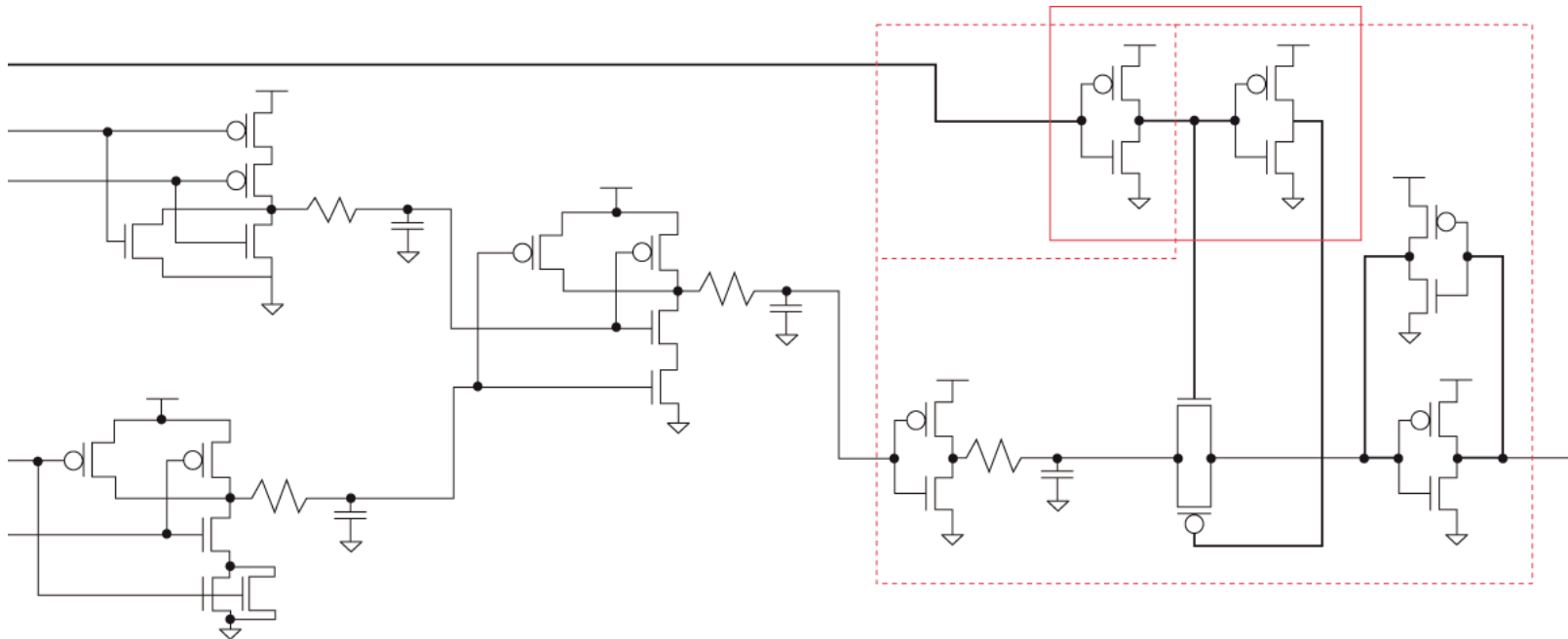
```
FIND_SUBCKT <pattern_name> <pattern_spice_netlist> <powers> <grounds> \  
<inputs> <outputs> <bidirs> <clocks> <optional_table_filename>
```

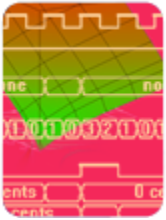
Note: a table file is only necessary when a manual override is desired for the function or vectors



Step 2: Merge Parallel Devices/Propagate Clocks/Identify Latches

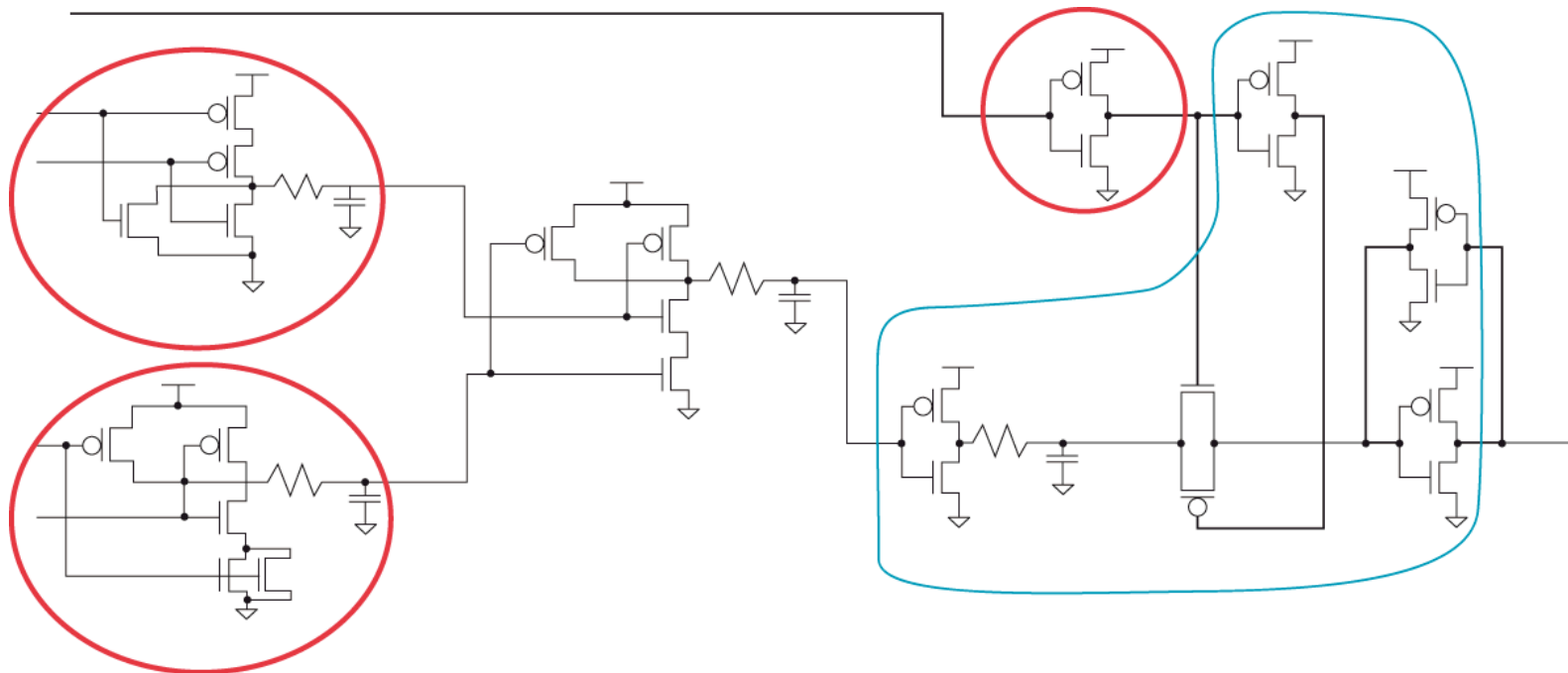
- Strip invalid devices and flatten
- Transform coupling capacitors
- Trace resistor connectivity
- Add power and ground & process device types
- Assign clock attributes
- Merge parallel devices & RCs
- Node/driver static/dynamic conditions
- KEEP_SUBCKT & FIND_SUBCKT
- clock pins propagated
- find inverters
- gating logic
- latching nodes and devices

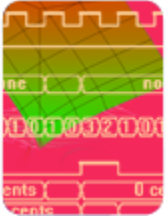




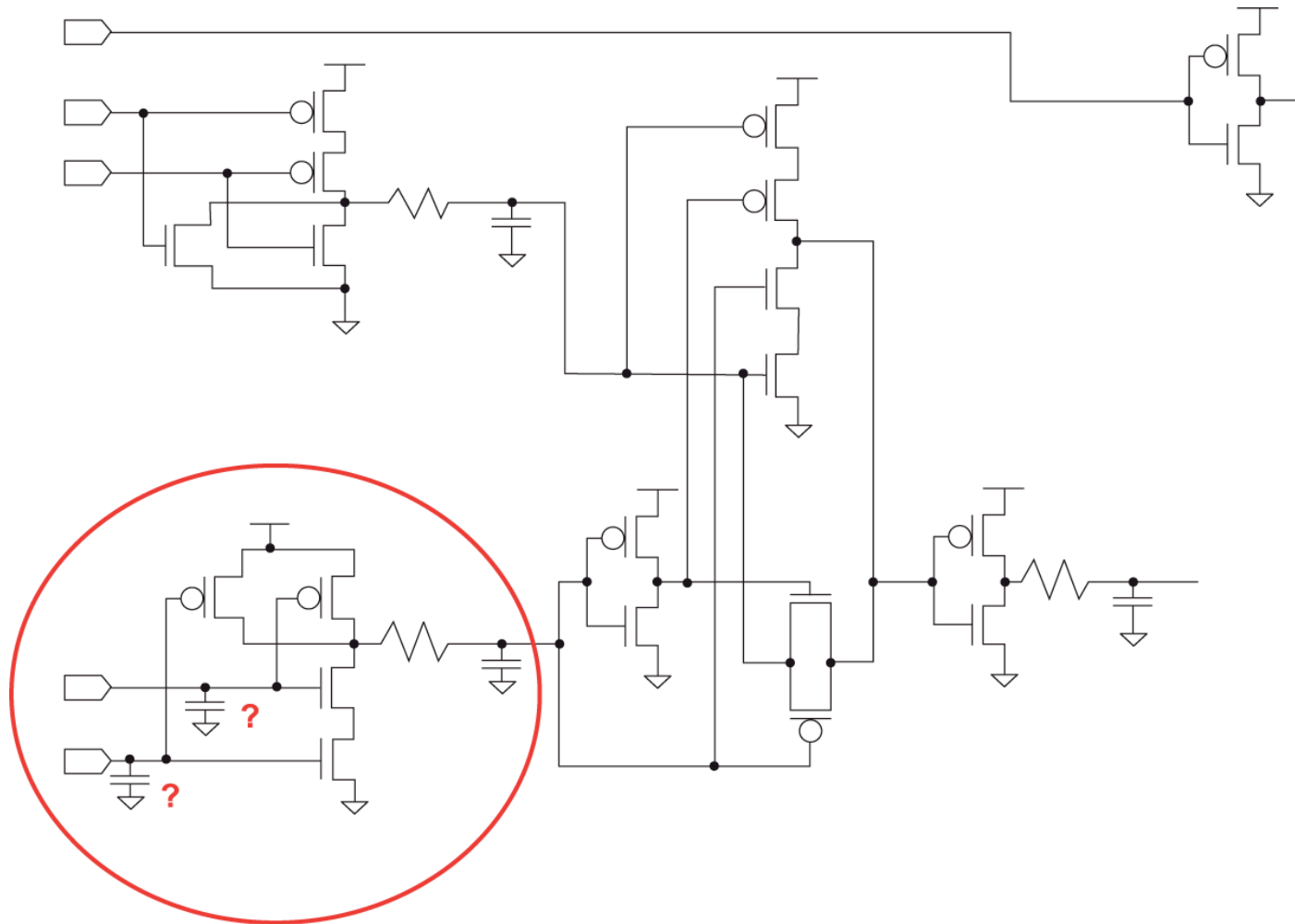
Step 3: Partition the netlist into Design Clusters (DCs)

- Design Clusters (DCs) are partitioned:
 - By Channel Connected Components (CCCs) – a set of nodes and attached transistors that are traced through Source-drain connections
 - By merging tightly coupled connected regions or feedback loop topologies
- Muxes are considered tightly connected regions





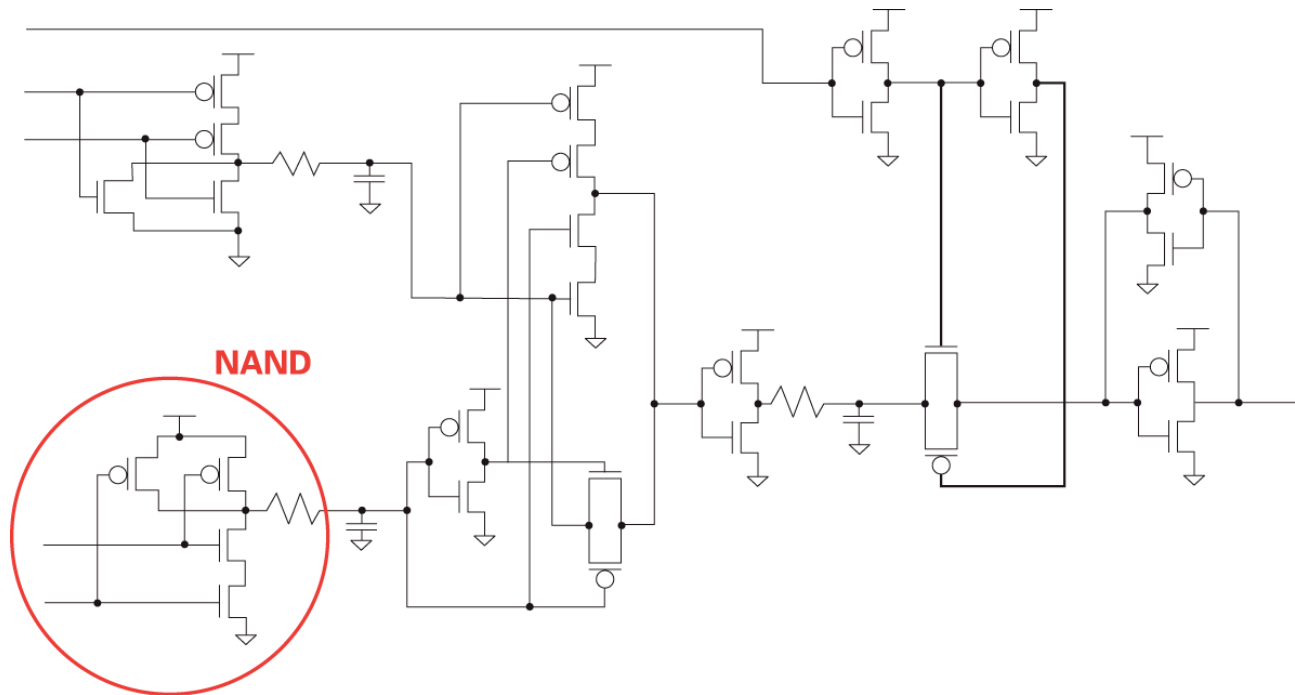
Step 4: Measures Input Pin Caps for Primary Input DCs





Step 5: Extract the DC Function & Generate Optimized Value

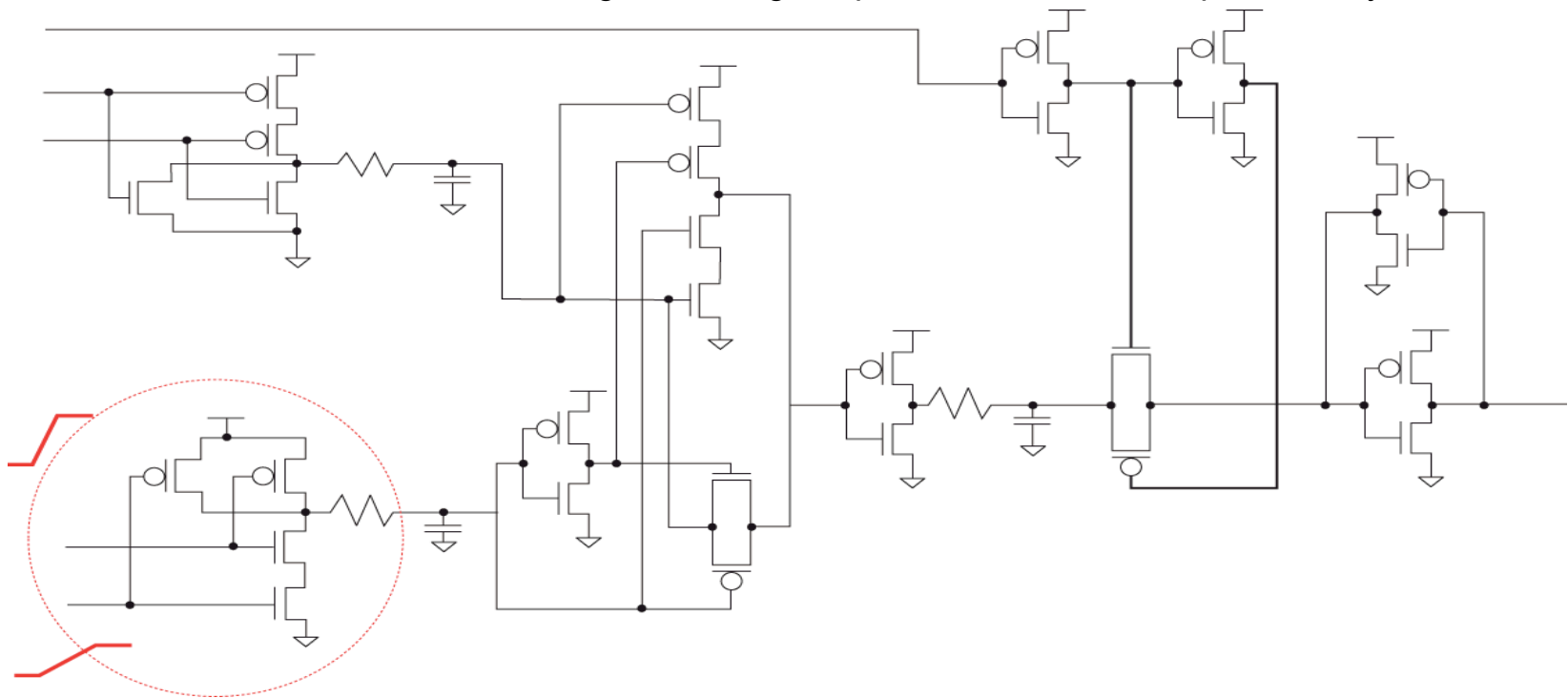
- Simucad uses a proprietary BDD based algorithm to determine the function of a given DC
- Simucad also uses a proprietary algorithm to automatically generate minimum simulation Vectors to characterize the DC
- False Paths are eliminated – Local (DC) false paths are removed if logically impossible

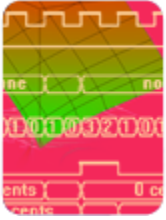




Step 6: Propagate Input Slopes and Actual “in-circuit” Output Loads for the DC to be characterized

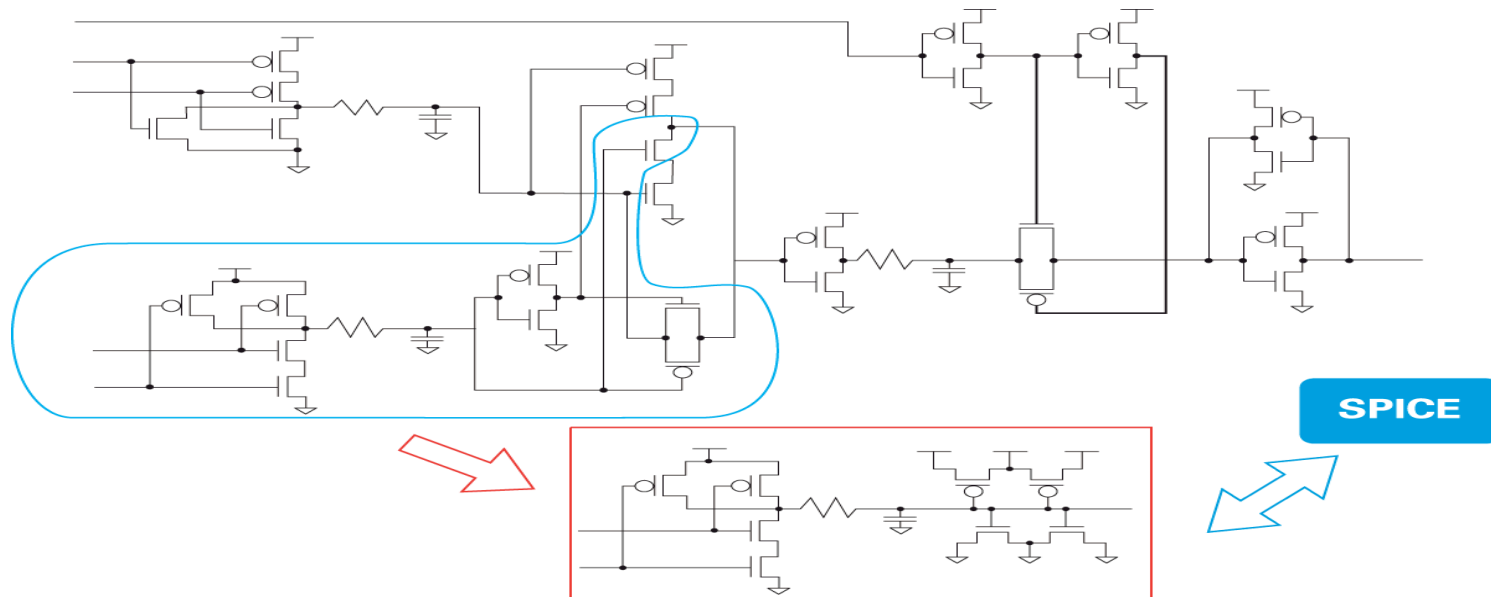
- Characterization requirements and slopes are pushed to the Design Cluster (DC).
- Actual “in-circuit” DC output device load instances are identified and transformed to remove sensitization requirements to ready for dynamic simulation characterization.
- If “folding” has been enabled to bypass simulation based characterization, the DC is tagged to derive its characterization info by lookup table from a prior result stored in the database.
- Determination of this is based on .cfg file settings of parameter tolerances provided by the user.

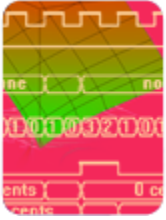




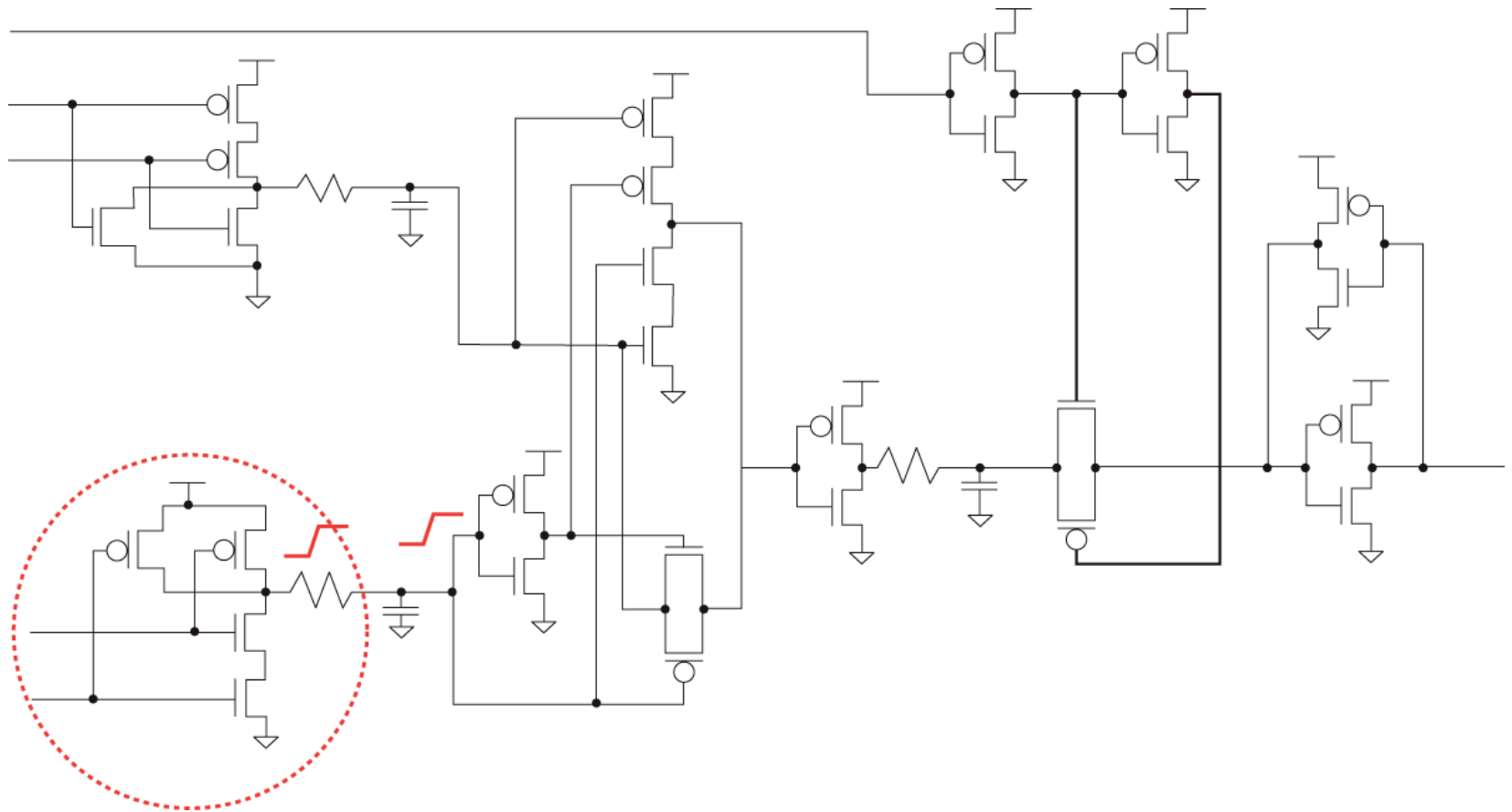
Step 7: Characterize the DC using Dynamic Simulation and Store Results

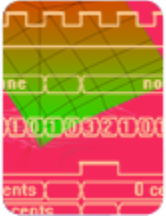
- For DC's not "folded" characterization simulation decks are generated.
- "Cell" delay propagation and transition rate characterization is performed first. setup & hold, and recovery & removal are deferred and scheduled for later processing.
- Link to SPICE simulator and pass the simulation decks. If SmartSpice in API mode - deck is passed directly via memory and both I/O and parsing overhead are skipped and each "related" timing arc simulation for the DC is parameter altered in a similar manner.
- Results to AccuCore firebird database via the API - avoiding ALL file I/O and translation overhead.
- Net result - significant performance gain over non-API and/or third-party external SPICE simulators.





Step 8: Propagate Output Slopes to the input slopes of the next DC to be characterized





Summary

- **Accuracy**
 - Dynamic Simulation
 - Propagation of Slopes Tables throughout Design
- **Easy to use - Setup, Maintenance**
 - Simple .tcl script-based config file
 - Automatic function extraction
 - Automatic Vector generation for Dynamic Simulation runs
 - No manual transistor direction setting
 - Automatic false path removal
- **Supports aggressive design styles**
 - High performance designs - dynamic logic
- **Complex mixed level static timing analysis tool built in**
 - Critical Paths, Sub-critical paths, timing checks, Slack reports
 - SPICE deck creation of Critical paths (ready-to-run in SPICE simulations)
 - Various types of Model generation for hierarchical design and full-chip STA

Quicker timing convergence – Incremental characterization

Reduces Design Cycle

Improves Design Quality