# JDE ARCHITECTURE OVERVIEW

## EXECUTIVE SUMMARY

The goal of the JSPICE Design Environment (JDE) is to create a modern AMS development environment that unifies design generation, simulation, measurement, characterization, data analysis, and AI-assisted engineering into a single extensible environment. JDE defines an open, common set of design, measurement and analysis languages and a common data model for waveform data coming from a simulator. This all enables process-independent designs and simulator-decoupled measurement and analysis scripts that are highly reusable.

Digital design has become highly driven by software standards and abstraction. It has achieved impressive productivity gains and is poised to benefit from the use of AI. Analog and mixed-signal (AMS) design has not. Much of AMS design remains tied to schematics, fragile scripts, manual analysis, and knowledge that exists primarily in the heads of a few experienced engineers. With JDE, analog designers can be at least 10X more productive. Adding AI and building a community may increase that to 100X or more.

| Activity | Digital Design | JDE Analog Design | Analog Design Today |
|---|---|---|---|
| Code | Text-based RTL with generation primitives and highly parameterized | Highly parameterized netlist text with powerful preprocessing | Schematic-based and married quickly to a PDK. |
| Cell based | Yes | Yes | No |
| Auto-routing | Yes | Yes | No (limited) |
| Placement | Auto | With human direction | By hand |
| Test/verification | System Verilog, UMV and highly reusable drivers | Powerful, parameterized test and analysis scripts | Ad hoc test, one-off reports, fragile scripts |
| Process independence | Yes, new cells and STA, parameters | Yes…, but must generate new cells | No, one design, one PDK |
| STA | Aimed at million gate logic | STA with transistor-based models using SPICE | None |

### The system has five central technical ideas:

- An extended SPICE input language and input preprocessor add significant enhancements and are on par with System Verilog.
- A programmable and compilable waveform measurement system rather than a limited set of simulator directives.
- A sweep and optimization engine that coordinates simulations, measurements and distributed execution.
- A Unix-style data analysis pipeline built around a common raw file representation.
- A probe framework that packages complex characterization knowledge into schematic-based electrical checks.

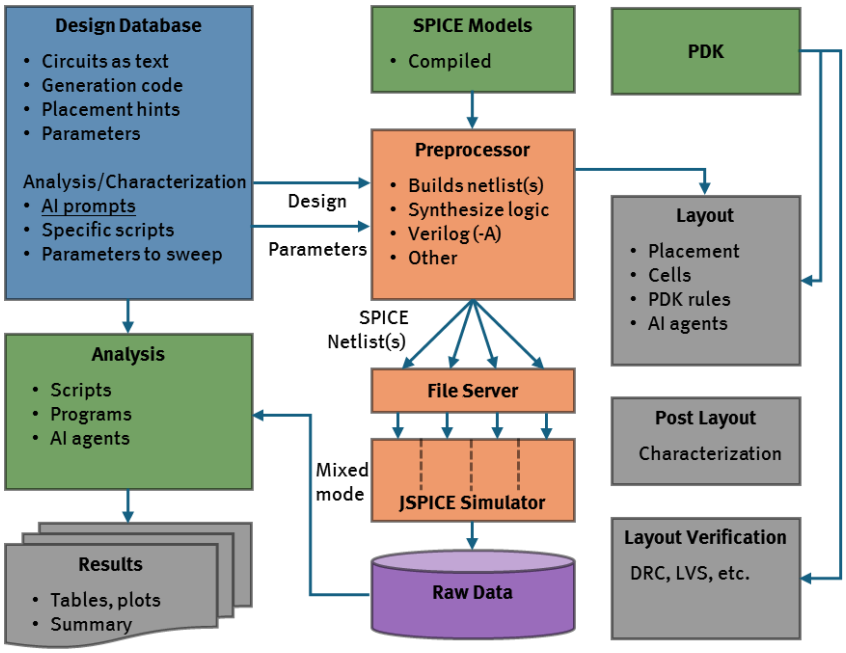
## THE SYSTEM AS A WHOLE

| | |
|---|---|
| 1. Source | Parameterized JSPICE deck, libraries, probe definitions, or measurement source |
| 2. Generation | Input preprocessor expands parameters, C, loops, buses, subcircuits, includes and libraries |
| 3. Simulation | A selected SPICE-like simulator runs the generated conventional deck; 1,000s of decks with different parameter can be automatically created and run in parallel |
| 4. Waveform Conversion | Simulator specific data can be converted to Berkeley SPICE rawfile form |
| 5. Measurement | Compiled or predefined postprocessors analyze waveforms and present results |
| 6. Characterization | Reuseable scripts drive everything needed to do a complete characterization |
| 7. Data reduction | Composable filters calculate, subset, combine, interpolate, normalize and summarize results |
| 8. Presentation | Tables, plots, annotated waveform plots, and probe-based reports expose the conclusions |

JDE: All the information about the design, the analysis and the characterization is in the design database. The design is process independent, while each specific implementation (like 28HPC) is defined by a group of parameters.

The design and parameters are sent to the preprocessor which expands statements, and then adds specific information from the models.

It can sweep parameters and create many netlists, each with a different set of them. Analysis is run on the raw data coming from the simulation. Data reduction programs create reports.



## Input Preprocessor: A Circuit Generator

Normal Berkeley SPICE and HSPICE decks remain legal input, but the JSPICE language adds compile-time programmability. The input language includes deck parameters, typed parameters, embedded C expressions, embedded C statements, multidimensional buses, parameterized subcircuits, preprocessing includes, literal sections, shell command execution, object libraries and post-preprocessing of selected devices.

- Parameters can control values, dimensions, model choices and topology.
- Embedded C provides variables, loops, conditionals, functions, arrays and calculations.
- Bus declarations and bus-aware subcircuits express replicated structures compactly.
- #include recursively preprocesses JSPICE source to create netlists passed to the simulator.
- .literal/.endlit protects simulator specific material from preprocessing.
- .execute allows external data or command output to enter the deck.
- .library and .reference support reusable compiled circuit and model resources.
- Post-preprocessing can modify MOSFET and capacitor statements after expansion.

The key consequence is structural generation: parameters and code can alter not only numerical values but also the number and connectivity of devices. A ring oscillator, gate family, delay line, or array can be described algorithmically instead of copied manually.

## Measurement Languages and Compiled Postprocessors

JSPICE offers several levels of measurement expression. At the lowest level, C code operates directly on waveform data using a measurement function library. Above that is an Extended HSPICE Measurement Language, and above that a Simplified Measurement Language with equivalent capability but lighter syntax. These levels can be mixed.

- Forward, reverse, and cascaded crossing searches.
- Trigger/target measurements, edge counts, averaging over repeated events and conditional searches.
- Arbitrary waveform and scalar arithmetic.
- Results that depend on earlier results.
- Calls into C measurement functions and embedded C statements.
- Operation across transient, DC, AC, and other available analyses.
- Generation of measurement plot data for visual verification.

Programs such as mkmeas, mkspp, evlmeas, and evlsmsr compile or generate postprocessors from measurement source. The compiled postprocessor reads waveform data, calculates results, and can be reused.

## Timing Analysis: Simulation-Based Characterization

JSPICE timing analysis much more than digital STA. It measures actual transistor-level or behavioral waveforms. When used methodically, it is very powerful, and the reason why TCI's first LPDDR5 PHY was 100% correct on first silicon.

- Propagation delay, transition time, pulse width, duty cycle, frequency and voltage levels.
- Setup, hold, flow-through and clock-to-output timing when encoded as latch checks.
- Phase, jitter, decay, statistics, noisy data evaluation and edge-based measurements.
- Measurements over many cycles rather than only a single trigger/target pair.
- Early simulation termination when a postprocessor has acquired enough data.

This is especially suitable for PLLs, DLLs, oscillators, memories, level shifters, clock paths, and other custom circuits where waveform shape and state-dependent behavior make a single timing arc inadequate.

## Sweeps, Optimization and Execution Control

The sweep engine treats characterization as a repeatable, reusable process, not up to each individual designer to create and implement. Parameter files define deck parameters, evaluation parameters, sweep ranges, node lists and measurements. The cross-product of sweep value is simulated unless a different control strategy is selected.

- Linear ranges, single values and explicit value lists.
- Optimization variables and objective/result handling.
- Partial result inspection and repair of incomplete sweep grids.
- Multiprocessing on one machine, JSPICE network servers and LSF queuing.
- Control of retained waveforms and measurement plot data.
- Fault-aware execution and continuation of large campaigns.

## Post-processing and Data Reduction

JSPICE standardizes results around Berkeley SPICE raw files and provides small programs that can be connected in command pipelines. A command pipeline becomes an executable record of how conclusions were produced.

- sptbl: formatted tabular output.
- spsplt and spspltm: result plots and measurement annotated waveform plots.
- spsub: selection of result subsets.
- spcvex: arithmetic and creation of derived variables.
- spstat: statistics, slope, intercept and related summaries.
- spxtr and spint: search, interpolation and resampling.
- spcomb, spmerg, and related tools: combine results or waveforms.
- spfix: make incomplete sweep files usable while a run is in progress.

## Probe-Based Electrical Checks

The probe system packages the rest of JSPICE into reusable schematic checks. A user places a probe instance, connects relevant nodes, selects checks and receives digested results. Experienced users define the underlying behavior in a unified probe definition file.

- STIM, MEAS, SWEEP, ANLYS, PROBE, PROBE_SET, and PROBE_ADD sections.
- Global variables and common stimulus, measurement and analysis sections.
- CDF information connecting schematic symbols to generated netlists.
- Checks that can run existing data, new simulations or parameter sweeps.
- Checks that can launch additional simulations based on earlier results.
- Graphical, tabular, or textual output, including annotated waveforms.

This framework turns engineering methodology into an executable product. A complex latch characterization can be hidden behind a simple symbol while retaining the full flexibility of the underlying languages.

## Network Processing and Practical Scale

JSPICE includes a lightweight network process server and clients for distributing work. Server policy can account for machine availability, time of day and load. The parametric engine can also use multiprocessing and LSF. The purpose is rapid turnaround, not merely throughput: short design characterization iterations are central to the product philosophy.

## Summary: What Makes JDE Special

- It treats netlist generation, simulation, measurement and reporting as one reproducible workflow.
- It permits full programming at both the circuit generation and waveform analysis stages.
- It compiles reusable analysis logic rather than restricting measurement to in-simulator directives.
- It supports expert-facing scripts and novice-facing schematic checks with the same infrastructure.
- It encourages small composable data tools and explicit command records.
- It captures engineering methods, not only simulator syntax.

## About True Circuits

True Circuits, Inc. offers a complete family of standardized, silicon-proven PLL, DLL and DDR PHY hard macros that spans nearly all performance points and features typically requested by ASIC, FPGA and SoC designers. True Circuits utilizes robust state-of-the-art circuits, a methodical and proven design and test strategy, and close associations with the major foundries. True Circuits' PLL, DLL and DDR PHY product portfolio is available in most TSMC, UMC and GLOBALFOUNDRIES processes and process variants from 180nm to 5nm.

Since 1998, True Circuits has distinguished itself as the technology leader in the timing IP space, and its PLLs and DLLs are used extensively around the world in its customers' products with production volumes in the billions.

For more information about JDE, visit www.truecircuits.com/jspice.html.

For discussion about JDE and how to get started, contact Brian Gardner, V.P. of Business Development, bgardner@truecricuits.com.