About ADE

Chip design flow 101

Chip design is a sequence of steps with the ultimate goal being a set of fabrication masks showing where each layer will and won't contain material. High-level overview of analog design steps:

1. Schematic generation. This is an abstract topology containing devices such as transistors, resistors, etc., represented as symbols.
2. Sizing. Each element in the topology has some parameters describing its physical or electrical properties. For example, a resistor's resistance or a transistors width. Using the topology, SPICE simulations (basically big matrix solves) compute the schematic's output characteristics given particular input conditions. The element parameters and topology is modified in an iterative process until the design's goals are met.
3. Layout. The abstract representation is transformed into a set of masks.

ADE is Sizing

Cadence's Virtuoso platform contains tools to do all of the steps above. ADE, or "Analog Design Environment", is the Sizing tool. Sizing is a difficult problem because thousands, or tens of thousands, of simulations must be run and the resulting potentially multi-TB data analyzed. A circuit must be correct not only in the lab but:

- With a low battery in the cold. With a full battery in the heat. And hundreds more combinations in-between.
- When first manufactured and 15 years later after thousands of power-on/power-off cycles and heat affects, like all of a car's microprocessors.
- Resilient to manufacturing processes and be able to manufactured inexpensively.
- Help the designer account for device parasitic effects: try to reduce abstraction wherever possible.

Big Data

Consequently, ADE behaves less like a traditional EDA (Electronic Design Automation, Cadence's field) tool and more like a Big Data tool, having capabilities/requirements such as:

- Parent/child distribution across hundreds of machines using TCP/IP to communicate among them. Data management of potentially TB of data.
- SQL backend database to collect and query GB of simulation data results.
- Careful GUI coding to ensure rapid response to user requests and up-to-date reporting on simulation progress. Result tables having hundreds-of-thousands of cells are common: careful data structures and execution flow is critical.
- We are expanding in multi-threading and other parallel computing and have to consider races and synchronization.
- End-user customizations and scripting support.
- Automated analysis and execution tools using machine-learning and statistical analysis.

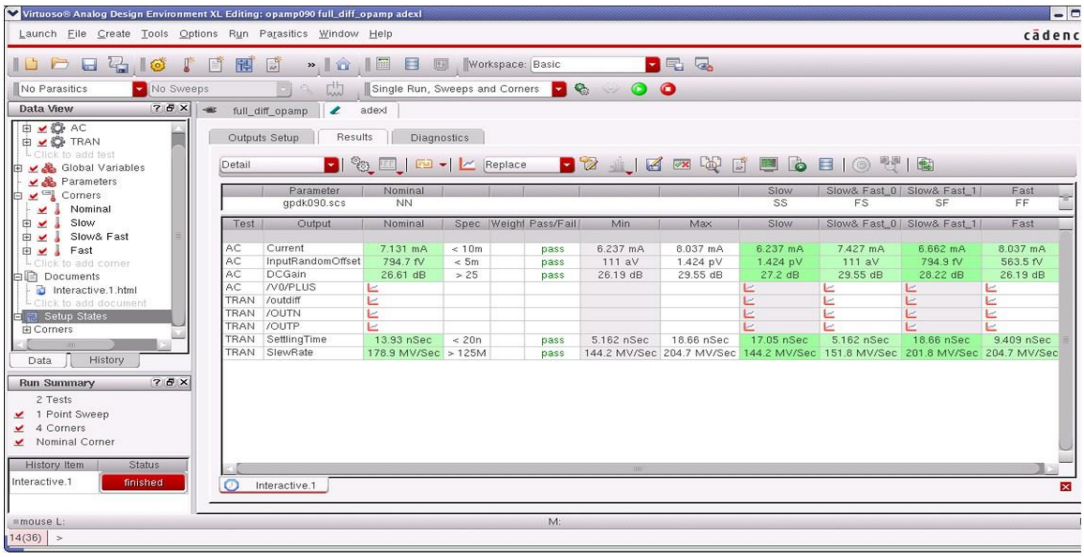
Technical overview

ADE's existed for about 25 years. There are two primary components:

- A user-customizable simulator interface written in SKILL, Cadence's internal Scheme/LISP-based language.
- C++ implementation using the Qt graphical toolkit. We use boost, a C++ toolkit providing memory-management and common data structures. The oldest code is about 10 years old; there's very little "terrible legacy code".

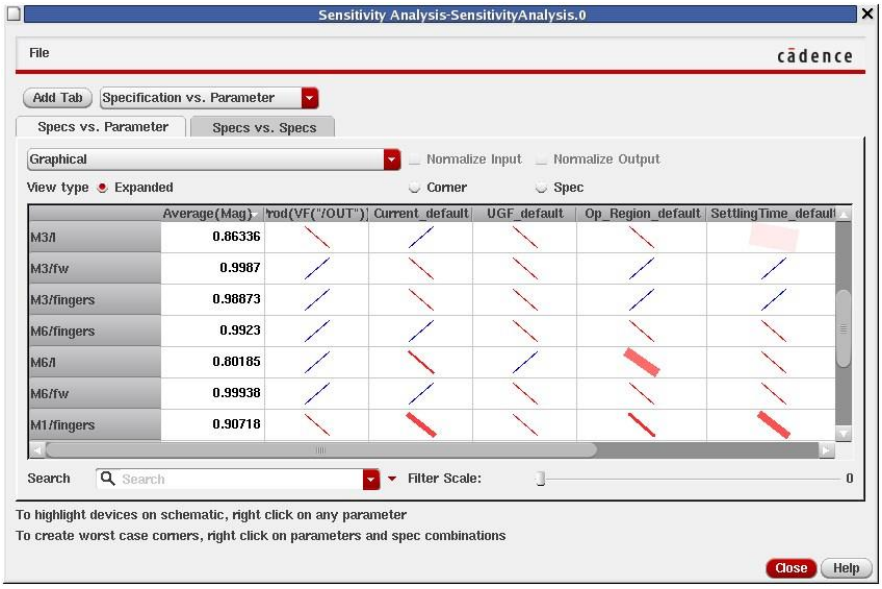
Most senior developers are versed in all of ADE's technologies, such as XML, SQL, Qt, and SKILL, but I am always seeking specialization if a particular area is interesting. EE experience is a plus, but we are a computer-science problem.

Some screenshots

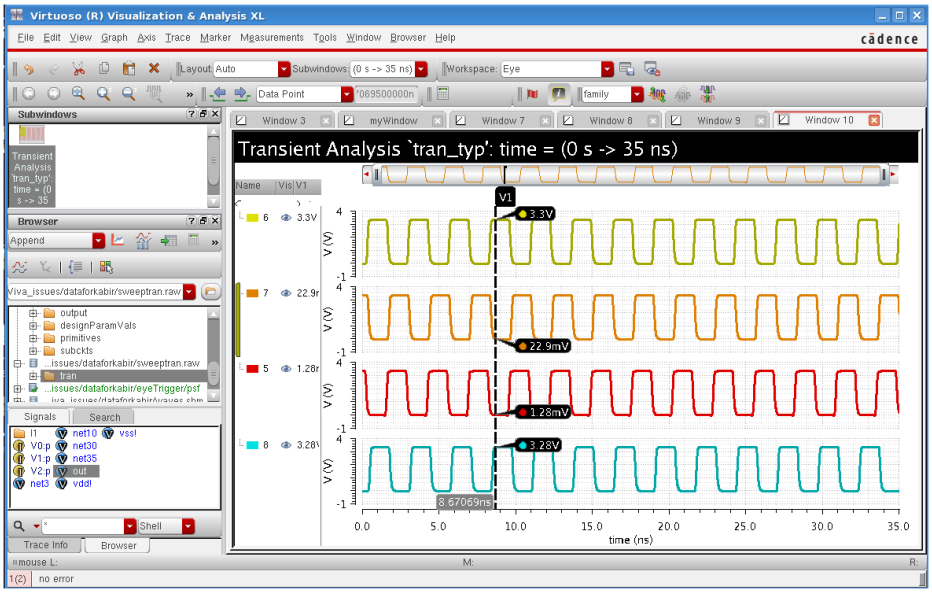


This screenshot shows the "default" view of ADE. The data view tree in the top-left is where the user describes their set up and simulations to be run, using other assistants not shown here. The table in the center shows the simulation results, with cell shading to report that the results have met the user's goals. This is a very small run: real runs will have hundreds of rows and thousands of columns and be distributed across hundreds of machines.

Each button above the big table performs some analysis or post-processing, such as sensitivity analysis:



Or plotting:



Or HTML datasheet creation or run-to-run comparison or various different view capabilities to allow the user to answer their “what-if” questions or assistants tailored to specific debugging tasks.