5.2.: SIMULATOR INTERFACE OF IC-CAP

IC-CAP, with its unique framework concept, always links to separate, standalone simulators. However, to make this concept work independently of the customer's simulator capabilities, three types of simulators are added to the CD-ROM containing IC-CAP:

UCB Spice2g6, which is the base of most customer-specific simulators
the latest version of UCB Spice (University of California, Berkeley)
HPSpice

This means that, provided the user has no local simulator available, he can always perform simulations based on these 3 types of simulators.

On the other hand, IC-CAP can also interface to other types o simulators. Links, included into the IC-CAP Analysis Module, are depicted in fig.1 below. It should be mentioned that these simulators need not necessarily be available on the local IC-CAP workstation, but that also remote simulation is supported (see IC-CAP file $ICCAP_ROOT/lib/iccap/usersimulators).

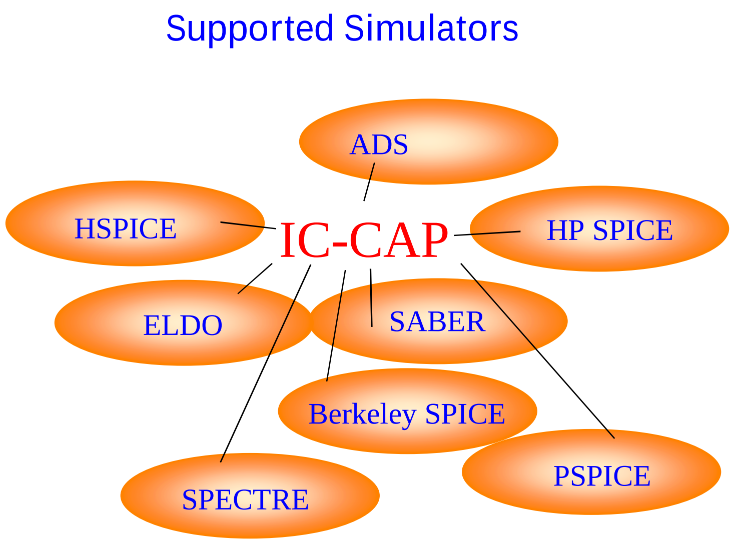

Fig.1: IC-CAP and its links to simulators.

Using IC-CAP's PEL language, or the userC functions, it would be also possible to include the model equations directly into IC-CAP. However, this would imply the following disadvantages:

- if the model equations change, esp. true for user-specific models, this has to be reflected in the IC-CAP model equations.
- if an additional component has to be added to the specified model in IC-CAP, one would have to redefine all model equations. Example: an extra series resistor for a diode model would reduce the voltage drop across the implemented model equations, thus affecting the diode equations.

In order to avoid this dilemma, it is desirable to always use external simulators. In this case, and coming back to the example from above, an additional series resistor is simply added to the simulator sub-circuit. Then, there is no need to modify any model equations inside IC-

CAP. The only modification would be to add a series-resistor parameter extraction module in order to determine the value of this additional parasitic resistor.

Because IC-CAP also supports simulators which allow the user to specify its own proprietary model formulas like ADS (sdd), Saber (template), Eldo (HDLA) etc., it is possible to add extra components also to user-specific models.

With this in mind, it is of particular interest, how IC-CAP interfaces with the simulators. This is explained in fig.2.

Step-by-step:

1. after the component has been measured, the user selects and specifies a suitable sub-circuit in IC-CAP's Circuit description window (fig.3)
2. then, the model parameters are extracted and exported form the IC-CAP transform hierarchy into the Model Parameter window (which, on the other hand, is a consequence of the user-specific Circuit Window), see fig.3 again.
3. When a simulation is demanded by the user (or the optimizer), IC-CAP takes the circuit deck of the Circuit Window, replaces the default values defined there with the actually extracted model parameters from the Parameter List Window, adds the stimulus information of the Setup Inputs and also the requested type of simulation specified with the Setup Output(s) and writes the whole information as a simulator input deck to the /tmp directory.
4. The simulation is executed, locally or remote, using the syntax 'simulator input_deck output_deck'
5. The simulator perform the calculations and exports its simulation results into two files, both located again in the /tmp directory: one ASCII file containing the execution time and the error messages, and a binary file containing the very simulation result.
6. IC-CAP reads the binary file and places the simulation results under the Output icons of the Setup window. The ASCII file is only read if the Simulation Debugger window is open. Otherwise, this file is ignored. (See fig.4)

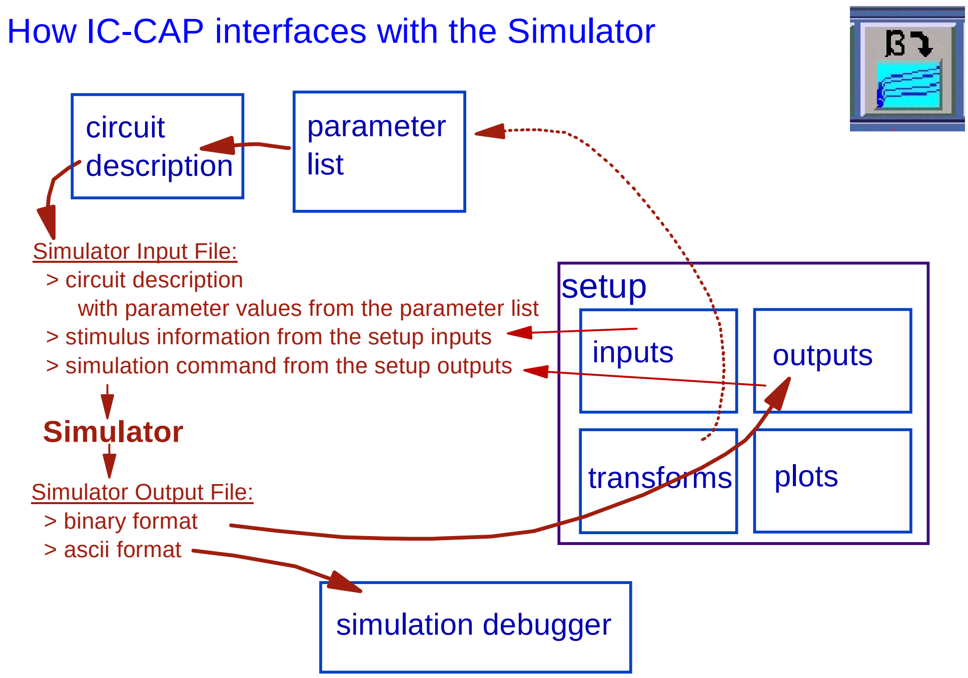

Fig.2: how IC-CAP interfaces with the simulators

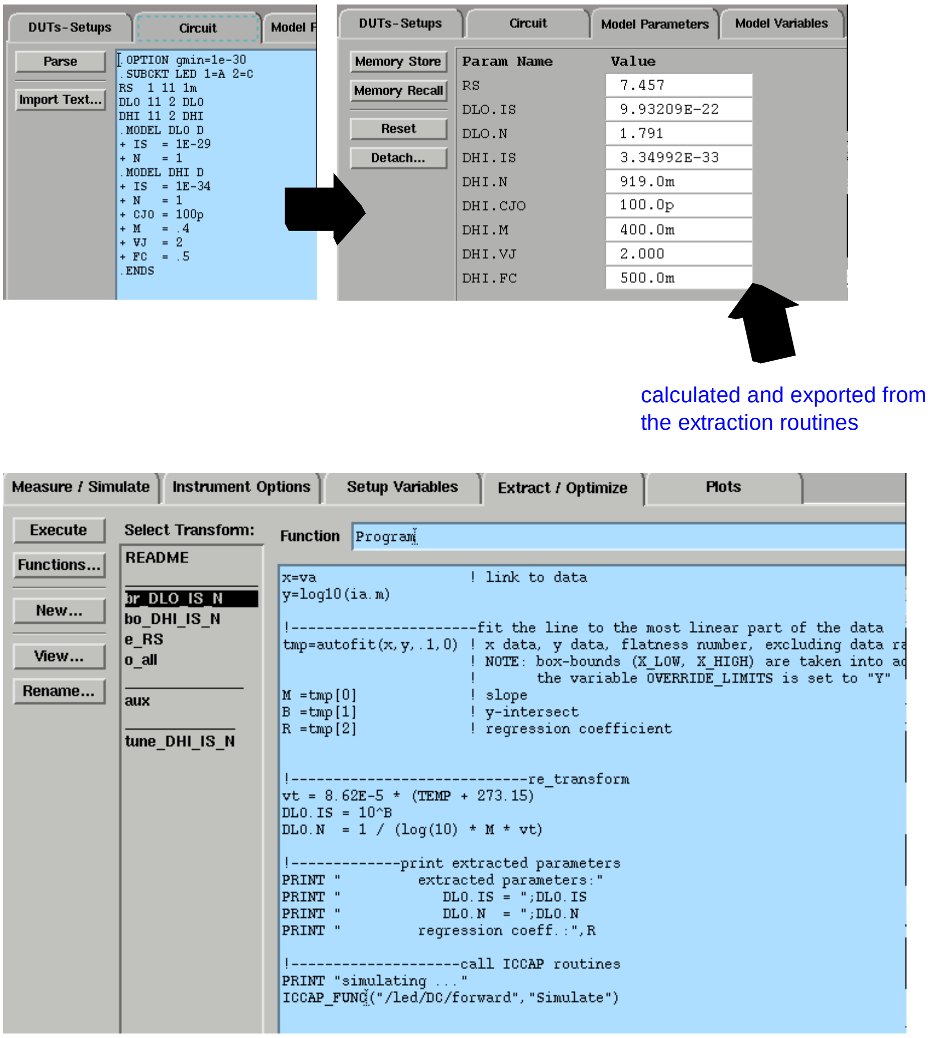

Fig.3: First, a suitable sub-circuit is selected and defined for the simulations. Then, the exacted model parameters are exported into the Model Parameters window. The model parameters showing up in this window are a consequence of the user's sub-circuit definition.

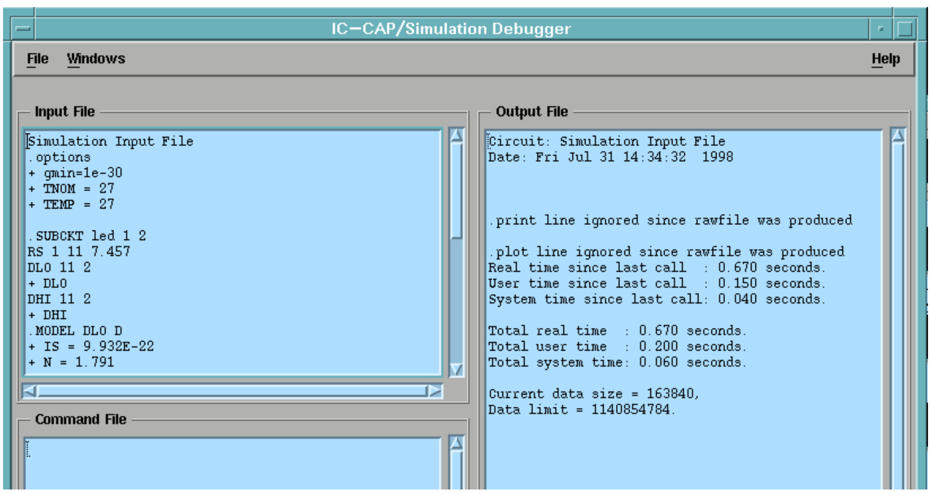

Fig.4: The Simulation Debugger window.