New Features in SmartSpice 2.3.4.C

Stop-Continue Feature

A powerful new feature "Stop-Continue" has been added to ***SmartSpice*** to allow the user to suspend a transient simulation and investigate the output before resuming the simulation run from the suspended state. This allows generated data checks during the simulation run and therefore ensuring relevant simulation data is generated. This feature allows a user to check intermediate simulation results and/or save (if necessary) on the fly. The user can allocate CPU priority to the most important simulation job with a unix system command such as "nice" if multiple simulation jobs are running on the same machine and if a user wants to finish one job as soon as possible.

The Stop-Continue algorithm is activated by adding the ".OPTOINS STOPCONT" line in the input deck to be activated and works in a command mode(smartspice -c) or a command line on New ***SmartSpice*** GUI with some SmartSpice commands:

1. pause: pauses a current simulation.
2. cont: resumes a current paused simulation
3. cancel: stops current simulation.

Note: The command to continue a suspended process is not "continue" but "cont".

The keyword "continue" is reserved for other purpose.

The following message is displayed on the ***SmartSpice*** main window after sourcing an input deck and running it:

--> run

STOP-CONTINUE MECHANISM PROGRESSES...

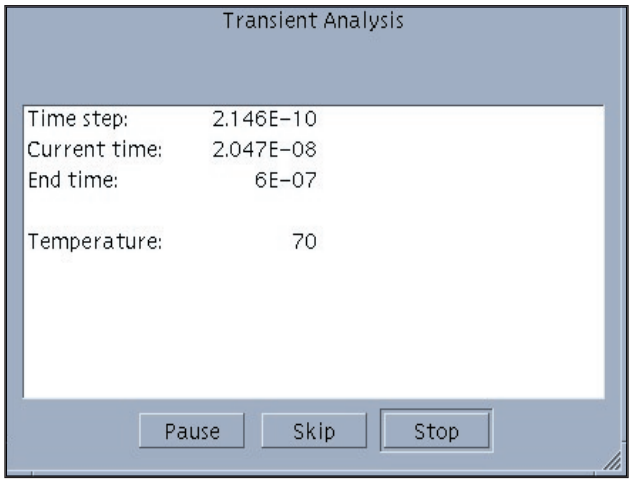

Figure 1. Run time screen with Pause button.

Once the feature is turned on, a user can interactively operate pause/continue/stop using the above three SmartSpice commands, or Pause and Stop buttons on the SmartSpice run time screen(Fig.2). Clicking on Pause and Stop buttons behave as pause and cancel commands, respectively.

In the case of Stop, a ***SmartSpice*** simulation job is immediately terminated and cannot be resumed with the Pause button (it's just an interactive stop operation in ***SmartSpice*** GUI).

The run time window disappears when the Pause button is pressed and the interactive ***SmartSpice*** window is released for a user to do various kinds of post processing operations as if the simulation was completed. A user can now check or save an intermediate simulation result and after the check is done, to resume the simulation, type a command 'cont' (Figure 2) and the run time dialog will be displayed on the screen again.

Note : Not all ***SmartSpice*** commands are available. Some commands cannot be issued in the Stop-Continue algorithm including in conjunction with a multi-thread mode as described below, and such commands can be checked by issuing a special help command "mt_notsafe_com".

The option ".OPTIONS STOPCONT" also enables ***SmartSpice*** to split a transient simulation job into two modes - foreground and background operations. This means that a simulation job is running in a background mode and post-processing, such as .print or .measure processes in a foreground. This foreground operation is also available for a command line (Figure 2) or interactive ***SmartSpice*** GUI operations without a Pause command as described above.

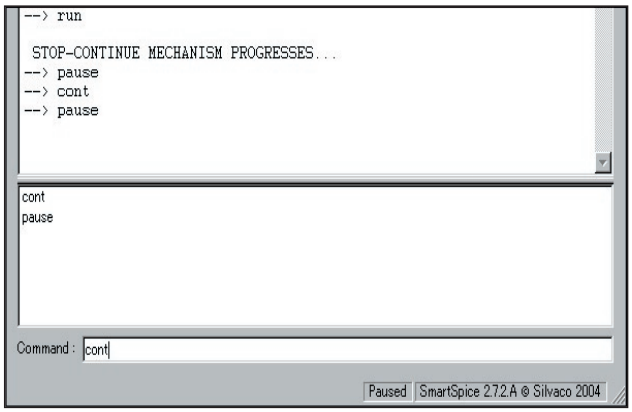

Figure 2. SmartSpice command line Windows.

The option enables not only the Stop-Continue feature but also the foreground operation on the fly but we recommend a user to pause a simulation when he or she wants to operate the post-processing. The reason is that the CPU sometimes might be loaded for such post-processing operations and the simulation itself might result in taking more time than the user expects.

This Stop-Continue feature can also function in a multi-thread mode, but due to the same reason, one extra processor is recommended to allocate post-processing, e.g. 4 CPU machine, for "smartspice -P 2 or 3" and one remaining processor can be used for the post-processing in this case.

A New SmartSpice Variable for Safe Mode

Nowadays, high performance computing systems are available at a very low price, and even a server class PC environment is recently replacing conventional and expensive EWSs by a high cost performance PC systems equipped with a huge hard drive and physical memory, such as Windows and Linux platforms.

However, we could also say that some application software consume a lot of system resources at the same time, and such software might sometimes cause the system to crash when they are simultaneously executed on a same machine.

SmartSpice supports a new feature "Safe Mode" to guard your simulation job from such unexpected risk.

The features is turned on by adding one ***SmartSpice*** variable "set safemode=true" into your .SmartSpice.ini and it monitors system resources during a simulation. If either memory or hard disk space is getting exhausted(the limit is set to 50MB), ***SmartSpice*** notifies a user through a pop-up window(fig.3) and outputs the current status of system resources in order for ***SmartSpice*** to safely continue a current simulation job on the fly.

The following warning messages illustrate when .options RAWPTS=N is specified, which stores data at into a raw file every specified transient data points N and reached the disk space limit 50MB(below 50MB) during a simulation:

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Insufficient disk space.

The job has been terminated.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Total physical memory size:	267304960 bytes
Available physical memory size	22835200 bytes
Total virtual memory size:	1058312192 bytes
Available virtual memory size:	598319104 bytes
Available disk size:	17571840 bytes

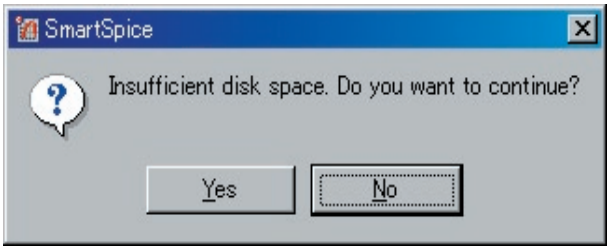

Figure 3. Pop-up window for system resources warning

Thanks to the new feature, it's still time to save the current simulation. A user now has to free up disk space by deleting some unnecessary files on the HDD and continue the simulation answering "Yes" after securing enough disk space. Likewise, memory resources are also being monitored during the simulation.

Decending Sub-Circuit Paths in *SmartSpice*

With parasitic elements becoming more important in circuit performance and characterization it is important that the ***SmartSpice*** deck is constructed in the right way. This then is a short guide to what needs to be specified.

Input Deck Structure

The SPICE input deck must contain circuit heirachy and this must be evaluated first to provide a structure reference on which the path name is mapped (i.e. like a look-up table reference for the X call mapping). For example if you use "." as the sub-circuit delimiter you can have say a capacitor connected between nodes "x1.xnext.xlast" and ground (statement line in deck c1 x1.xnext.xlast 0 10pf).

If this occours at the start of the input deck then there is no reference heirachy and the node name at the top level is the string "x1.xnext.xlast". ***SmartSpice*** allows a large muixture of characters to be used as a node name for customer convienience but for clarity and debugging it is best to reserve some syntax for heirachical traversing only.

So you can specify the path to the connections of a lumped circuit element but in extracting the parasitics it may be necessary to break the element like a transmition line up into a distributed representation. In this case ***Hipex*** will create a sequence of internal nodes of the form "~1", "~2" …….. "~N" to allow the internal breakup of the element.

SPICE has a reference "#1" say to this node but the user is not allowed to connect to these internal device nodes of a distributed element.

The default sub-circuit delimiter is a period "." But this can be changed by the use of the variable of a variable definition (subckt_delimiter=":")