Introduction to VA-Debugger

Introduction

In recent years, Verilog-AMS hardware description language (Verilog-A) has been widely used in analog and mixed-signal design. Correspondingly, most EDA vendors provide simulation tools for Verilog-A. As one of those vendors, SIMUCAD also added support for Verilog-A in SmartSpice several years ago. Recently, a Verilog-A debugger (VA-Debugger) has been developed and added as a component of SmartSpice. VA-Debugger is a convenient and powerful tool for debugging Verilog-A source code during simulation. It can help a designer to locate a design issue quickly and greatly improve efficiency. In this note, several important aspects of VA-Debugger will be introduced.

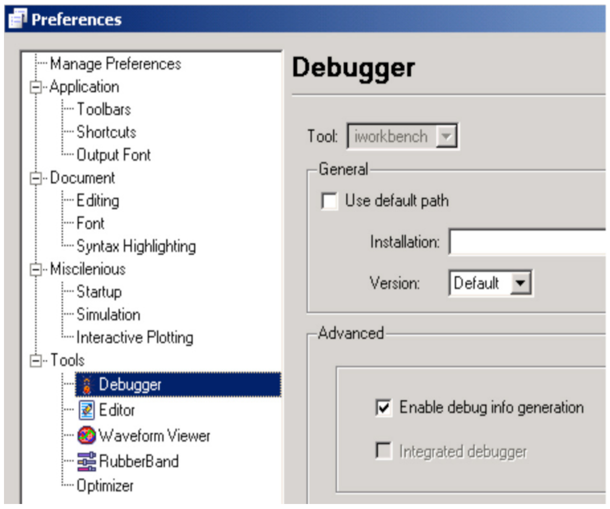
Steps to Invoke VA-Debugger

As a component of SmartSpice, VA-Debugger can't be invoked directly. Several steps must be followed to run the debugger:

1) Verilog-A option "-debug" needs to be set for the Verilog-A file. There are three ways to set it:
   a) in the netlist file with a .options command:
      .options veriloga-args="-debug"
   b) in the netlist with a .verilog command:
      .verilog "<file-name-string>" –debug
   c) set environment variable SIMUCAD_VERILOGA_ARGS to '-debug'. Example under UNIX:
      %setenv SIMUCAD_VERILOGA_ARGS -debug

By setting this option, a Verilog-A source file will be processed without optimization, and the necessary debug information will be generated.

2) In a SmartSpice window, choose the pull-down menu Edit->Preferences and check the box "Enable debug info generation" as shown in Figure 1.



Debugger

Tool: iworkbench

General

☐ Use default path

Installation:

Version: Default

Advanced

☑ Enable debug info generation

☐ Integrated debugger

Figure 1. Set preference for VA-Debugger.

This preference setting will be saved when you exit SmartSpice and will be restored when you run SmartSpice again.

3) After the input deck is loaded, click the 'Debug…' button under 'Analysis' menu as shown below to run simulation with Verilog-A debugger

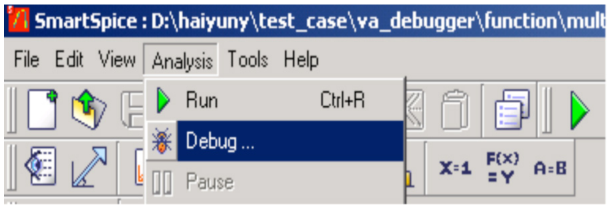
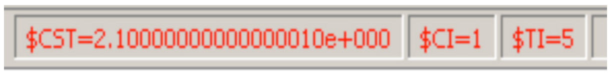

SmartSpice : D:\haiyuny\test_case\va_debugger\function\mult
File Edit View Analysis Tools Help
Run Ctrl+R
Debug ...
Pause
X=1
F(X)=Y
A=B

Figure 2. Menu to run VA-Debugger.

However, if no Verilog-A source file in the current input deck is processed in "-debug" mode as specified in step (1) VA-Debugger will not be invoked and the simulation will run in optimization mode since debug information is not generated for Verilog-A source files.

Status Information Description

At the bottom of the VA-Debugger window there are a few status bars. Three of those bars have special meanings.



$CST=2.10000000000000010e+000 $CI=1 $TI=5

Figure 3. Status bar.

$CST:	Current simulation time
$CI:	Iterations in current simulation time
$TI:	Total iterations

If any value of those variables changes, it will be highlighted in red, otherwise it will be displayed in black. This design helps users to monitor the progress of the simulation while paying minimal attention to actual numbers.

It is worth mentioning how $TI is updated in VA-Debugger. This variable counts only effective simulation iterations and it only updates once after simulation finishes in one time spot. This means that if the simulation can't converge at some time spot after several iterations, these iterations will not be counted in $TI. The value of $CI updates with every iteration, no matter if the simulation converges or not. Therefore, the value of $TI does not synchronize with $CI, which is normal here.

Set Breakpoint

Like every debugger, VA-Debugger provides several ways to set and configure a breakpoint. The most popular and convenient way to set a breakpoint is left-clicking the mouse in front of a source code line. A "stop" mark will show in front of that line if a breakpoint is successfully set. If a line doesn't contain debug information, like port and variable declarations, a breakpoint can't be set at that line.

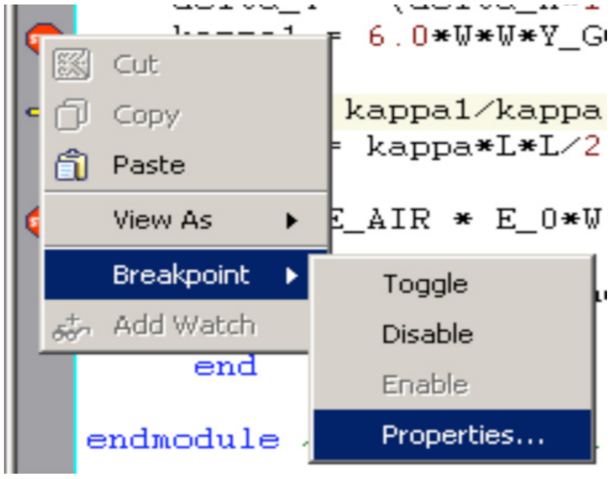
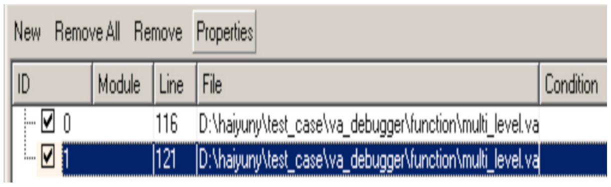


Figure 4. One way to invoke breakpoint setting dialog window.

Conditional breakpoints are very useful in debugging and VA-Debugger also supports these. There are two ways to open the dialog box to set a conditional breakpoint. One is to right-click on an existing breakpoint and choose Breakpoint->Properties from the menu.

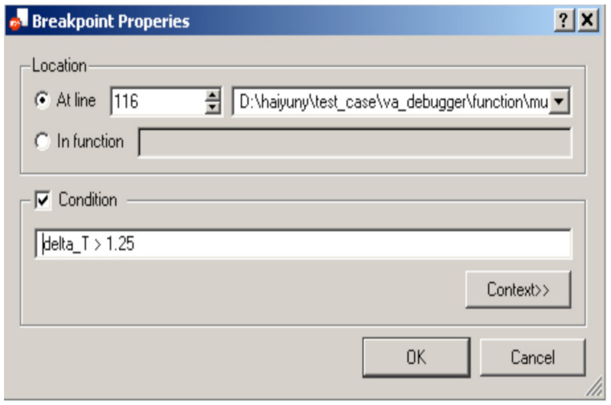
Another way is to choose a breakpoint from the breakpoint list in breakpoint window and clicking the "properties" button to get the dialog box.



ID	Module	Line	File	Condition
☑ 0		116	D:\haiyuny\test_case\va_debugger\function\multi_level.va	
☑ 1		121	D:\haiyuny\test_case\va_debugger\function\multi_level.va	

Figure 5. Another way to get a dialog box for setting a breakpoint.

In the dialog box, there is a "Condition" field, where a conditional expression can be typed in. The conditional expression must be a logical expression that can be evaluated into a Boolean value. If the conditional expression can't be evaluated correctly (wrong expression type, unknown symbol in expression, etc.), this breakpoint will be removed and a message will be displayed in the "Input/Output" window.



(●) At line: 116 — D:\haiyuny\test_case\va_debugger\function\mu

( ) In function:

☑ Condition: delta_T > 1.25

Figure 6. Dialog box for setting a Breakpoint.

To help users to compose conditional expressions more easily, a "context" window is provided inside the breakpoint setting dialog box. The "context" window can be expand/hide by clicking the "context" button in the breakpoint setting dialog window.

The context window displays a list of variables and their values. The variable name or value can be easily inserted into the conditional expression field by right-clicking on a variable and choosing "Insert name" or "Insert value" from the menu. This design saves the work of copying and pasting and avoids switching between windows.

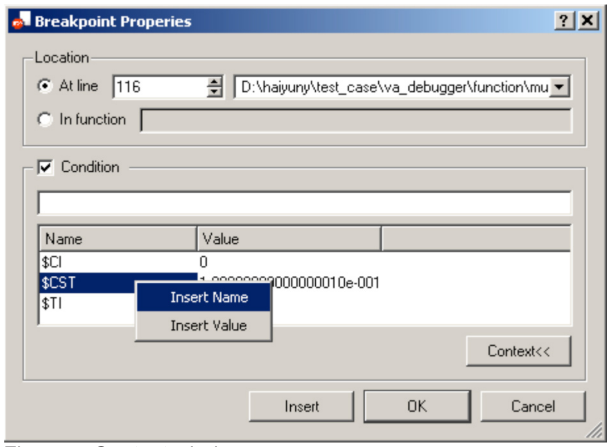


Figure 7. Context window.

Conclusion

In this application note, VA-Debugger is introduced by highlighting several features. The first section lists three steps the user needs to follow to correctly run VA-Debugger from SmartSpice. The second section describes three special variables in the status bar and explains how their values are updated. Finally, how to set a breakpoint, especially how to set a conditional breakpoint, is presented.