AMS Designer Migration, Usability, and Performance Improvements

Cadence Design Systems
Indu Jain, Tina Najibi, Lei Song, Junwei Hou,
Vijay Setia, Poonam Singhal

cadence™

Presented at

cadence designer network

cdn LIVE™

Silicon Valley 2007

AMS Designer from SpectreVerilog: Migration and Usability Improvements

Indu Jain Tina Najibi Lei Song Junwei Hou Vijay Setia Poonam Singhal

ABSTRACT

Virtuoso ® AMS Designer (AMSD) contains a wide breadth of state of the art features exceeding those of other solutions such as SpectreVerilog. In spite of this, many customers continue to use SpectreVerilog, as the migration path from SpectreVerilog to AMSD contains a number of barriers and ease of use issues. This paper will discuss the solutions to barriers such as PDK conversion, design modifications and verilog text usage. Additionally, this session will present performance gains as well as usability and debugability improvements. The introduction of a new netlister in the AMSD flow (the OSS netlister) and a new simulator flow makes these solutions possible. Additionally there have been key performance improvements in the AMSD simulator itself. The final section of the paper will walk the participants through the greatly simplified migration process and will present the specific performance improvements compared to SpectreVerilog. This new methodology will enable customers to quickly move to the feature and language rich AMSD with minimal effort, thereby greatly increasing designer productivity. Depending on a number of factors, defined in the paper, the migration to AMSD can be cut down from a month to less than a day. Designer productivity is further bolstered by the improvements in usability, debugability and performance. All of these will be presented in the paper along with a performance comparison.

INTRODUCTION

SpectreVerilog has been widely used since the early 1990s and for the most part, it performs the tasks an Analog designer needs. However, SpectreVerilog is starting to run out of steam. SpectreVerilog does not support the new languages such as Verilog-AMS and VHDL-AMS and does not support key functionality such as bi-directional ports. Verilog-XL does not support the latest Verilog 2001 standard, which is becoming critical as more and more digital IP adheres to that standard. Furthermore, whereas in the 90s almost all mixed-signal design was created by Analog designers in the Virtuoso ® Analog Design Environment (ADE), today an increasing number of digital designers and verification engineers are required to run mixed-signal full chip simulations on the command line. This is an area where SpectreVerilog is rarely, if ever, used and in fact was not created to meet this need.

AMS Designer (AMSD), on the other hand, was created to meet the demands of full chip simulations on the command line as well as the demands of Analog designers in ADE. AMSD is a feature and language rich simulator, built on the NC technology using the latest language standards.

So why are some customers still using SpectreVerilog and not the acknowledged better solution, AMSD? There are a number of migration barriers that makes it difficult for a designer to migrate to AMSD. This paper will walk through the issues and solutions. The paper will also discuss another important topic – usability and debugability and how these issues are solved. In the final section an example will be presented using SpectreVerilog and then AMSD, demonstrating the greatly simplified path to using AMSD.

PDK ISSUES

The first task a new AMSD user faces is PDK conversion. AMSD's cell based netlister (amsdirect) requires that specific AMSD information (ams simInfos and ams specific netlist procedures) be created in the PDK. A conversion toolbox is available to convert the spectre views in a PDK to ams simInfos. For the most part this tool works fine, although there could be cases that require manual intervention. All spectre-specific netlist procedures need to be manually rewritten for AMSD. This can take time, especially if the original author of the spectre netlist procedures is no longer available.

Debugging the ams simInfos and netlist procedures can also be time consuming, especially if the issue is not discovered until the elaborator or simulator either fails or gets incorrect results. Once the ams simInfos and netlist procedures have been added and debugged, the designer is ready to use AMSD. However, the amount of time to complete the PDK conversion can often take days to rarely months depending on a number of factors ranging from the number of custom netlist procedures that need to be written to something as simple as lack of write access to the PDK. The PDK conversion step can therefore be a migration barrier for some customers.

To address this issue, an OSS netlister for AMSD was created. The OSS netlister uses the standard spectre views and netlist procedures and is the same netlister used for Virtuoso ® Spectre ® Circuit Simulator, Virtuoso ® UltraSim Full-chip Simulator or SpectreVerilog.. If the PDK works for Spectre or SpectreVerilog, it will work for AMS. The OSS netlister completely eliminates the PDK conversion step thereby eliminating much of the initial setup and debug time.

LIBRARY SETUP ISSUES

The AMSD cell based netlister creates the netlist for each cell within the 5X library structure. Often, the library is read only and the designer can not write into it. This can be resolved by using temporary (explicit TMP, AllLibs, implicit TMP) directories. Using explicit TMP or AllLibs requires modification of the cds.lib file and is another step needed for AMS that is not needed for SpectreVerilog.

The OSS netlister does not write into the library, instead the netlist is written as one file in the netlist directory, and therefore does not need TMP directories of any kind. This results in one less minor setup step.

NETLISTING DIFFERENCES

In addition to PDK conversion and minor changes to the cds.lib to account for TMP directories, there are a few netlisting differences that the designer may need to resolve when using the cell based netlister for AMSD. The first difference is the config itself. A designer needs to create a new config for AMSD, as the SpectreVerilog config will not work for AMSD with the cell based netlister. This issue is solved with OSS, as the same SpectreVerilog config works for AMSD without requiring any changes.

A second difference is in the netlisted line itself. The Spectre netlister ignores the CDF parameter type. Regardless of whether a parameter type is string or float, it is printed as a number without quotes. The AMSD cell based netlister prints the values according to the CDF parameter type. Floats are printed without quotes and strings are printed with quotes. This presents another migration issue as many CDF parameter types are strings, which works for Spectre, but then fails for AMS, as the cell based netlister prints these numbers in quotes. This problem has two issues. The first issue is the fact that there may be no warning when this happens. The AMSD simulator can take the value of the quoted string to be the parameter value. For example, depending on how it is used, a quoted zero (“0”) has the value of 560. So the AMSD simulator could use the value of 560 when the designer intended the value to be 0. Depending on where that value is used, this can result in hard to debug, incorrect results. The second issue is the task of correcting all the parameter types once the problem is discovered. This problem is fixed with the OSS netlister, in that it will not place quotes around numbers, ignoring the CDF parameter type, as does the Spectre netlister.

The goal of the OSS netlister is to ensure that the same PDK, cds.lib, config, CDF and design that works for Spectre or SpectreVerilog will work for AMS without any design or library changes.

INCLUDING VERILOG

Including Verilog text libraries or directories is another area that makes it difficult to migrate to AMSD. Verilog is included in the SpectreVerilog flow using the library file (-v) and library directory (-y) options. This works fine with Verilog-XL, but does not work efficiently with the ncvlog, ncelab and ncsim (3-step) flow. There are several ways in which text can be included in the 3-step flow, but these differ with how text is included in the SpectreVerilog flow.

This problem is solved with the integration of the NC-Verilog ® Simulator (1 step) that works with

the OSS netlister. Ncverilog uses the –y/-v options as does SpectreVerilog.

DEBUGGING

When everything works, it usually does not matter where the files are located or what format they are in. It doesn't matter if there is one netlist file or hundreds of netlist files. However, when there is an problem, and the designer feels he needs to look at the netlist file to debug that problem, these factors all become important.

With the cell based netlister, the actual netlists (one netlist per cell) (vams files) are created within the lib/cell/view structure itself. If something goes wrong, there is no 'one place' to check the netlist file. This is somewhat made easier in ADE with the fact that all these separate netlist files are concatenated into one file – however, this is not the file that is actually simulated and it may contain name clashes making it a little harder to go through if there is an issue.

With the OSS netlister, the netlist is created in essentially one main file in the netlist directory. (VerilogA files are included via ahdl_include, similar to Spectre, and VHDL files are in a separate file in the netlist directory as well.) This makes debugging from files almost as simple as it is for Spectre – as there is essentially one place to look for issues.

The file that can be used to re-run the simulation on the command line (runSimulation) is also considerably simpler, with just one NC-Verilog command followed by the command line arguments. This makes it much easier to see what is used in a given simulation, as well as makes it easier to run from the command line. It is now considerably simpler to just 'vi' the netlist.vams file, make a simple change, and rerun the simulation on the command line in order to quickly try out a change or a fix or hand off the design to a designer who will run on the command line (digital, verification).

ERROR MESSAGES

In addition to an easier way to view the netlist file and rerun the simulation, error messages are an important aspect of debugability. There are several areas of improvement related to error messages.

First, there is a concentrated effort within the AMSD simulator itself to improve error messages to more clearly explain the issue and what can be done to fix it. Second, ADE now contains an 'Error Explanation' GUI. This GUI will provide additional information for most error mnemonics. For a group of mnemonics more information for an ADE designer is presented followed by the 'nchelp' output. For other mnemonics, the 'nchelp' output is provided. Along the same lines, another benefit of the OSS netlister is the fact that since it is the same netlister that is used for SpectreVerilog, the messages that it prints should be familiar.

PERFORMANCE IMPROVEMENTS

AMSD uses Verilog-AMS connect modules for more flexible and accurate analog-digital interface modeling. The Verilog-AMS connect modules in previous AMSD releases are often slower than the SpectreVerilog a2d and d2a built-in devices. This usually affects high level designs that have a relatively high number of A/D interface elements, and the analog models are mainly in Verilog-A. In such worst cases AMSD could be 3X slower than SpectreVerilog.

AMSD is now enhanced with the following features for better performance:

1. Fast cross: This approach reduces unnecessary analog time steps thereby speeding up the whole simulation.
2. C-interface Verilog-AMS connect modules: In AMSD, C-interface built-in functions are automatically generated from the Verilog-AMS connect module description. This approach retains the advantage of Verilog-AMS modeling of connect modules and matches the speed of build-in devices.
3. Verilog-A Compiled-C model: The Verilog-A modules in the ahdl_include flow are compiled into C code devices by the Verilog-A model compiler in AMSD as well as in Spectre and UltraSim, and this greatly improves the speed of Verilog-A model computation in simulation.

The fast cross approach achieves about a 3X speed up in some AMS customer test cases in the

AMS-Ultra flow when bi-directional connect modules are used and when there are high frequency signals at the D/A boundary. For the worst case examples where AMSD was slower by 3X, now AMSD is on par with or faster than SpectreVerilog.

TESTCASE EXAMPLE

In order to demonstrate how easy it is to run a previous mixed-signal design (run in UltraSimVerilog) in AMSD, an example PLL will be used in this paper to show the necessary migration steps. This PLL has a 25 M input signal and a 160 M output signal. The design has 305 mosfets, 97 resistors, 35 capacitors and more than 30 behavioral modules. (Fig. 1)

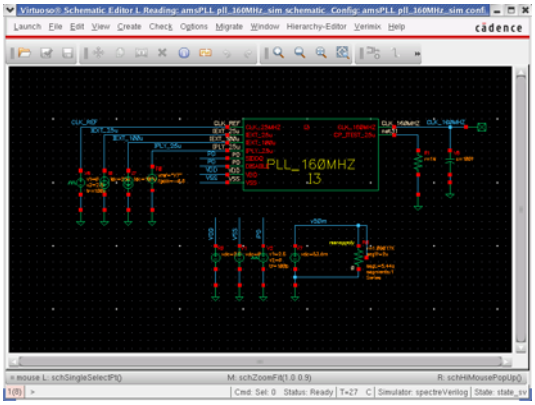

Fig. 1 Top Testbench

Figure 2 shows more details about the PLL block: the PLL_160MHZ_PDIV(I23) outputs a 5 MHz reference signal for the loop. PLL_160MHZ_MDIV(I24) outputs a 160 MHz signal and a 5 MHz feedback signal for PLL_FPD. When the two PD input signals are out-of-sync, the PD generates corrective pulses (UP, DN) to adjust the charge pump output voltages (vCNTL), which controls the frequency of the VCO. Whenever the PLL is locked, the FBCLK and 5MHZ_CLK signals are in phase and the VCO control signals v(vCNTL) are stable.

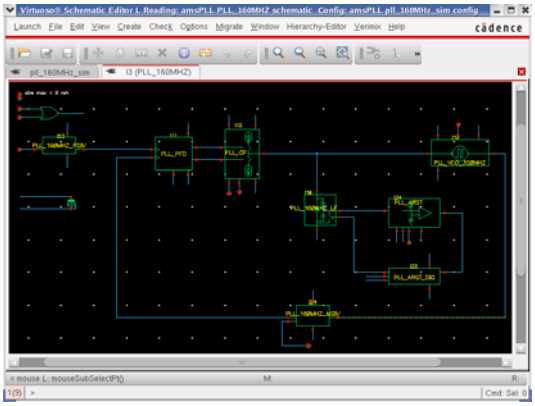

Fig. 2 PLL Block

To run this example, **IC611ISR, MMSIM61USR1 and IUS61u1** will be used. The UltraSim solver will be used (UltraSimVerilog and AMS/Ultra), however, the same flow applies to the Spectre solver.

1. UltraSimVerilog Run

This testcase runs in UltraSimVerilog, and the config view is generated from the UltraSimVerilog template (Fig. 3).

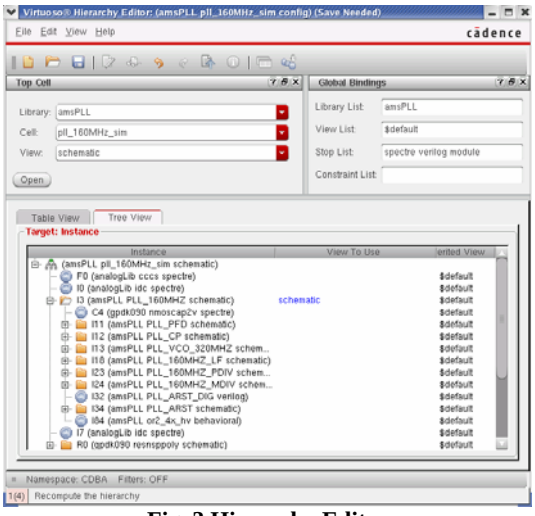

Fig. 3 Hierarchy Editor

Interface Element (IE) setup is quite simple for UltraSimVerilog because UltraSimVerilog only supports signal direction connect modules. Figure 4 shows the IE setup form for a UltraSimVerilog simulation. The designer only needs to set parameters for the a2d and d2a IEs.

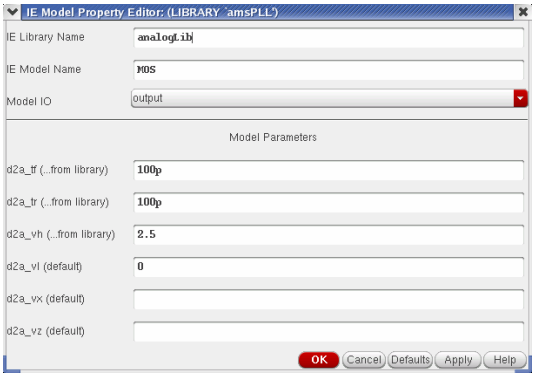

Fig. 4 IE Setup Form

Figure 5 shows the ADE (ADE-L) window. Note the reference frequency is defined as a design variable.

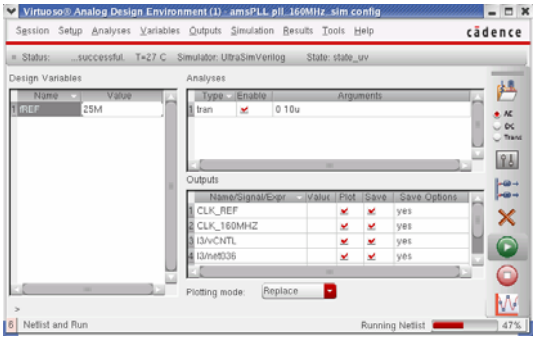

Fig. 5 ADE-L

Figure 6 and Figure 7 are the Analog Option and Digital Option forms. For the UltraSimVerilog run, sim_mode=ms, speed=3 and analog=2 are defined to achieve best performance and also accurate simulation result. Note that in the Digital Option form, text verilog files are included using the –v option.

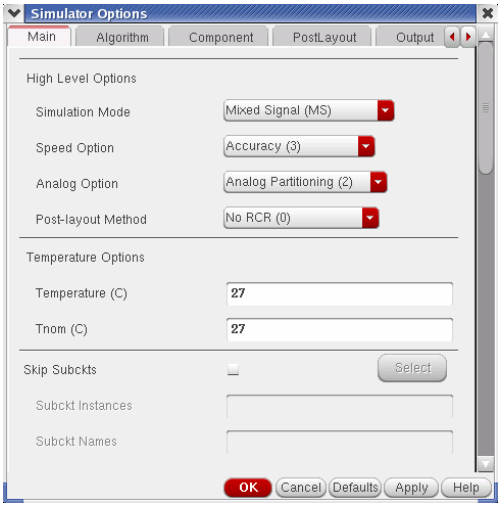

Fig. 6 Analog Option Form

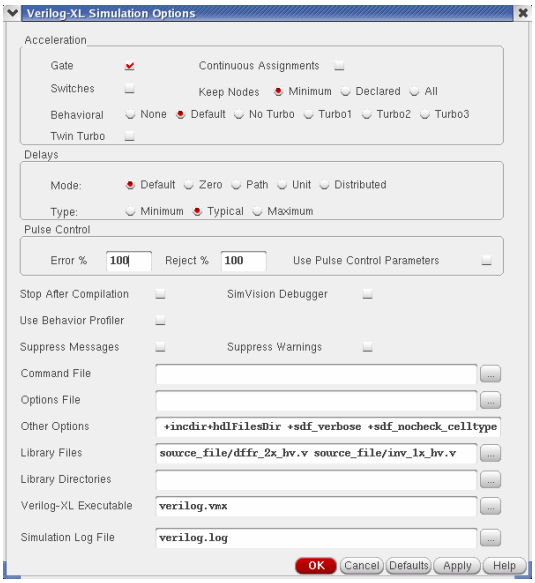

Fig. 7 Digital Option Form

Click on the Netlist and Run button to kick off the simulation.

After the simulation is done, the waveforms of the selected signals will pop up. (Fig. 8).

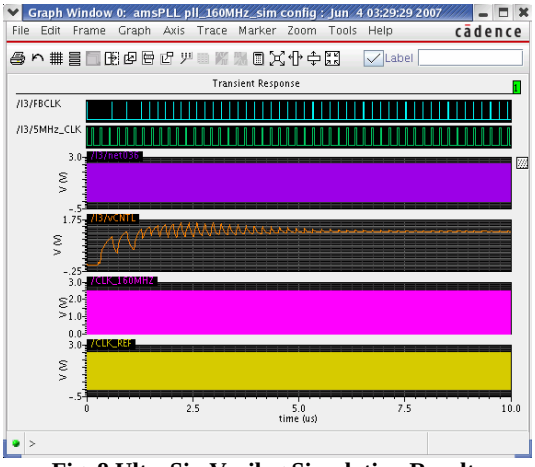

Fig. 8 UltraSimVerilog Simulation Results

2. AMSD Run

As already discussed in this paper, AMSD is a very powerful mixed-signal simulation tool but there are some barriers for designers to migrate to AMSD. The OSS netlister and NC-Verilog flow is designed to work through these barriers. Next we will run the same case in AMSD with OSS and NC-Verilog. It will be shown that only a few setup steps will be required to run this case in AMSD.

Since OSS uses the spectre view for netlisting, there is no work needed to modify the PDK and the same config will be used. (Fig. 3). No CDF or design changes are needed.

AMSD has one simulator kernel but includes the choice of one of two analog solvers: Spectre or UltraSim. They are designed for different type of usage. After choosing AMS as the simulator in ADE, set the Analog solver to UltraSim. (Fig. 9).

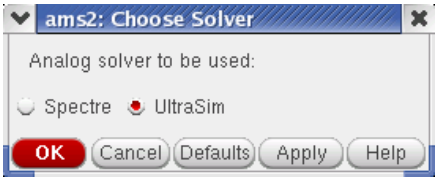

Fig. 9 Analog Solver Form

After selecting the analog solver, specify the OSS Netlister and NC-Verilog flow on the Run Options form (Fig 10). This will allow the run of the UltraSimVerilog testcase in AMSD while taking advantage of the advanced features of AMSD.

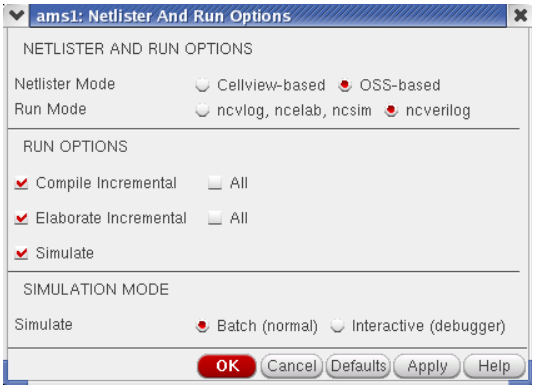

Fig. 10 Choose OSS + NC-Verilog

Setup and configuration of connect rules is different between AMSD and UltraSimVerilog. AMSD uses Verilog-AMS to describe the connect rules which support single direction and bi-direction interface elements. AMSD connect rules are more powerful allowing modelers and designers to characterize the analog-digital interface accurately in standard Verilog-AMS. They also support advanced features like driver-receiver segregation for the automatic calculation of a delay on a digital signal path due to analog loading effects.

During the migration from UltraSimVerilog to AMSD the designer needs to set the appropriate connect rule. There are several built-in connect rules that the user can choose from (Fig. 11), and then customize the threshold values (Fig. 12) to fit the design. Various templates for connect rules for different voltages are provided along with an easy way to use or modified them for different designs. In this example, connectLib_ConnRules_3V_basic is modified to a 2.5V voltage supply to fit the design. In addition to built-in connect rules, the user can also create his own connect rule files for the simulation.

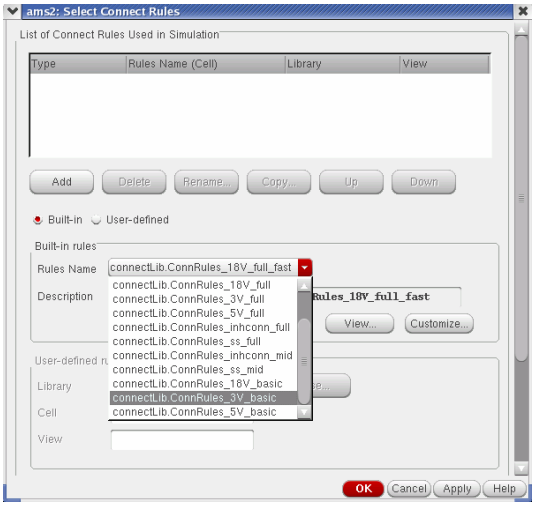

Fig. 11 Select Connect Rule

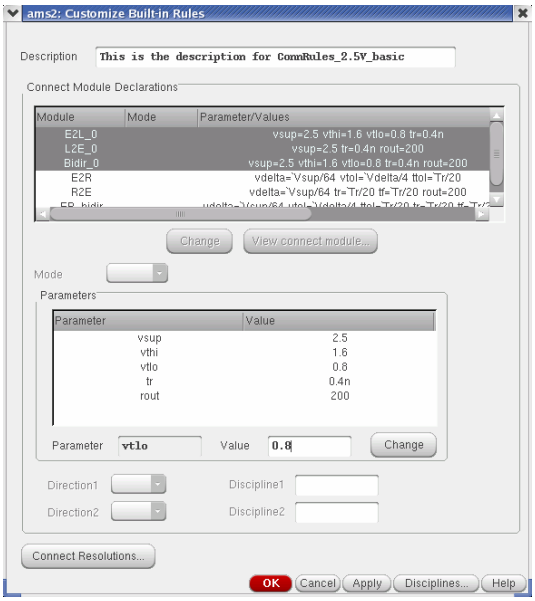

Fig. 12 Customize the Built-in CR

The last modification is to set appropriate options in the AMSD option form. As with the UltraSimVerilog simulation, the same –v feature is used to include the text verilog modules (Fig. 13).

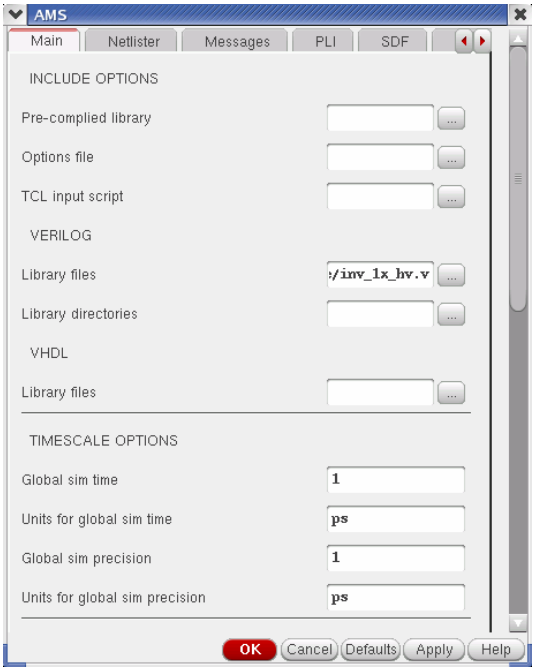

Fig. 13 NC-Verilog Option Form

The only difference between the UltraSimVerilog and AMS/Ultra setup in this example is the interface elements versus connect rules setup. In a future release, an interface element like GUI will be available, so that it will appear that the same simple setup can also be used in AMSD. The Connect Rules form will continue to exist for those users who want to take full advantage of the power of AMSD. After the setup, the simulation is run. The results are shown in Figure 14 and are identical to the UltraSimVerilog results (Fig. 8).

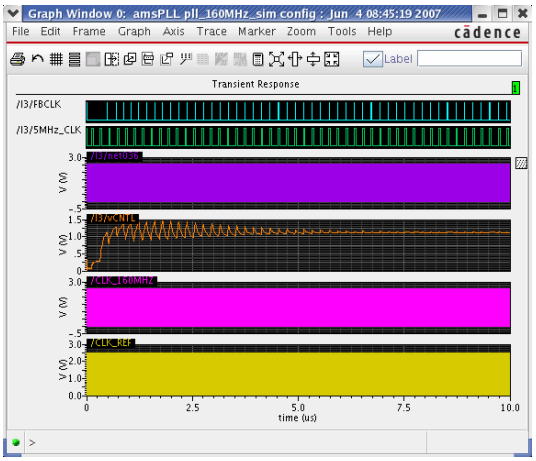

Fig. 14 AMSD Simulation Results

UltraSimVerilog uses two simulators (UltraSim and Verilog-XL) that communicate to each other through an IPC channel. AMSD uses only one kernel and all the new features that were mentioned previously to improve performance and capacity. In fact for this case, from Table 1 below we can see AMS/Ultra can save up to 22.42% simulation time compared to UltraSimVerilog.

	UltraSim Verilog	AMSD (UltraSim)	change
run time	598.06s	463.95s	22.42%

Table 1 Performance Comparison

FUTURE WORK

In addition to the work already completed, there are a few more areas where usability and ease of use can be improved to help ease the migration to AMSD.

In the current solution, mixed-signal options are stored in a file called props.cfg that exists within the config. The fact that this file is in some other location (the config) makes it again harder to debug issues, as everything that influences the simulation is not in the same place. In a future release, the information stored in the props.cfg file will be included within the analog control file in the netlist directory.

Interface elements, are less powerful, but easier to setup and understand than disciplines and connect rules. Therefore a simplified IE-like GUI will be created for the users who are new to AMS. It will resemble the very simple UltraSimVerilog a2d/d2a GUI. The current disciplines and connect rules GUI will continue to exist as they are very useful for the power users.

While the OSS netlister contains the features required for SpectreVerilog migration, it does not contain all the features of the cell based netlister. For example, text used within dfII needs to be imported using Verilog-In. These types of issues (feature compatibility with the cell based netlister) are planned to be resolved in future releases.

Other specific projects related to performance and usability are also planned.

SUMMARY

The OSS netlister and NC-Verilog integration provides a smooth migration path from SpectreVerilog. No PDK changes are needed and the same config, cds.lib and design works without changes. Much needed features such as the –y/-v Verilog library and file text inclusion work as they did for SpectreVerilog.

AMSD performance has been improved with the fast cross and c-interface for connect modules, and the Verilog-A compile-C features. For those worst cases where AMSD was 3X slower, AMSD is now on par with or faster than SpectreVerilog.

AMSD provides many state of the art advanced features and languages over that offered by SpectreVerilog. It supports Verilog-2001 and System-Verilog, etc. It provides capabilities of mixed-signal behavioral modeling in Verilog-AMS and VHDL-AMS. It also supports high performance modeling features such as discrete time real number modeling and its direct conversion with analog real signals. Now with the migration barriers lifted, this is all more easily within reach