New AHDL Based on 'C'

Introduction

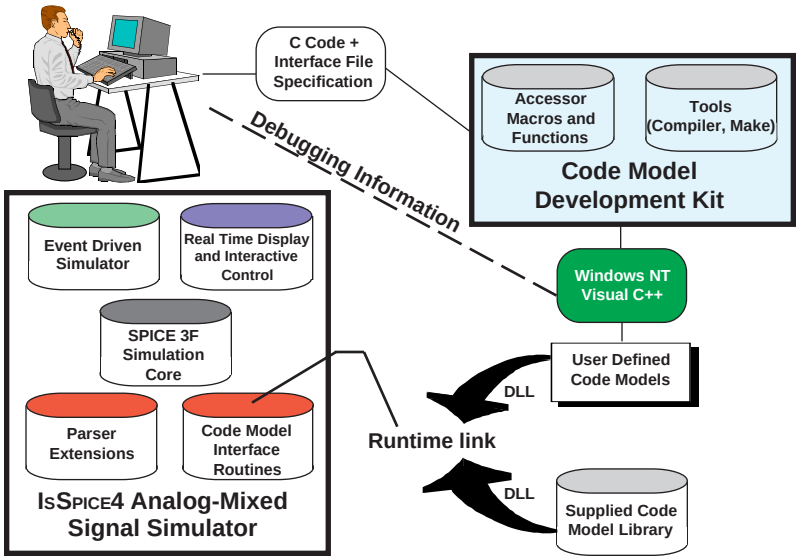
Intusoft, makers of the IsSpice simulator, have found a new way to extend the capabilities of SPICE by allowing engineers to easily add user defined models based on C code subroutines.

SPICE based simulators have progressed towards a viable AHDL dramatically since the old Fortran version was released over 20 years ago. Today, hardware vendors around the world are distributing thousands of models written in the SPICE language. At present, SPICE 3 based simulators offer a wealth of modeling features. For example, IsSpice4 allows math expressions, nonlinear differential equations, and If-Then-Else conditional branching, providing support for BOTH procedural AND behavioral model descriptions. Virtually all AHDL vendors agree that SPICE macromodeling is comparable to AHDL model development. For example, once a SPICE macromodel is developed it can be used like any of the traditional SPICE primitive elements. The only difference between current AHDL languages and SPICE is the number of constructs that are available to build new primitives.

Developing C Code Based SPICE Models

Often, the designer has a very clear idea of how a particular behavior could be modeled in a programming language like "C", but has no idea how to go about adding a new device model to SPICE. Those determined enough to examine the SPICE source code quickly come to the conclusion that the effort involved does not justify the potential benefits. Macromodeling, on the other hand, may not provide the required model flexibility or speed.

FIgure 1, The Intusoft **Code Model Development Kit** (CMSDK) allows you to add your own C based models directly into the simulator.



Intusoft has developed up with an easy way for an engineer to produce and add C code based subroutines to SPICE without having to deal with the details of interfacing with the SPICE software and data structures. What would have taken months to develop and integrate with SPICE can now be done in a matter of hours. The C code modeling capability completes the AHDL picture for SPICE giving it all the features and portability of even the most powerful AHDL languages.

The model development system consists of the IsSpice4 simulator and a C code model development kit (Figure 1). The code modeling architecture is based on the XSPICE program produced by the Georgia Tech
Research Corporation, a division of the Georgia Institute of Technology in Atlanta GA. XSPICE is an extended UNIX version of Berkeley SPICE 3C.1. Intusoft has converted and added the XSPICE enhancements to its powerful Windows based SPICE simulator, IsSpice4, which is based on SPICE 3F.5.

Code Model Development

The benefits of developing models using C code have not been widely recognized, mainly due to the limited programming and simulator support and extensive knowledge of SPICE that is required. The Intusoft Code Modeling Software Development Kit (CMSDK) alleviates these difficulties. It assists the user in writing compiling, linking, and using new C code subroutines with IsSpice4. The SDK works with additions to the simulator core to tell the simulator how to parse the extended circuit description and model statements and how to call C code that defines the model's behavior.

Creation of a new model requires 5 steps:

- Creation of the directories to hold the appropriate files
- Definition of the model interface specification
- Definition of the C subroutine (model behavior)
- Building and linking of the model

All the engineer needs to produce a new code model are an interface specification file describing the connections and parameters of the code model (entered in an easily readable text format, (Figure 2)), and a code model body written in ordinary "C" code (Figure 3). The toolkit provides a comprehensive set of macros and functions to simplify the model development process and to insulate developers from the underlying SPICE algorithms and data structures. For example, macros to access information in the simulator (voltages, parameter values) and functions for smoothing, model state storage functions, integration and convergence message handling, breakpoint handling, complex math and more are provided.

Next, the developer uses a provided makefile to compile and link the code model. Compiler and linker diagnostics aid in the debugging process as they would for any application being developed under Visual C++ (1.1 or 2.0). The result of a successful build is a code model library (DLL) that can be accessed (dynamically linked) with the IsSpice4 executable at runtime. A command line switch causes IsSpice4 to pause after dynamically loading code model libraries, which allows the user to set breakpoints, if desired, in the code model. Once a

FIGURE 2 (Interface Specification):

The Port_Table describes the model's nodal connections. The Parameter_Table describes the parameters that will be available in the code model's .MODEL statement.

```
NAME_TABLE:
C_Function_Name:      cm_gain
Spice_Model_Name:     gain
Description:          "A simple gain block"

PORT_TABLE:
Port_Name:            in                  out
Description:          "input"             "output"
Direction:            in                  out
Default_Type:         v                   v
Allowed_Types:        [v,vd,i,id,vnam]    [v,vd,i,id]
Vector:               no                  no
Vector_Bounds:          -                 -
Null_Allowed:         no                  no

PARAMETER_TABLE:
Parameter_Name:       in_offset           gain        out_offset
Description:          "input offset"      "gain"      "output offset"
Data_Type:            real                real        real
Default_Value:        0.0                 1.0         0.0
Limits:               -                   -           -
Vector:               no                  no          no
Vector_Bounds:        -                   -           -
Null_Allowed:         yes                 yes         yes
```

FIGURE 3 (Code Model Body):

```
void cm_gain(ARGS)   /* structure holding parms, inputs, outputs, etc.     */
{
      Mif_Complex_t   ac_gain;

      if(ANALYSIS != MIF_AC) {
            OUTPUT(out) = PARAM(out_offset) + PARAM(gain) *
                          ( INPUT(in) + PARAM(in_offset));
            PARTIAL(out,in) = PARAM(gain);
      }
      else {
            ac_gain.real  =  PARAM(gain);
            ac_gain.imag = 0.0;
            AC_GAIN(out,in) = ac_gain;
      }
}
```

Code Models are SPICE Primitives, like BJTs or resistors. They can be written in a matter of days in contrast to the months required for making new SPICE models.

FIGURE 4 (Sample SPICE Netlist):

```
A SIMPLE AMPLIFIER CIRCUIT
.OPTIONS  CML=GAIN.DLL
.TRAN 1E-5 2E-3
VIN  1 0  0.0 AC 1.0 SIN(0 1 1K)
CCOUPLE  1 IN  10UF
RZIN  IN 0  19.35K
AAMP  IN COLL  GAIN_BLOCK
.MODEL GAIN_BLOCK GAIN (GAIN = -3.9  OUT_OFFSET = 7.003)
RZOUT OUT COLL 3.9K
RBIG  COLL 0  1E12
.PRINT TRAN V(1) IN COLL OUT
.END
```

breakpoint is encountered, Visual C++ provides the ability to examine variables, step through the execution, and otherwise support debugging.

In order to ease model development, the code modeling approach shields the developer from most of the simulator's internal operation. This alleviates problems that were once the hurdles hindering SPICE model development. Prior to the simulation IsSpice4 automatically:

- Parses code model lines in the input card deck (Various circuit debugging feedback including verification of the right number of connections, range checking of parameters, etc.)
- Sets up the model data structures, including supplying default values for parameters not specified on the .MODEL line
- Sets up the circuit description data structures for code model instances.
- Creates equations in the matrix for voltage sources. Sets up the matrix pointers used during simulation to load the matrix.

During the simulation IsSpice4:

- Iterates through all models and instances of a specified code model device type, fills in the inputs for the model, calls the model, and then uses the outputs and partials returned by the model to load the matrix.
- Monitors conversion and truncation errors in code models

Once the user is satisfied that the model is performing correctly the code model library can be copied for use on other workstations running IsSpice4 under Windows 3.1x or Windows NT. Note that while development under Windows NT is required, simulation can be performed under Win32s (Windows 3.1x).

These steps are easily learned and only require a few minutes to perform once the C code is written. Currently, this process is supported by Intusoft using Microsoft Visual C++ 1.1 or 2.0 under Windows NT or Windows95. This setup was chosen because it is the only current viable 32-bit development alternative under Windows. Figure 3 shows a the final SPICE netlist that makes use of the sample gain model.

Advantages of Code Modeling

Since the XSPICE interface for C code models is publicly available, all models written for it are transportable to any other simulator that supports the code model interface. Intusoft's version accesses the C code model via a Windows DLL. This alleviates the need to recompile the SPICE executable each time a new model is added. Addition of the DLL features allows models to be easily distributed and copied by developers and users and copy protected by the programmer if required. Trading and/or purchasing code model libraries provides the means to greatly expand the capabilities of SPICE, even for those not interested in developing their own code models.

The C code modeling approach has several advantages:

- Provides higher level primitives for macromodeling purposes
- Provides a full range of digital primitives as well as ability to create custom digital elements
- Allows user-defined node/data types
- Allows new primitives to be added to the simulator core more efficiently than any other direct coding method
- Allows arbitrarily complex functions and behavior to be added to SPICE

Code models support:

- Any number of inputs or outputs
- Scalar, vector, real, integer, ground reference, differential, digital (12-states), resistance, conductance, voltage and current port types
- Unlimited number of model parameters of type real, integer, complex, string, or boolean
- Model parameter defaults and constraint checking
- Model information and message output.

IsSpice4 includes an embedded event-driven simulation with full 12 state digital modeling plus an extensive set of over 42 embedded analog, digital, and hybrid (mixed analog/digital) code models ready to use.

Some examples of the IsSpice4 primitives based on code models are:

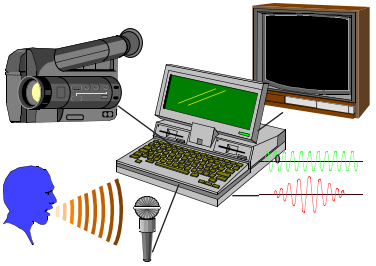
- System blocks (Laplace expressions, limiters, hysteresis, slew rate)
- Digital logic functions, state machine, RAM, frequency divider
- Magnetic core, zener diode, controlled oscillators

In summary, code modeling extends IsSpice4's capabilities to allow efficient simulation of mixed-signal, board, and systems-level circuits. It provides a practical alternative to macromodeling and the widely scattered AHDL languages. It uses the commonly available C language and shields the user from virtually all internal simulator related concerns through specially developed macros and functions.

What Can Code Modeling Do For Me?

In order to simulate a range of complexities and applications efficiently Intusoft provides several modeling techniques; the most powerful of which is a new AHDL. An AHDL allows you to easily develop just about any model or interface other functions to SPICE. The models may be as simple as a transfer function or as complex as a new MOSFET. They can perform virtually any function, analog, or digital, and be blended with any other SPICE model or behavioral modeling syntax.

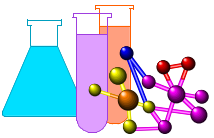
Code modeling opens up many areas of simulation to SPICE users that were formally closed because of the complexities of modeling. It extends IsSpice4's capabilities to allow efficient simulation of mixed-signal, board, and systems-level circuits using a top-down design methodology. It also provides a practical alternative to macromodeling and other current widely scattered AHDL offerings.



Real Time Audio and Video Signal Processing

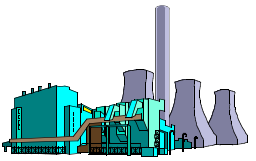
Array Processing

Automotive, Mechanical

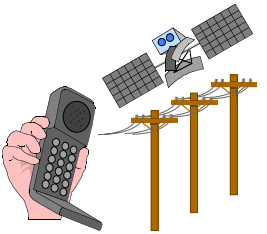


Chemical, Biological

Neural Networks



Control Systems



Communication Systems

With the CMSDK all types of systems and processes can be modeled and simulated whether they include electronic devices or not. It is now possible to interface IsSPICE4 with hardware, or other programs, or mix electrical and nonelectrical processes.

What's Included in the CMSDK

The CMSDK consists of a compiler and other tools, code model examples and source code, include files, and documentation on how to create AHDL models. The toolkit is supported under Microsoft Visual C++ 4.X or 5.X under Windows NT or Windows95.

References

1) "Code-level Modeling in XSPICE", Fred. L. Cox III, William B. Kuhn, Jeffery Murray, and Stephen, IEEE 0-7803-0593-0/92

2) "SPICE as an AHDL", Charles Hymowitz, Analog & Mixed Signal Conference, July 1994

3) XSPICE Users Manual, Georgia Research Corporation, Georgia Institute of Technology, Atlanta Georgia 30332-0800

Table 1 - Major benefits of Using the Intusoft CMSDK and XDL

Code Modeling Benefits To Users

- Create analog, digital, and mixed mode models using the C language.
- Develop specialized models easily: Radiation, RF, Neural Networks, Array Processing, Image Processing, System transfer functions, etc.
- Provide higher level models for system-level top-down design
- Interface other programs, simulators, and hardware, to SPICE
- Extend SPICE's macromodeling capabilities to nonelectrical applications
- Users can share compiled code models (DLL's) since any IsSpice4 program can access them (The CMSDK is only needed for development not usage.)
- The language and tools are affordable
- XDL is supported on Windows, Windows NT, and Unix

Code Modeling Benefits To Hardware Vendors

- Language based on publicly available, nonproprietary XSPICE standard
- Model's C source code is portable to ANY simulator that is compatible with XSPICE, even across platforms.
- Source code OR compiled DLL object can be distributed to users.
- Use of the DLL hides the model implementation. DLL can be copy protected.
- Code models can be combined with existing SPICE-based macromodels

Code Modeling Language Benefits

- XDL uses C; no AHDL language is more standard or well known
- XDL provides access to the operating system thus allowing greater flexibility in the area of interfaces and mixed domain simulations (hardware-software)
- XDL provides greater modeling flexibility than other solutions
  - Unlimited state variables
  - Unlimited model parameters
  - Unlimited nodal connections
  - Unlimited node and model types
  - Flexibility is limited only by what you can do in C
- XDL is much easier to learn since it is not a new language
- Developers can use SPICE modeling code and techniques
- XDL, SPICE 2, and SPICE 3 could become part of the VHDL-A spec

Code Modeling Benefits to Model Developers/Tool Vendors

- The tools are affordable and available on common platforms
- Develop models using commonly known language and commonly available tools
- Add models to SPICE in Days instead of Months
- Easier to adapt existing SPICE models than for other AHDLs
- Easier to adapt other simulators; much less resistance from since language was developed by academia
- The CMSDK shields the model developer from virtually all internal SPICE related concerns through specially developed macros and functions.
- The code model interface is nonproprietary and can be added to any SPICE program or other proprietary simulators.
- Primitives are added to SPICE via a Windows DLL; Users can add to the existing DLL or create separate DLLs
- No need to recompile the SPICE executable each time a new model is added.
- When paired with IsSpice4, the CMSDK can be used as a test bed for model development on other platforms and simulators.