HDLs for Speech Processing

by Charles Hymowitz, Intusoft

- New Non-Proprietary HDL based on C for Windows and Unix
  - Analog & Digital Modeling and More
- **XDL** - e**X**tended **D**escription **L**anguage
  - SPICE Today
  - What are AHDLs good for anyway?
- XDL Model Development
  - Model development made "easy"
- **SALT** - SPICE Array Linking Technology
  - Model development made even easier
  - OLE2.0 Integration
  - The New Frontier - Mixed Domain Simulations
- Speech processing using .wav file data, array processing algorithms, SPICE, and PC Hardware

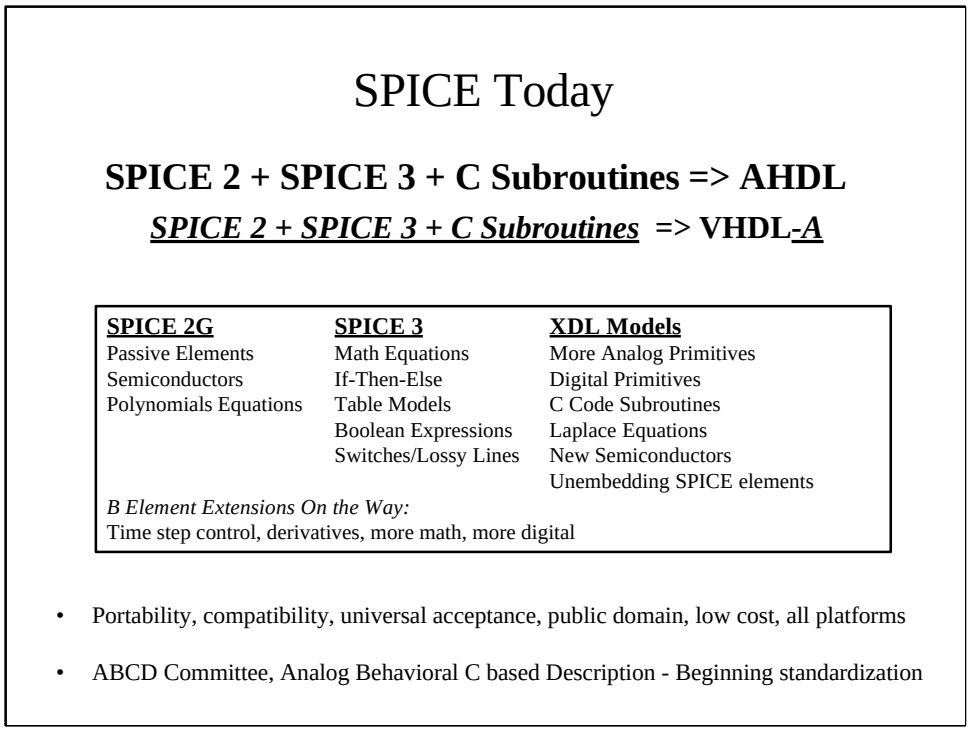
Current AHDLs focus on device modeling. To limit HDL usage to this realm is to waste a vast and powerful capability. HDLs will change the simulation world, but not just in the area of modeling as most proponents are proclaiming. This article discusses how XDL, a new public domain analog and mixed signal HDL released by the Georgia Institute of Technology, allows engineers to venture beyond modeling tasks to tackle new applications. As an example, a "mixed domain" simulation is featured whereby a SPICE simulation is enhanced with algorithmic processing and interfaced with PC hardware. Description of the mechanisms involved in the XDL model development, including the model architecture and use of OLE2.0, is covered. In addition, a new library of array processing models (the SALT modeling package) and hardware/software interface models is discussed.

SPICE Today

- Analog Simulation
- Mixed Mode and Mixed Technologies
- Applications: Mechanical, Hydraulic, Thermal, Physical, Biological, Fuzzy Logic, Neural Networks
- Modeling
  - SPICE 2 Constructs
  - SPICE 3 Constructs
  - Behavioral, Mathematical, and Procedural
  - Nonlinear Differential Equations
- Model Portability - Over 3K models from >15 vendors; some using SPICE 3

SPICE is a proven simulator for all types of electrical AND nonelectrical simulations. An AHDL capability is NOT needed to perform mechanical, hydraulic, thermal, or physical simulations. SPICE simulators today support three classes of constructs; SPICE 2, SPICE 3, and SPICE extensions. Some common extensions, available in most commercial versions of SPICE, are shown on the next page. These advanced modeling constructs are the main reason why users don't need an AHDL to model the vast majority of the analog and mixed signal functions.

In terms of model portability, SPICE models are, by far, the most prolific. Over 3000 models from more than 15 vendors are now being exchanged throughout the industry. More are continuously created and distributed. While some syntactical incompatibilities exist, they are rarely insurmountable. For example, the syntax for procedural If-Then-Else constructs [1], Table models [2], and switches [3] are different between different SPICE vendors. However, models using these SPICE 3 extensions are in widespread use today with the syntax differences causing no difference in performance and little difficulty in translation.



For an AHDL to be accepted it must address several issues. The language should offer portability between simulators and a comprehensive set of modeling constructs. It should allow analog, digital, mixed mode, and nonelectrical elements to be simulated together and have a library of predefined models. The language should have both academic and industry backing, and be available at a reasonable cost on a variety of platforms. These qualities are also largely applicable for analog extensions to VHDL and Verilog.

Clearly, today's SPICE 3 based simulators support virtually all of these requirements without the need for any AHDL language-like support [4]. Not enough is known about the advances in behavioral modeling due to new Berkeley SPICE 3 versions and industry wide vendor improvements. Current features are often overlooked by AHDL proponents, to the public's detriment. In many cases, AHDLs are compared to SPICE 2, which is almost 20 years old, rather than current state of the art SPICE 3 based simulators.

While SPICE presents some powerful capabilities there are many areas where it can not venture. These are most easily covered by offering an interface to the C programming language. Standardization of current SPICE 3 extensions plus a C code subroutine ability would effectively eliminate the need for any specific "AHDL language" and would result in far greater benefits to the engineering community. This is the task of the ABCD committee which has begun a standardization effort on several fronts.

XDL - A New Non-Proprietary HDL

- What advantages does XDL have?
  - Non-proprietary (Georgia Institute of Technology)
  - Tight linkage with Berkeley SPICE
  - Windows and Unix versions
  - Based On C
    - The "real" standard language
    - Access to the operating system
    - Windows DLL object, OLE 2.0 integration
    - Access to the latest software technology without vendor interference
- AHDL Benefits
- Ability for users to define their own "signal" (node) types as arbitrary C data structures; Array Node for example

A primary example of the type of C subroutine architecture that can be added to SPICE is XDL. XDL (eXtended Description Language) is based on the XSPICE program from the Georgia Institute of Technology.
XSPICE starts with SPICE 3C.1 and adds a built-in (native) general purpose event-driven simulator and the ability to add C code subroutines as new model primitives. To help interface the C subroutines, a series of compilers, macros, and functions are included to ease model development and shield the developer from the intricacies of SPICE. This additional non-SPICE functionality can be extracted from the SPICE 3C simulator core in XSPICE and added to other proprietary simulators. For instance, the XSPICE additions have been ported to Windows by several vendors including Intusoft and Cad-Migos.

There are many benefits to using a C based AHDL, like XDL, under Windows (see the following page). While some benefits result from being under Windows, others are inherent in the HDL itself. One key benefit is that C provides access to the operating system. Another is that XDL allows designers to define their own "signal" (node) types as arbitrary C data structures.

These features may not seem that significant, but the combination of the user-defined signal capability, an event-based simulation capability, and access to the operating system, allows XDL to support advanced mixed-technology simulations and support application domains well beyond conventional simulation boundaries.

XDL Benefits

XDL Modeling Benefits To Users

- Creates analog, digital, and mixed mode models using C.
- Implements specialized models: mechanical, radiation, stripline, neural networks, image processing, etc.
- Provides high level models for system-level top-down design
- Interfaces other programs, simulators, and hardware to SPICE
- Mixes hardware test and measurement with SPICE simulations
- Can add or share models without the need to buy more software or wait for updates
- Currently supported on Windows, Windows NT, and Unix based simulators

XDL Modeling Benefits to Model Developers/Tool Vendors

- Adds models to SPICE in Days instead of Months
- Avoids the need to recompile the SPICE executable each time a new model is added
- Easier to learn; no need to learn a new language or new tools
- Easier to adapt existing SPICE models (compared to other AHDLs)
- Easier to adapt to other simulators; much less resistance since language was developed by academia and is virtually free
- The interface is non-proprietary and can be added to any SPICE program or other proprietary simulators.
- Predefined library of functions shields the developer from virtually all internal SPICE related concerns
- Primitives are added to the simulator via a Windows DLL
- SPICE elements can be separated from the simulator (several have already been extracted)
- Tools are affordable and available on common platforms

XDL Modeling Language Benefits

- XDL uses C; no AHDL language is more standard or pervasive
- XDL provides greater modeling flexibility than other solutions
  - Unlimited state variables
  - User-defined node/data types
  - Greater flexibility; only limited by what you can do in C
  - Unlimited number of model parameters of type real, integer, complex, string, or Boolean
  - Unlimited ports ; Scalar, vector, real, integer, ground reference, differential, digital (12-states), resistance, conductance, voltage and current port types
  - Definition of model ports and parameters via ASCII text file
  - Access to the operating system
- Developers can use SPICE modeling code and techniques
- XDL, SPICE 2, and SPICE 3 could become part of the VHDL-A/Verilog-A specification

XDL Modeling Benefits To Hardware Vendors

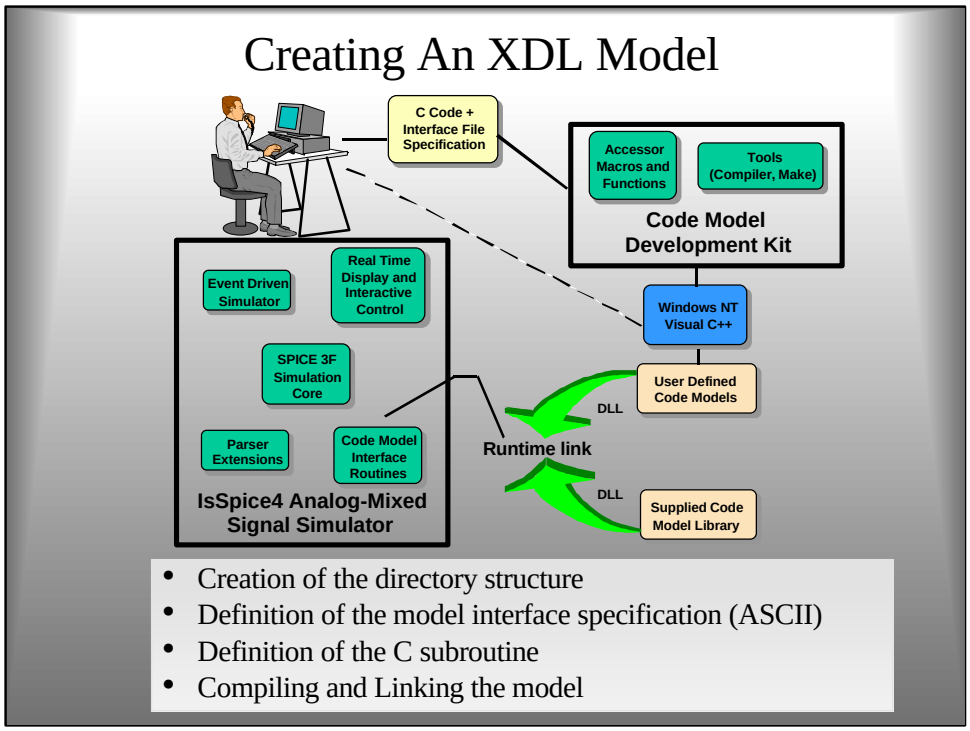
- Language based on publicly available, non-proprietary XSPICE
- Model source code is portable to ANY simulator that understands XSPICE
- Source code OR compiled DLL object can be distributed to users; hides proprietary information
- Use of the DLL hides the model implementation. DLL can be copy protected.
- XDL models can be combined with existing SPICE macromodels
- XDL can be used to encrypt existing SPICE macromodels

What Will AHDLs Be Used For?

- Modeling (from the analog/mixed signal designer's point of view)
  - AHDLs are redundant, difficult to use, overly complex
  - Semiconductor models are usually written in C anyway
  - Models have more simulator dependencies than in VHDL/Verilog
- C - key to expansion beyond modeling
- **Interfacing different simulators, utilities, programs**
- **Integrating Hardware <- with -> Simulation**
  - Integrated Test, Measurement, and Simulation
  - Access to real life conditions (stimulus, control)
  - Analog hardware modeler

HDL efforts to date have been myopic and too narrowly scoped. Design engineers need description languages and design environments that help them control the entire system design space, not just pieces of it. They need description languages that support broad static analysis, as well as simulation. They need design environments that support in-the-loop analysis of a mixture of simulated, prototyped, and fielded components along with their users. They need description languages and environments that support automatic derivation of manufacturing and maintenance testing during the design process while testability flaws can be corrected or avoided. These needs indicate description languages must exceed their current scope. XSPICE extensions to SPICE, including XDL, were developed with these objectives in mind.

The modeling capability of an AHDL represents only a small part of its utility. With access to the C language, designer's can find a wealth of opportunities to expand the reach of circuit simulation beyond today's boundaries. Access to the operating system allows SPICE to interface with other programs and with hardware. This opens the possibility of a "mixed domain" simulation where a software simulation is integrated with data from hardware.



XDL models are like traditional SPICE models except they are created with 2 files; an ASCII table, called the “interface file specification” and a C code subroutine describing the model’s behavior. The interface file specification describes the model’s ports and parameters and is compiled into a C file with a supplied compiler tool. In the case of the Windows version of XDL, the code describing the model is linked to the simulator via an external file (Windows DLL) rather than being bound within the executable program. This allows new primitive models to be added to the simulator, and old models changed, without having to recompile SPICE.

XDL was developed with the thought that the number of model users would be proportionately greater than the number of model developers. It is expected that the power modelers will have substantially higher technical expertise, so, for them, the full capability of a general purpose language, C, is provided. However, since no one really wants to have to deal with the intricacies of SPICE, XDL insulates developers from the internals of SPICE with a comprehensive set of accessor macros and functions. Given the general purpose programming nature of C, it is possible for XDL to provide support for most anything that can be programmed including other HDL languages.

The combination of a standard language (C), ASCII model definition file, macros and functions for common operations, and a popular compiling and debugging environment (Microsoft Visual C++) provides one of the easiest AHDL model development paths for the analog designer.

XDL Model Description - This page contains an example listing of an XDL GAIN block. Included are the interface file specification showing the model's ports and parameters and the XDL code describing the model's behavior.

ASCII Model Description File - Interface File Specification

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

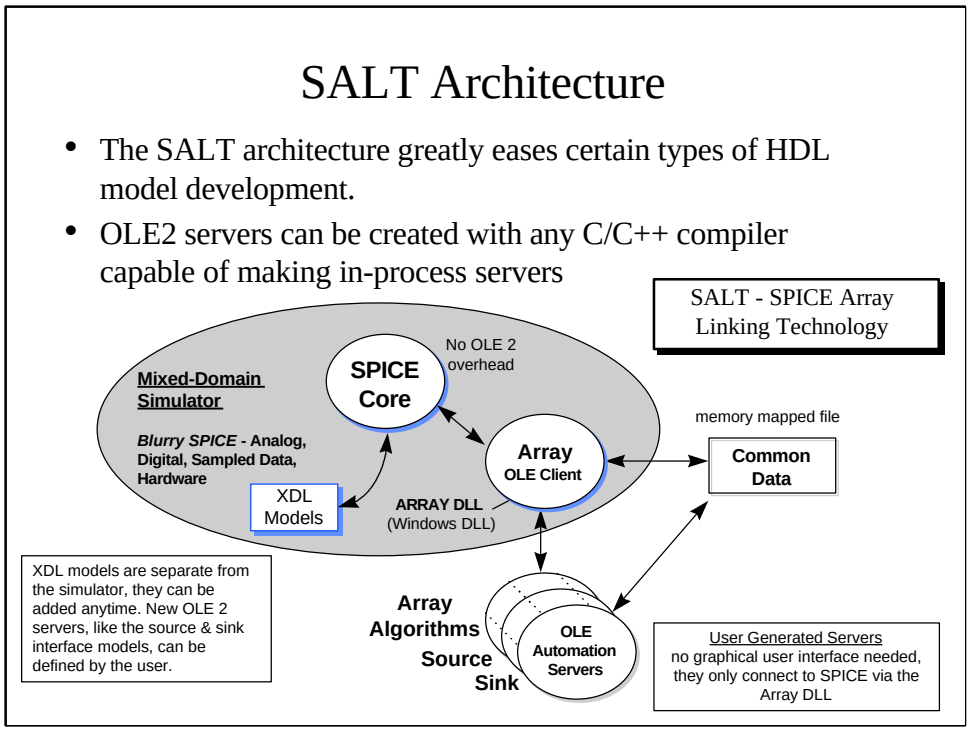
PARAMETER_TABLE:

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

XDL Model Body

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

      if(ANALYSIS != 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;
      }
}
```



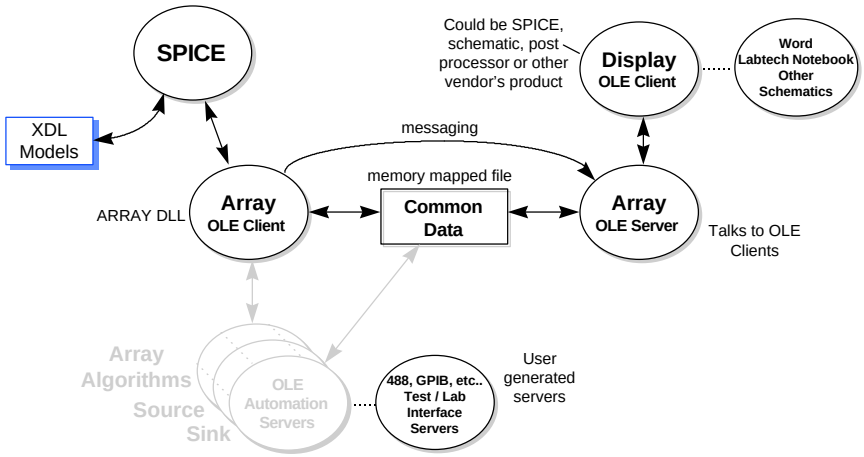
One example of the extent to which XDL can enhance SPICE is shown above. SALT, or SPICE Array Linking Technology, provides a generalized OLE 2.0 interface. It also provides various array processing functions and file/hardware input (source) and output (sink) functions that are ready to be used. These functions are OLE 2.0 automation servers, but act like standard SPICE models from the user's point of view. The interconnection between the SPICE simulator and the OLE 2.0 objects is provided by the ARRAY DLL, also provided with SALT.

The ARRAY DLL is an XDL model created with the Intusoft Code Model Software Development Kit (CMSDK). The CMSDK is used to create XDL models under Windows. SPICE interfaces are provided by the Array client. Windows-OLE2 interfaces are provided by an OLE automation server template.

The OLE 2.0 automation servers are the easiest way to develop high level models because they do not require the user to interface directly with Windows or SPICE. No graphical interface is required and no XDL related development is needed. Designers can add their own array processing algorithms and interfaces by using any C compiler capable of building OLE 2.0 32 bit in-process server DLLs.

Extending Simulation Capabilities

- The Array (OLE Client) is an XDL model. Other OLE components connect to SPICE via the Array DLL.
- New OLE automation servers can be added without the need for any Windows or XDL related coding.

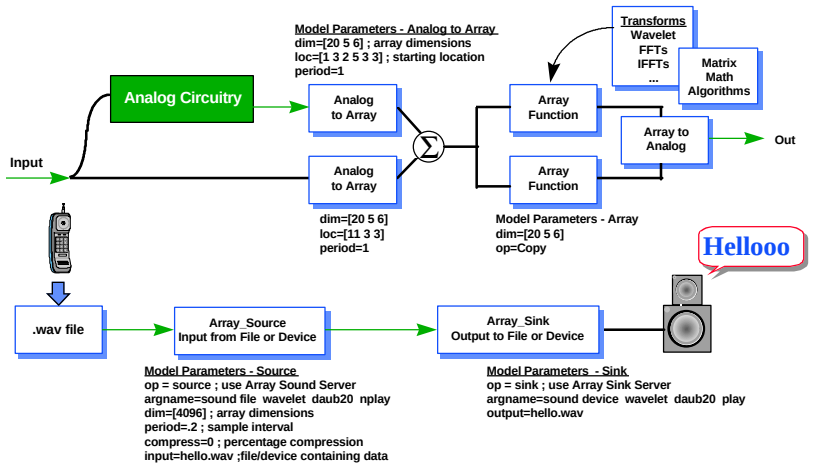


The OLE automation servers connect to array nodes that are, in reality, connected to memory mapped files. These files contain the data array and descriptors needed to identify data type and state information. The array nodes were created using the user defined signal capability available in the XDL. Input and output servers convert between real world data and memory mapped data. This makes it possible for sound to be input from a microphone which would then connect to an array node. The array node can then be converted to traditional analog or digital data.

The Array OLE server is a more advanced OLE component in that it has a graphical interface and can talk to other OLE clients. The OLE automation servers only talk to the ARRAY DLL. The Array server has methods needed by other applications. In particular, the Array OLE server interfaces to a Display container (Display OLE Client) which can render a pictorial or graphical view of the simulation data. The Display client is used to display the representation of the array waveform. The display client can work with many clients simultaneously; for example, in third party programs such as a schematic capture and data analysis applications.

Mixed Domain Simulations

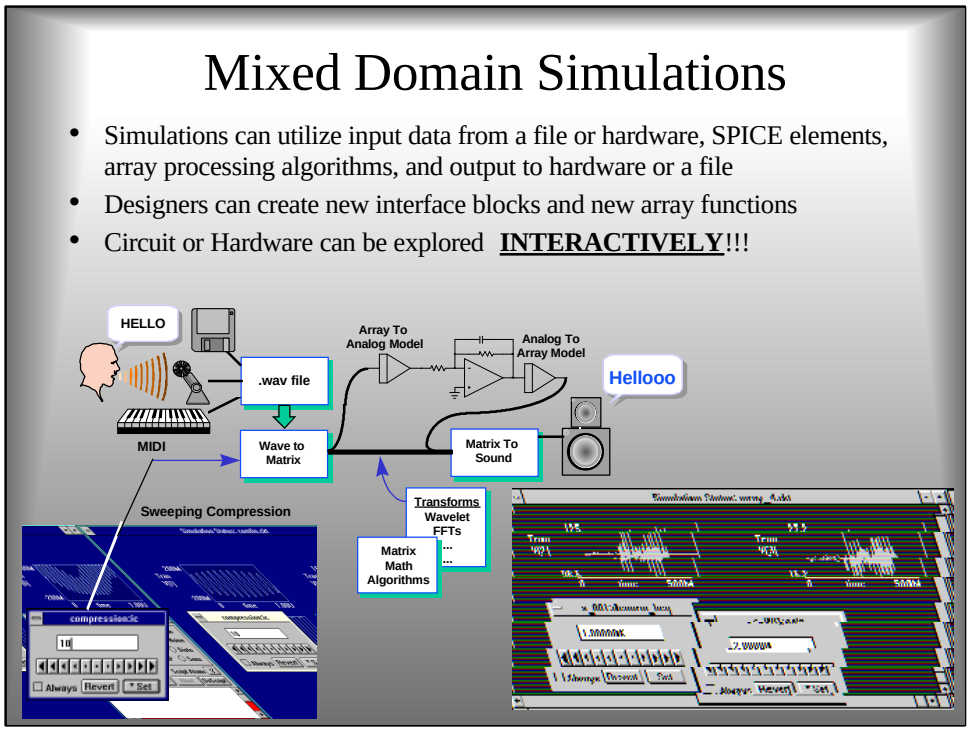
- Simulations can utilize input data from a file or a device, SPICE elements, array processing algorithms, and output to hardware or a file
- Designers can create new interface blocks and new array functions



Schematically, SALT objects connect like any other electronic device. The example above contains some basic illustrations. At the top, an input signal is feed into two analog to array blocks which convert the analog information into an array format. One signal is first passed through an analog filter. The array data is summed and then converted back into an analog signal (array to analog block). In this case, the Array Function blocks simply copy data from their input to their output.

In the bottom section, speech from a .wav file is read in and compressed by the Array Source block. The data is then uncompressed and played through a speaker by the Array Sink block. Both the top and bottom paths are simulated at the same time.

The source and sink objects convert between real world hardware, or data stored in files, and the array signal type. Array information can be processed with mathematical functions (FFTs, wavelet transforms, etc.) or be converted to an analog or digital signal. The model parameters for each block are shown in bold type.



In this next example, SPICE elements, array processing, and hardware interfacing are all utilized. Sound is read in (sampled) from a file as the input stimulus by the array source block. This time, some of the array elements are broken off and converted to an analog signal by the array to analog block. At this point the user is free to insert circuits with SPICE or XDL elements (analog, digital, nonelectrical, sampled-data, etc.). The processed data is then recombined into an array format before output to a speaker.

With the advent of integrating SPICE with real time processes, it is necessary to rethink SPICE's internal architecture in order to make it more appropriate for interactive rather than batch processing. This has been done in SPICE3 and to an even greater extent in IsSpice4, a commercial version of SPICE 3F. Using IsSpice4, both the electronic (filter bandwidth) and algorithmic portions (compression) of the design were interactively varied to see their effect on the sound output.

It should be noted that if there is no analog (SPICE) signal processing in the data path, then array information can be processed at or above real time speed.

OLE2 Server Possibilities

SALT

- Wavelet Transforms, Variable Radix FFTs, Matrix Math
- Sound input (file or microphone), Image input (TIFF)
- Sound output (speaker or file), Image output (TIFF or screen)
- Source Code
- **I/O From Hardware; Feedback with SPICE in the Loop**

User Defined

- Mathematical operations
- Connection of SPICE to other analytic engines, such as Matlab or Maple

The array processing library allows both sound and images to be used within a simulation. Both input and output are supported using file or hardware access. The audio and video interface models make it possible to include complex and realistic stimulus waveforms in a simulation rather than approximations. Several array transforms are also provided including wavelet transforms.

Connection of the SALT functions to the underlying analog simulator allows the user to push down into the details of a design hierarchy in order to compare a detailed implementation with a more abstract system view. For instance, complex blocks of circuitry can be replaced with algorithmic representations.

More important, however, is the ability for designers to add their own functionality. Interface models allow hardware devices to be connected directly to the simulation. Models can be developed that interface the simulation of newly designed circuit blocks to previously breadboarded circuitry. XDL models can also be used to connect SPICE to other analytic engines, such as MATLAB or Maple. One might use Prolog, for example, to emulate a human user of a system as an embedded part of a system simulation driven by IsSpice4, or perform symbolic analysis of waveforms generated by some other part of the simulated system. The possibilities are open ended.

Future SPICE and XDL Work

- Standardization of SPICE 3 extensions and XDL
- Standardization of the XDL Windows DLL object
- Separation of Models from SPICE

- Interactive integration with various schematic entry systems
- Integration with various Digital Simulators

- Ability to simulate VHDL and Verilog models

- Integration with Hardware Test, Measurement, and Control
- Analog Hardware Modeler

XDL opens up many areas of simulation to SPICE users that were formally closed because of the complexities of modeling and the lack of access to the operating system. It extends SPICE's capabilities to allow efficient simulation of mixed-signal, mixed-technology, and mixed domain systems. It also allows users to take advantage of the latest software integration technologies like OLE2.0.

Standardization of popular SPICE 3 extensions and a C subroutine capability, such as XDL provides, should greatly benefit the engineering community. This effort has commenced with the availability of the non-proprietary XSPICE program and the independent ABCD committee. SPICE is clearly the de facto standard for analog and mixed signal simulation. With XDL's ability to support other HDLs it appears that this situation is unlikely to change, at least for the analog designer. It remains to be seen how equivalent capabilities will be integrated into other AHDLs, VHDL and Verilog.

[1] "SPICE As An AHDL", Charles Hymowitz, Analog and Mixed Signal Design Conference, July 1994

[2] "Modeling Pentodes", Frederic Broyde, Charles Hymowitz, Intusoft Newsletter, April 1994

[3] "New IBIS Models From Intel", Charles Hymowitz, Intusoft Newsletter, May 1993

[4] "MCT Applications", Charles Hymowitz, Intusoft Newsletter, April 1994

[5] "New AHDL Based on C", Charles Hymowitz, Intusoft Newsletter, 10/94

[6] "Inside OLE2", Kraig Brockschmidt, Microsoft Press, 1994

[7] "Microsoft Developer Network, Windows NT 3.5 Workstation and Win32 SDK", Microsoft Corporation, January 1995