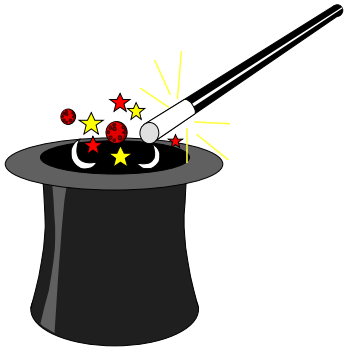


Using PLI 2.0 (VPI) with VCS
Yes, it really works!
(or does it…?)

Stuart Sutherland

Sutherland HDL, Inc.

Portland, Oregon

stuart@sutherland-hdl.com

Purpose

- VCS version 6.1 *claims* to support “PLI 2.0”
  - What is PLI 2.0?
  - What are the advantages of PLI 2.0?
  - How well does VCS support PLI 2.0?
  - Should I write future PLI applications using PLI 2.0?
  - Should I re-write existing PLI applications using PLI 2.0?

- This paper will answer all of these questions

Overview

- The topics discussed are:
  - The history of Verilog PLI standards
  - The advantages and limitations of the PLI 1.0 standard
  - The advantages and limitations of the PLI 2.0 standard
  - The nuts and bolts of PLI 2.0
    - What the PLI 2.0 library looks like
    - How PLI 2.0 PLI applications are linked into simulators
  - The VCS implementation of PLI 2.0
  - The answer to the question…

Should I switch to using PLI 2.0?

What is "PLI 2.0" and "VPI"

- To understand the advantages of PLI 2.0, we need to look at the history of the PLI
  - 1985: The TF library
  - 1988: The ACC library
  - 1990: The OVI PLI 1.0 standard
  - 1993: The OVI PLI 2.0 standard
  - 1995: The IEEE 1364-1995 standard
  - 2001: The IEEE 1364-2001 standard

See the following pages for more details on each of these generations of the PLI

Ah hate history lessons!

1985: The TF Library

- The first generation of the Verilog PLI
  - Can only access information listed in system task/function arguments

```
$my_pli_app(clock, data_bus, "vector_file");
```

system task/function arguments

  - Created to work with one product, Gateway Design Automation's "Verilog-XL" (now a Cadence product)
  - Designed to work with DEC PDP-11 memory architecture
  - Most routines in the library begin with "tf_"
  - Contained in a file called "veriuser.c"
    - Side note: VCS 6.1 now uses the IEEE standard veriuser.c file

1988: The ACC library

- The second generation of the Verilog PLI
  - Can access *structural* information anywhere in the simulation data structure
    - Module instances
    - Primitive instances
    - Nets
    - Delays
  - Created at the request of ASIC vendors to do delay calculation (before the SDF standard existed)
  - Created to work with one product, "Verilog-XL"
  - All routines in the library begin with "acc_"
  - Contained in a file called "acc_user.c"
    - Side note: VCS 6.1 now uses the IEEE standard acc_user.c file

The ACC library cannot access RTL and test bench modeling constructs, such as initial procedures, always procedures, continuous assignments or memory arrays

1990: The OVI PLI 1.0 Standard

- Verilog was released to the public domain in 1990
  - Allowed free use of the Verilog HDL modeling language
  - The PLI TF and ACC libraries were made public as well
- Open Verilog International was formed to maintain and promote the Verilog language
  - OVI labeled the TF and ACC libraries as "**PLI 1.0**"
    - Nothing new was added to the libraries
    - PLI 1.0 is just a term for the 1990 version of the PLI

The public domain documentation on PLI 1.0 was very poor, which resulted in different behavior in different simulators

1993: OVI's PLI 2.0 Standard

- OVI proposed a new PLI standard, called **PLI 2.0**
  - Combined the functionality of TF and ACC routines into a single library
  - Radically different than the TF and ACC libraries
    - 220+ routines reduced to about 20 routines
    - Added access to RTL models, memory arrays, etc.
    - Consistent, easy to use syntax and semantics
    - Thorough documentation

- PLI 2.0 was designed to replace PLI 1.0
  - Not backward compatible (on purpose)
    - Used same function and constant names as the ACC library, but with different definitions
  - PLI 2.0 and PLI 1.0 applications could not be used together

1995: IEEE 1364-1995

- The IEEE formally standardized Verilog in 1995
  - Primary goal: document the way Verilog was being used at that time
  - No enhancements were considered for the 1995 standard
- Both PLI 1.0 and PLI 2.0 were standardized
  - PLI 1.0 for backward compatibility
  - PLI 2.0 because it was so much better than 1.0
- OVI's PLI 2.0 was rewritten as the "VPI" library
  - Names begin with "vpi_"
  - No conflicts with PLI 1.0's ACC library

The terms "PLI 1.0" and "PLI 2.0" do not exist in the IEEE standard — it is one PLI with three libraries!

2001: IEEE 1364-2001

- Verilog-2001 adds over 45 major and minor enhancements to the Verilog HDL
  - Multi-dimensional arrays of any data type
  - Re-entrant tasks, recursive functions
  - Configurations and generate blocks
  - Attributes (to replace those `//synopsys` pragmas)

VCS 6.1 and Presto support many of these new features (see SNUG-2001 paper by Don Mills and Stuart Sutherland)

```
(* fullcase=TRUE, parallelcase=FALSE *) case (state)
```

- The PLI was enhanced to support the new HDL features
  - Access to all dimensions of arrays
  - Access to task and function call stacks
  - Access to the values of attributes

OK, But Why Use the VPI Library?

So what's the point of this history lesson?

1. To understand why the TF and ACC libraries are obsolete
2. To appreciate the advantages of the VPI library

Ah still hate history!

TF Library
Advantages and Limitations

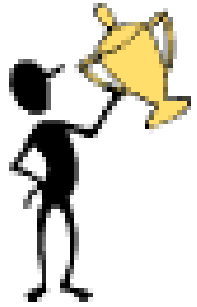
- Advantages
  - VCS is often 2x to 5x faster when only the TF library used
    - Limited access to simulation data structure = better optimization
- Limitations
  - Cannot analyze internal simulation data structure
  - Limited support for vectors greater than 64-bits wide
  - Many TF routines will not work on 64-bit operating systems
  - 110+ routines with inconsistent syntax and semantics
  - Non portable
    - Different simulators implement TF behavior differently
    - Applications may need to be “tweaked” to work with each simulator
  - New Verilog-2001 features are not supported 

ACC Library Advantages and Limitations

- Advantages
  - Access to structural portions of simulation data structure
  - Useful for many types of applications
    - Power usage, waveform displays, signal tracing, hierarchy trees, …
- Limitations
  - Cannot access RTL and test bench portions of Verilog HDL
  - Cannot access memory arrays
  - 110+ routines with inconsistent syntax and semantics
  - Poorly documented (no clue on data structure linked list)
  - Non portable
    - Different simulators implement ACC behavior differently
  - New Verilog-2001 features are not supported

VPI (PLI 2.0) Advantages and Limitations

- Advantages
  - 37 routines in entire library
  - Complete superset of TF/ACC libraries
  - Full access to structural, RTL and test bench constructs
  - Consistent, simple syntax and semantics
  - Encourages developers to write better, more efficient code
  - Portable to 64-bit operating systems
  - Well documented access methodologies
    - All simulators can implement the same way
  - And perhaps the most important advantage is…



- Limitations
  - Cannot be optimized as efficiently as TF library
  - Requires a better knowledge of structured C programming

The Most Important Advantage of the VPI Library

- **All the new features in Verilog-2001 are supported by the VPI library of the PLI**

  - The official IEEE 1364 policy is…
    - As Verilog evolves, only the VPI library will be extended
    - The TF and ACC libraries will be maintained, but not enhanced

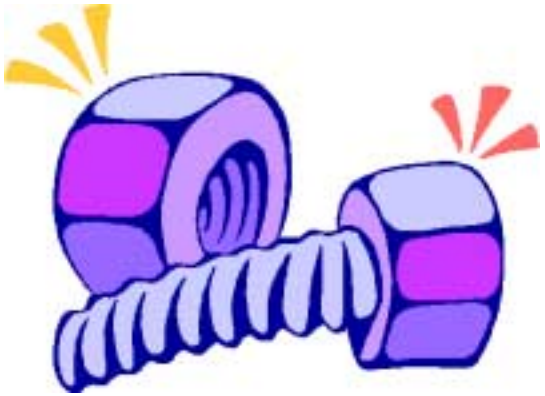
The VPI Library Is The Future!

- **Verilog-2001 is only supported by the VPI library**
  - It is not logical to continue to extend the TF and ACC libraries
    - Too large, too awkward, too inconsistent too platform dependent, … too obsolete

The TF and ACC libraries are cave man technology!

The nuts & bolts of the VPI library

(a brief look at how VPI routines work)

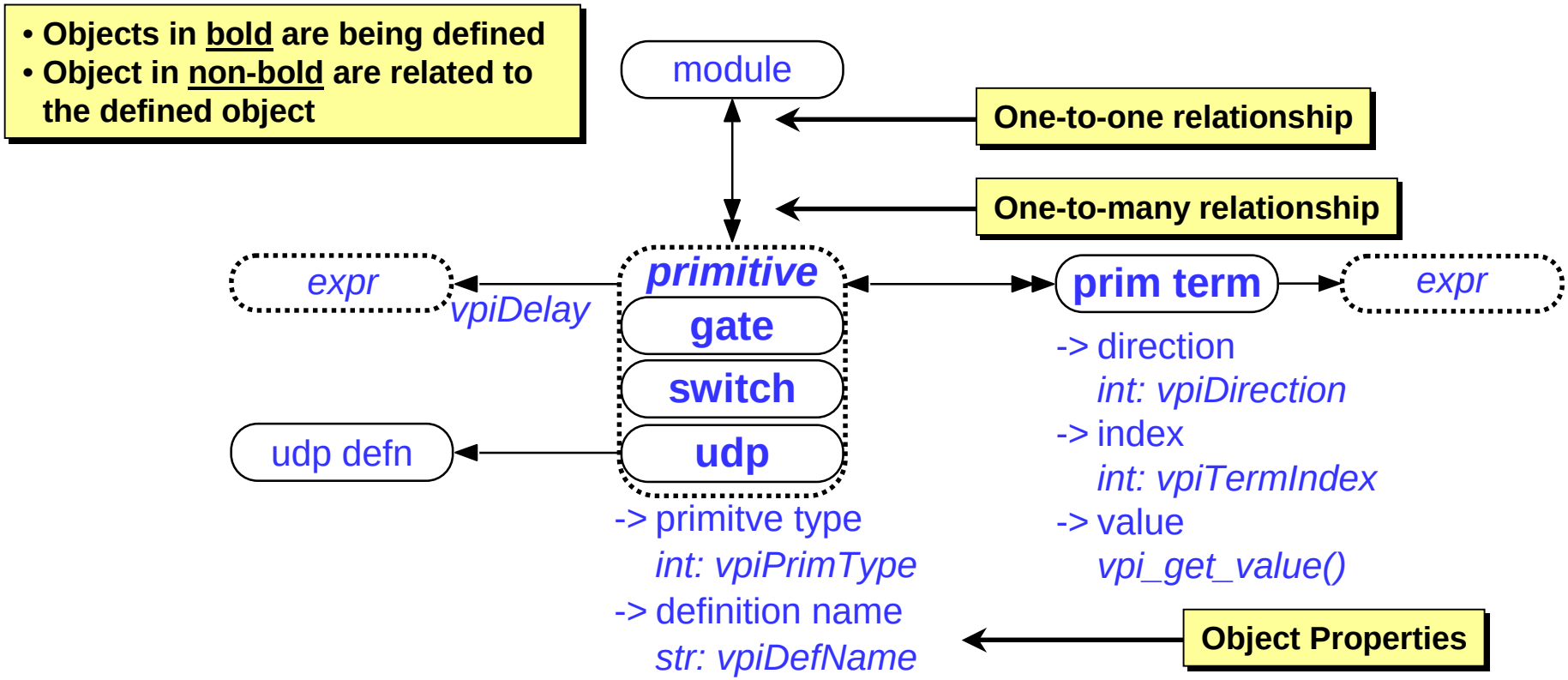


The Nuts and Bolts of the VPI Library

- 37 C functions, with structure and constant definitions
  - Locate object
  - Read and modify values
  - Synchronize to simulation activity
  - Utility routines for file I/O, controlling simulation, etc.
- Object handles
  - A pointer to a structure with information about an object
    - Module instance, net, RTL statement, memory array, etc
  - Each object is connected to one or more other objects
    - Verilog simulation data structures are linked lists of objects
- Object diagrams
  - Documents the properties and connections of an object

VPI Object Diagrams

- Document each object that exists in Verilog simulation
  - Shows what information can be accessed
  - Shows how to "traverse" the data structure to another object



Using the VPI Library

- Obtaining handles
  - The routine vpi_handle() obtains a handle to any object with one-to-one relationship
  - The routines vpi_scan() / vpi_iterate() obtain handles to all objects with one-to-many relationship
    - These 3 routines replace over 80 routines in the ACC library
    - Any type of object can be accessed: structural, RTL, behavioral
- Reading values
  - The routine vpi_get() reads any integer or boolean property
  - The routine vpi_get_str() reads any string property
  - The routine vpi_get_value() reads any logic value
  - The routine vpi_get_delays() reads any delay value
    - These 4 routines replace over 60 routines in the TF/ACC libraries

Synchronizing to Simulation

- "Simulation callbacks" are used to synchronize a PLI application to simulation
  - Start of simulation
  - End of simulation
  - Simulation breakpoint
  - Logic value change
  - End of a simulation time step

Also possible with TF/ACC libraries

  - Beginning of a simulation time step
  - End of blocking assignment event list
  - End of non-blocking assignment event list
  - Execution of an RTL statement

Unique to VPI library

Fastening the nuts to the bolts

Parts of a PLI 1.0 Application

- A PLI 1.0 application comprises:
  - A system task/function name
  - One or more C functions

```
always @(posedge clk)
  number = $my_random(seed);
```

Each PLI routine is a C function

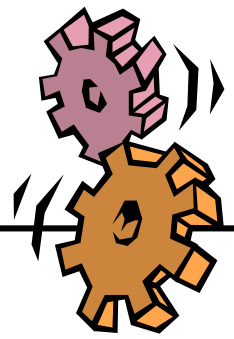
A calltf routine
- **Executes functionality of a system task/function**
- **Called during simulation**

A checktf routine
- **Used to verify usage of a system task/function**
- **Called before simulation time 0**

A sizetf routine
- **Specifies return bit width of a system function**
- **Called before simulation time 0**

A misctf routine
- **Synchronizes application to simulation events**
- **Called for miscellaneous reasons**

The PLI 1.0 Interface



- The PLI 1.0 Interface:
  - Specifies the system task/function name (begins with a $)
  - Specifies if the application is a task or a function
  - Specifies the names of the C functions for the *checktf*, *sizetf*, *calltf* and *misctf* routines
  - Specifies a “user-data” integer value
- The IEEE standard:
  - Defines what the interface should specify
  - Does **not** define how the information should be specified
  - Every Verilog simulator has a different way of specifying PLI 1.0 application information
    - Some simulators use a “veriuser.c” file
    - VCS uses PLI table “.tab” files

The VCS .tab File

- All the information about the PLI application is specified in a separate, user-defined file

```
$list_nets check=check_args call=listnets misc=cleanup data=0
$get_value check=getvalcheck call=getvalcall size=64 data=5 +acc=read:top.i1+
```

Where do I find all this information?
What if I specify something wrong?

Note: VCS does not support sizetf routines. Instead, the system function return size is hard coded in the .tab file

- **Disadvantages of the PLI 1.0 interface**
  - The user must know detailed information about the application
  - The user can inadvertently specify incorrect information
  - The interface is different for every simulator

Parts of a VPI Application

- A VPI application comprises:
  - A system task/function name
  - One or more C functions

Each PLI routine is a C function

```
always @(posedge clk)
  number = $my_random(seed);
```

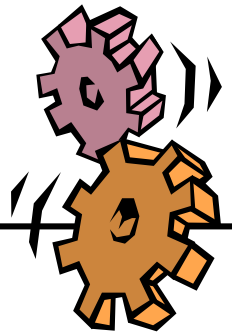
A calltf routine
- **Executes functionality of a system task/function**
- **Called during simulation**

A compiletf routine
- **Used to verify usage of a system task/function**
- **Called before simulation time 0**

A sizetf routine
- **Specifies return bit width of a system function**
- **Called before simulation time 0**

Any number of simulation callback routines
- **Synchronizes application to simulation events**
- **Called for miscellaneous reasons**

The VPI Interface



- The VPI Interface:
  - Specifies the system task/function name (begins with a $)
  - Specifies if the application is a task or a function
  - Specifies the names of the C functions for the checktf, sizetf, calltf and misctf routines
  - Specifies a “user-data” *pointer* value
- The IEEE standard:
  - Defines what the interface should specify
  - Defines *how* the information should be specified
  - Every IEEE compliant Verilog simulator will specify the information the same way!

Standardizing the PLI interface is a big advantage!
(and another major advantage will be shown in a moment…)

The VPI Interface (continued)

- The VPI interface mechanism "registers" VPI applications in a system task/function "registry"
  - The PLI application includes a C "register function" that:
    - Allocates an s_vpi_systf_data structure
    - Calls the routine vpi_register_systf() with a pointer to the data structure
  - The end-user simply informs the simulator of the name of the register function
    - The IEEE standard recommends simulators provide an array called vlog_startup_routines
    - Any number of register functions may be listed in the array

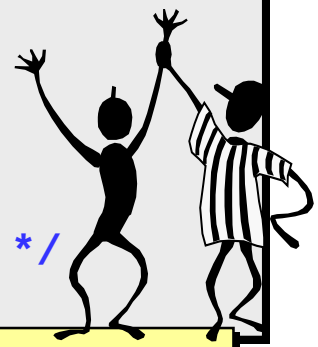
Simulators may provide alternate methods to specify the register functions, such as with invocation options

An Example VPI Register Function

- The register function:
  - Can be in the same file as the application
  - Can be in a separate file

```
void listnets_register()
{
  s_vpi_systf_data  tf_data;    /* allocate a systf data structure */
  tf_data.type        = vpiSysTask;
  tf_data.sysfunctype = 0;
  tf_data.tfname      = "$list_nets";
  tf_data.calltf      = listnets;
  tf_data.compiletf   = check_args;
  tf_data.sizetf      = NULL;
  tf_data.user_data   = NULL;

  vpi_register_systf(&tf_data); /* register the system task */
}
```



ADVANTAGE: All the end-user needs to know is the name of this function!

- **The end-user does not need to know—and cannot mess up—the internal information about the PLI application**

The Golden Question

Does VCS really support PLI 2.0 (the VPI library)?

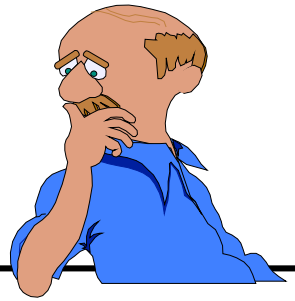
VCS 6.1 and the VPI Library

- ***Yes, VCS 6.1 really does support the VPI library!***
  - *Most* of the Verilog-2001 VPI library is supported
  - The subset is very usable
  - Unsupported routines (in VCS 6.1)
    - Simulation control (stop, finish, etc.)
    - Save/restart
    - Some minor limitations on reading and modifying delays
    - Some limitations on reading and modifying logic values
    - Some limitations on synchronizing to simulation

Synopsys will continue to add support for the remainder of the VPI library in future releases of VCS

- Enabling the VPI library with VCS 6.1
  - The **+vpi** invocation option must be specified to enable VPI

The VCS Version Of the VPI Interface



- VCS 6.1 does ***not*** implement the IEEE standard VPI interface
  - The old TF/ACC mechanism (the .tab file) is used
    - VPI access can be limited, using the same controls as ACC
    - Allows run-time performance optimization — very desirable

The author considers this non-compliance a major limitation

- The end-user must know the internal application information
- The user-data pointer cannot be used as intended
- The return size of system functions cannot be calculated
- PLI applications cannot be registered for execution prior to the start of simulation using a standard register function
  - There is a work around for this last limitation, but it may require modifying existing VPI applications

Should You Switch to the VPI?

- Now you know the history
  - The VPI library is the latest generation of the PLI
- Now you know the VPI advantages
  - ✓Only 37 routines
  - ✓Simple, consistent syntax
  - ✓Well documented access methodologies
  - ✓Full access to structural, RTL and test bench constructs
  - ✓Supports the new Verilog-2001 constructs
  - ✓Standard interface method (except in VCS 6.1)
  - ✓Requires more structured, proper coding styles
  - ✓Portable to other simulators
  - ✓Portable to 64-bit operating systems

With all these advantages…

Are There Any Reasons Not to Use the VPI Library?

- Portability—will VPI applications work on all your simulators?
  - All major simulators now support the VPI
  - Some minor simulators do not support the VPI
- Knowledge—does anyone at your company know the VPI?
  - The VPI library is very different than the TF/ACC libraries
  - Engineers will need to use a more structured coding style
  - The VPI is easier to learn and use (if you know C)
- Performance—will VPI applications slow down simulation?
  - Compilers can better optimize TF applications
    - Only VCS and NC take advantage of this
    - Proper specification of access limitations can reduce VPI overhead
  - The VPI library “encourages” a more efficient coding style

Is The PLI Obsolete?

- There are a number of extensions to Verilog to allow using C without a procedural interface
  - DirectC
    - Proprietary extensions to the Verilog HDL, by Synopsys
    - Allows Verilog and C functions to be intermixed
    - Only works with VCS
  - SystemVerilog
    - A proposed Accellera standard
    - Adds many C constructs and other extensions to Verilog-2001
    - Hoped to become the next IEEE Verilog standard
  - SystemC
    - An open-source standard
    - Allows hardware to be modeled in C++, without using an HDL
    - Intended for system-level modeling and verification

Is The PLI Obsolete?

- When is it best to use an alternative to the PLI?
  - Many PLI applications do not need a procedural interface
    - C-language bus functional models
    - Embedded software in System-On-Chip designs
    - Abstract constructs such as structures, records and pointers
  - DirectC and SystemVerilog are good alternatives for this
- When is it best to use the PLI?
  - The PLI provides access to the simulation data structure
    - Find objects, trace signals, traverse design hierarchy, modify delays
  - The PLI allows synchronization with simulation events
    - End of blocking events, start of nonblocking events, etc.
  - The PLI protects the simulation data structure
    - Applications access information through an interface, not directly

Conclusion

The Original Question:

Should I switch to using PLI 2.0?

Yes!

- There are many advantages to the VPI library
- Features in Verilog-2001 and beyond can only be accessed using the VPI
- There are very few exceptions for when the TF/ACC libraries are preferred

Follow-up Questions?

Additional Resources: Verilog Books

- IEEE Std 1364-2001
  - The official Verilog and PLI standard
- The Verilog PLI Handbook
  - by Stuart Sutherland—comprehensive book on using the PLI
  - Second edition covers new Verilog-2001 features
- Verilog PLI Quick Reference Guide, Verilog-2001 version
  - by Stuart Sutherland—lists all the PLI libraries, structures, constants, etc..
- Verilog-2001: A Guide to the New Features in Verilog
  - by Stuart Sutherland—more detailed than this presentation
- www.verilog-2001.com
  - Information about the Verilog-2001 standard

check www.sutherland-hdl.com for a list of over 35 Verilog books

About Stuart Sutherland and Sutherland HDL, Inc.

Sutherland HDL
Training engineers to be HDL wizards

- Sutherland HDL, Inc. (founded 1992)
  - Provides expert Verilog HDL, Verilog PLI and VHDL Training
  - Provides Verilog HDL, Verilog PLI and VHDL consulting services
  - Located near Portland Oregon, World-wide services
- Mr. Stuart Sutherland
  - Over 14 years experience with Verilog
  - Worked as a design engineer on military flight simulators
  - Senior Applications Engineer for Gateway Design Automation, the founding company of Verilog
  - Author of the popular “Verilog HDL Quick Reference Guide” and “The Verilog PLI Handbook”
  - Chair of the PLI task force, IEEE 1364 Verilog standards group
  - Editor of the SystemVerilog manual, Accellera HDL+ committee