SUTHERLAND HDL
Training Engineers to be HDL wizards

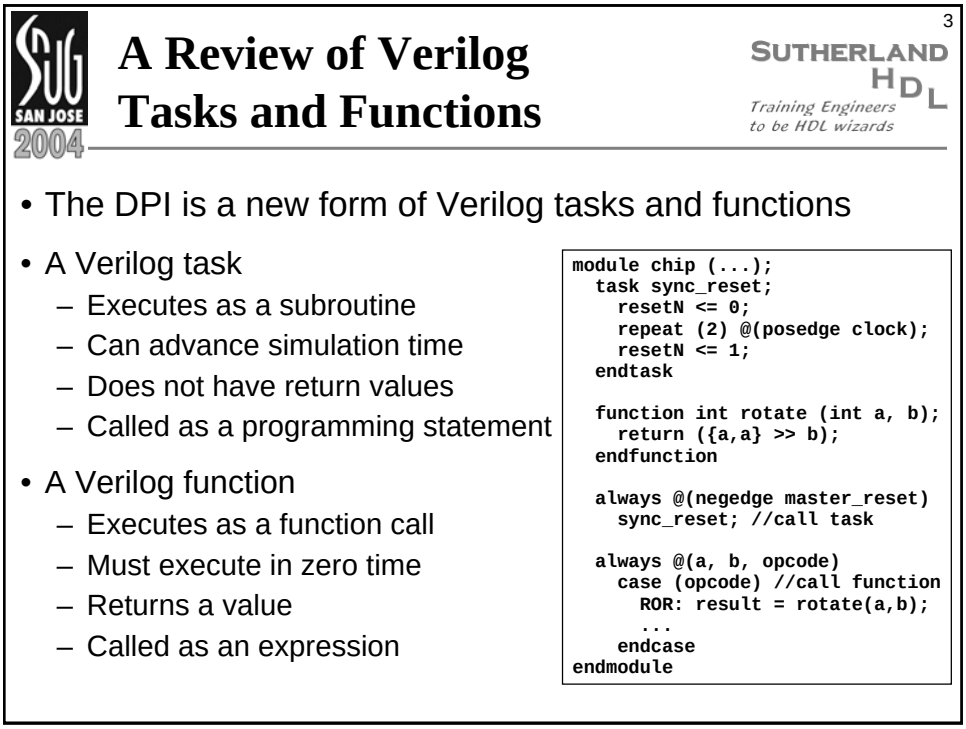
The Verilog PLI Is Dead (maybe)...
Long Live The SystemVerilog DPI!

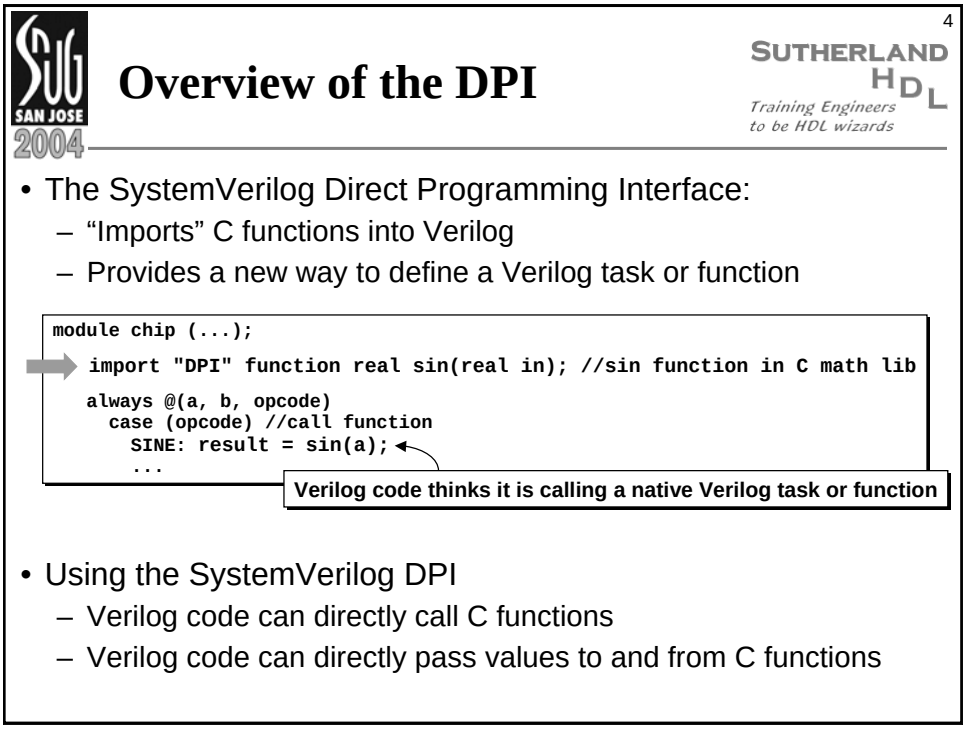
Stuart Sutherland
Sutherland HDL, Inc.
Portland, Oregon
stuart@sutherland-hdl.com

SUTHERLAND HDL
Training Engineers to be HDL wizards

Objectives

- Introduce the SystemVerilog DPI
- Verilog PLI strengths and weaknesses
- Show how the SystemVerilog DPI works
- Compare and contrast PLI and DPI
- Decide if the PLI is finally dead


3
A Review of Verilog Tasks and Functions
SUTHERLAND HDL
Training Engineers to be HDL wizards
• The DPI is a new form of Verilog tasks and functions
• A Verilog task
– Executes as a subroutine
– Can advance simulation time
– Does not have return values
– Called as a programming statement
• A Verilog function
– Executes as a function call
– Must execute in zero time
– Returns a value
– Called as an expression
module chip (...);
task sync_reset;
resetN <= 0;
repeat (2) @(posedge clock);
resetN <= 1;
endtask
function int rotate (int a, b);
return ({a,a} >> b);
endfunction
always @(negedge master_reset)
sync_reset; //call task
always @(a, b, opcode)
case (opcode) //call function
ROR: result = rotate(a,b);
...
endcase
endmodule


4
Overview of the DPI
SUTHERLAND HDL
Training Engineers to be HDL wizards
• The SystemVerilog Direct Programming Interface:
– "Imports" C functions into Verilog
– Provides a new way to define a Verilog task or function
module chip (...);
import "DPI" function real sin(real in); //sin function in C math lib
always @(a, b, opcode)
case (opcode) //call function
SINE: result = sin(a);
...
Verilog code thinks it is calling a native Verilog task or function
• Using the SystemVerilog DPI
– Verilog code can directly call C functions
– Verilog code can directly pass values to and from C functions

5

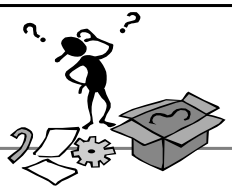
What's Next

SUTHERLAND HDL
Training Engineers to be HDL wizards

- ✓ Introduce the SystemVerilog DPI
- ❑ Verilog PLI strengths and weaknesses
- ❑ Show how the SystemVerilog DPI works
- ❑ Compare and contrast PLI and DPI
- ❑ Decide if the PLI is finally dead

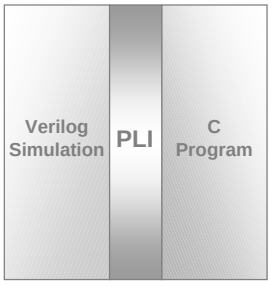
6

Verilog PLI Details



SUTHERLAND HDL
Training Engineers to be HDL wizards

- Need to know how PLI works in order to compare to DPI...
- The Verilog PLI is a *simulation interface*
  - Reads/modifies simulation data structures
  - Does not read Verilog source code
  - Does not work with synthesis compilers or other tools



The PLI is a protecting layer between *user programs* and *simulation data structure*

- *Indirect access* through PLI libraries
  - C program cannot directly read or modify the simulation data
  - Protects the simulator from bad C programs
  - Protects C programs from bad Verilog code

7

Overview of the Verilog PLI

SUTHERLAND HDL
Training Engineers to be HDL wizards

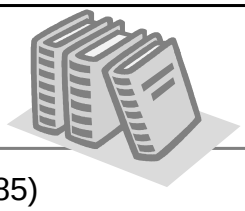
- The PLI allows users to extend Verilog by creating "user-defined system tasks and system functions"
  - Must begin with a dollar sign ($)
  - Called the same as with Verilog tasks and functions

```
always @(a, b, opcode)
  case (opcode) //call function
    SINE: result = $sine(a); // call a PLI application
```

- A system task/function invokes a "calltf routine"
  - A user-defined C function associated with the task/function name
    - Can indirectly read system task/function argument values
    - Can indirectly return values back to simulation
- Defining system task/function names and calltf routines is:
  - Complex
  - Varies from one simulator to another

8

PLI Libraries



SUTHERLAND HDL
Training Engineers to be HDL wizards

- TF library (introduced in 1985)
  - Provides access to system task/function arguments
  - Provides synchronization to simulation time and events
- ACC library (introduced in 1989)
  - Extends TF library to access to design structure
  - Created to enable ASIC delay and power calculation
- VPI library (introduced in 1995)
  - Superset of TF and ACC libraries
  - Adds access to behavioral and RTL models
- NOTE! The IEEE is considering deprecating (removing) the TF and ACC libraries from the 1364 standard
  - Only the VPI library supports the new Verilog-2001 features
  - Only the VPI library will support SystemVerilog enhancements

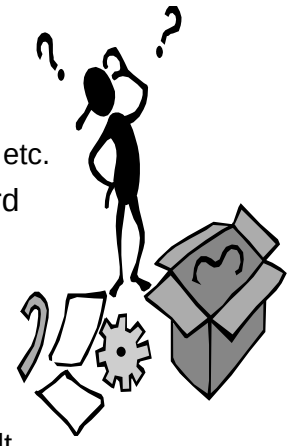
The Power of the PLI

SUTHERLAND HDL
Training Engineers to be HDL wizards

- Supports system task/function arguments of any type
  - Data types, instance names, scopes, null arguments, etc.
  - Allows a variable number of task/function arguments
- Safe conversion of Verilog values to/from C values
  - Automatically converts any Verilog type to any C type
- Can find any modeling object anywhere in the data structure
- Can synchronize to simulation time
  - After blocking assignments, after nonblocking assignments, after all events, at future simulation times, etc.
- Can synchronize to any type of simulation event
  - Start, stop (breakpoints), finish, save, restart, logic value changes, strength level changes, RTL statement execution, etc.
- Supports multiple instances of system tasks/functions

The Disadvantages of the Verilog PLI

SUTHERLAND HDL
Training Engineers to be HDL wizards



- Writing PLI applications is difficult to do
  - Must learn weird PLI terminology
  - Must learn what's in the PLI libraries
  - Must create checktf routines, calltf routines, etc.
- Linking PLI applications to simulators is hard
  - Multiple steps involved
    - Different for every simulator
  - Who does the linking...
    - The design engineer?
    - A CAE tool administrator?
  - Managing multiple PLI applications is difficult
- PLI code is seldom binary compatible
  - Must re-compile for every simulator

11

What's Next

SUTHERLAND HDL
Training Engineers to be HDL wizards

- ✓ Introduce the SystemVerilog DPI
- ✓ Verilog PLI strengths and weaknesses
- ❑ Show how the SystemVerilog DPI works
- ❑ Compare and contrast PLI and DPI
- ❑ Decide if the PLI is finally dead

12

The DPI Makes Calling C Easy!

SUTHERLAND HDL
Training Engineers to be HDL wizards

- The Direct Programming Interface makes it very simple for Verilog code to call C functions

```
module chip (...);
   import "DPI" function real sin(real in); //sin function in C math lib
   always @(a, b, opcode)
     case (opcode) //call function
       SINE: result = sin(a);
       ...
```

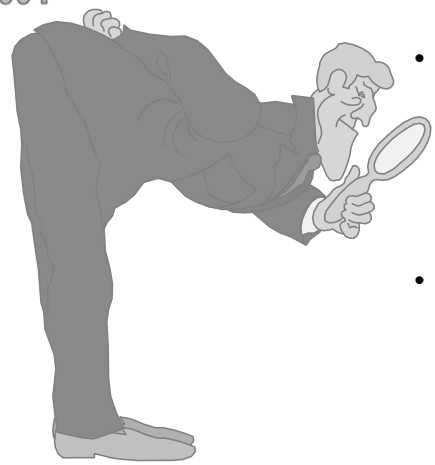
Verilog code directly calls the C function

- Can directly call C functions from Verilog
  - Do not need to define complex PLI system tasks/functions
- Can directly pass values to and from C functions
  - Do not need the complex PLI libraries to read/write values

13

A Closer Look at the SystemVerilog DPI

SUTHERLAND HDL
Training Engineers to be HDL wizards



- The Direct Programming Interface originates from:
  - The Synopsys VCS DirectC interface
  - The Co-Design Cblend interface (now owned by Synopsys)
- The Accellera standards committee
  - Merged the features of each donation
  - Added more functionality
  - Ensured full compatibility with the IEEE 1364 Verilog standard
  - Added rules for binary compatibility between simulators

14

More on DPI Import Declarations

SUTHERLAND HDL
Training Engineers to be HDL wizards

```
module chip (...);
  import "DPI" function real sin(real in); //sin function in C math lib
```

- Import declarations can be anywhere a function can be defined
  - Within a Verilog module
  - Within a SystemVerilog interface
  - Within a SystemVerilog package
  - Within a SystemVerilog “compilation unit”
- Import declarations must have a prototype of the arguments
  - Must exactly match the number of arguments in the C function
  - Must specify compatible data types (details on a later slide)
- The same C function can be imported in multiple locations
  - Each prototype must be exactly the same
  - A better method is to define one import in a package

15

Imported Function Arguments

SUTHERLAND HDL
Training Engineers to be HDL wizards

- C functions can be imported as a Verilog task or function

```
import "DPI" function real sin(real in); //sin function in C math lib

import "DPI" task file_write(input string data, output reg status);
```

- The C function arguments can be imported as input, output or inout (bidirectional)
  - Arguments are assumed to be inputs unless declared otherwise

16

Task/Function Argument Data Types

SUTHERLAND HDL
Training Engineers to be HDL wizards

- The import declaration ***must*** specify SystemVerilog data types that are compatible with the actual C function data types

	SystemVerilog Data Type	C Data Type
SystemVerilog types that map directly to C types	byte	char
	shortint	short int
	int	int (32-bit)
	longint	long long
	real	double
	shortreal	float
	chandle	void*
	string	char*
	enum (using default int type)	int
SystemVerilog types that require extra coding in C	bit (2-state type, any vector size)	abstract array of int types
	logic (4-state type, any vector size)	abstract aval/bval pair arrays (like PLI)
	packed array	abstract representation
	unpacked array	abstract representation

17

Using More Complex Data Types

SUTHERLAND HDL
Training Engineers to be HDL wizards

- Structures and unions
  - Can be passed to equivalent C structures and unions
  - Must use compatible types within the structure or union
- Arrays (unpacked)
  - Can be passed to equivalent C arrays
  - Must use compatible types within the structure or union
  - The array indexing might change (C must always starts with 0)
- Vectors of 2-state data types (packed arrays of bit type)
  - Must be converted to integer arrays in C
  - Requires lots of extra coding in C
- Verilog 4-state data types (packed arrays of reg, wire, etc.)
  - Uses the Verilog PLI aval/bval encoding
  - Requires lots of extra coding in C

18

Compatibility Warning!

SUTHERLAND HDL
Training Engineers to be HDL wizards

Warning! Warning!

- It is the user's responsibility to correctly declare compatible data types in an import statement
  - Improper declarations can lead to unpredictable run-time behavior
  - The DPI does not check for type compatibility
  - The DPI does not provide a way to check and adjust for the data types on the Verilog side (the PLI can do this)

19

DPI Function Return Values

SUTHERLAND HDL
Training Engineers to be HDL wizards

- A C function return value must be compatible with the basic SystemVerilog data types

Imported Return Type	C Function Return Type
byte	char
shortint	short int
int	int (32-bit)
longint	long long
real	double
shortreal	float
chandle	void*

Imported functions cannot return 1-bit values, vectors, structures, unions, or enumerated types

Warning!

  - It is the user's responsibility to correctly declare an import statement that matches the C function return value
    - Incorrect import declarations can lead to unexpected behavior

20

Pure, Context and Generic C Functions

SUTHERLAND HDL
Training Engineers to be HDL wizards

- A pure function result depends solely on the function inputs

```
import "DPI" pure function real sin(real, in);
```

  - Must have a return value; cannot have output or inout arguments
  - Cannot call any other functions or use static or global variables
  - Can be highly optimized for simulation performance **Advantage!**
- A context function is aware of the Verilog scope in which it is imported

```
import "DPI" context task print(input int file_id, input bit [127:0] data);
```

  - Can be a void function, and can have output and inout arguments
  - Can call functions from C libraries (for file I/O, etc.)
  - Can call many of the functions in the PLI libraries (more on this later)
- A generic function is one that is not declared as pure or context
  - Can be a void function, and can have output and inout arguments
  - Can call other C functions
  - Cannot call most functions in the PLI libraries

21

Declaration Warning!

SUTHERLAND HDL
Training Engineers to be HDL wizards

You must do this right!

- It is the user's responsibility to correctly declare pure and context functions
  - The DPI does not check for proper declarations
  - Improper declarations can lead to unpredictable simulation behavior
  - Improper declarations can lead to software crashes

22

Exporting Verilog Tasks and Functions

SUTHERLAND HDL
Training Engineers to be HDL wizards

- Verilog tasks and function can be exported to C
  - Exporting allows C code to call Verilog code
    - C thinks it is calling a native C function
    - C functions can synchronize to Verilog time by calling a Verilog task with time controls

```
module chip (...);
  task sync_reset(inout resetN);
    resetN <= 0;
    repeat (2) @(posedge clock);
    resetN <= 1;
  endtask

  export "DPI" sync_reset;
  ...
endmodule
```

```
function void C_model()
{
  ...
  sync_reset(rst); // call Verilog task
  ...
}
```

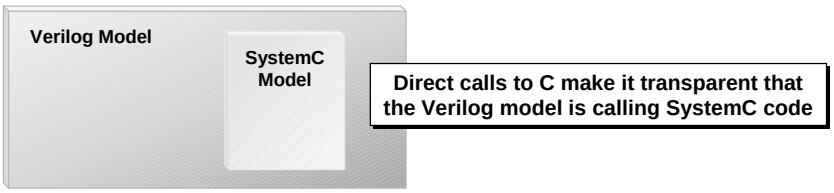
Export declarations do not have argument prototypes

- C calling Verilog tasks and functions is unique to the DPI!
  - There is no equivalent in the PLI

23

Using the DPI with SystemC

- The DPI can be used to interface SystemC models with Verilog models

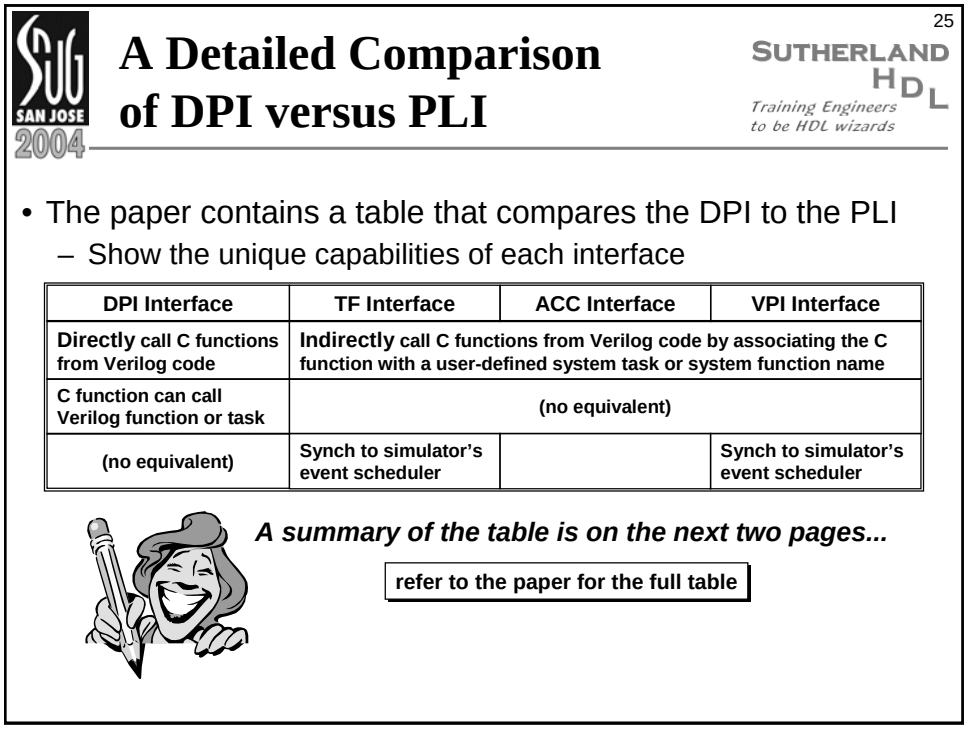


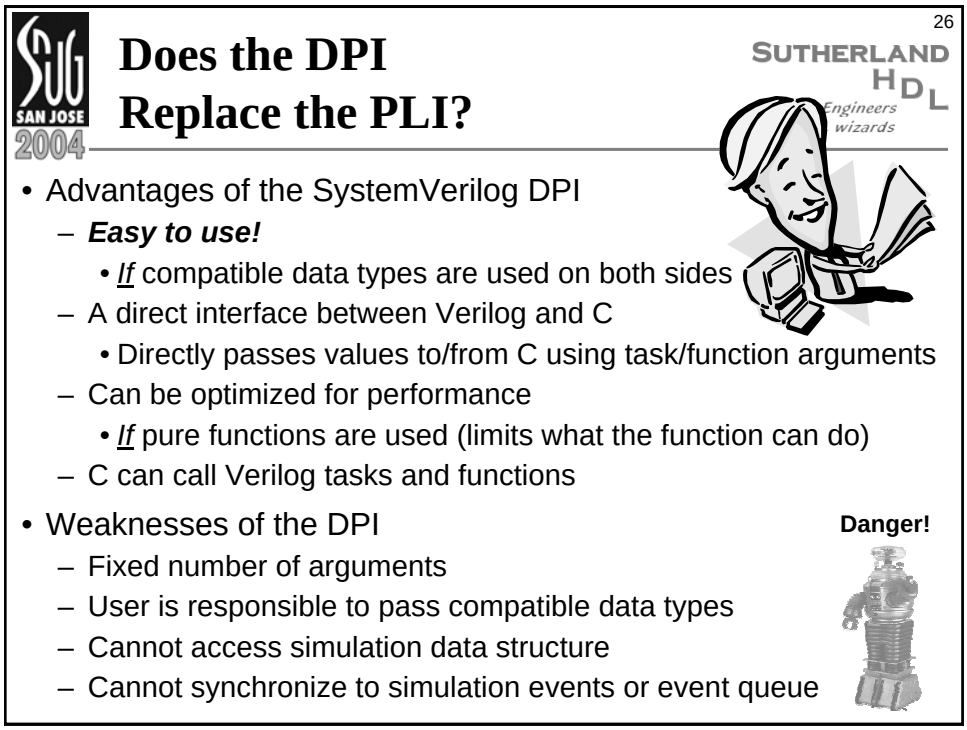
  - Verilog code can directly call an imported C function that connects to the SystemC model
  - Values can be directly passed to and from the SystemC model
- Using the DPI to interface Verilog with SystemC
  - Eliminates the need for a complex PLI interface
  - Eliminates the need for proprietary, non-portable interfaces

24

What's Next

- ✓ Introduce the SystemVerilog DPI
- ✓ Verilog PLI strengths and weaknesses
- ✓ Show how the SystemVerilog DPI works
- ❑ Compare and contrast PLI and DPI
- ❑ Decide if the PLI is finally dead


25
A Detailed Comparison of DPI versus PLI
SUTHERLAND HDL
Training Engineers to be HDL wizards
• The paper contains a table that compares the DPI to the PLI
– Show the unique capabilities of each interface
DPI Interface | TF Interface | ACC Interface | VPI Interface
Directly call C functions from Verilog code | Indirectly call C functions from Verilog code by associating the C function with a user-defined system task or system function name
C function can call Verilog function or task | (no equivalent)
(no equivalent) | Synch to simulator's event scheduler | | Synch to simulator's event scheduler
A summary of the table is on the next two pages...
refer to the paper for the full table


26
Does the DPI Replace the PLI?
SUTHERLAND HDL
Engineers wizards
• Advantages of the SystemVerilog DPI
– Easy to use!
• If compatible data types are used on both sides
– A direct interface between Verilog and C
• Directly passes values to/from C using task/function arguments
– Can be optimized for performance
• If pure functions are used (limits what the function can do)
– C can call Verilog tasks and functions
• Weaknesses of the DPI
– Fixed number of arguments
– User is responsible to pass compatible data types
– Cannot access simulation data structure
– Cannot synchronize to simulation events or event queue
Danger!

27

Does the DPI Replace the PLI?

SUTHERLAND HDL
Training Engineers to be HDL wizards

- Advantages of the Verilog PLI
  - Allows access to entire simulation data structure
    - Find any Verilog object, anywhere in the design
  - Can synchronize to simulation
    - Synch to time (before or after blocking assignments, ...)
    - Synch to value changes, breakpoints, finish, ...
  - Provides an indirect access between Verilog and C
    - Protects simulation data structure from user C code
    - Automatically converts values between Verilog and C types
- Disadvantages of the Verilog PLI
  - Difficult to learn — even simple things are hard
  - Not binary compatible — different for every simulator
  - Can slow down simulator performance (significantly)

28

Using the DPI to Simplify the PLI

SUTHERLAND HDL
Training Engineers to be HDL wizards

- DPI capabilities can be extended by using the PLI libraries
  - A DPI context function can call many of the PLI library functions
    - Eliminates creating a system task/function name (e.g. $sine)
    - Eliminates the complex PLI binding mechanism
- NOTE: The DPI context is not the same as the PLI context
  - DPI context is the scope in which the import declaration occurs
    - Matches the behavior of native Verilog task and functions
  - PLI context is the scope in which a system task is called
  - Context functions cannot fully utilize the PLI libraries
    - Cannot use PLI checktf and misctf routines
    - Cannot access all objects in the simulation data structure

29

What's Next

SUTHERLAND HDL
Training Engineers to be HDL wizards

- ✓ Introduce the SystemVerilog DPI
- ✓ Verilog PLI strengths and weaknesses
- ✓ Show how the SystemVerilog DPI works
- ✓ Compare and contrast PLI and DPI
- ❑ Decide if the PLI is finally dead

30

Conclusion

SUTHERLAND HDL
Training Engineers to be HDL wizards

- The Verilog PLI and the SystemVerilog DPI are both needed
- Use the DPI to
  - Directly call C functions that do not need to access the simulation data structure
  - Directly call PLI applications that only need limited access to the simulation data structure
  - Interface to C and SystemC models
- Use the PLI to
  - Access any object anywhere in the simulation data structure
  - Synchronize to the simulation event queue
    - Blocking assignments, nonblocking assignments, etc.
  - Synchronize to simulation events
    - Simulation start, stop, finish, save, restart, reset, etc.

31

Is the PLI Dead?

SUTHERLAND HDL
Training Engineers to be HDL wizards

- We cannot chant:

  "The Verilog PLI is dead...long live the SystemVerilog DPI"

  – The DPI is great, but...
  – The PLI has many capabilities not in the DPI

- The correct chant is:

  "The PLI TF/ACC libraries are dead... long live the PLI VPI and the SystemVerilog DPI, together!"

  – The DPI greatly simplifies many applications
  – The VPI replaces the old TF and ACC libraries

32

Questions?

SUTHERLAND HDL
Training Engineers to be HDL wizards

33

Supplemental: About the Author

- Involved with Verilog since 1988
- Considered a Verilog PLI expert
  - Wrote that 800 page book on the PLI
- Member of IEEE 1364 standards group since beginning
  - Co-chair of the Verilog PLI task force
  - Editor of the PLI sections of the standard
- Member of Accellera SystemVerilog standards group
  - Editor of the SystemVerilog standard
- Develops and presents expert-level Verilog, PLI and SystemVerilog training courses

34

Supplemental: PLI Standards

- OVI (now Accellera) PLI 1.0 (1990)
  - Standardized the original TF and ACC libraries
- OVI PLI 2.0 (1993)
  - Intended to replace PLI 1.0
  - Not backward compatible, so never implemented in simulators
- IEEE 1364-1995
  - Standardized PLI 1.0 as TF and ACC libraries
  - Rewrite of PLI 2.0 to be backward compatible, called VPI library
- Accellera SystemVerilog 3.1 (2003)
  - Adds the Direct Programming Interface (DPI) to Verilog
- Accellera SystemVerilog 3.1a (projected for April 2004)
  - Extends VPI library to support all SystemVerilog constructs
- IEEE 1354-2005 (2006 ?) [projected date]
  - Will include SystemVerilog DPI along with many VPI extensions

35

Supplemental: SystemVerilog Features

SUTHERLAND HDL
Training Engineers to be HDL wizards

SystemVerilog

3.1a
- assertions
- test program blocks
- clocking domains
- process control
- mailboxes
- semaphores
- constrained random values
- direct C function calls

3.0
- interfaces
- nested hierarchy
- unrestricted ports
- automatic port connect
- enhanced literals
- time values and units
- specialized procedures
- packages
- 2-state modeling
- packed arrays
- array assignments
- queues
- unique/priority case/if
- compilation unit space

from C / C++
- classes
- inheritance
- strings
- dynamic arrays
- associative arrays
- references
- int
- shortint
- longint
- byte
- shortreal
- void
- alias
- globals
- enum
- typedef
- structures
- unions
- casting
- const
- break
- continue
- return
- do–while
- ++ -- += -= *= /=
- >>= <<= >>>= <<<=
- &= |= ^= %=

Verilog-2001
- ANSI C style ports
- generate
- localparam
- constant functions
- standard file I/O
- $value$plusargs
- `ifndef `elsif `line
- @*
- (* attributes *)
- configurations
- memory part selects
- variable part select
- multi dimensional arrays
- signed types
- automatic
- ** (power operator)

Verilog-1995
- modules
- parameters
- function/tasks
- always @
- assign
- $finish $fopen $fclose
- $display $write
- $monitor
- `define `ifdef `else
- `include `timescale
- initial
- disable
- events
- wait # @
- fork–join
- wire reg
- integer real
- time
- packed arrays
- 2D memory
- begin–end
- while
- for forever
- if–else
- repeat
- + = * /
- %
- >> <<

36

Supplemental: Import Function Arguments

SUTHERLAND HDL
Training Engineers to be HDL wizards

- C functions can be imported as a Verilog task or function

```
import "DPI" function real sin(real in); //sin function in C math lib

import "DPI" task file_write(input string data, output reg status);
```

- The C function arguments can be imported as input, output or inout (bidirectional)
  - Inputs behave as if copied into the C function when it is called
    - The C function should not modify input arguments
  - Outputs behave as if copied into Verilog when the function returns
  - Inouts behave as if copied in at call, and copied out at return
  - Arguments are assumed to be inputs unless declared otherwise

The "copying" values in and out of C functions describes the behavior. Software tools can implement this behavior in many ways (using pointers, etc.)