HDL Interoperability & IP-based System Verification

Dennis Brophy

Director of Strategic Business Development

Mixed-HDL Issues

- Why mix HDLs?
  - Use of blocks of IP in “the other” language.
  - Diversity of design teams
  - Respective language strengths of VHDL and Verilog
  - Legacy code (i.e. testbench environment)

“Mixed HDL already in use by 16% of design teams. Expected to grow by 30% in the next year.”

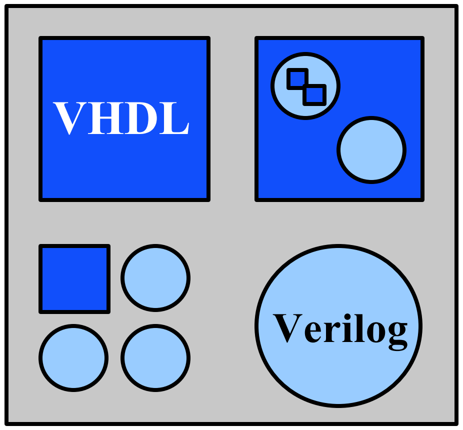
(Collett,"1999 IC/ASIC Physical Design & Layout Verification Study" April 1999)

Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

ModelSim: the Mixed-HDL Solution

- Ability to mix languages at any level
- Single kernel simulation (no backplanes)
- Full debug into both languages (no black boxes)
- Easy instantiation of one language into the other (no wrappers)



Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

Mixed-HDL Case 1: IP Integration

- Design in VHDL, IP in Verilog (or vice versa)
  - IP vendor might not use the same HDL as your team

```
architecture only of top is
……….. .
begin
block1_INST: memory
   GENERIC MAP(
      addr_size => counter_size,
      word_size => buffer_size
   )
   PORT MAP (
      clk => clk,
      addr => addr,
      data => data,
      rw => rw,
      strb => strb,
      rdy => rdv   );
```

VHDL instantiating Verilog

```
module memory(clk, addr, data, rw,
strb, rdy);
    input  clk, addr, rw, strb;
    output rdy;
    inout  data;

    parameter addr_size = 8
    parameter word_size = 16

    reg [`word_size-1:0] data_r;
```

Mixed-HDL Case 2: Legacy Testbench

- Design team wants to use testbench/regression content from previous design
- Also wants to use a different HDL for next generation components

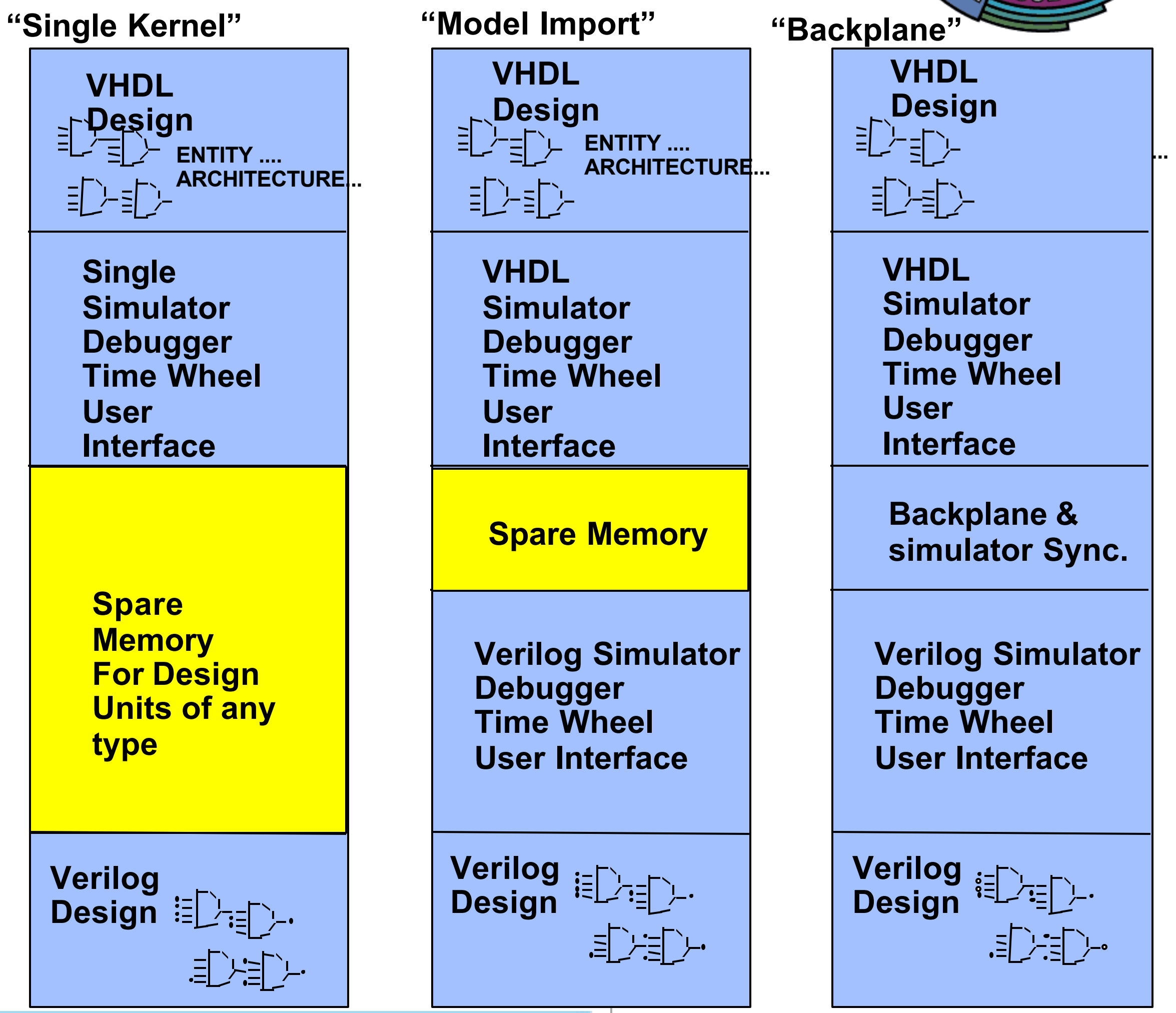
```
module cache(clk, paddr, pdata, prw, pstrb, prdy);
  input    clk, paddr, prw, pstrb;
  output  prdy;

  reg  [3:0] oen, wen;
  wire hit0, hit1, hit2;

  /**************** Cache sets ****************/
  cache_set #(8,5) s0(paddr, pdata, hit0, oen0,
wen0);
  cache_set #(8,5) s1(paddr, pdata, hit1, oen1,
wen1);
```

Verilog instantiating VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use work.std_logic_util.all;

entity cache_set is
generic(
    addr_size   : integer := 8;
    set_size    : integer := 5
);
port (addr: in
std_logic_vector(7 downto 0);
        data: inout
std_logic_vector(15 downto 0);
        hit: out   std_logic;
        oen: in    std_logic;
        wen: in    std_logic
);
end cache_set;
```

Mixed-HDL Verification Alternatives

- ♦ **Single Kernel**
  - Highest capacity, performance, full debug and analysis
- ♦ **Model import**
  - One user-interface
  - Two kernels, no debugging within foreign model
- ♦ **Backplane**
  - Only solution for Analog-Digital
  - Comm. overhead, 2 User I/F, Time synch, limited design capacity



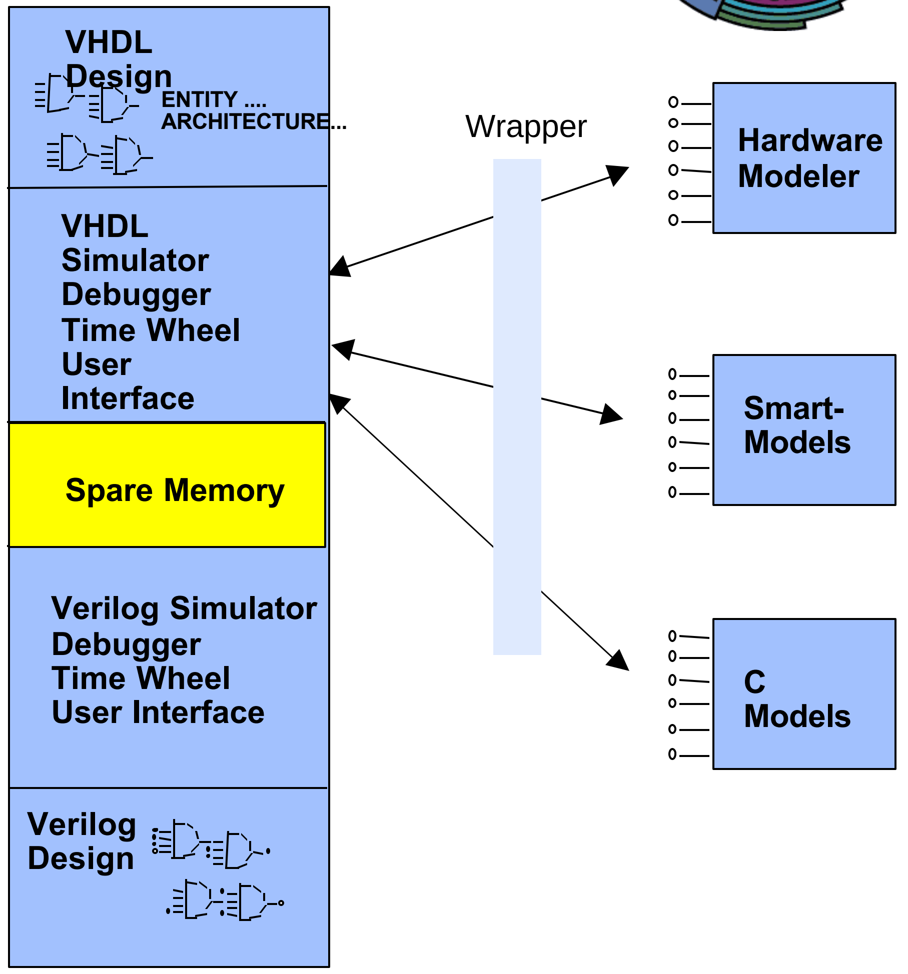
"Single Kernel"
VHDL Design
ENTITY ....
ARCHITECTURE...
Single Simulator Debugger Time Wheel User Interface
Spare Memory For Design Units of any type
Verilog Design
"Model Import"
VHDL Design
ENTITY ....
ARCHITECTURE...
VHDL Simulator Debugger Time Wheel User Interface
Spare Memory
Verilog Simulator Debugger Time Wheel User Interface
Verilog Design
"Backplane"
VHDL Design
VHDL Simulator Debugger Time Wheel User Interface
Backplane & simulator Sync.
Verilog Simulator Debugger Time Wheel User Interface
Verilog Design

Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

Model Import Capability

- Extends behavior through master-slave relationship between two simulators
- Master simulator's User-Interface used to debug both languages
  - Each model requires wrapper architecture manually entered
  - Behavior of imported model visible only through external pins

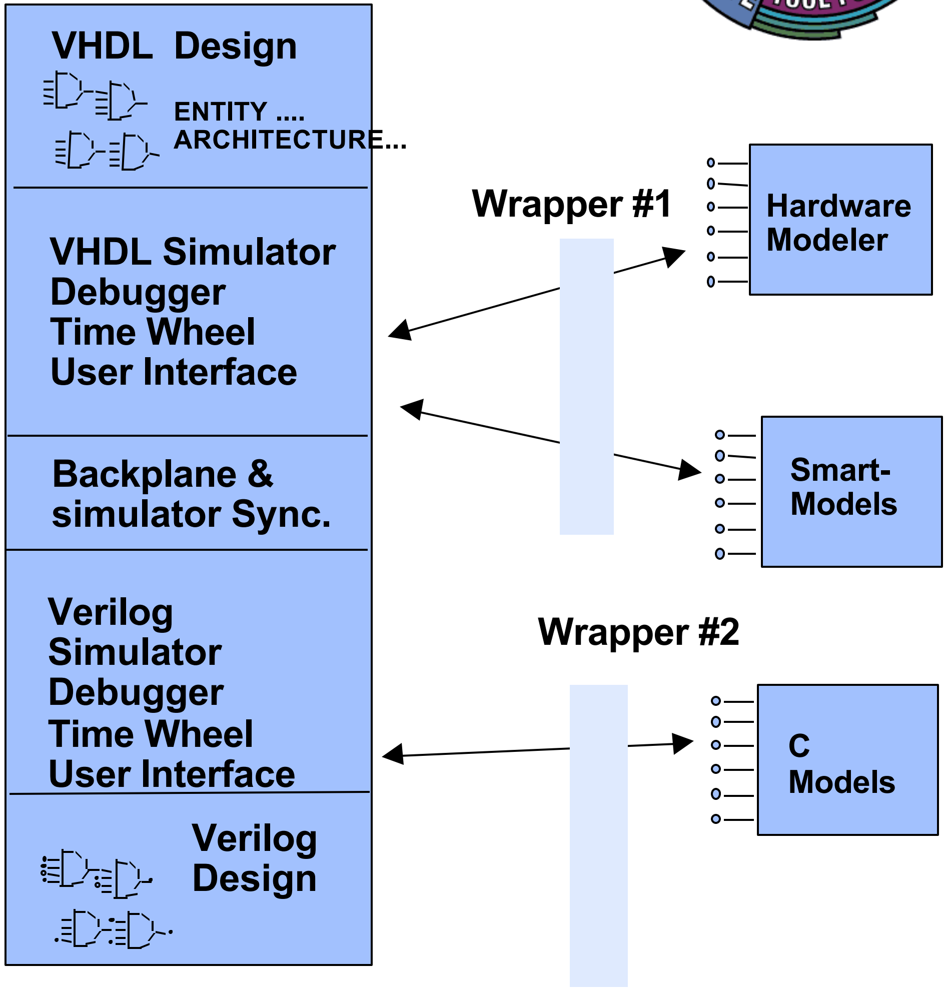


Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

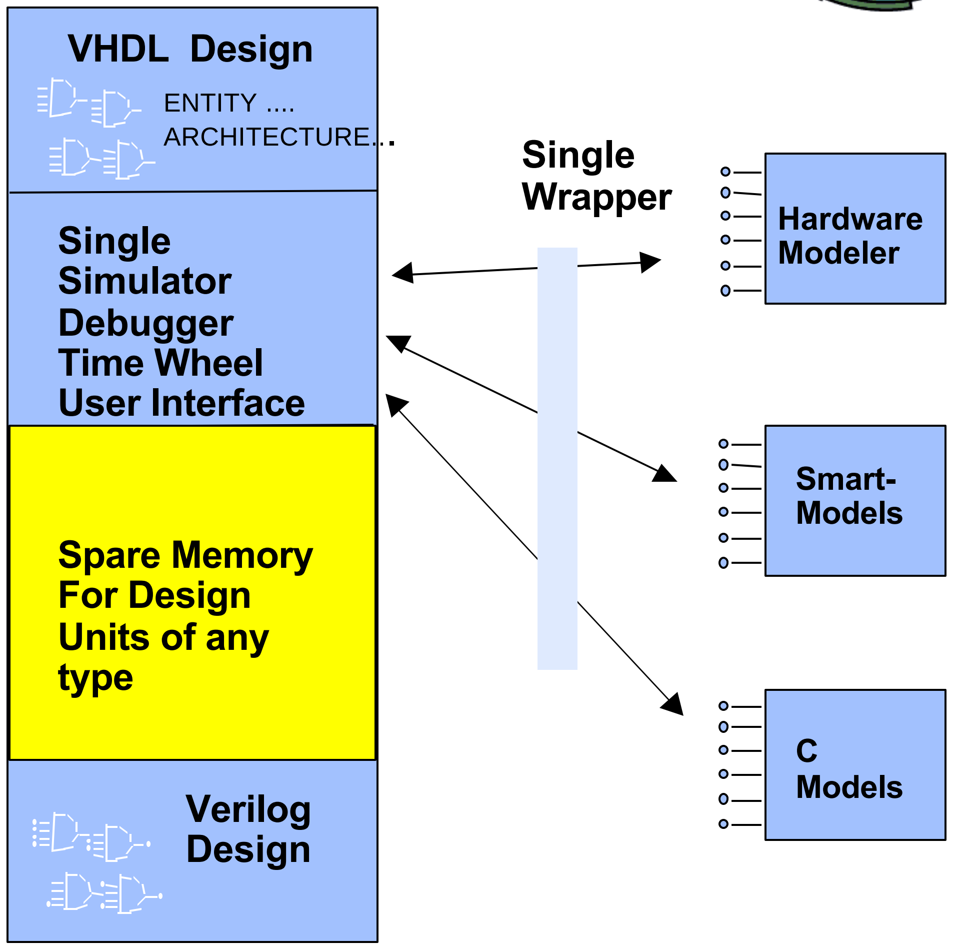
Backplane Technology

- Viable for some applications such as Analog/Mixed Signal
  - Limits simulator design capacity
  - Performance limitations due to communication overhead
  - Two user interfaces for debugging, requires expertise in both simulators
  - Synchronization issues



Single Kernel Mixed HDL Simulator

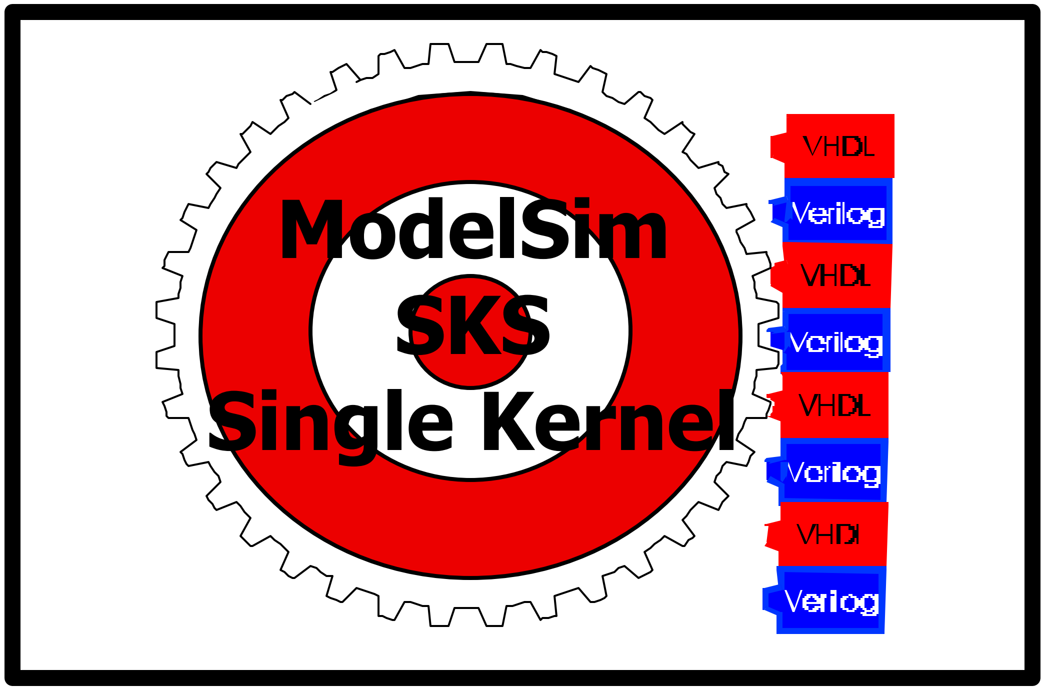
- **Single simulator kernel**
  - **Single debugger and User Interface**
  - **Maximum performance and design capacity**
  - **4x to 10x faster than existing co-simulation alternatives**



Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

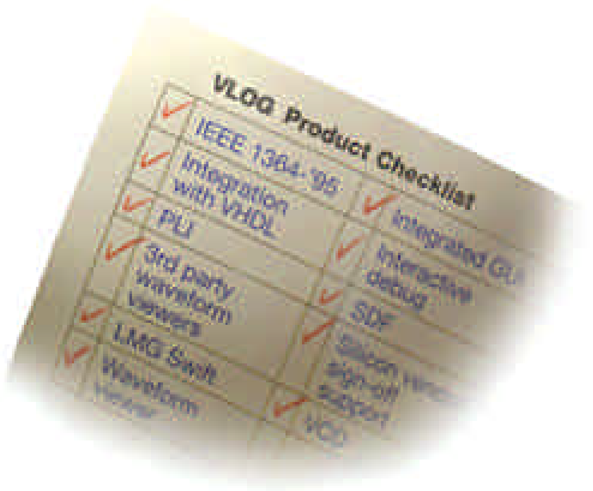
Mixed Language Simulation



- ✓ Single-kernel
- ✓ Freedom of instantiation
- ✓ Full debug at any level
- ✓ No wrappers

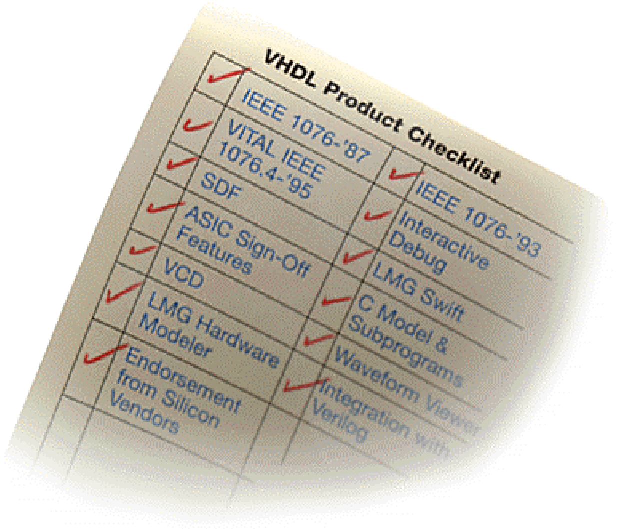
Model*Sim* Core Features

- Single kernel VHDL, Verilog, or mixed-HDL simulation
- Optimized Direct Compiled Code Architecture
  - Platform & Version Independence



VLOG Product Checklist

✓ IEEE 1364-'95	✓ Integrated GU
✓ Integration with VHDL	✓ Interactive debug
✓ PLI	✓ SDF
✓ 3rd party waveform viewers	✓ Silicon ve sign-off support
✓ LMG Swift	✓ VCD
✓ Waveform Viewer	



VHDL Product Checklist

✓ IEEE 1076-'87	✓ IEEE 1076-'93
✓ VITAL IEEE 1076.4-'95	✓ Interactive Debug
✓ SDF	✓ LMG Swift
✓ ASIC Sign-Off Features	✓ C Model & Subprograms
✓ VCD	✓ Waveform Viewer
✓ LMG Hardware Modeler	✓ Integration with Verilog
✓ Endorsement from Silicon Vendors	

- Complete Standards support
- Fast, comprehensive debugging through the GUI
- Protected use & distribution of IP models

Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

Instantiation

VHDL source code, uses Verilog module

```
library IEEE;
use IEEE.std_logic_1164.all;
entity adderN is
    generic(N:integer :=16);
    port(  a    : in   std_logic_vector(N downto 1);
            b    : in std_logic_vector(N downto 1);
          cin    : in std_logic;
          sum    : out std_logic_vector(N downto 1);
          cout   : out std_logic);
    end adderN;

-- struct. implementation of N-bit adder

signal carry : std_logic_vector(0 to N);
begin
    carry(0) <= cin;
    cout        <= carry(N);
-- instantiate single bit adder N times
gen: for I in 1 to N generate
    add: adder port map(
        a => a(I),
        b => b(I),
        cin => carry(I-1),
        sum => sum(I),
        cout => carry(I));
end generate;
```

Verilog adder module

```
module adder (a,b,cin,sum,cout);
     input a;
     input b;
     input cin;
     output sum;
     output cout;

     assign {cout,sum} = a+b+c+cin;
end module
```

Note that the powerful "generate" statement of VHDL is being used to replicate a Verilog single bit adder 16 times.

The designer can use any logic value system, including the full 134 states supported by Verilog if desired. Here, the IEEE std logic system is in use.

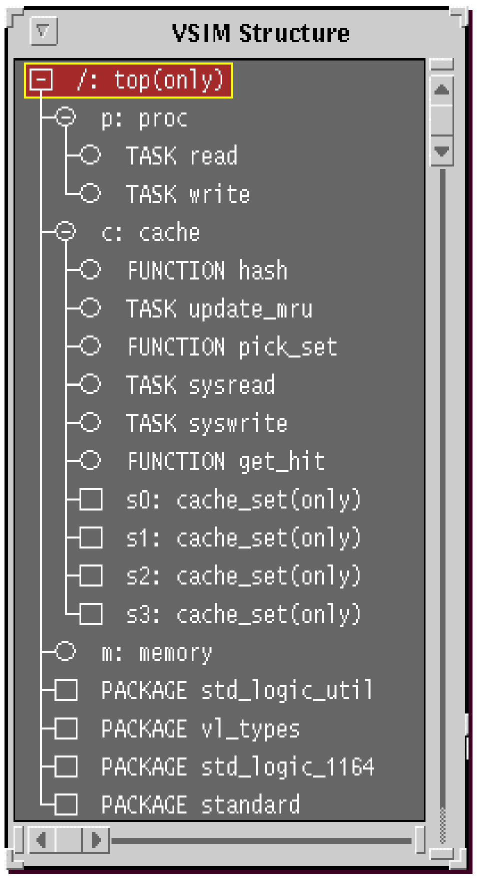
With Model*Sim*, Verilog code can be inside VHDL code and vice-versa, to any degree.

Debugging

- Structure Chart enables hierarchical view of Design
- VHDL structures identified with square
- Verilog modules identified by Circles.
- Dynamic linking of debug windows simplify debugging

□

○

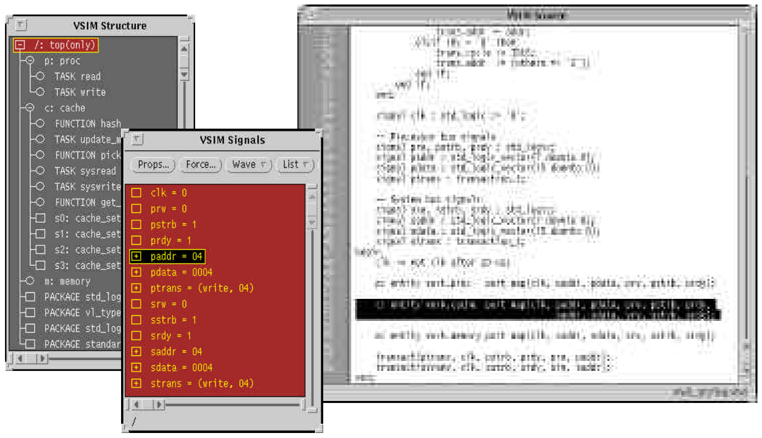


VSIM Structure

```
/: top(only)
 p: proc
   TASK read
   TASK write
 c: cache
   FUNCTION hash
   TASK update_mru
   FUNCTION pick_set
   TASK sysread
   TASK syswrite
   FUNCTION get_hit
   s0: cache_set(only)
   s1: cache_set(only)
   s2: cache_set(only)
   s3: cache_set(only)
 m: memory
 PACKAGE std_logic_util
 PACKAGE vl_types
 PACKAGE std_logic_1164
 PACKAGE standard
```

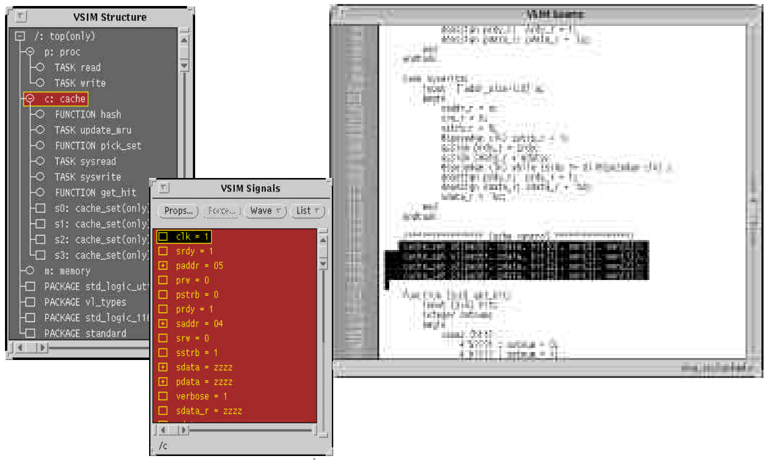
Dynamic Linking

- Selection within the structure, signal, variable, dataflow , or process window automatically updates dynamically linked windows.
- Selected signals easily added to wave, list, and process window



```
VSIM Structure
/: top(only)
  p: proc
    TASK read
    TASK write
  c: cache
    FUNCTION hash
    TASK update_m
    FUNCTION pick
    TASK sysread
    TASK syswrite
    FUNCTION get_
    s0: cache_set
    s1: cache_set
    s2: cache_set
    s3: cache_set
  m: memory
  PACKAGE std_log
  PACKAGE vl_type
  PACKAGE std_log
  PACKAGE standar
```

```
VSIM Signals
clk = 0
prw = 0
pstrb = 1
prdy = 1
paddr = 04
pdata = 0004
ptrans = (write, 04)
srw = 0
sstrb = 1
srdy = 1
saddr = 04
sdata = 0004
strans = (write, 04)
/
```



```
VSIM Structure
/: top(only)
  p: proc
    TASK read
    TASK write
  c: cache
    FUNCTION hash
    TASK update_mru
    FUNCTION pick_set
    TASK sysread
    TASK syswrite
    FUNCTION get_hit
    s0: cache_set(only
    s1: cache_set(only
    s2: cache_set(only
    s3: cache_set(only
  m: memory
  PACKAGE std_logic_ut
  PACKAGE vl_types
  PACKAGE std_logic_11
  PACKAGE standard
```

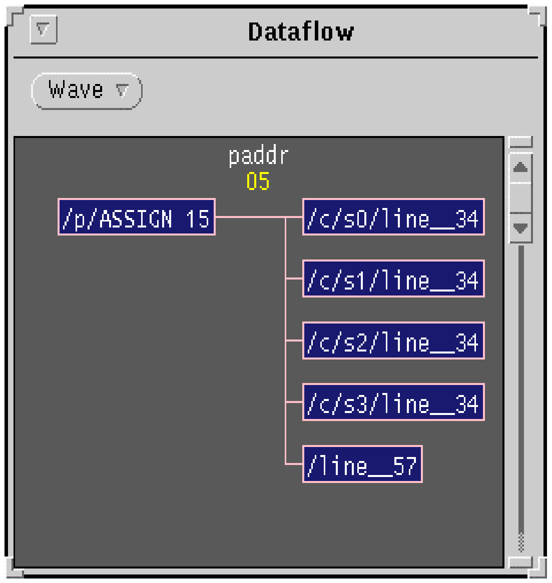
```
VSIM Signals
clk = 1
srdy = 1
paddr = 05
prw = 0
pstrb = 0
prdy = 1
saddr = 04
srw = 0
sstrb = 1
sdata = zzzz
pdata = zzzz
verbose = 1
sdata_r = zzzz
/c
```

Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

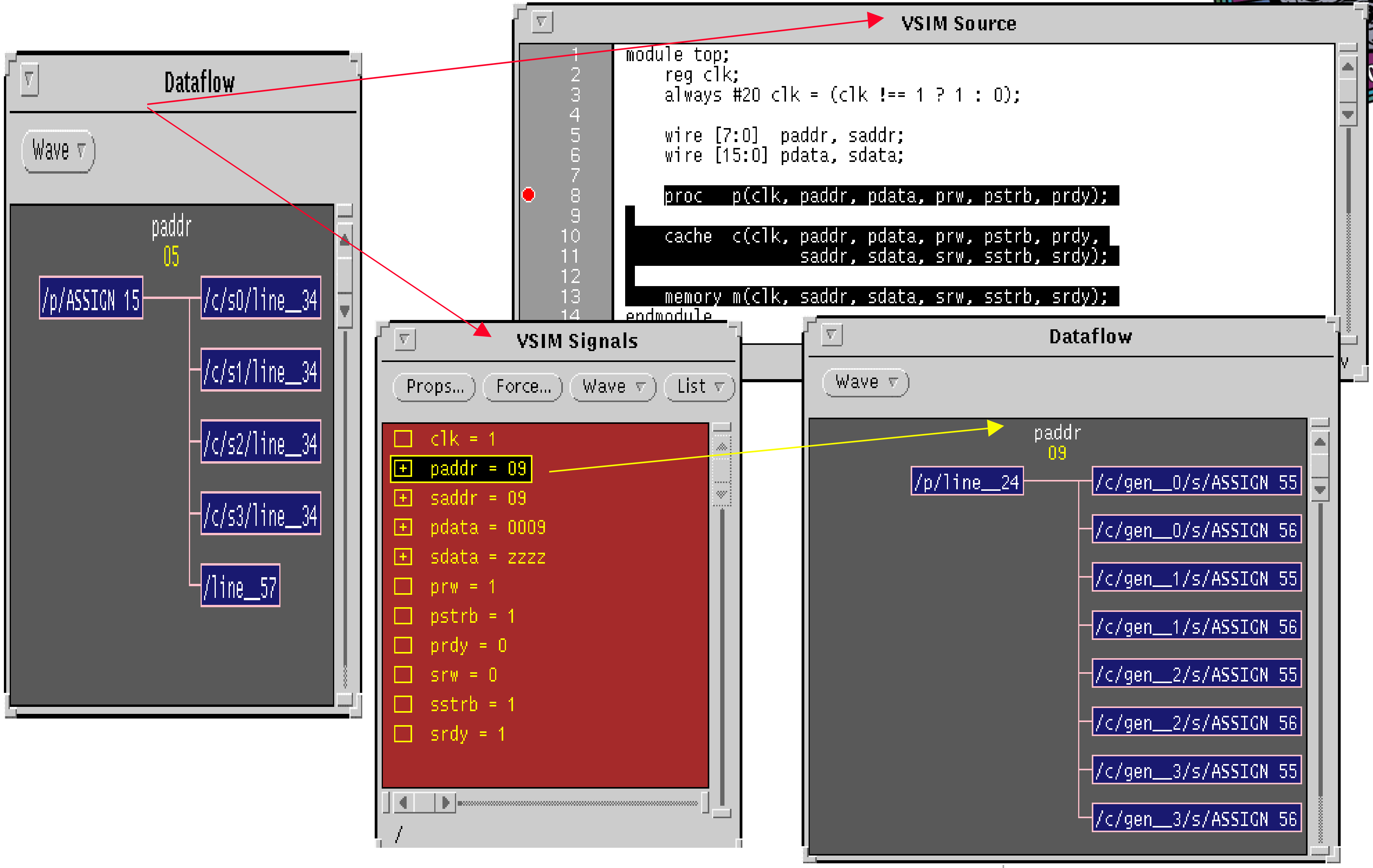
Dataflow Window

- Locate cause of design problem
- Displays signals traversing design hierarchy
- Displays inputs and outputs of a specific process
- Select processes and dynamically update debugging windows
  - Source, structure, process, and variable window



Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

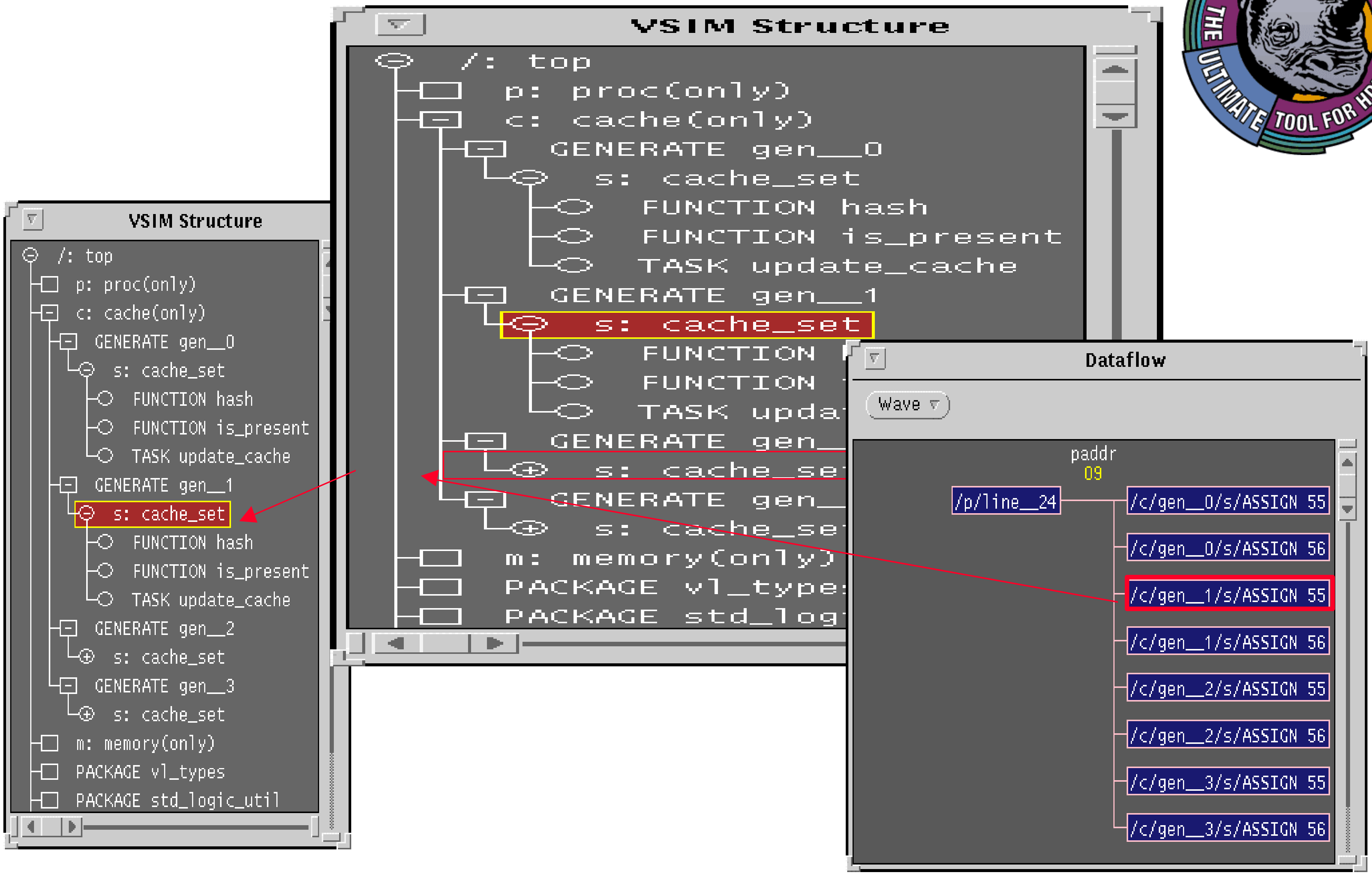


```
module top;
    reg clk;
    always #20 clk = (clk !== 1 ? 1 : 0);

    wire [7:0]  paddr, saddr;
    wire [15:0] pdata, sdata;

    proc   p(clk, paddr, pdata, prw, pstrb, prdy);

    cache  c(clk, paddr, pdata, prw, pstrb, prdy,
                  saddr, sdata, srw, sstrb, srdy);

    memory m(clk, saddr, sdata, srw, sstrb, srdy);
endmodule
```

```
clk = 1
paddr = 09
saddr = 09
pdata = 0009
sdata = zzzz
prw = 1
pstrb = 1
prdy = 0
srw = 0
sstrb = 1
srdy = 1
```

Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com



VSIM Structure

```
/: top
  p: proc(only)
  c: cache(only)
    GENERATE gen__0
      s: cache_set
        FUNCTION hash
        FUNCTION is_present
        TASK update_cache
    GENERATE gen__1
      s: cache_set
        FUNCTION hash
        FUNCTION is_present
        TASK update_cache
    GENERATE gen__2
      s: cache_set
    GENERATE gen__3
      s: cache_set
  m: memory(only)
  PACKAGE vl_types
  PACKAGE std_logic_util
```

VSIM Structure

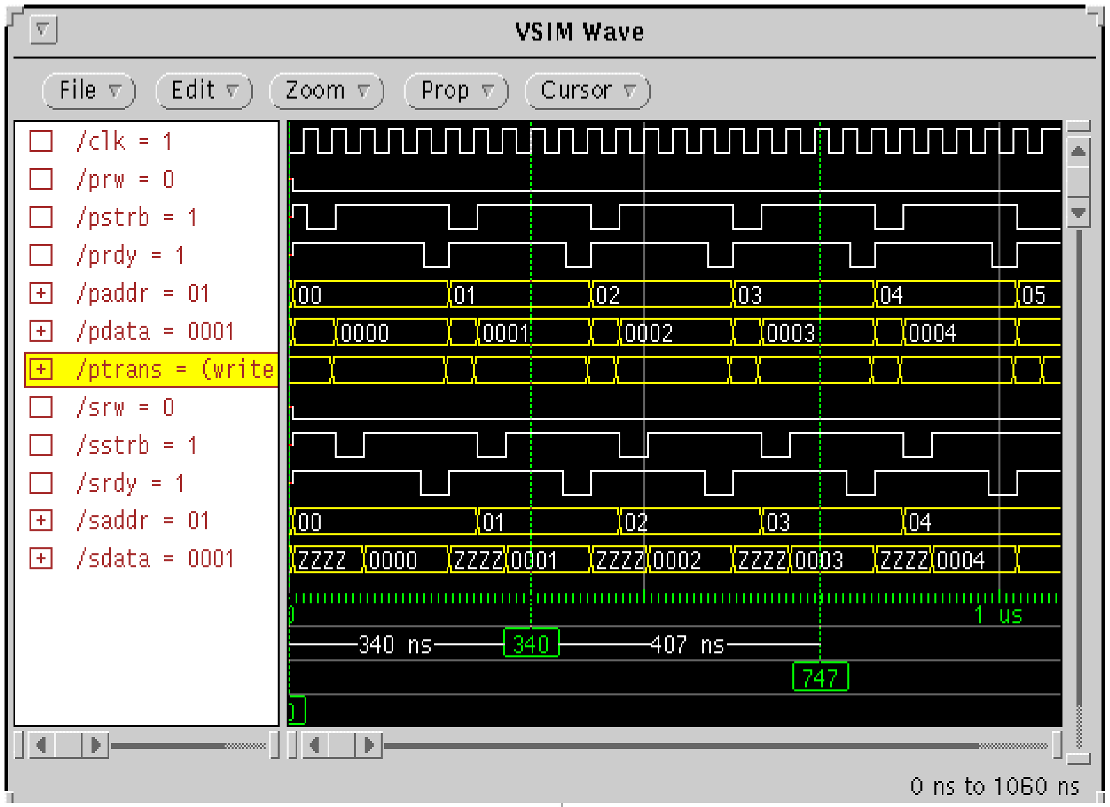
```
/: top
  p: proc(only)
  c: cache(only)
    GENERATE gen__0
      s: cache_set
        FUNCTION hash
        FUNCTION is_present
        TASK update_cache
    GENERATE gen__1
      s: cache_set
        FUNCTION h
        FUNCTION i
        TASK upda
    GENERATE gen_
      s: cache_se
    GENERATE gen_
      s: cache_se
  m: memory(only)
  PACKAGE vl_type
  PACKAGE std_log
```

Dataflow

```
paddr
09
/p/line__24    /c/gen__0/s/ASSIGN 55
               /c/gen__0/s/ASSIGN 56
               /c/gen__1/s/ASSIGN 55
               /c/gen__1/s/ASSIGN 56
               /c/gen__2/s/ASSIGN 55
               /c/gen__2/s/ASSIGN 56
               /c/gen__3/s/ASSIGN 55
               /c/gen__3/s/ASSIGN 56
```

Waveform Display

- Displays enumerated data types or Verilog signal values
- Expandable view
- Interactive measurements signal management
- Displays digital and analog waveforms

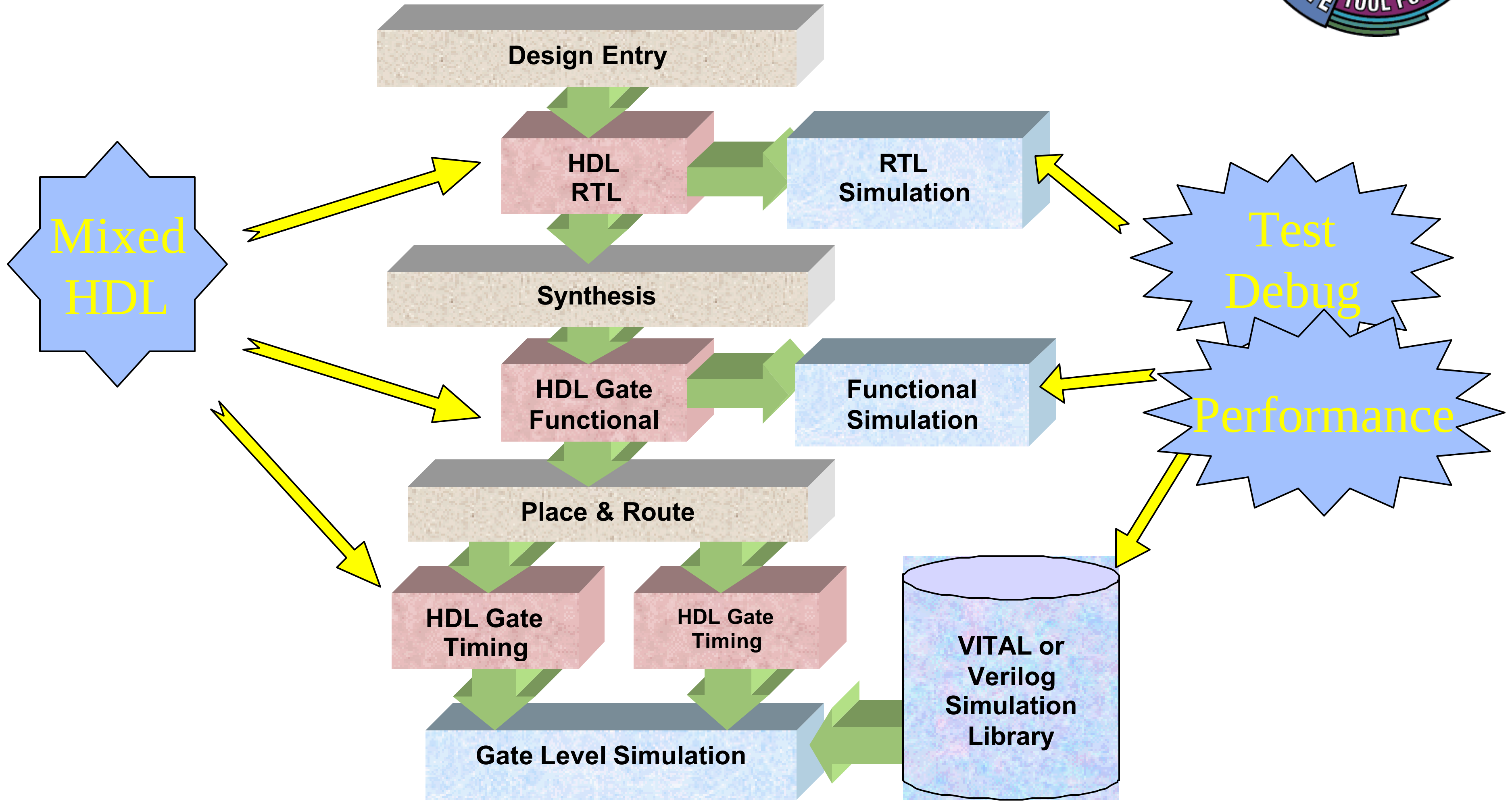


```
/clk = 1
/prw = 0
/pstrb = 1
/prdy = 1
/paddr = 01
/pdata = 0001
/ptrans = (write
/srw = 0
/sstrb = 1
/srdy = 1
/saddr = 01
/sdata = 0001
```

Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

HDL Design Simulation Flow



Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

Forging Industry-wide HDL Interoperability

- Define interoperability
  - Design Unit
  - PLI
  - VCD
  - SDF
- Each language can instantiate design units from the other language
- Define event management
  - One event manager
  - One simulation engine
- Benefit
  - Either VHDL or Verilog can be design master
  - Can freely switch from VHDL to Verilog through out design hierarchy

Model Technology
A MENTOR GRAPHICS COMPANY

www.model.com

Interoperability Details

- VHDL Design Unit objects
  - entity/architecture pairs
  - inputs
  - outputs
  - generics
- Verilog Design Unit objects
  - modules
  - inputs
  - outputs
  - parameters

Interoperability Details

- Define cross-HDL instantiations, adaptations and data type conversion
- Define subset of connected Verilog nets / VHDL signals
  - VHDL signals connected to Verilog
    - bit
    - bit_vector,
    - std_logic
    - std_logic_vector
  - State/Strength Mapping defined with user override capability
  - VHDL Generic values are passed to Verilog parameters

Next Steps

- ♦ United OVI*/VI** TSC Reviewing proposal
  - Anticipate Technical Sub-Committee (TSC) authorization Q4-99
  - Use current de facto technology to seed effort
  - Work with IEEE working groups when complete
- ♦ IEC TC 93 WG 2 championing interoperability
- ♦ How can you participate?
  - Contact Dennis Brophy:
    - Email: dennisb@model.com
    - Tel: +1 (503) 526-1694
    - Fax: +1 (503) 526-5465

* Open Verilog International
** VHDL International

Conclusion

- Design drives verification
  - Multiple languages for teams to use
  - Verification must embrace them all
- IP-based design drives multiple language use
- Testbench re-use drives multiple language use
- Industry standards groups should address design language interoperability issues