Verdi³ NPI Training

DM Model

Based on Verdi³ 2013.01

Glossary

- VIA = Verdi Interoperability Applications
- NPI = Novas Programming Interface
- Novas KDB = Novas Knowledge Database
- FSDB = Fast Signal Database
- VPI = Verilog Procedural Interface

Overview

- DM Model Introduction
- NPI Commands
- NPI Object Diagrams
- Case Study

Overview

- DM Model Introduction
- NPI Commands
- NPI Object Diagrams
- Case Study

DM Model Introduction

What is DM Model?

- Lets users traverse and manipulate design source code from a module-base perspective
  - The traversal scope is in the module definition

- Traversing and manipulating module definitions works similar to editing HDL code by a hardware designer
  - Adding or deleting a port in a module definition impacts all the module instantiations in the design

Overview

- DM Model Introduction
- NPI Commands
- NPI Object Diagrams
- Case Study

NPI Commands

Get Handles by Name

- **npi_dm_handle_by_name –name** *ObjName* **–scope** *ObjScope*
  - Get a handle to an object with a specific name
    - If the *ObjScope* is null, then the name will be searched from the top level
    - Range information of the name will be ignored. For example, “top.a[0]” will be treated as “top.a”
- Example:

```
% set TOP [npi_dm_handle_by_name -name "TOP" -scope ""]
```

Get module named TOP

```
% set ndm_net [npi_dm_handle_by_name -name "TOP.ndm_net" -scope ""]
```

Get net named ndm_net in TOP module

NPI Commands

Get Handles by Specific Types

- **npi_dm_module_by_name –name** *ModuleName*
  - Get the module handle by a module name

- **npi_dm_port_by_name –name** *PortName* **–scope** *Scope*
  - Get the port handle by name within a module or interface scope

- Example:

```
% set MOD [npi_dm_module_by_name -name "MOD"]
```

Get module named MOD

```
% set port_b [npi_dm_port_by_name -name "b" -scope $MOD]
```

Get port named b in MOD

NPI Commands

Get Handles by Specific Types

- **npi_dm_handle –type** *Type* **–ref** *ReferenceHandle*
  - Obtain a handle to an object with a one-to-one relationship

- Example:

```
% set inst_i0 [npi_dm_handle_by_name -name "TOP.i0" -scope ""]
```

Get the handle for instance *TOP.i0*

```
% set def_i0  [npi_dm_handle -type npiDmDefine -ref $inst_i0]
```

Get the definition of the instance

NPI Commands

Create the Iterator and Iterate Items

- **npi_dm_iter_start –type** *Type* **–ref** *ReferenceHandle*
  - Obtain an iterator handle to objects with a one-to-many relationship
- **npi_dm_iter_next –iter** *Iterator*
  - Obtain the next handle of the iterator
- **npi_dm_iter_stop –iter** *Iterator*
  - Release the iterator

- Example:

```
set MOD [npi_dm_module_by_name -name MOD]
set port_iter [npi_dm_iter_start -type npiDmPort -ref $MOD]
if { $port_iter != "" } {
  while { [ set port [npi_dm_iter_next -iter $port_iter ] ] != "" } {
    # Actions on port
  }
  npi_dm_iter_stop -iter $port_iter
}
```

Create the iterator for all ports to an instance

Scan objects for the iterator

Free the iterator

NPI Commands

Get Properties from Handles

- **npi_dm_property –type** *Type* **–ref** *ReferenceHandle*
  - Get the value of an integer or Boolean property of an object
- **npi_dm_property_str –type** *Type* **–ref** *ReferenceHandle*
  - Get the value of a string property of an object

- Example:

```
% set CPU [npi_dm_module_by_name -name CPU]

% set cpu_name [npi_dm_property_str -type npiDmName -ref $CPU]
```

Get the string property (name) of the reference handle

```
% set top_flag [npi_dm_property -type npiDmTopModule -ref $CPU]
```

Get the Boolean property (top module or not) of the reference handle

NPI Commands

Create Data Types (1/2)

- **npi_dm_create_range –left** *LeftRange* **–right** *RightRange*
  - Create a range object
- **npi_dm_create_npiDmHandleArray –array_list** *HandleArray*
  - Create an array structure in Tcl
- **npi_dm_create_npiDmBasicDataType –type** *Type* **–sign** *Sign* **–packed_dim** *DimArrayObj*
  - Create a basic data type structure in Tcl
  - Refer to the *NPI Reference Manual* for all available data types

NPI Commands

Create Data Types (2/2)

- Example:

```
% set npiRange1 [npi_dm_create_range –left 7 –right 0]
% set npiRange2 [npi_dm_create_range –left 4 –right 0]
```

Create two range variables: *$npiRange1* for **[7:0]** ; *$npiRange2* for **[4:0]**

```
% set npiArray [npi_dm_create_npiDmHandleArray –array_list
“$npiRange1 $npiRange2”]
```

Create an array variables: *$npiArray* for **[7:0] [4:0]**

```
% set npiDataType [npi_dm_create_npiDmBasicDataType
–type npiDmDtLogic –sign npiDmSignNone –packed_dim $npiArray]
```

Create a non-signing logic type variable: *$npiDataType* for **logic [7:0] [4:0]**

NPI Commands

Delete Data Types

- **npi_dm_delete_npiDmHandleArray –array** *HandleArray*
  - Delete an array structure in Tcl

- **npi_dm_delete_npiDmBasicDataType –data_type** *DataType*
  - Delete a basic data type structure in Tcl

- Example:

```
% set npiPRange1 [npi_dm_create_range –left 7 –right 0]
% set npiPRange2 [npi_dm_create_range –left 4 –right 0]
% set npiPArray [npi_dm_create_npiDmHandleArray –array_list
“$npiPRange1 $npiPRange2”]
% set npiDataType [npi_dm_create_npiDmBasicDataType –type
npiDmDtLogic –sign npiDmSignNone –packed_dim $npiPArray]
% npi_dm_delete_npiDmHandleArray –array $npiPArray
% npi_dm_delete_npiDmBasicDataType –data_type $npiDataType
```

Delete the array handle and data type handle

NPI Commands

Create Modules

- **npi_dm_create_module –name** ModuleName **–file** FileName
  - Create a new module object

- Example:

```
% npi_dm_create_module -name ndm_module -file ndm_module.sv
% npi_dm_write_text_mode -dir DM_LIB
```

The result will be saved in: **DM_LIB/ndm_module.sv**

The content is:

```
module ndm_module;
endmodule
```

NPI Commands

Add Nets to a Scope

- **npi_dm_add_net –scope** *Scope* **–name** *NetName* **–data_type** *DataType* **–unpacked_dim** *Dim*
  - Add a new wire net to a scope
- Example:

```
set mod [npi_dm_module_by_name -name MOD]
set range_7_to_0   [npi_dm_create_range -left 7 -right 0]
set range_255_to_0 [npi_dm_create_range -left 255 -right 0]
set packed_dim [npi_dm_create_npiDmHandleArray -array_list $range_7_to_0]
set data_type [npi_dm_create_npiDmBasicDataType -type npiDmDtDefault -sign npiDmSignNone -packed_dim $packed_dim]
set unpacked_dim [npi_dm_create_npiDmHandleArray -array_list $range_255_to_0]
npi_dm_add_net -scope $mod -name ndm_net -data_type $data_type -unpacked_dim $unpacked_dim
```

```
module MOD;
  wire [7:0] ndm_net [255:0];
  wire a;
endmodule
```

NPI Commands

Add Vars to a Scope

- **npi_dm_add_var –scope** *Scope* **–name** *VarName* **–data_type** *DataType* **–unpacked_dim** *Dim*
  - Add a new variable to a scope
- Example:

```
set mod [npi_dm_module_by_name -name MOD]
set range_7_to_0   [npi_dm_create_range -left 7 -right 0]
set range_255_to_0 [npi_dm_create_range -left 255 -right 0]
set packed_dim [npi_dm_create_npiDmHandleArray -array_list $range_7_to_0]
set data_type [npi_dm_create_npiDmBasicDataType -type npiDmDtDefault -sign npiDmSignNone -packed_dim $packed_dim]
set unpacked_dim [npi_dm_create_npiDmHandleArray -array_list $range_255_to_0]
npi_dm_add_var -scope $mod -name ndm_var -data_type $data_type -unpacked_dim $unpacked_dim
```

```
module MOD;
  bit [7:0] ndm_var [255:0];
  logic a;
endmodule
```

NPI Commands

Add Ports to a Scope

- **npi_dm_add_port –scope** *Scope* **–name** *PortName* **–dir** *Dir*
  - Add a new port that is connected with an existing signal to a scope

- Example:

```
% set mod [npi_dm_module_by_name -name MOD]
% npi_dm_add_port -scope $mod -name a -dir npiDmDirInput
% npi_dm_add_port -scope $mod -name b -dir npiDmDirOutput
```

```
module MOD(a,b);
  input wire a;
  output logic b;
endmodule
```

NPI Commands

Add Instances to a Scope

- **npi_dm_add_instance –inst_def** *InstDef* **–name** *InstName* **–dest_scope** *DestScope*
  - Add a new instance to a scope

- Example:

```
set npiTop [npi_dm_module_by_name -name TOP]
set npiMod [npi_dm_module_by_name -name MOD]
npi_dm_add_instance -inst_def $npiMod -name inst2 -dest_scope $npiTop
```

```
module MOD(a, b);
  input a;
  output b;
endmodule

module TOP;
  wire w1, w2;
  MOD inst1(w1, w2);
  MOD inst2();
endmodule
```

NPI Commands

Delete Objects in a Scope

- **npi_dm_delete_signal –signal** *SignalObject*
  - Delete a variable or a net
- **npi_dm_delete_port –port** *PortObject*
  - Delete a port
- **npi_dm_delete_instance –inst** *InstObject*
  - Delete an instance

- Example:

```
% set a [npi_dm_handle_by_name -name "MOD.a" -scope ""]
% set port [npi_dm_port_by_name -name "a" -scope $MOD]
% set ndm_instance [npi_dm_handle_by_name -name "TOP.ndm_instance" -scope ""]
% npi_dm_delete_signal -signal $a
% npi_dm_delete_port -port $port
% npi_dm_delete_instance -inst $ndm_instance
```

Delete object handles

NPI Commands

Create Connections - Ports

- **npi_dm_make_port_connection –inst** *InstObject* **–port** *PortObject* **–expr** *ExpressionObject*
  - Connect an instance port with an expression

- Example:

```
set inst1 [npi_dm_handle_by_name -name "TOP.inst1" -scope ""]
set MOD   [npi_dm_handle -type npiDmDefine -ref $inst1]
lappend conn [npi_dm_handle_by_name -name  "TOP.w1" -scope ""]
lappend conn [npi_dm_handle_by_name -name  "TOP.w2" -scope ""]
set i 0
set port_iter [npi_dm_iter_start -type npiDmPort -ref $MOD]
if { $port_iter != "" } {
  while { [ set port [npi_dm_iter_next -iter $port_iter ] ] != "" } {
    npi_dm_make_port_connection -inst $inst1 -port \
        $port -expr [lindex $conn $i];
    set i [expr $i+1]
  }
  npi_dm_iter_stop -iter $port_iter
}
```

```
module MOD(a, b);
  input a;
  output b;
endmodule

module TOP;
  wire w1, w2;
  MOD inst1( .a( w1 ),
  .b( w2 ) );
endmodule
```

NPI Commands

Create Connections - Assignments

- **npi_dm_make_assign_connection –scope** *ScopeObj* **–lhs** *LHSObject* **–rhs** *RHSObject*
  - Add a continuous assignment connection to a scope

- Example:

```
set MOD [npi_dm_module_by_name -name "MOD"]
set a   [npi_dm_handle_by_name -name "a" -scope $MOD]
set b   [npi_dm_handle_by_name -name "b" -scope $MOD]

set assign [npi_dm_make_assign_connection -scope $MOD
-lhs $a -rhs $b]
```

```
module MOD;
  logic a, b;
  assign a = b;
endmodule
```

NPI Commands

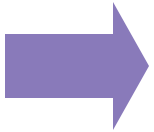
Delete Connections - Ports

- **npi_dm_delete_port_connection –inst** *InstObject* **–port** *PortObject*
  - Disconnect a port connection within an instance

- Example:

```
set MOD  [npi_dm_module_by_name -name MOD]
set inst [npi_dm_handle_by_name -name "TOP.ndm_instance" -scope ""]
set port [npi_dm_port_by_name -name "a" -scope $MOD]
npi_dm_delete_port_connection -inst $inst -port $port
```

```
module TOP;
  wire a, b;
  MOD ndm_instance( a, b );
endmodule
module MOD( input a, b );
endmodule
```



```
module TOP;
  wire a, b;
  MOD ndm_instance( .a(),
 .b( b ) );
endmodule
module MOD( input a, b );
endmodule
```

NPI Commands

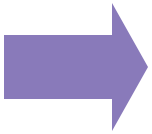
Delete Connections - Assignments

- **npi_dm_delete_assign_connection –cont_assign** *AssignObject*
  - Delete a continuous assignment

- Example:

```
set MOD [npi_dm_module_by_name -name MOD]
set iter [npi_dm_iter_start -type npiDmContAssign -ref $MOD]
if { $iter != "" } {
  while { [ set cont_assign [npi_dm_iter_next -iter $iter ] ] != "" } {
    npi_dm_delete_assign_connection -cont_assign $cont_assign
  }
  npi_dm_iter_stop -iter $iter
}
```

```
module MOD;
  wire a, b;
  assign a = b;
endmodule
```



```
module MOD;
  wire a, b;

endmodule
```

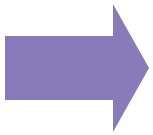
NPI Commands

Move Instances

- **npi_dm_move_instance –inst** *InstObject* **–dest_scope** *DestScopeObject*
  - Move an existing instance to another scope
- Example:

```
% set TOP [npi_dm_module_by_name -name "TOP"]
% set i1  [npi_dm_handle_by_name -name "TOP.i0.i1" -scope ""]
% npi_dm_move_instance -inst $i1 -dest_scope $TOP
```

```
module TOP;
  wire w1, w2;
  MOD1 i0(w1, w2);
endmodule
module MOD1(a, b);
  input a;
  output b;
  MOD2 i1(a, b);
endmodule
module MOD2(a, b);
  input a;
  output b;
endmodule
```



```
module TOP;
  wire w1, w2;
  MOD1 i0(w1, w2);
  MOD2 i1( .a(),
  .b() );
endmodule
module MOD1(a, b);
  input a;
  output b;
endmodule
module MOD2(a, b);
  input a;
  output b;
endmodule
```

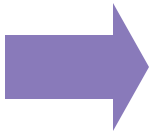
NPI Commands

Replace Modules

- **npi_dm_replace_module –inst** *InstObject* **–module** *ModuleObject*
  - Replace the module of an instance. The module candidate should have the same parameter list and port list as the original module
- Example:

```
% set MOD2 [npi_dm_module_by_name -name "MOD2"]
% set i0  [npi_dm_handle_by_name -name "TOP.i0" -scope ""]
% npi_dm_replace_module -inst $i0 -module $MOD2
```

```
module TOP;
  wire w1, w2;
  MOD1 i0(w1, w2);
endmodule
module MOD1(a, b);
  input a;
  output b;
endmodule
module MOD2(a, b);
  input a;
  output b;
endmodule
```



```
module TOP;
  wire w1, w2;
  MOD2 i0( .a( w1 ),
  .b( w2 ) );
endmodule
module MOD1(a, b);
  input a;
  output b;
endmodule
module MOD2(a, b);
  input a;
  output b;
endmodule
```

NPI Commands

Read SDC File

- **npi_dm_sdc_read –sdc_file** *FileName*
  - Read the SDC file in design manipulation

- **npi_dm_sdc_set_top –top** *TopScopeHier* **–delimiter** *Delimiter*
  - Set the SDC top scope in design manipulation
  - The SDC top should be set before **npi_dm_sdc_read** is called and it should be set only once

- Example:

```
% npi_dm_sdc_set_top -top "TOP.i0" -delimiter "."
% set  result [npi_dm_sdc_read -sdc_file "input.sdc"]
```

NPI Commands

Set Directives

- **npi_dm_set_directive –id** *Name* **–type** *Type*
  - Set the compiler directive status
  - Available *Types* are: **npiDmCdIfdef**, **npiDmCdIfndef**, **npiDmCdElsif**, **npiDmCdElse**, and **npiDmCdEndif**
- Example:

```
set scope [npi_dm_module_by_name -name "TOP"]
set cell [npi_dm_module_by_name -name "MOD"]
npi_dm_set_directive -id "FPGA" -type npiDmCdIfdef
npi_dm_add_instance -inst_def $cell -name "i1"
-dest_scope $scope
npi_dm_set_directive -id "" -type npiDmCdElse
npi_dm_add_instance -inst_def $cell -name "i2"
-dest_scope $scope
npi_dm_set_directive -id "" -type npiDmCdEndif
```

```
module TOP;
  MOD i0();
endmodule
module MOD;
endmodule
```

```
module TOP;
  MOD i0();
`ifdef FPGA
  MOD i1();
`else
  MOD i2();
`endif
endmodule
module MOD;
endmodule
```

NPI Commands

Write Out the Modified Design

- **npi_dm_write_text_mode –dir** *DirName*
  - Generate the modified design to the target directory
  - If the SDC file is loaded, the updated SDC file will also be written out
  - The “include” files or macro definitions will be auto expanded if needed

- Example:

```
% set TOP [npi_dm_handle_by_name -name "TOP" -scope ""]
% npi_dm_add_net -scope $TOP -name "a" -data_type ""
-unpacked_dim ""
% npi_dm_write_text_mode -dir "DM_LIB"
```

```
Original design location:
/home/user/allen/input.v
```

```
The modified design will be saved to:
DM_LIB/home/user/allen/input.v
```

Overview

- DM Model Introduction
- NPI Commands
- NPI Object Diagrams
- Case Study

How to Use Object Diagrams?

Guidelines for Reading Object Diagrams

- Which object diagram should I refer to?
  - Find the object diagram based on your reference object handle
  - For example, if you are looking for all ports for a module, you need to check the "**module**" object diagram

- Which NPI command should I use to get the handle?
  - Refer to the object diagram according to your reference object handle, and check the arrow type
    -  Single arrow (one to one):

      npi_dm_handle
    -  Double arrow (one to many):

      npi_dm_iter_start

How to Use Object Diagrams?

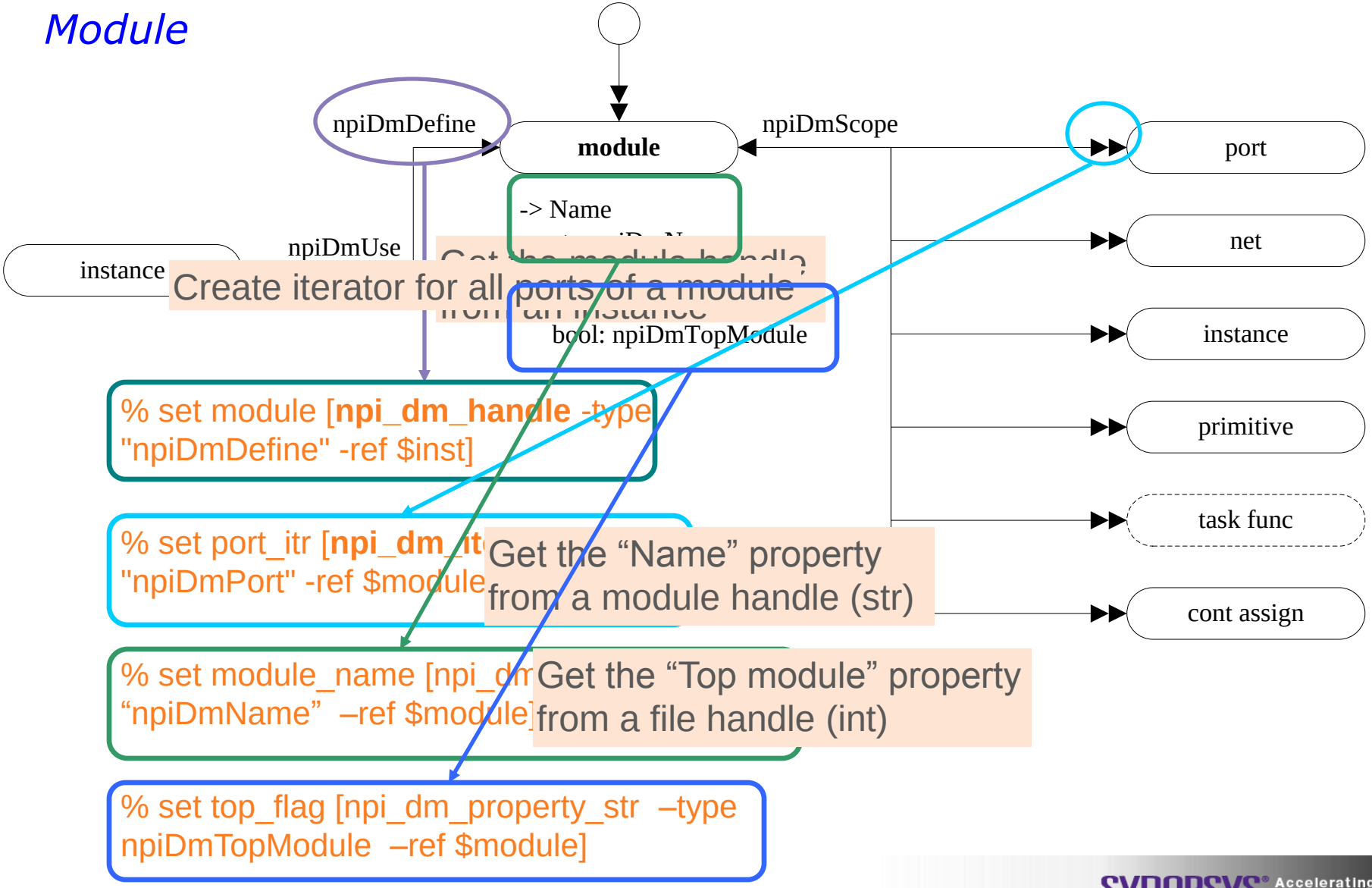
Guidelines for Reading Object Diagrams

- How can I get the type string of **–type** option for **npi_dm_handle** and **npi_dm_iter_start**?
  - Refer to the object diagram according to your reference object handle, and check the name for each object
    - Change the first character to uppercase, and then add "**npiDm**" as the prefix

- How can I get the type string of **–type** option for **npi_dm_property** and **npi_dm_property_str**?
  - Refer to the object diagram according to your reference object handle, and check the description under the object diagram
    - **str**: means the property is string type
      Use **npi_dm_property_str** command
    - **Int**: means the property is integer type
      Use **npi_dm_property** commands
    - **bool**: means the property is Boolean type
      Use **npi_dm_property** command

Object Diagram

Module



Object Diagram

Other Object Diagrams

- Refer to the documentation for other object diagrams
  - The document is located in: <*Verdi_install*>*/doc/VIA_NPI.pdf*
  - Check the **DM Model → Object Diagram** chapter

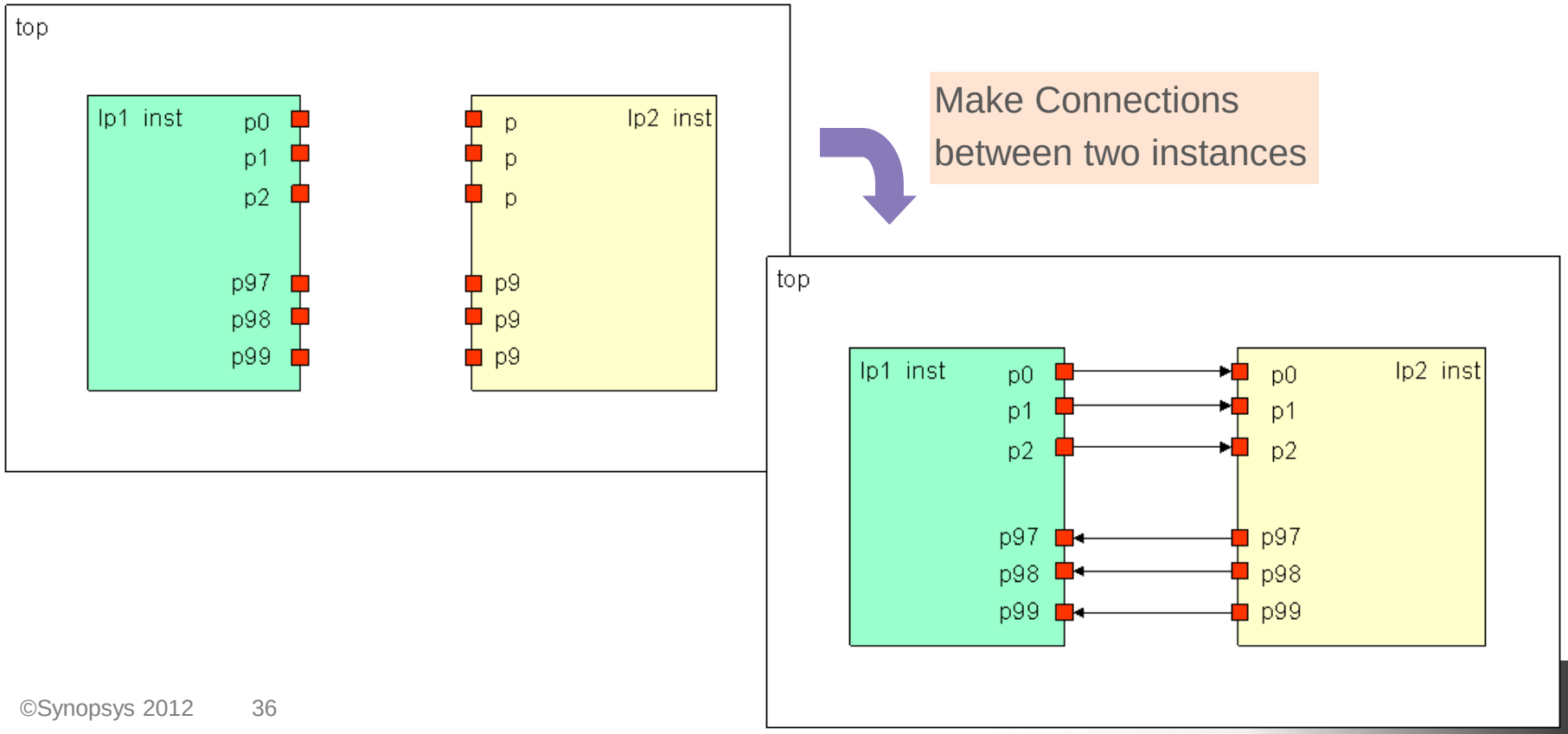
Overview

- DM Model Introduction
- NPI Commands
- NPI Object Diagrams
- Case Study

Case Study

Overview (1/2)

- Requirement:
  - Two instances have been integrated to Top scope, a Tcl script is required to make a connection for these two instances automatically



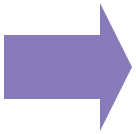
Case Study

Overview (2/2)

```
module top;
  IP1 Ip1_inst();
  IP2 Ip2_inst();
endmodule

module Ip1_def( input p0, p1, p2, p3,
…
, p95, p96, output p97, p98, p99 );
endmodule

module Ip2_def( output p0, p1, p2, p3,
…
, p95, p96, input p97, p98, p99 );
endmodule
```



```
module top;
  wire w0;
  …
  wire w99;
  IP1 Ip1_inst( .p0(w0), .p1(w1), …, .p99(w99) );
  IP2 Ip2_inst( .p0(w0), .p1(w1), …, .p99(w99) );
endmodule

module Ip1_def( input p0, p1, p2, p3,
…
, p95, p96, output p97, p98, p99 );
endmodule

module Ip2_def( output p0, p1, p2, p3,
…
, p95, p96, input p97, p98, p99 );
endmodule
```

Case Study

Script (1/3)

```
debImport -sv top.sv
```

Load the design

```
set top [npi_dm_module_by_name -name "top"]
```

Get the module handle *$top*

```
set net_list ""
for {set i 0} {$i<100} {incr i} {
  set net_name "w$i"
  set net [npi_dm_add_net -scope $top -name $net_name -data_type
""
-unpacked_dim ""]
  lappend net_list $net
}
```

Add net w0 ~w99
to the *$top* module

```
set lp1_inst [npi_dm_handle_by_name -name "lp1_inst" -scope $top]
```

Get the instance handle *$ lp1_inst*

```
set lp1_def [npi_dm_handle -type "npiDmDefine" -ref $lp1_inst]
```

Get the module handle *$lp1_def* for the instance

Case Study

Script (2/3)

```
set Ip1_inst [npi_dm_handle_by_name -name "Ip1_inst" -scope $top]
```

Get the instance handle *$ Ip1_inst*

```
set Ip1_def [npi_dm_handle -type "npiDmDefine" -ref $Ip1_inst]
```

Get the module handle *$Ip1_def* for the instance

```
set Ip1_port_iter [npi_dm_iter_start -type "npiDmPort" -ref $Ip1_def]
```

Get the iterator *$Ip1_port_iter* for all ports of the module

```
set i 0
while { [ set Ip1_port [npi_dm_iter_next -iter $Ip1_port_iter] ] != "" } {
  npi_dm_make_port_connection -inst $Ip1_inst -port $Ip1_port -expr
[lindex $net_list $i]
  incr i
}
```

Connect created nets (w0 ~w99) to all ports in *$Ip1_inst* instance

Case Study

Script (3/3)

```
set lp2_inst [npi_dm_handle_by_name -name "lp2_inst" -scope $top]
set lp2_def [npi_dm_handle -type "npiDmDefine" -ref $lp2_inst]
set lp2_port_iter [npi_dm_iter_start -type "npiDmPort" -ref $lp2_def]
```

Get the instance handle *$lp2_inst*, module handle *$lp2_def*, and the port iterator *$lp2_port_itr*

```
set i 0
while { [ set lp2_port [npi_dm_iter_next -iter $lp2_port_iter] ] != "" } {
  npi_dm_make_port_connection -inst $lp2_inst -port $lp2_port -expr
[lindex $net_list $i]
  incr i
}
```

Connect created nets (w0 ~w99) to all ports in *$lp2_inst* instance

```
npi_dm_write_text_mode -dir DM_LIB
```

Write out the modified design to *DM_LIB* directory

```
debExit
```

Exit *Verdi*