Overview: The purpose of this is to correct inconsistencies between the bind BNF and the text. Specifically:

1. The text after the BNF states that “Possible target scopes include module, program, and interface declarations” but the BNF only shows modules and interfaces, and the text at the top only states modules and instances.
2. Clarify that what is allowed for a bind instantiation is restricted by what is allowed to be instantiated within the bind target scope.
3. Rewrite of the first section for clarity

Since the previous approval of 1722, the following changes have occurred:

1. updated section numbers to correspond to draft 3a
2. Note 1 was modified in two places per Brad’s comment that “ contrary to the footnote, a "bind_target_instance" couldn't be an "interface_identifier". Perhaps, the intent was to say "When the ...bind_target_instance is an interface_instance_identifier ..."?
3. The sentence “Possible target scopes include module, program, and interface declarations.” Was replaced with: “A bind target scope shall be a module or an interface. A bind target instance shall be an instance of a module or an interface”.
4. Incorporated the changes from 1855 that resulted from moving the section from Chapter 16 to Chapter 22:
    a. Changed the title of 22.10 from “Binding properties to scopes or instances” to “Binding auxilliary code to scopes or instances”
    b. clause 16 in section 1.6 (page 26 of draft 3) needs to be updated to exclude assertion binding and clause 22 should include it.
    c. Syntax 22.3 Module item syntax replicates the bind directive from Annex A. The note that was added to Annex A needs to be replicated here too.

Replace BNF in A.1.4:

bind_directive ::=
    bind bind_target_scope [**:** bind_target_instance_list] bind_instantiation **;**
    | **bind** bind_target_instance bind_instantiation **;**

With

1 **(Note to the editor: fill in the appropriate number)**

bind_directive ::=
    bind bind_target_scope [**:** bind_target_instance_list] bind_instantiation **;**
    | **bind** bind_target_instance bind instantiation **;**

1) if the bind_target_scope is an interface_identifier or the bind_target_instance is an interface_instance_idenifier, the bind instantiation must be an interface instantiation.

Replace in 22.10:

22.10 Binding properties to scopes or instances

To facilitate verification separate from design, it is possible to specify properties and bind them to specific modules or instances. The following are some goals of providing this feature:

— It allows verification engineers to verify with minimum changes to the design code and files.
— It allows a convenient mechanism to attach verification Internet Protocol (IP) to a module or an instance.
— No semantic changes to the assertions are introduced due to this feature. It is equivalent to writing properties external to a module, using hierarchical path names.

With this feature, a user can bind a module, interface, or program instance to a module or a module instance.

With

~~**22.10 Binding properties to scopes or instances**~~

~~To facilitate verification separate from design, it is possible to specify properties and bind them to specific modules or instances. The following are some goals of providing this feature:~~

~~— It allows verification engineers to verify with minimum changes to the design code and files.~~
~~— It allows a convenient mechanism to attach verification Internet Protocol (IP) to a module or an instance.~~
~~— No semantic changes to the assertions are introduced due to this feature. It is equivalent to writing properties external to a module, using hierarchical path names.~~

~~With this feature, a user can bind a module, interface, or program instance to a module or a module instance.~~

22.10 Binding auxilliary code to scopes or instances

It is often desired to keep verification code separate from the design code. SystemVerilog provides a ***bind*** construct that is used to specify one or more instantiations of a module, interface, or program block without modifying the code of the target. So for example, an assertion-based checker that is encapsulated in a module, interface, or program can be instantiated in a target module or a module instance in a non-intrusive manner. Similarly, an assertion-checker that is encapsulated in an interface can be bound to a target interface or interface instance.

Replace in Syntax 22-3

bind_directive ::=
    bind bind_target_scope [**:** bind_target_instance_list] bind_instantiation **;**
    | **bind** bind_target_instance bind_instantiation **;**

With

1 **(Note to the editor: fill in the appropriate number)**
bind_directive ::=
    bind bind_target_scope [**:** bind_target_instance_list] bind_instantiation **;**
    | **bind** bind_target_instance bind_instantiation **;**

1) if the bind_target_scope is an interface_identifier or the bind_target_instance is an interface_instance_idenifier, the bind instantiation must be an interface instantiation.

Replace in Syntax 22-7

bind_directive ::=
    bind bind_target_scope [**:** bind_target_instance_list] bind_instantiation **;**
    | **bind** bind_target_instance bind_instantiation **;**

With

1 **(Note to the editor: fill in the appropriate number)**
bind_directive ::=
    bind bind_target_scope [**:** bind_target_instance_list] bind_instantiation **;**
    | **bind** bind_target_instance bind_instantiation **;**

1) if the bind_target_scope is an interface_identifier or the bind_target_instance is an interface_instance_idenifier, the bind instantiation must be an interface instantiation.

Replace in 22.10

Possible target scopes include module, program, and interface declarations.

With

~~Possible target scopes include module, program, and interface declarations.~~
A bind target scope shall be a module or an interface. A bind target instance shall be an instance of a module or an interface.

Replace in section 1.8 Contents of this standard

Clause 16 describes immediate assertions, concurrent assertions, properties, sequences, sequence operations, multiclock sequences, clock resolution, and assertion binding.

…

Clause 22 describes how hierarchies are created in SystemVerilog using module instances and interface instances, and port connection rules. Also discusses the $root top-level instances, nested modules, extern modules, identifier search rules, and how parameter values can be overridden.

With

Clause 16 describes immediate assertions, concurrent assertions, properties, sequences, sequence operations, multiclock sequences, and clock resolution. ~~clock resolution, and assertion binding~~.

…

Clause 22 describes how hierarchies are created in SystemVerilog using module instances and interface instances, and port connection rules. Also discusses the $root top-level instances, nested modules, extern modules, identifier search rules, ~~and~~ how parameter values can be overridden, and binding auxilliary code to scopes or instances.