11.5 `include

This directive is used to insert the entire contents of a source file in another file during compilation. The result is as though the contents of the included source file appear in place of the **`include** compiler directive. **`include** can be used to include global or commonly used definitions and tasks without encapsulating repeated code within module boundaries.

This directive can be useful in the following situations:

- providing an integral part of configuration management;
- improving the organization of Verilog-AMS HDL source descriptions; or
- facilitating the maintenance of Verilog-AMS HDL source descriptions.

Addressing Issue #15 >>

The syntax for **`include** is shown in Syntax 11-7.

include_compiler_directive ::= system_include | user_include

user_include ::= **`include "**filename**"**

system_include ::= **`include** <filename>

Syntax 11-7—Syntax for include compiler directive

The compiler directive **`include** can be specified anywhere within the Verilog-AMS HDL description. The *filename* is the name of the file to be included in the source file. The *filename* can be a full or relative path name. Since the Verilog-AMS standard uses external files to define standard constants and disciplines, and those files will normally be located with the simulator rather than in the users directories, the *angle-bracket* syntax is used to differentiate which is intended. If '"' quoting is used then the user directories are searched first and then the simulator installation directories, and vice versa if '<>' quoting is used.

Only white space or a comment can appear on the same line as the **`include** compiler directive.

A file included in the source using **`include** can contain other **`include** compiler directives. The number of nesting levels for included files are finite.

Examples:

```
`include <constants.h> // Include constants.h from simulator
                       // installation
`include "constants.h" // Include constants.h from user directories
```

```
`include "parts/count.v"

`include "fileA"
`include "fileB" // including fileB
```

Note: Implementations can limit the maximum number of levels to which include files can be nested, but this limit shall be a minimum of 15 levels.