Using Pragmas

A pragma is a meta-comment, something entered in a code comment, that is a message for the software that reads the comment. These pragmas are sometimes called compiler directives even though they do not have the syntax of a compiler directive defined in the IEEE Std 1364.

You use pragmas to specify an FSM that VCS might not automatically extract.

With these pragmas you tell VCS the following things about the FSM:

- The vector signal, part-select of a vector signal, or concatenation of signals that hold the current state of the FSM.
- The vector signal that hold the next state of the FSM, unless the FSM doesn't use a signal for the next state (in which case VCS displays a warning and assumes that the current state and next state are in the same signal, part-select of a signal, or concatenation of signals).
- The possible states of the FSM that are specified in a parameter declaration. When using pragmas to specify an FSM there must be a parameter declaration to specify the possible states of the FSM.

Specifying The Signal That Holds The Current State

You use the following pragma to identify the vector signal that holds the current state of the FSM:

```
/* VCS state_vector signal_name */
```

VCS and `state_vector` are required keywords. You must enter this pragma inside the module definition where the signal is declared.

You also must use a pragma to specify an enumeration name for the FSM. This enumeration name is also specified for the next state and the possible states, associating them with each other as part of the same FSM. There are two ways you can do this:

- Use the same pragma:

```
/* VCS state_vector signal_name enum enumeration_name */
```

- Use a separate pragma in the signal's declaration:

```
/* VCS state_vector signal_name */
reg [7:0] /* VCS enum enumeration_name */ signal_name;
```

In either case `enum` is a required keyword, if using the separate pragma, VCS is also a required keyword. Also, when using a separate pragma, enter the pragma immediately after the bit range of the signal.

Specifying The Part-Select That Holds The Current State

You can specify that a part-select of a vector signal holds the current state of the FSM. When cmView displays or reports FSM coverage data it names the FSM after the signal that holds the current state. If in your FSM a part-select holds the current state, you must also specify a name for the FSM that cmView can use. The FSM name is not the same as the enumeration name.

You specify the part-select with the following pragma:

```
/* VCS state_vector signal_name[n:n] FSM_name enum
enumeration_name */
```

Specifying The Concatenation That Holds The Current State

Like specifying a part-select, you can specify a concatenation of signals to hold the current state, when you do you also specify an FSM name and an enumeration name:

```
/* VCS state_vector {signal_name, signal_name,...} FSM_name
enum enumeration_name */
```

The concatenation is of entire signals. You cannot include in the concatenation bit-selects or part-selects of signals.

Specifying The Signal That Holds The Next State

You also specify the signal that holds the next state of the FSM with the pragma that specifies the enumeration name:

```
reg [7:0] /* VCS enum enumeration_name */
signal_name
```

If, and only if, the FSM does not have a signal for the next state, you can omit this pragma.

Specifying The Current and Next State Signals In The Same Declaration

If you use the pragma for specifying the enumeration name in a declaration of multiple signals, VCS assumes that the first signal following the pragma holds the current state and the next signal holds the next state, for example:

```
/* VCS state_vector cs */
reg [1:0] /* VCS enum myFSM */ cs, ns, nonstate;
```

In this example VCS assumes that signal cs holds the current state and signal ns holds the next state. It assumes nothing about signal nonstate.

Specifying The Possible States of The FSM

You also specify the possible states of the FSM with the pragma that specifies the enumeration name:

```
parameter /* VCS enum enumeration_name */
   S0 = 0,
   s1 = 1,
   s2 = 2,
   s3 = 3;
```

Enter this pragma immediately after the keyword `parameter` unless you specify a bit width for the parameters, in which case enter the pragma immediately after the bit with:

```
parameter [1:0] /* VCS enum enumeration_name */
   S0 = 0,
   s1 = 1,
   s2 = 2,
   s3 = 3;
```

Pragmas In One Line Comments

These pragmas work in both block comments, between the `/*` and `*/` character strings, and one line comments, following the `//` character string, for example:

```
parameter [1:0] // VCS enum enumeration_name
   S0 = 0,
   s1 = 1,
   s2 = 2,
   s3 = 3;
```

Example FSM Specified With Pragmas

```
module m3;

   reg[31:0] cs;
   reg[31:0] /* VCS enum MY_FSM */ ns;
   reg[31:0] clk;
   reg[31:0] rst;

// VCS state_vector cs enum MY_FSM

   parameter // VCS enum MY_FSM
             p1=10,
             p2=11,
             p3=12;

  endmodule // m3
```