8.6 Supplementary driver access functions

The following driver access functions are provided for access to digital events which have been scheduled onto a driver but might not have matured by the current simulation time.

These functions can be used to create analog waveforms which cross a specified threshold at the same time the digital event matures, thus providing accurate registration of analog and digital representations of a signal. This assumes there is at least as long a delay in the maturation of the digital signal as the required rise/fall times of the analog waveform.

Note: Because the scheduled digital events can be scheduled with an insufficient delay or cancelled before they mature, be careful when using these functions.

8.6.1 $driver_delay

$driver_delay returns the delay, from current simulation time, after which the pending state or strength becomes active. If there is no pending value on a signal, it returns zero (`0`). The syntax is shown in Syntax 8-13.

```
driver_delay_function ::=
    $driver_delay ( signal_name , driver_index )
```

Syntax 8-13—Syntax for $driver_delay

driver_index is an integer value between `0` and `N-1`, where `N` is the total number of drivers contributing to the signal value. The returned delay value is an integer.

8.6.2 $driver_next_state

$driver_next_state returns the pending state of the driver, if there is one. If there is no pending state, it returns the current state. The syntax is shown in Syntax 8-14.

```
driver_next_state_function ::=
    $driver_next_state ( signal_name , driver_index )
```

Syntax 8-14—Syntax for $driver_next_state

driver_index is an integer value between `0` and `N-1`, where `N` is the total number of drivers contributing to the signal value. The pending state value is returned as 1’b`0`, 1’b`1`, 1’b`x`, or 1’b`z`.

8.6.3 $driver_next_strength

$driver_next_strength returns the strength associated with the pending state of the driver, if there is one. If there is no pending state, it returns the current strength. The syntax is shown in Syntax 8-15.

```
driver_next_strength_function ::=
    $driver_next_strength ( signal_name , driver_index )
```

Syntax 8-15—Syntax for$ driver_next_strength

driver_index is an integer value between `0` and `N-1`, where `N` is the total number of drivers contributing to the signal value. The pending strength value is returned as an integer between `0` and `7`.

8.6.4 $driver_type

Digital primitives (like nand and nor gates) should always provide data about their scheduled driver changes, i.e. a gate with a 5ns delay should provide 5ns of look-ahead. Behavioral code with blocking assigns cannot provide look-ahead, but non-blocking assigns with delays can, however since the capability is implementation and configuration dependent this function is provided so that the connect module can adapt for a particular instance. The syntax is shown in Syntax 8-16.

```
driver_type_function ::=
    $driver_type( signal_name , driver_index )
```

Syntax 8-16—Syntax for$ driver_type

The returned value is an integer with its bits set according to the system header file “driver_access.v”. The names and values of these bit values are as follows:

```
‘define DRIVER_UNKNOWN      32’b00000000 // No information
‘define DRIVER_DELAYED      32‘b00000001 // driver has fixed delay
‘define DRIVER_GATE         32’b00000010 // driver is a primitive
‘define DRIVER_UDP          32’b00000100 // driver is a user defined primitive
‘define DRIVER_ASSIGN       32’b00001000 // driver is a continuous assignment
‘define DRIVER_BEHAVIORAL   32’b00010000 // driver is a reg
‘define DRIVER_SDF          32’b00100000 // driver is from backannotated code
‘define DRIVER_NODELETE     32’b01000000 // events won’t be deleted[1]
‘define DRIVER_NOPREEMPT    32’b10000000 // events won’t be preempted
```

If a simulator cannot support this functionality fully e.g. it cannot provide look-ahead on a gate's driver it should return 'DRIVER_UNKNOWN, rather than return 'DRIVER_GATE.

Connect modules for digital to analog conversion can use the returned information to help minimize the difference between the digital event time and the analog crossover when the user swaps between coding styles and performs backannotation[1].

1. For optimization
1. SDF backannotation will not change which D2A is inserted.