23. Simulation-related constructs

This clause describes simulation-related actions and expressions. See also Clause 9.

Simulators can be attached to the ***e*** runtime environment by means of a simulator adapter. In addition, it may be necessary to infer support structures, such as extra registers, to facilitate the integration. Such structures are placed in a *stubs file*, which can be generated by the ***e*** compiler. The simulation interface and stub file generation functions are implementation-dependent.

23.1 force

Purpose	Force a value on an HDL object	
Category	Action	
Syntax	**force** '*HDL-pathname*' **=** *exp*	
Parameters	*HDL-pathname*	The full pathname of an HDL object (see 23.3), including any expressions.
	exp	Any scalar expression or literal constant, as long as it is composed only of `1`'s and `0`'s. No `x` or `z` values are allowed. Thus, `16'hf0f1` or `sys.my_val+5` are legal values.

This forces an HDL object to a specified value, overriding the current value and preventing the DUT from driving any value. The HDL object remains at the specified value until a subsequent **force** action from ***e*** or until freed by a **release** action (see 23.2). The following also apply:

— If part of a vectored object is forced, the **force** action is propagated to the rest of the object.

— To force single elements of an array of a scalar integer or enumerated type, use the predefined routine **simulator_command()** (see 28.11.1).

Syntax example:

```
force '~/top/sig' = 7
```

23.2 release

Purpose	Remove a **force** action from an HDL object	
Category	Action	
Syntax	**release** '*HDL-pathname*'	
Parameters	*HDL-pathname*	The full pathname of an HDL object previously specified in a **force** action.

This releases the HDL object that previously has been forced (see 23.1).

Syntax example:

```
release 'top.sig'
```

23.3 Tick access: 'hdl-pathname'

Purpose	Access HDL objects, using *full-path-names*	
Category	Expression	
Syntax	'*HDL-pathname*[*index-exp* | *bit-range*] [**@(x | z | n)**]'	
Parameters	*HDL-pathname*	The full pathname of an HDL object, including any expressions and composite data.
	index-exp	Accesses a single bit of a Verilog vector, a single element of a Verilog memory, or a single vector of a VHDL array of vectors.
	bit-range	*bit-range* has the format [*high-bit-num* **:** *low-bit-num*] and is extracted from the object from the high bit to low bit. Slices of buses are treated exactly as they are in HDL languages. They need to be specified in the same direction as in the HDL code and reference the same bit numbers.
	@x | **@z**	Sets or gets the `x` or `z` component of the value. When this notation is not used in accessing an HDL object, ***e*** translates the values of `x` to zero (`0`) and `z` to one (`1`). When reading HDL objects using **@x** (or **@z**), ***e*** translates the specified value (`x` or `z`) to one (`1`) and all other values to zero (`0`). When writing HDL objects, if **@x** (or **@z**) is specified, ***e*** sets every bit that has a value of `1` to `x` (or `z`). In this way, **@x** or **@z** acts much like a data mask, manipulating only those bits that match the value of `x` or `z`.
	@n	When this specifier is used for driving HDL objects, the new (simulator) value is visible immediately (now). The *default mode* is to buffer projected values and update only at the end of the tick.

This accesses Verilog and VHDL objects from ***e***.

Syntax example:

```
'~/top/sig' = 7;
print '~/top/sig'
```