

SMEIL Language Reference

Grammar

$\langle module \rangle$	$::= \{ \langle import-stm \rangle \}$ $\{ \langle type-def \rangle \}$ $\langle entity \rangle \{ \langle entity \rangle \}$
$\langle import-stm \rangle$	$::= \text{'import' } \langle import-name \rangle [\langle qualified-specifier \rangle] \text{' ;'}$ $ \text{'from' } \langle import-name \rangle$ $\text{'import' } \langle ident \rangle \{ \text{' , ' } \langle ident \rangle \} [\langle qualified-specifier \rangle]$ ' ;'
$\langle import-name \rangle$	$::= \langle ident \rangle \{ \text{' . ' } \langle ident \rangle \}$
$\langle qualified-specifier \rangle$	$::= \text{'as' } \langle ident \rangle$
$\langle type-def \rangle$	$::= \text{'type' } \langle ident \rangle \text{' : '}$ $\langle type-name \rangle \text{ (type alias)}$ $ \langle bus-signal-decls \rangle \text{ (bus definition)}$ ' ;'
$\langle entity \rangle$	$::= \langle network \rangle$ $ \langle process \rangle$
$\langle network \rangle$	$::= \text{'network' } \langle ident \rangle \text{' (' } [\langle params \rangle] \text{') '}$ $\text{' { ' } \{ \langle network-decl \rangle \} \text{' } \}$
$\langle process \rangle$	$::= [\text{'clocked' }] \text{'proc' } \langle ident \rangle$ $\text{' (' } [\langle params \rangle] \text{') ' } \{ \langle process-decl \rangle \}$ $\text{' { ' } \{ \langle statement \rangle \} \text{' } \}$
$\langle network-decl \rangle$	$::= \langle inst-decl \rangle$ $ \langle bus-decl \rangle$ $ \langle const-decl \rangle$ $ \langle gen-decl \rangle$ $ \langle connect-decl \rangle$

$\langle process-decl \rangle$	$::=$ $\langle var-decl \rangle$ $ $ $\langle const-decl \rangle$ $ $ $\langle bus-decl \rangle$ $ $ $\langle enum-decl \rangle$ $ $ $\langle func-decl \rangle$ $ $ $\langle inst-decl \rangle$ $ $ $\langle gen-decl \rangle$
$\langle params \rangle$	$::= \langle param \rangle \{ , \langle param \rangle \}$
$\langle param \rangle$	$::=$ $[[[\langle integer \rangle]]] \langle direction \rangle \langle ident \rangle [':' \langle type-name \rangle$ $]$
$\langle direction \rangle$	$::=$ 'in' (input signal) $ $ 'out' (output signal) $ $ 'const' (constant input value)
$\langle var-decl \rangle$	$::=$ 'var' $\langle ident \rangle$ $':'$ $\langle type-name \rangle ['=' \langle expression \rangle] [\langle range \rangle] ';' ;$
$\langle range \rangle$	$::= \text{'range'}$ $\langle expression \rangle$ 'to' $\langle expression \rangle$
$\langle enum-decl \rangle$	$::=$ 'enum' $\langle ident \rangle$ $\text{'{'}$ $\langle enum-field \rangle \{ , \langle enum-field \rangle \}$ $\text{'}'$ $ ';' ;$
$\langle enum-field \rangle$	$::= \langle ident \rangle ['=' \langle integer \rangle]$
$\langle const-decl \rangle$	$::= \text{'const'}$ $\langle ident \rangle$ $':'$ $\langle type-name \rangle$ $'='$ $\langle expression \rangle$ $ ';' ;$
$\langle bus-decl \rangle$	$::= [\text{'clocked'}] \text{'bus'}$ $\langle ident \rangle$ $\langle bus-decl-content \rangle$ $ ';' ;$
$\langle func-decl \rangle$	$::=$ 'function' $\langle ident \rangle$ '(' $\langle params \rangle$ $\text{'})'}$ $\text{'{'}$ $\{ \langle statement \rangle \}$ $\text{'}'$ $ ';' ;$
$\langle bus-decl-content \rangle$	$::=$ $\text{'{'}$ $\langle bus-signal-decls \rangle$ $\text{'}'$ $ $ $\langle type-name \rangle$
$\langle bus-signal-decls \rangle$	$::= \langle bus-signal-decl \rangle \{ \langle bus-signal-decl \rangle \}$
$\langle bus-signal-decl \rangle$	$::=$ $\langle ident \rangle$ $':'$ $\langle type-name \rangle ['=' \langle expression \rangle] [\langle range \rangle]$ $ ';' ;$
$\langle connect-entry \rangle$	$::= \langle name \rangle \text{'->'}$ $\langle name \rangle$
$\langle connect-decl \rangle$	$::= \text{connect}$ $\langle connect-entry \rangle \{ , \langle connect-entry \rangle \} ';' ;$
$\langle inst-decl \rangle$	$::=$ 'instance' $\langle instance-name \rangle$ 'of' $\langle ident \rangle$ '(' $[\langle param-map \rangle \{ , \langle param-map \rangle \}]$ $\text{'})'}$ $ ';' ;$

$\langle \text{instance-name} \rangle$	$::= \langle \text{ident} \rangle \text{'['} \langle \text{expression} \rangle \text{'}'}$ (indexed instance) $ \langle \text{ident} \rangle$ (named instance) $ \text{'_'}$ (anonymous instance)
$\langle \text{param-map} \rangle$	$::= [\langle \text{ident} \rangle \text{'.'}] \langle \text{expression} \rangle$
$\langle \text{gen-decl} \rangle$	$::= \text{'generate'} \langle \text{ident} \rangle \text{'='} \langle \text{expression} \rangle \text{'to'} \langle \text{expression} \rangle$ $\text{'{' } \{ \langle \text{network-decl} \rangle \} \text{'}'}$
$\langle \text{statement} \rangle$	$::= \langle \text{name} \rangle \text{'='} \langle \text{expression} \rangle \text{';'}$ (assignment) $ \langle \text{ident} \rangle \text{'('} \langle \text{param-map} \rangle \text{'')'}$ (function call) $ \text{'if'} \text{'('} \langle \text{expression} \rangle \text{'')' } \text{'{' } \{ \langle \text{statement} \rangle \} \text{'}'}$ $\{ \langle \text{elif-block} \rangle \} [\langle \text{else-block} \rangle]$ $ \text{'for'} \langle \text{ident} \rangle \text{'='} \langle \text{expression} \rangle \text{'to'} \langle \text{expression} \rangle$ $\text{'{' } \{ \langle \text{statement} \rangle \} \text{'}'}$ $ \text{'switch'} \langle \text{simple-expression} \rangle$ $\text{'{' } \langle \text{switch-case} \rangle \{ \langle \text{switch-case} \rangle \} [\text{'default'} \text{'{' } \langle \text{statement} \rangle$ $\{ \langle \text{statement} \rangle \} \text{'}' }] \text{'}'}$ $ \text{'trace'} \text{'('} \langle \text{format-string} \rangle \{ \text{','} \langle \text{expression} \rangle \} \text{'')'}$ $ \text{'assert'} \text{'('} \langle \text{expression} \rangle [\text{','} \langle \text{string-literal} \rangle] \text{'')'}$ $ \text{'break'} \text{';'}$
$\langle \text{switch-case} \rangle$	$::= \text{'case'} \langle \text{simple-expression} \rangle \text{'{' } \{ \langle \text{statement} \rangle \} \text{'}'}$
$\langle \text{elif-block} \rangle$	$::= \text{'elif'} \text{'('} \langle \text{expression} \rangle \text{'')' } \text{'{' } \{ \langle \text{statement} \rangle \} \text{'}'}$
$\langle \text{else-block} \rangle$	$::= \text{'else'} \text{'{' } \{ \langle \text{statement} \rangle \} \text{'}'}$
$\langle \text{format-string} \rangle$	$::= \text{'\"'} \{ \langle \text{format-string-part} \rangle \} \text{'\"'}$
$\langle \text{format-string-part} \rangle$	$::= \text{'{'}}$ (placeholder string) $ \langle \text{string-char} \rangle$
$\langle \text{simple-expression} \rangle$	$::= \langle \text{literal} \rangle$ $ \langle \text{name} \rangle$
$\langle \text{expression} \rangle$	$::= \langle \text{simple-expression} \rangle$ $ \langle \text{expression} \rangle \langle \text{bin-op} \rangle \langle \text{expression} \rangle$ $ \langle \text{un-op} \rangle \langle \text{expression} \rangle$ $ \text{'('} \langle \text{expression} \rangle \text{'}'}$ $ \text{'('} \langle \text{name} \rangle \text{'')' } \langle \text{expression} \rangle$ (type cast)
$\langle \text{bin-op} \rangle$	$::= \text{'+'}$ (addition) $ \text{'-'}$ (subtraction) $ \text{'*'}$ (multiplication) $ \text{'/'}$ (division) $ \text{'%'}$ (modulo)

		'==' (equal)
		'!=' (not equal)
		'<<' (shift left)
		'>>' (shift right)
		'<' (less than)
		'>' (greater than)
		'>=' (greater than or equal)
		'<=' (less than or equal)
		'&' (bitwise-and)
		' ' (bitwise-or)
		'^' (bitwise-xor)
		'&&' (logical conjunction)
		' ' (logical disjunction)
$\langle un-op \rangle$::=	'-' (negation)
		'+' (identity)
		'!' (logical negation)
		'~' (bitwise-not)
$\langle literal \rangle$::=	$\langle integer \rangle$
		$\langle floating \rangle$
		$\langle string-literal \rangle$
		'[' $\langle integer \rangle$ { ',' $\langle integer \rangle$ } ']' (Array literal)
		'true'
		'false'
		'U' (Undefined value)
$\langle string-literal \rangle$::=	'"{ $\langle string-char \rangle$ }"'
$\langle intrinsic-type \rangle$::=	'i' $\langle integer \rangle$ (signed integer)
		'int' (arbitrary-width signed integer)
		'u' $\langle integer \rangle$ (unsigned integer)
		'uint' (arbitrary-width unsigned integer)
		'f32' (single-precision floating point)
		'f64' (double-precision floating point)
		'bool' (boolean value)
$\langle type-name \rangle$::=	$\langle intrinsic-type \rangle$
		$\langle name \rangle$ (type definition)
		'[' [$\langle expression \rangle$] ']' $\langle type-name \rangle$ (array of type)
$\langle ident \rangle$::=	$\langle letter \rangle$ { $\langle letter \rangle$ $\langle number \rangle$ '_' '-' } (identifier)
$\langle name \rangle$::=	$\langle ident \rangle$
		$\langle name \rangle$ '.' $\langle name \rangle$ (hierarchical accessor)
		$\langle name \rangle$ '[' $\langle array-index \rangle$ ']' (array element access)

$\langle array-index \rangle$::= '*' (wildcard) $\langle expression \rangle$ (element index)
$\langle integer \rangle$::= $\langle number \rangle$ { $\langle number \rangle$ } (decimal number) '0x' $\langle hex-digit \rangle$ { $\langle hex-digit \rangle$ } (hexadecimal number) '0o' $\langle octal-digit \rangle$ { $\langle octal-digit \rangle$ } (octal number)
$\langle floating \rangle$::= { $\langle number \rangle$ } '.' $\langle number \rangle$ { $\langle number \rangle$ }
$\langle number \rangle$::= '0' - '9'
$\langle letter \rangle$::= 'a' - 'z' 'A' - 'Z'
$\langle hex-digit \rangle$::= $\langle number \rangle$ 'a' - 'f' 'A' - 'F'
$\langle octal-digit \rangle$::= '0' - '8'
$\langle string-char \rangle$::= (ISO-8859-1 char with value > 26)

Operator precedence

Precedence	Operators
0	+ - ! ~ (unary)
1	* / %
2	+ -
3	<< >>
4	< > <= >=
5	== !=
6	& ^
7	&&
8	

Keywords

- as
- async
- await
- barrier
- break
- bus
- case
- const
- connect
- clocked
- default
- elif
- else
- enum
- exposed

- for
- from
- func
- generate
- if
- import
- in
- instance
- network
- of
- out
- proc
- range
- return
- switch
- sync
- to
- unique
- var
- wait
- where