SMEIL Language Reference

Grammar

⟨*module*⟩   ::=   { ⟨*import-stm*⟩ }
{ ⟨*type-def*⟩ }
⟨*entity*⟩ { ⟨*entity*⟩ }

⟨*import-stm*⟩   ::=   'import' ⟨*import-name*⟩ [⟨*qualified-specifier*⟩] ';'
  | 'from' ⟨*import-name*⟩
'import' ⟨*ident*⟩ { ',' ⟨*ident*⟩ } [⟨*qualified-specifier*⟩]
';'

⟨*import-name*⟩   ::=   ⟨*ident*⟩ { '.' ⟨*ident*⟩ }

⟨*qualified-specifier*⟩   ::=   'as' ⟨*ident*⟩

⟨*import-decl*⟩   ::=   ⟨*type-def*⟩
  | ⟨*const-decl*⟩
  | ⟨*enum-decl*⟩
  | ⟨*func-decl*⟩

⟨*type-def*⟩   ::=   'type' ⟨*ident*⟩ ':'
⟨*type-name*⟩ (type alias)
| ⟨*bus-signal-decls*⟩ (bus definition)
';'

⟨*entity*⟩   ::=   ⟨*network*⟩
  | ⟨*process*⟩

⟨*network*⟩   ::=   'network' ⟨*ident*⟩ '(' [⟨*params*⟩] ')'
'{' { ⟨*network-decl*⟩ } '}'

⟨*process*⟩   ::=   ['clocked'] 'proc' ⟨*ident*⟩
'(' [⟨*params*⟩] ')' { ⟨*process-decl*⟩ }
'{' { ⟨*statement*⟩ } '}'

⟨*network-decl*⟩   ::=   ⟨*inst-decl*⟩
  | ⟨*bus-decl*⟩

| ⟨*const-decl*⟩
| ⟨*gen-decl*⟩
| ⟨*connect-decl*⟩

⟨*process-decl*⟩   ::= ⟨*var-decl*⟩
| ⟨*const-decl*⟩
| ⟨*bus-decl*⟩
| ⟨*enum-decl*⟩
| ⟨*func-decl*⟩
| ⟨*inst-decl*⟩
| ⟨*gen-decl*⟩

⟨*func-decls*⟩   ::= ⟨*var-decl*⟩
| ⟨*const-decl*⟩
| ⟨*enum-decl*⟩

⟨*params*⟩   ::= ⟨*param*⟩ { , ⟨*param*⟩ }

⟨*param*⟩   ::= ['`[`' [⟨*integer*⟩] '`]`'] ⟨*direction*⟩ ⟨*ident*⟩ [':' ⟨*type-name*⟩]

⟨*direction*⟩   ::= '`in`' (input signal)
| '`out`' (output signal)
| '`const`' (constant input value)

⟨*var-decl*⟩   ::= '`var`' ⟨*ident*⟩ ':'
⟨*type-name*⟩ ['=' ⟨*expression*⟩] [⟨*range*⟩] ';'

⟨*range*⟩   ::= '`range`' ⟨*expression*⟩ '`to`' ⟨*expression*⟩

⟨*enum-decl*⟩   ::= '`enum`' ⟨*ident*⟩
'`{`' ⟨*enum-field*⟩ { ',' ⟨*enum-field*⟩ } '`}`' ';'

⟨*enum-field*⟩   ::= ⟨*ident*⟩ ['=' ⟨*integer*⟩]

⟨*const-decl*⟩   ::= '`const`' ⟨*ident*⟩ ':' ⟨*type-name*⟩ '=' ⟨*expression*⟩ ';'

⟨*bus-decl*⟩   ::= ['`clocked`'] '`bus`' ⟨*ident*⟩ ⟨*bus-decl-content*⟩ ';'

⟨*func-decl*⟩   ::= '`function`' ⟨*ident*⟩ '(' ⟨*params*⟩ ')' { ⟨*func-decls*⟩ } '`{`'
{ ⟨*statement*⟩ } '`}`'

⟨*bus-decl-content*⟩   ::= '`{`' ⟨*bus-signal-decls*⟩ '`}`'
| ⟨*type-name*⟩

⟨*bus-signal-decls*⟩   ::= ⟨*bus-signal-decl*⟩ { ⟨*bus-signal-decl*⟩ }

$\langle$*bus-signal-decl*$\rangle$ ::= $\langle$*ident*$\rangle$ ':' $\langle$*type-name*$\rangle$ ['=' $\langle$*expression*$\rangle$] [$\langle$*range*$\rangle$]
';'

$\langle$*connect-entry*$\rangle$ ::= $\langle$*name*$\rangle$ '->' $\langle$*name*$\rangle$

$\langle$*connect-decl*$\rangle$ ::= 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*$\rangle$ }] ')' ';'

$\langle$*instance-name*$\rangle$ ::= $\langle$*ident*$\rangle$ '**[**' $\langle$*expression*$\rangle$ '**]**' (indexed instance)
| $\langle$*ident*$\rangle$ (named instance)
| '**_**' (anonymous instance)

$\langle$*param-map*$\rangle$ ::= [$\langle$*ident*$\rangle$ ':'] $\langle$*expression*$\rangle$

$\langle$*gen-decl*$\rangle$ ::= '**generate**' $\langle$*ident*$\rangle$ '=' $\langle$*expression*$\rangle$ '**to**' $\langle$*expression*$\rangle$
'{' { $\langle$*network-decl*$\rangle$ } '}'

$\langle$*statement*$\rangle$ ::= $\langle$*name*$\rangle$ '=' $\langle$*expression*$\rangle$ ';' (assignment)
| $\langle$*ident*$\rangle$ '(' $\langle$*param-map*$\rangle$ ')'';' (function call)
| '**if**' '(' $\langle$*expression*$\rangle$ ')' '{' { $\langle$*statement*$\rangle$ } '}'
{ $\langle$*elif-block*$\rangle$ } [$\langle$*else-block*$\rangle$]
| '**for**' $\langle$*ident*$\rangle$ '=' $\langle$*expression*$\rangle$ '**to**' $\langle$*expression*$\rangle$
'{' { $\langle$*statement*$\rangle$ } '}'
| '**switch**' $\langle$*simple-expression*$\rangle$
'{' $\langle$*switch-case*$\rangle$ { $\langle$*switch-case*$\rangle$ } ['**default**' '{' $\langle$*statement*$\rangle$
{ $\langle$*statement*$\rangle$ } '}'] '}'
| '**trace**' '(' $\langle$*format-string*$\rangle$ { ',' $\langle$*expression*$\rangle$ } ')'';'
| '**assert**' '(' $\langle$*expression*$\rangle$ [',' $\langle$*string-literal*$\rangle$] ')'';'
| '**break**' ';'

$\langle$*switch-case*$\rangle$ ::= '**case**' $\langle$*simple-expression*$\rangle$ '{' { $\langle$*statement*$\rangle$ } '}'

$\langle$*elif-block*$\rangle$ ::= '**elif**' '(' $\langle$*expression*$\rangle$ ')' '{' { $\langle$*statement*$\rangle$ } '}'

$\langle$*else-block*$\rangle$ ::= '**else**' '{' { $\langle$*statement*$\rangle$ } '}'

$\langle$*format-string*$\rangle$ ::= '"' { $\langle$*format-string-part*$\rangle$ } '"'

$\langle$*format-string-part*$\rangle$ ::= '**{}**' (placeholder string)
| $\langle$*string-char*$\rangle$

$\langle$*simple-expression*$\rangle$ ::= $\langle$*literal*$\rangle$
| $\langle$*name*$\rangle$

$\langle expression \rangle$ ::= $\langle simple\text{-}expression \rangle$
         | $\langle expression \rangle$ $\langle bin\text{-}op \rangle$ $\langle expression \rangle$
         | $\langle un\text{-}op \rangle$ $\langle expression \rangle$
         | '(' $\langle expression \rangle$ ')'
         | '(' $\langle type\text{-}name \rangle$ ')' $\langle expression \rangle$ (type cast)

$\langle bin\text{-}op \rangle$ ::= '+' (addition)
         | '-' (subtraction)
         | '*' (multiplication)
         | '/' (division)
         | '%' (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)

$\langle un\text{-}op \rangle$ ::= '-' (negation)
         | '+' (identity)
         | '!' (logical negation)
         | '~' (bitwise-not)

$\langle literal \rangle$ ::= $\langle integer \rangle$
         | $\langle floating \rangle$
         | $\langle string\text{-}literal \rangle$
         | '[' $\langle integer \rangle$ { ',' $\langle integer \rangle$ } ']' (Array literal)
         | 'true'
         | 'false'
         | ''U' (Undefined value)

$\langle string\text{-}literal \rangle$ ::= '"'{ $\langle string\text{-}char \rangle$ }'"'

$\langle intrinsic\text{-}type \rangle$ ::= '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)

⟨*type-name*⟩	::=	⟨*intrinsic-type*⟩
	|	⟨*name*⟩ (type definition)
	|	'[' [⟨*expression*⟩] ']' ⟨*type-name*⟩ (array of type)

⟨*ident*⟩	::=	⟨*letter*⟩ { ⟨*letter*⟩ | ⟨*number*⟩ | '_' | '-' } (identifier)

⟨*name*⟩	::=	⟨*ident*⟩
	|	⟨*name*⟩ '.' ⟨*name*⟩ (hierarchical accessor)
	|	⟨*name*⟩ '[' ⟨*array-index*⟩ ']' (array element access)

⟨*array-index*⟩	::=	'*' (wildcard)
	|	⟨*expression*⟩ (element index)

⟨*integer*⟩	::=	⟨*number*⟩ { ⟨*number*⟩ } (decimal number)
	|	'0x' ⟨*hex-digit*⟩ { ⟨*hex-digit*⟩ } (hexadecimal number)
	|	'0o' ⟨*octal-digit*⟩ { ⟨*octal-digit*⟩ } (octal number)

⟨*floating*⟩	::=	{ ⟨*number*⟩ } '.' ⟨*number*⟩ { ⟨*number*⟩ }

⟨*number*⟩	::=	'0' - '9'

⟨*letter*⟩	::=	'a' - 'z'
	|	'A' - 'Z'

⟨*hex-digit*⟩	::=	⟨*number*⟩
	|	'a' - 'f'
	|	'A' - 'F'

⟨*octal-digit*⟩	::=	'0' - '8'

⟨*string-char*⟩	::=	(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