Verilog Supported and Unsupported Constructs

Data Types | Expressions | Continuous Assignments | Procedural Assignments | Gate and Switch Level | Behavioral Modeling | Hierarchical Structures | System Tasks and Functions | Compiler Directives ||| Table of Contents

The following constructs are supported by version 5.5 of **verilog2vhdl**.

Data Types

Supported

- wire
- tri
- supply0
- supply1
- memory
- integer
- time
- real
- reg
- parameter

Not Supported

- nettype(s) **tri1, wand, triand, tri0, wor, trior, trireg**
- expand range, charge strength, drive strength, delay specification for net declaration

Expressions

Supported

- operand types : net, register, bit-select, bit-slice
    - binary logical operators:
    - ||, &&, !=, ==
    - binary relational operators:
    - <, <=, >, >=
- operand types : number, time, integer, net part-select, register part-select
    - operators:
    - {}, arith. operators, mod, !, ===, !==, <<, >>, ?:
    - unary operators:
    - &, ~&, |, ~|, ^, ~^ or ^~

Continuous Assignments

Supported

- Left hand side : net (vector or scalar), constant bit select of a vector net, constant part select of a vector net, concatenation
- Delays of type (rising) only, can be of (min/typ/max) type
- Net declaration Assignment

Not Supported

- drive strength
- delays of type (rising, falling, turnoff)
- (*force, release* continuous assignment

Procedural Assignments

Supported

- Left hand side : register (vector or scalar), constant bit select of a vector register, constant part select of a vector register, memory element, concatenation
- Blocking procedural assignment
- Non-blocking procedural assignment
- Delays of type min/typ/max

Not Supported

- procedural continuous assignment (*assign, deassign, force, release*)

Gate and Switch Level

Supported

- gate type: **and, nand, nor, or, xor, xnor, buf, not, bufif0, bufif1, notif0, notif1**
- Delays of type (rising) only, can be of **(min/typ/max) type**

Not Supported

- gate type: **nmos, pmos, cmos, rnmos, rpmos, rcmos, tran, tranif0, tranif1, rtran, rtranif0, pullup, pulldown**
- drive strength
- delays of type (rising, falling, turnoff)

Behavioral Modeling

Supported

- always

- initial
- conditional if-else-if
- case, casex, casez
- for
- forever, repeat, while loops
- tasks
- function calls

Not Supported

- named events
- parallel blocks (fork/join)

Hierarchical Structures

Supported

- module
- ports: input, output, inout
- module instantiation
- named port connection with concatenated names
- macromodule

Not Supported

- hierarchical names

System Tasks and Functions

Supported

- $display
- $fdisplay
- $write
- $fwrite
- $strobe
- $fstrobe
    - Supported
        - %b, %d format specification (%h and %o are treated as %b)
        - scalars, vectors of nets and registers, string, time, integer type
    - Not Supported
        - Type conversions for scalars, vectors of nets and registers, string, time, integer type

        - Format specification %c

- $readmemh
- $readmemb
    - Supported
        - 1 entry per line in data file; only '//' type of comments
        - No address or two addresses in $readmem call
        - constant integer addresses
    - Not Supported
        - Multiple entries per line in memory data file; '/* */' type of comments
        - One address in $readmem call
        - vector addresses

- $fopen
- $fclose
- $time
- $realtime
- $timescale
- $rtoi

Not Supported

- all other system tasks
- all other system functions

Compiler Directives

Supported

- `timescale
- `define
- `ifdef / `ifndef (with **vpp**)

Not Supported

- all others

Recommended Verilog Coding Style

This section is devoted to making the user conversant with what is best translated by **verilog2vhdl**. This is a very useful section for users who intend to use the tool on a continuous basis *e.g* library model developers. By enforcing the use of the constructs supported by **verilog2vhdl** in model development, a fully automatic VHDL translation can be achieved.

verilog2vhdl supports a very large subset of the Verilog language. The user should read the Supported and unsupported section for a complete understanding of supported constructs.

As a rule of the thumb, almost all synthesizable constructs are supported by the tool.

Unsupported constructs

The following constructs are NOT supported by **verilog2vhdl**. The user is strongly advised not to use these constructs in the input file. The user is warned via messages when these constructs are encountered in the input description. Comments are also inserted in the output file to pinpoint the relative location of the unsupported construct.

All **verilog2vhdl** generated comments have the following format:

```
-- *** NOTE: In file <input filename>, at line <line number in input
file>:
-- *** NOTE: <message indicating the type of construct not translated>
```

If the user has indeed no choice but to use the following constructs, the Known problems section has information on the manual editing required to obtain equivalent VHDL.

Verilog constructs not supported are listed below:

- UPDs
- *assign* and *deassign* procedural assignments
- *force* and *release* procedural assignments
- parallel blocks
- task disable
- specify blocks

Supported Constructs

The following Verilog constructs are supported. When applicable, each sub-section also has some relevant do's, dont's and caveats.

1. Numbers:

   All forms of numbers are supported. When using numbers in the binary, decimal, hex, or octal format always use sized vectors *e.g* 4'b 0010 for best results.

2. Identifiers:

   All VHDL keywords (see **verilog2vhdl** User's and Reference Manual) are included in the **verilog2vhdl** reserved list of identifiers.

   Always:

- Avoid hierarchical names
- Avoid using identifiers differing only in case; VHDL is a case-insensitive language
- Avoid using extended identifiers if not generating VHDL-93
- Make sure the Verilog identifier conforms to the following VHDL identifier requirement:
  letter{_}letter_or_digit

3. Data Types:

   Do not use

   - strengths in net declaration
   - net types other than `wire', `tri0', `tri1', `supply0' and `supply1'.

4. Operators and Expressions:

   Do not use

   - `signed and `unsigned compiler directives
   - delay of the type *(mintypmax_expression, mintypmax_expression, mintypmax_expression)*.

5. Continuous Assignments:

   Do not use

   - strengths in net assignment.
   - delay of the type *(mintypmax_expression, mintypmax_expression, mintypmax_expression)*.

6. Procedural Assignments:

   Do not

   - drive a Verilog register in more than one block. This can result in mismatches between synthesis and simulation models. If there are multiple drivers for a design, **verilog2vhdl** inserts the code needed to turn off drivers of inactive processes.
   - use a non-blocking assignment with intra-assignment delay.

7. Gate level Modeling:

   Do not use switches *e.g nmos, pmos*. All gates are supported.

8. Module instantiation:

   Do not

   - connect output ports using expressions; *i.e* do not use the concatenation operator in the expression.
   - use parameter value assignment.

9. always and initial blocks:

   If a signal needs to be initialized in the initial block, it needs to be assigned before any VHDL *wait* statement. In such cases, the initialize is moved up as a signal (or variable) initialization statement.

10. Tasks, Functions, Task enables and Function calls:

    Always

    - Declare tasks before they are called.
    - Declare functions before using them in function calls.
    - In functions, be sure to assign to the function return value or variables declared inside the function; *i.e* do not write to registers declared at the module level.

11. System tasks and functions:

    Always

    - limit the usage of system tasks and functions to those supported by the tool (see supported System Tasks and Functions).
    - avoid using $monitor{on,off} system tasks.

VHDL Supported and Unsupported Constructs

Entity Declaration | Architecture Declaration | Packages and Package Bodies | Data Types | Generics | Expressions | Sequential Statements | Concurrent Statements | Block Statements | Process Statements | Concurrent Signal Assignments | Component Instantiations | Generate Statement | Predefined Language Environment ||| Table of Contents

The following is a comprehensive list of VHDL constructs translated by **VHDL2verilog**.

Entity Declaration

Supported:

- Design with a single entity with architecture
- Entity ports: IN , OUT , INOUT , BUFFER , LINKAGE
- Interface element types:
    - BOOLEAN
    - BIT
    - BIT_VECTOR
    - STD_ULOGIC
    - STD_LOGIC
    - STD_ULOGIC_VECTOR
    - STD_LOGIC_VECTOR
    - INTEGER

Not supported:

- Interface element types:
    - CHARACTER
    - STRING
    - REAL
- Design with no architecture
- Entity statements

Architecture Declaration

Supported:

- Multiple architectures for single entity declaration
- (Simple) Configuration declaration

Not supported:

- Design with an architecture and no entity
- Configuration specification

Packages and Package Bodies

Supported:

- Signal, Variable, Constant declaration
- Type declaration
- Subtype declaration
- Enumerated type declaration
- Component declaration
- Subprogram declaration
- Subprogram body (requires additional manual translation)

NOTE: Packages are translated *only* when used in a design!

Data Types

Supported:

- Signals/Variables/Constants of BASE subtype:
    - integer
    - real
    - bit
    - bit_vector
    - std_{u}logic
    - std_{u}logic_vector
    - character
    - string
    - alias declarations
- Signal/Variable initialization
    - to bit
    - to vector
    - to hex
- Subtype declarations with range constraint
  Limited Support: 2D arrays of supported types
- 1D arrays of supported types equivalent to arrays of bits up to 2D
- **NOTE:** for the above two cases, array types have to be CONSTRAINED.
- Enumerated type declaration

    - Signal of enumerated type (state variable) translated to `reg`

- Enumerated types declared in a:

    - Package
    - Architecture
    - Block
    - Process

- Time types
- Record types
- Based literals (only base 2, 8, 10, 16)

Not supported:

    - Unconstrained types
    - Files

Generics

Supported:

- Generics of base type as described in *Data Types* with default expression

    - integer
    - real
    - bit
    - bit_vector
    - std_{u}logic
    - std_{u}logic_vector
    - TIME

Not supported:

- Generics without default expression

Expressions

Supported:

- Expressions using signal and variables of types described in *Data Types*.
- Expressions with all VHDL supported operators:

    - +
    - -
    - &

    - AND, OR, XOR, NAND, NOR, XNOR (93)
    - unary + and -
    - *
    - /
    - MOD
    - =, /=, <, >, <=, >=
    - SLL, SRL
    - ** (power) operator (*Only when used in a `GENERATE` statement*)

- Qualified expressions
- Type conversions
- Function calls
- Aggregate primaries in an expression

Not supported:

- ** (power) operator (*Except in generates*)
- Allocator primaries

Sequential Statements

Supported:

- Wait statement (only in a process sequential body)
- Signal assignment
    - with INERTIAL delay
    - with TRANSPORT delay
    - NULL assignment
- Variable assignment
- Assignment to an aggregate
- If statement
- Case statement
- Loop statement (FOR WHILE)
- Null statement
- ASSERT statement
- Procedure call statement

Not supported:

- REPORT statement
- NEXT statement
- EXIT statement
- RETURN statement

Concurrent Statements

Supported:

- Block statements
- Process statements
- Conditional signal assignments
- Selected signal assignments
- Component instantiation statements
- Generate statements

Not supported:

- Concurrent procedure calls
- Concurrent assertion statements
- Guarded signal assignments

Block statements

Supported:

- Declarative part:
    - Type declaration
    - Subtype declaration
    - Constant declaration
    - Signal declaration
    - File declaration
    - Component declaration
    - Use clause (package)
- Statement part
    - nested Blocks
    - Process
    - Concurrent assignment
    - Component instantiation
    - Generate

Not supported:

- Ports and port maps
- Generics and generic maps
- Guard expressions

Process Statement

Supported:

- Process variable declaration
- Process with sensitivity list
- Process without sensitivity list
- Process with a WAIT as the first or last sequential statement
- Process with an infinite wait at the end of a sequential body
- Process with a WAIT UNTIL at any place in the sequential body
- Edge-sensitive processes equivalent to Dffs
    - Dffs with/without reset
    - rising/falling_edge function
    - 'EVENT attribute
    - 'STABLE attribute

Concurrent Signal Assignments

Supported:

- Concurrent assignment with delay
- Concurrent assignment to an aggregate
- Concurrent assignment to a target with simple expressions in the range
  NOTE: in Verilog, the expressions have to be CONSTANT
- Conditional assignment
- Selected signal assignment

Not supported:

- Multiple waveform elements in a concurrent signal assignment

Component Instantiations

Supported:

- Generic Maps
    - Generic mapping by ordered list
    - Generic mapping by using formals and actuals
- Port Maps
    - Port mapping by ordered list
    - Port mapping by using formals and actuals
    - Port aspect of component declaration different from the entity
    - Scalar and vector OPENs
    - 2D arrays as ports

- Instantiation of components residing in the same file
- Instantiation of components residing in a package

Generate Statement

Supported:

- IF generate
- FOR loop generate
- Nested generates (FOR/IF)
- Identical labels in generate block and in block enclosing generate
- Component instantiations in generates
- Concurrent assignments in generates
- Processes in generates: regular, edge-sensitive (DFF-style)

Not supported:

- Declarations local to generate (block declarative items)
- Generates with loop parameters dependent on generics

Predefined Language Environment

Supported:

- NOW function
- Time type
- 'RANGE, 'LENGTH, 'LEFT, 'RIGHT, 'LOW, 'HIGH, 'EVENT, 'STABLE, 'LAST EVENT attributes

Not supported:

- Other attributes
- TextIO