Known issues in verilog2vhdl

1. DR 465
   - **Description:** Non-integer values of time_precision in *`timescale* compiler directive are not translated.
   - **Workaround:** reduce the time resolution off the *`timescale* directive so that all values are integers
     Example:

```
`timescale 1.5ns/150ps can be rewritten
as
`timescale 1500ps/150ps
```

2. DR 486
   - **Description:** $readmem calls with only ONE (start) address specified are not always translated correctly.
   - **Workaround:** always specify both the start and the finish addresses for the system task call.
     Example:

```
parameter word_length = 8,
mem_size = 8;
reg [word_length-1:0] mem [0:mem_size-1];
...
$readmemb("mem.dat", mem, 3); // this
will NOT translate correctly
// modify the $readmemb call to
equivalent
$readmemb("mem.dat", mem, 3, mem_size-1);
// this will translate correctly
```

3. DR 488
   - **Description:** Addresses other than integer (i.e. hex and oct) are not supported for $readmem calls.

```
// This call will generate an error
message
$readmemb("mem.dat", mem, 4'h2, 4'hB);
```

   - **Workaround:** Specify all addresses for $readmem calls in integer format

4. DR 504
   - **Description:** Memories used in concatenations on the left hand side of assignments do not get expanded to bits
     Example:

```
reg[7:0] RL [1:8];
reg[7:0] RH [1:8];
reg[7:0] E [1:8];
```

```
integer ADDRESS;
...
{E[ADDRESS],RH[ADDRESS],RL[ADDRESS]} =
DEC_READ(ADDRESS);
translates to:
TYPE MEMORY_0 IS ARRAY (1 TO 8) OF
std_logic_vector(7 DOWNTO 0);
SIGNAL E, RL,RL : MEMORY_0 REGISTER ;
(E(ADDRESS), RH(ADDRESS), RL(ADDRESS)) <=
std_logic_vector' (DEC_READ(REG, E, RH,
RL));
which is incorrect VHDL syntax.
```

- **Workaround:** Use a temporary vector to accommodate the return value of DEC_READ(ADDRESS), and then assign parts of the vector to corresponding memories.

5. DR 676
   - **Description:** Case statement with non-constants as case item expressions translates to VHDL case statement with illegal syntax. Example:

```
reg a;
case (1'b1)
a : d = boo[0];
default : d = boo[3];
translates to:
BEGIN
CASE '1' IS
WHEN a => -- wrong VHDL syntax
V2V_d <= to_stdlogic(boo(0));
WAIT FOR 0 NS;
WHEN OTHERS =>
V2V_d <= to_stdlogic(boo(3));
WAIT FOR 0 NS;
END CASE;
For such cases, verilog2vhdl will print a
warning, so that either Verilog input or
output VHDL can be easily fixed
manually.
```

6. DR 685
   - **Description:** In certain cases, translations of ternary operators with integer arguments produce wrong outputs during simulation if other arguments to the operator assume values X or Z.

7. DR 692
   - **Description:** Translation of registers to variables (-Make_Registers_Variables switch) does not work for tasks which do not have all parameters explicitly declared. Example:

```
task dummy3;
// this task uses global addressing
```

```
// including global referencing of
parameters to a procedure call
begin
s3 = a ^ b ;
dummy2(a, b, c, s4, out3);
end
endtask
```

- **Workaround:** Always specify all parameters to tasks explicitly if you want to use -mrv functionality

8. DR 695/697
   - **Description:** Function calls and task enables where actual parameters do not match formal parameters in size are not translated correctly. In such cases, verilog2vhdl will automatically resize the formal parameters with a size conversion function, which cannot be passed as an output or inout parameter. The same will happen if the actual parameter to a function call/task enable is a concatenation. Example:

```
task MAX;
inout [3:0] first;
input [3:0] second;
begin
if (first < second)
first = second;
end
endtask

...
reg [7:0] a, b;
MAX(a, b);
translates to:
PROCEDURE MAX (SIGNAL first : INOUT
std_logic_vector(3 DOWNTO 0) BUS;
second : IN std_logic_vector(3 DOWNTO 0))
IS
BEGIN
IF first < second THEN
first <= second;
WAIT FOR 0 NS;
END IF;
END;
SIGNAL a, b : std_logic_vector(7 DOWNTO
0);
MAX(to_stdlogicvector(a, 4),
to_stdlogicvector(b, 4));
```

   - **Workaround:** Always make sure that formal and actual parameters of tasks and functions match in size.

9. DR 696
   - **Description:** Subprogram calls made from within other subprograms may result in mismatches between formal and actual parameter classes, e.g. a

variable may be passed when a signal is required.

10. DR 698
    - **Description:** The Verilog preprocessor supplied with verilog2vhdl for the purpose of providing support for compiler directives presently is in beta version, and does not have any user messaging whatsoever.
      Usage:

```
vpp <input_file><output_file>
```

11. DR 771
    - **Description: Translations** of decimal constants are not sized properly in concatenations.
      Example:

```
reg [19:0] a;
...
a = {10'd0, 10'd0};

SIGNAL a : std_logic_vector(19 DOWNTO 0)
REGISTER ;
...
a <= std_logic_vector' (0 & 0); -- wrong
translation
```

    - **Workaround:** Use binary constants instead of decimal.

12. DR 911
    - **Description:** Verilog identifiers differing only in case are not resolved.
      Example:

```
reg a;
reg A;

will both correspond to
signal a : std_logic;
```

    - **Workaround:** Make sure that your source does not contain such identifiers.

13. DR 953
    - **Description:** In some rare cases, verilog2vhdl will put entities in the output VHDL file in the wrong order, i.e. an entity will be instantiated before defined in the file.
    - **Workaround:** Manually fix the order of entities in the output VHDL file.

14. BZ 21: Problems while translating functions in the output file
    Description: Translator gives an error while translating function from Verilog to VHDL.

Workaround: Function MUST be declared before it is used in the input Verilog file. Please move the function to the top of the module declaration before translating.

Known issues with VHDL2verilog

1. DR 550

    - **Description:** Overloaded VHDL functions cannot be discerned during function mapping. The problem comes about due to the fact that the function mapping takes into account only the name of the function.

15. **DR 583**
    - **Description:** In a design that uses packages, the comment header from the package file frequently gets printed in the main output file instead. No known workaround exists.
16. **DR 621**
    - **Description:** Real Delay values are translated into exponential values.
    - **Workaround:** Manually change the delay values to reals again. For example: `#(1.900000e+00)` can be changed to `#1.9`.
17. **DR 720**
    - **Description:** In entities with generates parameterized by generics, the generates are elaborated only for the default value of the generic(s). Hence, if the entity is instantiated with a generic value other than the default, the translation will not be correct.
      NOTE: this problem applies ONLY to the entities with generates.
    - **Workaround:** Replicate the problematic entity so that there is a separate entity declaration for each used value of the generic.
18. **DR 724**
    - **Description:** If different default values for a generic are specified in the entity and component declarations, vhdl2v takes the one in the entity, whereas it should take the one in the component declaration.
    - **Workaround:** Make sure that the entity declaration uses the proper default value of the generic.
19. **DR 788**
    - **Description:** An aggregate constant is translated to a concatenation. Example:

```
TYPE WSizeTableType IS
    ARRAY(INTEGER RANGE 1 TO 13) OF INTEGER;
CONSTANT TcWSizeTable: WSizeTableType :=
                (7, 9, 11, 13, 15, 17, 19, 21, 23, 25,
27, 29, 31);
```

translates to:

```
   parameter TcWSizeTable = {7, 9, 11, 13, 15, 17, 19, 21,
23, 25, 27, 29, 31};
```

- **Workaround:** No easy workaround known. One has to manually change the Verilog output according to the design intent.

20. **DR 848**
    - **Description:** Based numbers and octal and hex bitstrings in OTHERS are not correctly translated
      Example:

```
   sig_array16 <= ( OTHERS => 16#c# );
```

      is translated to:

```
    {sig_array16[3], sig_array16[2], sig_array16[1],
sig_array16[0]} = {(3 - 0 + 1){}};
          // the literal disappears
```

    - **Workaround:** Use bit string literals in aggregates with OTHERS

21. **DR 765**
    - **Description:** No support for GUARD expressions. The GUARD expression, if present in the block declaration, has special meaning in the case of VHDL guarded conditional signal assignments.
    - **Workaround:** The conditional signal assignment is equivalent to a VHDL signal assignment if the GUARD expression evaluates to true. Consider:

```
ARCHITECTURE ...
BEGIN

test : BLOCK (reset != '1') IS
BEGIN

a <= GUARDED '1';
END BLOCK;
END ARCHITECTURE;
```

      is the same as

```
ARCHITECTURE ...
BEGIN

PROCESS(reset)
BEGIN
if (reset != '1') THEN
    a <= '1';
end if;
END PROCESS;

END ARCHITECTURE;
```

22. BZ 19: Issue with translation of state machines

Description: In the the .vhd file there are multiple declarations in which state name ‘idle’ appears.
For example

type firststate is (idle, notidle) etc
type anotherstate is (idle, notsoidle) etc.

The translator defines a parameter in Verilog for each. In some cases, the name for common parameters (idle) gets mixed up in the output Verilog file. When there are assigned in the output fileTranslator mixes up these definitions and picks up the last definition of idle.

Workaround: Modify the input file and rename the type definition. In the example, the new declaration could be-

type firststate is (idle, notidle) etc
type anotherstate is (**idle1**, notsoidle) etc.

Note than name ‘idle’ in anotherstate is changed to ‘idle1’.