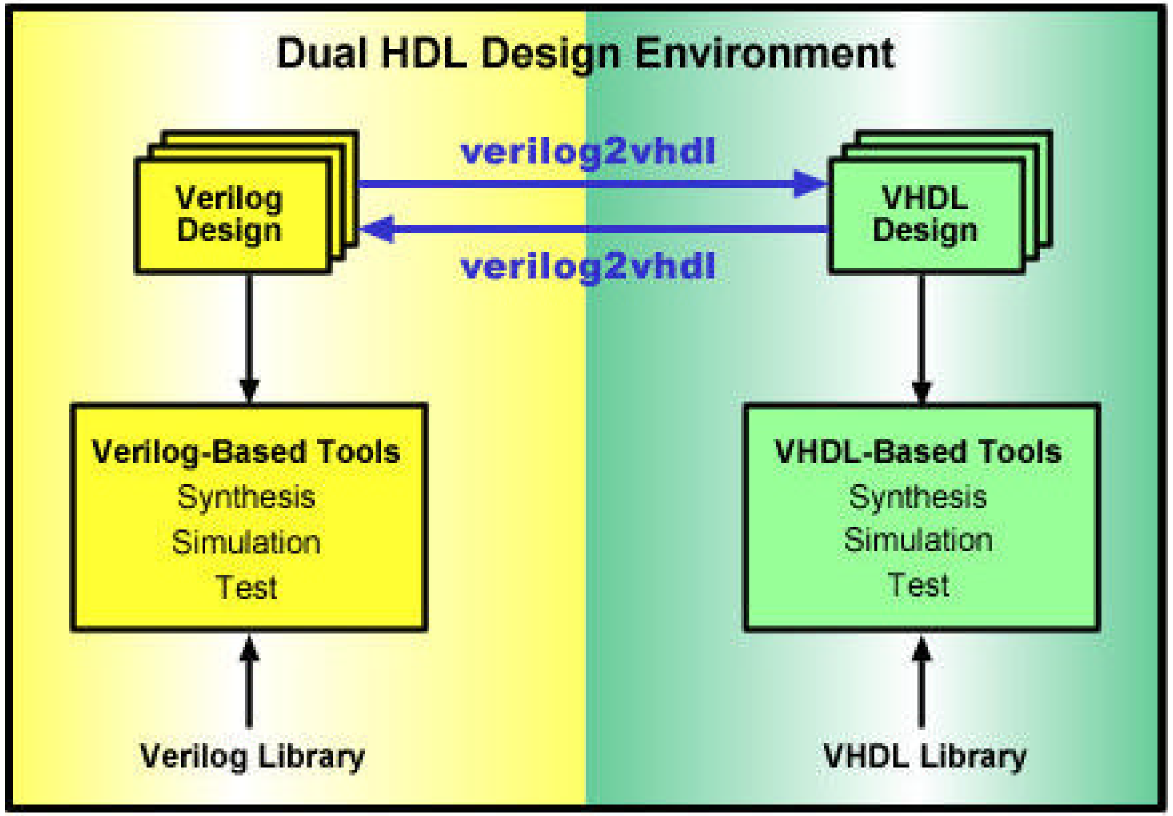
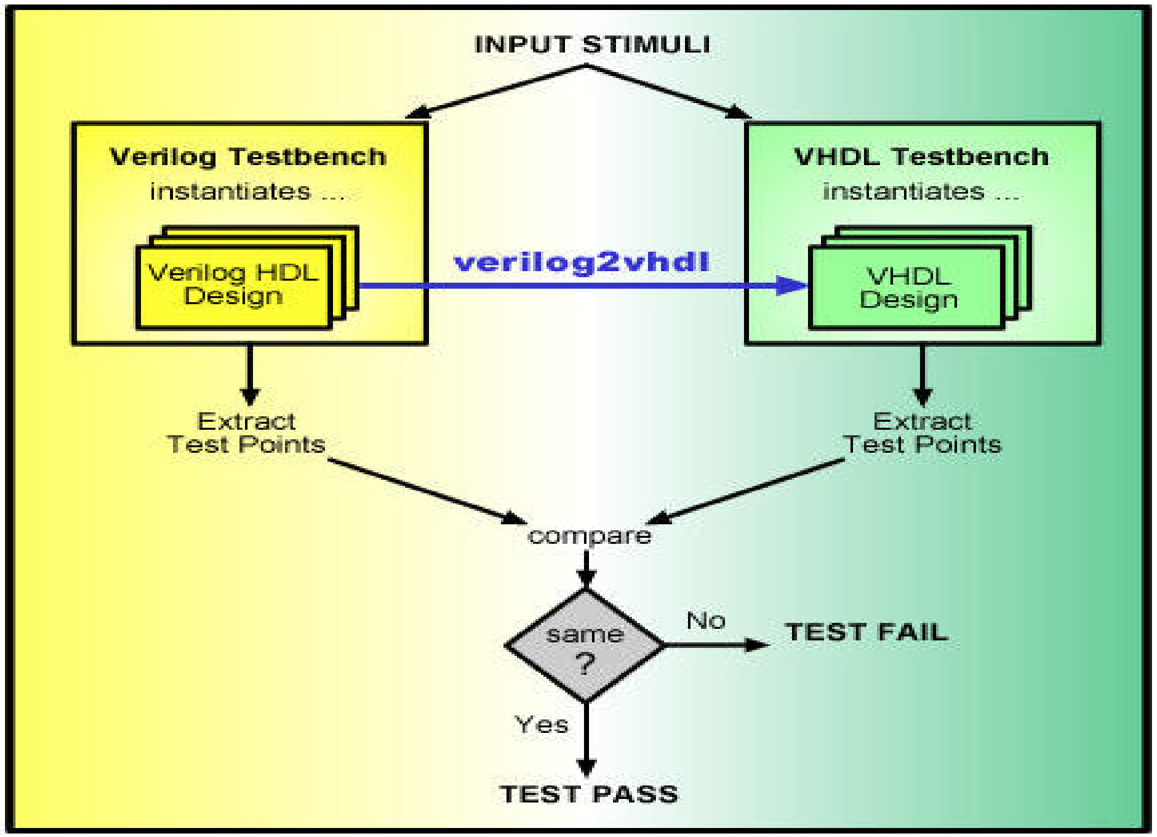
V2V



平均的に９5%程度のコードが自動変換

V2V



エラーなしに変換されたモジュールは、
シミュレーションでの確認が必要がないほど高品質

V2Vユーザ

- ✍ NTT未来研究所
- ✍ 他 40社 （日本）、xxも
- ✍ LSI Logic、Sisco、Intel、Lucent、ARC、ATI….
  100社程度

V2Vの変換時間

- ✍ 変換時間は当然、コンピュータ性能と設計サイズ（行数）による

- ✍ 10KラインでP500/Ultra10程度で数秒

- ✍ 実質的に瞬時と言える

V2Vの処理限界（回路規模）

- ✍ 100K行程度は普通に行われているようだが、問題になったことはない

- ✍ 内部的（ソフト設計）にはサイズは実質無制限

VHDL2Verilog変換制約-**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

VHDL2Verilog変換制約-**Architecture Declaration**

Supported:

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

Not supported:

- ✍ Design with an architecture and no entity
- ✍ Configuration specification

VHDL2Verilog変換制約-**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)

Not supported:

- ✍ Packages are translated *only* when used in a design !

VHDL2Verilog変換制約-**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

  * 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

VHDL2Verilog変換制約**-Generics**

Supported:

✍ Generics of base type as described in *DataTypes* with default expression

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

Not supported:

✍ Generics without default expression

VHDL2Verilog変換制約-**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
  - ABS
  - =, /=, <, >, <=, >=
  - SLL, SRL
  - ** (power) operator
- Qualified expressions
- Type conversions
- Function calls
- Aggregate primaries in an expression

Not supported:

- Allocator primaries

VHDL2Verilog変換制約-**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

VHDL2Verilog変換制約-**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

VHDL2Verilog変換制約-**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

Not supported:

VHDL2Verilog変換制約-**Concurrent Signal Assignments**

Supported:

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

Not supported:

- ✍ Multiple waveform elements in a concurrent signal assignment

VHDL2Verilog変換制約-**Component Instantiations**

Supported:

Not 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

VHDL2Verilog変換制約**-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

VHDL2Verilog変換制約-**Predefined Language Environment**

Supported:

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

Not supported:

- Other attributes
- TextIO

VHDL2verilogの変換概要1:

```
VHDL                                          Verilog
PACKAGE my_package is
CONSTANT Width : Integer := 16
END my_package;

USE ieee.my_package.ALL;                       module test(a,b);
ENTITY test IS                                 parameter Width = 16
PORT (a : IN std_logic;                                input a;
        b : OUT std_logic);                            output b;
END  ENTITY;
ARCHITECTURE behave OF test IS                     // other translated constructs
BEGIN
○ ○ ○ ○ ○ ○
END behave;                                     endmodule
```

VHDL2verilogの変換概要 2:

VHDL	Verilog
LIBRARY ieee;	module test(a,b,c);
USE ieee.std_logic_1164.ALL;	
ENTITY test IS	
PORT (a : IN std_logic;	input a;
b : OUT std_logic;	output b;
c : OUT std_logic);	output c;
END ENTITY;	
ARCHITECTURE behave OF test IS	//other translated construct;
BEGIN	wire b;
	reg c;
b <= '1'; -- concurrent assignment	assign b = 1;
END behave;	endmodule

VHDL2verilogの変換概要３:

VHDL

```
process_1 : process
CONSTANT tpd : std_logic := '1';
CONSTANT tpd1 : std_logic := '0';
begin
    wait on a, enable;
    if (enable = '0') then
         q <= '0';
    elsif a'event and a'last_value = '0' then
         q <= d;
         q <= '1' after 2 ns;
         q <= '0';
   end if;
end process process_1;
```

Verilog

```
always @(posedge a or negedge enable)
             begin : process_1
             parameter tpd = 'b 1;
             parameter tpd1 = 'b 0;

             if (enable == 'b 0)
                          q <= 'b 0;
             else
                      begin
                          q <= d;
                          q <= #2 'b 1;
                          q <= 'b 0;
                       end
             end
```

VHDL2verilogの変換概要 4:

VHDL	Verilog
VARIABLE status : boolean;	reg status;
-- status gets value	// status gets value
ASSERT status = FALSE REPORT	if (! (status == `false))
"Somemessage“ SEVERITYnote;	begin

```
VARIABLE status : boolean;
-- status gets value
ASSERT status = FALSE REPORT
  "Somemessage“ SEVERITYnote;
```

```
reg      status;
// status gets value
         if (! ( status == `false))
         begin
             $write("note:");
             $display("Some message");
             $display("Time: ", $time);
          end
```

VHDL2verilogの変換概要５:

VHDL :

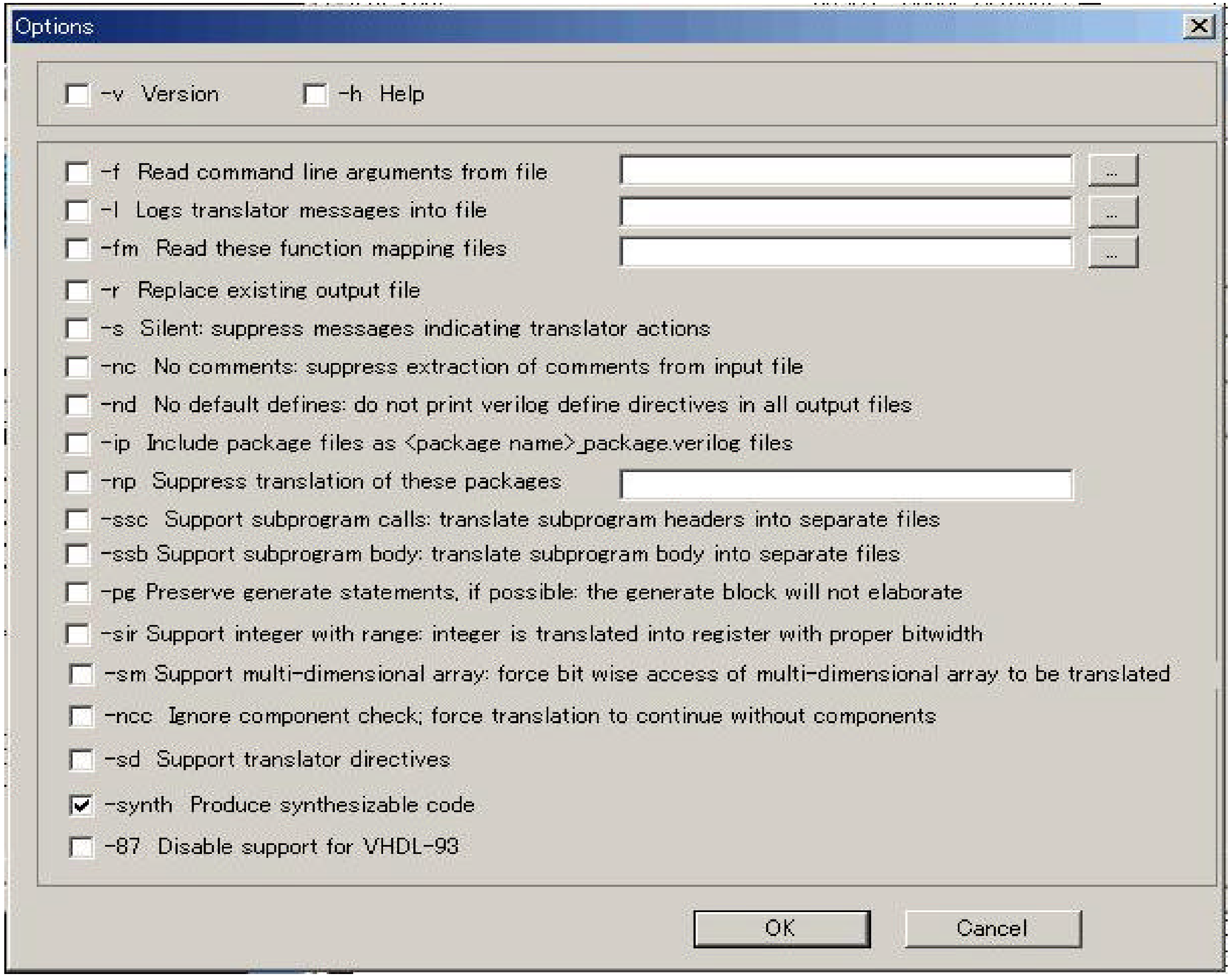
```
process (clk) begin
-- one of the following if expressions:
if rising_edge(clk) then
if (clk'event AND clk'last_value = '0') then
if (clk'event AND clk'last_value = '0' AND clk = '1') then
if (clk'event AND clk = '1' ) then
if (NOT clk'STABLE  AND clk'last_value = '0') then
if (NOT clk'STABLE  AND clk'last_value = '0' AND clk = '1') then
if (NOT clk'STABLE  AND clk = '1') then
q <= d;
end if;
end process;
```

Verilog:

```
always @(posedge clk)
begin
    q <= d;
end
```

＊ for Verilog negedge expressions (the 'clk' value being '0' instead of '1')

VHDL2verilog Options:



Options

- [ ] -v Version
- [ ] -h Help

- [ ] -f Read command line arguments from file
- [ ] -l Logs translator messages into file
- [ ] -fm Read these function mapping files
- [ ] -r Replace existing output file
- [ ] -s Silent: suppress messages indicating translator actions
- [ ] -nc No comments: suppress extraction of comments from input file
- [ ] -nd No default defines: do not print verilog define directives in all output files
- [ ] -ip Include package files as <package name>_package.verilog files
- [ ] -np Suppress translation of these packages
- [ ] -ssc Support subprogram calls: translate subprogram headers into separate files
- [ ] -ssb Support subprogram body: translate subprogram body into separate files
- [ ] -pg Preserve generate statements, if possible: the generate block will not elaborate
- [ ] -sir Support integer with range: integer is translated into register with proper bitwidth
- [ ] -sm Support multi-dimensional array: force bit wise access of multi-dimensional array to be translated
- [ ] -ncc Ignore component check; force translation to continue without components
- [ ] -sd Support translator directives
- [✓] -synth Produce synthesizable code
- [ ] -87 Disable support for VHDL-93

verilog2vhdl変換制約-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

verilog2vhdl変換制約-Expressions

Supported: Not 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 ^~

verilog2vhdl変換制約-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

verilog2vhdl変換制約-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*)

verilog2vhdl変換制約-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)

verilog2vhdl変換制約-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)

verilog2vhdl変換制約-Hierarchical Structures

Supported:

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

Not supported:

- ✍ hierarchical names

verilog2vhdl変換制約-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

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

Not supported:

- ✍ all other system tasks
- ✍ all other system functions

verilog2vhdl変換制約-Compiler Directives

Supported:

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

Not supported:

- all others

verilog2vhdl設計スタイル**-Not supported:**

Unsupported constructs:

✍ 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

verilog2vhdl設計スタイル-Supported:

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 hierachical 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

verilog2vhdl設計スタイル-Supported:

Supported constructs:

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)*.

verilog2vhdl設計スタイル-Supported:

Supported constructs:

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.

verilog2vhdl設計スタイル-Supported:

Supported constructs:

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.
    avoid using $monitor{on,off} system tasks.

verilog2vhdl の変換概要１

Verilog

```
..........

// wire_declaration;
// gate_instantiation;
...........
```

VHDL

```
architecture arch_name of entity_name  is
begin

-- concurrent signal assignment;

end architecture;
```

Example:

```
.............
`timescale 1ns/1ns
wire a,b,c;
wire d,e,f;
wire g,h,i;
nand (a,b,c);
nor  #3 (d,b,c,e,f);
not  #2 (g,b);
not (h,i,c);

..........
```

```
............

signal a,b,c : std_logic;
signal d,e,f : std_logic;
signal g,h,i : std_logic;
a <= b nand c;
d <= (((b nor c) nor e) nor f) after 3 ns;
g <= b after 2 ns;

h <= c;
i <= c;
...........
```

verilog2vhdl の変換概要 2

Verilog	VHDL
SYSTEM FUNCTIONS	
$time	NOW / <timeunit>
$fopen("filename.ext")	FILE F1: text open WRITE_MODE is "filename.ext"
SYSTEM TASKS	
$display	V2V_display
$fdisplay	V2V_display
$strobe	V2V_display
$fstrobe	V2V_display
$write	V2V_write
$fwrite	V2V_write

verilog2vhdl の変換概要３

Verilog

```
integer file1;
integer filechan;
...
file1 = $fopen("latch.list");
filechan = file1 | 1;
$fdisplay(filechan, , $time,
    "Change in qOut=%b with data=%b",
      qOut, data);
```

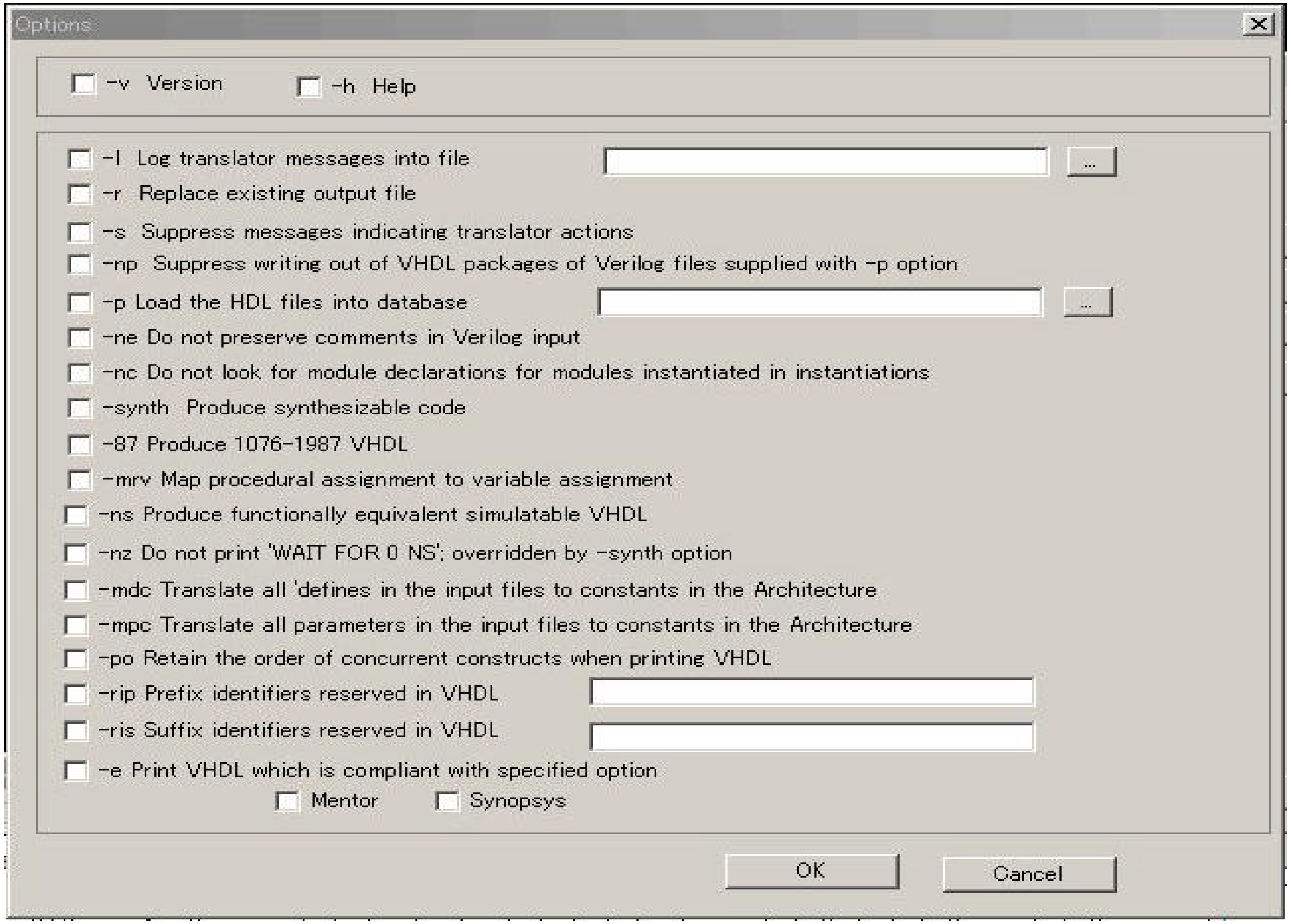
VHDL

```
SIGNAL file1 : integer;
SIGNAL filechan : integer;
SIGNAL std_io : integer := 1;
FILE F2 : text open WRITE_MODE is "latch.list";
FILE F1 : text open WRITE_MODE is "/dev/tty";
CONSTANT v2v_message0 : string (1 TO 1) := " ";
CONSTANT v2v_message1 : string (1 TO 15) :=
                    "Change in qOut=";
CONSTANT v2v_message2 : string (1 TO 11) :=
                    " with data=";
PROCEDURE V2V_display
                    (SIGNAL filechan : in integer;
                     message1 : IN string := "";
                    signal1 : IN bit_vector := "";
                     message2 : IN string := "";
                    signal2 : IN bit_vector := "";
                    message3 : IN string := "";
                    signal3 : IN bit_vector := "";
                    message4 : IN string := "";
                    signal4 : IN bit_vector := "";
                    message5 : IN string := "");
...
file1 <= 2;
filechan <= file1 OR 1;
V2V_display(filechan, v2v_message0, OPEN,
                    INTEGER'IMAGE(NOW/1 NS), OPEN,
                    v2v_message1, to_bitvector(qut),
                    v2v_message2, to_bitvector(data));data));
```

verilog2vhdlの予約語:

abs	false	nand	rol
access	file	new	ror
after	function	next	select
alias	generate	nor	severity
all	generic	not	signaland
architecture	guarded	of	sla
array	impure	on	sll
assert	in	open	sra
attribute	inertial	others	srl
block	inout	out	subtype
body	is	package	then
buffer	label	port	to
bus	library	postponed	transport
component	linkage	procedure	true
configuration	literal	process	type
constant	loop	pure	unaffected
disconnect	map	range	units
downto	mod	record	until
elsif		register	use
entity		reject	variable
exit		rem	wait
		report	when
		return	with
			xnor
			xor

verilog2vhdl Options:



Options

☐ -v Version ☐ -h Help

☐ -l Log translator messages into file
☐ -r Replace existing output file
☐ -s Suppress messages indicating translator actions
☐ -np Suppress writing out of VHDL packages of Verilog files supplied with -p option
☐ -p Load the HDL files into database
☐ -ne Do not preserve comments in Verilog input
☐ -nc Do not look for module declarations for modules instantiated in instantiations
☐ -synth Produce synthesizable code
☐ -87 Produce 1076-1987 VHDL
☐ -mrv Map procedural assignment to variable assignment
☐ -ns Produce functionally equivalent simulatable VHDL
☐ -nz Do not print 'WAIT FOR 0 NS'; overridden by -synth option
☐ -mdc Translate all 'defines in the input files to constants in the Architecture
☐ -mpc Translate all parameters in the input files to constants in the Architecture
☐ -po Retain the order of concurrent constructs when printing VHDL
☐ -rip Prefix identifiers reserved in VHDL
☐ -ris Suffix identifiers reserved in VHDL
☐ -e Print VHDL which is compliant with specified option
☐ Mentor ☐ Synopsys