GenSys®
Known Problems and Solutions

Version O-2018.06, June 2018

SYNOPSYS®

Copyright Notice and Proprietary Information

©2018 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/Company/Pages/Trademarks.aspx.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

1

GenSys Known Problems and Solutions

This document describes the following GenSys Known Problems and Solutions for:

- Unsupported RTL Language Constructs
- Various Other GenSys Features

Unsupported RTL Language Constructs

This section describes known problems related to unsupported RTL language constructs that are expected to be fixed in future releases.

1. Issues with Verilog handling in import/export flow
    - GenSys does not support user functions and Verilog math functions apart from $clog2 and $bits in `define.
    - Local define/undef is not supported
    - Generate blocks with local signal/wire cannot be ungrouped and hence cannot be restructured/modified in GenSys
    - Support for defparam construct has issues. Please see Issues related to defparam for details
    - Gate primitives, such as buf and bufif1 are not supported.
    - Constant functions are not supported.
    - Support for unsynthesizable construct has issues. Please see Issues with preserving Unsynthesizable constructs for details.
    - Delay value is not supported in RTL import.
    - Analog Verilog is not supported.
    - Input RTL formatting is not preserved in the generated RTL.
    - Compiler directives, except in the packaging flow, are not supported.
    - Escaped identifiers are not supported.
    - The generate statements get unrolled in the output.
    - The supply0 and supply1 net types are not preserved in the generated RTL.
    - GenSys does not properly handle cases in which sizes are specified for ports in a module port list. Following is the example of such case:

      ```
      module abc ( in1[1:0] , out1);
      ```
    - User cannot specify a parameter in the declarative region with same name as top-level.
    - Parameterized multidimensional ports and signals.
    - Partial access of multidimensional ports and signals.
    - Format of v2k style ports/parameters are not preserved.

- **Issues related to defparam:**
  - **defparam statements for different paths of an instance is not supported**: For example, the leaf_inst instance in the mid module can be reached via two paths due to two instances of mid inside the top module. Parameter override using defparam for instances, such as leaf_inst, is not supported.

```
module leaf(D);
parameter leaf_param = 3;
input [leaf_param  : 0] D ;

endmodule

module mid ( D) ;
input [7 : 0] D ;
leaf leaf_inst(D);
endmodule

module top;
wire [7:0] D ;
mid inst_1( D) ;
mid inst_2( D) ;
defparam top.inst_1.leaf_inst.leaf_param = 7;
defparam top.inst_2.leaf_inst.leaf_param = 8;
endmodule
```

  - Parameter override with defparam for instances in generate blocks and array of instances are not supported: The defparam statements used inside the generate block as in module mid is not supported.

```
module leaf(D);
parameter leaf_param = 3;
input [leaf_param  : 0] D ;
endmodule

module mid ( D) ;
input [7 : 0] D ;
leaf leaf_inst(D);

genvar i;
generate
for (i = 0; i < 3; i = i + 1) begin : genblk
leaf #(3)leaf_inst();
defparam genblk[i].leaf_inst.leaf_param = i ;
end
endgenerate

endmodule
```

2. Issues with the SystemVerilog support

   Following are the issues:

- GenSys does not support the following:
  - Array of interface not supported.
  - Parameter cannot be updated for Parameterized SV interfaces. Also, GenSys may not preserve parameter of interfaces.
  - Generic SV interfaces, if the same module is instantiated multiple times with different interface modports
  - Shortreal and real types
  - Extern modules, as shown in the following example:

```
module extern4(in1, in2, out1, out2);
   input [5:0] in1, in2;
   output [7:0] out1;
   output [14:0] out2;
   extern module leaf(in1, in2, out1, out2);
   leaf I1(.*);
   module leaf(.*);
      input [5:0] in1, in2;
      output [7:0] out1;
      output [14:0] out2;
      bot1 #(5, 7) I1(in1, in2, out1);
      bot2 #(5, 14) I2(in1, in2, out2);
      extern module bot1 #(parameter p1 = 8,
                     parameter p2 = 19)
                      (input signed [p1:0] in1, input
                       signed [p1:0] in2,output signed
                       [p2:0] out1);
      extern module bot2 #(parameter p1 = 8,
                     parameter p2 = 19)
                      (input signed [p1:0] in1, input
                       signed [p1:0] in2,
                       output signed [p2:0] out1);
      module bot1(.*);
         assign out1 = in1 & in2;
      endmodule
      module bot2(.*);
         assign out1 = in1 * in2;
      endmodule
   endmodule
endmodule
```

  - Nested modules, as shown in the following example:

```
  module instan2 #(parameter p1 = 6, parameter p2 = 15)
(input signed [p1-1 : 0] in1,                          input
signed [p1-1 : 0] in2,                           output signed
[p2-1 : 0] out1);    wire [p2-1 :0] w1;    nested_mod1
I1();    module nested_mod1;      nested_mod2 #(p1, p2) I1(in1,
in2, w1);      nested_mod2 #(p1, p2) I2(w1, w1,
out1);      module nested_mod2 #(parameter p1 =
6,                         parameter p2 = 15)
(input signed [p1-1 : 0] in1,                          input
signed [p1-1 : 0] in2,                           output signed
[p2-1 : 0] out1);      out1 = in1 *
in2;      endmodule    endmodule  endmodule
```

- Glue logic in interface, as shown in the following example:

```
   interface intf #(parameter BUS_WIDTH = 8,DATA       = 2)(input
logic [0:BUS_WIDTH-1] data[DATA-1:0]);   logic [BUS_WIDTH-1:0]
buffer[DATA-1:0];   logic [BUS_WIDTH-1:0] sum;   int
count;   always @*     for(count = 0; count < DATA;
count++)      buffer[count] = data[count];   initial     sum =
'0;   endinterface   module instance4 #(parameter WIDTH1 =
2,WIDTH2 =        32)(input clk, input logic
[0:WIDTH2-1]       data[WIDTH1-1:0], output [0:WIDTH2-1]
sum);   int count;   intf #(WIDTH2,WIDTH1)inst(data);   always
@(posedge clk)   begin     for(count = 0;count < WIDTH1;
count++)      inst.sum = inst.sum + inst.data[count];   end
assign sum = inst.sum;   endmodule
```

- Type parameters, as shown in the following example:

```
   extern module bot1(a, b, c);   extern module bot2 #(parameter
p1 = 8, parameter type                     TP = logic
[17:0])                        (input [p1:0] a, input [p1:0]
b,                         output TP c);   module extern2(input
[5:0] in1, input [5:0] in2,                 output [9:0] out1,
output [14:0] out2);   bot1 I1(in1, in2, out1);   bot2 #(5, logic
[14:0]) I2(in1, in2, out2);   endmodule   module
bot1(.*);   input [5:0] a, b;   output [9:0] c;   assign c = a +
b;   endmodule   module bot2(.*);   assign c = a * b;   endmodule
```

- Interface within interface (nested SV interfaces)
- Parameterized multidimensional ports and signals, as shown in the following example:

```
   `timescale 1ns / 10ps   /**   * Delay an signal by a fixed
number of cycles.    **/   module DELAY #(parameter WIDTH=32,
parameter   CYCLES=8)   (   input         CLK,   input
RESET_N,   input  [WIDTH-1:0] IN,   output [WIDTH-1:0]
OUT   );   /****************** Declare internal variables   reg
[WIDTH-1:0]    in_delayed[CYCLES-1:0];   /********************
Update internal state     always @ (posedge CLK or negedge
RESET_N) begin      integer i;       if (!RESET_N)         for
(i=0; i<CYCLES; i=i+1)          in_delayed[i] <= '0;
else         begin          in_delayed[0] <= IN;          for
(i=0; i<CYCLES-1; i=i+1)            in_delayed[i+1] <=
in_delayed[i];         end     end   /**********************
Assign Output    assign OUT = in_delayed[CYCLES-1];endmodule
```

- Partial access of multidimensional signals, as shown in the following example:

  ```
     `timescale 1ns/1ns   package p;   typedef logic
  l[3:0];   typedef bit [7:0] b;   typedef byte
  bt[1:0][1:0];   typedef reg [3:0] r_st
  [1:0];   endpackage   module decl4(   input logic [7:0] in1,
  logic [3:0] in2,     reg clk,   output logic [7:0] out1,
  logic [3:0] out2   ;   import p::*;   l lp;   b bp;   bt
  btp;   r_st rp;   assign out2 = lp[0] | (bp & rp[1]);   assign
  out1 =  {btp[0][0],lp[2],bp[2:0],rp[1][3:1]};   always@( posedge
  clk)   begin   lp[0] = '1;   lp[2] = '0;   bp = in1;   btp[0][0]
  = in1 ^ in2;   rp[1] = {in2,in1};   rp[0] = in1 |
  in2;   endendmodule
  ```

- Configurability around interface

○ User cannot edit SystemVerilog types in GenSys.

○ Virtual SV interfaces not supported. For example:

```
module testbench(intf.tb tb_if);
virtual interface intf.tb local_if; // virtual interface.
 ....
task read(virtual interface intf.tb l_if) // As argument to task
....
 initial
 begin
 Local_if = tb_if; // initializing virtual interface.
 Local_if.cb.read <= 1; //writing to synchronous signal read
 read(Local_if); // passing interface to task.
 end
endmodule
```

○ Task and function inside SV interfaces not supported. For example:

```
interface intf_AB (input bit clk);
logic        ack;
logic        ready;
logic        send;
logic [31:0] data;
   task send_data (input logic send_signal,
                   input logic [31:0] data_bus);
   ... // actual task definition
   endtask

   ... // rest of the interface definition here
endinterface

module moduleA (interface xyz);

   ...
   xyz.send_data(send);
   ...
```

```
endmodule
```

3. Issues with the VHDL support
   - Non-supported RTL constructs for RTL import are compiled. GenSys reports an error in some of the following cases:
     - VHDL attributes, especially attributes that depend on an instance and other design properties which could change, are not handled properly.
     - Buffer and linkage ports in VHDL are not handled.
     - VHDL alias is not handled properly. Therefore, the exported RTL for case having alias may contain issues.
     - GenSys naming conflict occurs when a VHDL entity has a port named P and an internal signal with name P_int.
     - Handling of extended identifier is currently not supported.
   - In VHDL, GenSys does not preserve the unsynthesizable constructs that are given inside a pragma.

     So when you set the -preserve_pragmas argument of the set_rtlimport_option Tcl command to TRUE, Gensys may report synthesis error/warning.
   - Parameterized multidiemensional complex ports and signals not supported for VHDL.
   - Partial parameterized connection not preserved for VHDL.
   - Where multiple architectures exist for an entity, only the elaborated architecture selections is preserved. GenSys does not maintain multiple architectures for a given entity.
   - The VHDL operators that are currently supported (mod, rem, **, and abs) work only on positive integer data. These operators may not work correctly for real or negative numbers.
   - Currently, support for VHDL operators, mod, rem, **, and abs, is not provided if they are imported from IP-XACT XML files.
   - VHDL parameter/constant names are case-insensitive in VHDL, but case-sensitive in GenSys.
   - Operator precedence is not considered while converting to/from Tcl expression to VHDL expression. Operator precedence is assumed to be same in Tcl and VHDL.
   - If due to a restructuring operation, the existing VHDL configuration becomes invalid, you will have to manually edit that VHDL configuration.
   - For VHDL RTL, net name is not preserved when any of the source/destination has a complex type port, that is, other than std_logic.

4. Miscellaneous Issues
   - Presence of local signal of the same name in different glue/generate block is not supported.
   - Presence of a local signal inside a glue/generate block sharing same name with a module level port or signal is not supported
   - During RTL import, delta delay value is not imported.
   - GenSys does not preserve nested pragma and complex pragma.
   - Handling of RTL constructs in same line number has issues.
   - Modules spanned over multiple files not supported.
   - Net names are preserved only for complete connections. For example, if certain bits, say n1[1:o], of the vector net n1[3:0] connect a source and destination, n1[1:o] is not preserved in the generated netlist. However, if the complete vector net n1[3:0] connects the source and destination, net n1[3:0] is preserved.
   - Not all DW libraries are supported by default. Some advanced DW libraries are supported under following option

     ```
     set_option  dw_options  dont_use_tpts
     ```

Various Other GenSys Features

This section describes various known problems in this release that are expected to be fixed in future releases.

1. Issues with adding/deleting rows in GenSys tables.

   Addition/deletion at the *N*th position in a table is memory-intensive. Also, if *N* is not equal to the last row of the table, then this operation is more costly (time-consuming -> O(n)). Therefore, it is recommended to do this judiciously. This issue is expected to be fixed in future releases of GenSys.

2. Issues with group/ungroup

   Following are the known issues with group/ungroup feature:
   - New instances can be added to an existing group only if the existing group is the only instance of its master. However, in this case, properties (like existing port name) of the original subsystem may change.
   - GenSys does not support interface open and interface loop-back connections while ungrouping.

- ❍ During ungroup, handling of open connections is available only for adhoc connections.
- ❍ GenSys does not allow ungroup or hierarchy dissolution of those generate blocks (pseudo hierarchy) that have a local wire/signal (declared inside the generate block) in the RTL code.

3. Issues with RTL generation

   Following is the known issue with RTL generation:

   - ❍ In case of overridden connections, redundant ports may get generated while creating hierarchy.

4. Issues with delta delay

   Following are the known issues with delta delay:

   - ❍ Delta delay is currently supported for adhoc and interface connections only.
   - ❍ For Verilog modules, after RTL generation, delta delay values are dumped in nano seconds only. No other unit is supported.

5. Issues with parameters

   Following are the known issues with parameters:

   - ❍ It is recommended to use Tcl commands for adding parameters of complex types using the add_type Tcl command since the GUI tables are not very intuitive.
   - ❍ Data entered in the Type table is not reflected in the Value table. The tables should be refreshed to view the updated data in the Value table.
   - ❍ Parameters are not resolved if options -prefix and -postfix are used with the Tcl command, export_interface.
   - ❍ Parameters are not preserved during restructuring (move or group).
   - ❍ Parameters are not preserved during export interface.

6. Issues with the Connections table

   Following are the known issues with the Connections table:

   - ❍ Parametric expression in connection LSB or MSB is not supported.
   - ❍ Logical and net connections are deprecated features in GenSys and we do not plan to support it.
   - ❍ GUI support is not available for exported interface connections. Tcl commands should be used.

- Connection attributes created using Tcl commands are not shown in GUI. These attributes can be handled by using Tcl commands.
- Appropriate errors/warnings are not flagged when incorrect tieoff connection are created for top design input ports.
- Appropriate warning messages are not flagged if open or temporary open connections are specified for a bit slice of a port.

7. Issues with the Interface Connections

   Following are the known issues with interface connections:

   - The reverse value of the -bits argument of the add_connection Tcl command is not supported for interface connections. This means that the while connecting interfaces, source and destination port bits cannot be connected in a reverse order. Hence, the first bit of the source port of an interface cannot be connected with the last bit of the destination port of an interface, and so on.
   - The delete_hier_connection command does not work for interface connections.

8. Issues with the Connection Builder dialog

   Following are the known issues with Connection Builder dialog:

   - For top-level design interface, open or temporary open connections are not supported.
   - Drop-down list is not shown for top-level design interface in case of multiple fan-outs.
   - After elaboration, the connectivity status of a connected top-level design interface does not show the exact connection. In such cases, the status is shown as Interface. However, a proper connectivity status with destination end details is shown for individual IPs.
   - If the user connects two interfaces that have incompatible directions, GenSys flags a warning. However, at the top-level, the status of the two interfaces is shown as connected. If the user disconnects such interfaces, GenSys continues to displays the status as connected. The user need to elaborate or refresh the view to display the correct connection status.
   - Reverse bit connections are not supported in the Connection Builder dialog.

9. Issues with RTL health check feature

   Following are the known issues with RTL health check feature:

   - In the RTL_Health_Check table, design interface is not shown in the Destination Interface column.
   - During RTL health check, vectorization is not done for multifanout scenario of design port.

10. Issues with IP-XACT import and export

    Following are the known issues with IP-XACT import and export:

    - While dumping IP-XACT design/sub-system in TCL, GenSys does not dump component view spirit extension tables.
    - Reverse bit connections are not supported for IP-XACT export and import.

11. Issues with support for IP-XACT 1.2

    Following are the known issues with support for IP-XACT 1.2:

    - GenSys imports/parses all the elements of IP-XACT XMLs. However, some of the elements are not translated natively into the GenSys database. The following table lists such elements/sub-elements:

Element	**Sub Elements**
Component	RemapStates
	ComponentGenerators (related to LGI)
	Configurators (related to TGI)
	WhiteBoxElements (for verification)
	CPUs (Some components may have internal CPUs)
	ComponentConstraintSets
	memoryMaps ->bank->bankBase->addressBlock-> addressBlockExtensions-> register
	addressSpace-> localMemoryMap-> bank->bankBase-> addressBlock->addressBlockExtensions-> register
	memoryMaps ->memoryRemap
	addressSpace-> executableImage
	addressSpace-> localMemoryMap-> subspaceMap
	OtherClockDrivers
	Parameter
	The component->model->parameter item is translated into com parameter
	FileSets

Element	Sub Elements
Component-> busInterface	System (group)
	Connection
	Index
	BitSteering
	Configurators
Registers	Dependency
	Parameter
	Field->parameter
Parameters	CrossRef
	Append
	RangeType
	ChoiceStyle
	Direction
	ConfigGroups
Component-> model	views
	clockDriver
	SingleShotDriver
	Export
	SignalConstraintSets
BusDefinition	DirectConnection
	MaxMasters
	MaxSlaves
	Signals-> requiresDriver
	Signals-> busDefSignalConstraintSets
Register	Mask value

- GenSys does not provide case-sensitive search for IP-XACT XML files.
- Some of the special characters of IP-XACT VLNV are not valid and are therefore replaced with '_' while doing COM to IP-XACT translation.
- IP-XACT has some extra rules for the comparison of Version and Name, which are currently not supported.

- BusDefinition->parameters are translated to COM attributes but live formula support is required to access the attributes value. When an interfaceLib is instantiated in a component, component->busInterfaces-> busInterfaceParameters is to be handled accordingly.

12. Issues while replacing master of an IP

    Following are the known issues faced while replacing master of an IP:

    - While replacing IPs, GenSys loads all the designs given in the Library Browser sequentially and reconfigures them. In this process, GenSys does not release memory after closing a design. This can lead to increase in memory consumption. Therefore, it is recommended to load only one design at a time in the Library Browser, replace IP for that design, close and reopen GenSys to release the memory and replace IP for the next design.
    - If, while updating master of an IP, GenSys marks any connection as INVALID, such connections are highlighted in RED only when the design is reloaded in GenSys.

13. Issues with Tcl commands

    Following are the known issues with Tcl commands:

    - The Tcl command, closeall, does not release memory consumed by GenSys designs/ components/interfaces.
    - The delete_connection Tcl command does not support partial deletion of interface connection unless it is an exact match. For example, the delete_connection Tcl command is not supported in the following case:

    ```
    add_connection -instance i1 -interface intf1 -splice{L1[1:2],
    L2[2:3]} -instance i2 -interface intf2 -splice{L1[1:2],L2[2:3]}
    delete_connection -instance i1 -interface intf1 -splice{L1[1:2]}
    -instance i2 -interface intf2 -splice{L1[1:2]}
    ```

    - GenSys does not support net names (specified by the -net_name/-net_prefix arguments of add_connection/add_port/set_interface_port Tcl commands) that are specified with LSB and MSB values. For example, if the user specifies net names as X[16:13] and X[12:9], GenSys will not group them as X[16:9]. Instead, it would treat them as different net names.
    - GenSys ignores the -net_name/-net_prefix arguments of the add_connection/ add_port/set_interface_port Tcl commands for the connections other than tieoff connection (default/forced/temporary).

14. Issues with user-defined Tcl commands

    Following is a known issue with user-defined Tcl commands:

    - The Tcl dump may not be correct in the following cases:

  - If the data in extension tables is incorrect (for example, there are duplicate values in the KEY column)
  - If there are more than one KEY columns in the table and there is dependency among them.
  - If KEY column values are autofilled and later changed manually.

15. Issues with special characters
    - It is recommended not to use special characters while specifying naming conventions. This is because GenSys does not support special characters in naming convention for some cases.
    - GenSys does not support escape characters within curly braces in Tcl expressions. For example, GenSys does not support the following Tcl expression:

      ```
      {=\[p ADR_LEN\]-1} example - set_interface_port -name addr
      -direction IN -msb {=\[p ADR_LEN\]-1} -lsb 0
      ```
      To support the above Tcl expression, re-write this expression in the following manner:

      ```
      {=[p ADR_LEN]-1} example - set_interface_port -name addr  -direction
      IN -msb {=[p ADR_LEN]-1} -lsb 0
      ```
      Note:
      Escape characters are not required within curly braces in Tcl expressions.
16. Issues with the VectorizeElaboratedConnections API
    - Connection description in the VHDL RTL disappears.
    - Impact of the set_rtl_option -uniquify_defval_rtldump TRUE command gets lost.
17. Issues with schematic
    - User should run gload Tcl command before printing a design.
    - The incr_sch_show -design <*des-name*> or mod_sch_show -design <*des-name*> work only for a top-level design. To print a sub-system, use the gload command first.

      Note:
      The select_design and set_active_object Tcl commands do not provide a workaround in this case.
    - Batch schematic does not use the .gensysrc schematic values. Therefore, it is recommended to specify the following Tcl command:

      ```
      sch_set_property -bgcolor WHITE
      ```
    - Color settings in Incremental Schematic do not work reliably, and should be avoided.
18. Issues with save/saveall Tcl commands

- GenSys may report the same INFO message (informing user about location where object is being saved) for an object multiple times in some cases. For other cases, GenSys might not report any such INFO message.

  This issue is expected to be fixed in a future release.

- In some particular cases, GenSys saves an object multiple times.

  For example, consider a top-level design, des, that contains two instances, comp0 (of the comp component) and inf0 (of the inf interface). Also consider that comp contains an instance, inf1, of the inf interface. Now when user saves the object as IP-XACT in this case, GenSys saves the inf interface three times, instead of saving it once.

  This issue is expected to be fixed in a future release.

- Consider a case in which a group is created, and its master (sub-design) is saved when the user saves it. Next time, when the user performs an ungroup operation on that group, the created sub-design still exists. If the user again creates a group and gives it the same name, the existing sub-design gets overwritten.

  This issue is expected to be fixed in a future release.

19. Issues with autoconnect support

    - Connect order, terminal attributes, and connection attributes are not saved in IP-XACT.
    - Connection attributes are not saved in TCL/MRS and are lost.
    - Hierarchical export is not recommended for autoconnect. It is recommended to use bottom-up export at every hierarchy in the following manner:
      - First apply autoconnect by specifying -export yes from IPs to the containing subsystem and save the modified subsystem definition.
      - Then apply the export on autoconnect from subsystem instances to the parent design/subsystem and repeat this step as you go up in the hierarchy.

20. Issues with auto-connect support for glue

    Auto-connection is not done in the following cases:

    - If the port of an instance that is to be driven by glue output port is connected to multiple ports, auto-connect is not done.
    - If the bit of a pin of an instance to which an output pin of a glue instance connects has multiple connections, auto-connect is not done.

21. Issues with the TGI Support

    - File-based TGI is not supported.

- ❍ Within the same session, user can not reconnect. User needs to initiate a new session in such cases.
- ❍ Some APIs related to abstractor and generator are not implemented.

22. Issues with SCR checks
    - ❍ Some SCR checks related to abstractor and generator are not implemented.
    - ❍ On demand run of SCR checks (which are linked to a library and VLNV) is not possible. Such checks are run during GenSys library binding.
23. Issues with IP packaging
    - ❍ While doing IP packaging of a VHDL RTL file that contains STX errors in architecture, user is required to specify a top-level entity by using the -top <*entity-name*> command along with the load_rtl Tcl command.
24. Issues with glue logic restructuring
    - ❍ Individual glue solidification is not available for VHDL.
    - ❍ The glue/generate instances solidified by using the group_glue Tcl command can only be moved and grouped. No other operation, such as rerouting and ungrouping is allowed on them.
    - ❍ GenSys cannot detect in a user-defined glue logic if a reg declaration is required for an output port or not. In case if it is required, user must specify it in the declarative region.
25. Issues with complex types in the VHDL and SystemVerilog

    Following are the known issues with complex types:
    - ❍ Tieoff connections are only supported for the type derived from Std_logic bit.

      For example, consider that you define the following type:

      ```
      type TLIGHT is {RED, GREEN, BLUE}
      ```
      Now, if you make a signal of type TLIGHT and assign it the value RED, GenSys will not support it.
    - ❍ You cannot create complex types in GenSys. You can only import them through RTL import. In addition, you cannot modify complex types and the ports of complex types in GenSys.
    - ❍ GenSys does not support creation of a connection between ports that require a conversion function to map each other. Only explicit type conversion are supported, such as SIGNED, UNSIGNED, and STD_LOGIC_VECTOR.

- If an architecture has complex types, the hierarchy movement of any instance connected with the signal of these types result in syntactically incorrect RTL generation.

  As the signal of these types come on the boundary, the type definition will not be available for such signals.
- In the generated RTL, parameters are not preserved in the ports that are of parameterized complex type.
- Only adhoc and interface connections are allowed for complex types. GenSys reports an error for other connections, such as logical, and splice.

26. Issues with the preserve configuration (ifdef) feature of RTL import

    Following are the known issues with the preserve configuration flow:

    - The ifdef support is not available for precompiled flows.
    - ifdef on `define is not supported and therefore, it is not preserved.

      Consider the following example:

      ```
      `ifdef X   `define A 1   `endif
      ```
      In the above example, the `define statement is preserved but ifdef around this is not preserved.
    - ifdef on `undef is not supported and therefore, it is not preserved. For example, the following statement is not preserved:

      ```
      `ifdef X    `undef A 1    `endif
      ```
    - ifdef on `timescale/timeunit/timeprecision is not supported and therefore, it is not preserved
    - ifdef on a module/`define/assertions/initial blocks/un-synthesizable constructs and SystemVerilog interfaces is not supported.

      Consider the following example:

      ```
      `ifdef X    module mod(a,b,c)    ...    `endif
      ```
      In the above example, ifdef on the module is not preserved and the module is dumped as the following with ifdef:

      ```
      module mod(a,b,c)...endmodule
      ```
    - ifdef on partial objects/expressions are not supported and may result into incorrect merge. If the complete declaration/definition of an object is under ifdef, that is preserved correctly.

      Consider the following example:

      ```
      input [`ifdef X7`else5`endif:0] p1;
      Or :
      `Ifdef Xinput`elseoutput`endif [7:0] p1;
      ```

The above kind of RTL syntax may lead to incorrect merging.

- ifdef support on the following objects is correctly available only if the object under ifdef is defined/declared in separate line of the file. Else, all the objects present in that line get the same configurability present at that line.
  - generate block
  - genvar, wire, reg declarations of glue block
  - positional port-map
  - positional param-map

    See the examples below.

Supported Syntax of generate Block

```
`ifdef FPGA
genvar i;
generate
for(i=0; i<SIZE; i=i+1) begin:sreggen
  sreg #(.Para1(7))s(clk,
        sdata[i], sdata[i1]);
end
endgenerate //supported
`endif
```

Unsupported Syntax of generate Block

```
genvar i;
generate
`ifdef FPGA
for(i=0; i<SIZE1; i=i+1)
`else
for(i=0; i<SIZE2; i=i+1)
`endif
begin:sreggen
  sreg #(.Para1(7))s(clk,
       sdata[i], sdata[i+]);
end
endgenerate
```

Supported Syntax of always Block

```
`ifdef BEHAVIOUR
  always @(posedge clk)
  begin
    out <= 8'b0 ;
  end
```

Unsupported Syntax of always Block

```
`ifdef BEHAVIOUR
  always @(posedge clk1)
`else
  always @(posedge clk2)
`endif
  begin
    out <= 8'b0 ;
  end
```

Supported syntax of complex assign

```
`ifdef X
assign a = b & c;
`else
assign a = d & e;
`endif
```

Unsupported syntax of complex assign

```
assign a = `ifdef X b & c `else d & e `endif;
```

Supported syntax of genvar/wire/reg

```
`ifdef X
genvar I;
wire w1;
reg r1;
`endif
```

Unsupported syntax of genvar/wire/reg

```
genvar l,`ifdef X m `endif,`ifdef Y n `endif;
wire w1,`ifdef X w2 `endif, `ifdef Y w3 `endif;
reg r1,`ifdef X r2 `endif, `ifdef Y r3 `endif;
```

- If objects, such as port, parameter, instance, and parameter override with the same name are visible inside multiple ifdef, they are handled by name conflict feature. However, this feature may not work in some situations.
- When multiple connections are present on a port-map terminal, extra buffer is generated to map connectivity.

27. Issues with preserving unsynthesizable constructs

    Following are the issues with the unsynthesizable constructs:

    - This feature does not consider SystemVerilog constructs.
    - For VHDL, when you load an RTL having unsynthesizable pragma blocks in any region (entity, architecture declaration, architecture body, package declaration, or package body), GenSys preserves only the unsynthesizable pragma blocks that are present inside an architecture body in the generated RTL.
    - Unexpected errors appear while loading RTL in the following cases:
        - When block comments are used as pragma comments (example: /*synopsis translate_off*/) and RTL codes are written outside the pragma block such that they appear in the same line on which block pragma comments are present

          For example, the following specification is incorrect:

          ```
          input a; /*synopsis translate_off*/initial begin a = 1; end /
          *synopsis translate_on*/ input b;
          ```
          The correct specification is as follows:

          ```
          input a;/*synopsis translate_off*/initial begin a = 1; end /
          *synopsis translate_on*/input b;
          ```

- While using `ifdef macros to encapsulate unsynthesizable constructs, you write RTL code on the same line as the macro.

  In such cases, RTL parts present with the macro in the same line are not preserved in the generated RTL.

  For example, the following specification is incorrect:

  ```
  `ifdef X initial begina = 1;end `endif
  ```
  The correct specification is as follows:

  ```
  `ifdef Xinitial begina = 1;end`endif
  ```
- In some cases, a pragma block is not identified as an unsynthesizable block even if it contains some unsynthesizable constructs.

  For example, the following pragma block is not considered as an unsynthesizable block even when function_a contains some unsynthesizable constructs:

  ```
  -- synopsys translate_offfunction_a();-- synopsys translate_on
  ```

28. Issues with the Memory Swapping Feature

    Following are the issues:

    - The following styles are not supported by this feature:
      - generate for and generate case styles
      - generate if or generate begin blocks that have glue logic or keywords other than if, else, begin, and end inside the generate block, that is the code between the generate and endgenerate keywords
    - You may see unexpected results if you perform memory swapping in the following cases:
      - If multiple overlapping generate blocks are present in the same line in RTL
      - If SystemVerilog RTL have a pre-label usage before the end keyword, as shown in the following example:

        ```
           generate begin:abc   if(sel == 1'b1)   A a();   abc : end
        endgenerate
        ```
      - If the generate begin style exists with multiple begin-end blocks. that is, both the begin-end blocks belong to the generate block, as shown in the following example:

        ```
           generate begin   if(sel == 1'b1)   A a();             end
        begin   if(sel == 0'b1)   B b();             end   endgenerate
        ```
      - If there are begin-end blocks that do not have any condition, as shown in the following example:

        ```
           generate  if(sel == 1'b1) : begin   A a();   begin     B b();
        end   end   endgenerate
        ```

- If nested if blocks exist inside a generate block.

29. Issues with the Selective RTL generation flow

    The issues are described below:

    - This flow is supported only for restructuring and not for assembly operations.
    - Modifications on VHDL is supported only on the std_logic type ports/signals.
    - Modifications to unpacked ports and VHDL/SV complex type ports are not handled by this flow. For all the modules for which such modifications are observed, RTL is generated by the default netlister.
    - Modifications on unrolled generate-if instances are not handled by this flow. Modification to such instances are handled by the default netlister.
    - Modifications to the unrolled array of instances are not handled by this flow. Modifications to such instances are handled by the default netlister.
    - The merge glue false flow is not supported by this flow. This flow cannot be enabled when the selective netlisting flow is enabled.
    - The group_glue flow is not supported by this flow. This flow cannot be enabled when the selective netlisting flow is enabled.
    - Glue movement is not supported by this flow. Therefore, glue movement is not possible in the selective netlisting flow.
    - Modification around the positional port/param-map is not handled by this flow. For all the modules for which such modifications are observed, RTL is generated by the default netlister.
    - Making existing parameters dependant on the newly added parameters causes STX errors and is not supported by this flow.
    - Deleting a parameter is not supported correctly by this flow.
    - Modifications on the modules with mixed style port/parameter is not handled by this flow. For all the modules for which such modifications are observed, RTL is generated by the default netlister.
    - Modifications of comments/pragmas on existing objects is not supported by this flow.
    - SV interfaces and interface connections are not supported by this flow.
    - Modules spanned over multiple files indirectly through `include are not supported by this flow.
    - Same net name cannot be reused in this flow.

- ❍ Pushing instances into a new hierarchy by using the group_design Tcl command causes loss of the look-and-feel of the generated RTL because the RTL of such modules is generated by the default netlister.
- ❍ This flow does not handle the named port map scenario where a port/terminal is outside ifdef but the connection is under ifdef. In such modification, RTL is generated by the default netlister.

30. Issues with Equivalence Checking feature
    - ❍ The equivalence checking feature is unsupported for the glue restructuring flow.
    - ❍ This feature does not work for the ifdef based configuration preservation flow.
31. Limitation of assembly flow
    - ❍ Assembly operations are not supported for SV interfaces and interface port connections.
    - ❍ Support for multi-d port, complex port and type in assembly flow has issues
    - ❍ Configurability (`ifdef and generate-if) is not supported in assembly flow
    - ❍ System Verilog global parameters are not supported in assembly flow
    - ❍ Pragma is not supported in assembly flow
32. Issues with Connector file generation
    - ❍ Connector file generation is not stable for assembly flow.
33. Other Issues
    - ❍ Cross-probing from the IP-XACT Viewer pane to a single-row table (such as the Info table) does not point to the correct cell.
    - ❍ IO assertion generation is not supported for one-hot muxes and adhoc/OEN mode.
    - ❍ GenSys GUI may hang on the following machines once the user performs the closeall operation:
        - ■ Red Hat Enterprise Linux ES release 4 (Nahant Update 6)
        - ■ Red Hat Enterprise Linux ES release 4 (Nahant Update 6)
        - ■ Red Hat Enterprise Linux Server release 5.2 Beta (Tikanga)
        - ■ Red Hat Enterprise Linux Server release 5.2 (Tikanga)
        - ■ Red Hat Enterprise Linux Server release 5.2 (Tikanga)

        - Flat RTLHC report on very large designs, involving more than 10k rows in the RTLHC table, is not loaded in the table view properly. The fix should be available in a future release.

    - GenSys may hang during any stage on the RHEL 6.0 platform.

      This problem is random and specific to the RHEL 6.0 platform.

      These problems will be fixed in a future GenSys release. Meanwhile, refer to the Platform Support section in the release notes document, exclude RHEL 6.0.