

Open Issues for Cleanup

in

Verilog-AMS

Revision: 0.9

Date : 1/22/2001
Author : Jon Sanders
Project ID: VerilogAMS LRM 2.0

1 Introduction

This document provides a list of issues that have been presented regarding issues with the Verilog-AMS LRM from Cadence that need to be clarified as the committee moves forward. Each item contains a title, a description of the problem and a potential solution if not obvious and available. These issues ordered by LRM section as much as possible. If you have questions please reply to the sender so that clarification can be provided to all.

1. (12) Section 3.2.1? - Coercion of strings to real allowed but not defined

```

module pureD;
parameter real rparam = "string1";
parameter integer iparam = "string2";
parameter sparam = "string3";
    initial begin
        $strobe("String Value assigned to real: \n PCg= %g \n
                PCf= %f \n PCna= ",rparam,rparam,,rparam );
        $strobe("String Value assigned to integer: \n PCs= %s
                \n PCd= %d \n PCf= %f \n PCna= ",iparam,iparam,
                iparam,,iparam );
        $strobe("String Value assigned to sparam: \n PCs= %s \n
                PCd= %d \n PCna= ",sparam,sparam,,sparam );
    end
endmodule

```

The above results in the following:

String Value assigned to real:

PCg= 1.76884e+09

PCf= 1768843057.000000

PCna= 1768843057

String Value assigned to integer:

PCs= ing2

PCd= 1768843058

PCf= 1768843058.000000

PCna= 1768843058

String Value assigned to sparam:

PCs= string3

PCd= 32497657065662259

PCna= 32497657065662259

The LRM implies that "string1" should be coerced to a real but does not describe how the coercion should be done. It makes a lot more sense that this would be an error. That would be far more useful for users. Similarly parameter integer iparam = "string2" should be an error.

Recommendation: Someone must better define this or we should make illegal. Note, this is certainly specified in 1364 so we must also work this through with them.

2. (22) Section 3.2.2 (When to do range checks?)

Should range checking be done on default values or only final values of an instance.

Recommendation: Checking should only be done on the final values as this feature is meant for users to set values not the model developers.

3. (20) Section 3.4.2 Connections to port expressions (whats a driver?)

Behavioral expression attached to ports

- should be able to indicate the disciplines of such ports somehow.

e.g;

```
module foo;
  reg a;
  reg b;
  Bar b1(a&b);
endmodule
```

How can the discipline of 'a&b' be indicated?

Recommendations: Cadence recommendation forth coming, several potential options

4. (21) Section 3.4.2 (OOMR disciplines on behavioral nets)

Should be limitations on OOMR declarations of nets that are used behaviorally - should be only

able to OOMR to a undeclared net. Therefore couldn't change the discipline of a net that was used behaviorally.

e.g;

```
module top;
  pll pll1();
  mechanical pll.f; // this should be illegal!!
endmodule
```

```
module pll (f);
  electrical f;
  analog begin
    V(f) <+ sin(w*$abstime);
  end
endmodule
```

V() is not the mechanical access function, it is the electrical access function, so is V(f) an error?!!

Recommendation: Make it illegal to use OOMRs to override the discipline of nets that are behavioral. Other nets should be able to be overridden to aid coercion.

5. (38) Section 3.4.3.2 (neutral disciplines)

Why do you need neutral disciplines if wire is already neutral

- should remove this feature?

Recommendations: Consider as part of discipline compatability issues of Annex E3

6. (8) Section 3.4.3.3 - LRM cleanup issue: TRI and WIRE are aliases

Recommendation: Specify that tri should be treated as wire at least in 3.6, other places?

7. (11) Section 3.5 - PCR 328294 Initial value of wreal nets not defined

The LRM says..

"If no driver is connected to a wreal net, its value shall be 0.0" It does not define the value of a

real net at $t = 0$. And of course a net cannot store a value (except trireg), its value as we know is only determined by its drivers.

If in an example however, we have a driver (continuous assignment.) to the real net. In the same example, if I removed the continuous assignment, at $t = 0$, $out2 = 0$. I am not sure as to what the value of $out2$ should be (at init.) when it has a driver.

Recommendation: value set to 0.0 if value hasn't been determined at $t=0$.

8. (5) Section 3.5 and 7.7.3 - Real value port examples have errors:

The examples in 3.5 and 7.3.3 have errors that prevent them from working without additional changes. The example in 3.5 should be changed as follows:

```
// The following three lines should be added so that a wreal is passed into foo
wreal wstim;
assign wstim=stim;
foo f1(wstim,load);
// foo f1(stim,load); //This line should be deleted3 as it is illegal for ports of type real
// dut d1 (load, out); // This line should be deleted as it provides not added value
```

The example in 7.3.3 should be changed as follows:

First there is no top level module that instantiates the two blocks so add:

```
module top ();

wreal stim;
reg clk;
wire [1:8] out;
teststim tb1 (stim, clk);
a2d    dut (out, stim, clk);
initial  clk=0;
always  #1 clk=~clk;
endmodule
```

In addition, the testbench module must be converted to use wreal since it is passing a real value through one of its ports (the whole reason for wreal). The following fixes this issue:

```
module teststim (wout,clk); // change output port to wout
```

```

input clk;
output wout; // change output port from out to wout
real out;
wire clk;
wreal wout; // add wout declaration as type wreal
assign wout=out; // assign wreal (wout) value to be value of real (out)
....

```

Recommendations: Make the above changes as shown

9. (32) Section 3.6 (default_discipline clarifications)

In the first paragraph, it seems that the word scope is used in two different ways which is potentially confusing;

1. as the scope of application of the compiler directive
2. as an argument to the default_discipline compiler directive.

Also does the scope argument only apply to the refer instance or does it apply to the children of that instance too?

Recomendation: Clarify this paragraph to make clear

10. (33) Section 3.6 (default_discipline only for digital?)

Is default_discipline only applicable to digital? If so then need to remove references to 'default_discipline electrical e.g. p3-20 of 2.0 LRM.

Recomendation: Resolve analog default_discipline (section 11.1) and then ensure that this section is in alignment.

11. (31) Section 3.7 (Discipline presendence issues)

'default_discipline as a compiler directive seems like a very poor approach for library-based simulation and for simulation based on configurations. It makes much more sense to put this information into a design unit such as connectrules or the config. It is also more consistent with the way configs, connectrules are handled.

The compiler directive 'default discipline is not a very suitable way to specify how a hierarchy flattening action (discipline resolution) is to be done.

Okay maybe someone could use the scope, qualifier fields of the default_discipline. However these refer to instance names and there is no precedence in compiler directives for doing that. Instances haven't been created at compile time.

How are conflicting default_discipline references to be resolved?

e.g. in one part of a file there is;

```
'default_discipline electrical top.foo.bar
```

and in another part of the file there is;

```
'default_discipline mechanical top.foo.bar
```

Recommendation: Items 3 and 4 in the precedence list (those referring to instances) should be removed from the list. For conflicts in compiler directives the last one should win but the LRM should be updated and a rule added that these compiler directives for the same net must be compatible.

12. (41) Section 3.7 - disciplines rules of branches

What are the rules for decid-ing the discipline of Branches? - LRM isnot clear on this. Section 3.7

Recommendation: Someone needs to define this.

13.(37) Section 3.9 branches - clarifications

- should say that branches cannot be declared using discrete nets.

- Clarification of vector branches is needs

- 1) It should be illegal to create a vector branch from Vector terminals of different sizes.
- 2) It should be illegal to create a vector branch from Vector terminals of different directions or else it should be specified how they are connected up.
- 3) When a vector branch is created from vector nets, it range size (direction?) should be the same as the vector terminals.

Recommendation: This feature either be clarified or removed

14. (42) Section 4.5.1 and 6.7.4 - Initial Conditions

What mechanism should be used to set initial conditions; (analysis("ic")) or @ (initial_step("ic"))? If so how does it work and if so, how will a piece of code like the following behave;

```
@(initial_step("ic"))
```

```
V(out) <+ V(in);
```

Recommend that this feature be removed or clarified

15. (24) Section 5.1.6 (Implicit Switch Branches?)

What is the behavior of

```
if (open)
  I(p,n) <+ 5;
```

Is this equivalent to;

```
if (open)
  I(p,n) <+ 5;
else
  I(p,n) <+ 0;
```

Recommendation: These should be considered the same, clarify in LRM.

16. (25) Section 5.3.2 (Indirect assignments in conditionals)

Indirect branch assignments should be illegal inside conditionally executed statements. The LRM doesn't state this and doesn't indicate what behavior should occur if it happens;

e.g.;

```
analog begin
  if ( xx == 2) then
    V(out) :ddt(V(x)) == 0;
  end
```

Recommendation: This should be stated as illegal, like other conditionals

17. (27) Syntax 6-1 and BNF

To consistent with 1364 formulation, there should be semi-colons after:

```
analog_branch_contribution
analog_indirect_branch_assignment
analog_procedural_assignment
procedural_assignment
```

Recommendation: Make above changes

18. (28) Syntax 6-3, Syntax 6-4, Syntax 6-5 and BNF

These should contain no semi-colons after changes to syntax 6-1 above.

Recommendation: Make above changes

19. (26) Section 6.4 (Switch branches illegal in BNF)

The BNF of the conditional_statement disallows switch branches. In 6.1, strongly recommend that attempts to limit analog_statements inside analog statements using BNF be removed and replaced by a semantic restriction. It is impossible as far as I can tell!

See restrictions on analog operators in 4.4.1 as this also implies that switch branches are illegal.

Recommendation: BNF should be adjusted to allow these. May need to do limitations on conditionals as semantic rules.

20. (19) Section 7.2 and 1364 - defparam vs. instantiation precedence

This comes up in netlisting as the 1364 LRM is really weird in this space. The defparam precedence is defined by last one seen (like a compiler directive) which in a netlisting environment sucks. If searching libraries then the result stated by 1364 is unknown, yek! This should be by level in the hierarchy and then instantiations should be treated as the same as defparams.

Recommendation: If this does not get cleaned up to be reasonable we will need to ensure that the issue is addressed when global design variables are supported.

21. (23) Section 8.2.4 : Compatible disciplines

Compatibility of continuous disciplines on the same signal. Should be stated that the continuous disciplines of a signal must all be compatible as they are solved as the same node.

Recommendation: Make the above changes to the LRM

22. (14) Section 8.3.6.4 and 8.3.1 : Clarification on X and Z .

These sections should be enhanced to ensure that users and implementers understand that when accessing a digital net, X and Z must be dealt with prior to assigning a value to a variable and certain functions like case, casex, casez, ===, and !== need to be used to prevent errors.

We need to close on if == and != can be used in analog on digital signals and if so does X or Z mean a failure of the comparison? Supporting them would require being able to do comparisons against X and Z only. The bottom line is analog cannot be assigned a value of X or Z. Do we need to allow the user to specify what X and Z would be set to?

Recommendation: Currently we should limit analog to not support == and != when the signal is digital and contains X or Z. We should clarify this more in the LRM with the core agreement that signals cannot be set to X or Z in analog.

23.(1) Section 8.4.4.1 and Annex F - Discipline Resolution: No clear definition on how to deal with "leaf level" wires.

Leaf level wires (net segments) are primarily the result of alias modules used in netlisters and pass through's as a result of synthesis. These wires (net segments) have no components connected to them at the specified hierarchy thus the only connections to this wire (net segment) are higher in the hierarchy.

While in the detail resolution the discipline is and can be passed down into this net segment if needed the same is not true for the default method. In default all disciplines are passed up the hierarchy but these net segments may not have a discipline to pass up.

Recommendation: The default method needs to have additional clarification that this special cases must resolve their discipline by looking up the hierarchy until a discipline can be defined. If two or more ports are connected to this leaf level wire that would pass down a different discipline the basic rules apply, disciplines must be compatible and continuous wins over discrete. Cadence will provide the update for section 8.4 and the Annex

24.(2) Section 8.4 and Annex F - Discipline Resolution: No clear definition on how to deal with out of module references (OOMRs).

OOMR connections such as (.out(top.middle.bottom.in)) must be considered in discipline resolution. There are two options in addressing this issue.

- a) OOMR reference receives discipline from connection
- b) OOMR connection receives discipline from referenced net

In one the referenced wire (top.middle.bottom.in) is impacted by the connected wire (out) discipline. In the second case the wire (out) is impacted by the resolved discipline of (top.middle.bottom.in)

Recommendation: The connection should drive the discipline of the OOMR referene. Cadence will provide the update for section 8.4 and the Annex

25. (29) Section 8.6 (bi-dir issues)

The text suggests that a bidir can be connected to a port that is being driven by a reg or an expression. This doesn't make an sense!

Recommendation: This needs to be fixed as a reg can drive but cannot be written to from outside the module.

26.(7) Section 8.10.5 - net_resolution function: No one liked this so if we are going to change it lets do it now.

We have used the assign `dVal = dVal;` without much problems or complaints. The real issue is explaining driver receiver segregation not the syntax. Also, in 8.10.5 if we keep this *syntax we should change it to be `net_resolution(port_identifier, net_identifier)`*. LRM needs to clear up some things here i.e. 1) assign `d=d` will not work if `d` is a reg, 2) `net_resolution` is pretty hokey.

Recommendation: Get rid of `net_resolution` and move back to assign statement. With connect modules now marked by the `connectmodule` keyword we should consider making assign `dVal=dVal` a default (not required to be specified) and then do a better job documenting how all of this works. What about if `dVal` is a reg?, does it get segregated in a connect module? probably not. Should reg's be allowed on connect module ports since they cannot "listen"? suspect this is needed so need to resolve this

27.(36) Section 8.11 (Supplementary drivers and delays)

Supplementary `driver_update` functions are insufficiently well defined in terms of what delays should be accounted for?

```
a = #1 b;
#1 a =b;
```

must both of these be accounted for? How about SDF delays?

How about;

```
#1;
$strobe("testing");
a = b;
```

Recommendation: These should be removed, made informative or more clearly defined.

28.Section 9 : Which solver starts first?

Initialization sequence of the simulation should be described - does the digital kernel or the analog kernel go first?

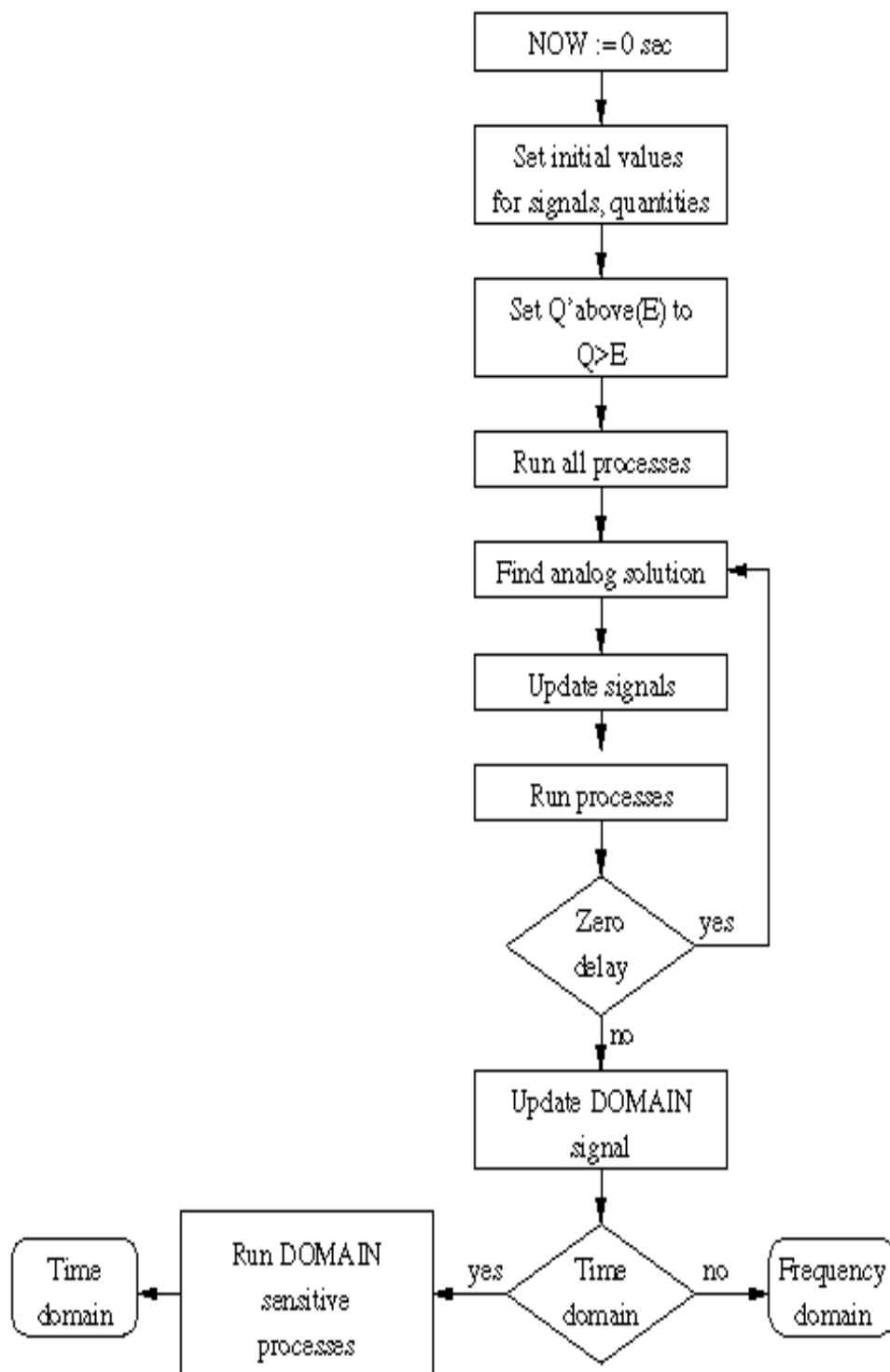
Recommendation: Specify that digital start first as part of the VHDL-AMS sync effort. Needed for simulator commands at time zero as well as mixed language simulators

29.(42) Section 9 Initialization method - Different from VHDL

With the movement to mixed language simulators this is an issue and while we could say it is their problem I think we need to address it. We can either switch to theirs or we can specify both methods and let the user select between them. For circuits with VHDL we might be forced to use only theirs.

Recommendation: Specify the same method as VHDL-AMS. Needed for simulator commands at time zero as well as mixed language simulators

The following is the VHDL method:



30. (15) Section 10.2 1364 sync-up: Random function for analog not clearly defined but in 1364 it is as the code is now provided.

Recommendation: Sync up with 1364-2001 ASAP

31. (39) Section 12.5.2 (VPI Issue)

Nature and discipline should not use param_assign, instead there should be a new object created called attr_assign.

Recommendation: Make the above change

32.(4) Annex A: BNF clarification

a. Connectrules:

LRM states that the connect statements can have 'zero or more' count in the connectrules block. For example:

```
connectrules AMSconnect;
endconnectrules
```

Recommendation: Either allow (might be useful for tools that create these rules) or changed to one or more.

b. Connectmodule:(8.5 and A6)

8.5 is missing net_resolution and both are missing digital sections at a minimum. Someone needs to look at this.

Recommendation: Add the missing data.

33. (18) Annex B - flow and potential, should these be global keywords?

May not be a big of an issue after 1364 pulled the rug out from us on the section specific keywords. Should we go back to one single list?

Recommendation: Consider impact of single list of keywords. Must move potential and flow to global keywords (from B2 to B1) as they are needed in accessing attributes (see 5.2.2)

34.(3) Annex C changes that were missed:

a. null argument change:

In VerilogA 1.0 the following was allowed: {} while in VerilogAMS 2.0 null arguments are defined by ", ," (comma-nullarg-comma). Thus {} would be represented as {,}. Null args for things like laplace and z-transforms of no numerator must be written as either { 1 } or { , }.

Recommendation: This needs to be added to the list of changes and possibly shown in the areas where most likely to occur such as in the laplace and or z transform sections.

b. The original 1.0 LRM version had the roots specified as the product of terms like:

$$(1) \quad (1 - z^{-1})/(r_r + j r_i)$$

...so the poles and zeros are roots of the polynomial in $z(-1)$, which makes sense given the z_i form is a polynomial in $z(-1)$ -- to get your poles and zeros you just factor the polynomial. But, the 1.4 LRM has:

$$(2) \quad (1 - z^{-1})*(r_r + j r_i)$$

...which means that the term goes to zero if $z = (r_r + j r_i)$, so the roots are of a polynomial in z , the inverse of the above.

Recommendation: Researching to determine why this change was made. Need to find out reason for change to determine if going back is even feasible. Need to add this to table also.

c. Verilog-A defined the argument to \$bound_step() to be constant.

Verilog-AMS allows "expression" as an argument to \$bound_step() which can be dynamic.

Recommendation: Add this Verilog-A 1.0 typo to the list of changes in Annex C.

d. The \$random changed and should be included I believe.

One may argue that only our implementation will change as others interpreted the Verilog-A LRM to be what is in 2.0. Also, we will be changing to match up with VerilogXL meaning shortly.

Recommendation: Add note in Annex C.

35. (16) Annex D -Discipline and Constants file corrections:

In review of the disciplines.vams file, the following were observed for discussion and correction. Are these suppose to be based on SI? how do we deal with those without SI units?

a. The units "coul" should be "C" for nature Charge. This is the standard SI symbol. Of course, that could confuse some folks with temperature would be "degC".

b. The unit on Angle would be "rad" to be SI compliant. And "rad/s" for

Angular_Velocity and “rad/s²” for Angular_Acceleration. Each of these put an “s” at the end of rad.

- c. The physical constants should be listed with a reference. What is the source of these values? The 1998 NIST values differ from those given for physical constants:**

charge: 1.602176462e-19

light: 2.99792458e-8

boltzmann:1.3806503e-23

planck: 6.62606876e-34

...etc...

Source:physics.nist.gov/cuu/Constants/index.html

- d. Both Boltzmann and Planck are misspelled in the constants file.**

Recommendation: Consider the above changes

36. (10) Annex D: -upcase issues with disciplines.vams file

There is also a clash between the nature "Force" and the Verilog keyword "force" when doing -UPCASE. In addition each nature like Voltage is case sensitive (note uppercase "V") which when used in -upcase cannot work unless we define something else.

Recommendation: Leave as is but provide a warning to the users about these conflicts.

37. (34) Annex E.

If a SPICE master has the same name as a Verilog master, which matches or is it an error?

Recommendation: Thoughts? Do we allow overriding of analog primitives? Will the LRM force a specific implementation?

38. (35) Annex E2: Case sensitive SPICE simulators

Some spice simulators are case-sensitive, this should be accounted for too. Issues exists with the case-insensitive matching of SPICE components. Verilog is a case-sensitive language and if I write a Verilog construct (in this case an instantiation), it should comply with with the rules of Verilog

i.e. case insensitivity. Believe it is better and more consistent with Verilog if all SPICE references must be lower-case, then the binding algorithm is much less complex.

Consider that if I define a model called ‘Cap’ and a module called ‘cap’. However if I type ‘cap’ and expect to get ‘Cap’, I would be wrong.

Recommendation: Require all SPICE references to be lower case.

39.(9,30) Annex E.3 - Disciplines of analog primitives: How to set and defaults.

Analog primitives cannot via the language get a discipline defined so we need to specify a default value and a method for setting the disciplines.

Recommendation: Lots of possibilities, part of discipline resolution method, use OOMR declarations, default_analog_discipline? (of course compiler directives suck in most of today’s solutions) See more below.

39b.Section 3.8 and Annex E.3 - Compatible disciplines to analog primitives: Certain primitives are not limited to electrical domain.

Primitives like sources, resistors, capacitors, ... can often be used in mechanical and other domains but if they are electrical then they will not be compatible.

Recommendation: Analog primitives should default to domain continuous (neutral) and the discipline should resolve as follows: (THIS NEEDS MORE WORK!)

i. The discipline of defined primitives and behavioral blocks connected to each port

- If incompatible disciplines exists then error

ii. The global analog discipline

iii. The default discipline

The above assumes that the default discipline will be electrical for continuous domains and that there is a mechanism(s) for setting the global discipline and the discipline of specific ports.

Disciplines of ports could be set via OOMR discipline declarations (mechanical top.II.I2.R1.a) where a is the port name.

40. (13) LRM Cleanup:(Typos)

a. Section: 3.4.1.1 and 3.4.3.5 : A user define attribute is specified called “max”, this is a keyword and not be used here. Should change to something like “maxvalue”.

- b. **Section: 4.4.14, table should have absdelay not delay**
- c. **Section: 6.7.4: In table 6-1, @final_step for DCOP should be 1 not 0.**
- d. **Section: 8 : Several places have the keyword merged listed as merge**
- e. **Section 8.2.3 : The figure has Net C.b_out which should be Net C.c_out**
- f. **Section: 8.3.1, 8.8 : Mixed signal examples falsely use == comparison of digital signals in analog context. Need to change examples to use methods that support X and Z.**

8.3.1 Example:

```

.....
real aout;
analog begin
  if (in === 1)
    aout = 3.0;
  else
    aout = 0.0;
  V(out) <+ aout;
end
endmodule

```

8.8 Example:

```

.....
analog
  V(e1) <+ transition( (cm === 1) ? 5.0 : 0.0 );
.....

```

- g. **Section: 8.3.2 : Next to last paragraph is in error, change to: *and statements that are casez, shall report ...* Add the that and drop the which.**
- h. **Section: 8.6: p8-17, Last line should be change to "and whose other connection is compatible with electrical"**
- i. **Section: 8.10.6 : In the example on 8-37 of the 2.0 LRM, has input and inout as the directions for the ports of the CM. They both had to be inout if one is**

- j. Section: 8.10.6 : The example has "initial net_resolution(d,out);" initial is not needed**
- k. Annex C : The Table C-1 first item should be \$abstime not \$atime.**

Recommendations: Make the above changes

41.(40) Global Issues: Support for global design variables (accessible through-out hierarchy)

Can do some of this via defparams and OOMR but can get very ugly and many limitations that make this difficult to be considered a viable solution as-is.

Recommendation: Need to consider the dynamic parameter proposal. Lots of issues with current capabilities that this must resolve.