Hi, All -

One of the stated purposes of creating the Accellera SystemVerilog 3.0 Standard was to give vendors a chance to start implementing features and discover problems tat could be corrected in the SystemVerilog 3.1 Standard.

I believe I have discovered use-model problems that require changes to one of the SystemVerilog 3.0 enhancements. The "logic" type is flawed and should be changed.

I have just spent the past six hours trying to code a simple-to-moderate 9-sub-block design using interfaces, wires and logic variables (and I am still not done) that took me only a half hour to code using .* port connections. I intend to share my results with a few key players when I get done. I do not intend to propose removal of interfaces, but it should be noted that interfaces are not the dream-interconnection scheme that I once thought they were. We should be careful about over-promoting interfaces as they are difficult to use, abstract to understand and verbose in most applications. The documentation in the SystemVerilog 3.0 Standard is sorely lacking in a number of areas, such as handling unconnected ports in an interface and how they are used in a multi-module design where port-lists are split between 3-10 connecting modules. What if an interface port is connected to another module but only part of the same port is connected to a third module? By the way, this is easily handled using .* connections.

In the mean time, I have come to near-despise the "logic" data type so my highly editorialized comments and serious proposals for fixing the shortcomings of this data type follow.

logic type features:

1. Variables of type logic can be assigned from either a procedural assignment, continuous assignment or driven by a module or primitive instance (user-define module, user-defined primitive and built-in Verilog gate or switch primitives) - This is good!

2. Feature #1 means that variables can be declared as a "logic" type and if an assignment to the "logic" variable changes from being a procedural assignment to a continuous assignment, no declaration change is required - This is good!

3. A variable of type logic does not permit assignments from more than one procedural block - This is good!

4. A variable of type logic does not permit assignments from both (a) procedural blocks, and (b) drivers, (drivers are defined as any one or more of the following: continuous assignments, gate or switch primitives, user module outputs or inouts) - This is good!

5. A variable of type logic does not permit assignments from multiple drivers, defined as any one or more of the following: continuous assignments, gate or switch primitives, user module outputs or inouts - THIS IS BAD!

6. Feature #6 prevents logic from becoming the default data type for SystemVerilog (something we considered a year ago), because it would break existing Verilog models with multiple drivers, such as multiply driven data buses, onehot multiplexers and bus crossbars. This means that all logic variables must be declared - THIS IS BAD!

For Verilog2001, the rules for using net types versus variable types was silly but simple: (1) anything that appears on the left hand assignment inside of a procedural block, must be declared as a variable type (typically reg), (2) everything else is a net. No exceptions! (silly but simple!).

1-bit nets do not have to be declared.
Net-buses that attach to ports do not have to be declared.
Internal net-buses must be declared.
All 1-bit variables must be declared either in a port declaration or as separate declarations.
All vector-variables must be declared either in a port declaration or as separate declarations.

For SystemVerilog 3.0, the guidelines will change to something strange:

For SystemVerilog, the rules for using net types, variable types and logic types is even more silly and not as simple: (1) anything that appears on the left hand assignment inside of a procedural block, must be declared as a variable type (typically reg) or as a logic type, (2) everything else is a net or a logic type, (3) logic types cannot be assigned from more than one procedural block, driver assignment, or combination thereof.

1-bit nets do not have to be declared.
Net-buses that attach to ports do not have to be declared.
Internal net-buses must be declared.
All 1-bit variables must be declared either in a port declaration or as separate declarations.
All vector-variables must be declared either in a port declaration or as separate declarations.
All 1-bit logic variables must be declared either in a port declaration or as separate declarations.
All vector-logic variables must be declared either in a port declaration or as separate declarations.
Most "interface" definitions require logic declarations because interfaces are typically assigned from a procedural block source and drive a destination.
Some "interface" definitions could use net declarations when the interfaces are assigned from a non-procedural block source and drive a destination.
Some "interface" definitions must use net declarations when the interface ports are assigned from multiple non-procedural block sources (such as a multiply driven data bus, crossbar switch or onehot mux) and drive one or more destinations.

Engineers (myself included) will avoid using the logic type because I have to declare it everywhere! The additional declarations add verbosity to a design and do not make the code any more readable or the model any more understandable. In my opinion, all of the extra "logic" keywords inserted into my code would serve more as a distraction than an enhancement.

I guess language trainers like myself should not complain because it means that the language is now more confusing and I should acquire more training opportunities due to the added complexity of the language.

The "logic" data type almost fixed a long-standing problem with the Verilog language (regs or wires?) but not quite. Because it did not fix the problem completely, we could not make the new "logic" data type the default data type for SystemVerilog.

Proposal #1 - remove the restriction described in feature #5. Allow multiple driver-assignments to logic types.

Proposal #2 - make the "logic" the default data type of SystemVerilog. This is possible and desirable if Proposal #1 passes. All existing Verilog models would work if proposals #1 and #2 pass.

For SystemVerilog with enhanced, default-logic type, the rules for using net types, variable types and logic types gets much easier. (1) use logic types everywhere, (2) only use variables types from a testbench when required and ask yourself if there isn't a better way to write the testbench using logic types instead of variable types!

1-bit logic variables do not have to be declared.
Vector-logic variables that attach to ports do not have to be declared.
Internal vector-logic variables must be declared.
1-bit variables or vector-variables must only be declared if multiple assignments from different procedural blocks are made to the variables (this will be rare and only occur in a testbench - good testbench coding styles would avoid this coding style).

All other exiting Verilog-2001 rules would apply, all existing Verilog-2001 models would still simulate and synthesize correctly, but they would be rarely used by educated SystemVerilog engineers!
(the following are still true but would become mostly obsolete and replaced by logic declaration coding styles)
1-bit nets do not have to be declared.
Net-buses that attach to ports do not have to be declared.
Internal net-buses must be declared.
All 1-bit variables must be declared either in a port declaration or as separate declarations.
All vector-variables must be declared either in a port declaration or as separate declarations.

Benefits of this enhancement -

- Most usage of net and variable types can cease. The logic type replaces almost all of them.

- No more changing declarations when an assignment moves from a procedural block to a continuous assignment.

- The logic type will become the preferred data type for modeling and synthesis.

- The logic type will "do the right thing!"

- The logic keyword will only be required for internal logic-vector-buses-variables. Fewer required declarations (unless you use that ill-advised `default_nettype none compiler directive, for designers or teams who feel that everything in a model should be declared - *gag!*)

Conclusion - There is still time to fix the "logic" data type. Let's not foist this shameful, ill-tempered new data type on the RTL-coding masses. Let's fix it and be proud of it!

I have decided to submit a paper for DAC next year. The title of the paper will be:
"The New SysteVerilog "Logic" Data Type - Friend or Foe?"

The Conclusion of the paper will either be a warning, or a glowing endorsement with usage guidelines. I am waiting for the outcome of this proposal before I decide which ending to write.

Regards - Cliff