Original issue is in red.
Response and action is in blue

AI-16: 13.3: Changes to Section 13.3 related to AI-16
Handled in CH-47

AI-17: 13.3: Clarify “after all nonblocking assignments”to be at the appropriate verification phase (from the Scheduling Semantics).
Handled in CH-84

AI-18: 13.5: Add clarification that the same clock can be in multiple clocking domains
AI-19: 13.5: Move discussion about drive semantics (last two sentences of paragraph) to section 14.4 and remove reference to "logic" semantics.
Handled in CH-85

AI-20: 13.11: Deal with the issue of sub-modules on clocking domains
AI-21: 13.11: Modify description to make similar to declaration for timeunit
Handled in CH-53

AI-22: 14.2: Add Reference to 13.9 as way to synchronize
=> *Note that the second example shows this explicitly*
Handled in CH-93

AI-23: 14.2: Rework definition to use event_control
=> A clocking variable is an expression. No new grammar is needed (excerpt is unchanged)
Handled in CH_94

AI-24: 14.2: Add example of shorthand is.
Cancelled due to lack of definition of action item.

AI-25: 14.2: Look at moving into Section 13 (evaluate issues including effect of assertions)
Handled in CH-95 and CH-101

AI-26: 14.4: Provide same semantics for delayed vs. non-delayed version of skew on signal drive.
AI-27: 14.4: Expand explanation to define what a drive means in this context and remove the confusion of signals, drives, etc ...
AI-30: 14.4: In delay definition reference 13.10.
AI-32: 14.4: Resolve change from signal to clockvar.
Handled in CH-96

AI-29: 14.4: Provide "equivalent" code expansion as part of the definition.
=> I don't know that this belongs in the LRM. Perhaps it should be in a separate Annex dedicated to implementation.

```
module( inp, out );
…
 clocking dom @(posedge clk);
   input  #1ps inp;
   output #2ps out;
 endclocking
 …
 reg r;

 r = dom.inp;
 ##5 dom.out = ~r;

endmodule

reg dom_clk_in, dom_clk_out;

always @$CK( dom_clk ) dom_clk_in <= dom_clk;
always @$EV( dom_clk ) dom_clk_out <= dom_clk;

// Pseudo code that implements the input sampling
wire dom_inp_itmp;
reg dom_inp;            // This is the input clockvar: dom.inp
assign #1ps dom_inp_itmp = inp;
always @(posedge dom_clk_in) dom_inp = dom_inp_itmp;

// Pseudo code that implements the output drive
reg dom_out;            // This is the output clockvar: dom.out
reg dom_out_otmp;

always @(posedge dom_clk_out ) dom_out_otmp = dom_out;
assign #2ps out = dom_out_otmp;

// Procedural block translation
r = dom_inp;
repeat (5) @(posedge dom_clk_out );
dom_out = ~r;
```

@$CK -> Active during the clocking phase
@$EV -> Active at the end of the verification phase
Note: These operators cannot be reliably implemented to be cycle accurate. It is possible to implement delta-cycle accurate versions using sequences of non-

blocking assignments. In general, the number of delta-cycles is unknown and may vary during simulation.

AI-31: 14.4: Move "write semantics" from 13.5 to the end of 14.4.
Handled in CH-97

AI-33: 14.4.1: Expand description in last paragraph
Handled in CH-98

AI-34: 14.4.2: Add reg vs. wire distinction to discussion and support for drive(reg) and variable.
Handled in CH-99

AI-44: 9.9: Resolve Jay's comments on Process control tasks from his email on 1/13/2003
Handled in CH-44, CH-45, CH-88, CH-100
AI-53 opened for Jay

9.1 - Reliance on an if statement as test and set
The phrase "a single statement (not a block)" is not “well defined”.
Verilog does not guarantee this type of non-interruptible statements.

Jay has action item to forward to SV-BC.

9.7 - "The statements can be any valid statement or block of statements enclosed by begin ... end."
The statement was removed in CH-44 and is approved.

9.7 - "the specification of execution in source order"
The statement was removed in CH-45 and is approved.

9.7 - use of keywords all, any, none
I agree with the editor that these keywords are too common. Much discussion has been had on the reflector, the suggestion of allowing a count is sort of growing on me.
Resolved in vote on 3 Feb. Covered in CH-88 and is approved.

9.7 - join none - sub-processes do not start until parent thread blocks
This is another attempt at creating determinism by defining a finer grained process execution. It should be removed.
The section was removed in CH-45 and is approved.

9.9.1 - $wait_child()
Why is this a system task. Perhaps a parameter or variation on the 'wait' command like wait fork;

Ch-100 changes $wait_child() to wait fork

9.9.2 - $terminate()
Why is this a system task? Why not just extend disable as 'disable fork;' to explicitly kill forked children?

CH-100 changes $terminate to disable fork

9.9.2 - termination of simulation
The paragraph begins. "By default, SystemVerilog terminates ... when all its programs finish executing".

Is this a mentioned elsewhere? Typically simulation will finish when there is no further activity of any kind, not just programs. What if these are in always blocks? A sequential block should be able to wait for its children, but it has nothing to do with when simulation terminates.

Is this an artifact of Vera calling tf_dofinish() when integrated through PLI?

CH-100 handles.

9.9.3 - $suspend_thread
Regular caveat ... Why is this a system task? Can a user override it. ...
Paragraph with "#0 ... may also be called after nonblock assignements where 0-delay is ill-advised"
Any use of zero-delay through $suspend_thread(), #0, R/W Sync callbacks to get deterministic results is ill-advised and this just adds another mechanism to get yourself into trouble.
What exactly is the semantic of this if it is not the same as a #0? Run again in the current event queue? Are you defining the relationship between other $suspended threads?
This should just be removed. It only adds to the non-determinism

This type of functionality is often necessary in the testbench. Every single testbench tool includes some sort of mechanism for accomplishing the same function, and SystemVerilog will need it too, whether ill advised or not.

Open Issue: Should this become a statement? suspend_thread instead of a system task? Leaving it as a system task would allow someone to turn this off if they felt it was such an ill advised feature

CH-100 modifies the definition of when $suspend_thread is called.