Cliff Cummings - SystemVerilog 3.1-Draft 4 Review - update

Responses and clarifications to some of David's responses.

14.1 - 1st paragraph - better wording

WAS: Although SystemVerilog is used for more than simulation, the semantics of the language are defined for event directed simulation, and everything else is abstracted from this base definition.

PROPOSED: Although SystemVerilog is used for more than simulation, the semantics of the language are defined for event directed simulation, and all other event ordering is abstracted from this base definition.

DWS: This changes the meaning of the sentence in a way that is incompatible with what was voted on. No change.

Cliff: how is "everything else" better than the proposed wording. If my wording is wrong, then let's at least fix "everything else," which to me, could include "all other event ordering." I was trying to narrow the scope of "everything else."

14.3 - 5th paragraph after numbered time-slot list - wrong word??? (device does not make sense)

WAS: The sampling time of sampled data ... This **#1step** construct is a conceptual device that provides a method of defining when sampling takes place, and it is not creating a requirement that an event be created in this previous time slot. ...

CORRECTED???: The sampling time of sampled data ... This **#1step** construct is a conceptual delay that provides a method of defining when sampling takes place, and it is not creating a requirement that an event be created in this previous time slot. ...

DWS: Actually device is correct. It is not a conceptual delay it is a device or mechanism that provides a method etc… No Change.

Cliff: Then could I suggest replacing "a conceptual device" with "an abstract conceptual mechanism?" The term "device" has hardware meaning and I have got to believe that the use of this term will confuse more hardware engineers than just me.

15.3 - 2nd paragraph after "clocking dram ..." example - is this right??? What are we thinking???
An input skew of **#0** forces a skew of zero. Inputs with zero skew are sampled at the same time as their corresponding clocking event, but to avoid races, they are sampled in the observed region. Likewise, outputs with zero skew are driven at the same time as their specified clocking event, as nonblocking assignments (in the NBA region).

Adding another ugly #0 rule is abhorrent. What is intuitive about a #0 input sample being taken AFTER a nonblocking assignment?? #0 delayed assignments are scheduled before nonblocking assignments. This syntax will probably meet stiff resistance in the IEEE VSG. Why would I ever sample an input after nonblocking assignment updates? Could someone show me a useful example? Even if there is a useful example, the syntax should be changed.

I guess this means you can drive outputs (nba region) and sample those outputs if they are also tied to the inputs (observe region) and cause this to trigger always blocks that schedule more blocking and nonblocking assignments?

DWS: This is not another ugly #0 rule. It is replacing an ugly #0 rule with a well-defined behavior. You were there. This has been thoroughly discussed. No change.

Cliff: (ugly) #0 blocking procedural assignments are scheduled into the inactive region, and executed before nonblocking assignments update (per 1364-1995 and 1364-2001). Now #0 input skew will be sampled after nonblocking assignment updates. Did I miss something? Have the ugly #0 blocking procedural assignments moved to a new region? If so, this will break existing designs. If not, we now have #0-uglies both before and after nonblocking assignments. This is my objection. I missed the second SSWG meeting and apparently I missed a very important discussion. I can't speak for the entire VSG, but I have got to believe this is going to cause VSG-grief.

15.4 2nd paragraph (after the "clocking cd1 ..." example) **-** wrong use of terms and confusing terms

In the IEEE Verilog Standard, a *part-select* is a range of bits within a word. A *slice* is something like a row of words out of an array. What is a "*combination of signals*??"

WAS: However, hierarchical expressions are not limited to simple names or signals in other scopes. They can be used to declare slices, concatenations, or combinations of signals in other scopes or in the current scope.

SEMI-BETTER (still needs correction): However, hierarchical expressions are not limited to simple names or signals in other scopes. They can be used to declare part-selects, concatenations, or ???combinations of signals??? in other scopes or in the current scope.

DWS: Slice is defined in 4.4 and is part of SystemVerilog 3.0. The other question I am too tired to figure out. Hopefully Arturo will respond.

Cliff: The VSG refers to part selects to pick a range of bits from a word. The VSG is currently discussing how an array slice (one or more dimensions of words) should be handled by the @* and always_comb. I am just giving you IEEE VSG info. Perhaps the 4.4 slice definition should change.

15.8 **-** module top example **-** to make the example clear, use named port connections. By the way, what is generating the phi1 and phi2 clocks??

```
module top;
        logic phi1, phi2;
        bus_A a (.clk(phi1));
        bus_B b (.clk(phi2));
        test main (.a(a), .b(b));
        cpu cpu1 (.a(a));
        mem mem1 (.b(b));
endmodule
```

DWS: No change since what is there is correct.

Cliff: It is correct but it is also confusing. Named ports really help with this example. I still don't know what is generating the phi1 and phi2 clocks.

15.8 - paragraph before last example and last example. "Alternatively" is confusing. Dangling clocking blocks are confusing.
The clocking domains of program test could have also been written using both interfaces and hierarchicalexpressions as shown program test2 below:

```
program test2 (bus_A.test a, bus_B.test b);
        clocking cd1 @(posedge a.clk);
                input data = a.data;
                output write = a.write;
                inout state = top.cpu.state;
        endclocking
        clocking cd2 @(posedge b.clk);
                input #2 output #4ps cmd = b.cmd;
                input enable = b.enable;
        endclocking
        // ...
endprogram
```

DWS: True but what is there is not unclear. Made a change to clarify the sentence in LRM-246.

Cliff: program-endprogram show definitively that clocking does not exist at the $root level. Have you ever taken a class where all of the coding examples were abbreviated-snippets and then tried to re-read your notes three months later? I have. The examples become very confusing. I don't want to do that to SystemVerilog LRM readers.

15.10 - example comments - misleading??

Example:

```
##5; // wait 5 cycles using the default clocking
##j + 1; // wait j+1 cycles using the default clocking
```

Does this wait for 5 and (j+1) "cycles" or "clocking events" (it would be different for @(clk) and @(posedge clk))??

DWS: The whole section is using clocking events and clock cycles the same here. What is the confusion?

Cliff: hardware engineers think of cycles as a full clock cycle, not a half clock cycle.

15.12 - example synchronization events - Consistency: "clock domain" or "clock-domain?" (with or without hyphen?)

DWS: clock-domain is used consistently in the LRM

Cliff: The first example in 15.12 uses "clock domain," (no hyphen) while the second example uses "clock-domain." Typo.

15.12 - last paragraph - in which event region??

The values used by the synchronization event control are the synchronous values, that is, the values sampled at the corresponding clocking event **which queue??**.

DWS: The term queue is not specified here. The discussion of regions is in 15.13.

Cliff: Sorry, I did not mean queue. But as someone who closely read the description, I was not clear in which region the sampling occurred. Which region is it? Could we state it here for clarity?

15.13 - 2nd paragraph - confusing text needs better explanation

Samples happen immediately (the calling process does not block **what does this mean??**). When a signal appears in an expression, it is replaced by the signal's sampled value, that is, the value that was sampled at the last sampling point.

DWS: It means it does not block.

Cliff: *Cute!* Block what?? If I'm confused, I think someone else will be confused too.

15.14 - 1st paragraph and example - change "slice" to "part-select" (explained earlier - this should be checked globally).

DWS: Likewise see above response.

Cliff: Likewise.

15.14 - Next paragraph - Do we need to specify which event region for clarity??

The second form of the synchronous drive uses the intra-assignment syntax. An intra-assignment event count specification also delays execution of the assignment. In this case the process does not block and the right-hand side expression is evaluated when the statement executes.

DWS: I do not think so.

Cliff: It would help!

15.14.2 - 2nd paragraph after 1st example - Is this statement true??

When the same variable is an output from multiple clocking domains, the last drive determines the value of the variable. This allows a single module to model multi-rate devices, such as a DDR memory, using a different clocking domain to model each active edge. For example:
(Is this true even when the drives happen in the same time-step?)

DWS: I believe it to be correct.

Cliff: Are we certain that different vendors will schedule the same last-assignment? Is this sufficiently defined? Is this a case similar to making two nonblocking assignments in the same time step from different procedural blocks?