Cliff's votes summarized:

See attached document for explanations.

1655 ___ Yes _X_ No

1732 _X_ Yes ___ No

1777 ___ Yes _X_ No

1371 ___ Yes _X_ No (but close?)

1384 _X_ Yes ___ No

1707 ___ Yes _X_ No

1680 _X_ Yes ___ No

1427 _X_ Yes ___ No

1723 ___ Yes _XX No !!!

1736 _X_ Yes ___ No

Cliff's votes:

Please mark your vote below by an x. If No, then specify a reason.
Send it to the reflector.

1655 ___ Yes _X_ No
This is an extensive proposal and one that I would like to better understandby way of discussion in an EC meeting.
For example, I am not convinced that 18.5.1 inferred that default cross bins would be created. They are not created for coverpoints unless the default keyword is used, and what does it mean that "the cross retains those automatically-generated bins ..."?
The proposal is a good starting point but this one should be discussed in a meeting. This is non-trivial.

1732 _X_ Yes ___ No

1777 ___ Yes _X_ No
The clarifications are generally good but I think that there are mistakes in the expanded examples of this proposal.

3 [->3]
is the same as
WAS: ...=>3=>...=>3=>...=>3
(seems to indicate two cycles required between samples and leading cycle before first example)
PROPOSED: 3...=>3...=>3
(shows eventually at least one cycle between samples)

3 [-> 3] => 5
is the same as
WAS: …=>3=>…=>3=>…=>3=>5
PROPOSED: 3…=>3…=>3=>5

3[=2]
is the same as ~~the transitions below excluding the last transition.~~
~~3=>…=>3…=>3~~
WAS: …=>3=>…=>3
PROPOSED: 3…=>3

A nonconsecutive repetition followed by an additional value is represented as follows:
3 [=2] => 6
is the same as
WAS: …=>3=>…=>3=>…=>6
PROPOSED: 3…=>3…=>6

TYPO (missing "the")
WAS: Transition bin b6 and b7 will each increment on 12th sample.
PROPOSED: Transition bin b6 and b7 will each increment on the 12th sample.

the rest of the proposal, including additions to the example and explanatory text all looks good.

1371 ___ Yes _X_ No (but close?)

Friendly amendment (program blocks are not part of a design)
WAS: If there is least one **`initial`** block within at least one **`program`** block in a ~~design~~, ...
PROPOSED: If there is least one **`initial`** block within at least one **`program`** block in a compilation unit, ...

Friendly amendment needed (awkward wording, even after the marked-for-deletion part is removed)
Calling `$exit` from a thread originating in an **`initial`** block of a program shall execute a **`disable fork`** **from within** as well as end all **`initial`** blocks in that **`program`** block.
(The red bold text - what does this mean??)

Is this possible?? If so, please embed an example into the sentence.
... Calling **`$exit`** from a thread that does not originate in an **`initial`** block in a program shall have no effect. ...
(A module cannot call a task in a program that has a $exit - can a module task have a $exit in the task? Where is this documented?)

1384 _X_ Yes ___ No

1707 ___ Yes _X_ No
I don't understand the issue well enough. I would like to discuss this in an EC meeting.

1680 _X_ Yes ___ No

1427 _X_ Yes ___ No

1723 ___ Yes _XX No !!!

I completely disagree! The num method tells me how many items I currently have stored in the associative array. If we add a size method, it should tell me the possible size of the associative array if it were filled.

For example, I would like to declare a bounded sparse associative array and then query the maximum array size with a size method. Something to replace the old Cadence DAMEM PLI code. Something like this:
typedef bit [9:0] addr_range_t;
logic [7:0] damem [addr_range_t];

This would be an associative array of 8-bit words that can be accessed with addresses in the range of 0-1023. Nice bounded associative array for memory modeling. Useful for especially large memories without the need to statically allocate all of the cells.

If I declare an associative array with an enumerated type index, I would like a size method to tell me how many array elements can be accessed and the use num to tell me how many have been written.

If I declare an associative array [*] (unbounded), the size method could return -1.

I think size is NOT the same as num for an associative array.

1736 _X_ Yes ___ No