# Section 12 Issues:

## *Section 12.1*

### (Neil)

- See editor's note. The semaphore and mailbox are being referred to as primitives. They are actually built-in classes. The second paragraph needs to be modified to not use the word "primitive".

  CH-107

## *Section 12.2*

### (Neil)

- Can the semaphore class be extended? Section 12.4.8 explicitly states that a mailbox class can be used as a base class. Section 12.2 needs to state the same thing.

  CH-107

- This section states the following for get() and put()

  - Obtain a key from the bucket: get()
  - Return a key from the bucket: put()

  More precisely they should say:
  "Obtain one or more keys..."
  "Return one or more keys..."

  CH-107

## *Section 12.2.1*

### (Kevin)

It says the key-count can be reset at any time, I think that also works against synthesis. Can we also add an optional maximum-keys value to limit the size of the bucket? That might help trap a few user errors too.

This is not meant to be used with synthesis. It is meant to be unbounded.

## *Section 12.2.3*

### (Neil)

- First sentence says "get() function", it should say "get() method".

- Second to last paragraph, first sentence, says "the task returns", it should say "the method returns".

  CH-107

### (Kevin)

It says "The semaphore waiting queue is First-In First-Out". I think semaphore queuing should not have an enforced order, this is another attempt to create order in the simulator which probably won't be there in the hardware (or won't be required in the hardware). If they need to be ordered you can create a subclass to do it; maybe using a mailbox to order the requests.

> This is not meant to model hardware. It is a synchronization mechanism. Determinism is desirable for synchronization.

## *Section 12.3*

### (Neil)

- Last sentence, says "the task returns", it should say "the method returns".

  CH-107

## *Section 12.4*

### (Neil)

- The num() method is missing from the list at the end of this section.

  CH-107

## *Section 12.4.1*

### (Neil)

- What is returned by new()? The prototype doesn't show the type of the returned value. The text refers to this value as being "the mailbox identifier, or null". I think it should say something like "the mailbox handle, or null".

  The prototype is correct. The prototype for new of all classes is special and doesn't show the type returned (the type is implicit).

  CH-107

- Last sentence of last paragraph says "If bound is non-zero, it represents the size of the mailbox queue." I assume that if bound is less than 0 that a runtime error should be produced.

CH-107

## *Section 12.4.2*

### (Kevin)

I would rather not have this function since there is no guarantee that if you call "num" and then call "get" that there will be no intervening "get" from another thread making the return from "num" obslete - "try_peek/get" seemsufficient.

If "num" is to be supported it should only be usuable while the calling thread has exclusive access to the mailbox - which means adding locking semantics and defining a "critical section" in some way.

Even with these limitations it is still useful. It was even requested at one point.

## *Section 12.4.3*

### (Neil)

- scalar is used in this section. Scalar is used all over the place in this chapter. See sections:

  12.4.3, 12.4.4, 12.4.5, 12.4.6, 12.4.7, 12.4.8

  scalar needs to be changed to singular. The definitions of scalar that are scattered around here should be deleted.

  CH-107

## *Section 12.4.5*

### (Neil)

- Next to last paragraph, first sentence, "Simple mailboxes are type-less..." Why is the word "simple" in there? Shouldn't this word be deleted?

  CH-107

- The same sentence says "a single mailbox can send and receive any type of data", it should say something like "a single mailbox can send and receive different types of data", since there are restrictions on the valid data types for a mailbox (e.g. singular).

  CH-107

### (Kevin)

As with semaphores the last sentence ("The mailbox waiting queue is FIFO") seems like an attempt to order simulation events unnecessarily. If it needs ordering create a subclass to do it.

> This is not meant to model hardware. It is a synchronization mechanism. Determinism is desirable for synchronization.

## *Section 12.4.6*

### (Neil)

- Last paragraph mentions that "the function returns", this should be replaced with "the method returns". There are 3 places to fix this in this paragraph.

  CH-107

## *Section 12.4.8*

### (Neil)

- Same comment as 12.4.6 (3 places where "function" needs to be changed to "method").

  CH-107

- Editor's note mentions section 19.4, it meant to say 12.4.

  CH-107

## *Section 12.5*

### (Neil)

- There is an inconsistency in the write-up as to whether there is a runtime error or a compiler error when there is a type mismatch. See first sentence, page 92 and the last paragraph of this section.

  The text is correct. When the mailbox is not parameterized then it uses the special ***dynamic_type*** that provides only runtime checking. If, however, the class is parameterized then it enables compile-tome checking,

- The first sentence on page 92 refers to run-time type-checking as "the default". This implies that there is a way to turn this off, is there a way to do this? If not, drop the part that refers to this as the default.

  This is the same as last item. The default mailbox is not parameterized and provides only for runtime type-checking. It can be turned off by parameterizing the mailbox.

- Last paragraph mentions that the compiler checks the type of data passed to put() and get(). What about peek()?

  CH-107

## *Section 12.6*

### (Neil)

- First paragraph, last sentence, mentions that events can be "dynamically allocated and reclaimed". This seems to be discussing implementation specifics. The user doesn't actually "allocate" an event does he?

  The user indirectly allocates an event when he declares an event in a dynamic context (a class or an automatic task/function).

- First paragraph, mentions twice about events that: "they can be arguments to tasks and that they can be dynamically allocated and reclaimed." This only needs to be stated once.

  CH-107

### (Kevin)

Still think ".active" is a better solution, these semantics and syntax will make it difficult for users reading other peoples' code. The $wait.. functions should be replaced by syntax using @.

> There is a proposal discussing all of the solutions being created by Stefen Boyd. This will be discussed and we will vote on which we like.

## *Section 12.6.1*

### (Neil)

- Last sentence, page 95, says that an edge triggers a latch. I think you meant to say that an edge triggers a flip-flop.

  CH-107

## *Section 12.6.4*

### (Neil)

- Page 97, second paragraph below the editor's note, the second sentence in this paragraph seems to contradict the first sentence.

"Both mechanisms can be used to wait for either a persistent or a non-persistent event. The wait construct is only meaningful when the event is persistent."

The second sentence seems to be incorrect.

There is no contradiction. Both @ and wait can be used with either type of event. But wait and @ behave exactly the same way when the event is not-persistent, thus, it is only meaningful when the event is persistent.

- Next to last paragraph, first sentence uses the phrase "generic task". This is an old Vera phrase. Shouldn't it just say task?

  It says generic because the task can be used to trigger any arbitrary event.

- I also have the same comment on the last paragraph that Stefen made this morning about the order of execution of the statements within a fork/join construct not being defined.

  CH-107

### (Stefen)

- Since CH-45 removed the requirement of a particular execution order in a fork/join, we need to change the first sentence of the last paragraph from stating that "the first process triggers the event blast" to "the first process may trigger the event blast"

  CH-107

### (Various)

Lots of discussion on event.active, NB event operator ->>, @, event bit, etc…

## *Section 12.7*

(Stefen)
This was already floating around, but wanted to raise it for discussion on Monday.

I would like to see $wait_xxx calls changed to use event control operators. This was discussed heavily, so I'll try to summarize what seemed to be the desired change:

```
$wait_any(a,b)      @( a or b)     // nothing new to create
$wait_all(a,b)      @( a and b)    // or always begged for an and
$wait_order(a, b)   @( a ; b)      // use operator from assertions
```

I'm using the ';' operator from the assertions, but use the meaning as defined for $wait_order. Unless the assertions change sequence operator...

> The assertions use ';' to indicate a clocked sequence, which this is not. The semantics of wait_order are very different because it stipulates exact ordering and

no repeat of events except in that order. For this reasons we shouldn't even consider the ';' syntax. Another alternative may be "wait seq( a, b );"

With these changes, 12.7 should be removed and 12.6.4 should be split into two sections. One for @ and one for wait (see separate email for 12.6.4 - wait)

Yes. provided we agree to those changes.

## *Section 12.8.1*

### (Neil)

- Should "null" always be in bold. There are a few places in this chapter where it isn't.

CH-107

### (Kevin)

SV regular events shouldn't be different to Verilog, they should have the same instantiation semantics. If you want to assign events to events you should use references instead E.g.:

```
event foo;
ref event bar = foo;
                        ...
@(bar) // wait for foo
bar = null;
@(bar) // doesn't block

foo = null; // illegal
```

Among other issues the ref support cannot be used this way. It is restricted to use on declarations. Pointers (or anything like them) are not supported within the language.

## *Section 12.8.2*

### (Kevin)

Is this necessary?

Need a mechanism to be able to assign events. This is it.

## *Section 12.9*

### (Neil)

- scalar is used here. (also on page 25)

CH-107

- Last paragraph, page 99, "Arguments to $wait_var() can be an array subscript expressions..." Either drop the s at the end of expressions or get rid of the "an" before array.

  CH-107

**(Kevin)**

What's wrong with "@(<variable> or <..." ?

> There is a proposal discussing all of the solutions being created by Stefen Boyd. This will be discussed and we will vote on which we like.