ERR-23

3.12 Class

- "An object can be declared as an argument input, output, inout, or~~,~~ ref."
- input, outputs etc. are not types!
- The comma after "or" is not needed.

Agreed. A correction in errata proposal is being prepared.

ERR-24

4.2 Packed and unpacked array + 4.6 Dynamic array + 4.9 Associative arrays

- Can an unpacked array be declared with a basic type "dynamic array" etc.? Yes.

```
typedef int int_a [8];
int_a k[];
int_a l[int];

typedef int int_da [];
int_da m[8];
int_da n[];
int_da o[int];

typedef int int_aa [int];
int_aa p[8];
int_aa q[];
int_aa r[int];
```

- "SystemVerilog accepts a single number (not a range) to specify the size of an unpacked array, like C."
- Is this possible for unpacked arrays, only? Why not for packed arrays?

The primary reason for providing the "short cut" of using a single number for the range was for convenience to those familiar with C (thereby the definition of the number represents the range from 0 to N). This was not extended to packed data types because for packed types the index direction is very important and a very common usage is to specify a range N:0, which is backwards from the C notation. Defining the order or the range with only the size specified differently for packed and unpacked would be confusing. Defining the order for packed arrays to be the same as for unpacked (and thereby 0:N) was considered confusing. To minimize the confusion (and possible unexpected behavior) the short cut was only defined for unpacked arrays.

ERR-25

5.3 Constants

- If a constant instance of a class references another object, can this object be changed?

Since the constant is the reference and not the class itself the contents of the class can be changed (except for those members that are declared **const**). In order to clarify this in the LRM an errata proposal is being prepared.

ERR-26

10.5.2

- If an instance of a class (used as “const ref” parameter) references inside another object, can this object be changed in the subroutine?

The const ref here refers to the ability to pass a reference to an object handle which cannot be changed. This is essentially the same as item 5.3 above.

ERR-27

11.3 Overview + 17.10 The property definition

- The term “property” is introduced twice into SystemVerilog for two different things!!

Agreed. The term is also used in a third way within the LRM that does not refer to class properties or to the **property** keyword. An errata proposal is being prepared.

ERR-28

11.7 Constructors

- Semantic of the following example should be defined.

```
class A ;
 integer j = 5;
  function new();
    j = 3;
  endfunction
endclass
```

Agreed. An errata proposal is being prepared.

ERR-29

11.8 Class properties

- It should be mentioned that class properties can be used without any instance as in the example from the LRM.

```
Packet p;
c = $fgetc(p.fileID );
```

Agreed. an errata proposal is being prepared.

ERR-30

11.9 Static methods

- Different semantic of “static task” and “task static” is very confusing.

Yes, this can be confusing. In a future version it might be considered to use another word (such as 'unique task' as an alias for 'static task'). The other option might be to disallow 'task static' for class methods. No errata proposal is being considered at this point.

ERR-31

11.19 Abstract classes

- I suggest to change the grammar that there is no “endfunction” in the example:

```
virtual class BasePacket;
virtual protected function integer send(bit[31:0] data);
endfunction
endclass
```

Agreed. An errata proposal is being prepared.

ERR-32

11.19 Abstract classes

- According to the LRM, non-abstract classes can have virtual methods (which must have a body). What is the difference to a “normal,” non-virtual methods? If there is none, they are not needed. As I know now:

  Virtual methods are a basic polymorphic construct. A virtual method overrides a method in all the base classes, whereas a normal method only overrides a method in that class and its descendants. One way to view this is that there is only one implementation of a virtual method per class hierarchy, and it is always the one in the latest derived class.

This is a very important info and should be inserted in the LRM in any case!

Agreed. An errata proposal is being prepared.

ERR-33

11.20 Polymorphism

- The method call “send” in the example can not be used if it is assumed a method from the example in 11.18 is called because they are protected. If there is no connection between the examples of the subsections, then the classes must be defined.
- Otherwise, the names of the classes are identical with the classes in the previous subsection. In addition they all have the method “send”, so every reader would assume that the used classes “BasePacket”, “EtherPacket” are the classes defined in the subsection before. But here “send” is defined as “protected”. So the only change, which is

needed, is the removing of the keyword “protected” in the declaration of “send” in 11.19.

Agreed. An errata proposal is being prepared.

ERR-34

11.22 Out-of-block declaration (please note the orthography)

- The location, where the out-of-block declarations have to be, must be declared.

Agreed. An errata proposal is being prepared.

ERR-35

12.5.2 Random Constraints

1. Only here the term “overload” is used (twice). Does this means the same as “override”? Definition is needed!!

Agreed. An errata proposal is being prepared.

ERR-36

13.2.1 new() (Semaphores)

- One sensible enhancement would be to define a maximal number of keys in the semaphores to avoid programming mistakes, which can happen very easily (e.g., a process puts two keys back instead of one so that two processes can get a key).

A suggested way to handle this is to create a derived class that implements the desired functionality. This is one of the advantages of defining Semaphores as a class.

ERR-37

13.2.3 get() (Semaphores)

2. “If the specified number of key are not available, the process blocks until the keys become available.”
3. It should be defined, when exactly get() will continue after blocking. In the same time step ….

It unblocks as any other event control waiting on a signal. No errata proposal is being prepared.

ERR-38

14.2 Event simulation

- Maybe, the special handling of always_comb etc. should be mentioned here.

Agreed. An errata proposal is being prepared.

ERR-39

14.3.1 The SystemVerilog simulation reference algorithm

- Initialization of consts are missing
- What is the initialization order in case of dependent initialization?

The missing item here is that **consts** are variables and are handled the same. The ordering issue exists for variables that aren't **const** as well. An errata proposal is being prepared.

ERR-40

19.5.3 An example of exporting tasks and functions
The tasks Read and Write are used (declared, defined, called) four times each. Sometimes they have one parameter, sometimes none. I assume that the slave defines the tasks and made them available via "export". It seems, the master want to use exactly these tasks because the tasks with parameters are not defined in the interface, and in the master module (cpuMod), the comment describes explicitly the use of the slave method--but the call is made with a parameter!!

The root of this confusion appears to stem from the fact that exported tasks do not require complete prototype (as defined in pg. 213 In a modport, the import and export constructs can either use task or function prototypes or use just the identifiers. The only exception is when a modport is used to import a function or task from another module, in which case a full prototype shall be used.). This can result in the same task being declared in different ways depending on the context.