SV-CC Face To Face Meeting notes - 10/16-17/2006

Terminology/language/glossary

We need to define a common language for working on the documents.

Decompilation is mostly used to make sure we have everything covered. Not really a common use model by customers.

Francoise described how things are done in VHPI. Chas likes the idea of doing the same thing as what was done in VHPI. Francoise thinks it is not so easy since the Verilog language is not as clearly defined.

It was proposed that if vpi_compare_objects() returns true, then there are no transitions which would give you different answers.

For a declaration of something like:
logic MDA[4:5][6:7][1:0]
What is the difference between MDA[4] and MDA[4][6:7]? Both are slices. First implies the second set of dimensions. What about MDA[4:5]?

Differentiate between changes caused by time, and changes caused by the creation/destruction of an object.

JimV wants to differentiate between what is in the HDL vs. iterations or other access done only in the VPI application.

Things like vpi_compare_objects() does it compare two VPI objects, or does it compare two design objects?

Terms:

Elaboration invariant	Statically elaborated objects
Source code invariant	
Dynamic expressions	
Temporal dependencies	Michael threw this one out
Slices	Talking about something with two bounds.
Dices	Pieces of slices?
Transition point	The change when you go from packed to unpacked dimensions.
PU boundary	The same as transition point
Canonical	

Names

Chuck took a crack at defining the differences between name and decompile. *Chuck needs to send out.*

Proposals:

- Var selects:
  1. Variable index vs. constant expression
  2. expr in source vs. selection by VPI operation
  3. static during lifetime vs. moving in memory
- We need a new object types for slices. Has to have two bounds. The language definition is in section 5.4. Part select is for bits; slice is for vectors. Part select should be for a set of bits, regardless of the number of dimensions. *This needs to be fixed in section 5.4.*
- vpiSlice and vpiIndexedSlice has the following properties/transitions:
  1. Parent could be a vpiArrayVar, packed-array type, or member or either one that is the immediate dimensioned array (prefix expression). I.e. a[1][2:3] -> a[1]
  2. The parent may be a var select. I.e. a[j][2:3]-> a[j]
  3. Have to be able to iterate on the ranges?
  4. Parent can be anything in the variables diagram.

JimV will take on the vpiSlice problem!

From Jim:

A slice (5.4) for which the range belongs to the right most packed dimension of an array shall be represented as a vpiPartSelect or vpiIndexedPartSelect. Any other slice shall be represented as a vpiSlice or vpiIndexedSlice.

For example, consider the declaration

```
reg [7:0][1:8] vec [1:3][0:7];
```

Then the expression "vec[1][0][7][1:4]" is a vpiPartSelect, while the expression "vec[1][0][7][I +: 2]" is a vpiIndexedPartSelect. Similarly, the expression "vec[1][0][1:4]" is a vpiSlice, while "vect[3][i +: 2]" is a vpiIndexedSlice.
end of stuff from Jim.

On the typespec page: bit, logic, struct, and union have multiple dimensions and therefore have a sub-element transition.

Most end users are not going to get the subtle differences between packed and unpacked objects.

Consider the following:

```
struct packed {
  int m1;
  byte m2;
} [65:53][3:0] s1;
```

s1[65][0][7:0] -> is a vpiPartSelect
s1[65][2:0] -> is a vpiSlice
s1[65] -> is a vpiBitVar (if the struct were mixed 2/4 state, it would be vpiLogicVar).

Consider the following:

```
struct packed {
  reg[7:0][4:0]vec;
  int m1;
  byte m2;
} [65:53][3:0] s1;
```

s1[65][0]vec[3][4:2] -> is a vpiPartSelect
s1[65][3]vec[3:2] -> is a vpiSlice

Consider the following:

```
module foo;
  reg[0:1][2:3] MDA [4:6][6:8];
```

MDA[6][8][1][3] -> is a vpiBitVar
vpiParent would return?
MDA[6][8][1]?
MDA[6][8]?
In 1364-2001 would require it to be the latter. We would prefer the former.
Should we have a “vpiPrefix” that would do the former?

Should make vpiParent un-ambiguous.

vpiParent on the variables, nets, and parameters diagram, shall return the higher level object, stopping at the first of:

- The struct, union, or class structure
- The struct, union, or class member
- The largest containing packed array object
- The largest containing unpacked array object

Chuck will take on vpiParent problem.

Make a LHS that points to objects inside a class object be vpiVarSelect?

End of Monday

Tuesday 10/17/2006

Type parameters discussion

```
module m;
    class C # (parameter types Ta = real);
    endclass
    C #(int) vc = new;
    parameter p = 3;
    parameter type T        = logic;
                            = bit[$bits(P):0];
                            = type(P);
```

Can't we just add another transition to typespec that would be for the original definition? Then add a few notes that would define what properties and transitions work for type parameters.

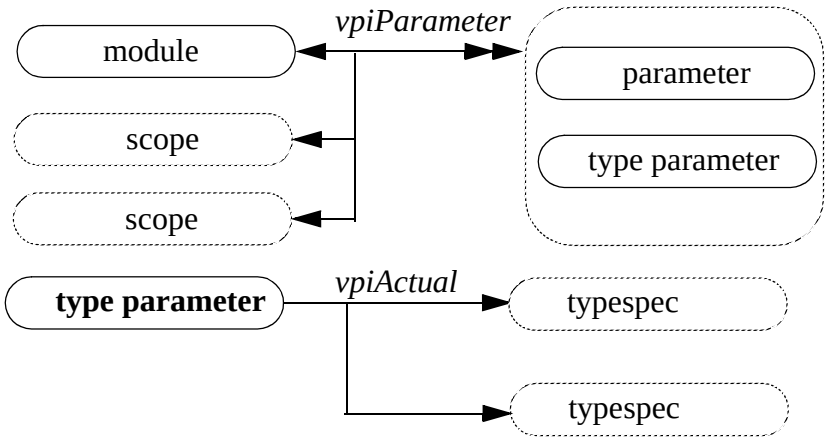
Problem with names for statics in typespecs for parameters.

Problem on 27.17: From a class typespec there is no way to get to the class defn

Problem on 27.17: The vpiName property is undefined for class typespecs.

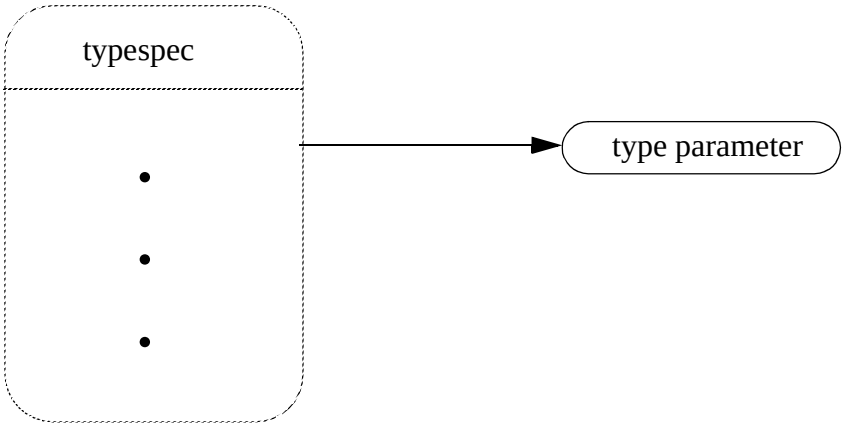
Chuck wants to add two new properties vpiArrayDim and vpiPackedDim.

Francoise's suggests changing 27.20 as follows:



Note: vpiActual returns the resolved typespec for the parameter. vpiTypespec returns the typespec that was in the definition.

And add to 27.27:



Note: If the typespec was defined in terms of a type parameter, then the transition to type parameter is defined, otherwise it returns NULL.

After some discussion, we decided having a full name to a typespec was not a good idea. Probably was not added because you generally cannot do it in the language. This was likely because of the extra effort it would add to the elaboration process.

Francoise will take a crack at putting this one through the committee.

parameterized classes and VPI

Main discussion came down to a naming issue. No names, generated names, or that the name is some convoluted thing which already exists (but is ugly). Abi will talk with others who know this subject better.

class type parameters discussion

Long discussion that I did not follow: *Ask Francoise*.

Discussion on comparing objects

Consider the following example:

```
const variable j = 1;
struct packed {
    reg [0:7] vec;
} ps;

    ps[j - 1] = ...;
```

ps.vec[7] = ...;

The values for ps[j - 1] and ps.vec[7] are the same.

We should take an action to fix vpi_compare_objects(). JimV's proposal: Same VPI type, any property will return same result (except file and lineno), strcmp or int compares are same. For any transition from the two objects would return a set of handles which would also be true using vpi_compare_objects().
Clearly compare objects does not work for some situations:

Possibility of a strict definition for compare objects

Possibility of creating a compare objects that decides if two objects point at the same location in memory at this time.

Abi will try to write up a discussion paper on this.

Compatibility issues between the different version of the spec (1385)

Uniquely muddled hybrid solutions (a Ralph-ism) adequately describe's the situation.
Abi is concerned about how we might write a vendor tool that is independent of the state of the application tool(s), i.e. how does it deal with an application that is not up to date with the tool?
Michael is concerned about the tools being up to date with each other. It also could mean that the applications cannot upgrade until all of the tools are updated.

What if you had version specific include files that the application could call? We would then be able to tell what the application was looking for when it made the call.

We may have a new potential solution. Bassam thinks we should look at the reader solution which might be another way to do the same thing. You could do this with a function tray, and it would be the application's responsibility to set up the correct function tray.

Chuck sent out writeup on techniques used to allow VPI applications to be coded in such a way as to be immune (standard-neutral) to the compatibility issues identified in 1385..

Action: Have Steve Dovich advise us on the best way to deal with this as far as the IEEE is concerned.

Michael took an action to come up with a well thought out proposal for the next SV-CC meeting.

Classes Discussion (continued)

Make LHS that point to objects inside a class object be vpiVarSelect?

A.i[0];

A is a classvar.
i is an int arrayvar
Is the whole thing an intVar, or a varselect?

Class variable never points to garbage. It either points at a class obj or NULL.
A class obj in the VPI model has an independent existence until the VPI application calls vpi_free_object().

Added two mantis items: 1631 and 1632.

Is var select defined at the language level? Bassam thinks there was a discussion on this. Looking for it.

For an indexed variable or net array, if the index(es) relative to it's parent are variable, then the object is a vpiVarSelect. Otherwise it will be a variable or net of the appropriate type. Note: we need to look at the net situation more. JimV will take a crack at this one.

Organization of the document
Jim thinks that the VPI extensions for coverage is so vague that we would really need some assistance from someone who knows coverage. We should only try to incorporate section 28 into the combined document. Sections 29 and 30 should remain separate because of the weaknesses in those sections. Group generally prefers the overall format of 1364. Add some of the stuff from the other sections.