SV-AC Feedback to Working group on the next SystemVerilog PAR

- Thomas Thatcher
  - Staff Engineer
- Oracle

Overview

- Relax restrictions on Checkers
- Concurrent assertion enhancements, local variables.
- Type enhancements
- Introduce temporal coverage
- New system functions
- Variable number of arguments to checkers, modules, interfaces, tasks, functions, etc
- AMS Extensions

Checker Overview

- A **`checker`** encapsulates verification code
  - Used to create assertion libraries
  - Encapsulate assertion with coverage
  - More flexible instantiation rules than a module.
    - Instantiate within a procedure,
    - Goal was to make a checker as flexible as an assertion
  - Introduced in SystemVerilog 2009
    - Several restrictions put in place
    - Concerns over implications for the type system, and not enough time to resolve them

Proposed Checker Enhancements

- Relax restrictions placed on checkers
  - > Allow parameters (Mantis 2111)
    - – Makes it easier to implement existing libraries using checkers
  - > Allow procedural control and looping statements
  - > Allow continuous assignments
  - > Allow output arguments (Mantis 2093)
    - – allow a checker to trigger another checker, assertion, or cover point
  - > Clarify that $display is allowed inside a checker (Mantis 2897)
- Will need to resolve type-system issues with above changes

Other checker enhancements

- Forcing design variables from checkers
  - > Make the verification environment more flexible.
- Clarify checker argument sampling

Concurrent Assertion Enhancements

- Expand scopes where concurrent assertions are allowed
  - > Extend language to allow concurrent assertions within functions, tasks, and classes.
  - > Allow checkers in functions, tasks, and classes.
- Allow inline local variable definitions

```
assert property (
   start |->
   (  dataType l_data = dataIn;
      tagType l_tag = tagIn;
      (complete && tagOut == l_tag)[->1]
      |-> dataOut l_data )  );
```

- Allow ##<variable_name> to specify a variable number of cycle delays.

Type enhancements

- Generic integral type

```
checker c_before (sequence en, integral a, b, logic clk);
    A1: assert property (@(posedge clk) en |-> !y[+] ##0 x);
endchecker

// integer i1; bit b1; logic [3:0] g3; sequence s1; property
p1
c_implication c1 (s1, i1, b1, clk);    // legal
c_implication c2(s1, g3, b1, clk); // legal
c_implication_c3(p1, s1, p1, clk); // illegal
```

  - > Move error messages to the checker interface
- New system function that reports whether two types are cast-compatible.

Introduce Temporal Coverage

- Allow assertion constructs inside covergroups

```
covergroup g @(posedge clk);
  A: coverpoint a { bins a[] = {1, 2};  }
  B: coverpoint b { bins b[] = {[1:3]}; }
  cp: cover property (trig #-# A[*2] ##1 B[*2]);
endgroup : g
```

- Alternate method would be to use the action block of a cover property to call the sample() function of a covergroup.
- Allow for binning over delay ranges
  - e.g. a ##[3:7] b: Generate bins for each latency.

System Functions

- $onehot(), $onehot0(), $countones(), $isunknown()
  - Clarify that these functions can accept an unpacked array as an argument
- Create 4-state versions of above functions
  - Include handling of X and Z

Variable Number of arguments

- Allow for checkers, modules, interfaces, program,s functions, and tasks to have a variable number of arguments.

```
checker same (a…);
  if (a.size < 2) $error(“a must have >= 2 elements”);
  assert #0 ((a[1:a.size].apply with
      (item == a[0]).reduce
        with (item1 && item2));
endchecker : same2
```

  - Desired for Checkers, but extendable to other constructs

AMS Extensions

- Allow real expressions in assertions.

```
assert property (
  @(posedge clk)
      Ain > 4.12m );
```