Verification

SV3.1a Assertions Donations

Surrendra Dudani
Sept. 18, 2003

Synopsys, Inc.

Assertions Submissions (SV-AC)

- NBA assignments for assertions
- Property assumptions
- Property assumptions with biasing
- Sequential constraints
- Access to sampled values
- Access to gated registers
- Parameterizing unbounded range
- Improved messaging

NBA Assignment For Assertions

variable_lvalue <= expression

- Needed to allow clock based assignments using temporal functions
- Assignment can only appear inside a clocking domain
- Variable declared outside the clocking domain
- Assignment evaluated only at clock ticks, and updated after the observe region
- Expression can include temporal functions such $past, $rose, ended, etc.

Property Assumptions

assume property (property_spec) ;

- Allow properties to be considered as assumptions for formal and dynamic simulation
- Tools must constrain the environment such that a property holds
- No obligation for tools to prove the property
- Assumptions may be used as embedded or as declarative

Property Assumptions with biasing

assertion_expression ::= expression |
expression *dist* { dist_list }

- Needed for driving random simulation
- Biasing provides a weighted choice to select the value for free variables
- Alignment with constraint block syntax
- For assertions and formal, *dist* converted to *inside*
- Properties must hold with or without biasing

Sequential Constraints in Constraint Block

constriant_block ::= property_instance ; |
…..

- Extend constraint block to allow property instances
- Properties become assumptions, and automatically create boolean constraints
- Boolean constraints from properties solved with other boolean constraints, whenever randomize function called
- Procedural control over randomization as before

Access To Sampled Values

$sampled (expression [, event_expression])

- Needed for debugging and reporting
- Can be used anywhere, the most recent sampled value returned
- event_expression can be omitted if used in action blocks or under a default clock
- event_expression takes prcedence, if specified

Access To Gated Register Values

$past (expression1 [, no_of_clocks] [, expression2])

- Requires complicated modeling when some registers are gated, while assertion clock is not gated
- Extend $past definition to allow gating expression
- If expression2 specified, returns the value on a clock tick that satisfies expression2
- If expression2 not specified, semantics are same as before

Passing unbounded range as a parameter

- $ used as a symbol for unbounded range [const:$]
- Re-use of assertion difficult, as duplicate assertions required for same behavior
- Proposal to allow $ as a System Verilog parameter value of integer type
- Allow $ as an actual argument to properties and sequences
- System function $isunbounded(const_expr), return if the argument is $

Improved messaging

$error (), $info(), $warning(), $fatal()

- Needed for debugging and reporting
- Functions that return true
  - Can be included inside any property expression
- Allow error message to be bound to a property