Automating sequence creation from a microarchitecture specification

Subramoni Parameswaran, Ravi Ram
2100 Logic drive, San Jose, CA 95124

Objectives
- Create stability and flexibility in the stimulus generation sequence classes
- Reduce footprint of testbench change required
- Automate sequence/test creation
- Improve ability to absorb modified verification requirements

Microarchitecture Specification
- Read request signals
  - clock
  - enable1
  - enable2
  - read_addr
  - read_req
- Microarchitecture Assumptions/Specification
  - enable1 before/after enable2
  - enable2 before/after read_addr
  - read_addr before/after read_req

Randomization driven by test – (2)
Move sequence configuration into a sequence configuration class that can be modified by the test writer at run time
Same sequence can be reused by multiple directed tests, reducing changes required in underlying sequences
Use the factory to override the sequence configuration class type in some or all tests if desired

Randomization in the sequences – (1)
Introduce random delays between signal transitions in the sequence
Use (min, max) parameter ranges to constrain delays

Ultimate sequence flexibility -(3)
Redefine the sequence configuration parameters as an executable microarchitecture specification for the entire sequence
Define and use a signal generator class that can be shared by all sequences to automate the creation of sequences

Conclusion
Final approach used successfully as part of shared VIP
Stable and flexible sequences can be created using UVM configuration data base
Test writer can create targeted random tests without changing underlying sequences
Sequences/signals can share code using signal generator class
Executable microarchitecture specification and shared signal generator class can be used to generate sequences