Common UVM Register Model Issues and Pitfalls

Uwe Simm <uwes@cadence.com>
DVClub
online
Feb 2019

cadence®

Presentation Focus

- UVMREG one of the major UVM facilities used in numerous projects
  - UVMREG is ~1/2 of UVM (36kloc code vs 16kloc)
- UVMREG usually fine with 'standard'(=byte|word addressing, 'static' addressing scheme), 'module-level' usage

- **However various issues, drawbacks are known on subsystem, interconnect and system level users should be aware of**
- Presentation is not a full list but should give you some heads-up
- concrete issues filed here: https://accellera.mantishub.io/view_all_bug_page.php

Structural/Conceptual Issues

- not really data-path aware, currently simplistic static, hierarchical address model
  - no direct remap support
  - does not support bitSteering/(un)packed addressing
  - In general full IPXACT to UVM mapping
- memory and compile intense especially for (sub)system
  - every reg, map is a class type deriving from uvm_object
  - fields are class instances (derived from a ‘fat’ uvm_object)
- register access arbitration is at the register level (highest) and not on the bus
  - cannot access the same reg from different points at the same time and let the bus arbitrate
- no clear separation of instances (block+regs) = 'containment' and addressing (map) *several issues and API limitations*
- no general separation of view (interpretation) and data storage for registers (mode dependent registers, fields)
- no general 'indexing' capability (single reg indexing set of regs only)

SoC/Interconnect Scenario Pitfalls

- No dynamic data routing model in contrast for instance to IPXACT
  - Simplistic register model generation done by most generators doesn't work anymore
- a map cannot be the child of multiple maps

```
UVM_ERROR : reporter [RegModel] Map 'm_reg.ce_core.default_map' is
already a child of map 'm_reg.mapa'. Cannot also be a child of map
'm_reg.mapb'
```

- cannot have a register added multiple times to the same map to different address
- cannot connect maps from different blocks
- addressing is broken for layered maps (<UVM-IEEE) especially when map properties change from map to map (n_bytes, bitSteering/(un)packed addressing, endianness) and/or multiple data routes to a register exist

Other Limitations

- addressUnitBits=k*8
- memory word width == width of enclosing map
  - k addresses must fit exactly into one memory word
  - one map address must cover k memory addresses
- .lock_model issues
  - .lock_model isn't adding any functionality but the address caching but prevents several things
  - bad checks imposing addon constraints
- registers cannot be at the same address although their fields do not overlap
  - exception is a RO+WO reg at the same address
- wide memories with a smaller data-path width than bus

Some Unexpected

- mirror() conflicts with explicit prediction or pipelined BFM
- mirror(check) collides with the register maps check setting
- various checks wrong with reg, map, block containment e.g.
  - `UVM_WARNING /home/uwes/uvm/src/reg/uvm_reg_map.svh(1633) @ 0: reporter [RegModel] In map 'block0.submap0' register 'block0.rfile0.ureg2' maps to same address as register 'block0.rfile0.ureg2': 'h20200`
  - `UVM_WARNING UVM/sv/src/reg/uvm_reg.svh(1075) @ 92000: reporter [RegModel] Register 'uvm_reg_model. i_csi2_ip.regs_blk.cfg' is not registered with any map`
  - `UVM_ERROR @ 1000: reporter [RegModel] Submap ‘a.b.c’ may not be added to this ‘b.c’, as the submap’s parent block ‘c’, is not a child of this map’s parent block ‘d’`

Summary

- UVMREG is good for ‘standard’, ’module-level’ use models

- (sub)system use models can be limited or may require expensive workarounds especially when
  - Multiple data routes
  - Dynamic data routes
  - Data routes involving addressing/property changes
- Several bugs, enhancements open with IEEE and UVM-WG to address the points. Likely not just ‘cosmetic’ changes

Q&A