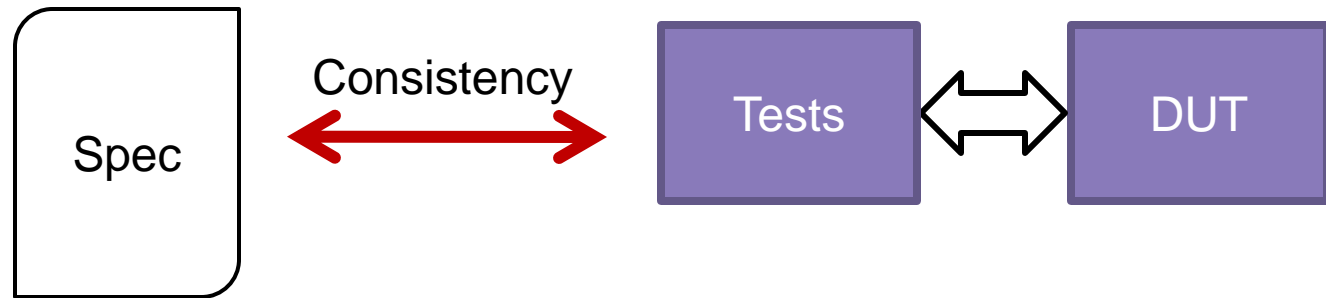


Structured Verification with UVM Register Abstraction

Janick Bergeron
Synopsys Fellow

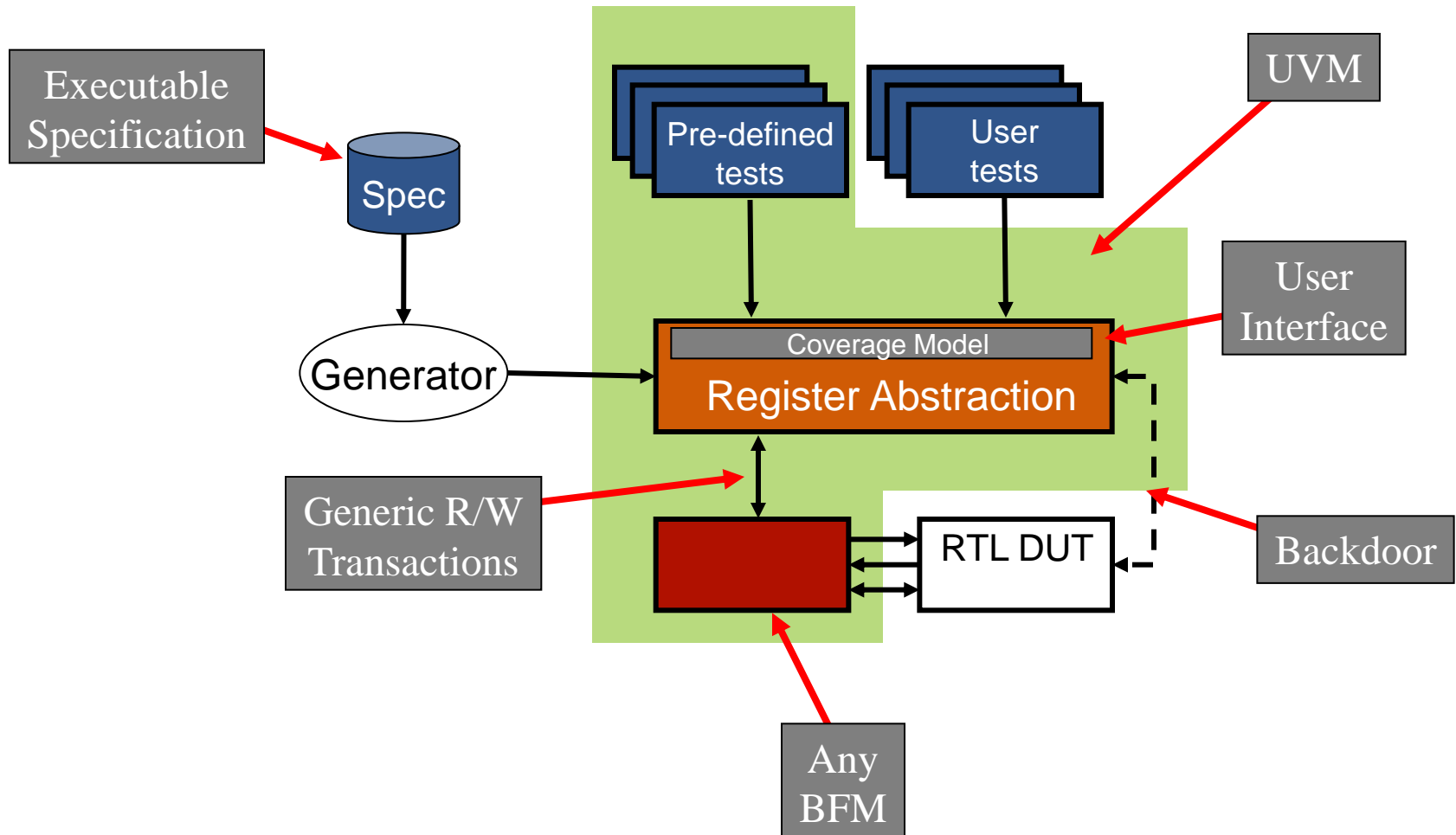
Register Verification

- High-maintenance
 - Modify tests
 - Modify firmware model



- Automation ensures consistency
 - Correct-by-construction

Register Abstraction Architecture



Register Abstraction Layer

- **Instead of:**

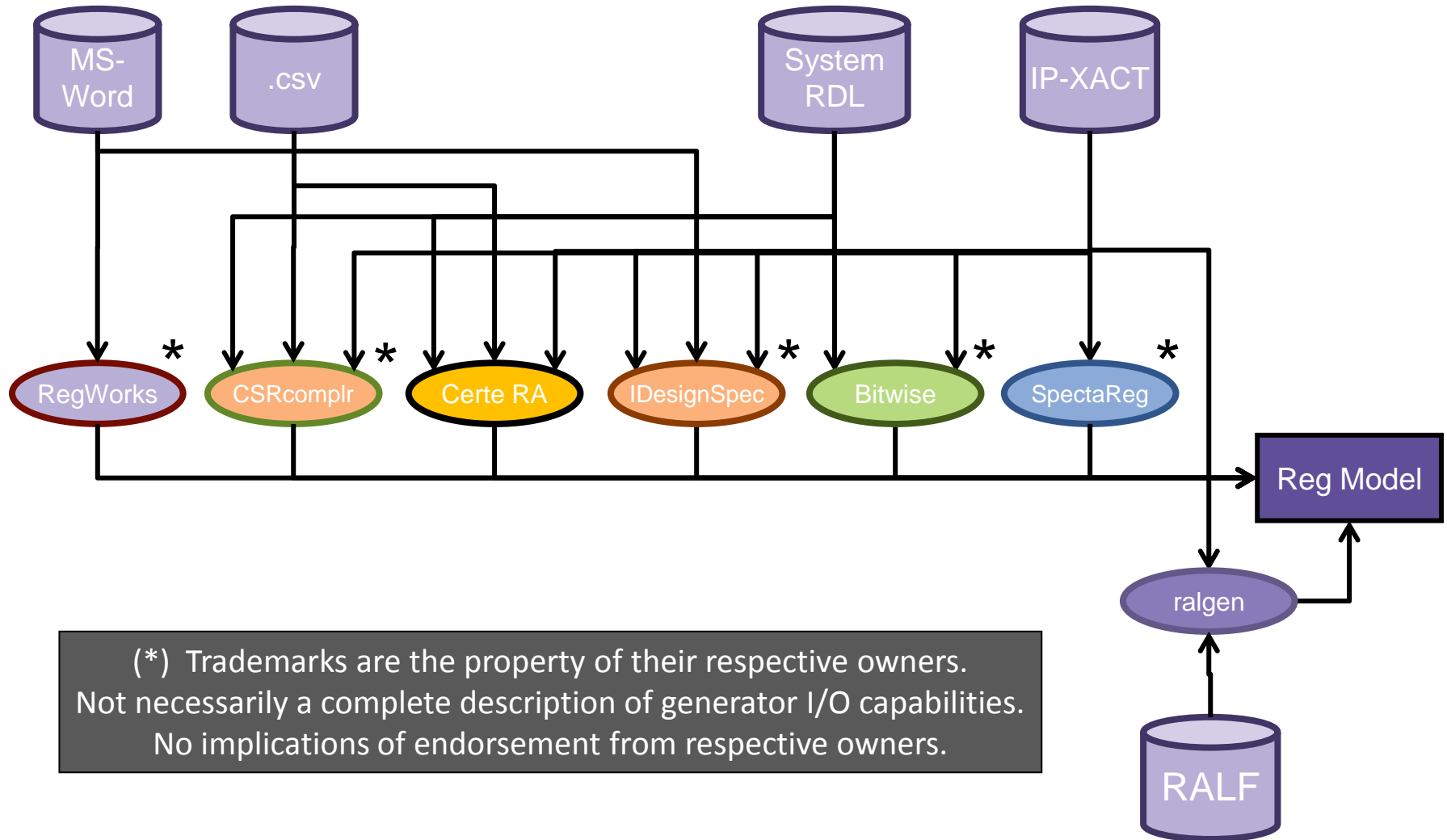
```
env.cpu.read(`IP0_STATUS_REG_ADDR, status);  
status[`IP0_STATUS_REG_FOO] = 0;  
env.cpu.write(`IP0_STATUS_REG_ADDR, status);
```
- **Use:**

```
reg_model.ip0.foo.write(0);
```
- **No more...**
 - ... magic address
 - ... magic bit offset
 - ... writing register tests

Register Abstraction Capabilities

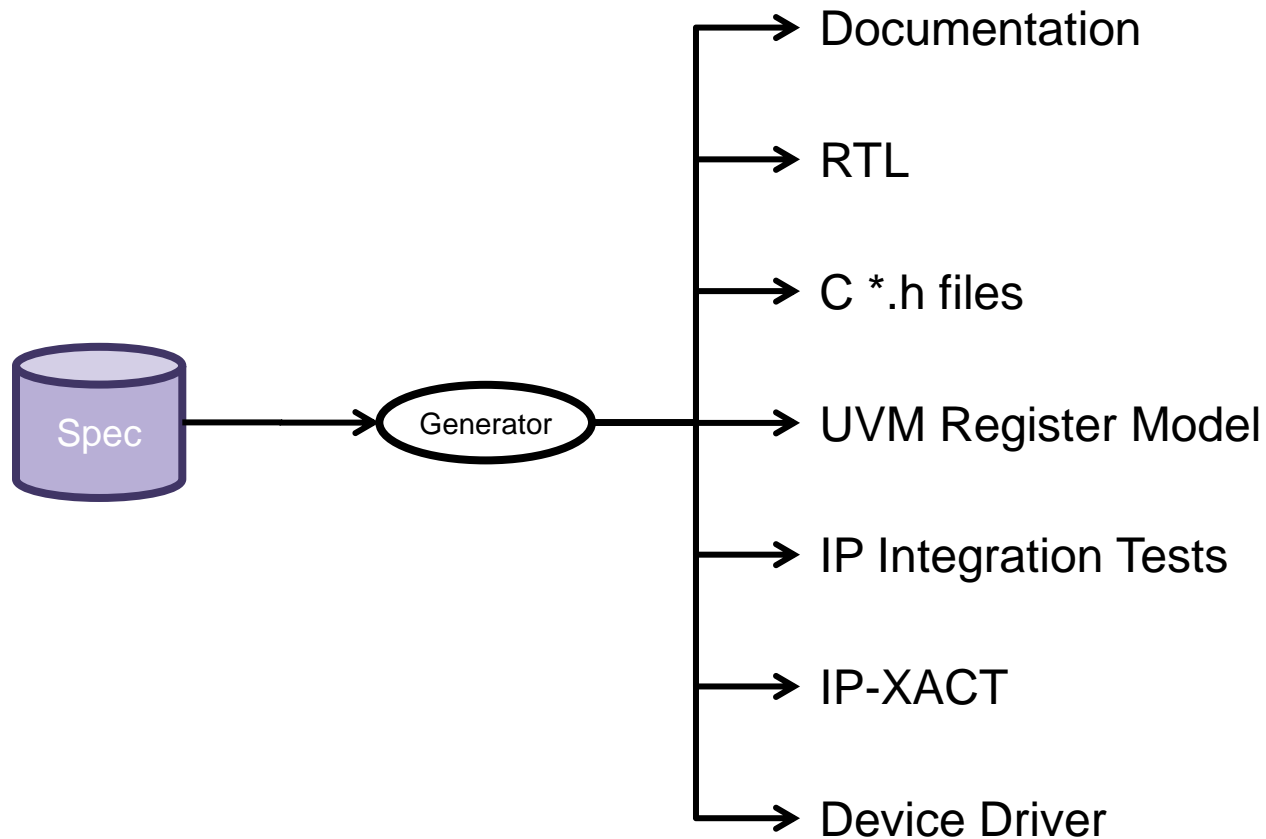
- Pre-defined register tests
- Model content randomization and constraints
- Automatic mirroring
- User-defined scoreboarding
- Extensible for user-defined behavior
- Vertical reuse of driver model

Generation Flow



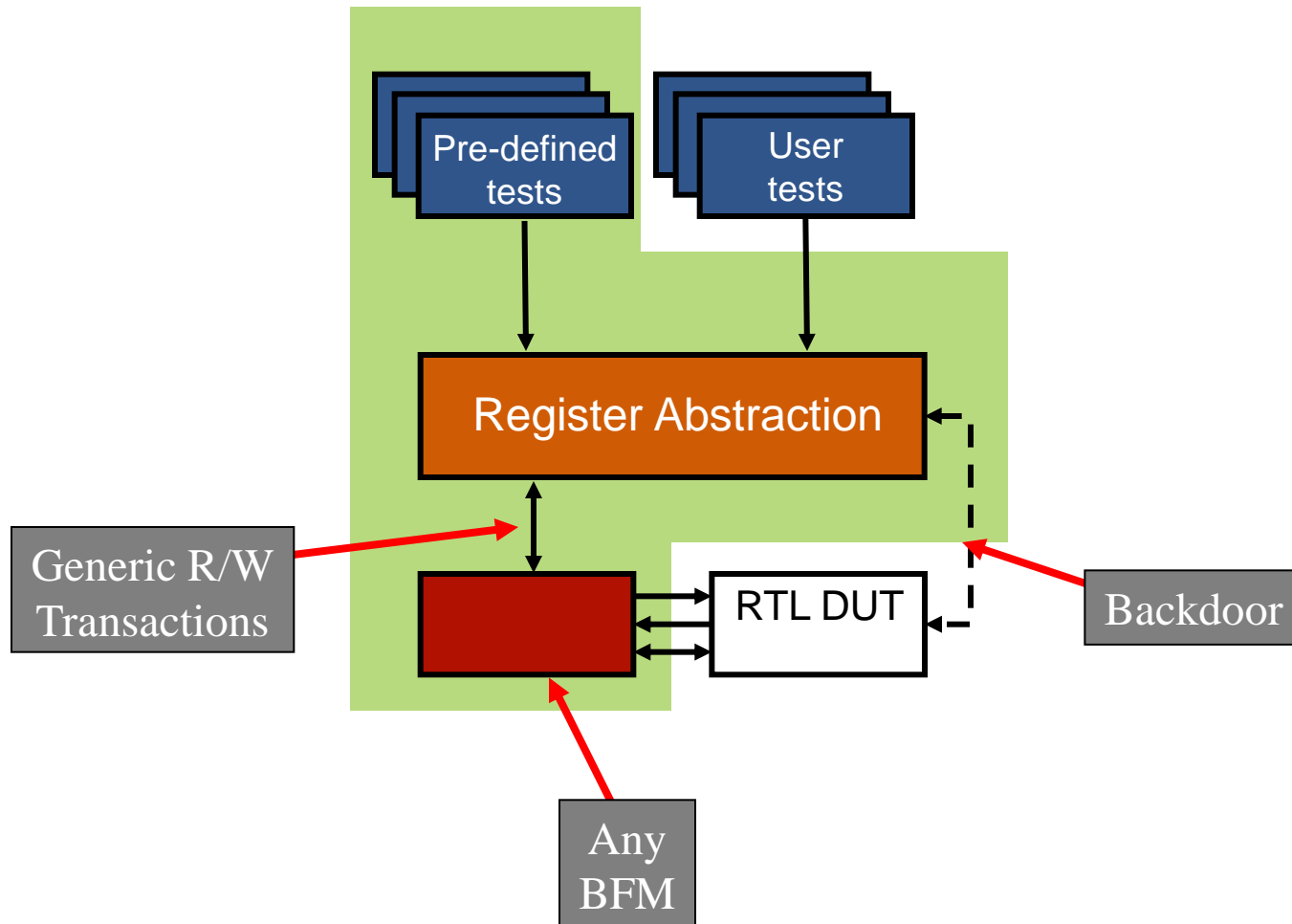
(*) Trademarks are the property of their respective owners.
Not necessarily a complete description of generator I/O capabilities.
No implications of endorsement from respective owners.

Register Interoperability



NEXT: MENTOR GRAPHICS

More Details on UVM Register Classes



FOLLOWING: SEMIFORE

Register Management in Action

