Transaction Based Assertion for Transaction Level Coverage, Property and Protocol Checking

Thinh Ngo & Sakar Jain

freescale

Synopsys Users Group
SILICON VALLEY 2013

Agenda

- Introduction
- Transaction Level Coverage/Checking
- Procedural Checking, TBA and SVA Comparison
- When to Use TBA
- TBA Structure
- Example of a Real-life TBA
- TBA Example Implementation
- What Next?
- Conclusion

freescale

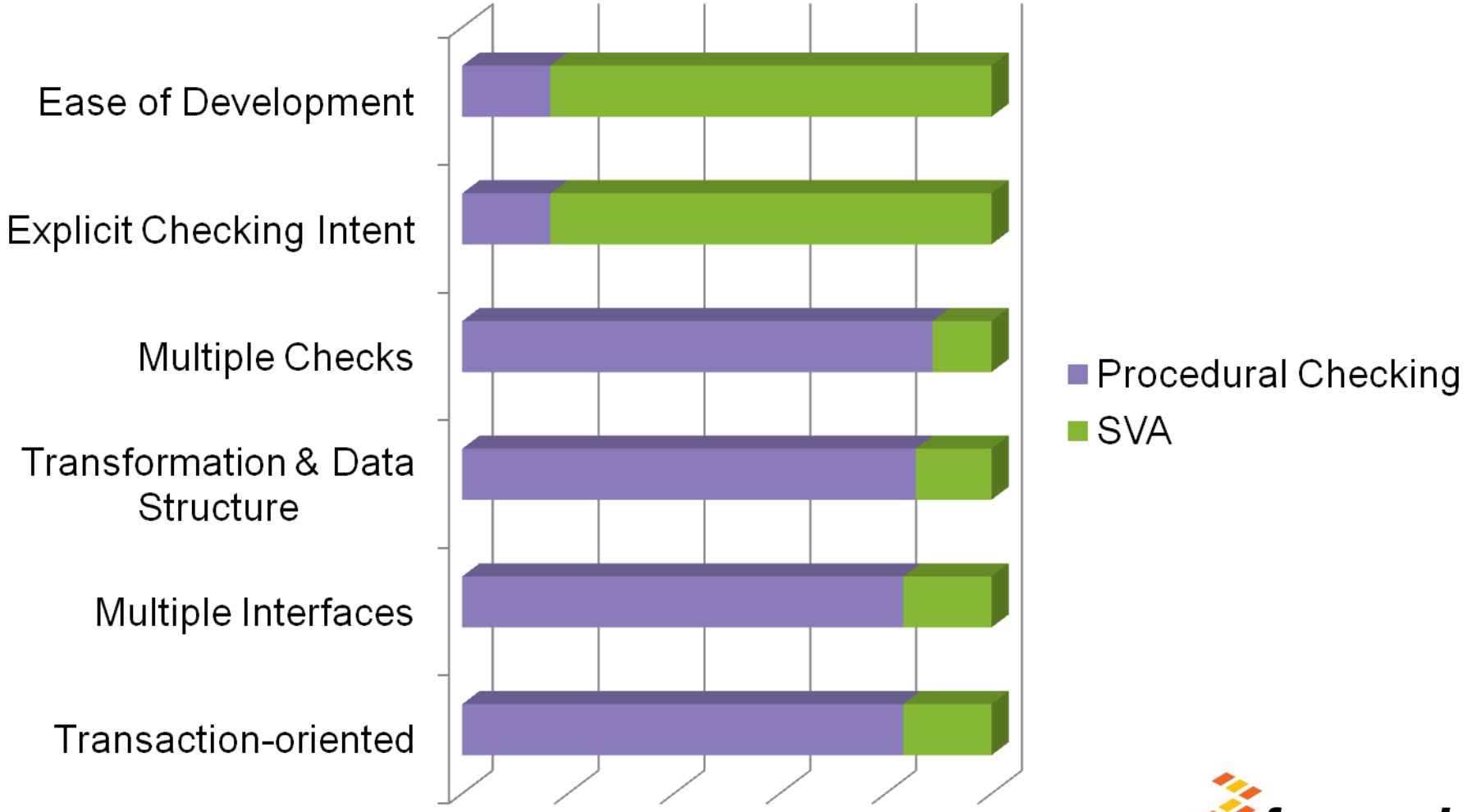
Introduction

- Transaction Based Assertion (TBA)
  - Works on transactions instead of signals
  - Natural and necessary extension of SVA
  - Higher level of abstraction
  - Increased productivity

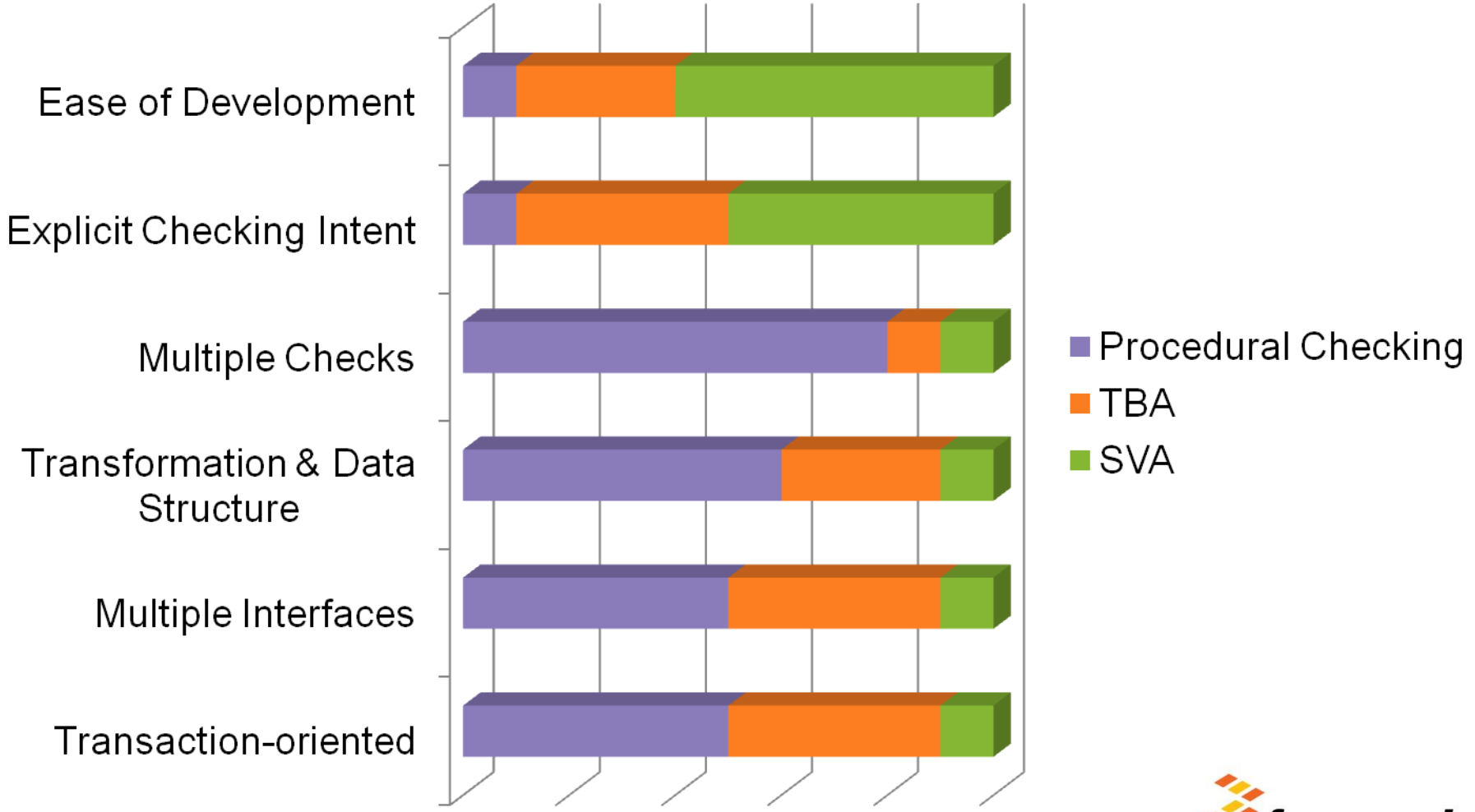
Transaction Level Coverage & Checking

- Currently exists for intra-block
- Need more
  - At inter-block, system-level, inter-IP, SoC-level
  - For end-to-end and global checking
- Challenge
  - Requires expertise of multiple blocks/IPs (Inter-block & Intra-block protocols)
- Solution
  - TBA

Procedural Checking – SVA Comparison



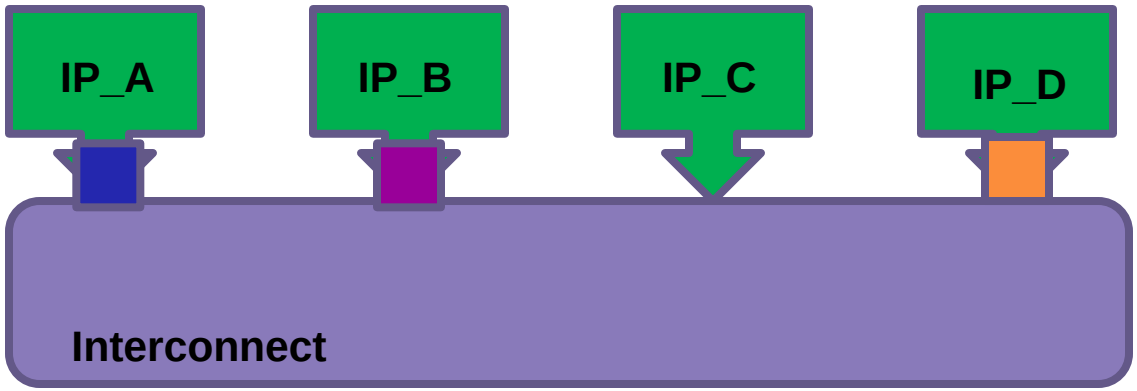
Procedural Checking - TBA - SVA Comparison

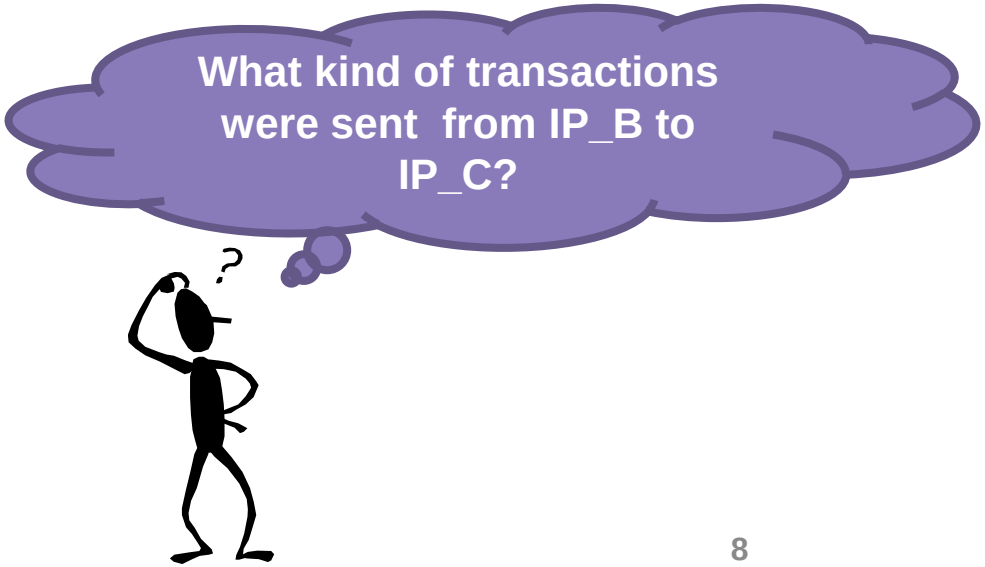


When to Use TBA

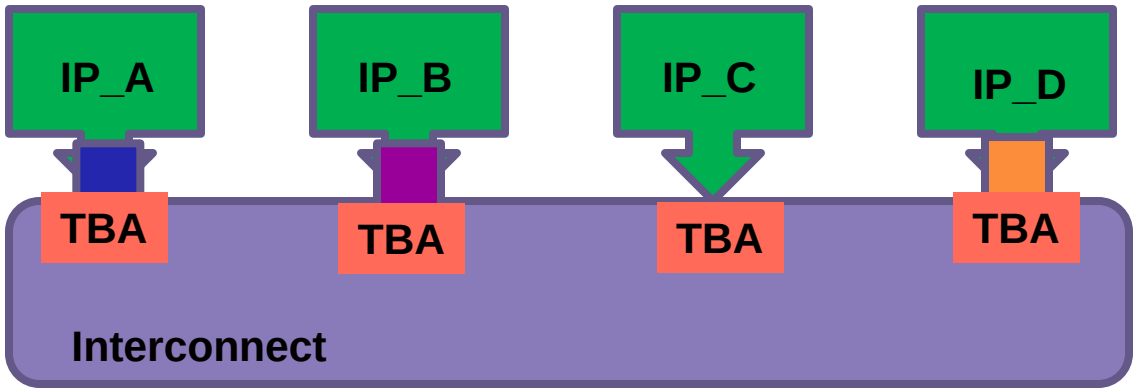
- Coverage & Checking
  - Groups of related signals (e.g. transactions)
  - Simple transformations and data structures
  - Global and transaction level protocol
    - SoC transaction traffic, packet switch, networking
    - Transformed from signal-based protocol (i.e. Cache)
  - End-to-end among external interfaces:
    - dma, cache, memory coherency, livelock/deadlock, ordering, semaphore, etc.
    - protocol participants & participation types

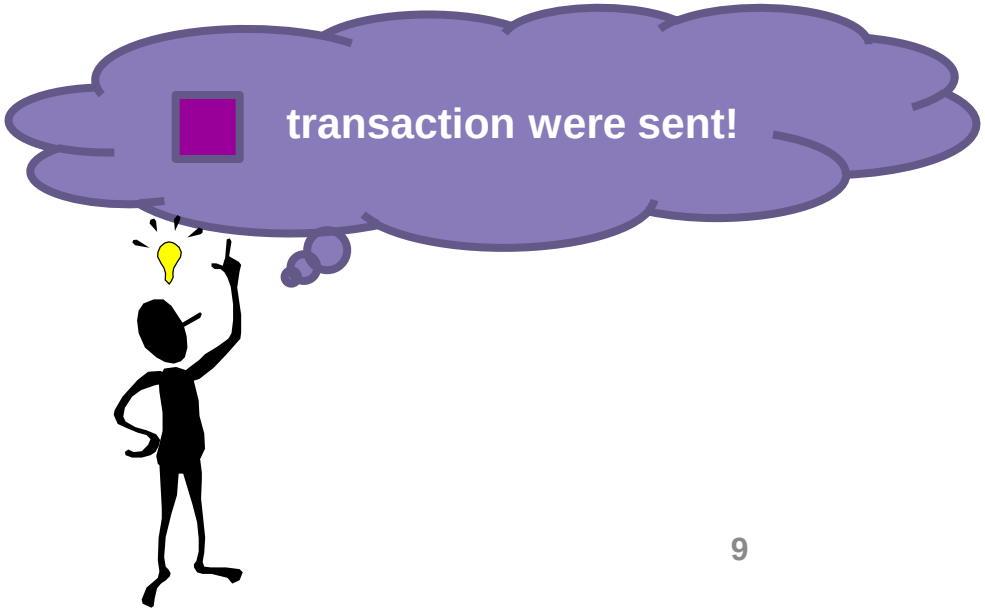
End-to-end Coverage & Checking without TBA





End-to-end Coverage & Checking with TBA





TBA Structure

- **Transaction class**
  - Emulates the role of SVA signals
  - Encapsulates interface behavior
  - Performs transformations for structural relationships among transactions at various interfaces
- **Assert_Cover class**
  - Emulates a SVA cover assertion
  - Accepts Transactions and their relationships
- **Assert_Property class**
  - Emulates a SVA property assertion
  - Accepts cause & effect Assert_Covers and their relationships

Example of a Real-life TBA

- Used TBA to verify 3 caches at the cache wrapper level where the following cache functionalities are accessible
  - miss/hit
  - fetch
  - write
  - invalidate

Protocols for the TBA Example

1. A cache miss must not be consecutively followed by a cache miss of the same cache line.
2. A cache miss must not be consecutively followed by a cache hit of the same cache line.
3. A cache miss must not be consecutively followed by a cache write of the same cache line.
4. A cache hit must not be consecutively followed by a cache miss of the same cache line.
5. A cache hit must not be consecutively followed by a cache write of the same cache line.
6. A cache hit must not be consecutively followed by a cache fetch of the same cache line.
7. A cache write must not be consecutively followed by a cache miss of the same cache line.
8. A cache write must not be consecutively followed by a cache write of the same cache line.
9. A cache write must not be consecutively followed by a cache fetch of the same cache line.
10. A cache fetch must not be consecutively followed by a cache miss of the same cache line.
11. A cache fetch must not be consecutively followed by a cache hit of the same cache line.
12. A cache fetch must not be consecutively followed by a cache fetch of the same cache line.
13. A cache invalidate must not be consecutively followed by a cache hit of the same cache line.
14. A cache invalidate must not be consecutively followed by a cache write of the same cache line.
15. A cache invalidate must not be consecutively followed by a cache fetch of the same cache line.

TBA Example Implementation Transaction class

- Properties: Type, Triggered, Comparing_Tag
- Methods: Monitor(), Compute_Comparing_Tag()

Transaction Cache Write monitor method:

```
task PCacheWriteTxn::monitor ();
@(PCacheIF.cb);
 if (PCacheIF.cb.pamu_pcache_write) begin
   Txn.Address[24:57] = PCacheIF.cb.pcache_updt_req_addr[24:57];
   Txn.Triggered = 1'b1;
   Txn.Comparing_Tag = Compute_Comparing_Tag();
```

Transaction Cache Write Compute Comparing Tag:

```
function logic [0:39] PCacheWriteTxn::Compute_Comparing_Tag();
logic [0:11] liodn;
logic [0:63] pcache_base_addr;
pcache_base_addr[0:63] = {pcache_base_addr_h[0:31],
                                          pcache_base_addr_l[0:31]};
liodn[0:11] = Txn.Address[46:57] - pcache_base_addr[46:57];
Compute_Comparing_Tag={(pcache_base_addr[24:57]+ liodn[0:11]), 6'b0};
```

TBA Example Implementation Assert_Cover class

- Properties: Asserted, Comparing_Tag
- Methods: Monitor()
  - Check the triggering status of each transaction
  - Check logical, temporal and structural relationships
  - Assert if involved transactions' triggerings and relationships are met

- **new** function prototype
  - Accepts Transactions and their relationships

```
function new(string name, string negate1 = "", CacheTxnType_enum
txn1type, string logical = "", string temporal = "", string
structural = "=", string negate2 = "", CacheTxnType_enum  txn2type
= "");
```

  - Example

    A cache miss and a cache hit of same cache line occur at the same time

```
Assert_Cover MISS_and_HIT;
MISS_and_HIT = new("MISS_and_HIT",, MISS, "and", "=", "=",, HIT);
```

Synopsys Users Group
SILICON VALLEY 2013

TBA Example Implementation Assert_Property class (1)

- Properties: Asserted, Comparing_Tag
- Methods: Monitor()
  - Check the assert status of cause Assert_Cover and effect Assert_Cover TBAs
  - Check logical, temporal and structural relationships of Assert_Cover TBAs
  - Assert if Assert_Cover TBAs' Asserted status and relationships are met

freescale

TBA Example Implementation Assert_Property class (2)

- **new** function prototype
  - Accepts cause & effect Assert_Covers and their relationships

```
function new(string name, string relax = "", string Pnegate = "",
 string negate1 = "",  CacheTxnType_enum txn1type, string logical1
 = "", string temporal1 = "", string structural1 = "=", string
 negate2 = "", CacheTxnType_enum  txn2type = "", string Pimplicate
 = "|=>", string Ptemporal = "", string Pstructural = "=", string
 negate3 = "", CacheTxnType_enum txn3type, string logical2 = "",
 string temporal2 = "", string structural2 = "=", string negate4 =
 "", CacheTxnType_enum txn4type = "");
```

  - Example

A cache miss must not be consecutively followed by a cache hit of same cache line

```
Assert_Property PCACHE_NOT_MISS_then_same_MISS;
PCACHE_NOT_MISS_then_same_MISS  =
new("PCACHE_NOT_MISS_then_same_MISS",
       "NOT",,, MISS,,,,,,        "|->",,"=",, HIT);
```

What Next?

- Make it part of the SystemVerilog language
  - Assert_Cover is like a SVA cover assertion
  - Assert_Property is like a SVA property assertion
- Add features to handle transaction streams
  - Producer-consumer
  - One-to-one, one-to-many or many-to-one
  - In-order, out-of-order, priorities

Conclusion

- Transaction Based Assertion
  - Is a natural extension of SVA
  - Works on transactions instead of signals
  - Combines the simplicity of SVA and the complexity of procedural checking
  - Is good for global and end-to-end coverage/checking on multiple interfaces