A bottom-up approach to top-down VMM

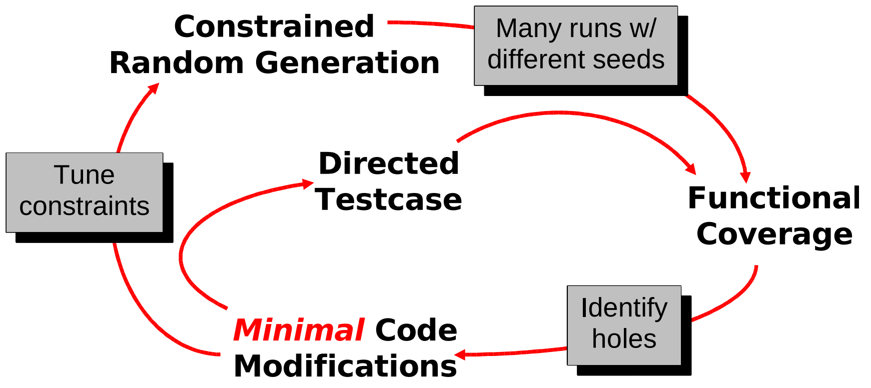
Xavier Caron
Design Flow Engineer
Atmel Rousset

The big idea behind IP verification

- An IP is a *reusable* chip *cornerstone*
  - We therefore *must* have confidence in it
- Confidence is built from *passed* tests
  - Since only tests actually exercise the IP
- Tests do not prove the code *is* right
  - Tests are simulated & *incomplete* stimuli sequences
- Tests just prove *when* the code is wrong
  - But they are usually very good at it
- Bottomline: prove RTL code matches *intent*
  - For – *and alas only for* – the *use cases* we exercise

The big idea behind IP verification (cont'd)

- We cannot possibly think about all cornercases
  - This is too daunting a task for our brain
- Besides, writing tests rapidly becomes...
  - Tedious, repetitive, boring and... irritating!
- We need a framework / methodology
  - To let the simulator "push" an IP in *any* possible way
  - While retaining enough control on the "any" concept


Constrained Random Generation
Many runs w/ different seeds
Directed Testcase
Functional Coverage
Tune constraints
Identify holes
Minimal Code Modifications

(Smart quote)

- Insight from a *real* code guru

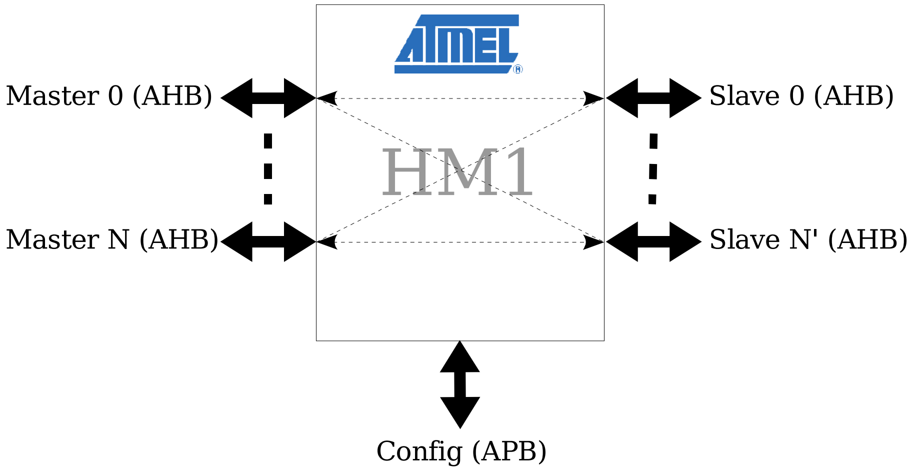
  "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

  -- Brian Kernighan

- One more challenge
  - Verifier & RTL coder are *supposedly* different persons
- How to accommodate for this neuronic deficit?
  - The brute-force way (~ target *all* possible combos)
  - The abstracted way (~ use a *strict* methodology)

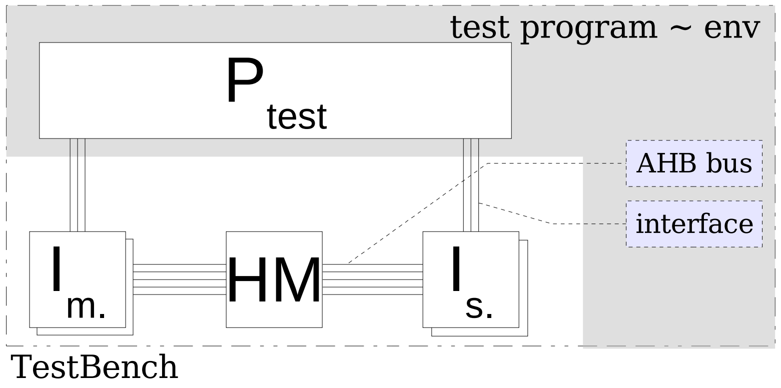
Our VMM experiment

- Main goal: pipe-clean a SVTB+VMM+VIP flow
  - We already had a promising Vera+RVM+VIP one
- Side goal: assert for RTL designer acceptance
  - Or bootstrap a dedicated (proto)verification group
- We picked up a not-too-complex IP
  - Our proprietary AHB-Lite “HMATRIX1” bus matrix



Testbench at a glance

- TB building blocks
  - DUT
  - Interfaces
  - Test program (within an “env” object)
- TB roles
  - Define system clock
  - Instance ad'hoc interfaces
  - Instance DUT and connect its signals to interfaces
  - Instance test program and connect it to interfaces



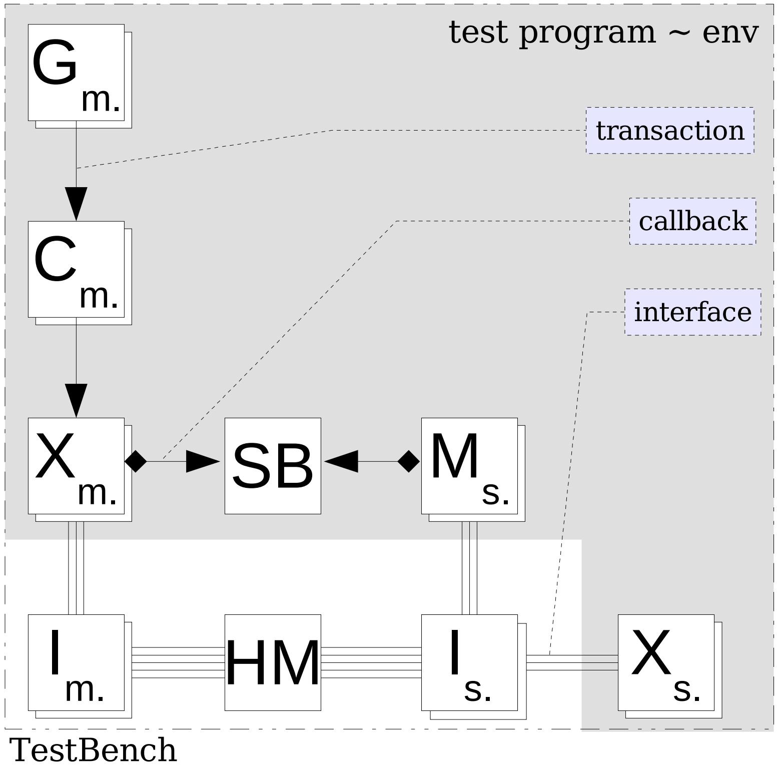
```
P  : Program
I  : Interface
HM : HMATRIX1
m. : master
s. : slave
```

Interfaces

- 3 kinds to be used (a bit of wishful thinking)
  - TB, clocked & reversed DUT-wise
  - DUT, unclocked & I/O as is
  - Monitor, unclocked & all-in
- Avoid the async. all-inout wiring scheme
  - Use clocking blocks / modports to constrain further
- Clocking blocks
  - For timing & synchronization requirements purposes
- Modports
  - To enforce correct I/O wiring
- Consistent glue to '95 style verilog IPs

Test program

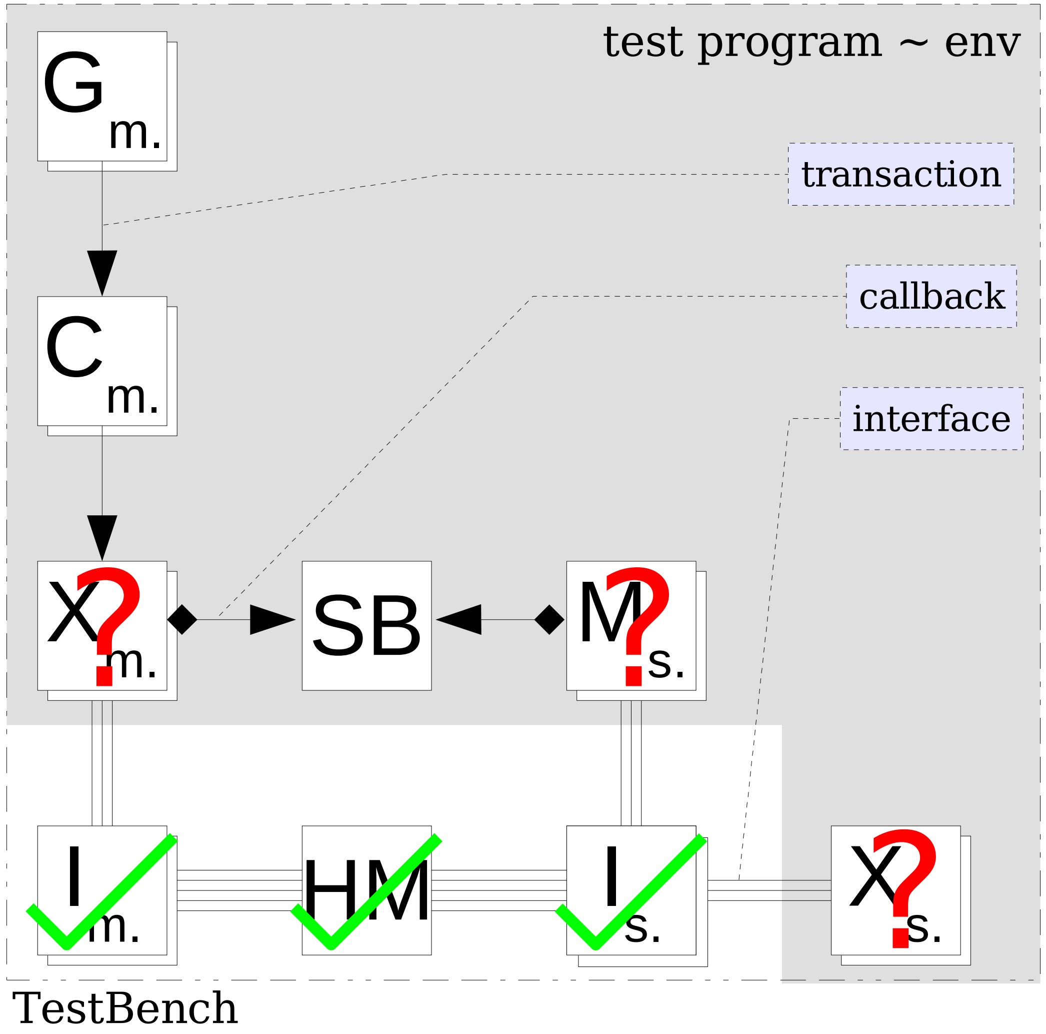
- Transactors
  - Xaction to “bus” *transcoders*
- Channels
  - P2P communication *media*
- Generators
  - Object (~ xaction) *factories*
- Monitors
  - Bus *spies*
- Scoreboard
  - The *focal* point



P	:	*Generator*
C	:	*Channel*
X	:	*Transactor*
I	:	*Interface*
M	:	*Monitor*

SB	:	*Scoreboard*
HM	:	*HMATRIX1*
m.	:	*master*
s.	:	*slave*

Milestones: to transactors



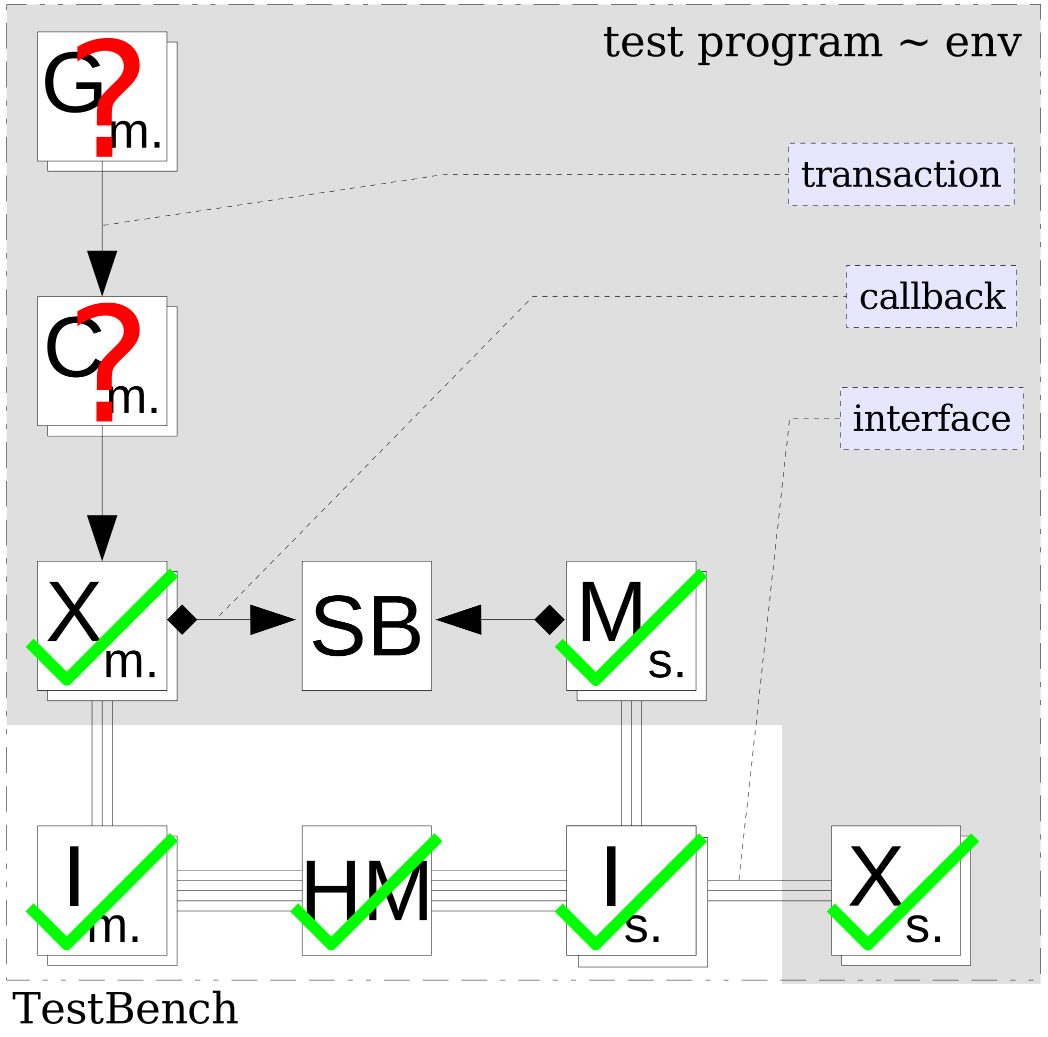
Transactors

- Mediators between 2 kinds of *representations*
  - *Abstract*, borne by xactions (queued class instances)
  - *Physical*, borne by wires (AHB “bus” pin wiggling)
- We needed 3 kinds of xactors
  - Namely *master*, *slave* and *monitor* (cf interfaces)
- Master
  - *Transcodes* instance properties into pin wiggling
- Slave
  - *Reacts* to pin wiggling and updates its own state
- Monitor
  - *Spies* pin wiggling to decipher xactions for the SB

Transactors (cont'd)

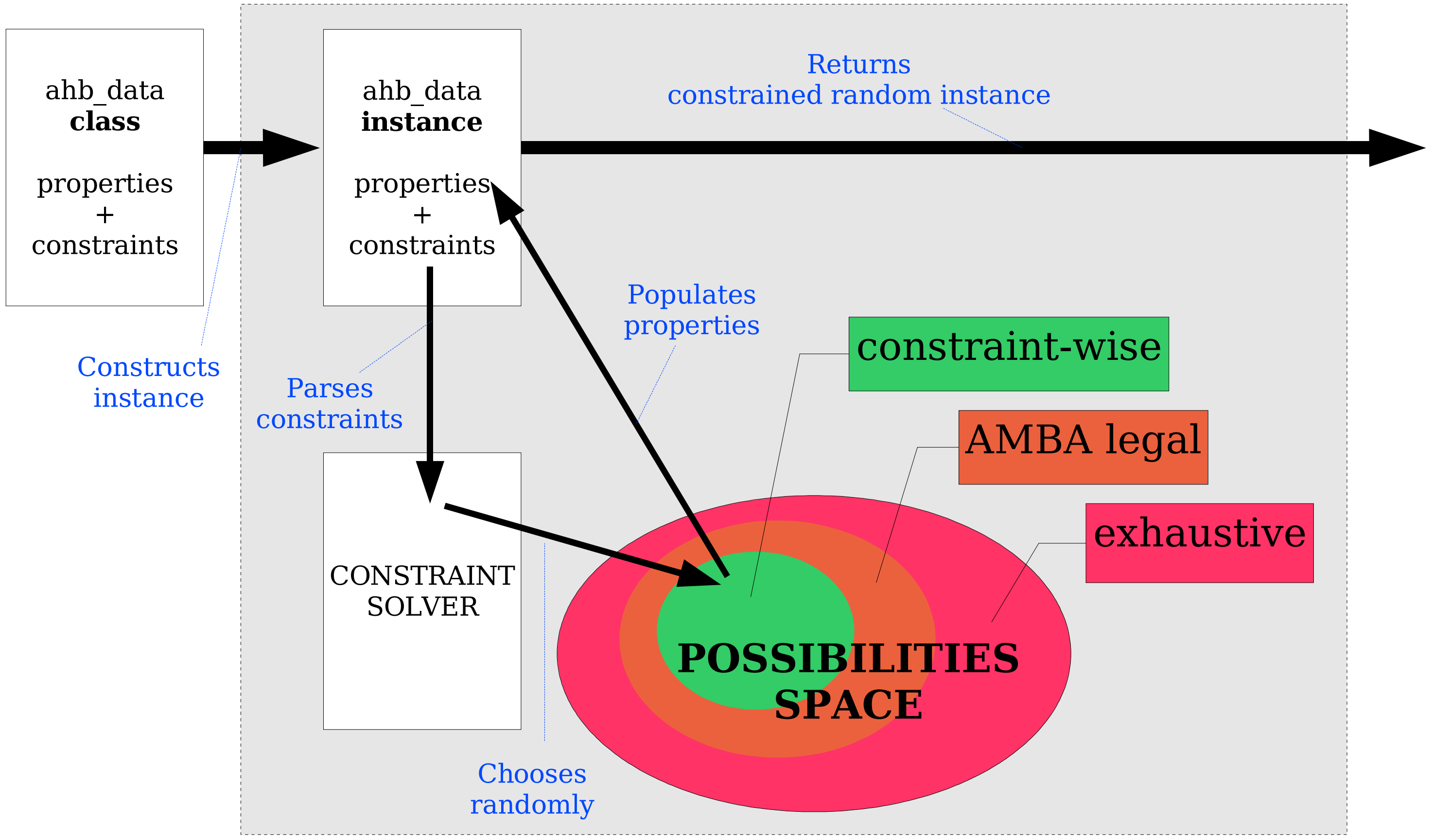
- VIP was not usable with VCS 2005.06-SP1
  - SystemVerilog / OpenVera interoperability is missing
  - Required VIPs are mainly coded in OpenVera
- We had to code our own makeshift xactors
  - With much less features than their VIP counterparts
  - We devised them as transient pieces of code anyway
- The hardest one is – by far – the monitor
  - Real-time on-the-fly xaction decoding is a tough job
- Our monitor has thus one big limitation
  - It only reports beat-wise xactions
  - Beat-to-burst aggregation delegated to SB instead

Milestones: to generators/channels

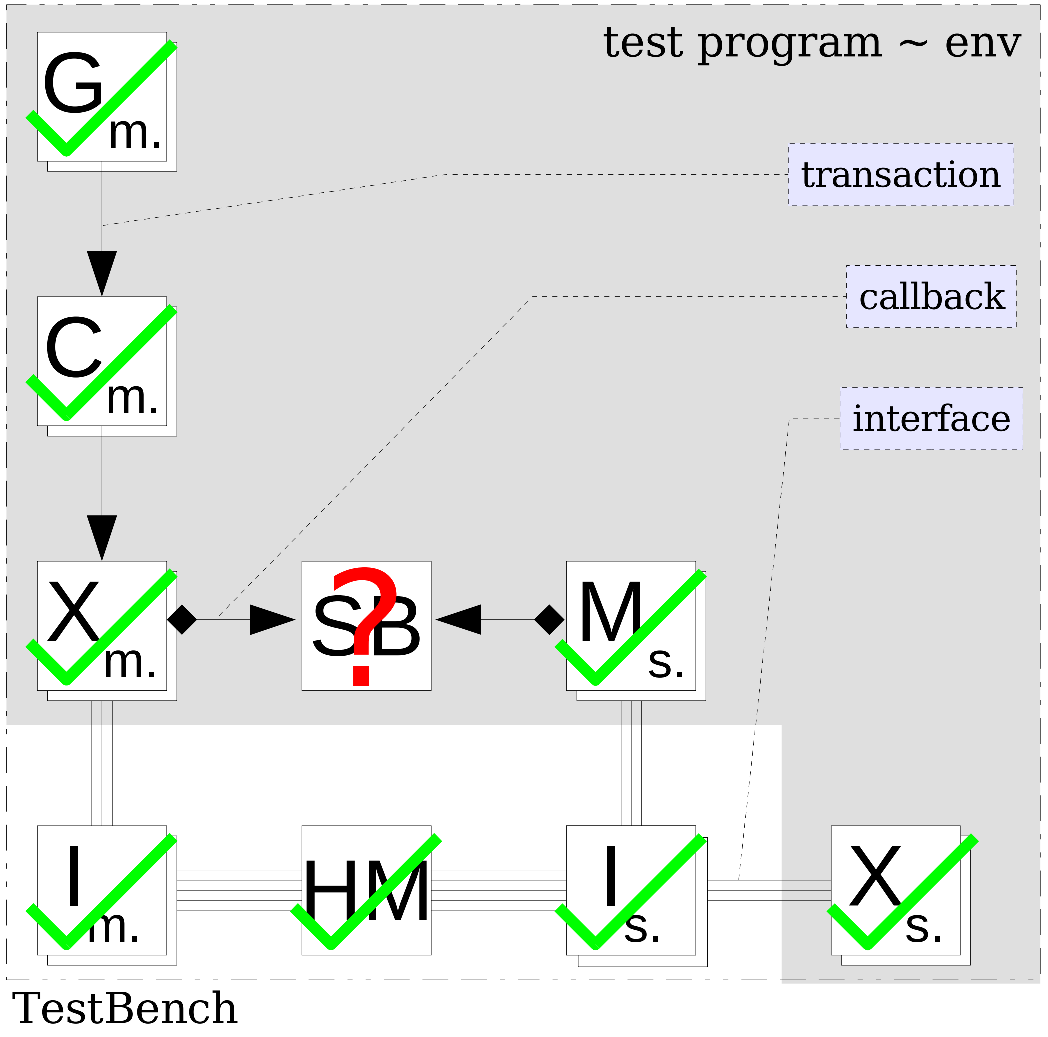


Constrained Random Testing paradigm

- A generator is an instance factory
  - Restlessly constructs, populates & queues xactions



Milestones: to scoreboard

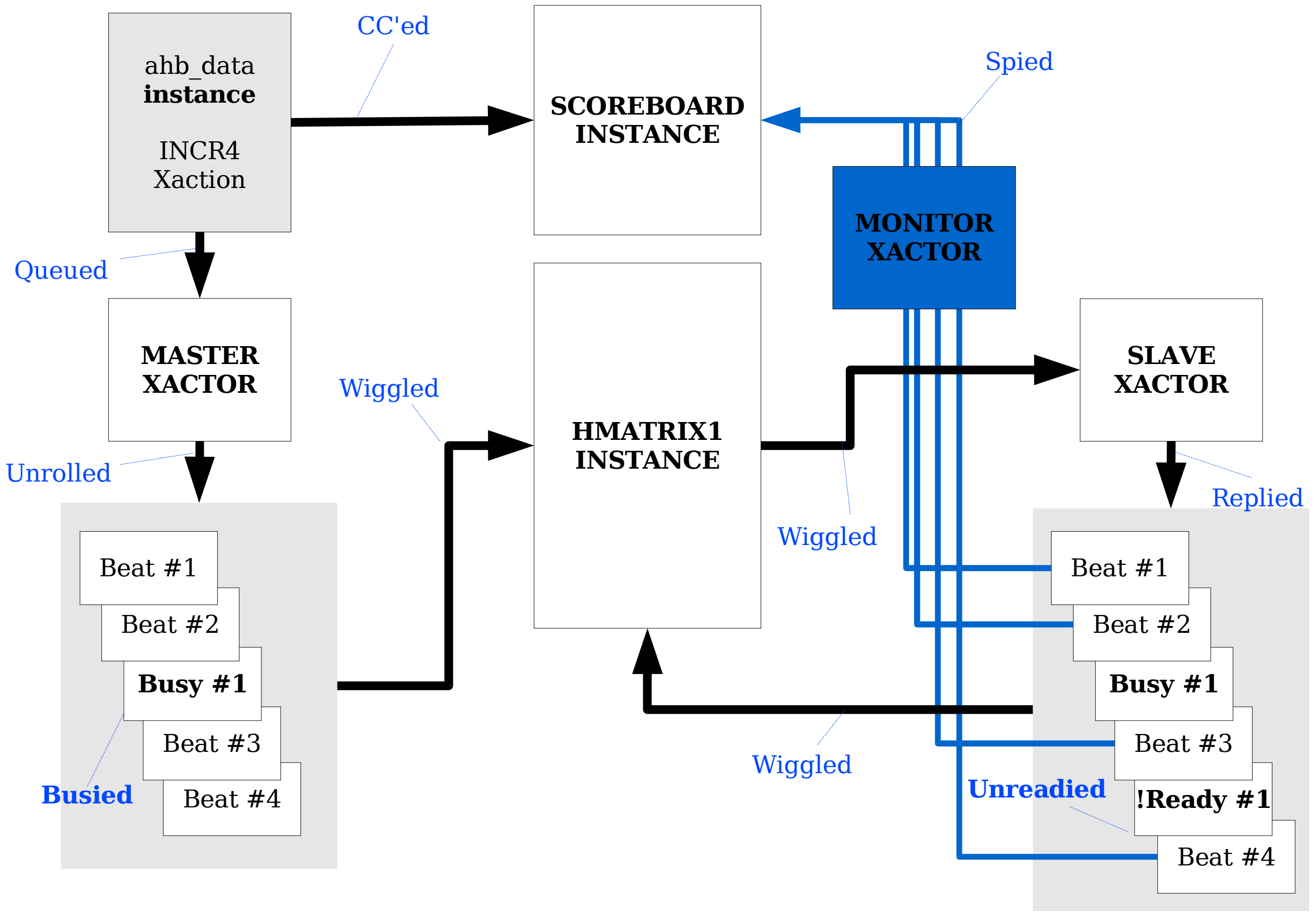


Scoreboard

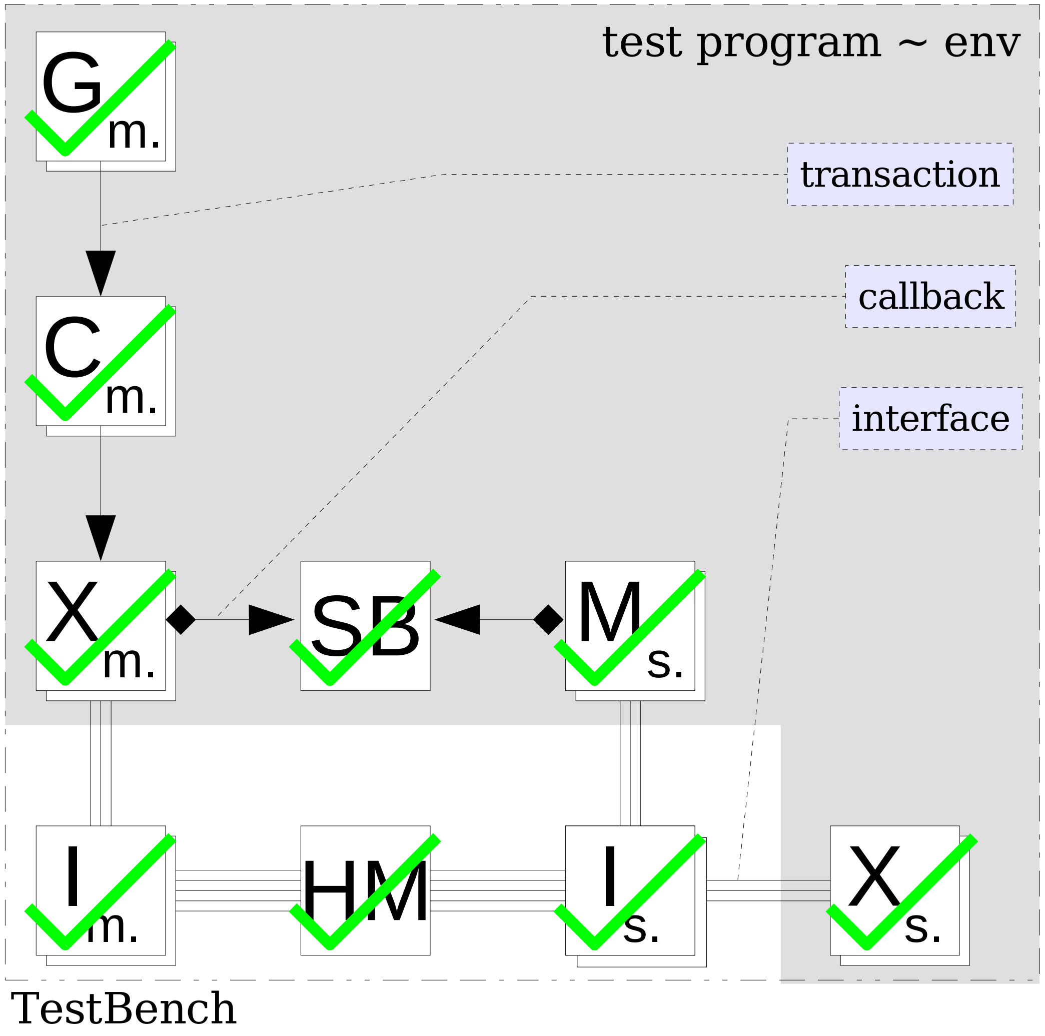
- Performs a “mere” *quantitative* 1-to-1 match
  - A posteriori treatment (no RT feedback loop)
  - On aggregated (master) vs atomic (slave) xactions
- Still does a “first stage” check only
  - Beat-wise integrity (~ atomic)
  - Only reflects correct *wiring* implementation
  - Does not take “beat pertains to burst” into account
- Will eventually need a “second stage” check
  - Burst-wise consistency (~ aggregated)
  - Should reflect higher-level *arbitration* features
  - Like broken bursts, latency, etc.

Scoreboard (cont'd)

- Dataflow diagram



Milestones: to functional coverage



Functional coverage

- CRT paradigm is coverage-driven
  - To avoid "blind" tests
  - To close the "run-*cover*-tweak" verification loop
- Coverage is a *qualitative* study of xactions
  - Used as a metrics
  - Gives a *statistical* summary of a simulation run
- To actually augment *functional* coverage
  - Run longer simulation (if possible)
  - Relax constraints (within allowed legal frame)
  - Beware of potential memory issues!
  - Since a posteriori SB processing implies *accumulation*

Current status

- Yet to be done
  - Switch to VCS 2006.06-B (features SV/OV interop.)
  - Some changes needed but not structural ones:
    - Replace our makeshift xactors by VIP instances
    - Replace our own xaction by VIP one
  - Implement missing / gated features (INCR, etc.)
  - Use assertions to formalize assumptions SB relies on
- Eventually
  - Hand out the whole TB to a “lucky” RTL win^Hcoder
  - For usability review (cf *“minimal code modifications”*)
  - And... Acceptance!

Conclusions (albeit partial ones)

- VMM
  - Sensible, but has a *very steep* learning curve
  - You do not want to start by 1st reading VMM book
  - Use RVM tutorial & one day training instead
- VIP
  - Has desired aggregated xaction reporting features
  - But was not designed to handle P2P AHB connections
  - Yet works if HTRANS is gated by HSEL (workaround)
- SVTB
  - Support via VCS 2005.06-SP1 was more than decent
  - Rich enough, allows for not-so-inelegant workarounds

Questions?

?