Concordia
UNIVERSITY

Department of Electrical
and Computer Engineering

Assertion and Model Checking of SystemC

Sofiène Tahar

Hardware Verification Group
Department of Electrical and Computer Engineering
Concordia University
Montreal, Quebec, CANADA

First Annual North American SystemC Users Group (NASCUG) Meeting

June 07, 2004

North American SYSTEMC User's Group

Outline

SystemC Verification: State-of-the-art

Proposed Verification Approach

Model Checking

Assertion Based Verification

Conclusion

SystemC Verification Problematic

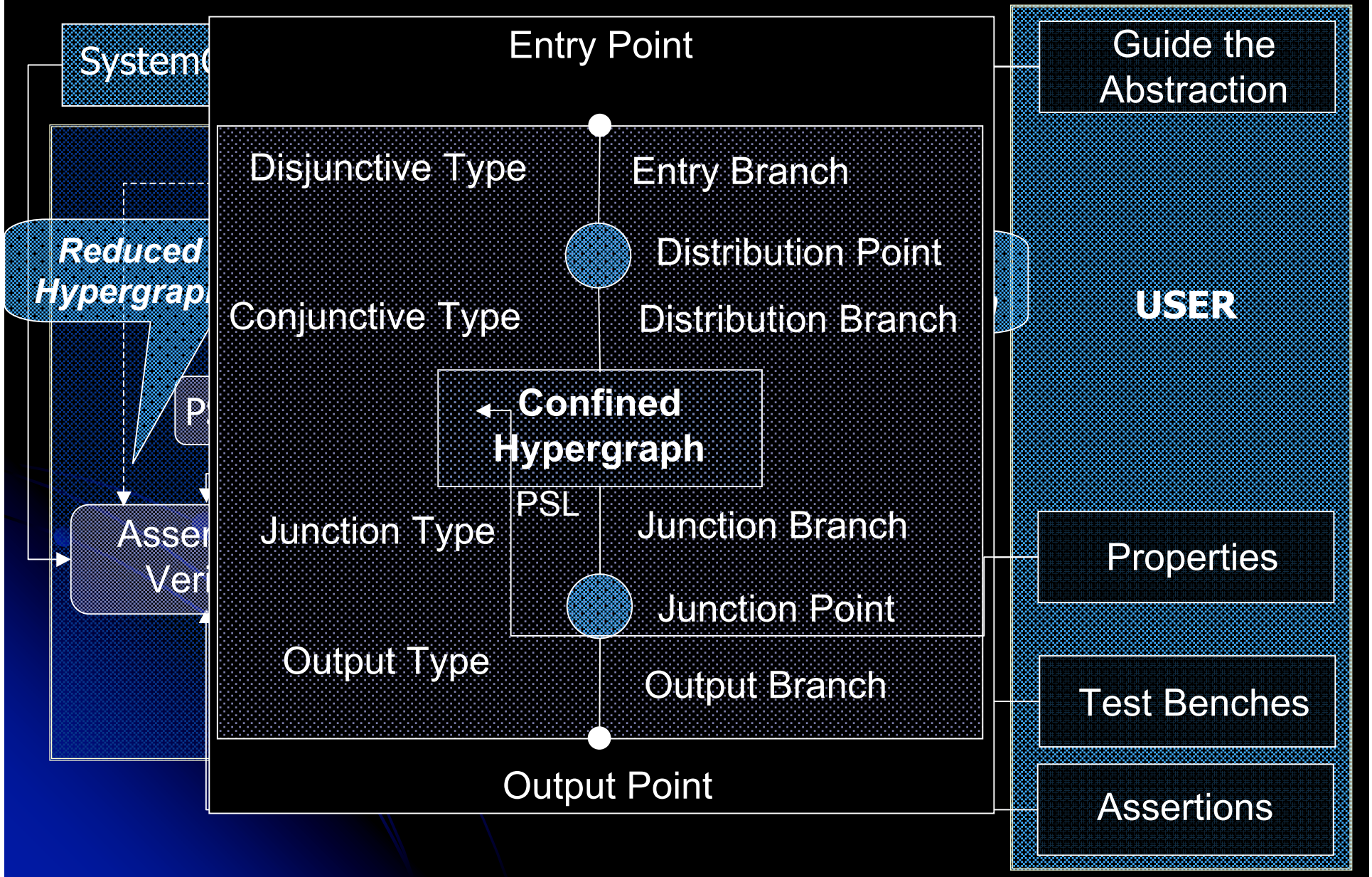
SystemC Verification:

- Functional verification of SoC design (in particular SW) is difficult.
- Checking each feature/subsystem separately is not enough to ensure correct operation.
- Verifying SystemC under construction only makes things worse: adding HW/SW immaturity issues.

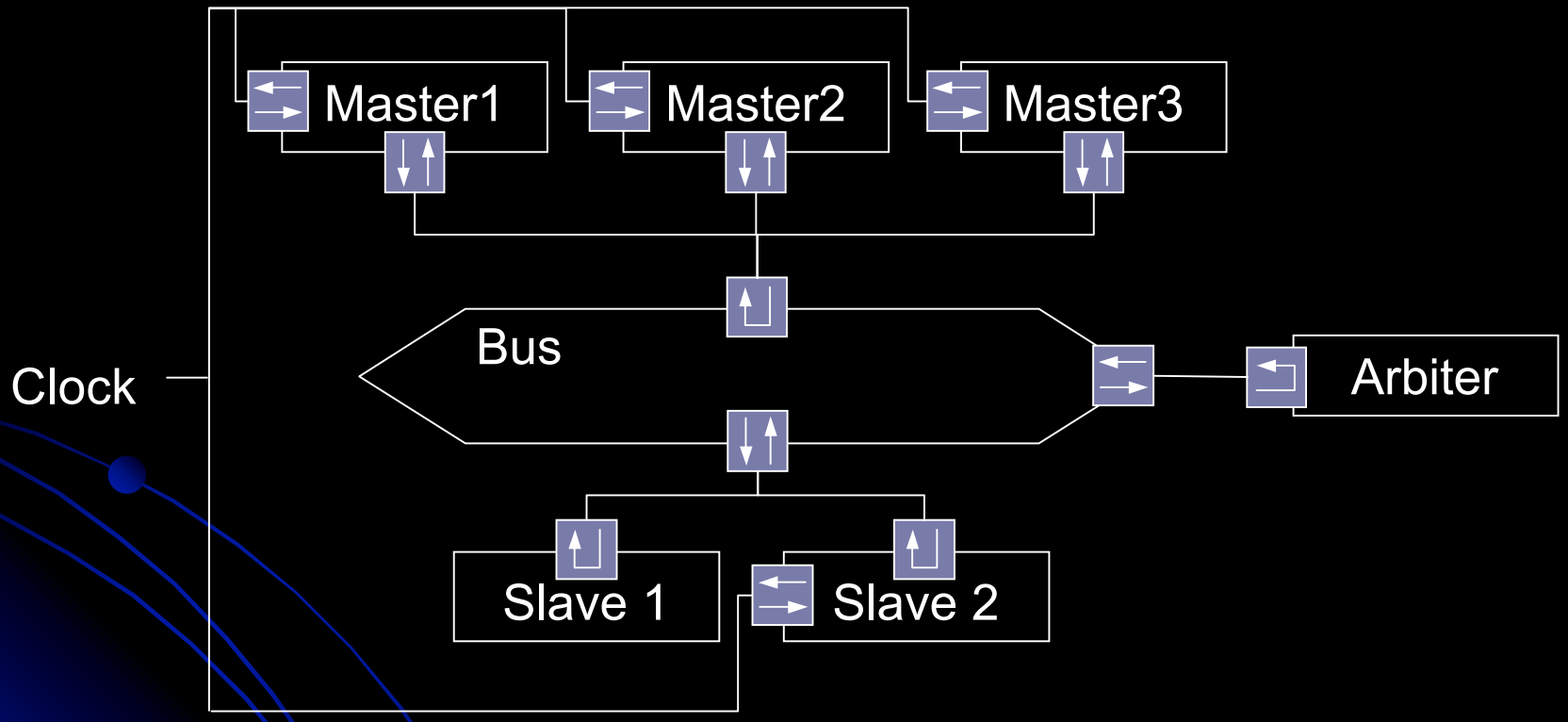
State-of-the-art:

- No relevant new techniques.
- Adapting existing methodologies using:
  - Assertion based verification (ABV).
  - Model checking.
  - Guiding test vector generation (functional coverage).

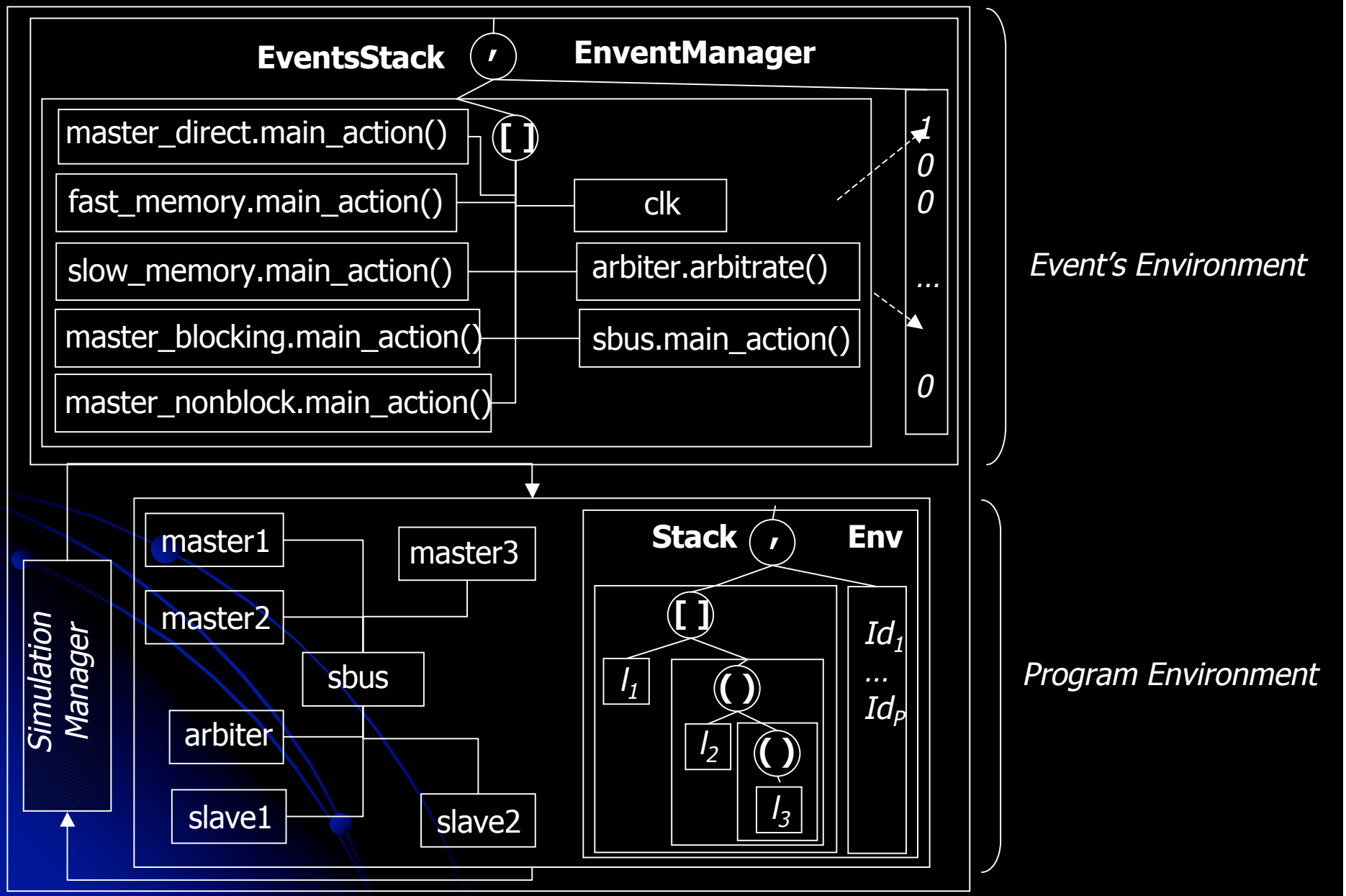
Proposed Solution for SystemC Verification



Case Study: Bus Structure



Snapshot of the Hypergraph



Static Code Analysis

Static code analysis:

- Verify some properties statically.
- Simplify the system structure.
- Offer an interactive and graphical platform.

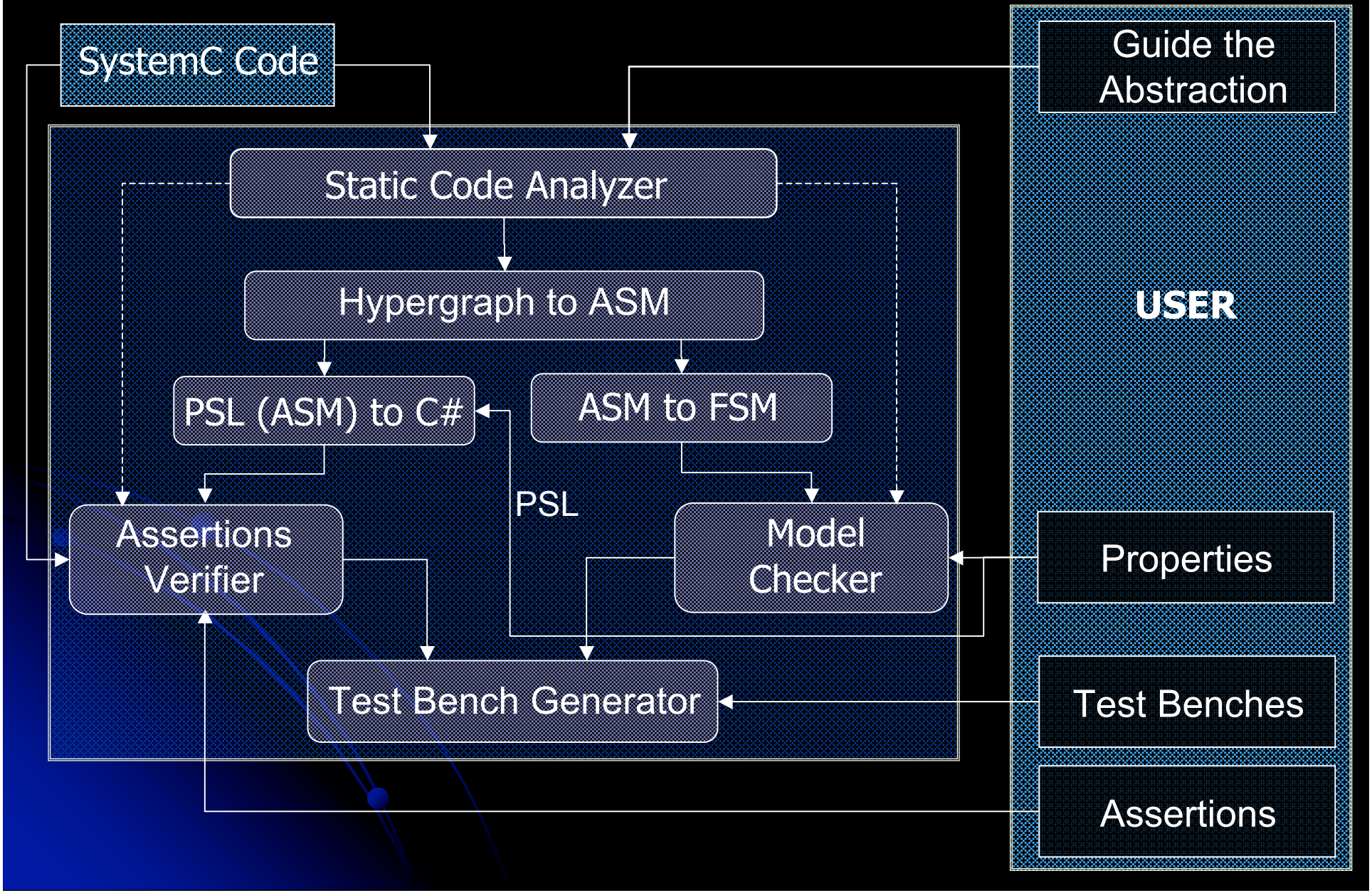
Hypergraph statically executed:

- Extract a reduced representation.
- Verify some of the system's properties (infinite loops, etc.)

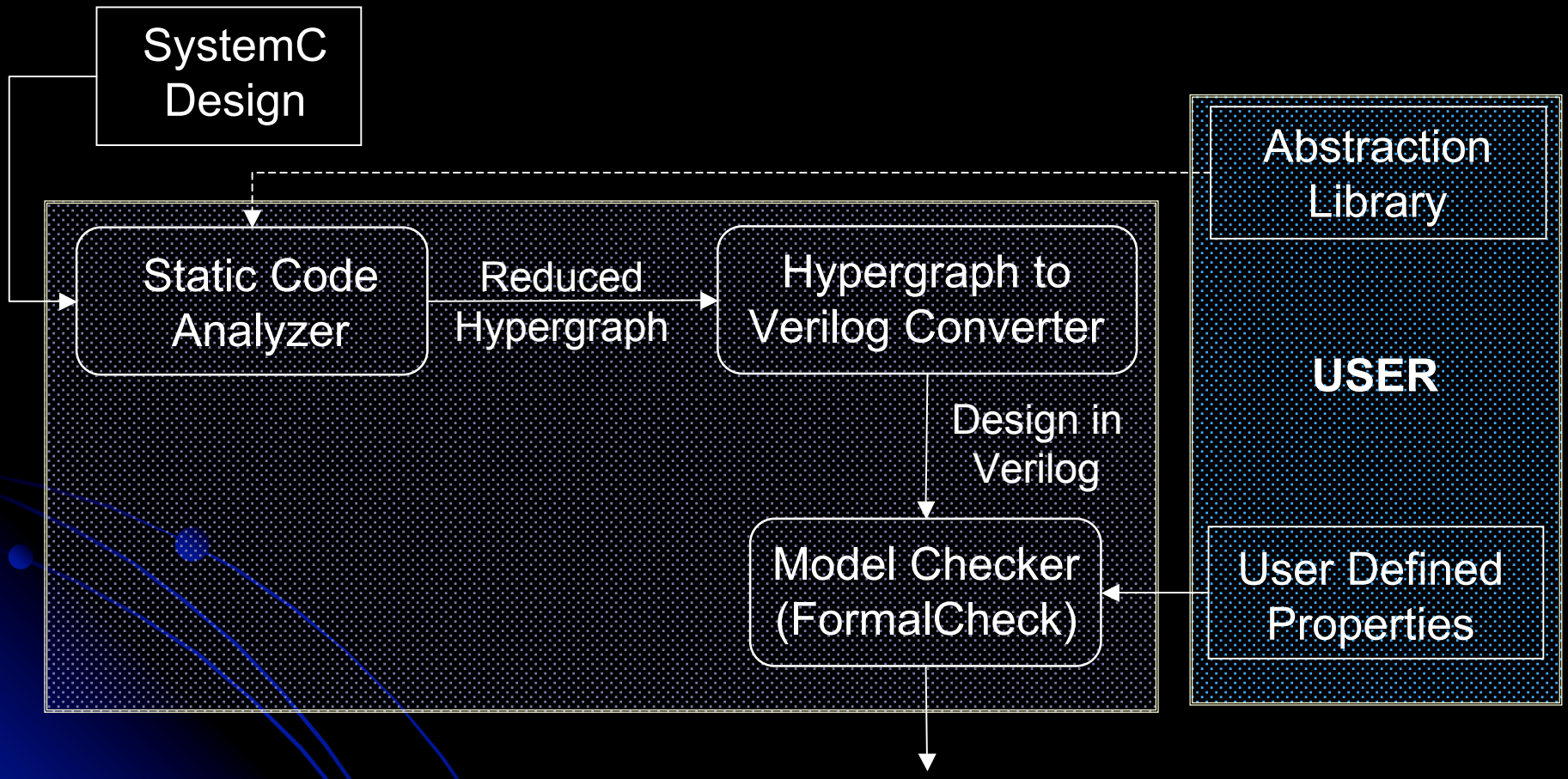
Reduced hypergraph:

- Translated to a format supported by the model checker.
- Used in abstract debugging.
- Executed to get "simplified" graphical snapshots of the system during execution.

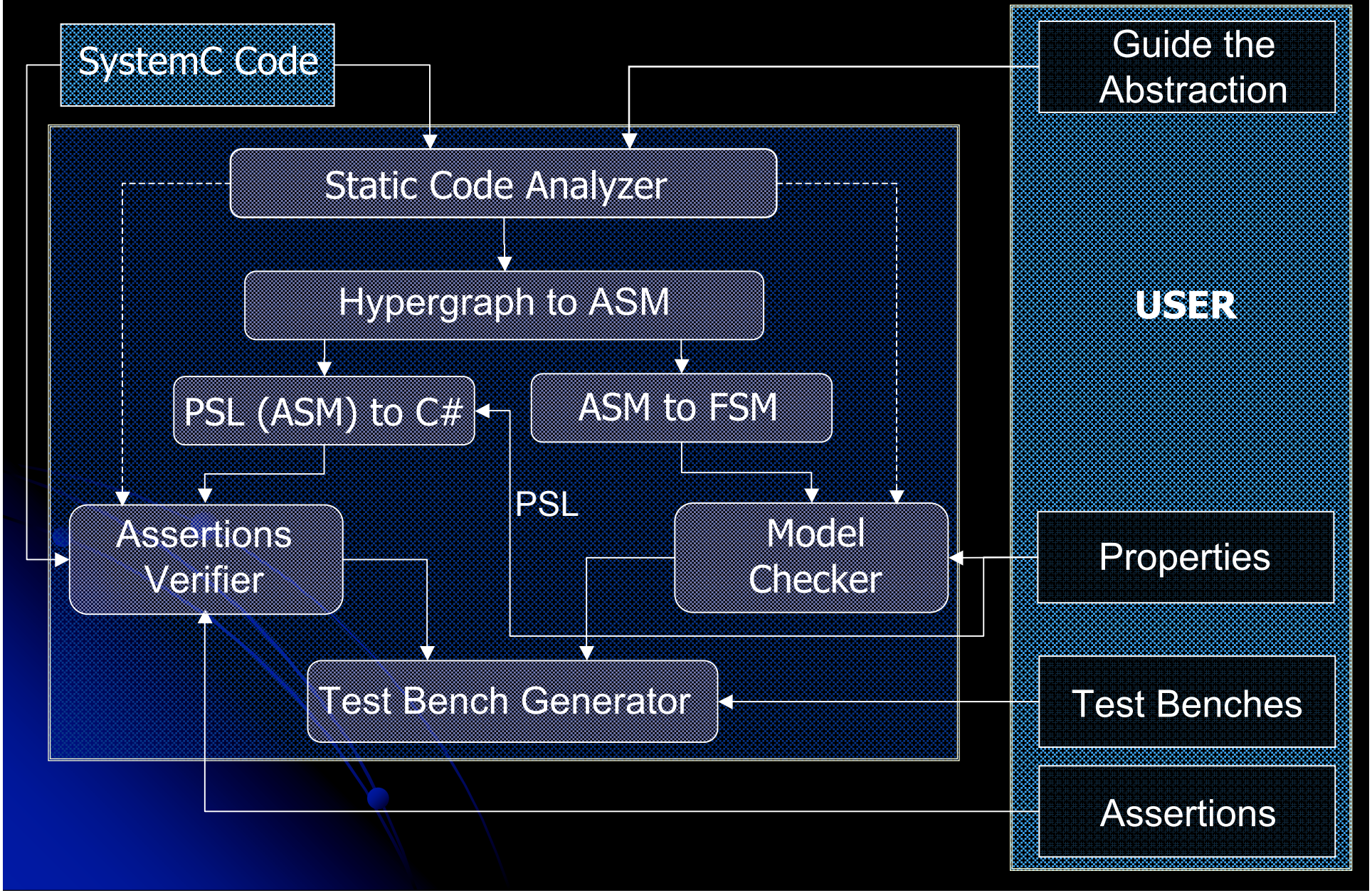
Model Checking: A Simple Approach



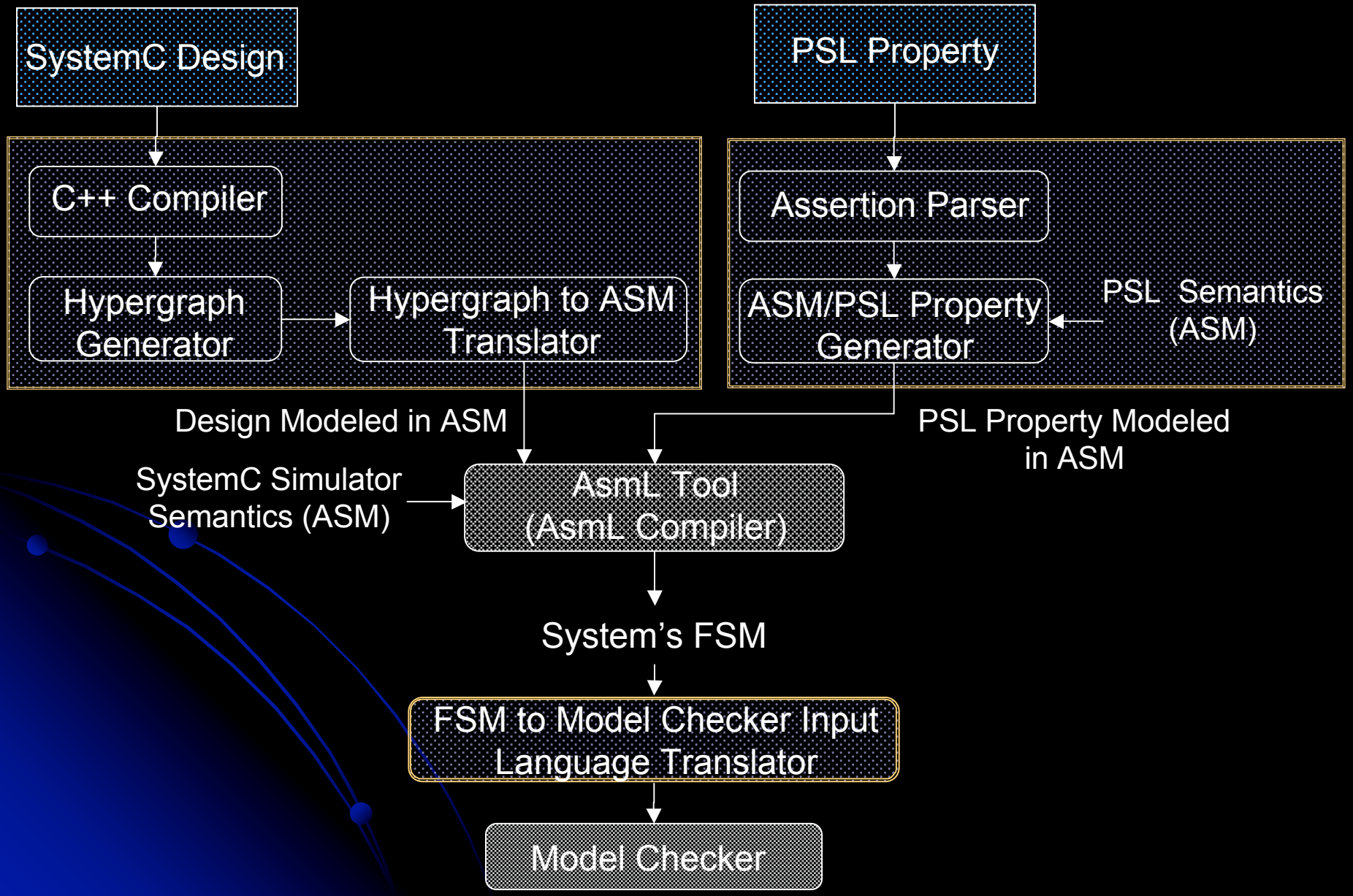
Model Checking: A Simple Approach



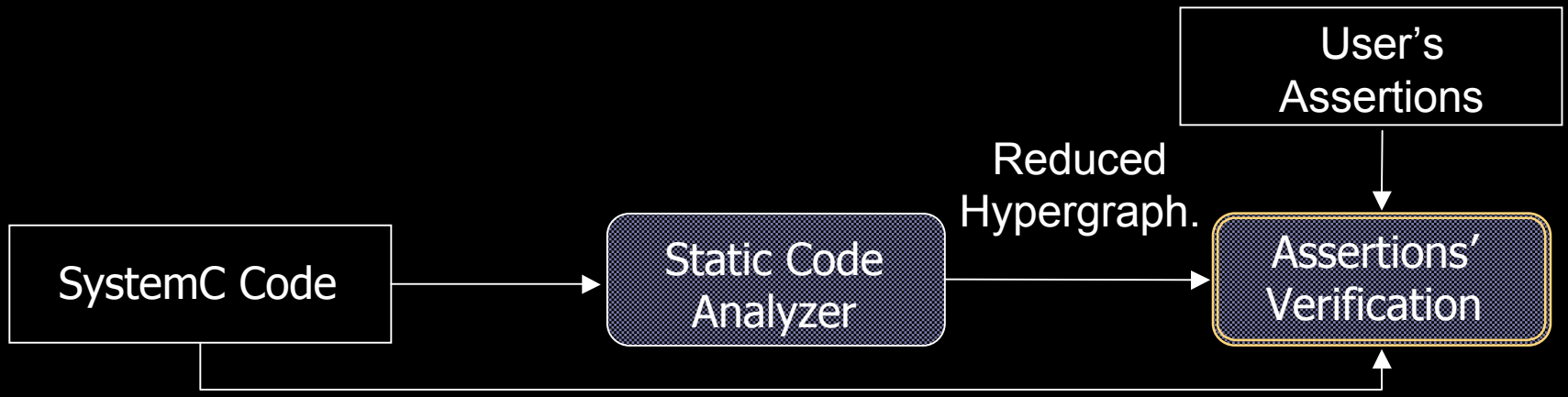
Model Checking: An ASM Based Approach



Model Checking: An ASM Based Approach



Assertion Based Verification (ABV)



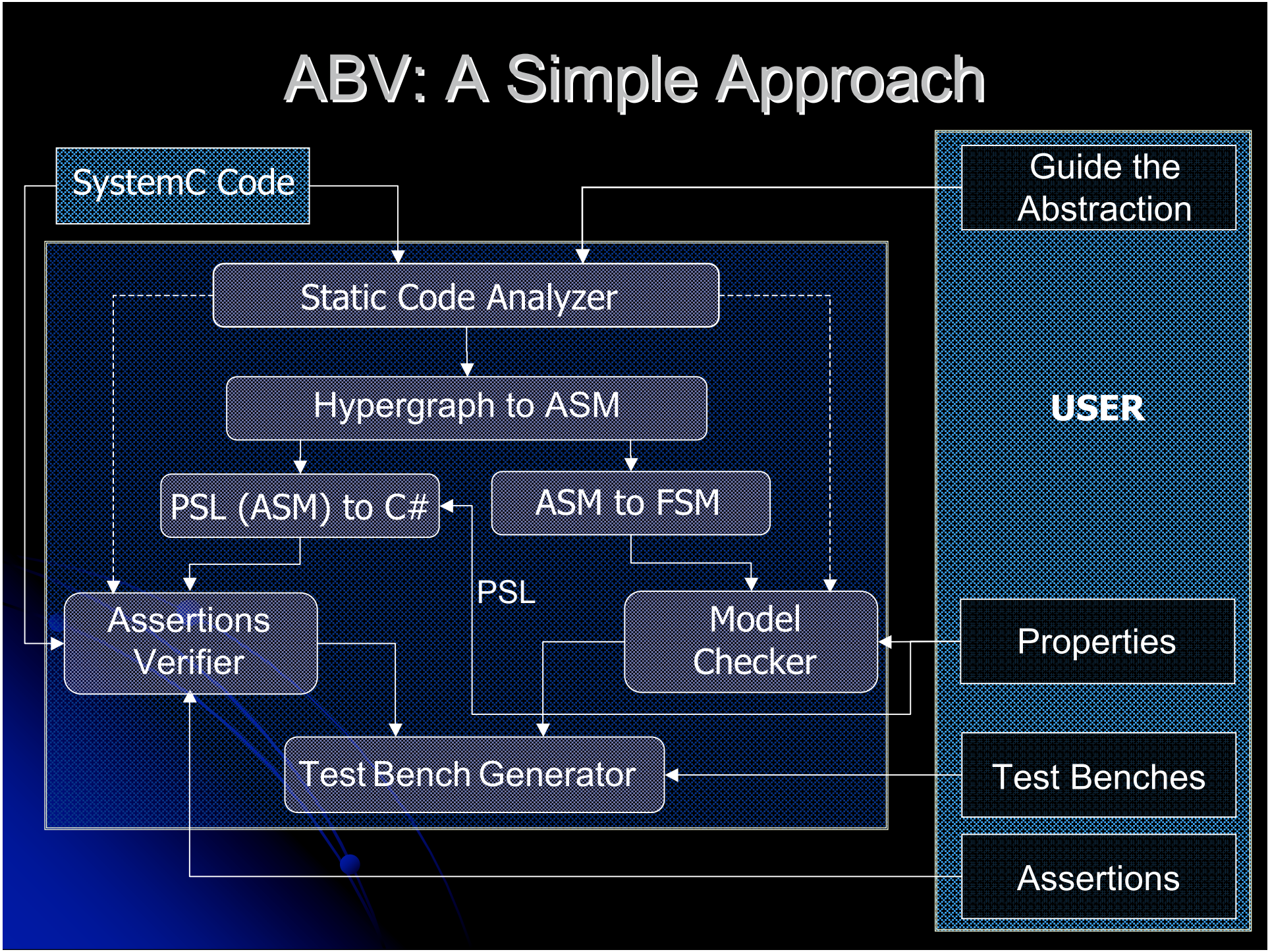
ABV integration on top of SystemC:

- External monitors: easy to integrate and re-use.
- Start with SystemVerilog Assertion (SVA) then extend to Sugar's assertions.

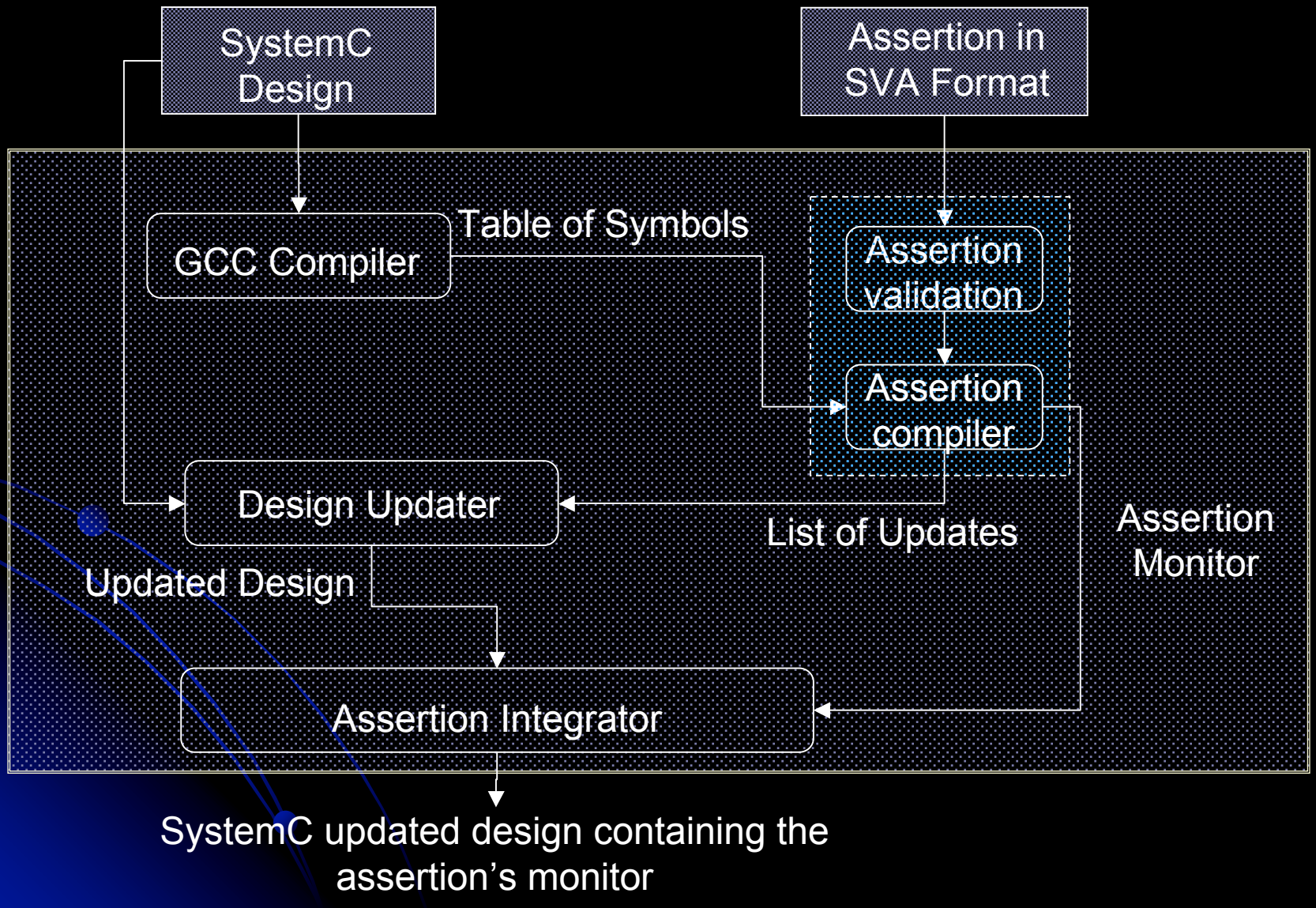
ABV verification: ⇒ Need to abstract the system and apply heuristic algorithms.

- Needs good coverage.
- Requires guiding the test vector's generation.
- Yields very large system state space.

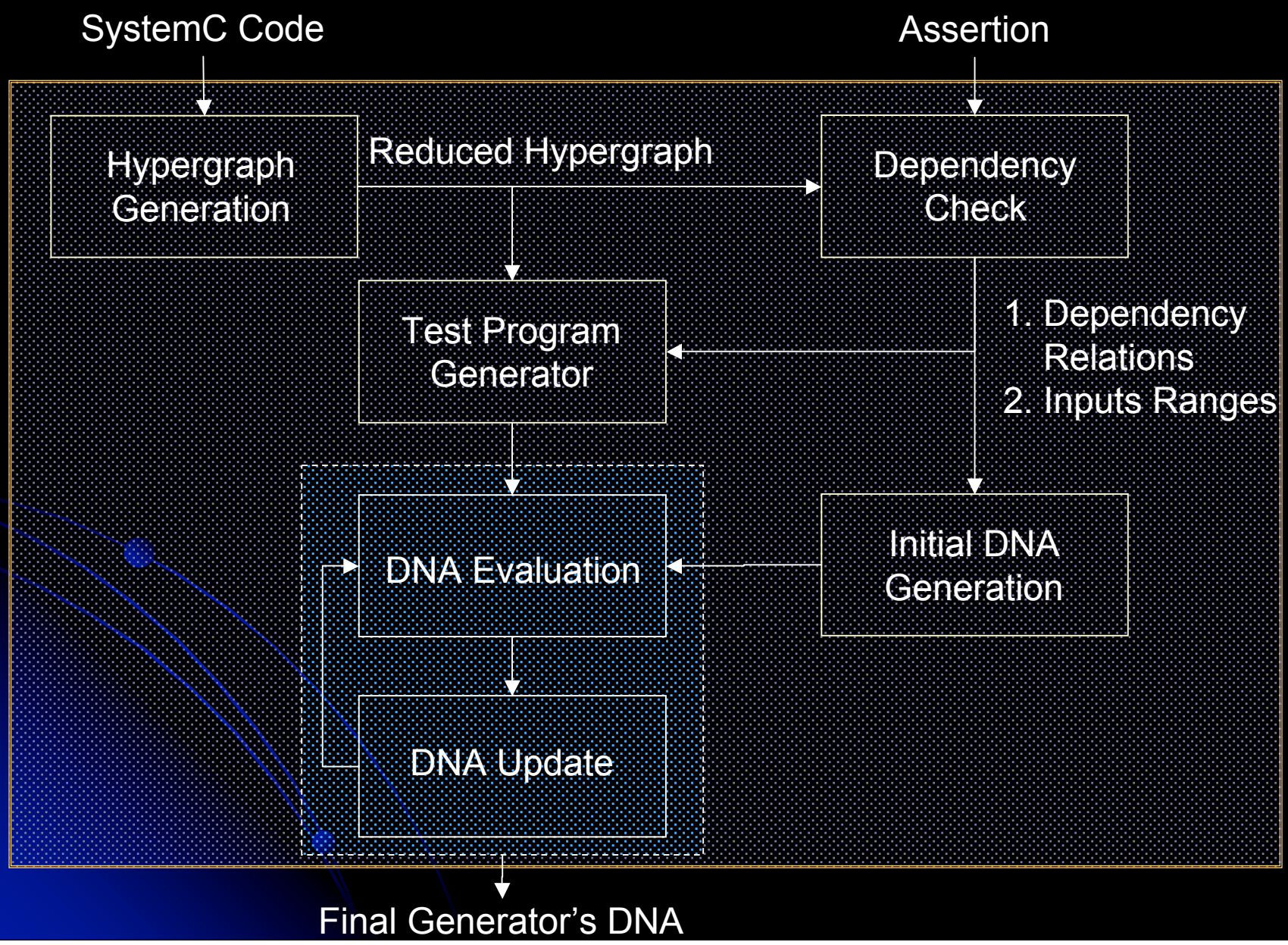
ABV: A Simple Approach



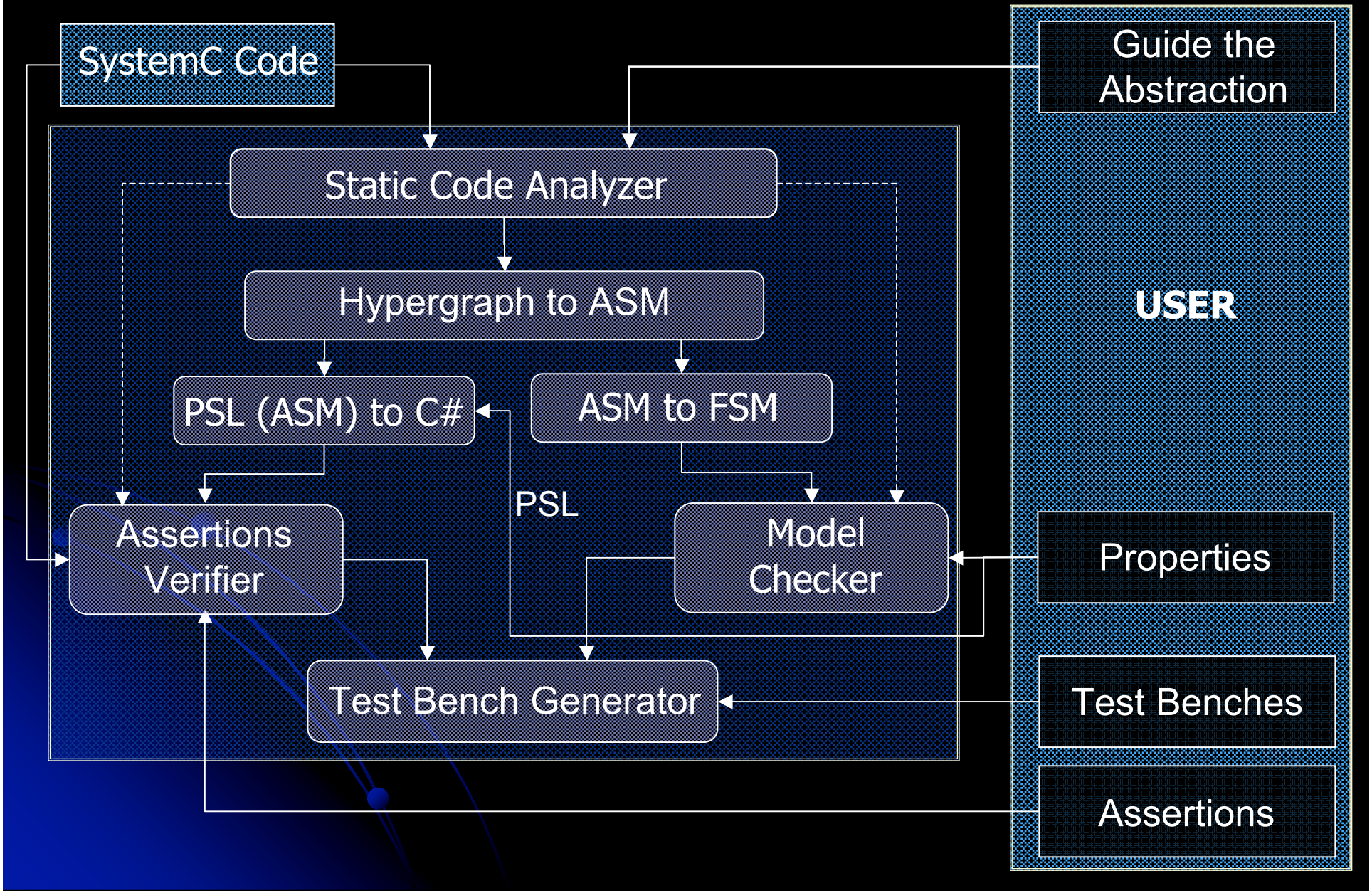
ABV: A Simple Approach



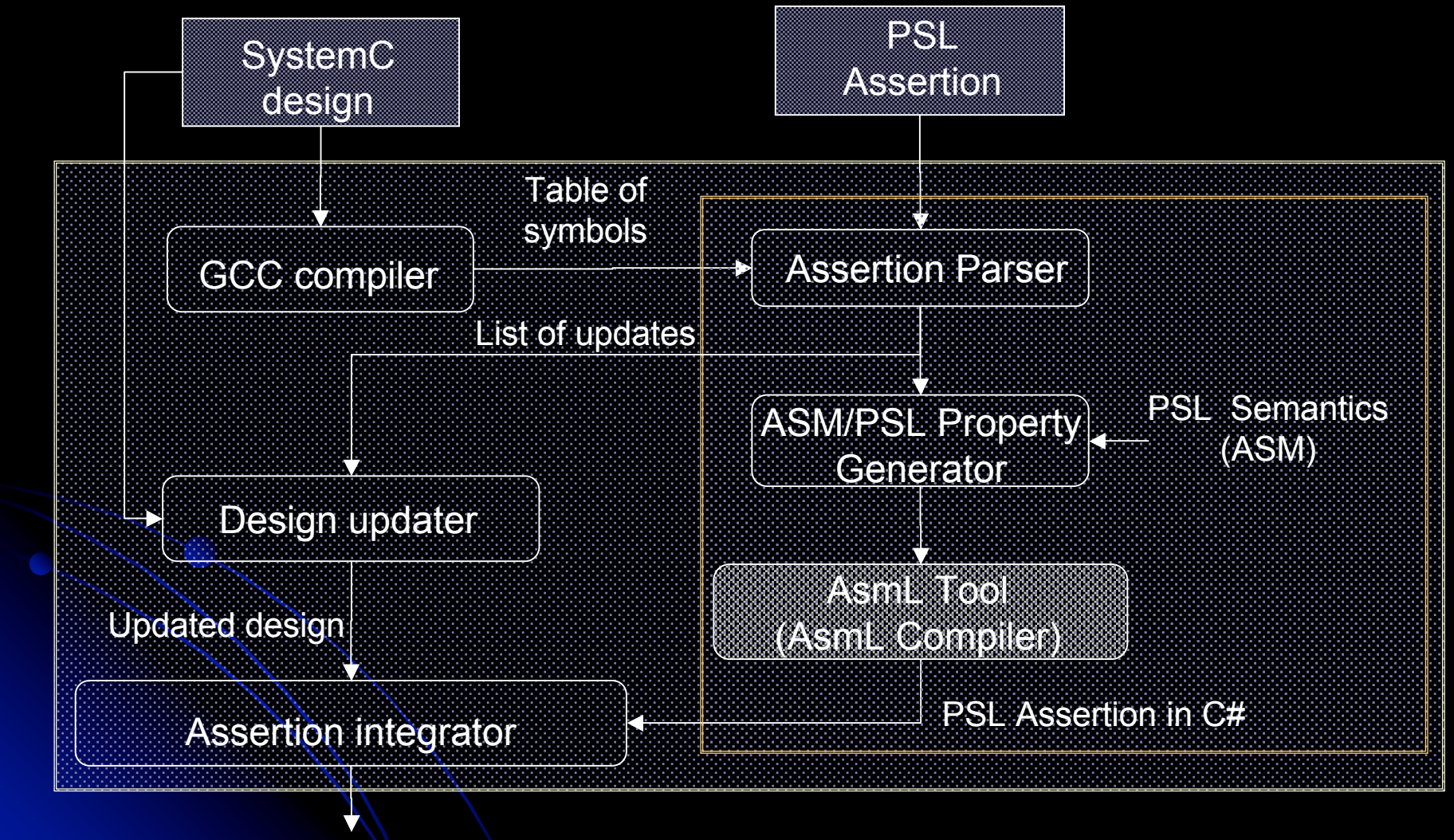
ABV: An Enhanced Approach



ABV: An ASM Based Approach



ABV: An ASM Based Approach



Related Work

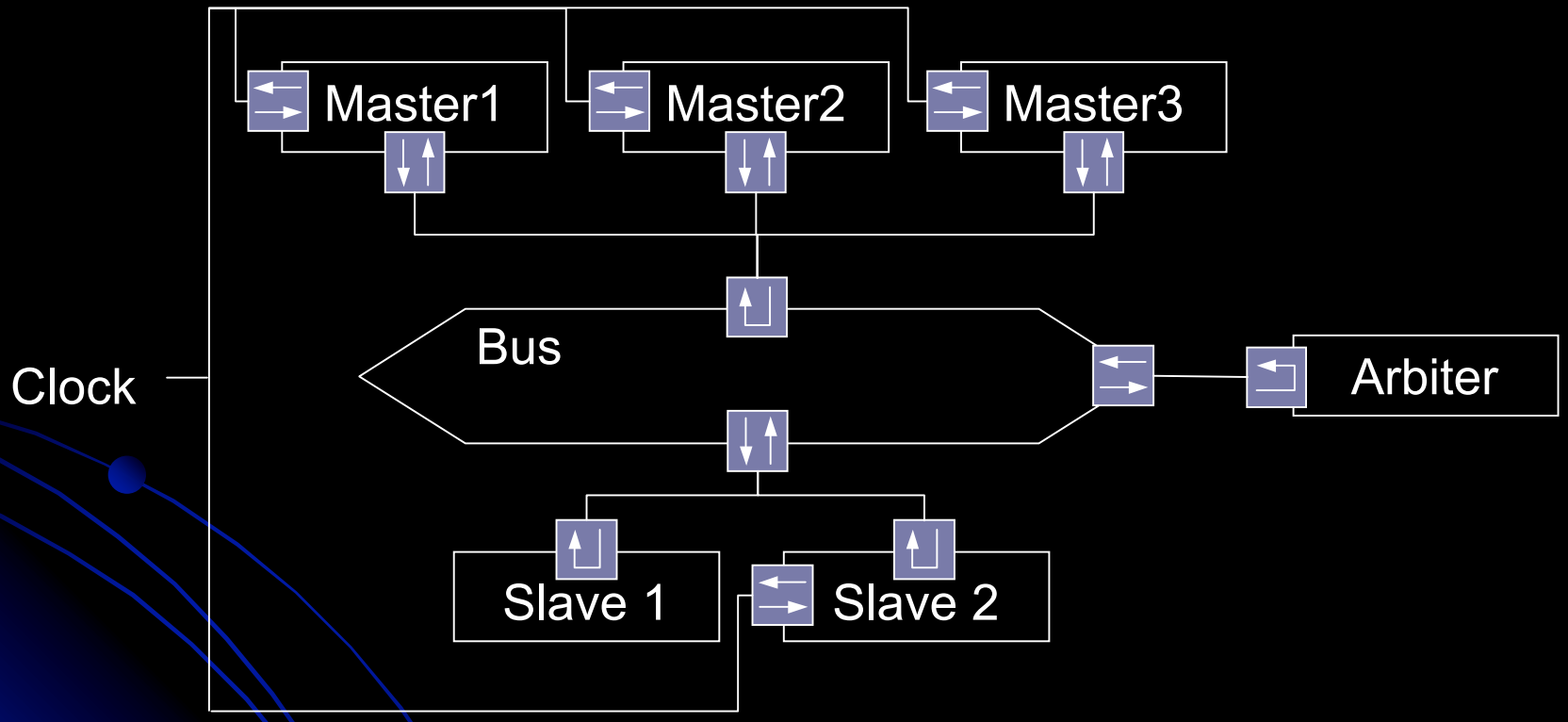
Finite-state verification of software:

- JAVA: BANDERA [Kansas University]: provides support for the extraction of safe, compact, finite-state models suitable for verification.
- C: SLAM [Microsoft], BLAST [University of Berkeley], etc.

Static code analysis of object-oriented languages:

- [Ferdinand'96]: Abstract debugging.
- OO languages abstraction:
  - [Chambers'90] : stack allocation and synchronization (Java programs)
  - [Vederine'00] : a platform for total analysis (both C++ and ML).

Case Study: Bus Structure



Bus Verification

Property 1:

NEVER((simple_bus.request == true) &&

(simple_bus.status != Bus_OK))

Property 2:

AFTER((simple_bus.request == true) &&

(simple_bus.request.block == Bus_OK))

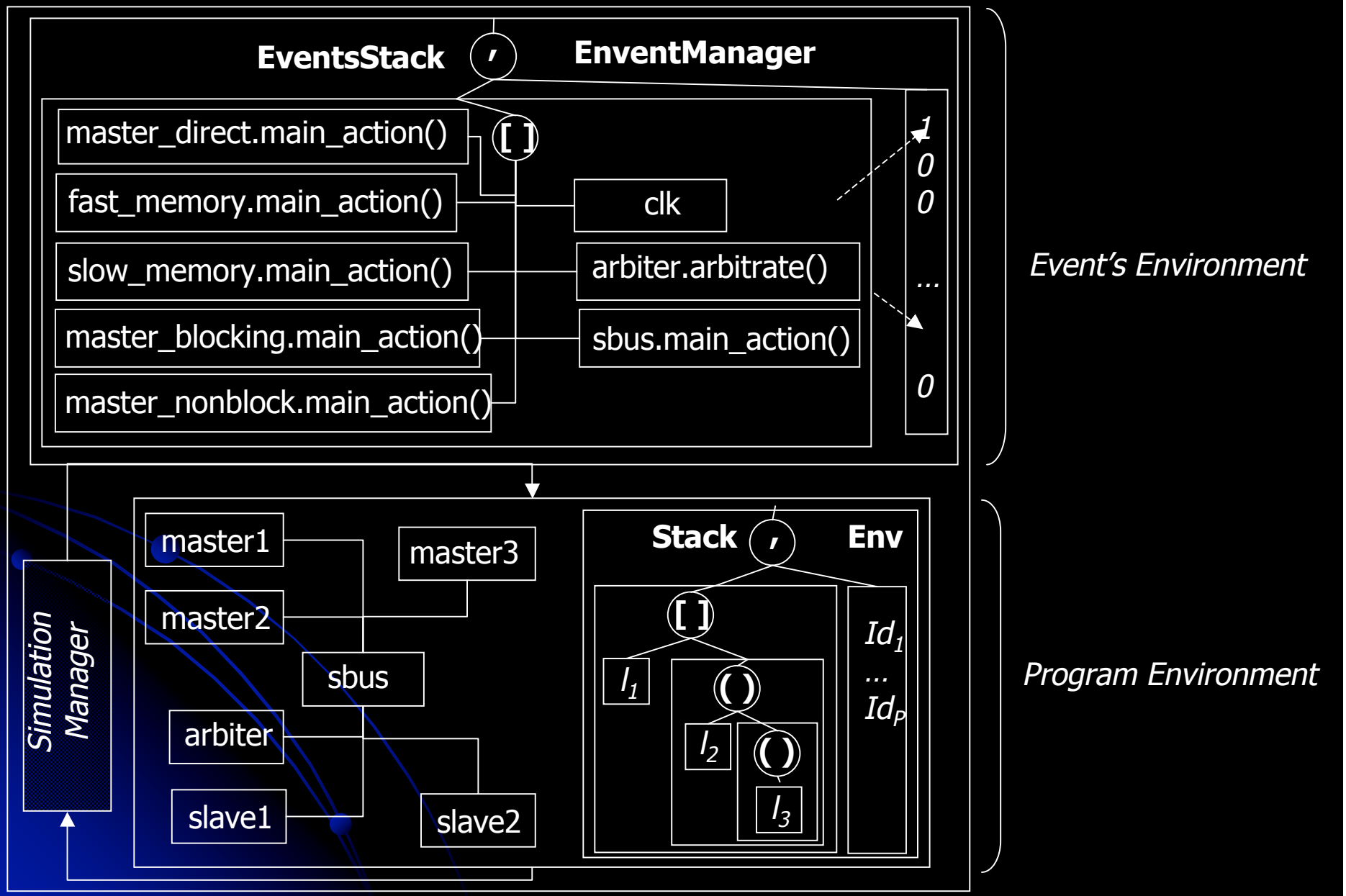
EVENTUALLY (simple_bus.status == BUS_BLOCK)

Property 3:

EVENTUALLY (simple_bus.status == BUS_OK)

Direct verification using FormalCheck: failed to complete after few minutes with out of memory problem!

Snapshot of the Hypergraph



Experimental Results

The main reductions concern:

- Transforming the SystemC simulator into a while loop.
- Reducing the packet to its packet header.
- Reducing some of the variables sizes.

Property	CPU Time	Memory (in MB)
P1	6:59:12	93.59
P2	15:23:02	183.91
P3	17:46:54	293.63

Assertions Definition

Assertion 1:

```
assert property1 @ (posedgeclk) (st == ack) && (flag == 1)  → !req[*8])
```

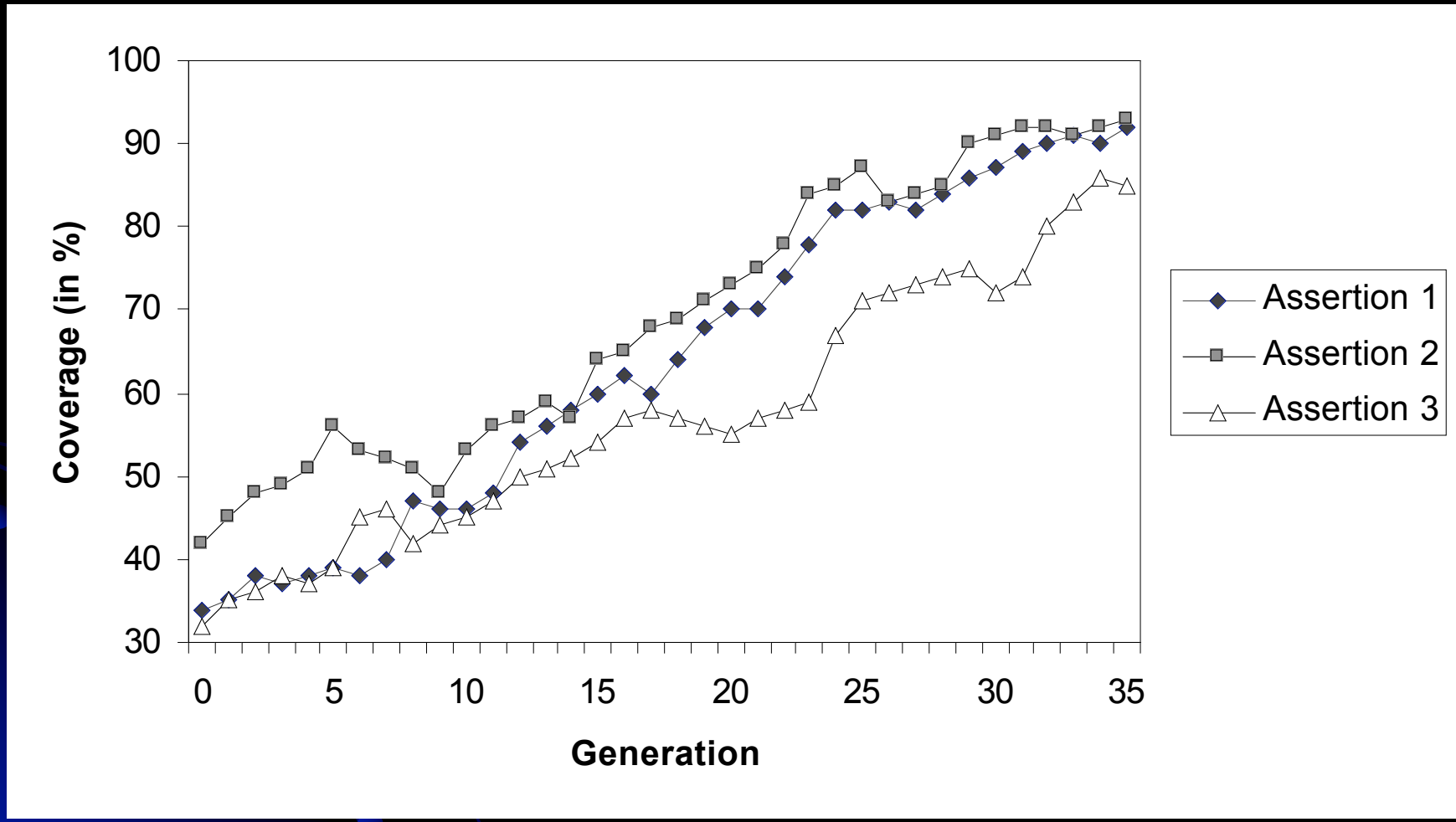
Assertion 2:

```
assert property2 @(posedgeclk)   (simple bus.request == true) &&
                    (master1.active||master2.active||master3.active)
```

Assertion 3:

```
assert property3 @(posedgeclk) (simple bus.request == true) &&
        (simple bus.request.nonblock == true)
                    → simple bus.status == BUS OK[*1])
```

Coverage as Function of the Generation

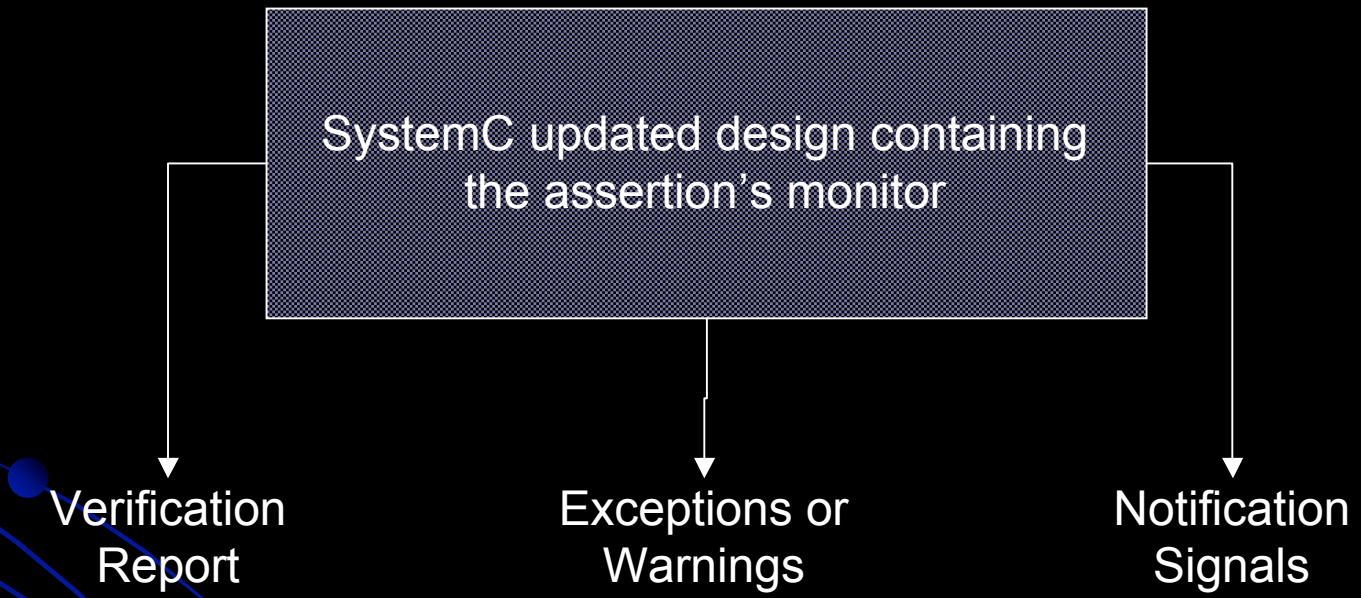


Genetic versus Random Coverage

	Assertion 1	Assertion 2	Assertion 3
Random Test (%)	10	8	12
Initialization of GA (%)	34	42	32
GA after 35 Iterations (%)	92	93	85

- Static code analysis refines the space of possible values.
- GA allows to offer better coverage of the assertions (more than 90% for Assertion 1).
- Mutation mechanism: overcomes the local maxima problem.

ABV Monitors Output



Conclusion

SystemC functional verification:

- Difficult for complex SoCs.
- Verification must consider IPs and their interaction.
- SystemC software functionality testing is inherently difficult.

→ Need to develop SystemC verification methodologies.

Our solution: combination of several techniques:

- Static code analysis.
- Model checking.
- Assertion based verification.
- Interface SystemC to existing tools through ASM semantics.

For any further details, visit the project webpage at:
System-on-Chip Verification <http://hvg.ece.concordia.ca/Research/SoC/>

Thanks!

Hardware Verification Group
2004