HAVEN: An Open Framework for FPGA-Accelerated Functional Verification of Hardware

Marcela Šimková Ondřej Lengál Michal Kajan

Brno University of Technology

Czech Republic

Haifa Verification Conference 2011

December 8, 2011

Motivation

Motto

It has been observed that verification becomes a major bottleneck in hardware design development, up to 80 % of the overall development cost and time.

— R. Drechsler *et al*, *Advanced Formal Verification*

Motto

The design and testing of an advanced microprocessor chip is among the most complex of all human endeavors.

— John Barton, Intel vice-president

Verification

Hardware verification approaches:

Verification

Hardware verification approaches:

- simulation and testing,

Verification

Hardware verification approaches:

- simulation and testing,
- formal analysis and verification,

Verification

Hardware verification approaches:

- simulation and testing,
- formal analysis and verification,
- functional verification.

Functional Verification

Process that checks whether a model of the system (DUT: Design Under Test) respects the specification.

Functional Verification

Process that checks whether a model of the system (DUT: Design Under Test) respects the specification.

+ Additional verification techniques:

Functional Verification

Process that checks whether a model of the system (DUT: Design Under Test) respects the specification.

+ Additional verification techniques:
  - constrained-random stimulus generation,

Functional Verification

Process that checks whether a model of the system (DUT: Design Under Test) respects the specification.

+ Additional verification techniques:
  - constrained-random stimulus generation,
  - coverage-driven verification,

Functional Verification

Process that checks whether a model of the system (DUT: Design Under Test) respects the specification.

+ Additional verification techniques:
  - constrained-random stimulus generation,
  - coverage-driven verification,
  - assertion-based verification,

Functional Verification

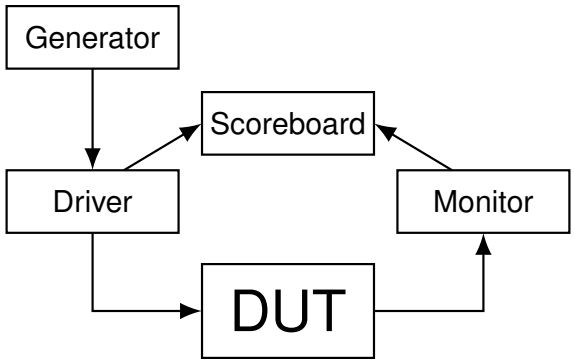
Process that checks whether a model of the system (DUT: Design Under Test) respects the specification.

+ Additional verification techniques:
  - constrained-random stimulus generation,
  - coverage-driven verification,
  - assertion-based verification,
  - self-checking mechanisms.

Functional Verification

Process that checks whether a model of the system (DUT: Design Under Test) respects the specification.

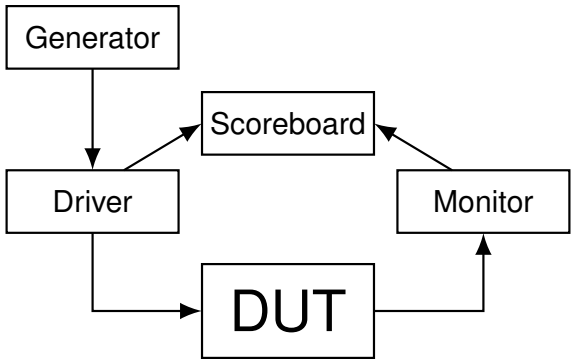
+ Additional verification techniques:
  - constrained-random stimulus generation,
  - coverage-driven verification,
  - assertion-based verification,
  - self-checking mechanisms.



Functional Verification

Process that checks whether a model of the system (DUT: Design Under Test) respects the specification.

+ Additional verification techniques:
  - constrained-random stimulus generation,
  - coverage-driven verification,
  - assertion-based verification,
  - self-checking mechanisms.



- Duration of the verification process is a problem!

Need of Acceleration

- Simulation of inherently parallel hardware is extremely slow when compared to the speed of real hardware.

Need of Acceleration

- Simulation of inherently parallel hardware is extremely slow when compared to the speed of real hardware.

- Moreover, this disproportion widens with:
  - increasing complexity of the hardware system,
  - growing number of tested input vectors.

Acceleration Approaches

- Current solution: use proprietary emulators
  - Mentor Graphics' Veloce, Cadence's TBA, ...
  - Full integration with a simulator.
  - Internal signals visibility, assertion and coverage analysis.
  - Issues:
    - expensive,
    - limited frequency (in the order of MHz).

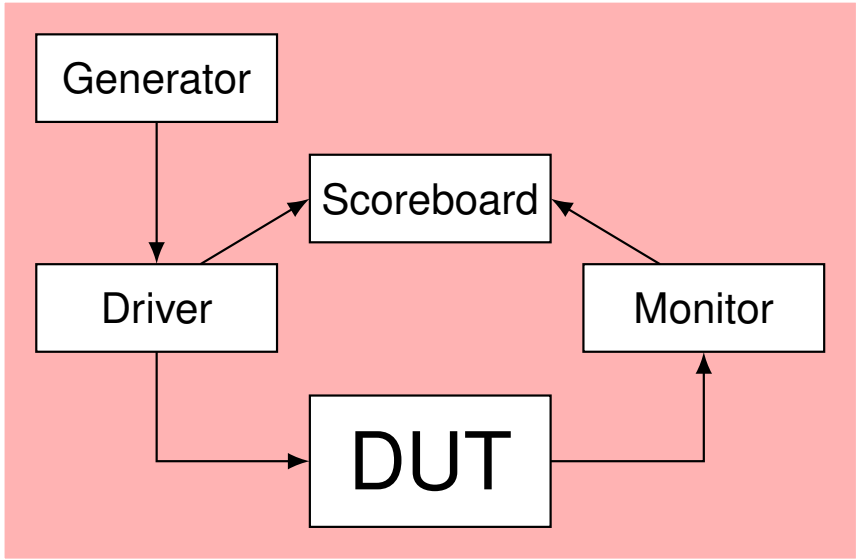
Acceleration Approaches

- Current solution: use proprietary emulators
  - Mentor Graphics' Veloce, Cadence's TBA, ...
  - Full integration with a simulator.
  - Internal signals visibility, assertion and coverage analysis.
  - Issues:
    - expensive,
    - limited frequency (in the order of MHz).

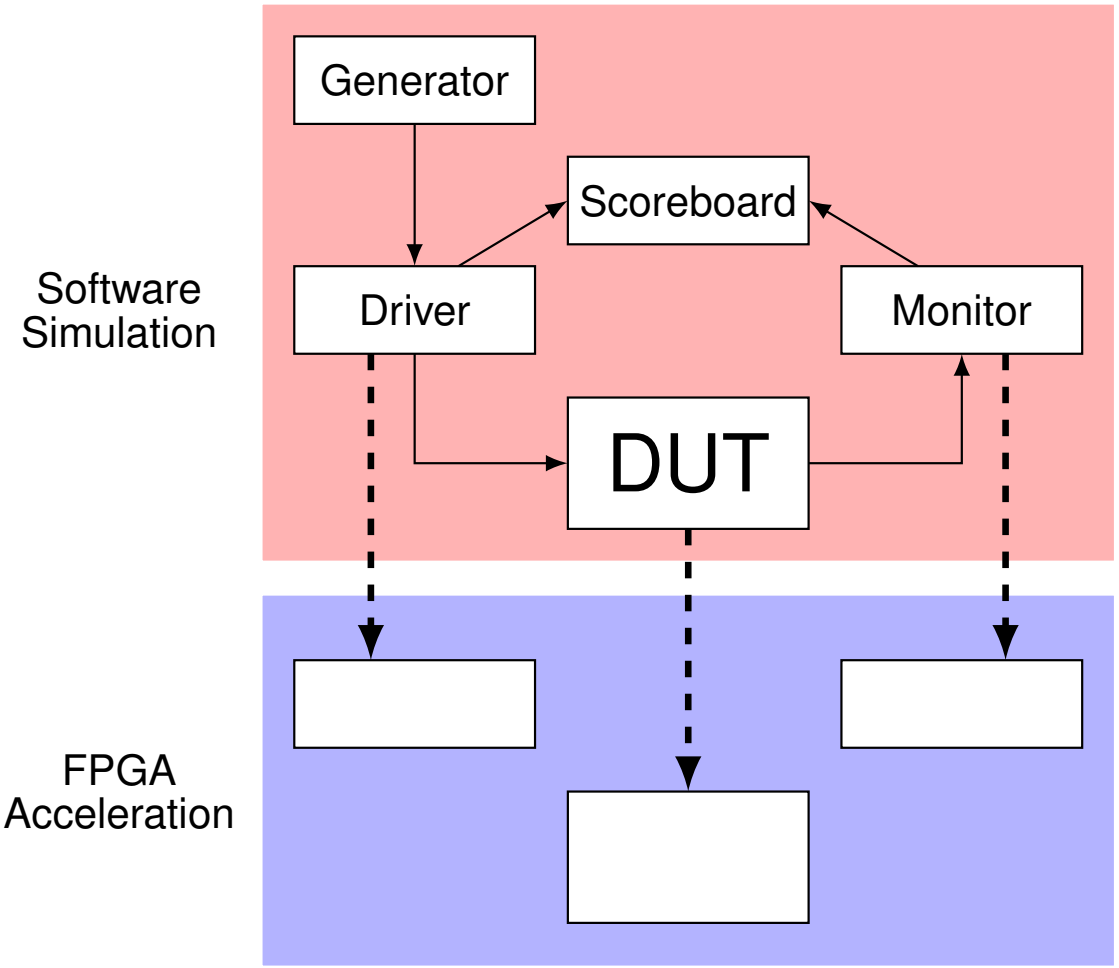
- We propose
  - HAVEN: Hardware Accelerated Verification ENvironment
    - Allows acceleration of functional verification by moving some components of the verification environment into FPGA (field-programmable gate array).
    - Runs at the frequency limited by the FPGA ($\sim$ 100 MHz).
    - Freely available and open source
      `http://www.fit.vutbr.cz/~isimkova/haven/`

HAVEN Design

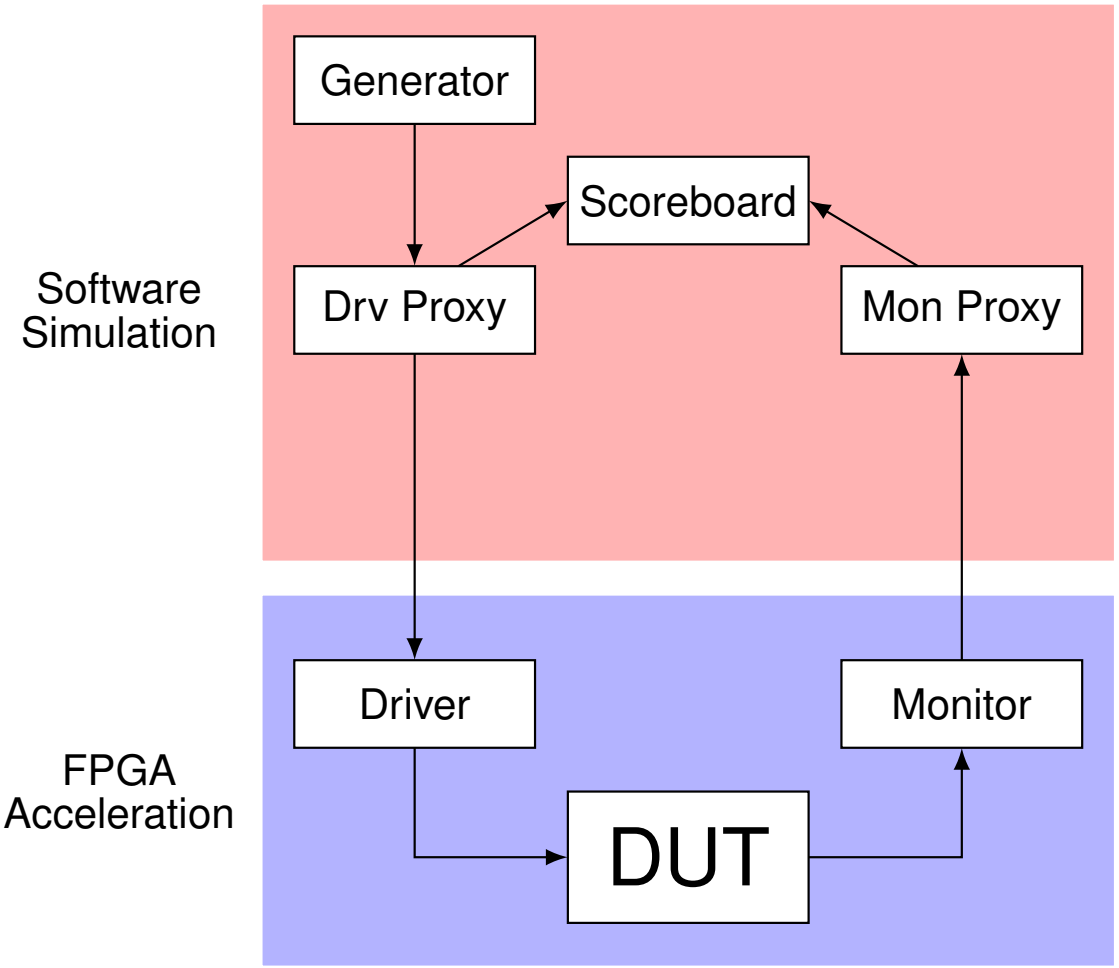
Software Simulation



HAVEN Design



HAVEN Design



The HAVEN Framework

- Two versions available:

The HAVEN Framework

- Two versions available:

Non-accelerated version

- DUT is simulated in a software simulator,
- software verification environment,
- perfect debugging environment in a simulator,
- used in the initial phase of the verification process,
- suitable for small number of input transactions (up to thousands).

Accelerated version

- DUT is synthesized and placed into FPGA,
- software-hardware verification environment,
- limited debugging capabilities,
- verification of complex systems,
- suitable for large number of input transactions (millions).

- switching between them done by setting a single parameter.

Features of HAVEN

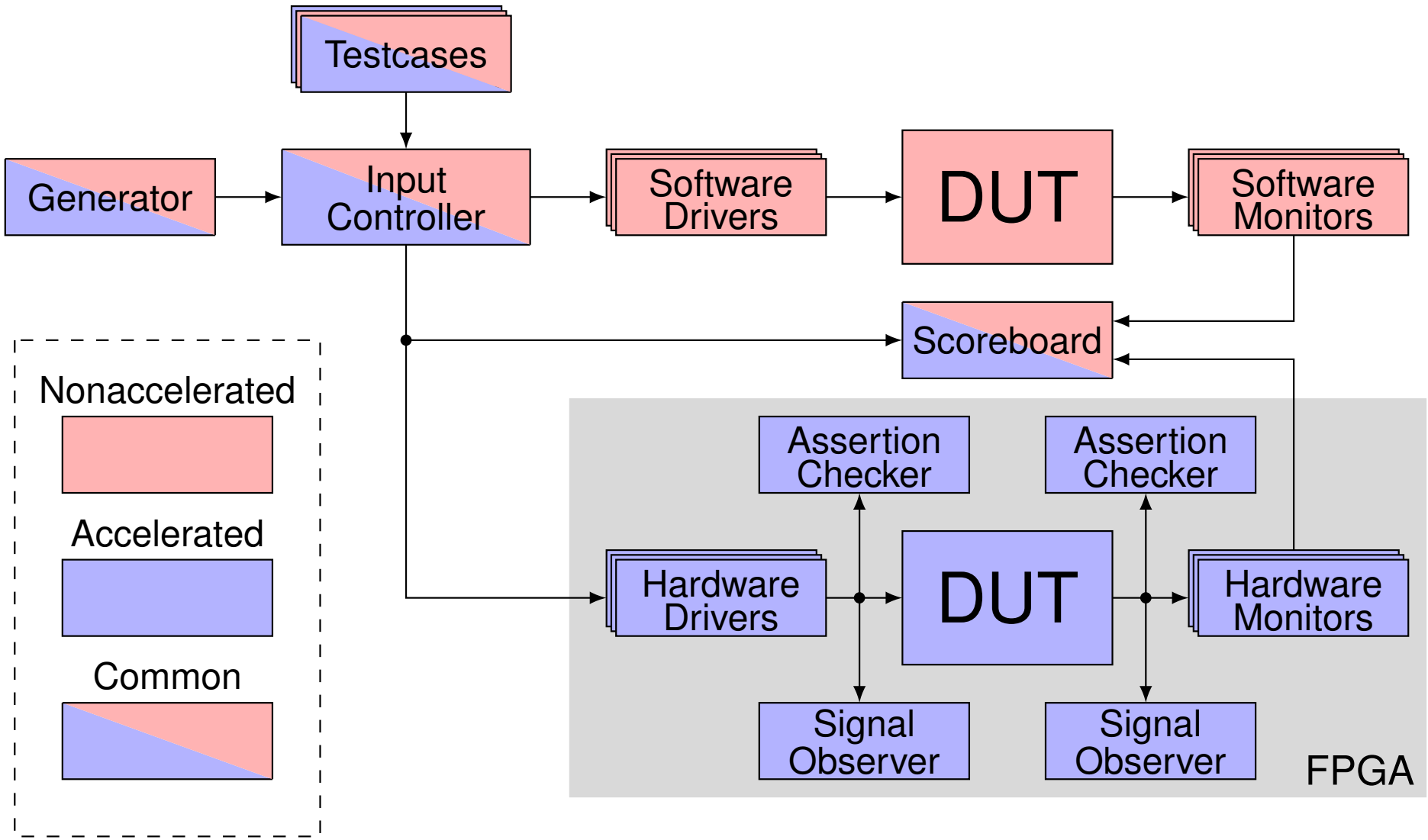
- Acceleration:
  - the NetCOPE platform with COMBOv2 (`http://www.liberouter.org/netcope/`)
  - FPGA: Xilinx Virtex-5
- High level of abstraction (easy to adapt/extend).

- Cycle-accurate behaviour of verification runs in both versions.
  - Obtained by enabling/disabling the clock signal for the DUT.
- Acceleration of verification of arbitrary synchronous units.
- For an FPGA system, verifies directly the system, not a *model*.

Features

- Features implemented to make the framework competitive to commercial solutions:
  - Assertion Checkers
    - Assertion analysis during a verification run in the FPGA.
    - A linear temporal formula is implemented as a finite state machine (Büchi automaton).
  - Signal Observers
    - Observes values of signals during verification run in the FPGA.
    - Stores values into a VCD file.
    - Enables debugging of failing scenarios directly in hardware.

Architecture of the Framework

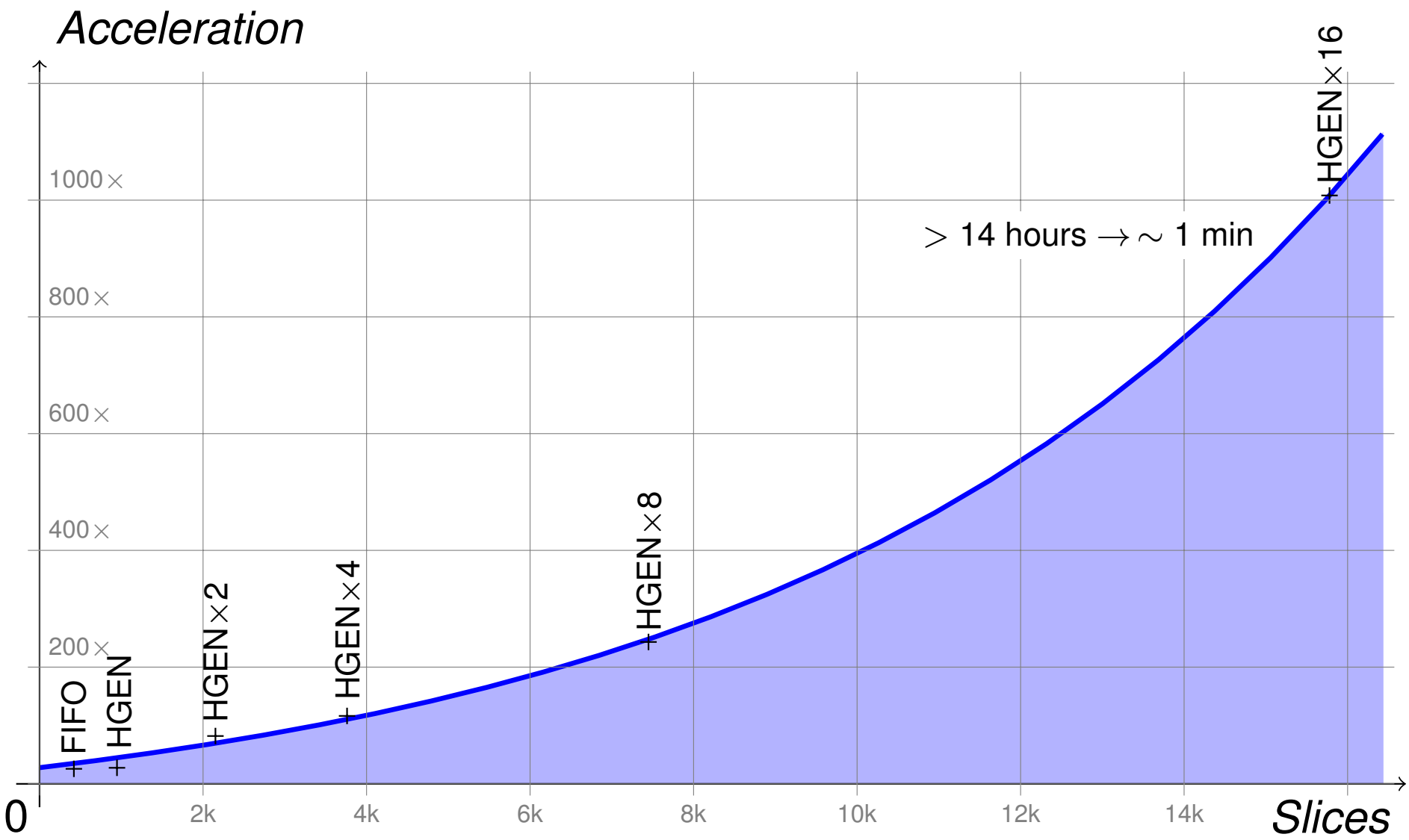


Results of Experiments

- FIFO: a simple buffer.
- HGEN: computes the Bob Jenkins's *Lookup2* hash function.
- HGEN$\times k$: a system with k (2,4,8,16) HGEN units.

Component	Slices
FIFO	420
HGEN	947
HGEN$\times$2	2,152
HGEN$\times$4	3,762
HGEN$\times$8	7,448
HGEN$\times$16	15,778

Experiments



Conclusion

- Framework that supports easy acceleration of functional verification in an FPGA.
- Accelerated version of framework retains assertion analysis and signal observability.
- Acceleration over 1000$\times$ achieved.
- Freely available and open source.

Future work

- Automated synthesis of SystemVerilog Assertions.
- Hardware-accelerated generation of test vectors.
- Providing the SCE-MI interface for communication between HW and SW.

Questions?