Hybrid Coverage Assertions for Efficient Coverage Analysis across Simulation and Emulation Environments

Hsuan-Ming Chou, Hong-Chang Wu, Yi-Chiao Chen, Jean Tsao, and Shih-Chieh Chang

National Tsing Hua University, Taiwan

Outline

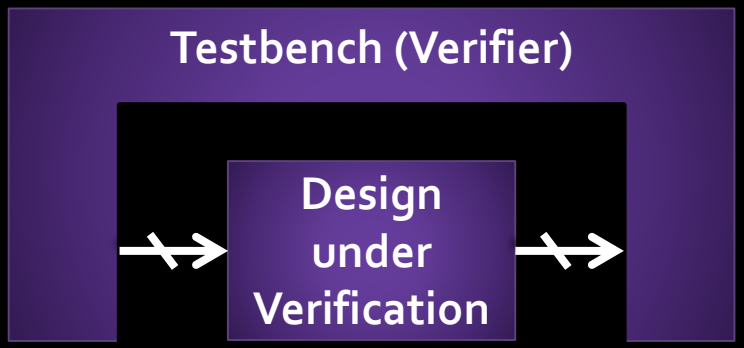
- Introduction
- Coverage Analysis In a Hardware-Accelerated Environment
- Optimization of Coverage Assertions
- Experimental Results
- Conclusions

Outline

- Introduction
- Coverage Analysis In a Hardware-Accelerated Environment
- Optimization of Coverage Assertions
- Experimental Results
- Conclusions

Coverage Metric

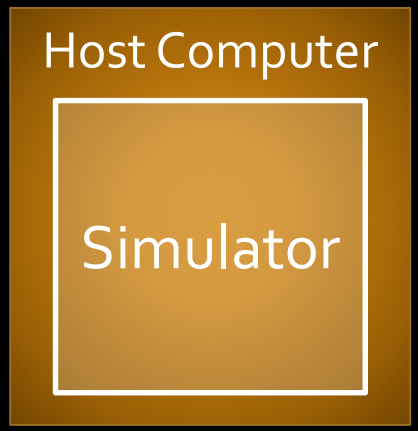
- Functional verification is a process to ensure that a design implements intended functionality.



- Coverage are used as a metric
  - Avoid unnecessary repetitions of verification.
  - Quantify the completeness of the test suites.

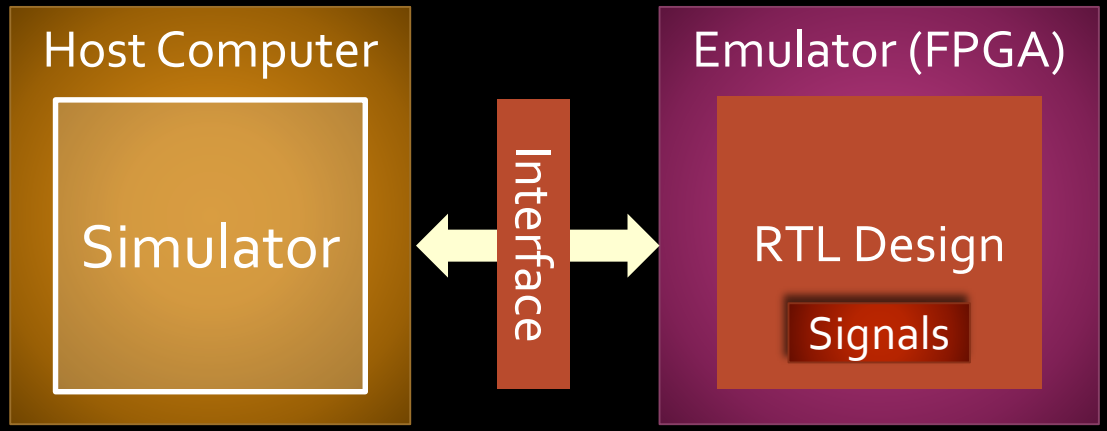
Coverage Analysis in Simulator

- Traditionally, coverage metrics are analyzed in a simulation environment.



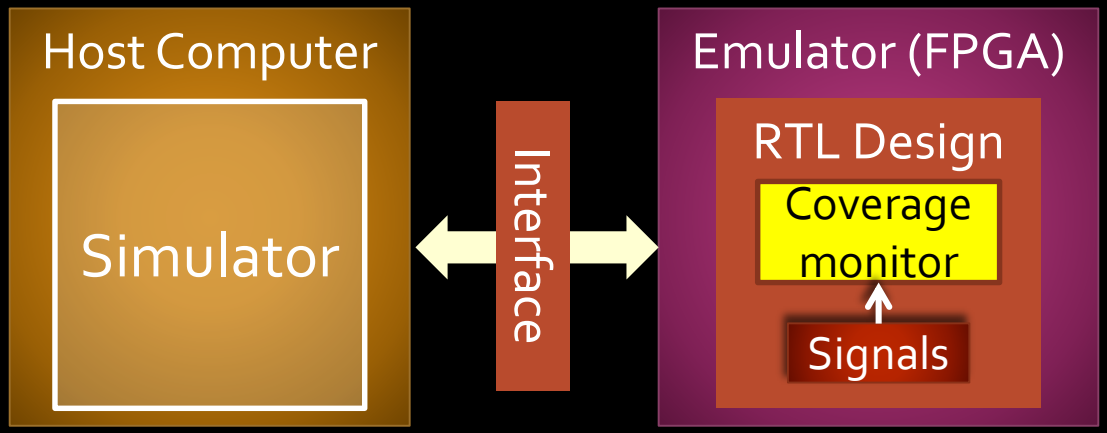
Coverage Analysis in Emulator

- Due to slow simulation speed, we can emulate the designs using hardware accelerator (FPGA).
- Since many signals in emulator are unobservable, conventional coverage analysis methods for simulator cannot be directly applied to emulator.



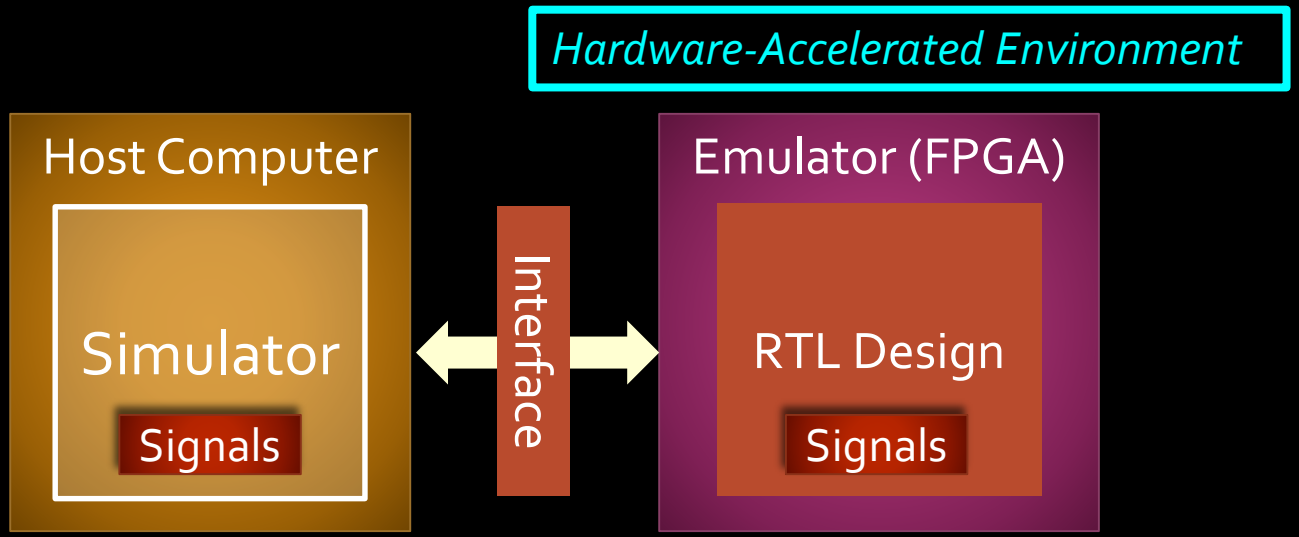
Previous Works of Hardware Coverage Monitor

- Several previous works proposed hardware coverage monitors
  - *Balston, Kyle, et al."Post-Silicon Code Coverage for Multiprocessor System-on-Chip Designs" (IEEE Computers 2013)*
  - *Grinwald, Raanan, et al. "User defined coverage—a tool supported methodology for design verification." (DAC 1998)*
  - *Bojan, Tommy, et al."Functional Coverage Measurements and Results in Post-Silicon Validation of Core™2 Duo Family" (HLDVT 2007)*



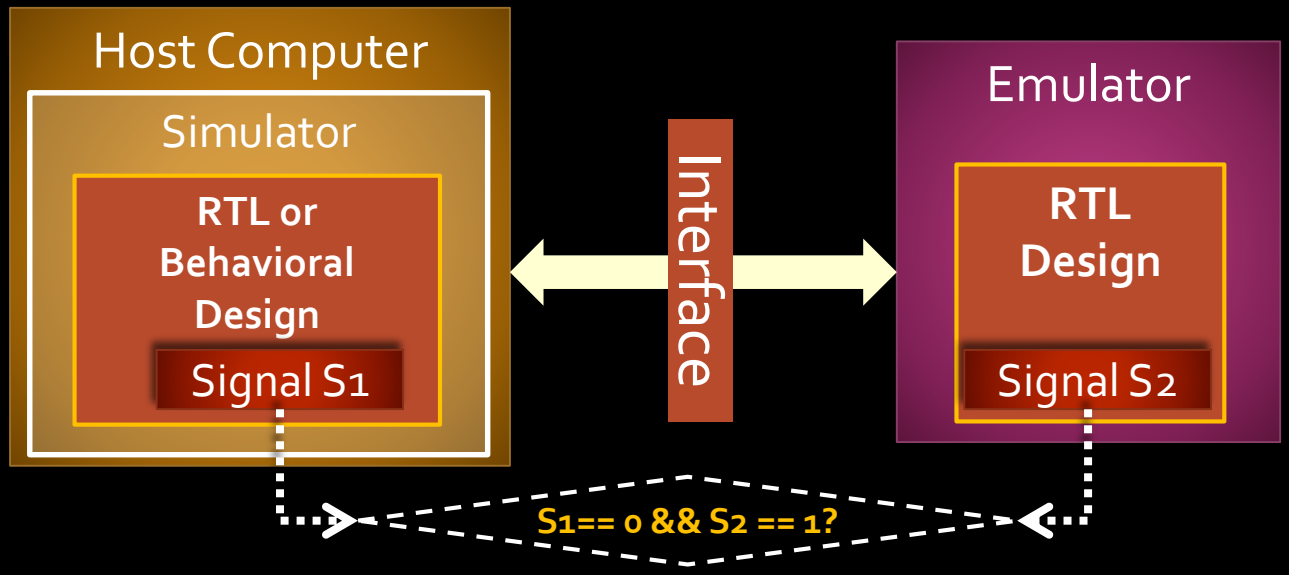
Coverage Analysis in Hardware-Accelerated Environment

- Only parts of the design are synthesized to the emulator:
  - Behavioral code is non–synthesizable.
- Emulator has poor observability.
- The designs are put in both simulator and emulator.



Coverage Analysis in Hardware-Accelerated Environment

- Coverage signals may be analyzed across a simulator and an emulator.
- Neither conventional coverage techniques nor hardware coverage monitors can be applied.



Motivation

- We propose a comprehensive methodology to analyze coverage in a hardware-accelerated (emulator + simulator) environment.
- Our methodology uses modified assertions.
  - Assertions can provide rich expressions to detect coverage events.

Contributions

- We propose different types of coverage assertions extending conventional assertions to analyze coverage.
- An Assertion Operation Graph (AOG) is proposed to represent the operations of coverage assertions.
- A set of graph-based algorithms are proposed to minimize the hardware and performance overheads of coverage assertions.

Outline

- Introduction
- Coverage Analysis In a Hardware-Accelerated Environment
- Optimization of Coverage Assertions
- Experimental Results
- Conclusions

Coverage Event and Assertion

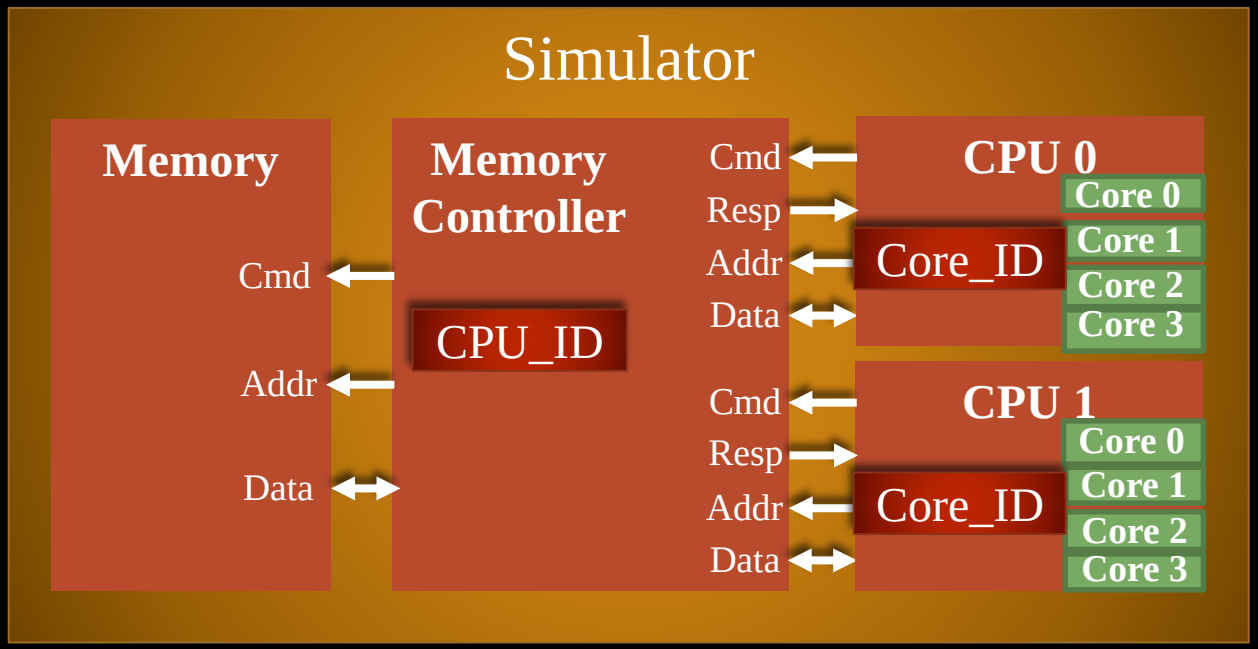
- In hardware-accelerated environment, a coverage event may consist of signals in the simulator, emulator, or even both.
- Three types of coverage events:
  - Simulated event
  - Emulated event
  - Hybrid event

- An assertion used to detect a coverage event is called a coverage assertion.
- Three types of coverage assertions:
  - Simulated assertion (S_Cassert)
  - Emulated assertion (E_Cassert)
  - Hybrid assertion (H_Cassert)

Simulated Coverage Assertion

- A coverage assertion for detecting the coverage event inside the simulator.
  - Event: "Core 1 in CPU 0 successfully accesses memory"
  - Assertion: *S_Cassert S1(CPU_ID == 0 && CPU0.Core_ID == 1);*

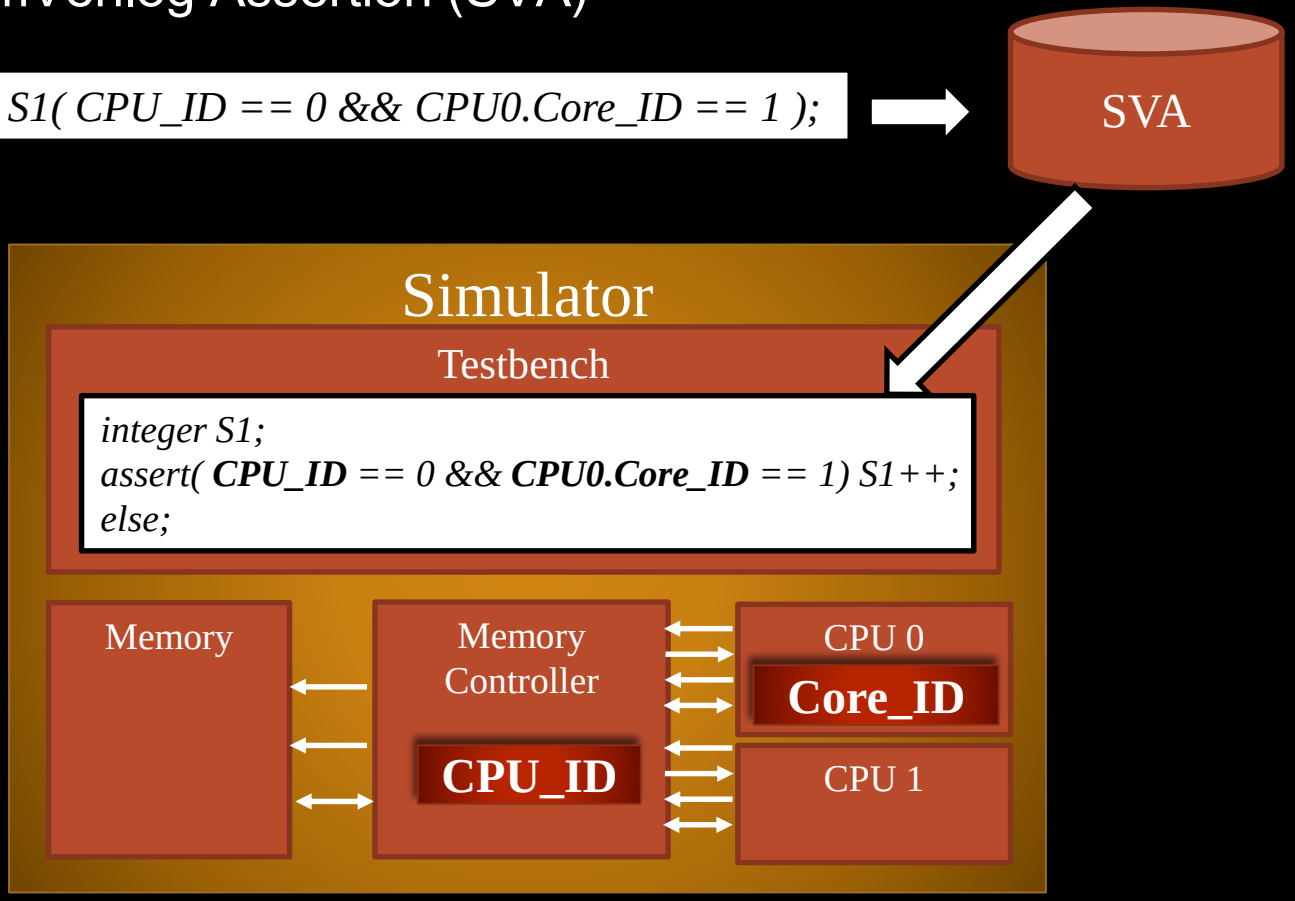
Attribute	Values
CPU_ID (1 bit)	0,1
Core_ID (2 bits)	0,1,2,3



Impl. of Simulated Coverage Assertion

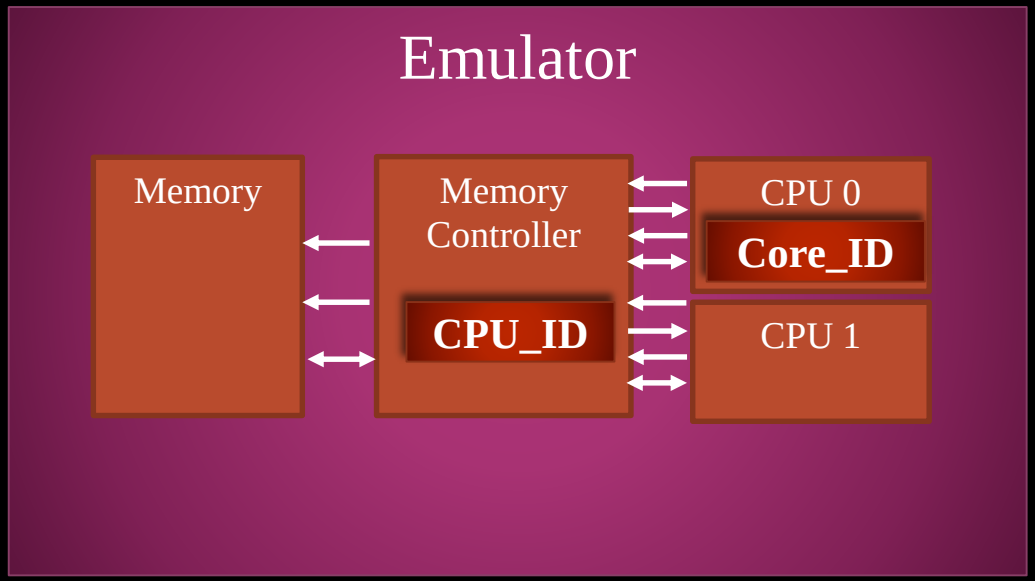
- Existing assertion languages can be used
  - SystemVerilog Assertion (SVA)

S_Cassert S1(CPU_ID == 0 && CPU0.Core_ID == 1);

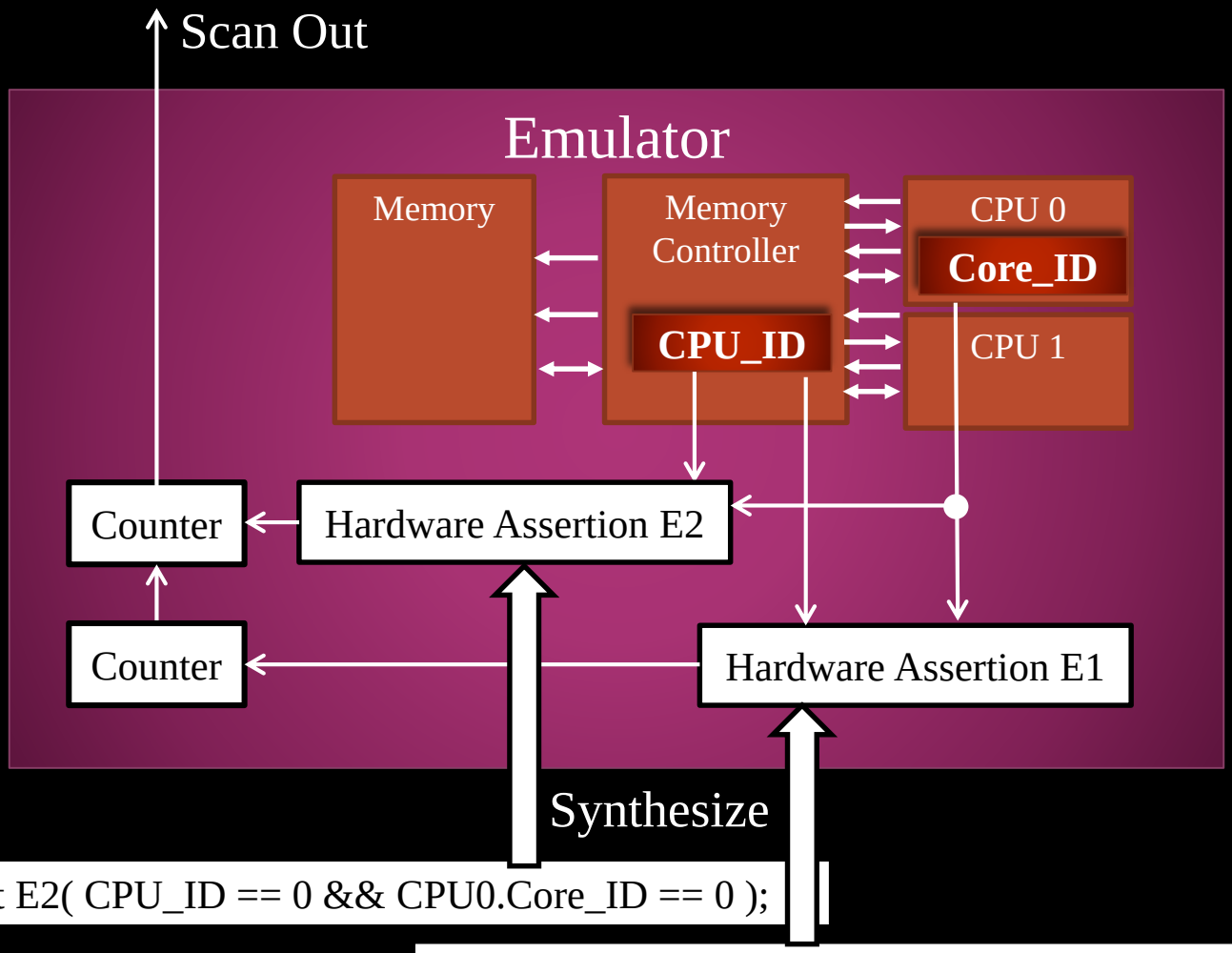


Emulated Coverage Assertion

- A coverage assertion for detecting the coverage event inside the emulator.
  - Event: "Core 1 in CPU 0 successfully accesses memory"
  - Assertion: *E_Cassert E1(CPU_ID == 0 && CPU0.Core_ID == 1);*

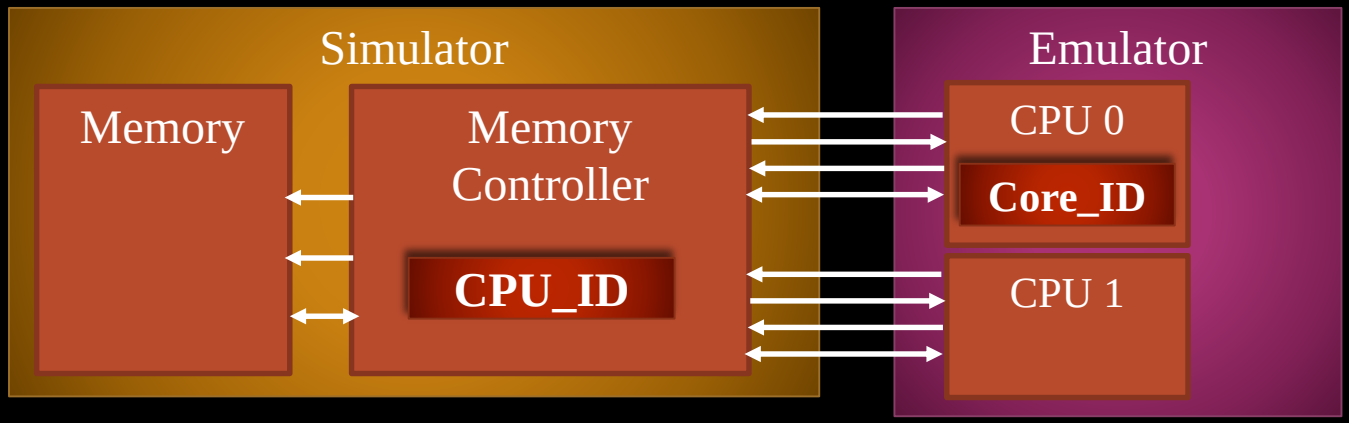


Impl. of Emulated Coverage Assertion

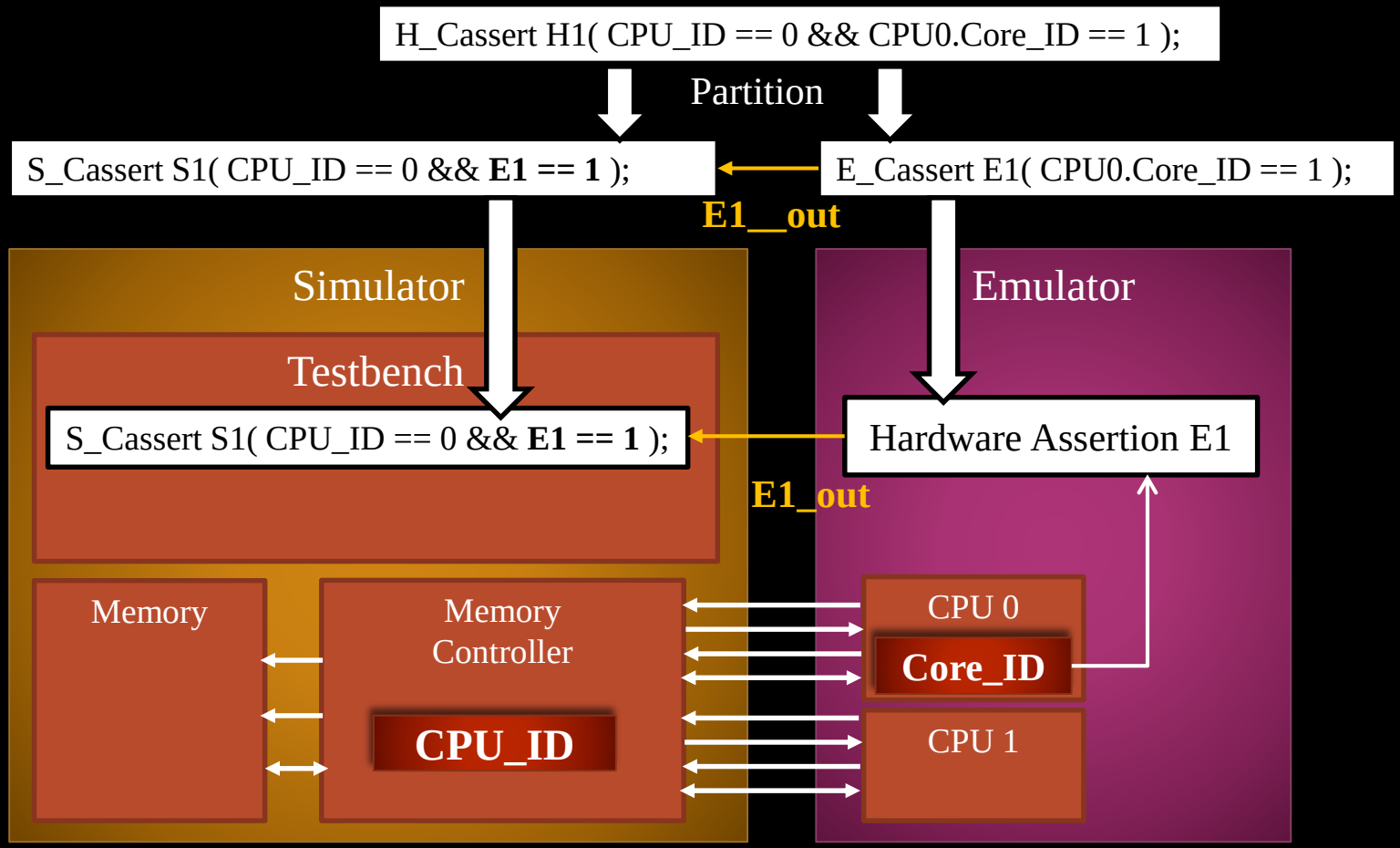


Hybrid Coverage Assertion

- Detect the coverage event composed of both simulated signals and emulated signals.
  - Event: "Core 1 in CPU 0 successfully accesses memory"
  - Assertion: *H_Cassert H1(CPU_ID == 0 && CPU0.Core_ID == 1);*
- A hybrid coverage assertion is decomposed into
  - Simulated coverage assertions
  - Emulated coverage assertions
  - Auxiliary signals



Impl. of Hybrid Coverage Assertion



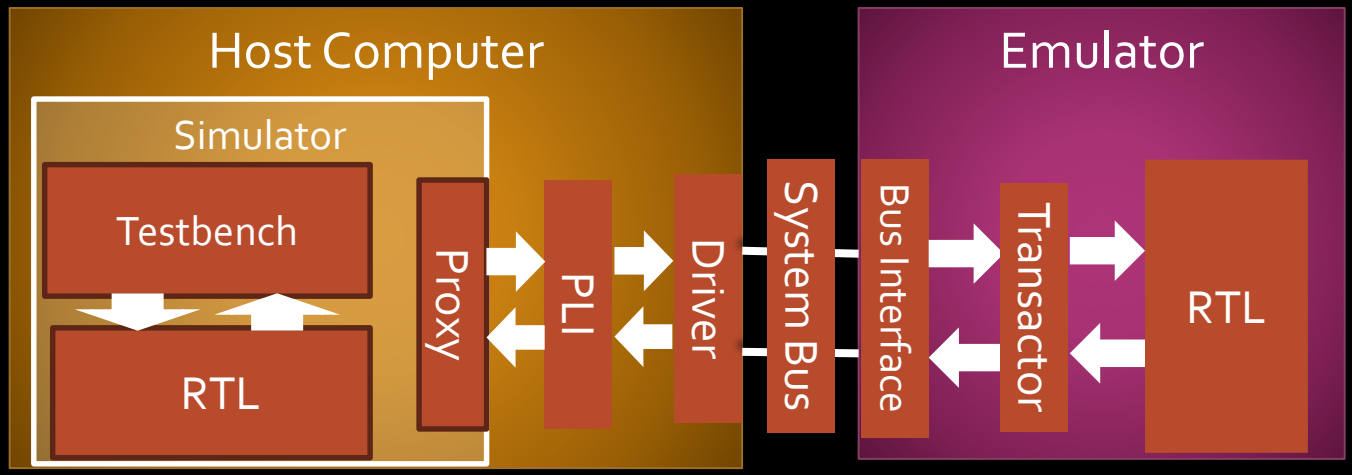
(Different ways of implementation may lead to different overheads)

Outline

- Introduction
- Coverage Analysis In a Hardware-Accelerated Environment
- Optimization of Coverage Assertions
- Experimental Results
- Conclusions

Overheads of Coverage Assertions

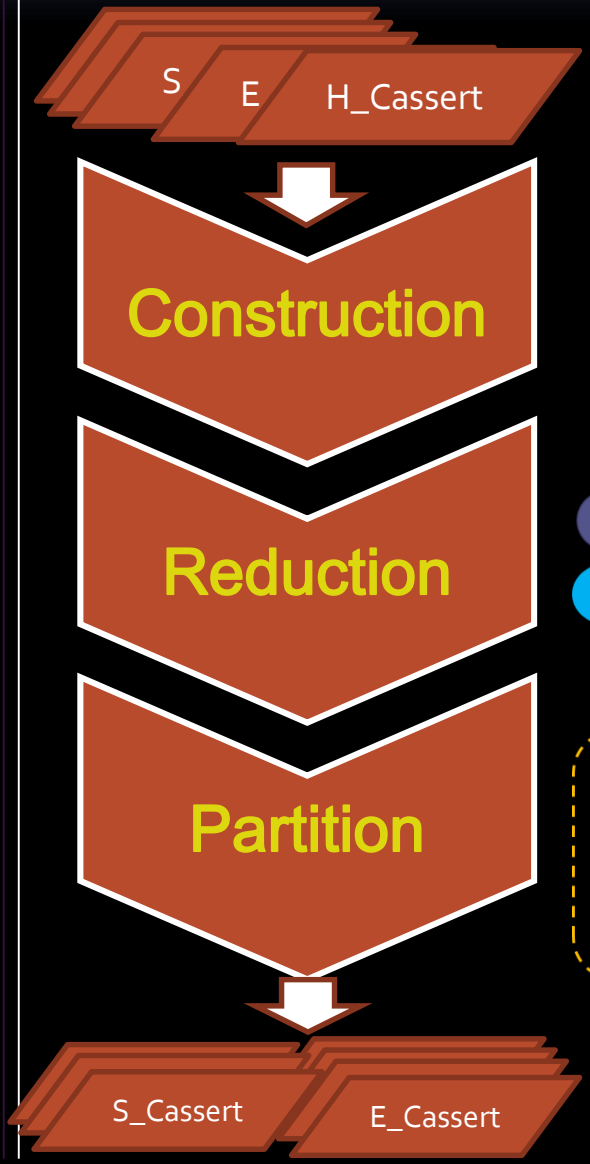
- Hardware cost
- Simulation time
- Synchronization time

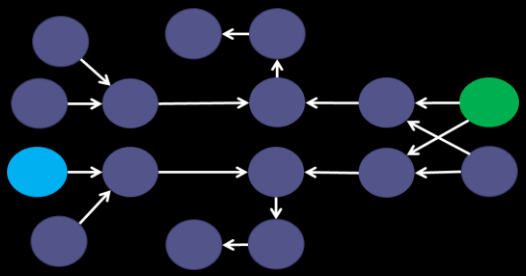


Problem Description

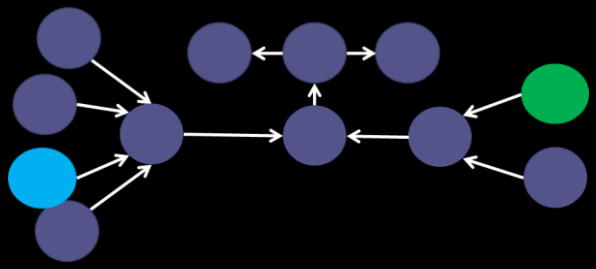
- Problem Description
  - Given: A set of coverage assertions
  - Goal: Reduce the hardware and performance overheads caused by coverage assertions

Optimization Flow

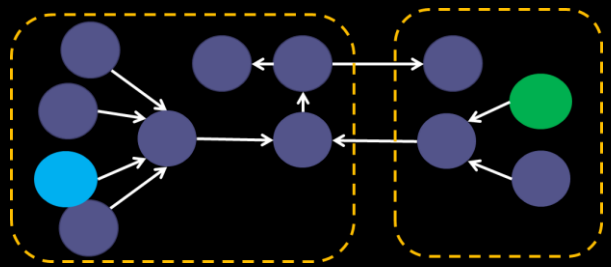




Assertion Operation Graph (AOG).



Reduce the nodes on AOG.

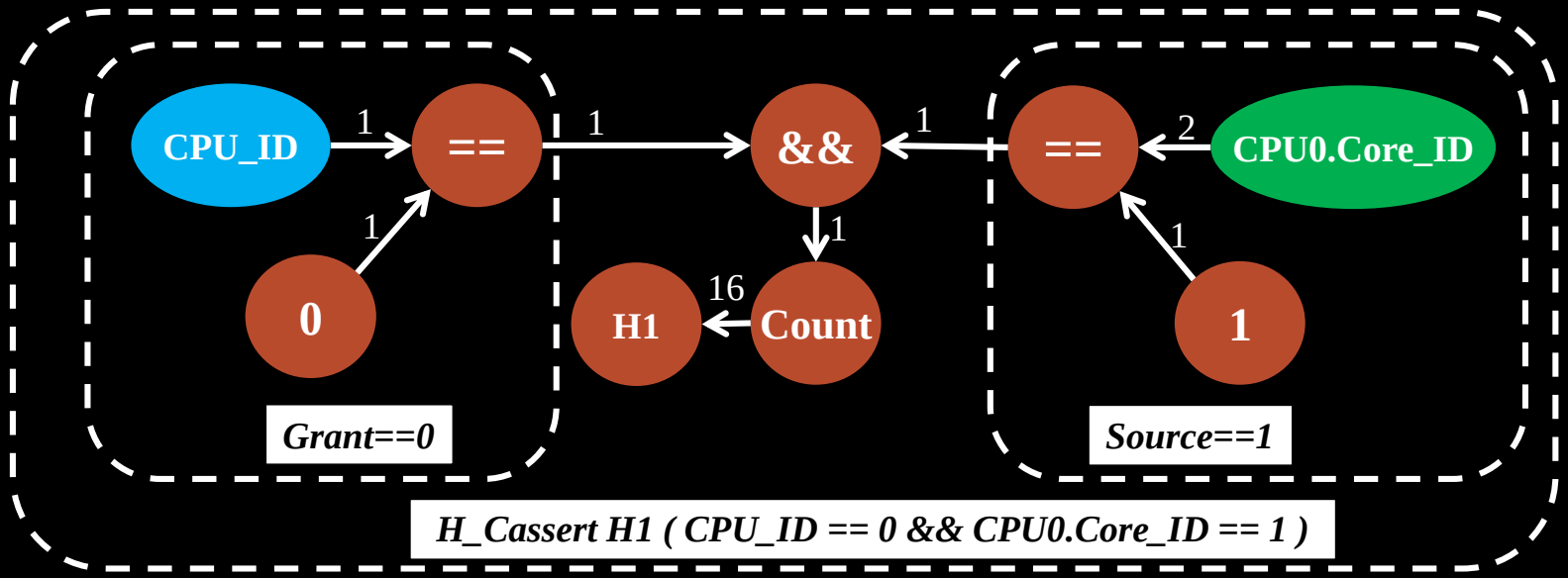


Partition the AOG into two sub-graph.

Construction → Reduction → Partition

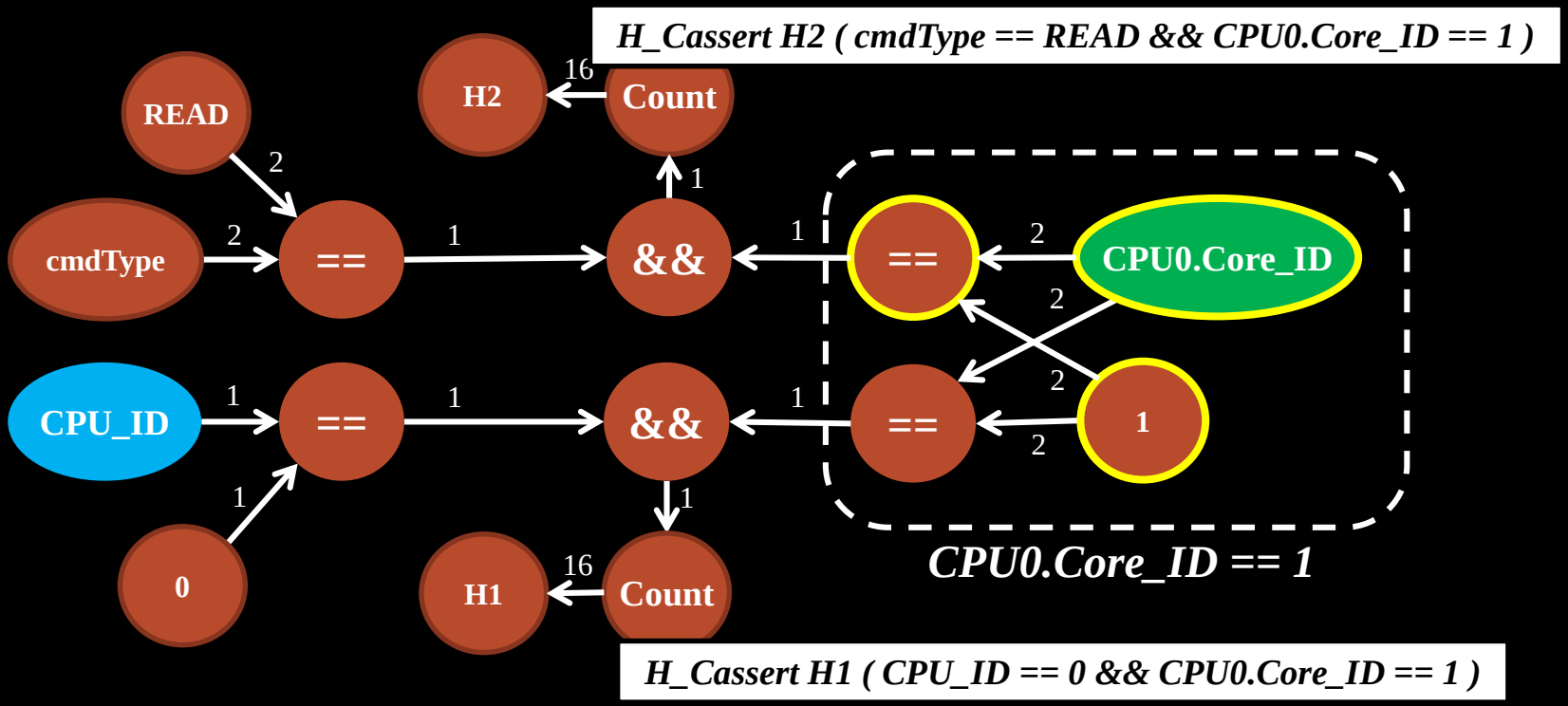
- ***Simulated Node*** : operation perform in simulator
- ***Emulated Node*** : operation perform in emulator
- ***Unassigned Node*** : operation has not been assigned

- Weight of each edge: data width



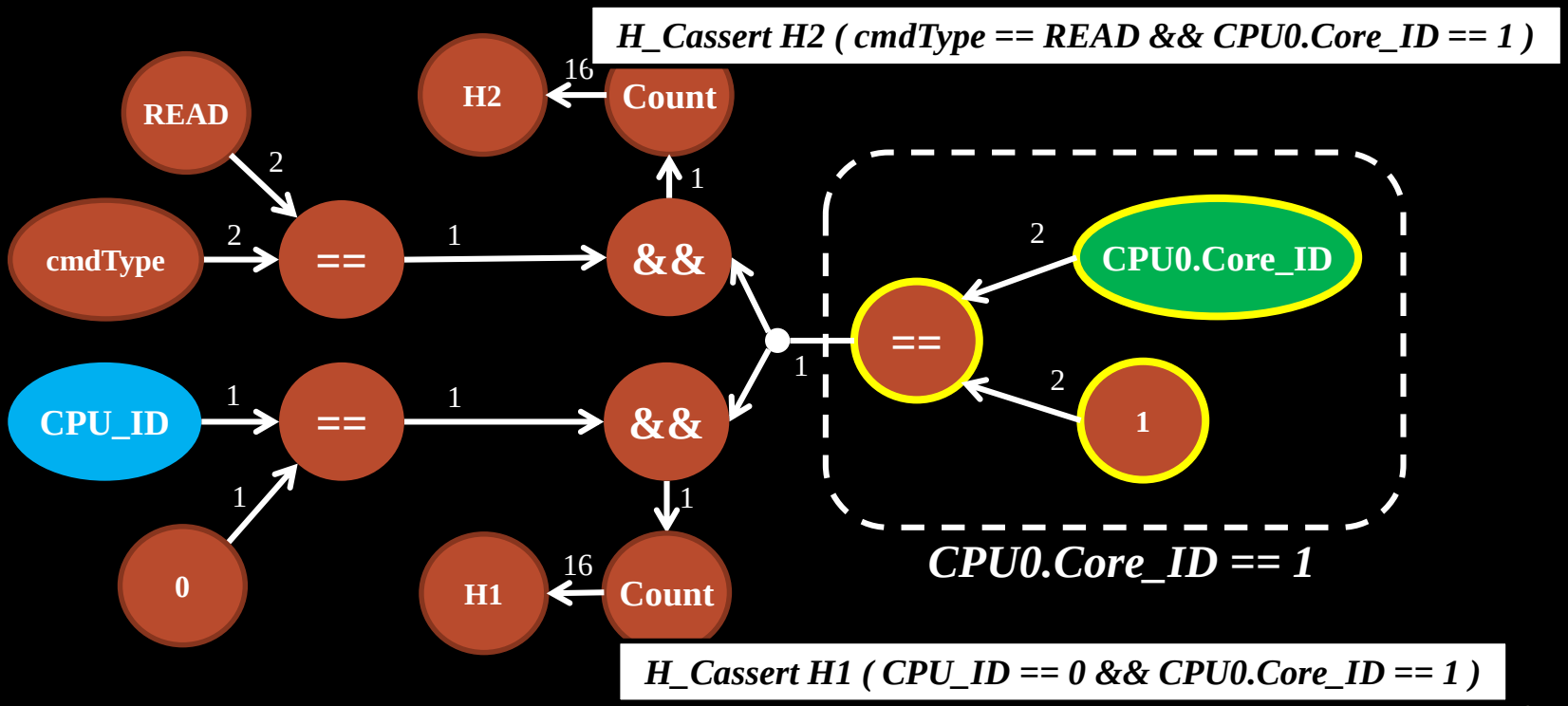
Construction Reduction Partition

- An AOG is similar to a data flow graph. The optimization approaches on the data flow graph can be modified to apply on an AOG.
  - e.g. Common Sub-expression Elimination



Construction Reduction Partition

- An AOG is similar to a data flow graph. The optimization approaches on the data flow graph can be modified to apply on an AOG.
  - e.g. Common Sub-expression Elimination



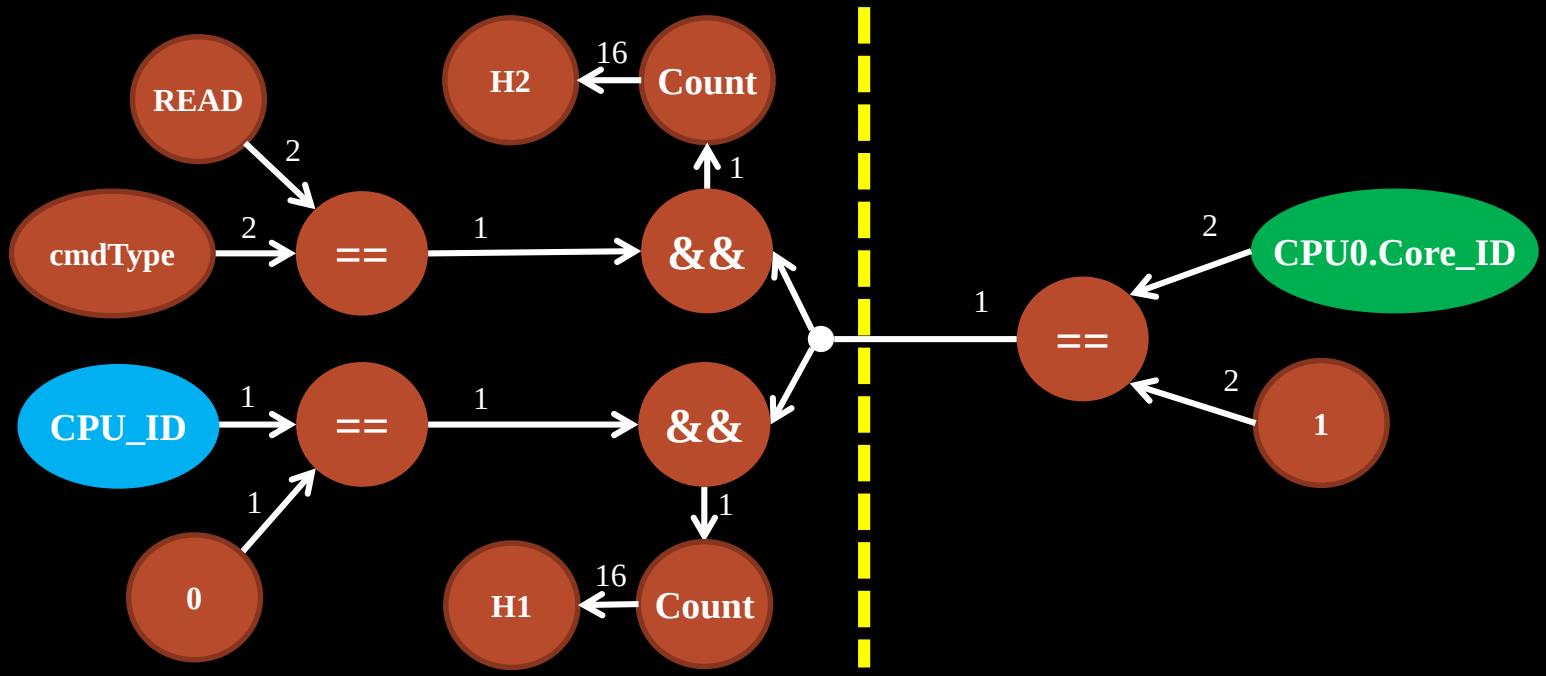
Construction → Reduction → **Partition**

- All unsigned nodes need to be assigned either to a simulator or an emulator.
- Careless assignment of those nodes can lead to high hardware overhead in the emulator or high total CPU time.
- Since synchronization time is considered as the most consuming part of the total CPU time, our partition algorithm focuses on optimizing synchronization time.

Construction → Reduction → Partition

- Different assignment results could lead to different synchronization overhead amounts.
- Our goal is to minimize the number of data bits needed for communication.

- This problem can be modeled as a constrained two-way partitioning problem.
- We first obtain an initial solution and then modify the Fiduccia–Mattheyses (FM) algorithm to solve this problem.



Outline

- Introduction
- Coverage Analysis In a Hardware-Accelerated Environment
- Optimization of Coverage Assertions
- Experimental Results
- Conclusions

Environment Setting

- Hardware-accelerated platform
  - Simulator: ISim simulator in Xilinx ISE
  - Emulator: Vertex-6 FPGA emulator
  - Interface: JTAG

- Adopted Design: *http://opencores.org/*
  - LCD Controller
  - ADPCM Encoder and Decoder
  - Deblocking Filter

Generated Coverage Assertions

- Adopted coverage metrics:
  - Cross-product coverage
  - Branch coverage

- Numbers of coverage assertions:

Design	# of S_Cassert	# of E_Cassert	# of H_Cassert
LCD Controller	19	20	34
ADPCM Enc./Dec.	121	115	268
Deblocking Filter	170	198	381

Area Reduction of Coverage Assertions

Design	original (#LUTs)	optimized (#LUTs)	Reduction Ratio
LCD Controller	2665	1595	40.2%
ADPCM Enc./Dec.	9920	7552	23.8%
Deblocking Filter	48583	30730	36.7%
Average			**33.6%**

Outline

- Introduction
- Coverage Analysis In a Hardware-Accelerated Environment
- Optimization of Coverage Assertions
- Experimental Results
- Conclusions

Conclusions

- To measure coverage in a hardware-accelerated environment, we propose using three types of coverage assertions.
- In addition, an Assertion Operation Graph (AOG) and graph-based algorithms are proposed to optimize the overheads of coverage assertions.
- The experimental results showed that we can analyze coverage metrics across a simulator and an emulator. Also, we achieved an encouraging reduction of overheads caused by coverage analysis.

Thank you!