

Accelerated Coverage Closure with inFact in verification of 802.11 Device



Chandramouli Ganapathy

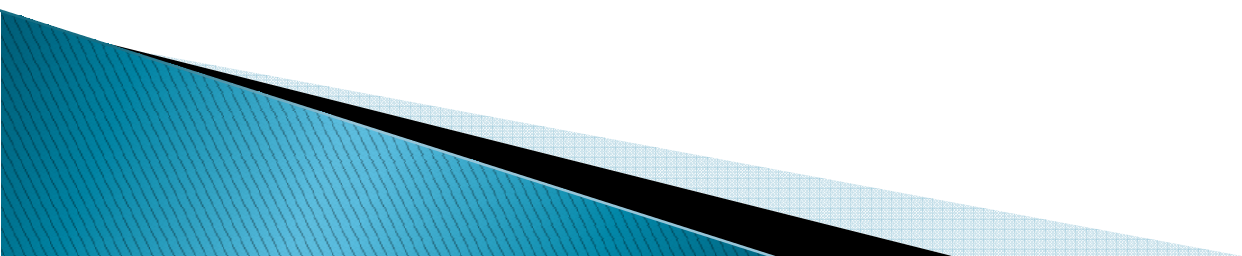
Senior Member of Tech Staff

Digital Design

Atheros Communications Inc.

802.11 Device Verification Complexity

- 37+ MAC frame types
- 5 different security algorithms
- 24 different data rates
- 5 different PHY parameters
- 2 different modulation schemes
- 2 different frame set up (frame, Aggregate)



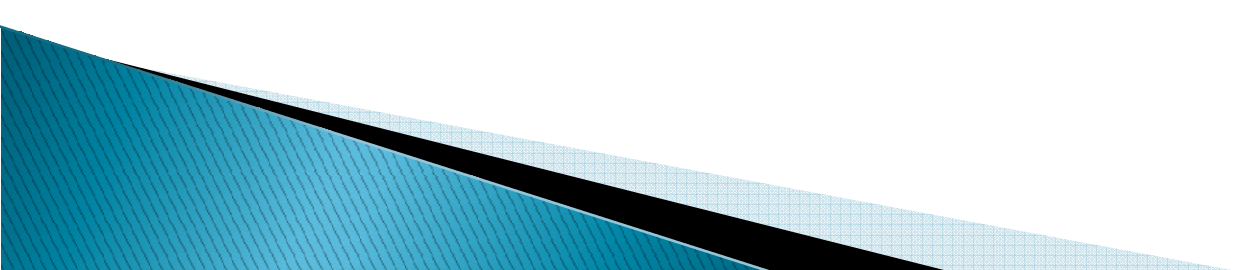
Legacy Verification Practice & Drawbacks

➤ Legacy Practice

- Native Test Bench (NTB) based DV environment
- 200+ block specific directed tests to run in system level
- Random tests for multiple scenario generation
- Random frame generator with 132 variables and 30 constraints for random testing
- Functional Coverage to measure the quality of verification with target of 95% (max 97%; 3% due to unsupported features)

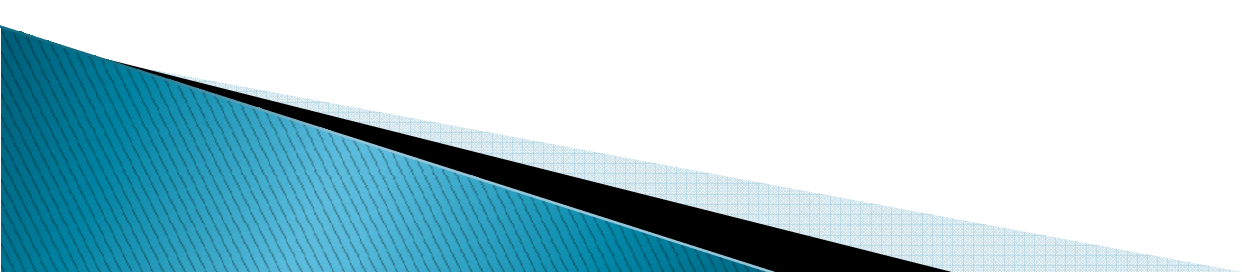
➤ Drawbacks

- Long convergence time to achieve required functional coverage
- Large variance in convergence time between different runs
- Modification of constraints can cause constraint solver failures
- Interactive skilled work requirement to modify constraints and to add directed tests according to required coverage



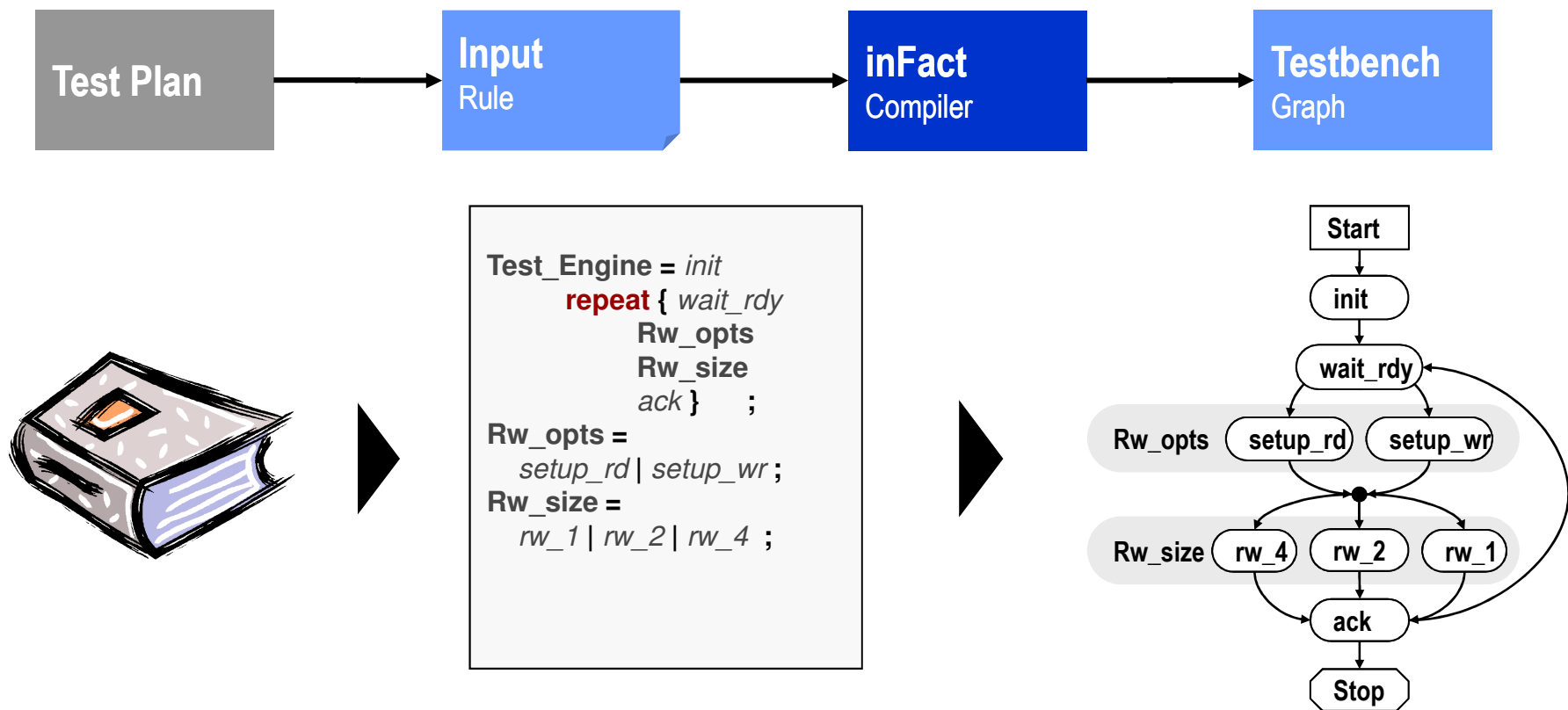
Solutions that were thought of...

- Brute force Approach – throw in more CPU's & licenses – How many? How much?
- Usage of query method in the functional coverage and changing constraints – but how often to invoke? Increases simulation time
- Development of “in-house” tree algorithm based stimuli generator – large man hours needed & maintenance cost
- CCT – Coverage Convergence Technology from Synopsys – a Limited Customer Application, but still uses CRM

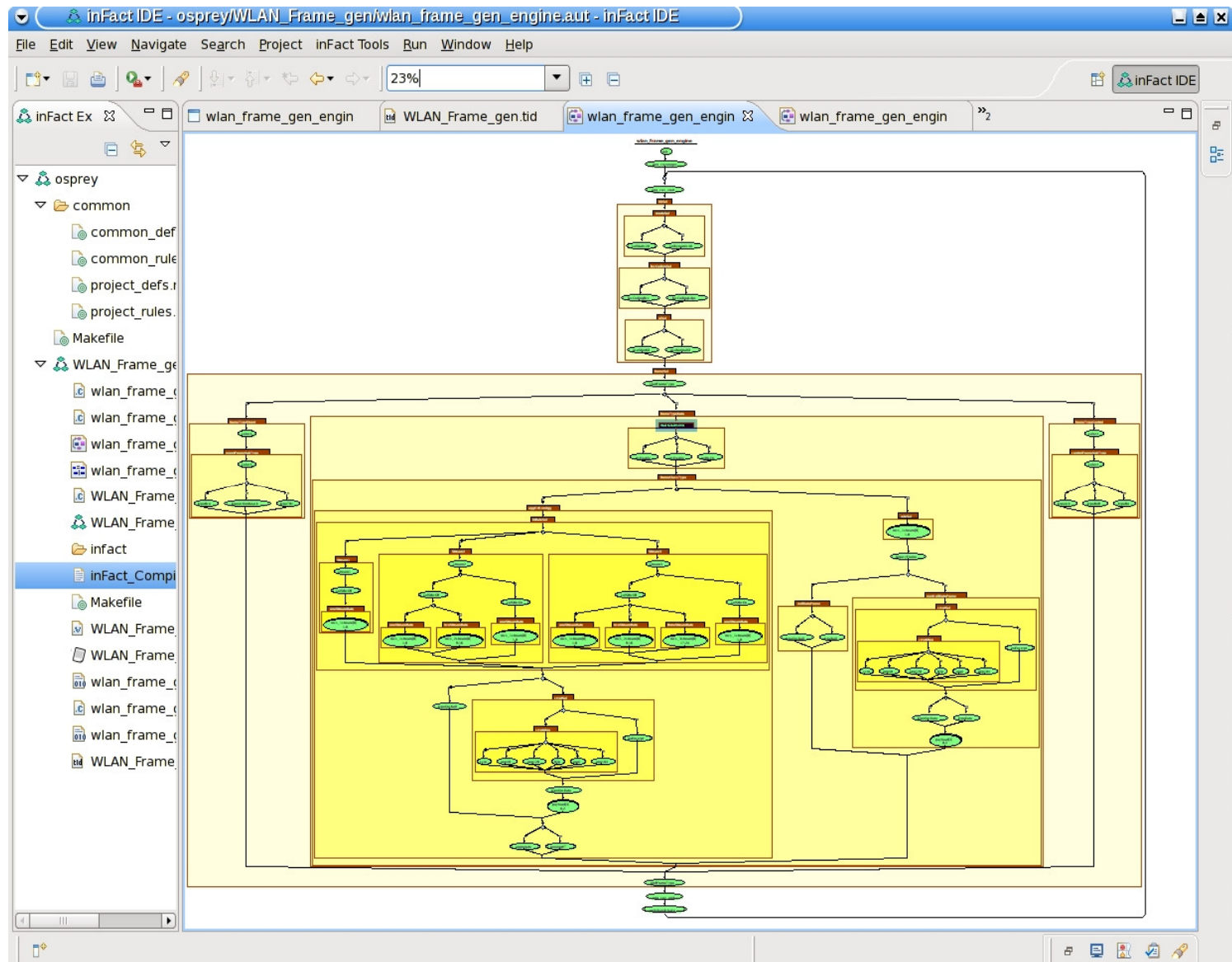


inFact Graph-Based Solution

Graphs Replace Constraint-Driven Stimulus Targeting Coverage Goals Without Test Duplication

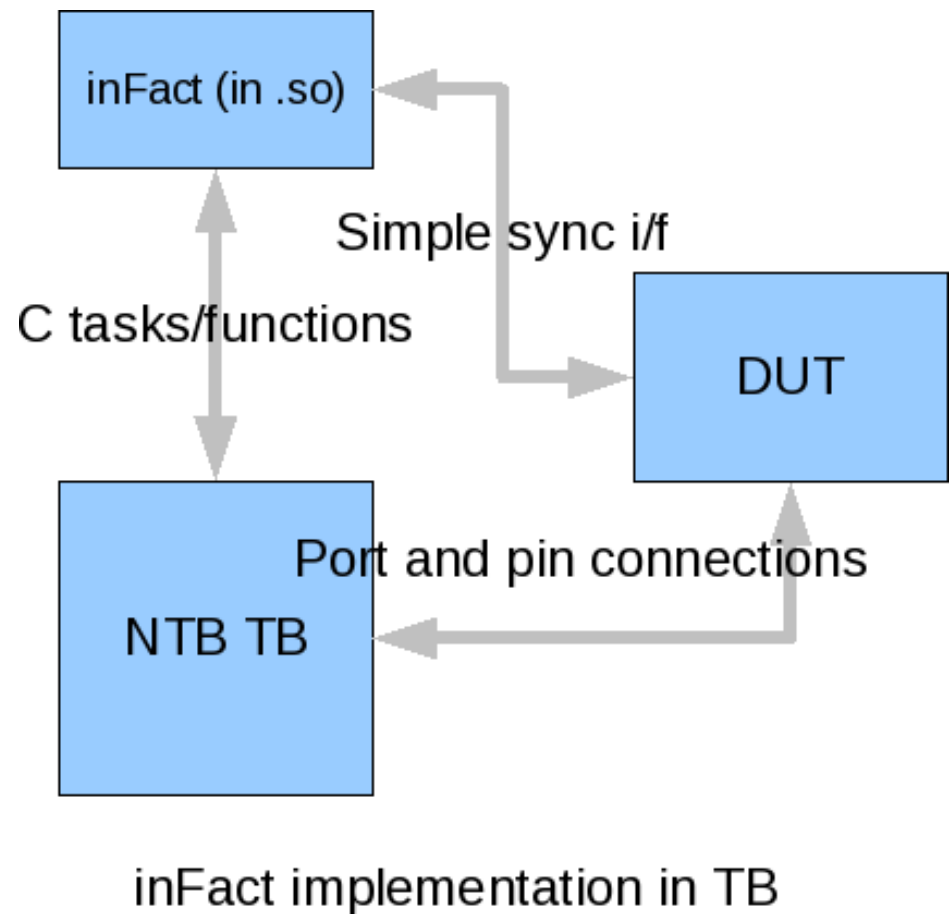


inFact Rule Graph for 802.11x



inFact implementation in NTB based TB

- 3 hours training and 2 trial runs to implement
- 12 hours to develop inFact rules and object-level interface
- Since TB used NTB, inFact C++ style code and object-level interface used.
- Auto generated verilog module provides the synchronization between TB & inFact and is instantiated in DUT



inFact Usage

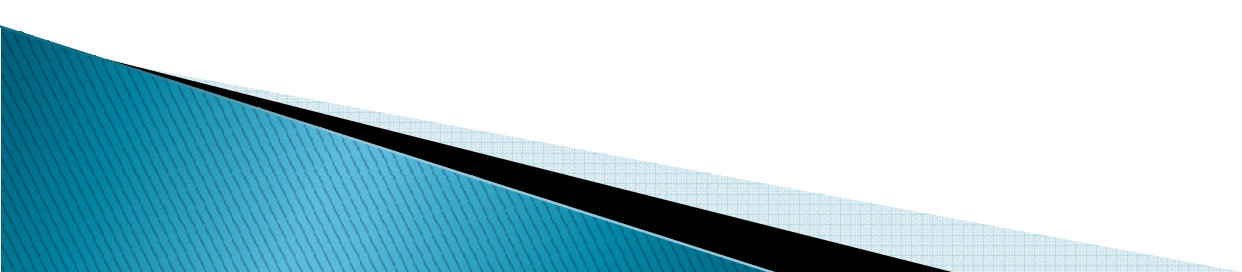
inFACT was added to the existing methodology to achieve accelerated coverage closure.

➤ inFact Application Method-1 (inFACT AM-1)

- Directed tests were untouched
- inFact based coverage-driven frame generator was created by generating rule graph
- 2 tests were triggered using this coverage-driven frame generator

➤ inFact Application Method-2 (inFACT AM-2)

- Aside with the setup above following steps were also followed:
- inFact based frame generator for directed tests
- Randomizing untouched values in directed tests
- Required 20 man hours in porting around 200 directed tests to support this



Results

Methodology	Functional Coverage	Total Real Time	Total CPU Time
Legacy	95%	175 hrs *	4375 hrs
inFACT AM-1	96%	85 hrs **	1274 hrs
inFACT AM-2	97%	72 hrs **	1248 hrs

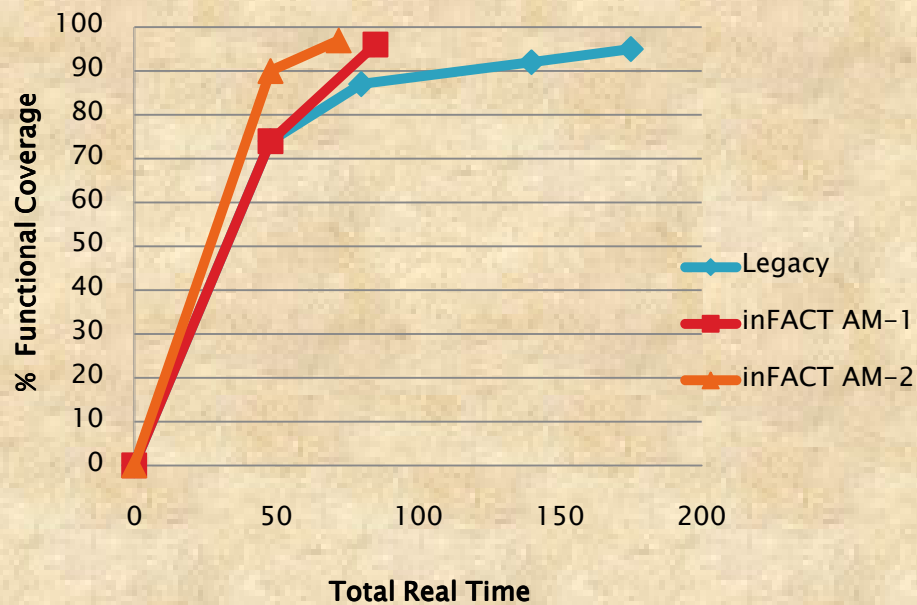
*25 CPUs were used for Legacy ; **only 2 CPUs were used to run inFACT

Directed tests contribute (48 hours * 25 CPUs =) 1200 CPU hours. Omitting this common factor, comparing the methodologies for CPU time the results are:

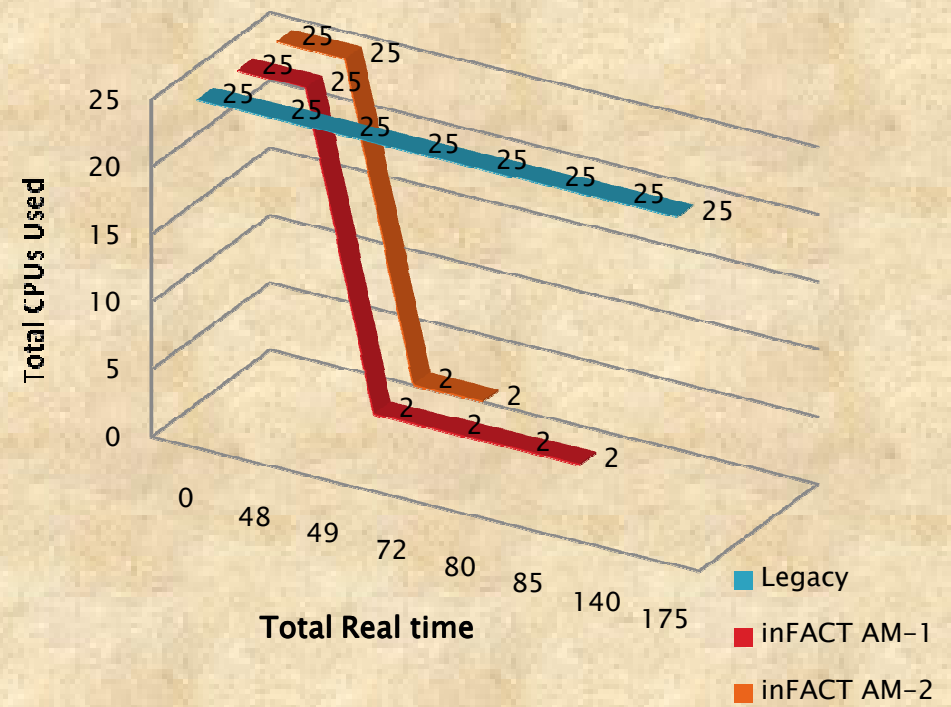
Methodology	CPU Time	Performance Gain
Legacy	3175 hrs	–
inFACT AM-1	74 hrs	43x
inFACT AM-2	48 hrs	66x

Comparison Chart

Coverage Vs Total Real Time Comparison



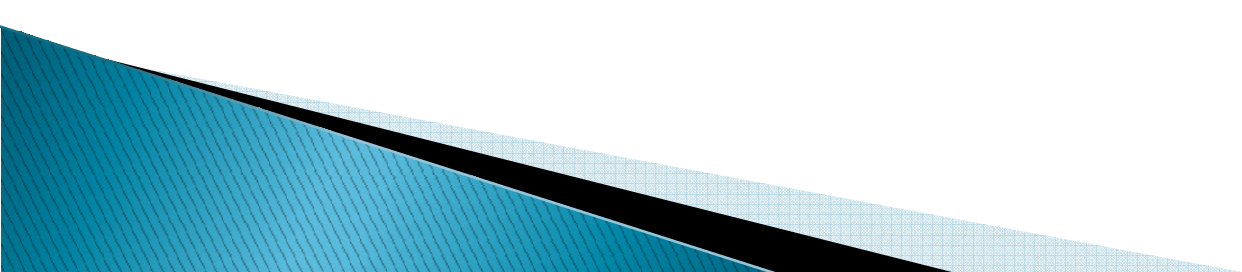
CPUs Vs Total Real Time Comparison



Note: Directed tests ran before Constraint Random Test and inFACT

Comparison Results

- inFact based TB converged 50X faster than traditional CRM based TB in average
- inFact based TB easily maintainable as change in rule can be verified graphically
- Portability/Support to various platforms, enables to use a common stimuli generation platform for simulation and emulation
- The cost of coverage is also low, as 2 random tests were run using inFact, while 40 tests were run in legacy TB
- Easy to control the distribution of stimuli during run time



Conclusion

- inFact based test bench reduces cost of coverage convergence by 50X in average with a range of 10X-100X speed up
- inFact allows same test bench to run in random, directed or directed random mode with very less changes
- Rule generation is simple and easily verifiable

