BackSpace: Formal Analysis for Post-Silicon Debug

Alan J. Hu
University of British Columbia
with Tor Aamodt, Andre Ivanov, Steve Wilton
and Flavio De Paula, Marcel Gort

Outline

- The Vision
  - Motivation, Problem Statement
  - Proposed Solution
- Current Status
  - Theoretical Results
  - Basic Architecture Estimates
  - Preliminary Experiments

The Crisis

- Post-silicon validation is now more than 50% of total verification time. – Andreas Kuehlmann
  - And it's the highest variance, too!

The Scenario

- Die comes back from fab:
  - Yield wasn't great, but several good die make it to the bring-up lab.
- 30 seconds into booting Windows, the chip crashes.
  - Uh-oh… all the "good" die crash in the same way.
  - Scanning out the crash dump shows TLB (or multi-core coherence directory) corrupted. "That's impossible. How did that happen?"
- What do you do now?

The Problem

- As integration increases, there simply isn't enough I/O to the die
  - Insufficient observability and controllability means no way to figure out what went wrong.
- Other issues as well:
  - Greater variability with technology scaling
  - More faults. Testability is harder, too.
  - Etc.
  - We will ignore these for now. More later…
  - Focus on the "1 million DPM defects"!

Our Goal

- Increase observability (and maybe controllability). Allow figuring out what happened.
- Focus on design errors. Extend later…

- NOT trying to "diagnose" problems:
  - No realistic "fault models" for design errors.
  - No guarantee solution is what really went wrong.
  - Computationally too expensive.
  - Unreasonable assumption of repeated fault simulation of error trace.

Simple, Ideal Solution

- Replace every latch with a shift register
  - On each clock cycle, record history of latch.
  - 100 cycles? 1000? 1,000,000?
  - On a crash, can scan out the last thousand (or million) cycles. Enough history to figure out what went wrong.
- Hey, transistors are free! (And people used to think scan latches were too expensive…)

Simple, Ideal Solution

- Replace every latch with a shift register
  - On each clock cycle, record history of latch.
  - 100 cycles? 1000? 1,000,000?
  - On a crash, can scan out the last thousand (or million) cycles. Enough history to figure out what went wrong.
- Unrealistic, but it **would** make debugging easier!

The Objective

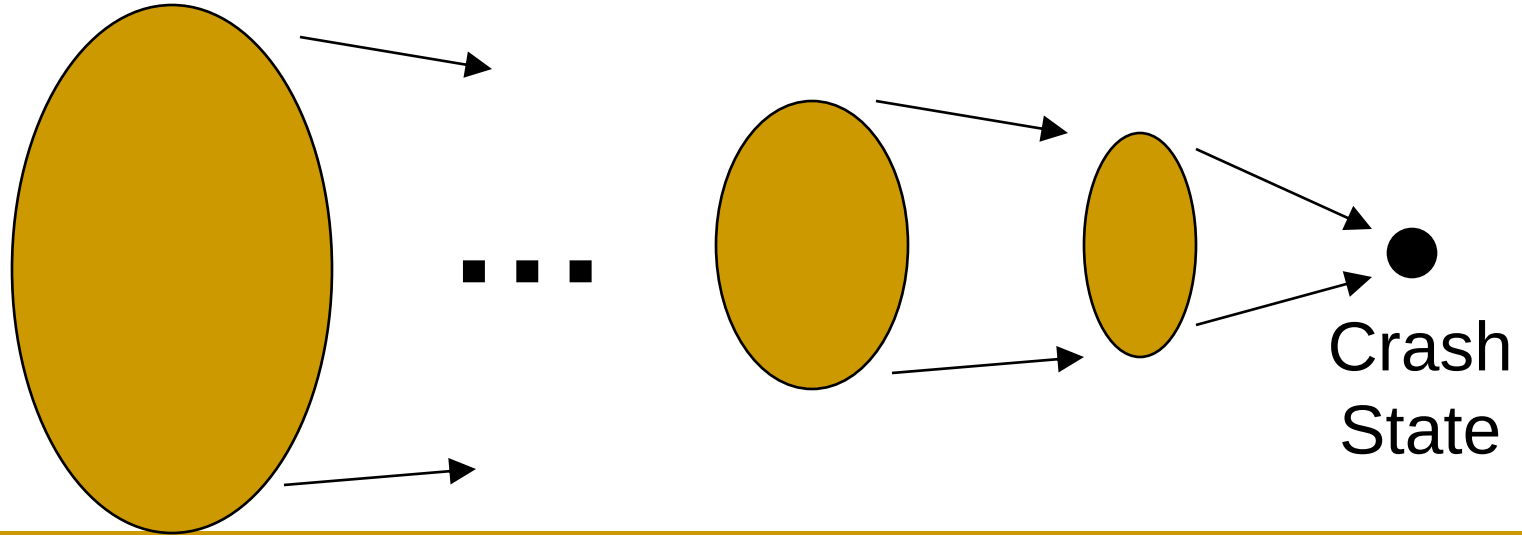
- How do we get the effect of the simple, ideal solution, but with reasonable overhead?
- This is hard. We propose a multi-pronged approach.
- Disclaimer: This is only just beginning!
  - cf. Model checking in 1983?

Assumptions

- Assume we can scan out the state that crashed.
- Obviously, if formal verification can hit crash state, then problem is solved. But formal verification capacity is limited.
- Similarly, if bug is shallow enough (and not speed-dependent) we can reproduce it via test-interface and single-stepping.
- Otherwise…

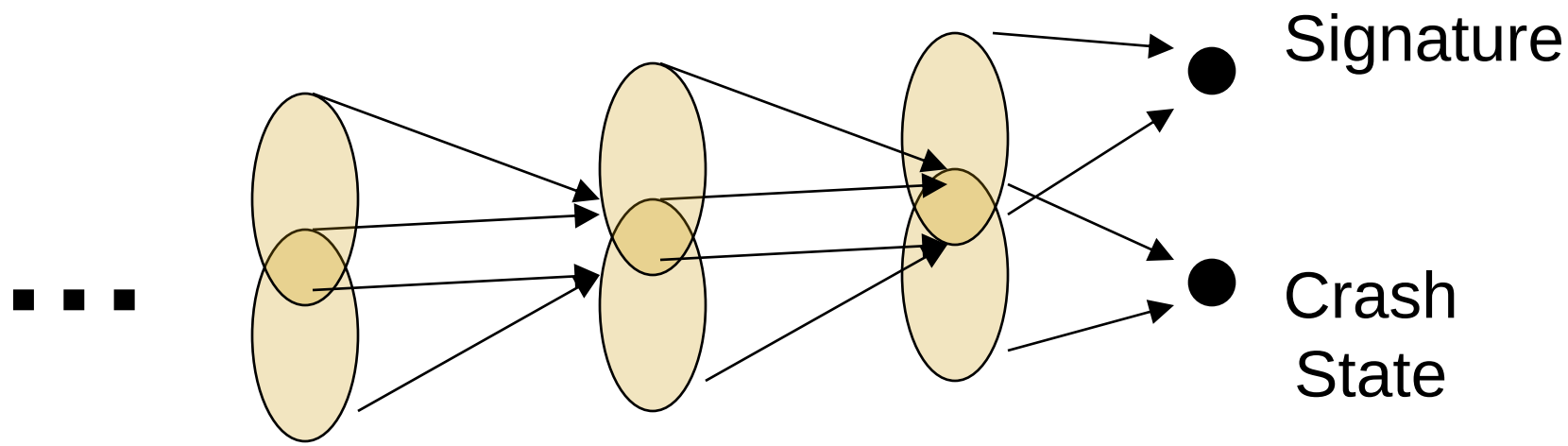
Proposed Solution v0.0

- Key Insight: Ignore faults for now. Therefore, RTL is golden. Use formal analysis on RTL to rewind from crash state.
  - Pre-image from single state is easy.



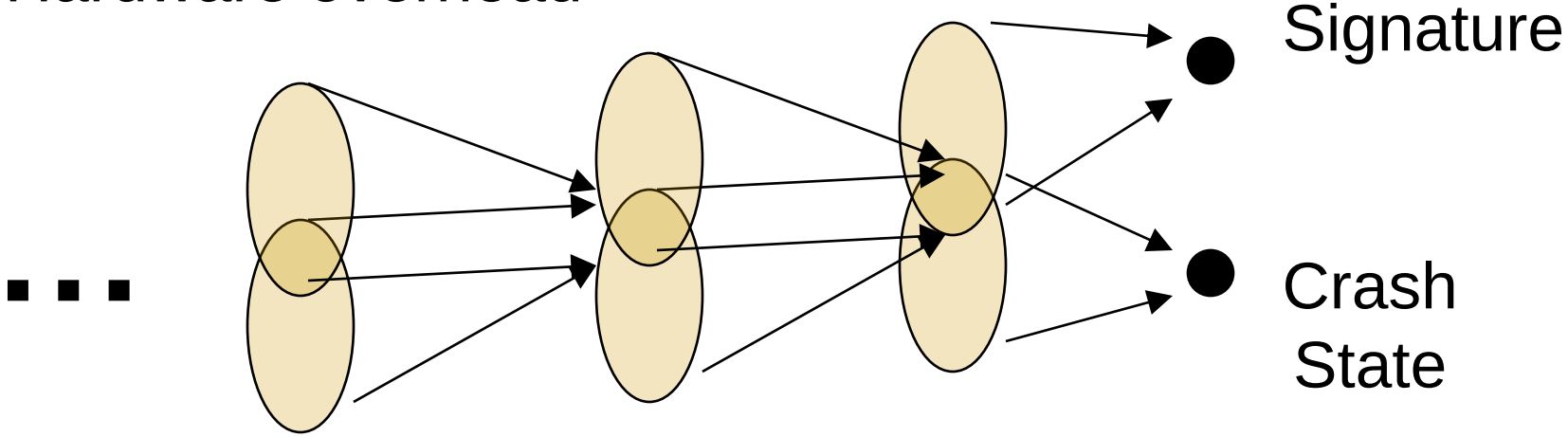
Proposed Solution v0.1

- Key Idea: Add a BIST-like signature generator that records/summarizes state.
  - Use pre-images of signature to prune trace.



Proposed Solution v0.1

- Research Questions:
  - Getting formal theory to work
  - Avoiding pre-image blow-up
  - Signature generation
  - Hardware overhead

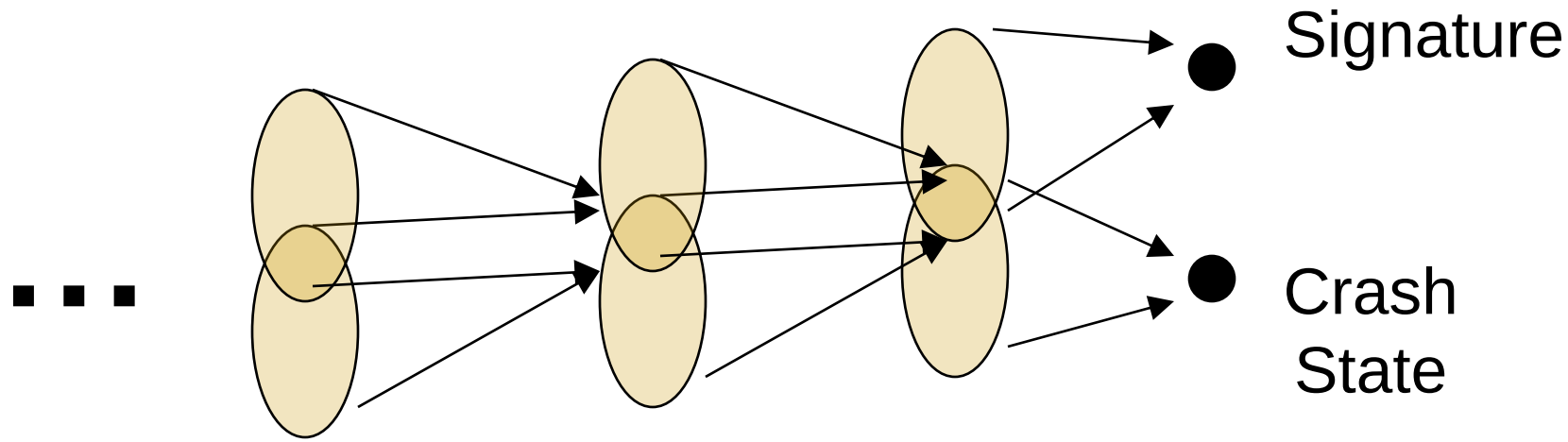


Proposed Solution v0.1

- Limitation: To get unique back trace, number of bits in signature lower-bounded by:

  cycles x lg(backward branching factor)

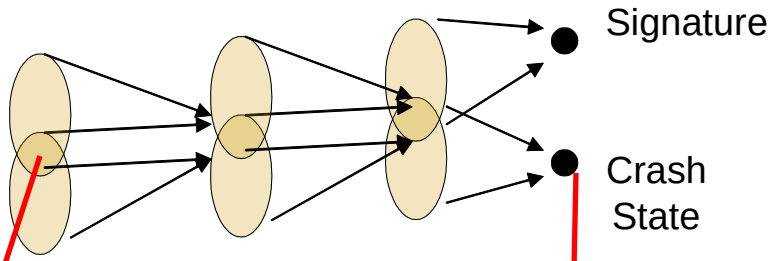
- Therefore, can rewind only tens of cycles.



Proposed Solution v0.2

- Use Solution v0.1 to rewind tens of cycles.
- This generates new target state!
  - Design errors are repeatable (with enough tries).
  - If we can generate a trace that *can* tickle bug, that's likely enough to help find problem.
- To get chip to "crash" at new target state, we need **programmable breakpoints on-chip**:
  - Breakpoints trigger on target signature or pattern.
  - FPGA-like technology to program breakpoint.
  - Architecture adds breakpoint to trigger interrupt.

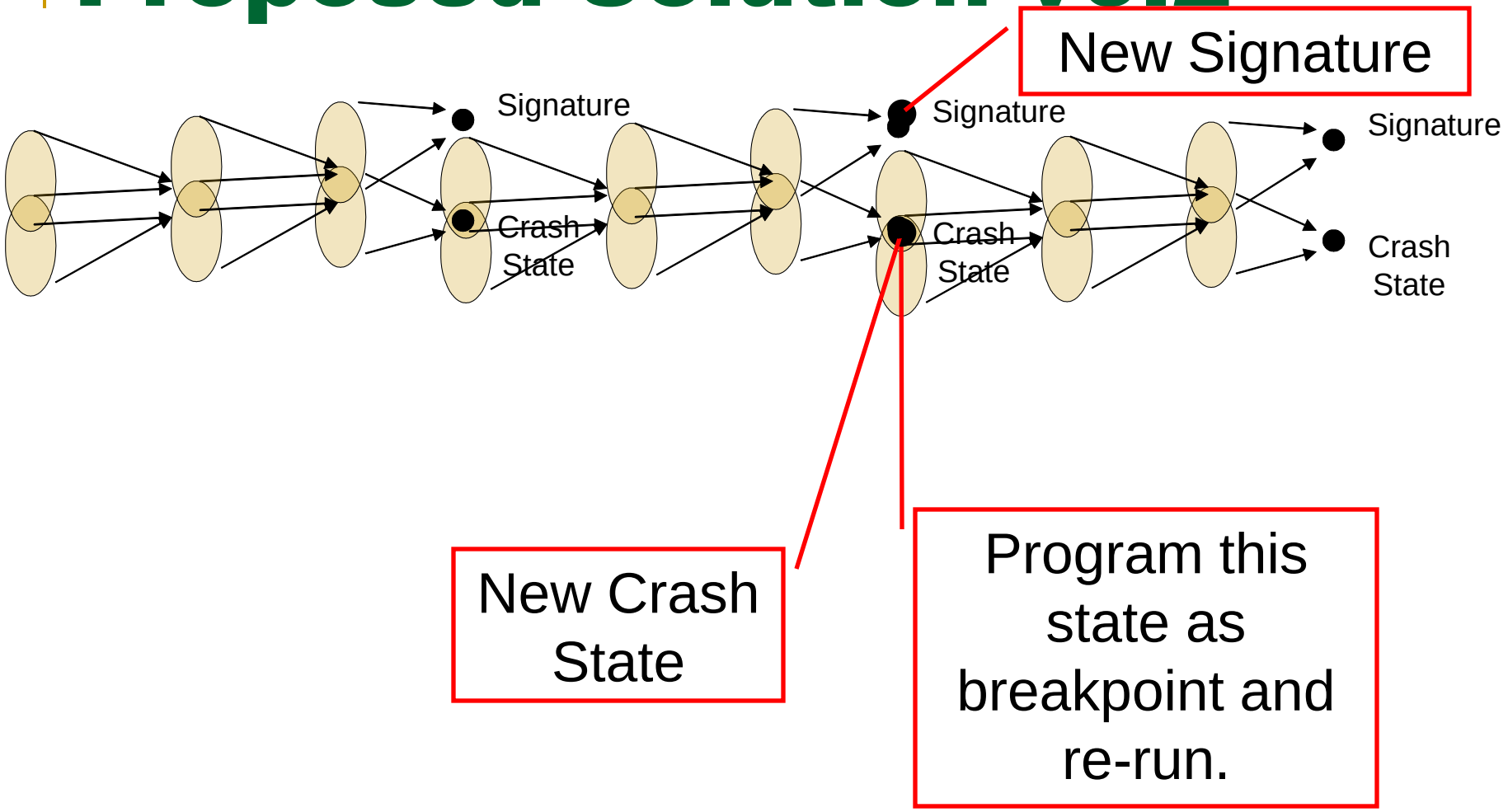
Proposed Solution v0.2



Formally Computed Predecessor State

Original Crash State and Signature

Proposed Solution v0.2



Proposed Solution v0.2

- Research Questions:
  - How to do this with minimal area, delay, and power penalties? And is it routable?
  - Any micro-architectural problems, or can this reuse existing mechanisms?
  - Can we stitch together the formally derived trace segments? How do we minimize the challenge of non-determinism?

Proposed Solution v0.3

- Given programmability on-chip, can we use programmability to reduce overhead?
- Research Questions:
  - What's a flexible architecture that lets us monitor different micro-architectural state and compute different signatures?
  - Programmability feeds back to the formal analysis. Given a crash state, what is the right programming to get maximum error trace information?

Proposed Solution v0.4

- What about transient errors, faults, and variability?
- Can handle this similarly to fault simulation:
  - User provides “fault model” – list of likely possible divergences of design behavior from the RTL
    - This requires deep knowledge. Not good for automation!
  - Assuming each possible divergence, use preceding method to attempt explanation of crash behavior.
  - Anything that fits the observed behavior is very likely the problem.

Current Status

That was then. This is now.

Executive Summary

- Mostly according to roadmap:
  - Basic theory works out.
  - Basic architecture is doable.
- Some surprises, e.g.:
  - Better way to do pre-image computation
  - Don’t need unique pre-images
  - Breakpoints are easy, memory is small, concentrator is large.
- Short Term Changes/Refinements:
  - Backspace Coverage
  - Initially, minimal programmable logic

Outline

- The Vision
  - Motivation, Problem Statement
  - Proposed Solution
- **Current Status**
  - **Theoretical Results**
  - **Basic Architecture Estimates**
  - **Preliminary Experiments**

Bad News

- Our approach relies on adding state bits to get a unique predecessor state.
- There exists states where you basically must record **all** previous state bits to get unique predecessor:
  - Imagine a resettable counter. What is the pre-image of the 0 state?

Backspace Coverage

- We introduce the concept of *backspace coverage* to deal with this problem.
- What fraction of the reachable states have a unique pre-image with the added signature bits?

- A high backspace coverage is likely good enough.

Crash State History Algorithm

- Given state (s_0,t_0) of a backspaceable augmented state machine M', compute a finite sequence of states (s_0,t_0), (s_1,t_1),… as follows:
  - Since M' is backspaceable, let s_{i+1} be the unique pre-image state (on the state bits) of (s_i,t_i).
  - Run M' (possibly repeatedly) until it reaches a state (s_{i+1},x). Let $t_{i+1} = x$.

Theorem (Correctness)

If started at a reachable state, the sequence of states computed by the preceding algorithm is the (reversed) suffix of a valid execution of M.

Theorem (Probabilistic Termination)

- If the forward simulation is random, then with probability 1, the preceding algorithm will reach an initial state.

Pre-Image Efficiency

- Pre-image with BDDs blew up.
- Pre-image with ALL-SAT is immature.
- Fortunately, we discovered that we can use conventional SAT!
  - We want a unique (or almost unique) pre-image.
  - Ask a normal SAT solver for a pre-image state.
  - Add a blocking clause and try again.
  - Repeat.

Handling Non-Determinism

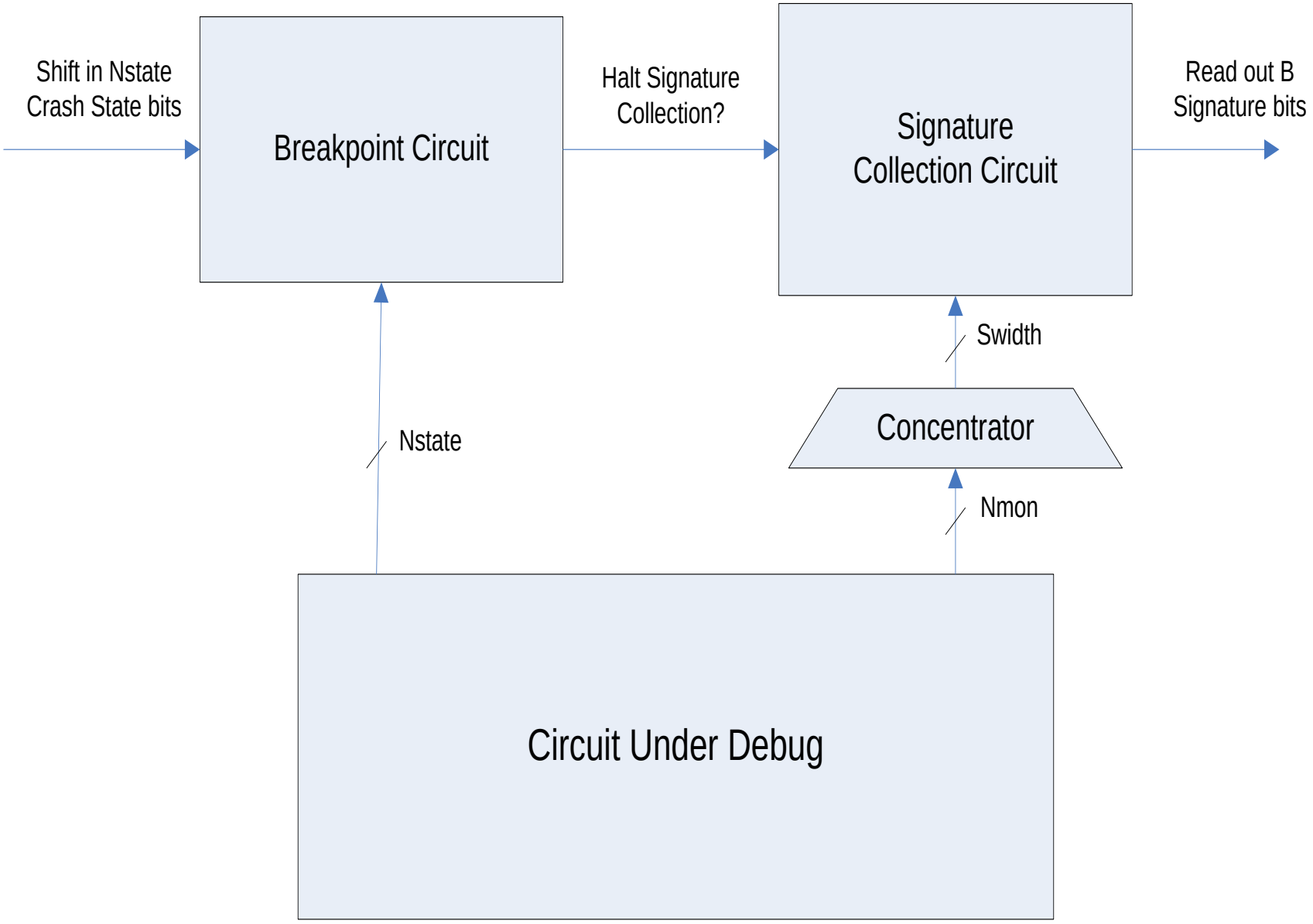
If non-determinism becomes randomness during simulation, then the preceding algorithm still works, with a slowdown factor of n, if it takes expected n trials to hit the same target state.

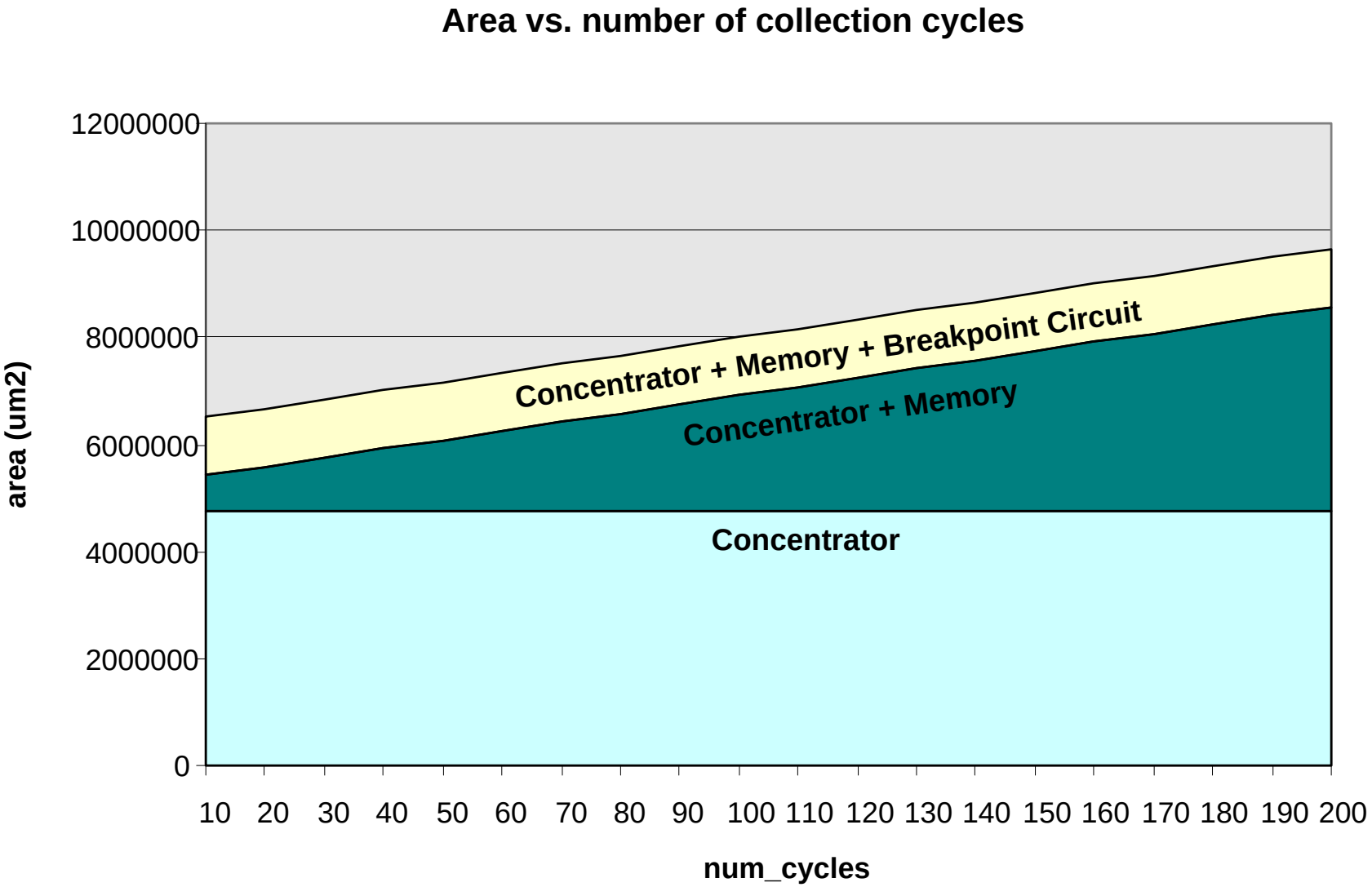
Non-Unique Pre-Images

If there are on average n states in each pre-image, the preceding algorithm still works, with a slowdown factor of n.

Main Architecture Components

- Breakpoint Circuit
  - Programmable target register
  - Only needs to halt signature collection, not chip under debug.
- Concentrator Network
  - Highly efficient network to get any k of n signals
  - Programmable
- Signature Memory
  - Study assumes SRAM architecture for density
- Everything can be pipelined


Shift in Nstate
Crash State bits
Breakpoint Circuit
Halt Signature
Collection?
Signature
Collection Circuit
Read out B
Signature bits
Swidth
Concentrator
Nmon
Nstate
Circuit Under Debug


Area vs. number of collection cycles
area (um2)
12000000
10000000
8000000
6000000
4000000
2000000
0
Concentrator + Memory + Breakpoint Circuit
Concentrator + Memory
Concentrator
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200
num_cycles

First Experiment -- Signatures

- Treat RTL running in simulator as if it were silicon.
- Use simulation testbench to generate 10 random target states (and record their predecessors).
- Try different signatures and see how often we can get a unique pre-image.

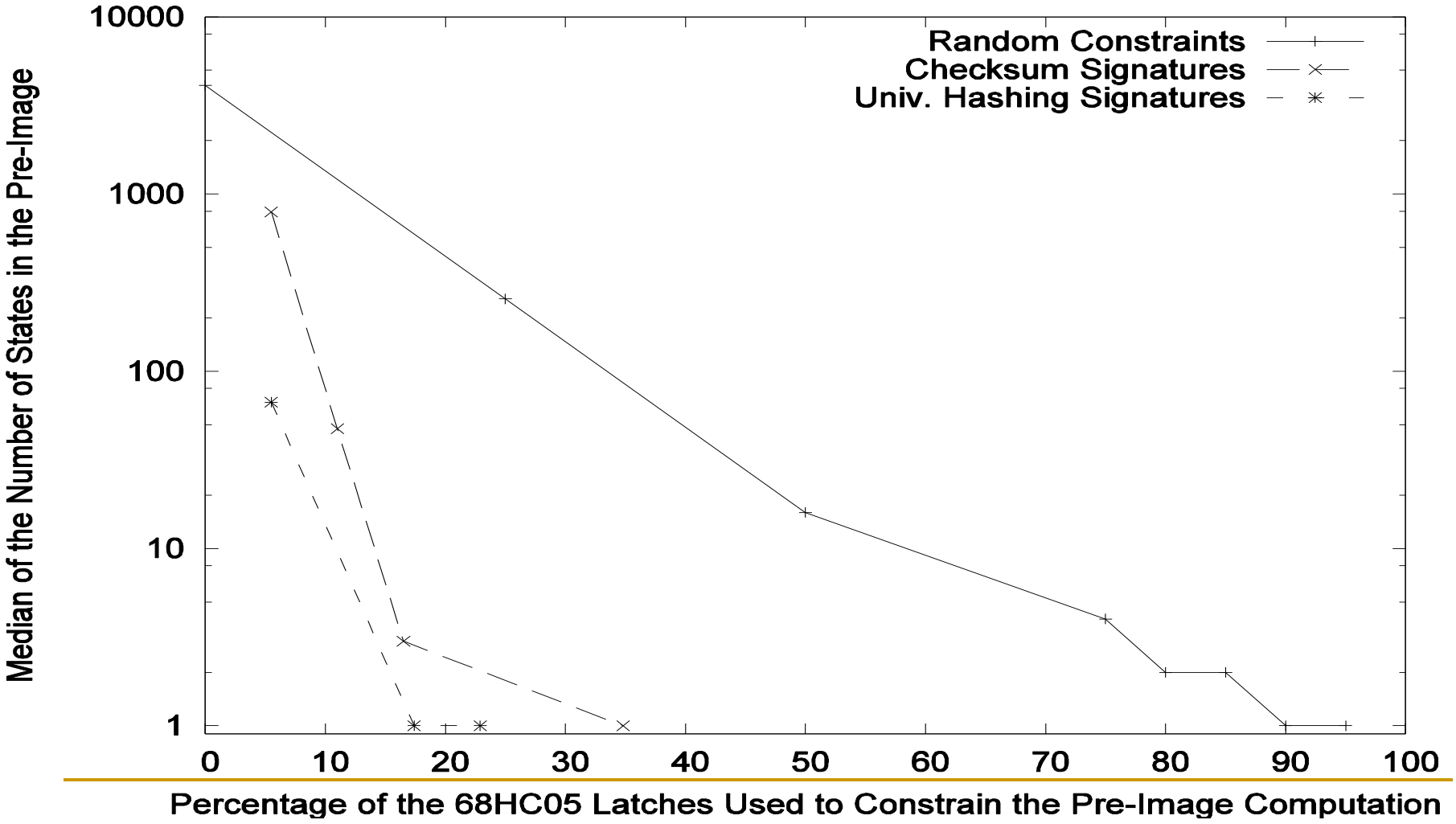
Design Examples

- Research is still highly exploratory.
- More important to focus on a small number of small designs, to drive the ideas forward.
- Not yet appropriate to work on large benchmark suites!
- Three open-source designs:
  - Ethernet Controller, 347 latches
  - 68HC05, 109 latches
  - 8051, 702 latches

Experiments

- Signature as random fraction of state bits.
- Signature as hand-chosen set of state bits.
- Signature based on simple checksums of hand-chosen state bits.
- Signature based on universal hashing (aka X-Compact)

Signature Size vs. States in Pre-Image



Does the whole thing work?

- Prototyped BackSpace framework
  - Using commercial logic simulator as if it were chip on silicon.
- Successfully able to compute predecessor traces from randomly selected “crash states”

68HC05 w/ 38-Bit Manual Signature

Crash State	# of Cycles	Max Pre-Img	Sim Time	SAT Time	Manager Time
S1	54	4096	63.44	13.11	204.42
S2	1	65536	1.45	0.27	4.38
S3	37	4096	62.65	8.89	139.67
S4	7	4096	37.76	1.67	27.11
S5	53	4096	116.16	13.18	200.34
S6	500	1	1261.48	118.79	1884.31
S7	500	1	2384.29	120.68	1890.91
S8	500	1	4575.41	138.87	1893.89
S9	2	4096	22.93	0.53	22.93
S10	9	65536	2424.55	2.31	34.86

68HC05 w/ 38-Bit X-Compact

Crash State	# of Cycles	Max Pre-Img	Sim Time	SAT Time	Manager Time
S1	500	2	1097.25	119.98	8524.15
S2	500	2	2011.04	119.32	8397.09
S3	500	2	2737.15	118.98	8335.45
S4	500	2	2988.38	118.57	8477.88
S5	500	2	3358.40	116.84	8398.14
S6	500	1	3176.94	117.27	8175.62
S7	500	1	6247.61	117.40	8280.93
S8	500	1	12207.49	115.96	8297.21
S9	500	2	15280.79	115.06	8173.19
S10	500	1	34084.53	114.34	8125.62

8051 w/ 281-Bit Manual Signature

Crash State	# of Cycles	Max Pre-Img	Sim Time	SAT Time	Manager Time
T1	205	512	2841.07	51.49	6048.46
T2	500	256	21759.74	134.24	14720.71
T3	500	257	8326.66	119.07	14746.10
T4	500	257	10342.40	120.92	14772.03
T5	500	256	11587.21	130.32	14742.81
T6	500	256	11581.93	131.07	14735.07
T7	500	255	25767.40	131.04	14742.60
T8	500	256	13581.20	131.96	14759.73
T9	500	257	22493.04	134.48	14735.48
T10	500	257	24793.42	126.03	14759.77

8051 w/ 281-Bit X-Compact

- Results just completed. We don't have the data analyzed into a table yet.
- For all 10 target states, we could backspace to our 500 cycle limit.
- Manager is vastly faster now, too. (Just better coding.)

Next Steps…

- Prototype on FPGA of non-trivial example!
  - Advice, suggestions on example? (Looking at Leon3, OpenRISC, eventually a Sun SPARC?)
- Efficient architectures for signatures
  - Good, low overhead. Reuse existing structures?
- Building a robust tool
- Exploring programmability vs. complexity trade-off (e.g., concentrators versus fixed routing, programmable hashing)

The Team

Tor Aamodt, Assistant Professor, ECE

- Senior architect in GPU engineering at nVidia
- Graduate intern in Intel MRL
- PhD from U of Toronto, NSERC Fellowship
- Research on microarchitecture and memory system design, including issues arising from multi-threading and multi-core

The Team

Alan J. Hu, Professor, CS

- Over 15 years research on automated, practical formal verification
- Frequent industry interaction
  - Funding from Fujitsu, Intel, Microsoft
  - Proposal stems from discussions with Jin Yang of Intel
- PC of every major CAD and FV conference
  - (Co-)Chaired CAV, FMCAD, HLDVT
- ITRS 2001
- Research: Model checking and eq checking, BDDs and SAT, hardware and software, finite-state and parameterized, formal and semi-formal, gate level to multiprocessor memory protocols
- PhD from Stanford, ONR Fellowship, etc.

The Team

Andre Ivanov, Professor, ECE

- IEEE Fellow
  - “for contributions to infrastructure intellectual property(IP) for system-on-a-chip (SoC) testing”
- Three time recipient IEEE TTTC Meritorious Service Award
- PC of VTS, MSTW; Associate Editor of JETTA
- Over 100 refereed publications: device-level to algorithmic
- Key Research Relevant to Proposal:
  - Test Networks-on-Chip
  - High-Speed Interconnect
  - Fast signature generators and space compactors for BIST
    - Including programmable space compactors

The Team

Steve Wilton, Associate Professor, ECE

- Over 15 years research on FPGAs and architecture
  - Both FPGA architecture and CAD
    - Wilton switch block
    - FPGA power model
  - Also work on general computer architecture
    - e.g., Jouppi and Wilton, CACTI
- PC of all major FPGA conferences
  - (Co-)Chaired FPGA, FPL
- Key Research Relevant to Proposal:

  Post-Silicon Debug using Programmable Logic Cores
  - Concentrator network architecture allows observability and controllability
  - Low overhead – e.g., 80M gate design and 7200 observable signals, concentrator network and PLC overheads both < 1%

The SoC Lab

- We are all members of the Soc Lab, which is very collegial. Other members will support as needed:
  - Mark Greenstreet – high-speed analog/digital, formal verification
  - Guy Lemieux – FPGA, architecture, reconfigurable computing
  - Shahriar Mirabbasi – analog and mixed signal IC and systems
  - Res Saleh – SoC design, global interconnect, power grid, clock
- In-house and available design examples:
  - E.g., Bluetooth SoC, PowerPC and NIOS processors, etc.
- FPGA prototyping expertise, access to 90nm through CMC
- State-of-the-art test lab
  - High speed probe stations, test equipment, etc.
  - Roberto Rosales, full-time test lab manager