Challenges of in-circuit functional timing testing of System-on-a-Chip

David and Gregory Chudnovsky

Institute for Mathematics and Advanced Supercomputing

Polytechnic Institute of NYU

Deep sub-micron devices with billions of transistors

- The scale of features in chip designs we will talk today - from 90 nm for a mature (read: obsolete) technology to 22-32 nm for a current (b)le(e/a)ding edge manufacturing technology.
- Systems on the chip with hundreds of cores of conventional or special-purpose processors
- Various inter-processor communication topologies, from meshes to full crossbars

Mitigate failures

- Fault-tolerance as “Partial Goods”, with the built-in manufacturing PGT (“Partial Goods testing”)

- Conventional in-circuit testability – BIST and LBIST. Good for storage and memory elements (eDRAMs, SRAMs, RFs).

- Redundancies – TMR/QMR and other modular redundancies. Unacceptably high cost for customers. Repeat compute = half of the performance.

In-system real-time stress test of all devices

- Without system downtime.
- PDF (Path Delay Fault) model rather than a single timing fault of a wire or a gate.

- No perfect framework exists – unlike for static faults.
- Conventional methods – LOS (launch on shift), 2-cycle patterns differing by 1-bit shift, and LOC (launch on capture). Majority of timing paths not covered this way.

- BUT – majority of timing faults cannot be functionally triggered…
- Example: decoded vs. encoded states.

Functional Timing Testing – a long story..

- Takes longer than actual logical and physical chip design and verification. That is why we decided to talk about it.
- Chip A – functional testing as an afterthought, though the first PGT during manufacturing process. Consequence – exponentially hard functional timing testing.
- Chip B – tools for the functional testing are (kind-of) built in the design. Makes the job of complete functional timing testing (somewhat) feasible.

Mathematical tools

- Use of Mathematica environment for development of functional timing testing tools, in addition to Boolean satisfiability and BDD tools. Also – temporal logic rules, multi-cycle symbolic analysis, and symbolic simulations (in some cases up to 16 cycles forward or backward).

- Functional testing definition:
- Generation of actual native language programs, that can be executed from the front end of the machine, as a part of a normal workload. No placement of the system into a special test state, such as a JTAG or user-defined scan state.

Observations

- For Chip B we were able to determine in “a finite time” which timing paths are functionally coverable and which are not, and develop a method of generating an assembler program covering coverable path.
- 1. For both cases, a majority of top (worst) 100K timing paths was functionally uncoverable.
- 2. The percentage of functionally coverable paths increased with a slack of the path.
- 3. Even for a large slack of 25% of a cycle period, ~55% of timing paths were functionally uncoverable.
- 4. 1-3 do NOT mean that one can run chips faster!

Observations and results

- …Still, if one would have known prior to design completion only functionally coverable timing paths, chips could have been sped up by, perhaps, ~5%.

- 5. For the Chip B, about 3400 programs (C and assembler) were created using our new tools or manually (!). They provide 100% of timing coverage for all individual segments (wires, nets and gates), and coverage of timing paths with a slack of 20% of the cycle period.

Why timing testing…

- Static (single fault) testing is easy.
- Functional timing testing is crucial in large scale systems where sources of “noise” are hard to pin down (unlike downright failures).

- Process variation and voltage compression are enemies, with the latter often generated in a resonance style by the program execution. Tight control of parallelism increases chances of the resonances generated at a sub-nanosecond rate (from 0 to 100% of the dynamic power generation in one cycle, as a part of nicely optimized loop).

2 bits of advise

- 1. For a “giant” system (and what is not giant these days), sacrifice the performance and increase latency to de-synchronize parallel program execution. Borrow from classical books on parametric resonance in mechanical and electrical engineering.
- Voltage compression is a complex subject, and due to exponential instability in voltage dependency, little protection often goes nowhere.
- 2. Do not expect manufacturing and infant mortality testing to give perfect components. Most will be reliable and last long time with expected frequency degradation.

Overdesign

- Do not rely on a small set of benchmarks only. Build functional testing, or, at least, in-circuit at-speed scan testing.
- Better yet, slow the system down ~15% and increase ~20% power for a better protection.

Timing testing is needed because..

- Aggressive demands these days are on power/performance rather than just on the pure performance or price/performance, e.g.
  - "Give me 1 Teraflop for 1 Watt" (not yet)

- For chip's real estate it is all about location, location,…
- Nowadays, each mm^2 is unique due to process variation. Thus during the design and sign-off we are likely to over-design, using conservative deterministic or high sigma statistical timing. But this will be too conservative, especially in terms of power, or else inaccurate.

Power/performance tweaking of individual parts

- Power reduction, and Power/Performance optimization are now new metrics of the design, be it for mobile or supercomputer systems.
- Many tricks were used in our recent designs.
- They include software and hardware power gating built into the design. These various clock and data gating allow for a software-driven power optimization, where at compile time one can set “power goals”, and allow for a desired skew of Power vs. Performance to be achieved.
- Ultimately, one want to optimize the power per individual chip – frequency binning is no longer good enough.

The real reason for timing testing – pushing the envelope.

- Ideally we want to have continuously adjusted voltage binning, allowing for the best power of individual chips for a fixed global frequency.
- Similar to gamers over-clocking – but with functional testing that over-test the logic by a “guard band” for possible “noise” (variation in the PVT).

- One should be able to determine the exact region of voltage, frequency and cooling at which each chip can safely operate.

Exhaustive testing, if you must and can

- As long as chip “performs correctly”, its Power and Performance metrics can be adjusted in a working system.
- But how to guarantee “correctness”?

- The real problem is the scale of the system, allowing for a great variation.
- In the case of Chip B, the system built of them has 36.8 million processor cores.

Truly Large Numbers

- 1 sec. of the life of the system has about 2 x 10^16 processors' cycles. Even in the case of perfect normal distribution of events, every second one should expect an 8.25 "sigma-event".
- 10^16 in a "real world".

- But most events are not normally distributed, and have fat tails. For "Laws of Truly Large Numbers" in practice see, e.g. a paper of Diaconis and Mosteller (J. Amer. Stat. Assoc., 1989, v.84, 853-861). This law in terms of coincidences is also known as "the Jeane Dixon effect".

Other obstacles

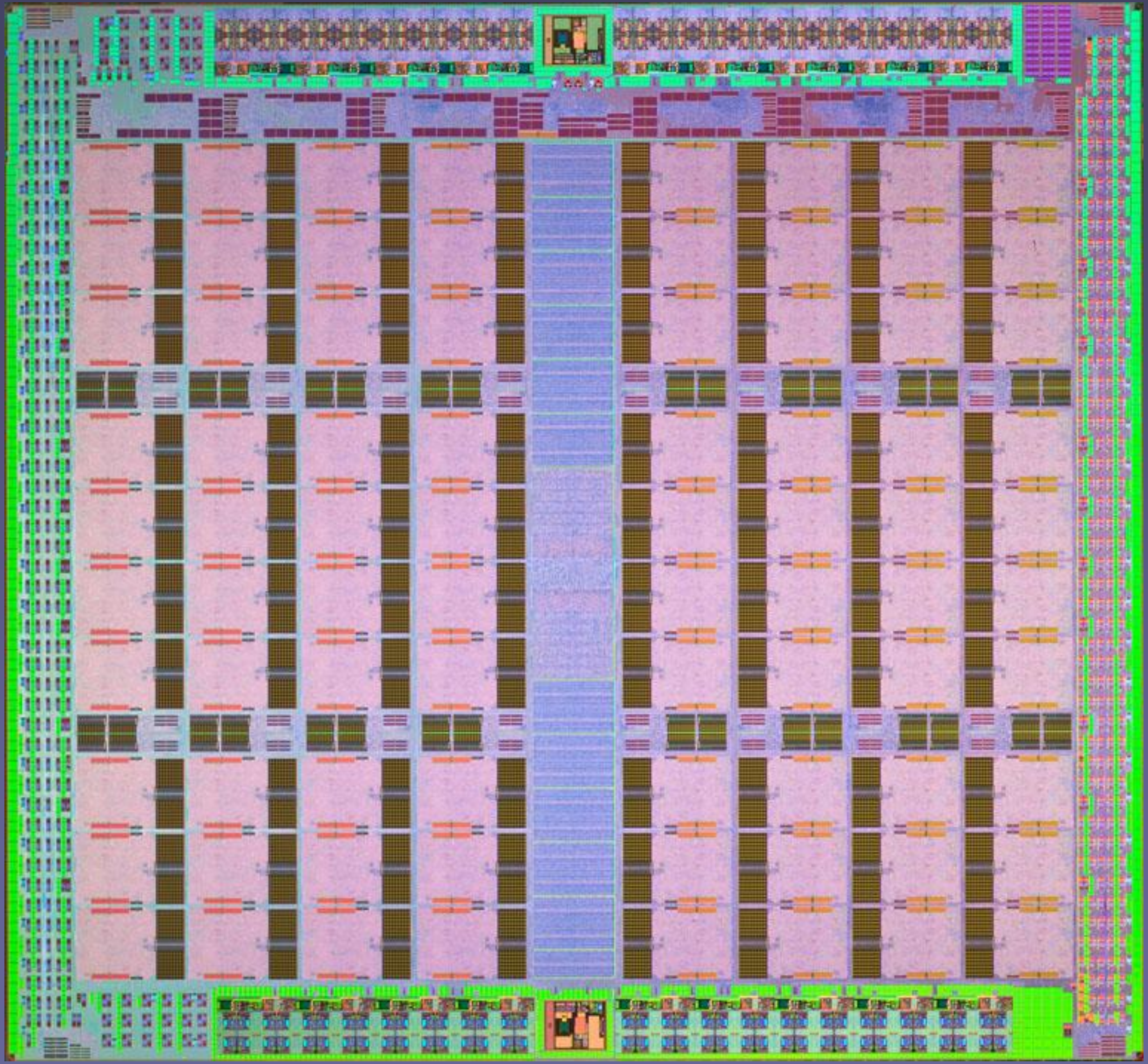
- While timing failures often are perfectly deterministic (just hard to nail down), there are truly transient errors, often too expensive to protect against. To this category belong “soft errors” caused by high energy events.
- Memories, due to their regular structures, can be made reliable using coding theory methods.
- There is no such strong protection for functional units and random logic. Only multipliers and adders can be build with error-detection capability. One ought to expect soft failures in modern (32-14 nm) technology, even at the sea level.

Be Prepared!

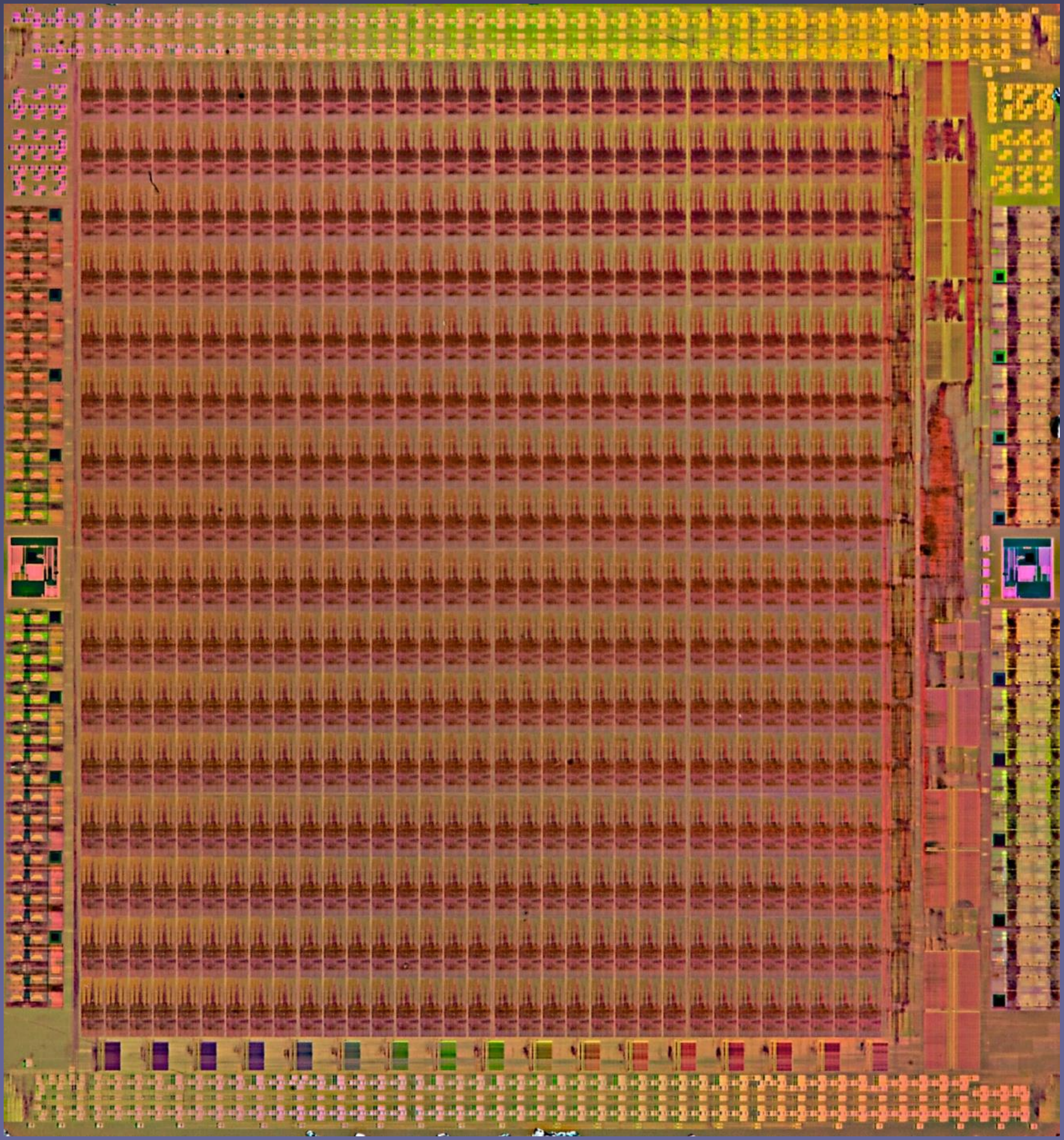
- Expect failures – one can only
- -- eliminate the deterministic static and timing failures
- -- minimize the probability of transient errors through redundancies, and checking/repeating operations.

- Example of expected rare event. Clock jitter – clock generator gives you ~10ps jitter, but in 1 year of Chip B machine's life one expects at least one 9.5 sigma-event. This gives a 95 ps jitter on the clock source, which has to be built in as a hard assertion in the beginning of chip's design.

Chip A – “cyclops”



Chip B – “mrbit”



Acknowledgements

- This work was performed in cooperation with T. Morgan (IMAS), W. Donath (IBM), and other members of IBM T.J. Watson Research Center.
- We thank M. Denneau (IBM) for his long-term commitment to new computer architectures.

The Scale

- The size of the smallest rectangular feature is about 25 nm

- Dimensions of chips A and B are 22.8 mm X 21.2 mm

How one works on the Physical Design?

- With great difficulty
- In 90nm “mature designs” we had –
- 40 million complex cells in the chip
- 200 million wire linear segments on the chip
- 170 million vias on chip connecting 10 wiring metal layers
- Total length of all wires (independently of their width) is over a mile (more than 1.7 kilometers).

- In 22 nm designs everything is multiplied by 16!
- You need all your mathematical tools to make it happen

Cyclops board -- 2008