Repair and Recovery

- Repair: How do we eliminate faults that won't clear up over time?
- Recovery: How do we correct any corrupted computation?

Repair and Recovery

- Repair: How do we eliminate faults that won't clear up over time?
  - Granularity
  - Design for repair
  - Other issues
  - Interfaces

- Recovery: How do we correct any corrupted computation?

Granularity of Repair

- Gate, FF, microarchitecture, core?
- Tradeoff complexity, overhead, testability/validation
  - E.g., Core level easier to implement, but higher overhead(?) vs. microarchitecture level repair may need more testing
  - Is gate level repair ridiculous?
    - Logic devices minimally designed – Vdd, delay knobs
    - Vs. analog designed with many knobs
    - Technologies where logic devices have different knobs???
- Related to the fault model
- Interaction with detection, diagnosis, testing
  - Must isolate fault to repair granularity, determine what repair needed
- **What is the sweet spot?**

Design for Repair

- Graceful degradation vs. cold spares
- What are surplus resources, how to allocate for repair (wires in fpgas)
- One of a kind components vs. inherent redundancy
- Testing, self-check, isolation
- Depends on application and organization
- **How do we connect parts into systems that do repair?**

Other Issues

- Value-add with repair
  - Use spares for upgrades
- Business model – currently sell based on end of life performance
  - Can we sell part at best case?
    - Big servers – s/w licenses based on performance of machine (vs. laptops)
  - Getting the best from your system
- Who is responsible for repair
  - Firmware? Multilayer?
  - Interfaces (separate slide)
- Control timeframe of degradation – delay/control repair?
- Connection with adaptation

Interfaces

- What hooks do we need to enable repair?
  - Need some standards
  - Accommodate application-specific needs
  - Independent of OS
- Think about the problem hierarchically
  - Lower level tells upper level what its repair features are, impact of those features
  - Higher level decides how to use them based on application
- Each layer should decide how much detail it needs
  - API should have two interfaces:
    - Alarm interface: higher layer tells the lower layer what the alarm level is
    - Detailed interface: if or when needed
- Security

Recovery

- Acceptable layer for recovery depends on fault rate/model and application
- Availability important
- I/O issues