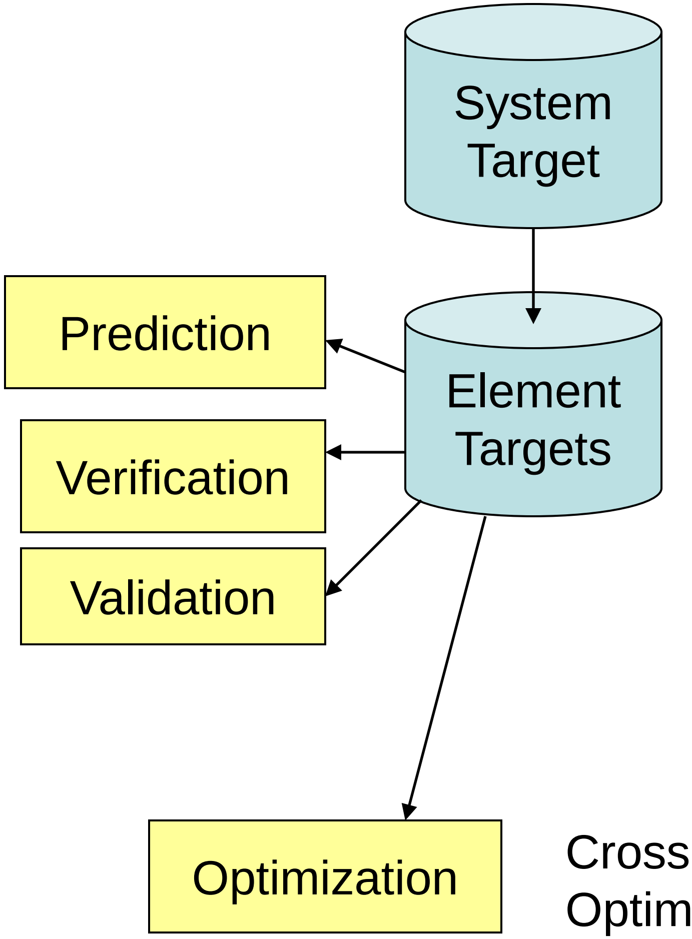
Cross-Layer Reliability Metrics

Subteam Update

with Whole Team

October 29, 2009

Metrics



Customer provides quantitative target
(Gov. program; commercial customer)

Supplier divides up the
System target into elements
Targets (e.g. processors,
Network adapters, etc.)

Crosses levels (application SW/OS/protocol/FW/HW)
Optimize Reliability as QoS with performance, power & cost
All of the above at runtime

Agency Program Management Perspective

- Define the error modes that matter at the system level
  - Hard errors have detected, uncorrectable (DUE) and detected, correctable (DCE) components*
    - FIT Rate for hard errors leading to system failure
    - FIT Rate for hard errors leading to partition failure (one OS instance fails, job is lost)
    - FIT Rate for hard errors that are recovered (fail over)
  - Soft errors also have DUE and DCE components
    - FIT Rate for soft errors leading to system failure (crashes, checkstops, hangs)
    - FIT Rate for soft errors leading to partition failure
    - FIT Rate for soft errors that are detected and corrected (e.g. instruction retry)
    - SDC FIT Rate for undetected soft errors
- Target requirements would include system failure rates, partition failure rates, and SDC
- Conditions for prediction, verification & validation would be included
  - Error rates would be evaluated for worst case and typical conditions

*a hard error can manifest first as undetected but will eventually get detected

Target Metrics Drive Evaluation Metrics

- Evaluate the cross-layer design and make the changes needed to deliver on targets
- Optimize the cross-layer design and make the changes needed to deliver on targets F(reliability, cost, performance, power)
- Verify that the design meets targets
  - Has the protection really been implemented?
  - Does detection and recovery work?
  - Statistical fault injection to verify error rates/vulnerability
- Validate that the hardware + firmware meets targets
  - Worst case and typical case workloads (at least)
  - Standardized particle beam testing (neutrons & protons)

Standardized Error Rates (original)

Grade/Level	FIT Rate	Description
1	< 0.001	0.001 FIT ~ 1 in 100M yrs
2	0.001 to 0.01	0.01 FIT ~ 1 in 10M yrs
3	0.01 to 0.1	0.1 FIT ~ 1 in 1M yrs
4	0.1 to 1.0	1 FIT ~ 1 in 100,000 yrs
5	1 to 10	10 FIT ~ 1 in 10,000 yrs
6	10 to 100	100 FIT ~ 1 in 1000 yrs
7	100 to 1000	1,000 FIT ~ 1 in 100 yrs
8	1000 to 10,000	10,000 FIT ~ 1 in 10 yrs
9	10,000 to 100,000	100,000 FIT ~ 1 in 1 yr
10	100,000 to 1M	1M FIT ~ 1 per month

Apply to the whole system and each of the chips in the system

Standardized Error Rates (inverted at workshop)

Grade/Level	FIT Rate	Description
1	100,000 to 1M	1M FIT ~ 1 per month
2	10,000 to 100,000	100,000 FIT ~ 1 in 1 yr
3	1000 to 10,000	10,000 FIT ~ 1 in 10 yrs
4	100 to 1000	1,000 FIT ~ 1 in 100 yrs
5	10 to 100	100 FIT ~ 1 in 1000 yrs
6	1 to 10	10 FIT ~ 1 in 10,000 yrs
7	0.1 to 1.0	1 FIT ~ 1 in 100,000 yrs
8	0.01 to 0.1	0.1 FIT ~ 1 in 1M yrs
9	0.001 to 0.01	0.01 FIT ~ 1 in 10M yrs
10	< 0.001	0.001 FIT ~ 1 in 100M yrs

Apply to the whole system and each of the chips in the system

Layers: Why Cross-Layer Metrics?

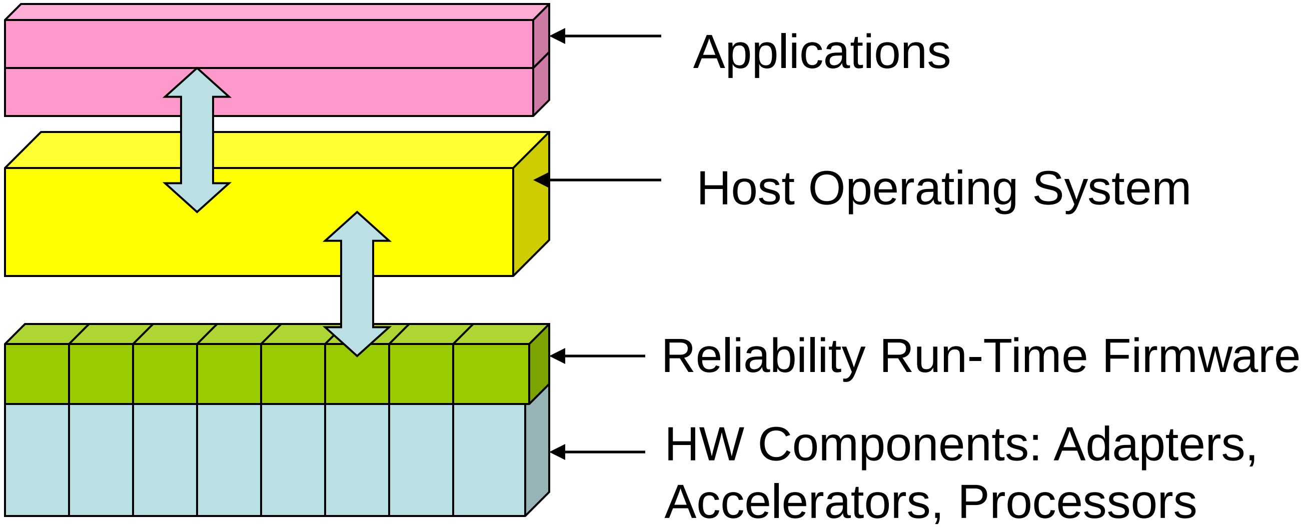
Error rates are not "just" the sum of hardware components

They depend on the RAS functions including firmware implementations

They depend on what the operating system does

They depend on the applications running

Reliability Metrics Need to Cross Layers


Applications
Host Operating System
Reliability Run-Time Firmware
HW Components: Adapters, Accelerators, Processors

Metrics for Circuits (Roadmap Team)

- FIT per Million Circuits
- DUE FIT (e.g. permanent latch failures)
- DCE FIT (e.g. “raw SRAM FIT” detected, corrected by ECC) *aha! A cross-layer metric!*
- SDC FIT (e.g. extrinsic noise + marginal device)

Improving Accuracy of SER Prediction

- Need tools that identify protocol checked bits and filter these
- Need tools to identify when software is doing checking (e.g. Oracle metadata attributing and checking) and filter these
- SER depends on workload, so we need tools that generate worst and typical test cases
- Need tools to run the test cases and generate a system level SER
- Tools need standard ways & definitions (metrics) to describe the above effects
- If we don’t, we will overpredict or underpredict

Verification

- Need tools to check if the design targets are met
- Check circuit protection
- Statistical fault injection to mimic beam experiment in simulation
- Detect cross layer protection (e.g. checksum, crc)
  - Avoids overdesign
  - Avoids overestimation
- All of the above for multiple sources of IP
- Generation of test cases (e.g. “virus” worst case test; typical test)
- Standardized verification methodology and metrics
  - Standardized test cases/workloads
  - Use the standard metrics as described in “targets” page

Validation

- Standardized hardware validation (test cases/workloads; standard metrics for reporting)
- Include firmware *cross-layer*
- Include system level interventions (e.g. OS intervention for recovery) *cross-layer*

Software Level

- Application software can have its own checking
- Applications can have sensitive sections of code vs less sensitive & insensitive sections
- Cross-level opportunity here
  - Compiler detection
  - Code annotation
- Send sensitivities to lower levels for optimization *cross-layer metrics*

Optimization Opportunities

- Optimize reliability, power, cost, performance
- Dynamic run-time optimization methodologies and algorithms
- Metrics to send the information across layers
- Scheduling that includes all of the above
- Example: datacenter is running a set of applications over many partitions. Know the work being done where on system of nodes; knows how much power is being consumed; knows how customers are being billed and QoS of everyone’s jobs; Optimize the current job in the context of everyone else to deliver QoS required (and keep the power nice)

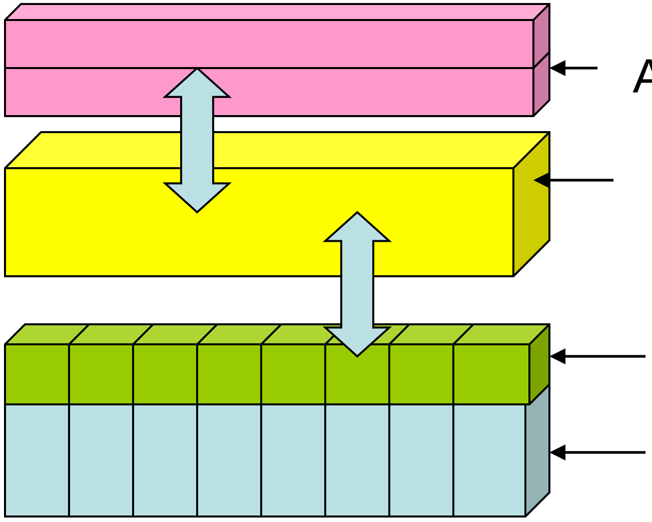
Variable Reliability

Vision: Future Heterogeneous Systems will have Standard Interfaces for Tunable Reliability

Interfaces will pass attributes to execute tunable reliability

Metrics provide the measures by which the ***System*** can quantitatively assess and control its reliability based upon its components..

Reliability Metrics Need to Be Passed Across Layers



Applications

Host Operating System

Reliability Run-Time Firmware

HW Components: Adapters, Accelerators, Processors

Backup

- Charts from July workshop follow
- Subhasish, please include in writeup
- Charlie and Prabhakar need to complete the equations
- Alan offered to help with the writeup

Outline of Report

- Subteam Members
- Subteam Process and Timeline
- Problem Statement and Subteam Mission
- Address each of the key points made in the Mission Statement
  - Pain points: More heterogeneous, more IP from more places in more formats some of which are not readable, large systems are getting larger
  - Metrics needed to fill the gap between current and needed prediction
  - Need for cross-layer metrics
  - Metrics for validating designs
  - Metrics to classify grades of reliability
  - Metrics for prognostics
  - Metrics for optimization
  - Metrics for variable runtime reliability

Predicting Chip-Level SER Today

$$SER^{derated}(chip) = \sum_{\mathrm{circuits,\ macros}} SER^{\mathrm{nominal}} * \mathrm{TD} * \mathrm{LD}$$

- No commercial tool (yet) available
- Some use proprietary tools, which take nominal SER of state holding devices from lookup tables (from prior simulations)
- Some roll up circuit SER into a chip level accounting (e.g. spreadsheet) by macro/unit
  - For each macro, look up element SER based on simulations
  - LD may be assigned globally through empirical testing or simulation, or at the unit or macro level

Soft vs Hard in Prediction

- Hard error rates will always surface because they are “stuck errors” (unlike soft errors that don’t repeat, they are “toggled errors”)
  - So they don’t get derated (derating factor is always “1”)
  - They might not surface until they get stimulated (workload dependence) but you can count on them to stick
- What matters are the sum of error rates and whether there is recovery
- Metrics to describe hard error rates at the system level exist:
  - FIT rate for errors
  - FIT rate for unrecoverable errors
  - FIT rate for recovered errors (failover)*
- *History such as failover and current configuration and wear go into prognostics

Metrics for Accurate Error Rate Prediction

Requirements for metrics

- Common language for system integration requirements
- Measure both current and prognostic reliability
- Common requirements for composition of large scale systems, and smaller systems
- Correlation between system components
- Capture both SER and HER

Error Rate Dependencies

- Error rates (both current and prognostic) for both components and system are affected by
  - Environment
  - Configuration
  - Utilization

Component Error Rate

- Define error rate as a range COMP_i ER = [min-max] for components/hierarchies over:
  - Configurations in which component is used within system
  - Environment in which a component may be used
  - How the workload uses the component (hit rate and line usage in memory for example)

System integration

- $EFF\_ER^{COMP\_i}$ = FUNCTION1 (ER^{COMP_i}, $CORR_j^{COMP\_i}$, $UTIL^{COMP\_i}$)

- Such a function will be computed hierarchically where each node in the hierarchy becomes a component at the next level
  - ER is rated error rate of component i under certain conditions
  - $CORR_j^{COMP\_i}$ is the correlation variable that captures the relationship between the error rate of component i and other components j in the system
    - for example, the data rate of an IO device connected to a bus may be limited by data rate of bus
    - DIMMs may be configured many different ways based on other components in system
  - $UTIL^{COMP\_i}$ is a variable that captures the dependency of the error rate on the workload
- ER can be defined as any one of SDC, Checkstops, performance loss etc.

Variable Reliability

- Example 1:
  - Chip xyz is showing signs of wearout
  - It switches a wear indicator bit to “on” *(reliability metric)*
  - System middleware detects the xyz wear indicator bit has flipped to “on” and sends message to console “XYZ running in degraded mode” and field repair action is initiated *(reliability metric is passed between component to host)*

- Example 2:
  - Chip xyz1 is showing signs of wearout
  - It switches a wear indicator bit to “on” *(reliability metric)*
  - System firmware fails xyz1 out to spare xyz2 and sends message to console “XYZ failed and swapped to spare” and field repair action is initiated to replace spare *(reliability metric is passed between component to host)*

- Example 3:
  - Workload abc requires mainframe data integrity on accelerator efg but efg is a commodity part
  - Workload task carries a QoS bit *(reliability metric)* that indicates “mainframe reliability” and runtime software launches duplicate tasks on duplicate accelerators *(reliability metric is acted upon by runtime software)*
  - Results are crosschecked to be correct and result is sent to host

Prognostic error metrics

- Predict future error rate (system fragility) based on knowledge of components
  - Example: if spare (processor/redundant line) etc are already used up, prognostic error rate is high
- Predictability of such an error rate and/or sensitivity of a component
  - Will help pre-empt failure
  - Identify critical components on the verge of failure AND whose failure would cause system wide outages and/or SDCs
  - Focus service requests requirements

A prognostic equation

- Prognostic System Error rate =
  - ∑ FUNCTION2 ($\text{EFF\_ER}^{\text{COMP\_i}}$, $\text{CRIT}^{\text{COMP\_i}}$, $\text{UTIL}^{\text{COMP\_i}}$, $\text{FAIL/ENV\_STATUS}^{\text{COMP\_i}}$);

- Variable $\text{FAIL/ENV\_STATUS}^{\text{COMP\_i}}$ is used to capture the current state of fragility of component or hierarchy

Work to do

- In the context of some full systems, and diverse application domains (HPC to consumer), define:
  - 1. Define ∑j CORRj $^{COMP\_i}$, for each component/hierarchy i, sum the correlation of error rate between component i and all other j components in the system precisely
  - 2. Define CRIT $^{COMP\_i}$, i.e., criticality of component i in the system precisely
  - 3. FAIL/ENV_STATUS $^{COMP\_i}$, potential of component error rate to increase (either SER or HER based on current failure/environmental conditions precisely.
  - 4. Precise definition for UTIL$^{COMP\_i}$. This may be tricky based on component type (processor, memory, IO, disk etc).
- Identify case study systems and evaluate these and other required metrics