Hardware Level Fault-Tolerance Techniques

EECE 513: Design of Fault-tolerant Digital Systems

Learning Objectives

- List the techniques for improving the reliability of commodity & high end processors
- Design coding techniques for memory soft errors and evaluate their trade-offs
- Understand the benefits of chipkill ECC, sparing and scrubbing
- List the techniques used in the I/O sub-system
- Putting it together: Stratus Case Study

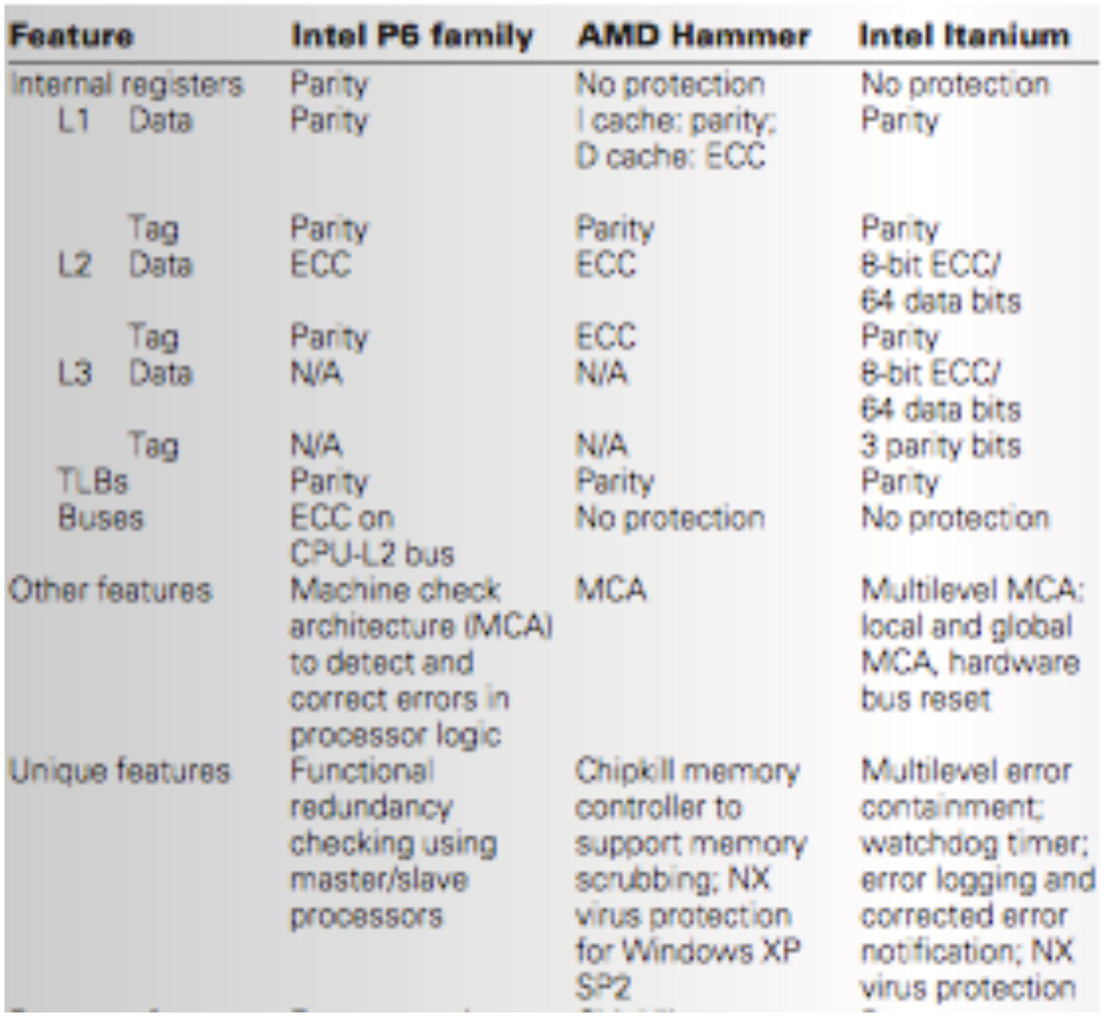
High-Availability Systems

- **IBM G5 Mainframes**
  - Duplicated execution units on each core
  - Redundant CPU logic
  - Inline checking in I/O sub-system
  - ECC in memory and registers
- Error Recovery is accomplished using instruction retry
  - Transparent to the S/W

- **Tandem Non-Stop**
  - Duplicated processors running in lock-step
  - Process pairs for checking
  - End-to-end disk checksums, CRC
  - ECC in memory only
- Error recovery is achieved by swapping in backup processes
  - S/W needs to be involved in the failover

Commodity Micro-processors

Source: **Recent Advances and New Avenues in Hardware-Level Reliability Support,** by Iyer, Nakka, Kalbarczyk and Mitra, IEEE Micro 2005.



Feature	Intel P6 family	AMD Hammer	Intel Itanium
Internal registers	Parity	No protection	No protection
L1 Data	Parity	I cache: parity; D cache: ECC	Parity
Tag	Parity	Parity	Parity
L2 Data	ECC	ECC	8-bit ECC/ 64 data bits
Tag	Parity	ECC	Parity
L3 Data	N/A	N/A	8-bit ECC/ 64 data bits
Tag	N/A	N/A	3 parity bits
TLBs	Parity	Parity	Parity
Buses	ECC on CPU-L2 bus	No protection	No protection
Other features	Machine check architecture (MCA) to detect and correct errors in processor logic	MCA	Multilevel MCA: local and global MCA, hardware bus reset
Unique features	Functional redundancy checking using master/slave processors	Chipkill memory controller to support memory scrubbing; NX virus protection for Windows XP SP2	Multilevel error containment; watchdog timer; error logging and corrected error notification; NX virus protection

Learning Objectives

- List the techniques for improving the reliability of commodity & high end processors
- Design coding techniques for memory soft errors and evaluate their trade-offs
- Understand the benefits of chipkill ECC, sparing and scrubbing
- List the techniques used in the I/O sub-system
- Putting it together: Stratus Case Study

Memory Errors: History

- Memory elements have long been the target of soft-errors since the late 70's
  - In 1978 May and Woods reported "A New Physical Mechanism for Soft Errors in Dynamic Memories"
  - In 1979, "Alpha-Particle-Induced Soft Errors in Dynamic Memories."
  - SRAMs saw problems approximately 2 years later

Soft Errors Today

Baumann, R.; , "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," *Electron Devices Meeting, 2002.*

DRAMs

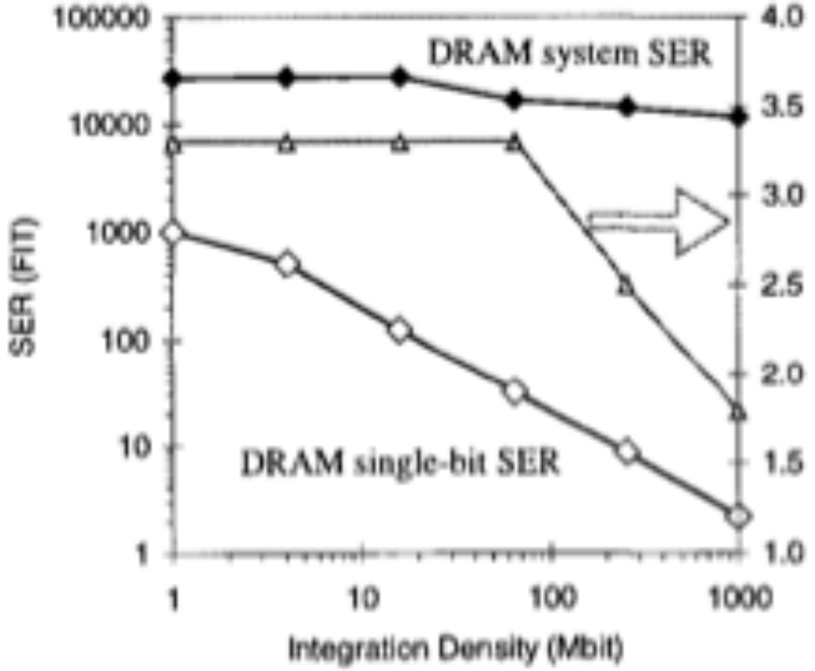


Figure 2. The single bit (white diamonds) and system (black diamonds) SER trends for DRAM as a function of technology node. The operating voltage at each node is represented by the curve with gray triangles.

SRAMs

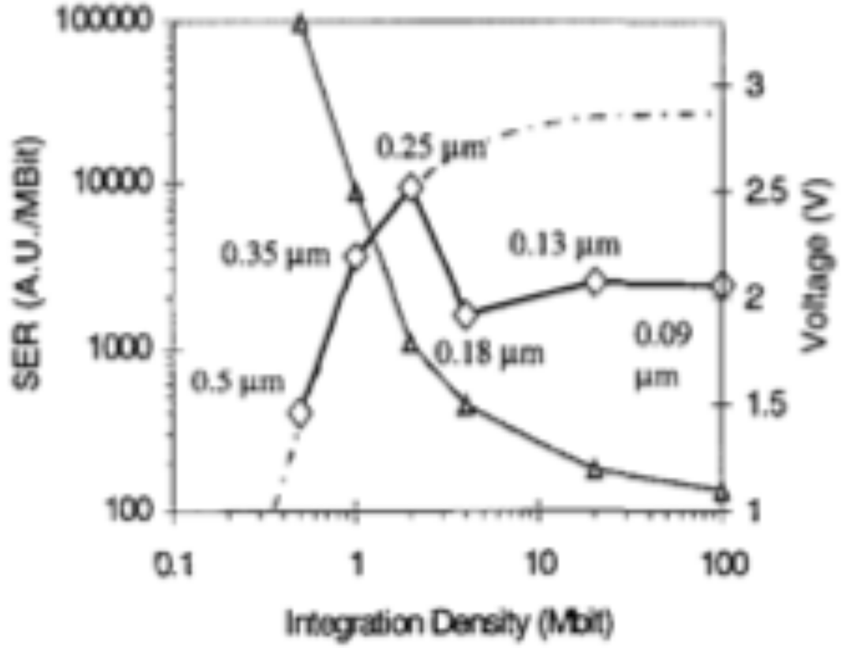


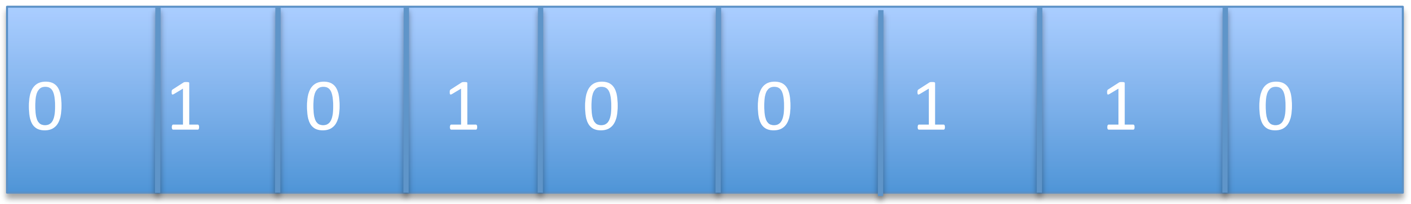
Figure 3. The single bit SER trend (white diamonds) in SRAM devices as a function of technology node. The rapid scaling down of operating voltages is evident shown by the curve with gray triangles. The ~ 10x reduction in SER after the 0.25 µm node is the effect of removing BPSG.

Error Trends in today's memories

- DRAM reliability has remained relatively constant over many years
  - Thanks to improvement in fabrication
  - May be different in eDRAMs and mobile DRAM
- SRAM reliability becoming an increasing concern with shrinking cell sizes and voltage
- DRAM hard errors are emerging as a problem [Schroeder'09][Dell'08]

Parity Protection - 1

- Single bit added to each memory byte/word to detect a single error
  - Cannot detect multiple errors
  - Cannot correct the error
  - Affordable alternative to ECC memory



Parity bit – even parity

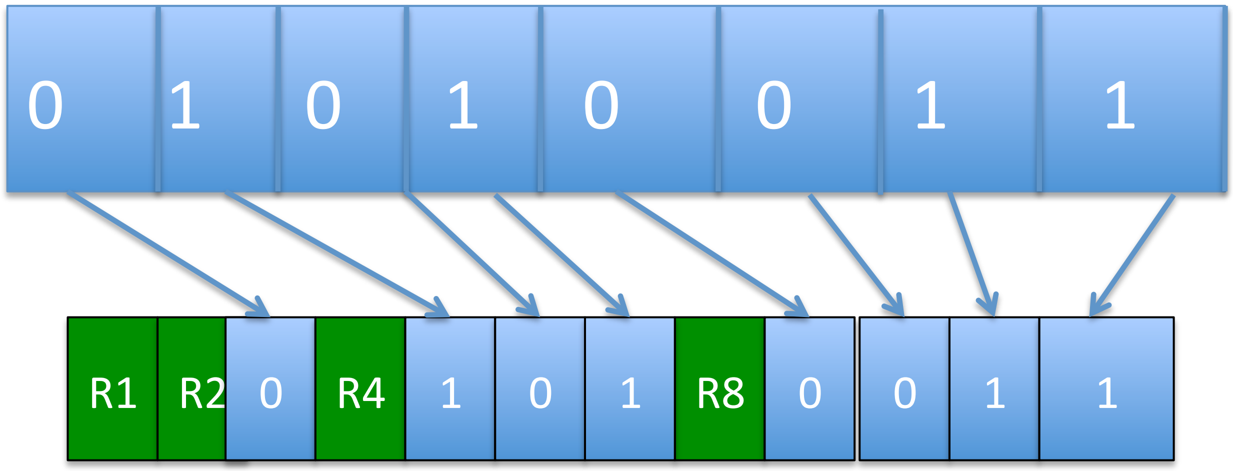
xp = x0 ^ x1 ^ x2 ^ x3^ x4 ^ x5 ^ x6 ^ x7

Parity Protection - 2

- Requires an additional operation on reads/writes to memory → extra access latency
- Circuitry to compute parity bit is simple, but requires additional area and power
- Used mainly in SRAM structures where error rates were low and access times are important
- For DRAMs, no added benefit of using parity over ECC when the memory data width is greater than 8 bytes

ECC Memory - 1

- For every memory word of size 'n' bits, we need at least $\log_2(n)$ bits of ECC memory
  - For 64 bit memory, we need at least 6 bits of ECC
  - The check bits are distributed throughout word
  - Each bit is protected by multiple checkbits given by the index (the sum of the checkbits matches index)

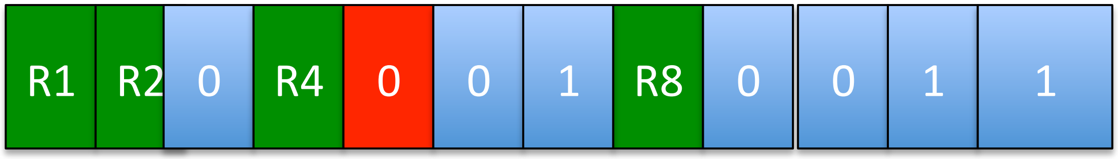


R1 = R3^R5^R7^R9^R11
= 0 ^ 1 ^ 1 ^ 0 ^ 1
= 1

R4 = R5^R6^R7^R12
= 1^ 0 ^ 1 ^ 1
= 1

ECC Memory - 2

- Let’s say you had a single bit error in R5 (1→0)



Check Bits are recomputed and compared.

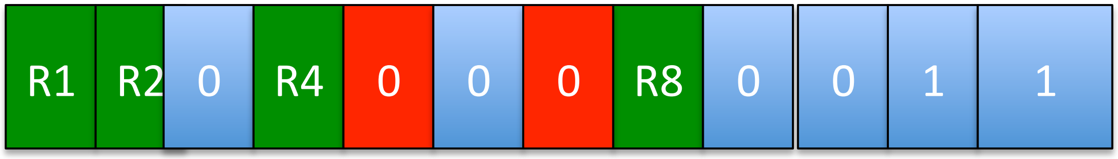
R1 = R3 ^ R5 ^ R7 ^ R9 ^ R11 = 0 ^ 0 ^ 1 ^ 0 ^ 1 = 0

R4 = R5 ^ R6 ^ R7 ^ R12 = 0 ^ 0 ^ 1 ^ 1 = 0

Both check-bits R1 and R4 differ from their computed values. These are called the syndromes. So we can infer that the bit R5 had an error in it, and can correct the error.

ECC Memory - 3

- Let's say you errors in bits R5 and R7 (double-error)



Let's compute check-bits R1, R2 and R4

R1 = R3 ^ R5 ^ R7 ^ R9 ^ R11 = 0 ^ 0 ^ 0 ^ 0 ^ 1 = 1 (Same as prior value)

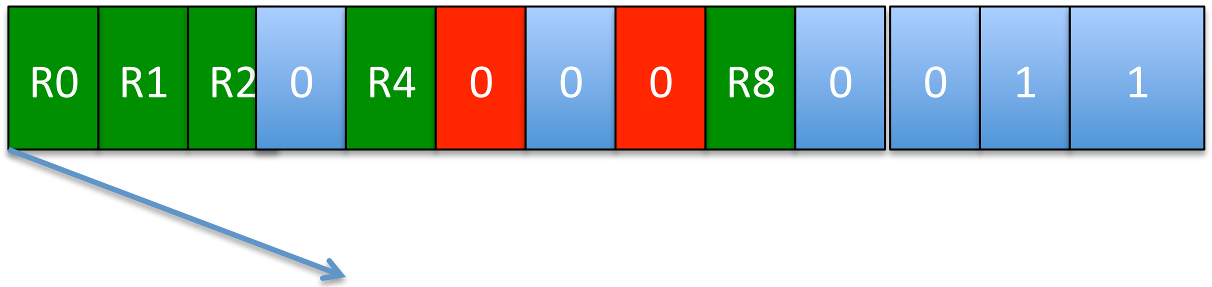
R2 = R3 ^ R6 ^ R7 ^ R10 ^ R11 = 0 ^ 0 ^ 0 ^ 0 ^ 1 = 1 (Differs from prior value)

R4 = R5 ^ R6 ^ R7 ^ R12 = 0 ^ 0 ^ 0 ^ 1 = 1 (Same as prior value)

How do we distinguish this case from the one where bit R2 is corrupted ?

ECC Memory - 4

- Add an extra parity bit R0 for the entire word



Extra parity bit for the word is added

- In the case of a single bit error, both syndrome bit(s) and R0 bit will differ → can be corrected
- In case of a double error, only syndrome bit(s) differs → can be detected but not corrected

ECC: Implementation Trade-offs

- ECC memory is not free !
  - Performance overheads for read/write operations
    - 3 to 4 % more for PC133 CAS2 ECC SDRAM
    - Up to 33 % for high-speed SRAMs
  - Area overhead for error-detection/correction ckts
    - 20 % die overheads
  - Additional costs as chipset support is needed
    - 10 to 25 % more for entire chip
  - Effectiveness: Corrects more than 90% of errors

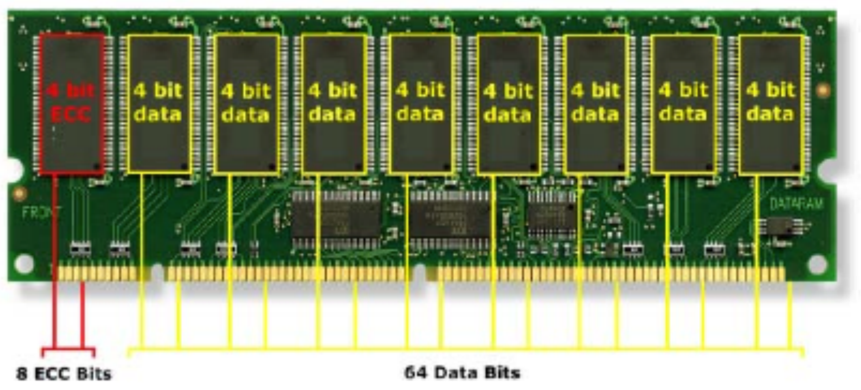
Above nos. are from the Terazzon white paper.

Learning Objectives

- List the techniques for improving the reliability of commodity & high end processors
- Design coding techniques for memory soft errors and evaluate their trade-offs
- Understand the benefits of chipkill ECC, sparing and scrubbing
- List the techniques used in the I/O sub-system
- Putting it together: Stratus Case Study

ChipKill ECC - 1

- ECC can detect 2 bit and correct 1 bit errors
  - Provided the entire memory chip does not fail
  - Chip failure can lead to data loss even with ECC



Traditional SEC/DED ECC for a 64-bit word with eight check-bits of ECC

ChipKill ECC - 2

- Solution: Use Chip-kill ECC ™ (IBM S/390)
  - Spread the ECC check bits over multiple chips
  - Bit-steering → Steer the checkbits of adjacent bits in a memory word to different words in the ECC

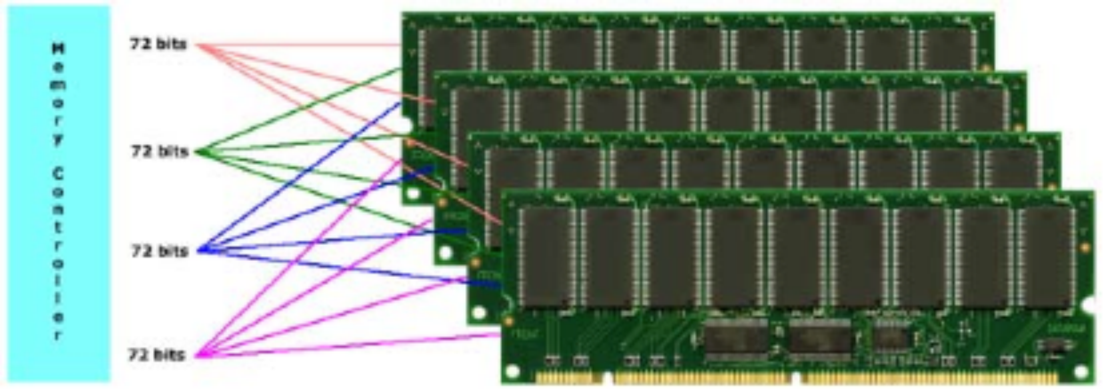


Figure 2

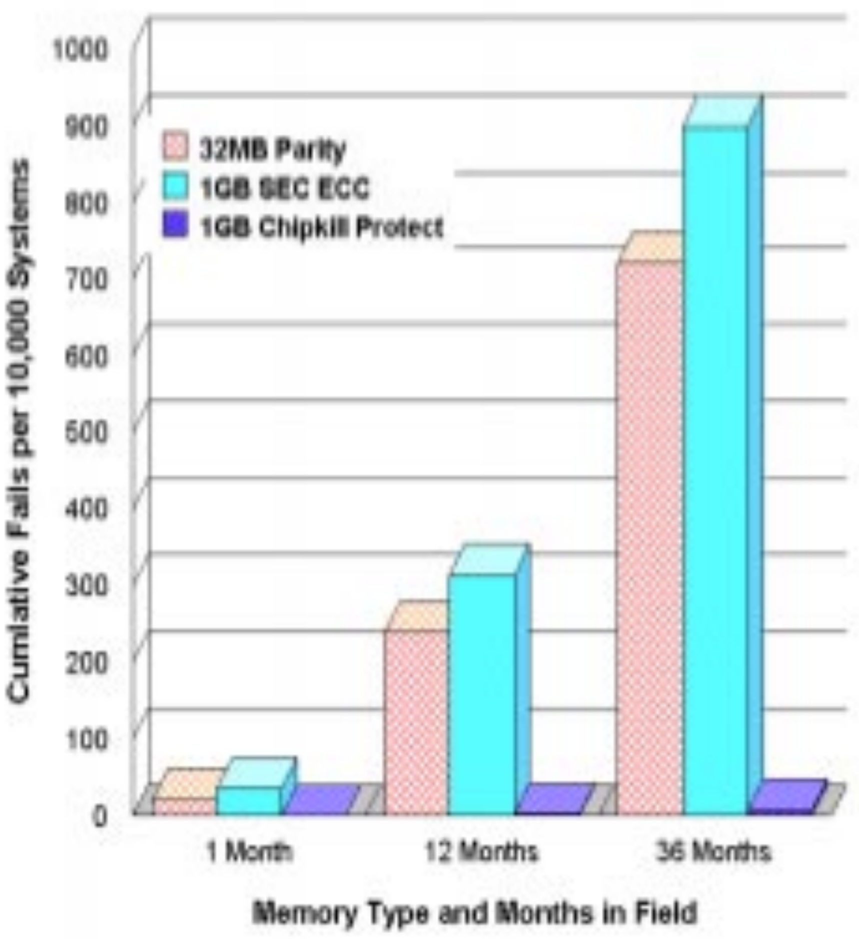
Chip-kill ECC
Note how the bits are scattered across different modules

ChipKill ECC: Implementation Tradeoffs

- Incurs four times the overhead of traditional ECC
  - Can be optimized using very wide ECC words
  - Provide detection of chip failures but not correction
- Compaq proposed a clever interleaving solution to combine two ECC words into one module
  - Provides the benefits of Chipkill ECC with only as much cost as parity protection
  - After a chip has failed, the Compaq ECC is unable to provide protection from single/double bit errors

Parity, ECC and ChipKill- Comparison

- Simulation data gathered by IBM over 36 months comparing:
  - 32 MB Parity protected memory
  - 1 GB SEC ECC
  - 1 GB Chipkill ECC



Other variations of ECC

- **Scrubbing**
  - ECC memory only checks the bits during reads/ writes
  - However, infrequent accesses may lead to bit errors accumalating
  - Solution: Scrub memory periodically by performing reads/writes to unaccessed memory

- **Sparing**
  - Correlated or large area defects cannot be combated with ECC alone
  - Use spare rows/columns in conjunction with ECC
  - Leads to an order of magnitude reliability improvement over ECC alone for hard faults

Learning Objectives

- List the techniques for improving the reliability of commodity & high end processors
- Design coding techniques for memory soft errors and evaluate their trade-offs
- Understand the benefits of chipkill ECC, sparing and scrubbing
- List the techniques used in the I/O sub-system
- Putting it together: Stratus Case Study

I/O Sub-system - 1

- Disk and other storage media protected using RAID technologies
  - Fairly mature, industry standard
  - However, data is susceptible when it is buffered
  - Firmware controllers and I/O processor errors

- Need to ensure end-to-end consistency of data from I/O initiation to disk read/write

I/O Sub-system - 2

- Techniques for end-to-end I/O checking
  - Checksums on data before and after reads
  - Checking of header fields for consistency
  - Watchdog timer for ensuring no deadlocks or livelocks of I/O devices
  - System-level consistency checks. e.g., read back data written to disk in chunks and check them
  - Use multiple file organizations to store data

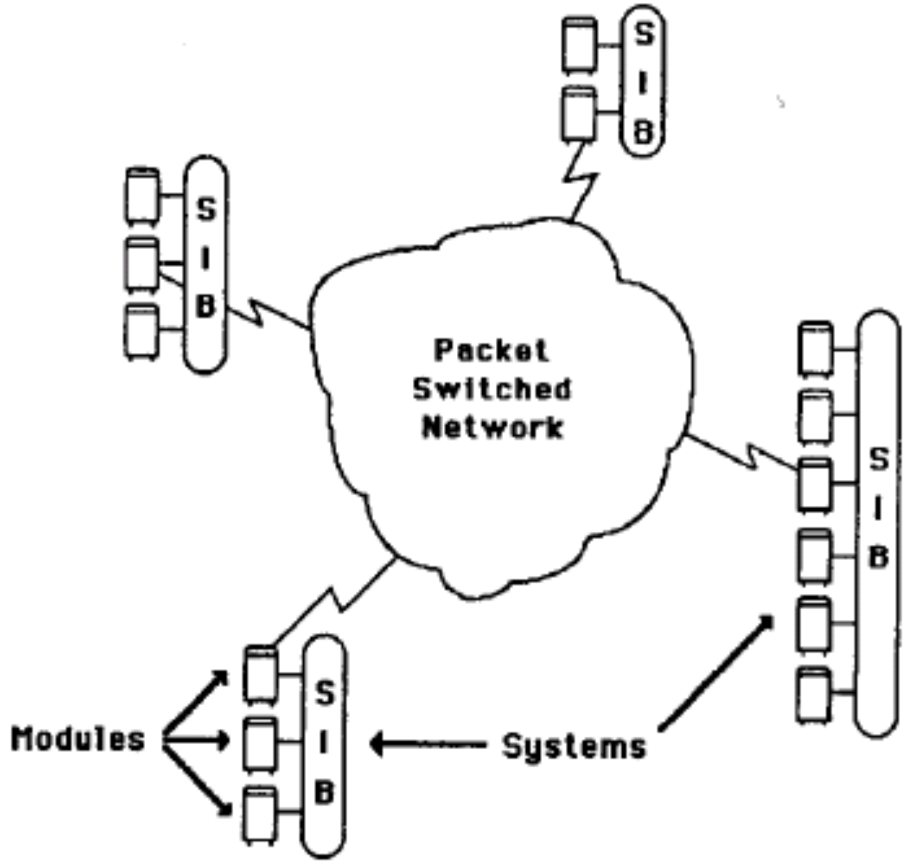
Learning Objectives

- List the techniques for improving the reliability of commodity & high end processors
- Design coding techniques for memory soft errors and evaluate their trade-offs
- Understand the benefits of chipkill ECC, sparing and scrubbing
- List the techniques used in the I/O sub-system
- Putting it together: Stratus Case Study

Stratus System Architecture Overview

- System composed of up to 32 computers connected through Stratus bus called *StrataLink*
  - Two independent coaxial links
  - System software monitors if a link is up and can switch over in case of failure
- Each computer has a backplane bus called *StrataBus*
  - Major boards interface with the StrataBus
  - 32 logical slots in StrataBus, grouped into two independent sets, say Bus A and Bus B with independent power supply
  - Each board does its own power regulation
  - Each board interfaces with both buses
  - Parity signals on each bus detects bus failures
  - Additionally, memory boards detect bus problems such as open bus lines, shorted bus lines, or failure of bus driver on a board

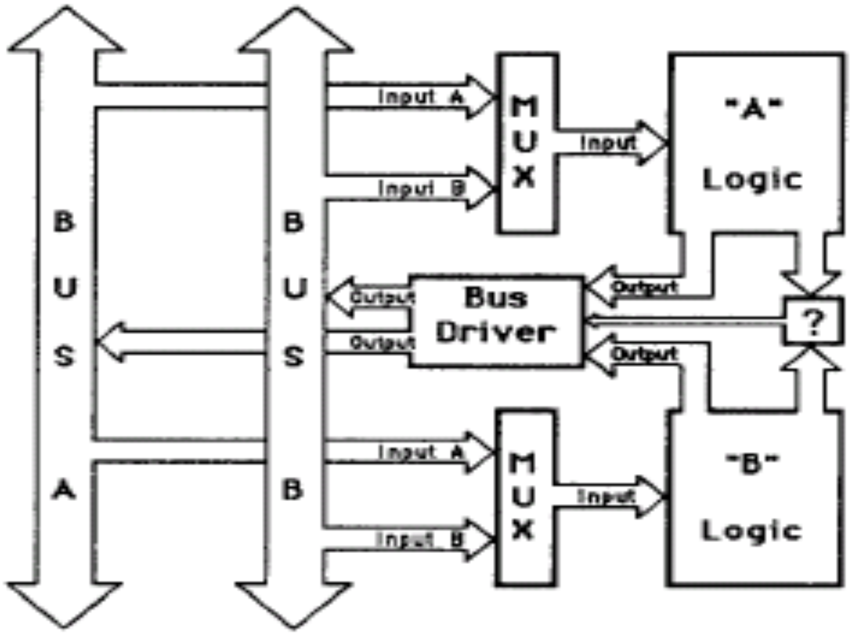
Stratus System View



Stratus Hardware Fault-Tolerance

- Enough battery backup to do stage-wise shutdown
- System Boards
  - Self-checking by having independent duplicate logic run synchronously
  - Outputs run through comparator which run the bus drivers
  - Requirement for self-checking is there be no don't care states in FSMs
  - Boards almost always occur in pairs, called *partner boards*
  - Partner boards may run in
    - Synchronous lockstep: Send same signals to StrataBus at same clock tick
    - Logically paired: System software keeps the pair in sync
  - Partner boards are placed in even-odd slots of StrataBus

A Self-Checking Board



- A pair of identical logic
- Comparator enables or disables bus driver
- Incoming signals from bus are fed through a multiplexer

Fault-Tolerant Disk Subsystem

- Organized into logical volumes with each disk paired
- Two disks run through different disk controller boards
- OS uses several techniques to ensure the duplexed disk contains the same data – *disk mirroring*
- Each block of data written to disk is checksummed and checksum written to disk
- Any disk block that goes bad is remapped to an alternate block
  - Diagnosis of bad block during reading creates a delay and read data is written to a new block
  - Diagnosis of bad block during writing creates a delay and new block is allocated and data written serially to both blocks

Disk Recovery Scenarios

- What if controller fails?
  - OS keeps track of writes to the mirrored disk driven by the functioning controller
  - When bad controller is repaired and brought back online, only incremental writes are done to the disk on that controller
  - Called *Fast Recovery*
- What if disk fails?
  - When repair or replacement is done, all data from functioning disk is written to new disk
  - Dual port disks so parallel reading and writing can go on, but done serially for fault tolerance purposes
  - Called *Normal Recovery*

What If Comparator Fails?

- Reports false failure
  - One of the boards in the pair is diagnosed as faulty and taken offline
- Misses reporting a failure
  - Possibility 1: Bad board drives a ‘0’ on the bus and functioning board drives a ‘1’. OR-ing puts the correct data on the bus
  - Possibility 2: Bad board drives a ‘1’ on the bus. Parity checking logic on the bus detects a problem and takes the bus offline
  - Potential for problem in the latter case if both buses are taken offline
- Liveness failure
  - Board does not put any value on the bus
  - A bus monitoring process to detect activity

Learning Objectives

- List the techniques for improving the reliability of commodity & high end processors
- Design coding techniques for memory soft errors and evaluate their trade-offs
- Understand the benefits of chipkill ECC, sparing and scrubbing
- List the techniques used in the I/O sub-system
- Putting it together: Stratus Case Study

Summary

- Processor design must be self-checking
  - Error detection and recovery part of design
  - Duplication incurs up to 35 % area overheads (G5)
  - Commodity processors cannot afford such high costs
- Memory elements can be protected using ECC
  - ECC alone is not enough for chip failures -> chipkill
  - ECC has power, performance and area costs
- I/O systems need end-to-end consistency checks

Further Reading

- T. Slegel et al. , IBM's S/390 G5 Microprocessor Design. *IEEE Micro* 19, 2 (Mar. 1999), pp.12-23.
- Soft Errors in Electronic Memory, A white paper by Terrazon Semiconductors, 2004.
- Baumann, R.; , "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," *Electron Devices Meeting, 2002.*
- Timothy J. Dell, A White Paper on the Benefits of Chipkill-Correct ECC for PC Server Main Memory, (1997).
- Schroeder, B., Pinheiro, E., and Weber, W. 2009. DRAM errors in the wild: a large-scale field study. In *Proceedings of the Eleventh international Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '09.
- “The Stratus architecture”, Steve Webber & John Beirne, Symposium on Fault-Tolerant Computing, 1991 (FTCS-21).