20
Fault Tolerance & Memory Hierarchy

18-548/15-548 Memory System Architecture

Dan Siewiorek

November 23, 1998

Required Reading: Cragon pp. 278-283
Handout from Siewiorek & Swarz
Supplemental Reading: Hennessy & Patterson 6.5
Koopman & Siewiorek 5.7 (in library)
IBM Tech. Note on fault tolerance & DRAMs

Preview

- **Many terms have multiple usage that can lead to confusion when used out of context**
  - Sources of error
- **Faults go through at least ten stages from inception to repair - so designer better plan for all ten stages**
  - Relationship between sequence of events in handling a fault and mathematical measures
- **Coding can be considered selection of a subset of all the possible bit patterns to maximize the "distance" between code words**
  - Carefully position valid data representations in *n*-space so that bit changes do not lead to another valid data point (*i.e.* code word)
- **Error correcting codes designed to tolerate different fault types**
  - Random place/value, known place/random value, burst
- **Coding is an effective application of redundancy to processor, bus, and memory**
  - Examples: generic, Titan

DEFINITIONS & THE LIFE OF A FAULT

Definitions

- **RELIABILITY:**
  SURVIVAL PROBABILITY
  - When repair is costly or function is critical
- **AVAILABILITY:**
  THE FRACTION OF TIME A SYSTEM MEETS ITS SPECIFICATION
  - When service can be delayed or denied
- **REDUNDANCY:**
  EXTRA HARDWARE, SOFTWARE, TIME

Stages in System Development

STAGE	ERROR SOURCES	ERROR DETECTION
Specification & design	Algorithm Design Formal Specification	Simulation Consistency checks
Prototype	Algorithm design Wiring & assembly Timing Component Failure	Stimulus/response Testing
Manufacture	Wiring & assembly Component failure	System testing Diagnostics
Installation	Assembly Component failure	System Testing Diagnostics
Field Operation	Component failure Operator errors Environmental factors	Diagnostics

Manufacturing Process Flow

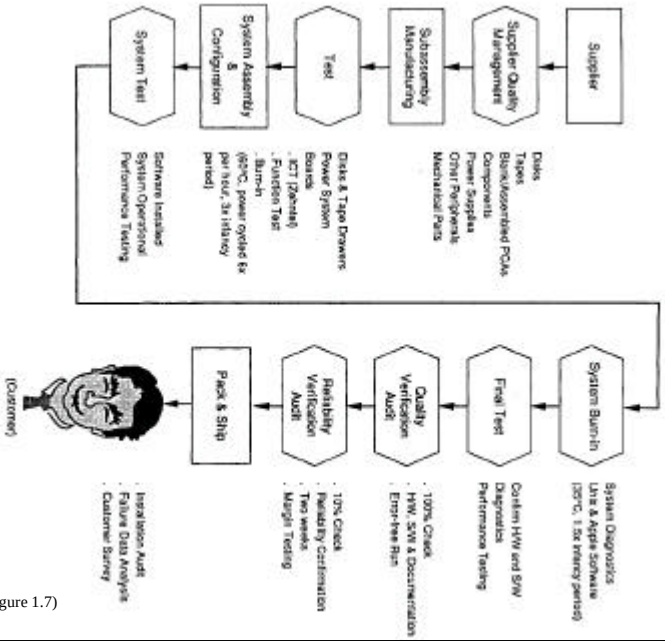
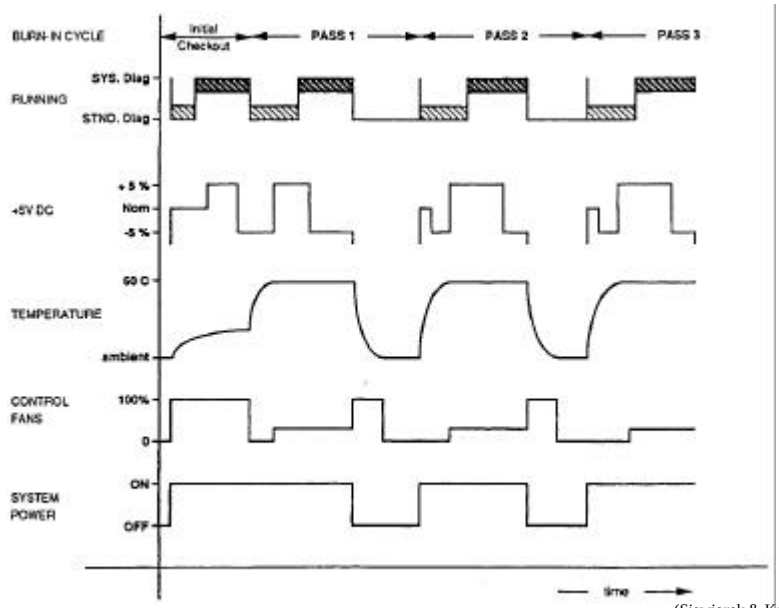


FIGURE 1.7. Manufacturing process flow.

(Siewiorek & Koopman Figure 1.7)

Typical Burn-in Profile



(Siewiorek & Koopman Figure 1.6)

FIGURE 1.6. Typical burn-in profile.

Cause-Effect Sequence and Duration

- **FAILURE:** **component does not provide service**
- **FAULT:** **deviation of logic function from design value**
  Hard, Transient
- **ERROR:** **manifestation of a fault by incorrect value**

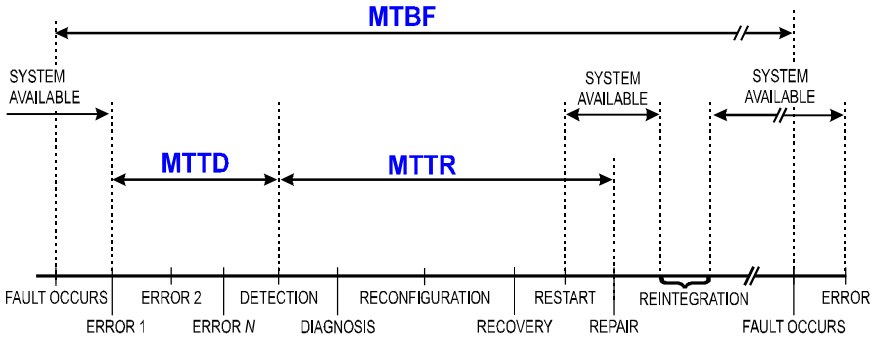
- **DURATION:**
  - Transient- design errors, environment
  - Intermittent- repair by replacement
  - Permanent- repair by replacement

Basic Steps in Fault Handling

- **Fault Confinement**
- **Fault Detection**
- **Fault Masking**
- **Retry**
- **Diagnosis**
- **Reconfiguration**
- **Recovery**
- **Restart**
- **Repair**
- **Reintegration**

MTBF -- MTTD -- MTTR

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$



A Scenario for on-line detection and off-line repair. The measures – MTBF, MTTD, and MTTR are the average times to failure, to detection, and to repair.

CMU Andrew File Server Study

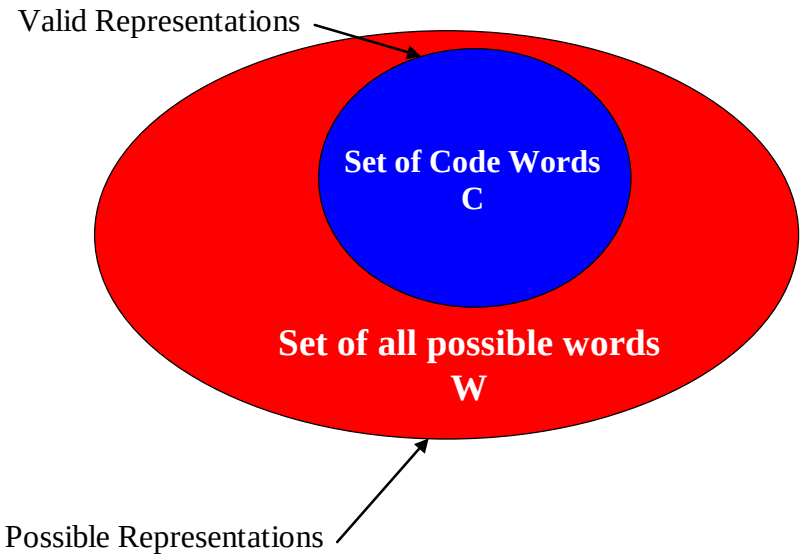
- **Configuration**
  - 13 SUN II Workstations with 68010 processor
  - 4 Fujitsu Eagle Disk Drives
- **Observations**
  - 21 Workstation Years
- **Frequency of events**
  - Permanent Failures 29
  - Intermittent Faults 610
  - Transient Faults 446
  - System Crashes 298
- **Mean Time To**
  - Permanent Failures 6552 hours
  - Intermittent Faults 58 hours
  - Transient Faults 354 hours
  - System Crash 689 hours

Some Interesting Numbers

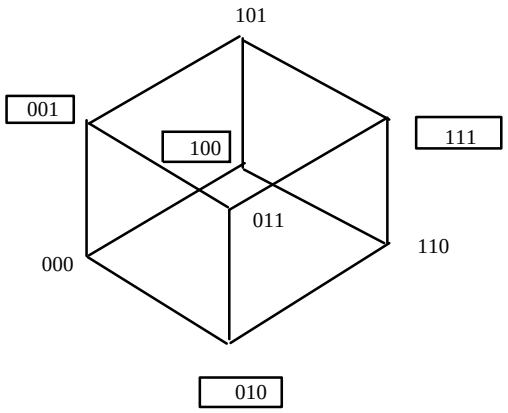
- **Permanent Outages/Total Crashes = 0.1**
- **Intermittent Faults/Permanent Failures = 21**
  - Thus first symptom appears over 1200 hours prior to repair
- **(Crashes - Permanent)/Total Faults = 0.255**
- **14/29 failures had three or fewer error log entries**
  - 8/29 had no error log entries

BASICS OF CODING THEORY

Code Space



Simple 3-bit Error Detecting Code Space



Boxed words = odd parity; Unboxed words = even parity

Error Code Definitions

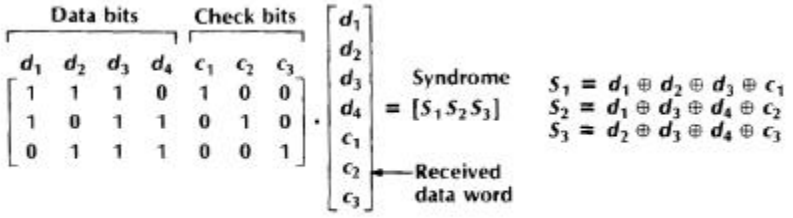
- **Systematic application of redundancy to information**
  - For all possible words, only a subset represent valid information - which is the set of code words
  - The remaining words are invalid - words in this set can only appear if an error has occurred
- **Hamming distance**
  - The number of bit positions in which two code words differ
- **Minimum distance, d, of a code**
  - Minimum Hamming distance between any two code word
- **Error detection code can detect p errors if the code distance d >= p+1**
- **Error correction code can correct t errors if d >= 2t + 1**
- **Distance-d code can correct up to t errors and detect an additional p errors if d >= 2t + p + 1**

Linear Error-Correcting Codes - Terminology

- **(n,k) Codes**
  - n = length of code word
  - k = length of actual data
  - n-k = c = redundancy (sometimes referred to as check bits)
- **Hamming Single Error Correction Codes (SEC)**
  - k data bits
  - c check bits
  - n = k + c = code bits
  - 2**c > = c + k + 1
  - Separable Code
    - parity check matrix with one bit per column for check bits
    - all columns are unique
    - no all zero column

ERROR CORRECTING CODES

Parity Check Matrix for (7,4) Hamming SEC



a. Parity-check matrix and syndrome formation for a (7,4) Hamming SEC code

(Siewiorek & Swarz)

Example Received Code Words and Syndromes

	Data bits				Check bits			Syndrome			
Code word (no error)	1	1	1	0	1	0	0	0	0	0	(Zero syndrome implies no error)
One error (box)	1	1	1	[1]	1	0	0	0	1	1	(Matches d_4 column)
Two errors (boxes)	1	[0]	1	0	1	[1]	0	1	1	1	(Matches d_3 column—results in erroneous correction)

b. Received code words and their syndromes for 0, 1, and 2 errors

(Siewiorek & Swarz)

(7,4) Hamming SEC with binary coded syndromes

(Siewiorek & Swarz)

c_1	c_2	d_1	c_3	d_2	d_3	d_4
1	0	1	0	1	0	1
0	1	1	0	0	1	1
0	0	0	1	1	1	1

c. Parity-check matrix for (7,4) Hamming code for which syndrome is the binary-coded position of the bit in error

Hamming SEC/DED Code

- **Add an extra row/column to parity check matrix**
  - column with a single 1
  - row with all 1's (overall parity)
- **Non Column matching syndrome indicates a multiple error**
  - sum of columns in error can not equal any other column

Hamming SEC/DED Parity Check Matrix

$$\begin{array}{c} \begin{matrix} d_1 & d_2 & d_3 & d_4 & c_1 & c_2 & c_3 & c_4 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \cdot \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = [S_1 S_2 S_3 S_4]$$

a. Parity-check matrix for (8,4) Hamming SEC/DED code

(Siewiorek & Swarz)

Example SEC/DED Received Words &Syndromes

Number of errors	Received data bits				Received check bits				Syndrome			
	d_1	d_2	d_3	d_4	c_1	c_2	c_3	c_4	S_1	S_2	S_3	S_4
Zero	1	1	1	0	0	1	0	0	0	0	0	0
One	1	[0]	1	0	0	1	0	0	1	1	0	1
Two	1	[0]	1	0	0	1	[1]	0	0	1	1	1

b. Received words and their syndromes

(Siewiorek & Swarz)

Erasure Codes

- **Prior codes have assumed random value and random position for errors**
- **Erasure codes have random value, but known position due to noise, prior history**
- **A code with Hamming distance d can correct d - 2 erasure errors**

Double-Complement Algorithm for Erasures

The following example of the double-complement algorithm shows an 8-bit word

Original word	*1 1 0 0 1 1 0 0*	
Hard and soft faults	*0 S*	
Read R	***0 1 0 0 1 1 1 0***	*Double error*
Write $\overline{R}$	1 0 1 1 0 0 0 1	
Read W	0 0 1 1 0 0 0 1	
Form $\overline{W}$	*1 1 0 0 1 1 1 0*	Single error
Hard erasure: $R \oplus \overline{W}$	*1* 0 0 0 0 0 0 0	
Soft error	0 0 0 0 0 0 *1* 0	

(Siewiorek & Swarz)

Truncated Codes

- **Data requires k bits and total number of columns in check matrix is k+c+1. But c check bits have 2**c patterns**
  - So fraction of columns in check matrix to all possible columns is (k+c+1)/2**c
  - For k = 64, c = 7; only 72/128 of the possible columns are used
- **By selecting the values for the columns of the parity check matrix appropriately, new detection properties are provided by the code**
  - For example, columns could be selected so that the sum (XOR) of any pair does not represent a column in the parity check matrix. This code could then detect physically adjacent errors.

SYSTEM-LEVEL ERROR TECHNIQUES

Error Detection Techniques in Generic System

- **Memory**
  - Double-error-detection code on memory data
  - Parity on address and control information
- **Cache**
  - Parity on data, address, control information
- **Translation Buffer**
  - Parity on tag, data, valid bits
- **Input/Output**
  - Parity on data and control
- **CPU**
  - Parity on data paths
  - Parity on registers
  - Output parity of ALUs, shifters
  - Parity on control store
  - Duplication and comparison of control logic

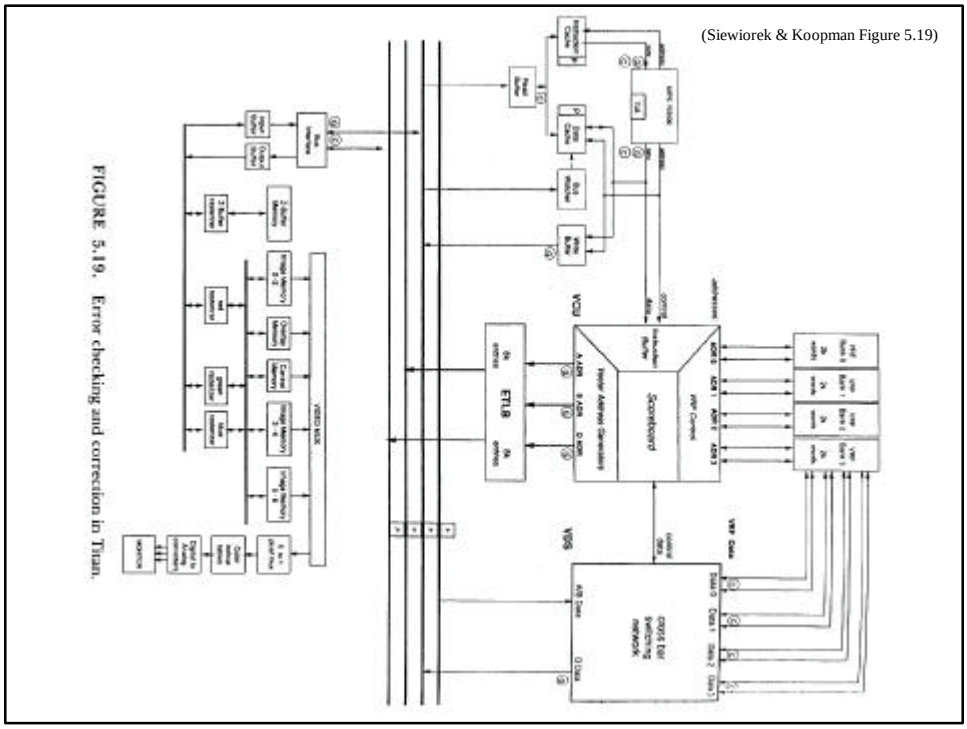
Error Recovery Techniques in Generic System

- **Memory**
  - Single-error-correction code on data
  - Retry on address or control information parity error
- **Cache**
  - Retry on address or control information parity error
  - Disable portions of cache on data parity errors
- **Translation buffer**
  - Refill on error
- **Input/Output**
  - Retry on data or control parity errors
- **CPU**
  - Retry on control store parity error
  - Register file copies for performance
  - Invert sense of control store
  - Macroinstruction retry

Titan Error Detection/Correction Techniques

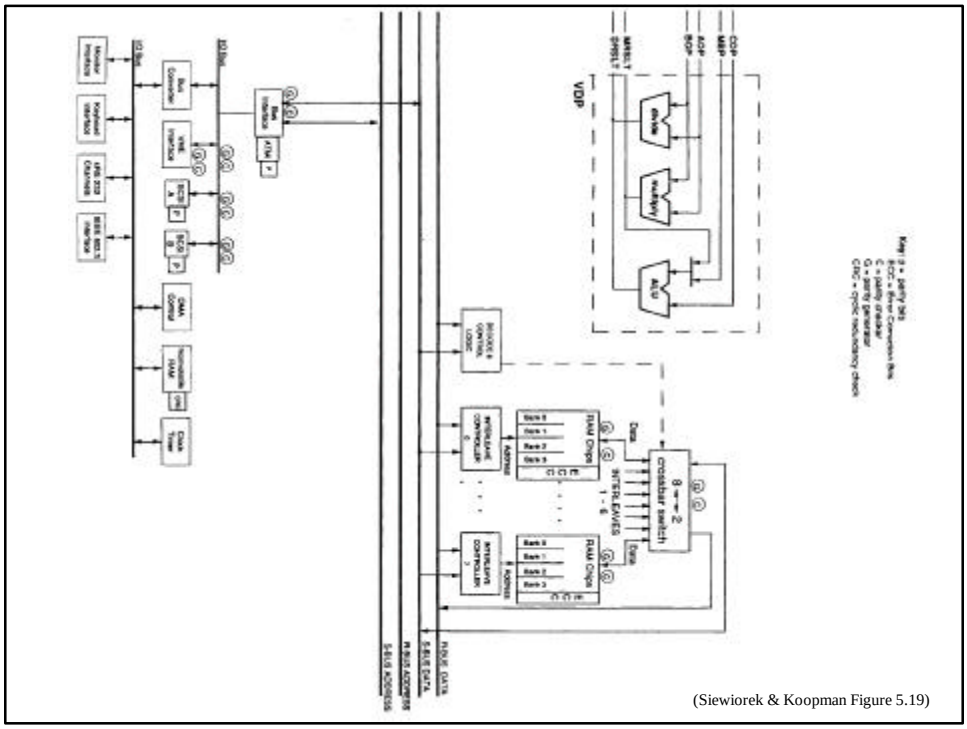
- **Memory - composed of 4-bit wide memory chips**
  - Single Error Correction, Multiple Error Detection
    - Detection of 2-bit errors
    - Detection of 3- and 4-bit errors in the same chip
  - Memory scrubbing
- **Cache**
  - Parity on I, D cache
- **Memory Bus**
  - Single parity on each control, request number, return ID
  - Byte parity on address, data
  - Detection of non-existent address, bus time out, invalid transfer type, address alignment error
  - Sender at time of error recorded
- **I/O Bus**
  - Parity on I/O Address Translation Map
  - Cyclic Redundancy Check on Nonvolatile Memory

TITAN
ERROR TECHNIQUES



(Siewiorek & Koopman Figure 5.19)

FIGURE 5.19. Error checking and correction in Titan.



(Siewiorek & Koopman Figure 5.19)

REVIEW

Review

- **Many terms have multiple usage that can lead to confusion when used out of context**
  - Establish context of all parties in a design discussion
- **Faults go through at least ten stages from inception to repair - so designer better plan for all ten stages**
  - Faults will go through all ten stages whether designer plans for them or not
- **Coding can be considered selection of a subset of all the possible bit patterns to maximize the "distance" between code words**
  - Important classes of codes are linear (can be decoded with XOR trees) and separable (decoding can go on in parallel with data processing minimizing performance degradation)
- **Error correcting codes designed to tolerate different fault types**
  - Can customize codes to the physical partitioning of the design
- **Coding is an effective application of redundancy to processor, bus, and memory**
  - Redundancy only log to the base 2 of the number of data bits