Review: 1-T Memory Cell (DRAM)

- **Write:**
  1. Drive bit line
  2. Select row
- **Read:**
  1. Precharge bit line to \(\frac{Vdd}{2}\)
  2. Select row
  3. Cell and bit line share charges
     - Very small voltage changes on the bit line
  4. Sense (fancy sense amp)
     - Can detect changes of \(-1\) million electrons
  5. Write: restore the value
- **Refresh**
  1. Just do a dummy read to every cell.

DRAM Capacitors: more capacitance in a small area

- **Trench capacitors:**
  - Logic ABOVE capacitor
  - Gain in surface area of capacitor
  - Better Scaling properties
  - Better Planarization

- **Stacked capacitors**
  - Logic BELOW capacitor
  - Gain in surface area of capacitor
  - 2-dim cross-section quite small

4 Key DRAM Timing Parameters

- \(t_{RAC}\): minimum time from \(RAS_L\) line falling to the valid data output.
  - Quoted as the speed of a DRAM when buy
  - A typical 4Mb DRAM \(t_{RAC}\) \(= 60\) ns
  - Speed of DRAM since on purchase sheet?
- \(t_{RC}\): minimum time from the start of one row access to the start of the next.
  - \(t_{RC} = 110\) ns for a 4Mbit DRAM with a \(t_{RAC}\) of 60 ns
- \(t_{CAC}\): minimum time from \(CAS_L\) line falling to valid data output.
  - 15 ns for a 4Mbit DRAM with a \(t_{RAC}\) of 60 ns
- \(t_{PC}\): minimum time from the start of one column access to the start of the next.
  - 35 ns for a 4Mbit DRAM with a \(t_{RAC}\) of 60 ns
Main Memory Performance

• DRAM (Read/Write) Cycle Time >> DRAM (Read/Write) Access Time
  – 2:1; why?
• DRAM (Read/Write) Cycle Time:
  – How frequent can you initiate an access?
  – Analogy: A little kid can only ask his father for money on Saturday
• DRAM (Read/Write) Access Time:
  – How quickly will you get what you want once you initiate an access?
  – Analogy: As soon as he asks, his father will give him the money
• DRAM Bandwidth Limitation analogy:
  – What happens if he runs out of money on Wednesday?

Increasing Bandwidth - Interleaving

Access Pattern without Interleaving:

Access Pattern with 4-way Interleaving:

Main Memory Performance

• Timing model
  – 1 to send address,
  – 4 for access time, 10 cycle time, 1 to send data
  – Cache Block is 4 words
• Simple M.P. = 4 x (1+10+1) = 48
• Wide M.P. = 1 + 10 + 1 = 12
• Interleaved M.P. = 1+10+1 + 3 = 15

Avoiding Bank Conflicts

• Lots of banks
  int x[256][512];
  for (j = 0; j < 512; j = j+1)
    for (i = 0; i < 256; i = i+1)
      x[i][j] = 2 * x[i][j];
• Even with 128 banks, since 512 is multiple of 128, conflict on word accesses
• SW: loop interchange or declaring array not power of 2 ("array padding")
• HW: Prime number of banks
  – bank number = address mod number of banks
  – bank number = address mod number of banks
  – address within bank = address / number of words in bank
  – modulo & divide per memory access with prime no. banks?

Finding Bank Number and Address within a bank

Problem: We want to determine the number of banks, N_b, to use and the number of words to store in each bank, W_w, such that:
• given a word address x, it is easy to find the bank where x will be found, B(x), and the address of x within the bank, A(x).
• for any address x, B(x) and A(x) are unique.
• the number of bank conflicts is minimized
Finding Bank Number and Address within a bank

Solution: We will use the following relation to determine the bank number for \( x \), \( B(x) \), and the address of \( x \) within the bank, \( A(x) \):

\[
B(x) = x \mod N_b
\]

\[
A(x) = x \mod W_b
\]

and we will choose \( N_b \) and \( W_b \) to be co-prime, i.e., there is no prime number that is a factor of \( N_b \) and \( W_b \) (this condition is satisfied if we choose \( N_b \) to be a prime number that is equal to an integer power of two minus 1).

We can then use the Chinese Remainder Theorem to show that \( B(x) \) and \( A(x) \) is always unique.

Fast Bank Number

• Chinese Remainder Theorem
As long as two sets of integers \( a_i \) and \( b_i \) follow these rules and \( a_i \) and \( a_j \) are co-prime if \( i \neq j \), then the integer \( x \) has only one solution (unambiguous mapping):

- bank number = \( b_0 \), number of banks = \( a_0 \)
- address within bank = \( b_1 \), number of words in bank = \( a_1 \)
- \( N \) word address 0 to \( N-1 \), prime no. banks, words power of 2

3 banks \( N_b = 3 \), and 8 words per bank, \( W_b = 8 \).

Fast Memory Systems: DRAM specific

• Multiple CAS accesses; several names (page mode)
  - Extended Data Out (EDO): 30% faster in page mode

• New DRAMs to address gap; what will they cost, will they survive?
  - RAMBUS: startup company; reinvent DRAM interface
    » Each Chip a module vs. slice of memory
    » Short bus between CPU and chips
    » Does own refresh
    » Variable amount of data returned
    » 1 byte / 2 ns (500 MB/s per chip)
  - Synchronous DRAM: 2 banks on chip, a clock signal to DRAM, transfer synchronous to system clock (66 - 100 MHz)
  - Intel claims RAMBUS Direct (16 b wide) is future PC memory
  - Niche memory or main memory?
    - e.g., Video RAM for frame buffers, DRAM + fast serial output

Fast Page Mode Operation

• Regular DRAM Organization:
  - \( N \) rows \( \times \) \( N \) columns \( \times \) \( M \) bits
  - Read & Write \( M \) bit at a time
  - Each \( M \) bit access requires a RAS / CAS cycle

• Fast Page Mode DRAM
  - \( N \times M \) "SRAM" to save a row
  - After a row is read into the register
    - Only CAS is needed to access other \( M \) bit blocks on that row
    - RAS_L remains asserted while CAS_L is toggled

Fast Page Mode Operation Diagram

<table>
<thead>
<tr>
<th>Bank Number</th>
<th>Seq. Interleaved</th>
<th>Module Interleaved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>22</td>
</tr>
</tbody>
</table>

DRAM History

• DRAMs: capacity +60%/yr, cost –30%/yr
  - 2X cells/area, 1.5X die size in 3 years
• ’98 DRAM fab line costs $2B
  - DRAM only: density, leakage v. speed
• Rely on increasing no. of computers & memory per computer (60% market)
  - SIMM or DIMM is replaceable unit
  - computers use any generation DRAM
• Commodity, second source industry
  - high volume, low profit, conservative
  - little organization innovation in 20 years
• Order of importance: 1) Cost/bit 2) Capacity
  - First RAMBUS: 10X BW, +30% cost => little impact

DRAM Future: 1 Gbit+ DRAM

<table>
<thead>
<tr>
<th>Company</th>
<th>Mitsubishi</th>
<th>Samsung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocks</td>
<td>512 x 2 Mbit</td>
<td>1024 x 1 Mbit</td>
</tr>
<tr>
<td>Clock</td>
<td>200 MHz</td>
<td>250 MHz</td>
</tr>
<tr>
<td>Data Pins</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>Die Size</td>
<td>24 x 24 mm</td>
<td>31 x 21 mm</td>
</tr>
<tr>
<td>Metal Layers</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Technology</td>
<td>0.15 micron</td>
<td>0.16 micron</td>
</tr>
</tbody>
</table>
**DRAMs per PC over Time**

<table>
<thead>
<tr>
<th>DRAM Generation</th>
<th>'86</th>
<th>'89</th>
<th>'92</th>
<th>'96</th>
<th>'99</th>
<th>'02</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Mb</td>
<td>4 Mb</td>
<td>16 Mb</td>
<td>64 Mb</td>
<td>256 Mb</td>
<td>1 Gb</td>
<td></td>
</tr>
<tr>
<td>Minimum Memory Size</td>
<td>4 MB</td>
<td>8 MB</td>
<td>16 MB</td>
<td>32 MB</td>
<td>64 MB</td>
<td>128 MB</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Potential DRAM Crossroads?**

- After 20 years of 4X every 3 years, running into wall? (64Mb - 1 Gb)
- How can keep $1B fab lines full if buy fewer DRAMs per computer?
- Cost/bit –30%/yr if stop 4X/3 yr?
- What will happen to $40B/yr DRAM industry?

**Something new: Structure of Tunneling Magnetic Junction**

- **Tunneling Magnetic Junction RAM (TMJ-RAM)**
  - Speed of SRAM, density of DRAM, non-volatile (no refresh)
  - “Spintronics”: combination quantum spin and electronics
  - Same technology used in high-density disk-drives

**MEMS-based Storage**

- Magnetic “sled” floats on array of read/write heads
  - Approx 250 Gbit/in²
  - Data rates: IBM: 250 MB/s w 1000 heads
  - CMU: 3.1 MB/s w 400 heads
- Electrostatic actuators move media around to align it with heads
  - Sweep sled ±50µm in < 0.5µs
- Capacity estimated to be 1-10GB in 10cm²

See Ganger et al: http://www.lcs.ece.cmu.edu/research/MEMS

**Motivation:**
- DRAM is dense ⇒ Signals are easily disturbed
- High Capacity ⇒ Higher probability of failure

**Approach: Redundancy**
- Add extra information so that we can recover from errors
- Can we do better than just create complete copies?

**Block Codes: Data Coded in blocks**
- k data bits coded into n encoded bits
- Measure of overhead: Rate of Code: K/N
- Often called an (n,k) code
- Consider data as vectors in GF(2) [i.e. vectors of bits]

**General Idea: Code Vector Space**

- Not every vector in the code space is valid
- Hamming Distance (d): Minimum number of bit flips to turn one code word into another
- Number of errors that we can detect: (d-1)
- Number of errors that we can fix: ½(d-1)
Error Correction Codes (ECC)

- Memory systems generate errors (accidentally flipped-bits)
  - DRAMs store very little charge per bit
  - "Soft" errors occur occasionally when cells are struck by alpha particles or other environmental upsets.
  - Less frequently, "hard" errors can occur when chips permanently fail.
  - Problem gets worse as memories get denser and larger

- Where is "perfect" memory required?
  - servers, spacecraft/military computers, ebay, …

- Memories are protected against failures with ECCs

- Extra bits are added to each data-word
  - used to detect and/or correct faults in the memory system
  - in general, each possible data word value is mapped to a unique "code word". A fault changes a valid code word to an invalid one - which can be detected.

Correcting Code Concept

- Detection: bit pattern fails codeword check
- Correction: map to nearest valid code word

Simple Error Detection Coding

- Each data value, before it is written to memory is "tagged" with an extra bit to force the stored word to have even parity:
  - Each word, as it is read from memory is "checked" by finding its parity (including the parity bit).
  - A non-zero parity indicates an error occurred:
    - two errors (on different bits) is not detected (nor any even number of errors)
    - odd numbers of errors are detected.
  - What is the probability of multiple simultaneous errors?

Hamming Error Correcting Code

- Use more parity bits to pinpoint bit(s) in error, so they can be corrected.

- Example: Single error correction (SEC) on 4-bit data
  - use 3 parity bits, with 4-data bits results in 7-bit code word
  - 3 parity bits sufficient to identify any one of 7 code word bits
  - overlap the assignment of parity bits so that a single error in the 7-bit work can be corrected

- Procedure: group parity bits so they correspond to subsets of the 7 bits:
  - p1 protects bits 1,3,5,7
  - p2 protects bits 2,3,6,7
  - p3 protects bits 4,5,6,7

\[\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
p1 & p2 & d1 & p3 & d2 & d3 & d4 \\
\end{array}\]

- Bit position number
- Note: number bits from left to right.

- On writing to memory:
  - parity bits are assigned to force even parity over their respective groups.
- On reading from memory:
  - check bits \((c_{i},c_{j},c_{k})\) are generated by finding the parity of the group and its parity bit.
  - If an error occurred in a group, the corresponding check bit will be 1, if no error the check bit will be 0.
  - check bits \((c_{i},c_{j},c_{k})\) form the position of the bit in error.

Hamming Code Example

- Example: \(c = c_{0}c_{1}c_{2}c_{3} = 101\)
  - error in 4,5,6, or 7 (by \(c_{3}\))
  - error in 1,3,5, or 7 (by \(c_{1}\))
  - no error in 1, 3, 6, or 7 (by \(c_{2}\))
- Therefore error must be in bit 5.
- Note the check bits point to 5
- By our clever positioning and assignment of parity bits, the check bits always address the position of the error!
- \(c=000\) indicates no error
- eight possibilities

Interactive Quiz

- You receive:
  - 111110
  - 000010
  - 1010010
- What is the correct value?
Review: Hamming Error Correcting Code

- Overhead involved in single error correction code:
  - let \( p \) be the total number of parity bits and \( d \) the number of data bits in a \( p + d \) bit word.
  - If \( p \) error correction bits are to point to the error bit (\( p + d \) cases) plus indicate that no error exists (1 case), we need:
    \[ 2p \geq p + d + 1 \]
    
    \[ p \geq \log_2(p + d + 1) \]

  - Typical modern codes in DRAM memory systems:
    - 64-bit data blocks (8 bytes) with 72-bit code words (9 bytes).

- Adding on extra parity bit covering the entire word can provide double error detection:
  - 8 data \( \Rightarrow \) 4 parity
  - 16 data \( \Rightarrow \) 5 parity
  - 32 data \( \Rightarrow \) 6 parity
  - 64 data \( \Rightarrow \) 7 parity

- On reading the C bits are computed (as usual) plus the parity over the entire word, P:
  - \( C = 0 \), \( P = 0 \), no error
  - \( C \neq 0 \), \( P = 1 \), correctable single error
  - \( C \neq 0 \), \( P = 0 \), a double error occurred
  - \( C = 0 \), \( P = 1 \), an error occurred in \( p_4 \) bit

Review: Code Types

- Linear Codes: \( C = G \cdot D \cdot S = H \cdot C \)
  - Code is generated by \( G \) and in null-space of \( H \)

- Hamming Codes: Design the \( H \) matrix
  - \( d = 3 \Rightarrow \) Columns nonzero, Distinct
  - \( d = 4 \Rightarrow \) Columns nonzero, Distinct, Odd-weight

- Reed-solomon codes:
  - Based on polynomials in GF(2^k) (i.e. k-bit symbols)
  - Data as coefficients, code space as values of polynomial:
    \[ P(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1} \]
    \[ Coded: P(0), P(1), P(2), \ldots, P(n-1) \]
  - Can recover polynomial as long as get any \( k \) of \( n \)
  - Alternatively: as long as no more than \( n-k \) coded symbols erased, can recover data.

  - Side note: Multiplication by constant in GF(2^k) can be represented by \( k \times k \) matrix: \( a \times \)
  - Decompose unknown vector into k bits: \( x = x_0 + 2x_1 + \ldots + 2^{k-1}x_{k-1} \)
  - Each column is result of multiplying \( a \) by \( 2^i \)

Motivation: Who Cares About I/O?

- CPU Performance: 60% per year
- I/O system performance limited by mechanical delays (disk I/O)
  - \( < 10 \% \) per year (I/O per sec or MB per sec)
- Amdahl’s Law: system speed-up limited by the slowest part!
  - 10% IO & 10x CPU \( \Rightarrow \) 5x Performance (lose 50%)
  - 10% IO & 100x CPU \( \Rightarrow \) 10x Performance (lose 90%)
- I/O bottleneck:
  - Diminishing fraction of time in CPU
  - Diminishing value of faster CPUs

I/O Systems

Technology Trends

- Today: Processing Power Doubles Every 18 months
- Today: Memory Size Doubles Every 18 months
- Today: Disk Capacity Doubles Every 18 months
- Disk Positioning Rate (Seek + Rotate) Doubles Every Ten Years!

Storage Technology Drivers

- Driven by the prevailing computing paradigm
  - 1950s: migration from batch to on-line processing
  - 1990s: migration to ubiquitous computing
    - computers in phones, books, cars, video cameras, ...
    - nationwide fiber optical network with wireless tails
- Effects on storage industry:
  - Embedded storage
    - smaller, cheaper, more reliable, lower power
  - Data utilities
    - high capacity, hierarchically managed storage
Historical Perspective

- 1956 IBM Ramac — early 1970s Winchester
  - Developed for mainframe computers, proprietary interfaces
  - Steady shrink in form factor: 27 in. to 14 in.
- 1970s developments
  - 5.25 inch floppy disk form factor (microcode into mainframe)
  - Early emergence of industry standard disk interfaces
    - ST506, SASI, SMD, ESDI
- Early 1980s
  - PCs and first generation workstations
- Mid 1980s
  - Client/server computing
  - Centralized storage on file server
    - Accelerates disk downsizing: 8 inch to 5.25 inch
  - Mass market disk drives become a reality
    - Industry standards: SCSI, IPI, IDE
    - 5.25 inch drives for standalone PCs, End of proprietary interfaces

Disk History

<table>
<thead>
<tr>
<th>Year</th>
<th>Data Density (Mbit/sq. in)</th>
<th>Capacity (MBytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1973</td>
<td>1.7</td>
<td>140</td>
</tr>
<tr>
<td>1979</td>
<td>7.7</td>
<td>2,300</td>
</tr>
</tbody>
</table>

1997:
- 3090 Mbit/sq. in
- 8100 MBytes

"Makers of disk drives crowd even more data into even smaller spaces"

Disk Performance Model/Trends

- Capacity
  - +100%/year (2X / 1.0 yrs)
- Transfer rate (BW)
  - +40%/year (2X / 2.0 yrs)
- Rotation + Seek time
  - 8% / year (1/2 in 10 yrs)
- MB/$
  - >100%/year (2X / <1.5 yrs)
  - Fewer chips + areal density

"Makers of disk drives crowd even more data into even smaller spaces"
**Disk Device Terminology**

- Several **platters**, with information recorded magnetically on both **surfaces** (usually)
- Bits recorded in **tracks**, which in turn divided into **sectors** (e.g., 512 Bytes)
- **Actuator** moves **head** (end of arm/surface) over track (“seek”), select **surface**, wait for sector rotate under head, then read or write
  - “**Cylinder**”: all tracks under heads

**Disk Performance Example**

**Disk Latency** = Queuing Time + Seek Time + Rotation Time + Xfer Time + Ctrl Time

Order of magnitude times for 4K byte transfers:
- Seek: 12 ms or less
- Rotate: 4.2 ms @ 7200 rpm = 0.5 rev/(7200 rpm/60m/s) (8.3 ms @ 3600 rpm)
- Xfer: 1 ms @ 7200 rpm (2 ms @ 3600 rpm)
- Ctrl: 2 ms (big variation)

Disk Latency = Queuing Time + \((12 + 4.2 + 1 + 2)\text{ms} = QT + 19.2\text{ms})

Average Service Time = 19.2 ms

**Disk Time Example**

- **Disk Parameters**:
  - Transfer size is 8K bytes
  - Advertised average seek is 12 ms
  - Disk spins at 7200 RPM
  - Transfer rate is 4 MB/sec
- **Controller overhead** is 2 ms
- Assume that disk is idle so no queuing delay
- **What is Average Disk Access Time for a Sector?**
  - Ave seek + ave rot delay + transfer time + controller overhead
  - \(12 + 4.15 + 2 + 2 = 20\text{ ms}\)
- Advertised seek time assumes no locality: typically 1/4 to 1/3 advertised seek time: 20 ms \(\Rightarrow\) 12 ms

**Snapshot: Ultrastar 72ZX**

- 73.4 GB, 3.5 inch disk
- 26/MB
- 10,000 RPM
- 3 ms = 1/2 rotation
- 11 platters, 22 surfaces
- 15,110 cylinders
- 7 Gbit/sq. in. areal den
- 17 watts (idle)
- 0.1 ms controller time
- 5.3 ms avg. seek
- 50 to 29 MB/s(internal)

source: www.ibm.com; www.pricewatch.com; 2/14/00
What Kind of Errors

• In Memory
• In Disks?
• In networks?
• On Tapes?
• In distributed storage systems?

Concept: Redundant Check

• Send a message M and a “check” word C
• Simple function on <M,C> to determine if both received correctly (with high probability)
• Example: XOR all the bytes in M and append the “checksum” byte, C, at the end
  – Receiver XORs <M,C>
  – What should result be?
  – What errors are caught?

Example: TCP Checksum

• TCP Checksum a 16-bit checksum, consisting of the one's complement of the one's complement sum of the contents of the TCP segment header and data, is computed by a sender, and included in a segment transmission. (note end-around carry)
  – Summing all the words, including the checksum word, should yield zero

Example: Ethernet CRC-32

CRC concept

• I have a msg polynomial M(x) of degree m
• We both have a generator poly G(x) of degree m
• Let r(x) = remainder of M(x)x^n / G(x)
  – M(x)x^n = G(x)p(x) + r(x)
  – r(x) is of degree n
• What is (M(x)x^n – r(x)) / G(x) ?

Galoi Fields - the theory behind LFSRs

• LFSR circuits performs multiplication on a field.
• A field is defined as a set with the following:
  – two operations defined on it:
    – “addition” and “multiplication”
  – closed under these operations
  – associative and distributive laws hold
  – additive and multiplicative identity elements
  – additive inverse for every element
  – multiplicative inverse for every non-zero element
• Example fields:
  – set of rational numbers
  – set of real numbers
  – set of integers is not a field (why?)
• Finite fields are called Galois fields.
• Example:
  – Binary numbers 0,1 with XOR as “addition” and AND as “multiplication”.
  – Called GF(2).
  – 0+1 = 1
  – 1+1 = 0
  – 0+1 = ?
  – 1+1 = ?
Galois Fields - The theory behind LFSRs

- Consider polynomials whose coefficients come from GF(2).
- Each term of the form $x^r$ is either present or absent.
- Examples: 0, 1, $x$, $x^2$, and $x^2 + x + 1$.
- With addition and multiplication these form a field:
  - “Add”: XOR each element individually with no carry.
  - “Multiply”: multiplying by $x^r$ is like shifting to the left.

Polynomial division

- When MSB is zero, just shift left, bringing in next bit.
- When MSB is 1, XOR with divisor and shift.

CRC encoding

- These polynomials form a Galois (finite) field if we take the results of this multiplication modulo a prime polynomial $p(x)$.
  - A prime polynomial is one that cannot be written as the product of two non-trivial polynomials $q(x)r(x)$.
  - Perform modulo operation by subtracting a (polynomial) multiple of $p(x)$ from the result. If the multiple is 1, this corresponds to XOR-ing the result with $p(x)$.
- For any degree, there exists at least one prime polynomial.
- With it we can form $GF(2^n)$.

Galois Fields - The theory behind LFSRs

- Additionally, every Galois field has a primitive element, $\alpha$, such that all non-zero elements of the field can be expressed as a power of $\alpha$. By raising $\alpha$ to powers (modulo $p(x)$), all non-zero field elements can be formed.
- Certain choices of $p(x)$ make the simple polynomial $x$ the primitive element. These polynomials are called primitive, and one exists for every degree.
- For example, $x^4 + x + 1$ is primitive. So $\alpha = x$ is a primitive element and successive powers of $\alpha$ will generate all non-zero elements of $GF(16)$. Example on next slide.
Galois Fields – Primitives

\[
\begin{align*}
\alpha_0 &= 1 \\
\alpha_1 &= x \\
\alpha_2 &= x^2 \\
\alpha_3 &= x^3 \\
\alpha_4 &= x^4 + 1 \\
\alpha_5 &= x^5 + x \\
\alpha_6 &= x^6 + x^2 \\
\alpha_7 &= x^3 + x + 1 \\
\alpha_8 &= x^8 + 1 \\
\alpha_9 &= x^9 + x \\
\alpha_{10} &= x^2 + x + 1 \\
\alpha_{11} &= x^3 + x^2 + x \\
\alpha_{12} &= x^3 + x^2 + x + 1 \\
\alpha_{13} &= x^3 + x^2 + 1 \\
\alpha_{14} &= x^3 + 1 \\
\alpha_{15} &= 1
\end{align*}
\]

• Note this pattern of coefficients matches the bits from our 4-bit LFSR example.
• In general finding primitive polynomials is difficult. Most people just look them up in a table, such as:

\[x^4 \mod x^4 + x + 1 = x^4 \oplus (x^4 + x + 1) = x + 1\]

Building an LFSR from a Primitive Poly

- For \(k\)-bit LFSR number the flip-flops with FF1 on the right.
- The feedback path comes from the Q output of the leftmost FF.
- Find the primitive polynomial of the form \(x^k + \ldots + 1\).
- Each term of the form \(x^n\) corresponds to connecting an xor between FF \(n\) and \(n+1\).
- To build an 8-bit LFSR, use the primitive polynomial \(x^8 + x^4 + x^3 + x^2 + 1\) and connect xors between FF2 and FF3, FF3 and FF4, and FF4 and FF5.

Alternative Data Storage Technologies: Early 1990s

<table>
<thead>
<tr>
<th>Technology</th>
<th>Cap (MB)</th>
<th>BPI</th>
<th>TPI</th>
<th>BPI*TPI</th>
<th>Data Xfer Access</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional Tape:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cartridge (.25&quot;)</td>
<td>150</td>
<td>12000</td>
<td>104</td>
<td>23240</td>
<td>1.2</td>
<td>92</td>
</tr>
<tr>
<td>IBM 3490 (.5&quot;)</td>
<td>800</td>
<td>22860</td>
<td>38</td>
<td>102240</td>
<td>0.9</td>
<td>3000</td>
</tr>
<tr>
<td>Helical Scan Tape:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video (8mm)</td>
<td>4600</td>
<td>43200</td>
<td>1638</td>
<td>7447200</td>
<td>71</td>
<td>492</td>
</tr>
<tr>
<td>DAT (4mm)</td>
<td>1300</td>
<td>61000</td>
<td>1870</td>
<td>2377000</td>
<td>114</td>
<td>183</td>
</tr>
<tr>
<td>Tape vs. Disk:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Magnetic &amp; Optical Disk:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hard Disk (5.25&quot;)</td>
<td>1200</td>
<td>33528</td>
<td>1880</td>
<td>6335520</td>
<td>63</td>
<td>3000</td>
</tr>
<tr>
<td>Sony MO (5.25&quot;)</td>
<td>640</td>
<td>24130</td>
<td>18796</td>
<td>45422400</td>
<td>62</td>
<td>4250</td>
</tr>
<tr>
<td>Tape</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Primitive Polynomials

\[
\begin{align*}
x^2 + x + 1, & \quad x^4 + x^3 + x + 1, \quad x^8 + x + 1 \\
x^3 + x + 1, & \quad x^5 + x^4 + x + 1, \quad x^6 + x + 1 \\
x^4 + x^3 + x + 1, & \quad x^7 + x^6 + x + 1, \quad x^9 + x^2 + 1 \\
x^5 + x^4 + 1, & \quad x^7 + x^6 + x^3 + x + 1, \quad x^{10} + x^7 + x^5 + x^3 + \ldots + 1 \\
x^6 + x^5 + 1, & \quad x^8 + x^7 + x^5 + x^4 + x^3 + \ldots + 1, \quad x^{12} + x^8 + x^7 + x^6 + \ldots + 1 \\
x^7 + x^6 + 1, & \quad x^9 + x^8 + x^7 + x^6 + \ldots + 1
\end{align*}
\]

• CRC-16: \(G(x) = x^{16} + x^{15} + x^2 + 1\)
  - detects single and double bit errors
  - All errors with an odd number of bits
  - Burst errors of length 16 or less
  - Most errors for longer bursts

• CRC-32: \(G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^3 + x^2 + x + 1\)
  - Used in ethernet
  - Also 32 bits of 1 added on front of the message

Generating Polynomials

- Initialize the LFSR to all 1s

Tape vs. Disk

- Longitudinal tape uses same technology as hard disk; tracks its density improvements
- Disk head flies above surface, tape head lies on surface
- Disk fixed, tape removable
- Inherent cost-performance based on geometries: fixed rotating platters with gaps
  (random access, limited area, 1 media / reader)
- Helical Scan (VCR, Camcoder, DAT)
  removable long strips wound on spool (sequential access, “unlimited” length, multiple / reader)
- New technology trend: Helical Scan (VCR, Camcoder, DAT)
  Spins head at angle to tape to improve density

\[\text{Initialization of the LFSR to all 1s}\]

\[\text{Shift left and XOR-ing with the coefficients of } p(x)\]

\[\text{Evaluation of } x_k\]
Current Drawbacks to Tape

- Tape wear out:
  - Helical 100s of passes to 1000s for longitudinal
- Head wear out:
  - 2000 hours for helical
- Both must be accounted for in economic / reliability model
- Long rewind, eject, load, spin-up times; not inherent, just no need in marketplace (so far)
- Designed for archival

Automated Cartridge System

STC 4400

8 feet

10 feet

6000 x 0.8 GB 3490 tapes = 5 TBytes in 1992
$500,000 O.E.M. Price
6000 x 10 GB D3 tapes = 60 TBytes in 1998
Library of Congress: all information in the world; in 1992, ASCII of all books = 30 TB

Relative Cost of Storage Technology—Late 1995/Early 1996

Magnetic Disks

<table>
<thead>
<tr>
<th>Size</th>
<th>Capacity</th>
<th>Price/MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.25&quot;</td>
<td>9.1 GB</td>
<td>$2129</td>
</tr>
<tr>
<td>3.5&quot;</td>
<td>4.3 GB</td>
<td>$1985</td>
</tr>
<tr>
<td>2.5&quot;</td>
<td>514 MB</td>
<td>$299</td>
</tr>
<tr>
<td>1.1 GB</td>
<td></td>
<td>$345</td>
</tr>
</tbody>
</table>

Optical Disks

<table>
<thead>
<tr>
<th>Size</th>
<th>Capacity</th>
<th>Price/MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.25&quot;</td>
<td>4.6 GB</td>
<td>$1695+199</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$1499+189</td>
</tr>
</tbody>
</table>

PCMCIA Cards

<table>
<thead>
<tr>
<th>Type</th>
<th>Capacity</th>
<th>Price/MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static RAM</td>
<td>4.0 MB</td>
<td>$700</td>
</tr>
<tr>
<td>Flash RAM</td>
<td>40.0 MB</td>
<td>$1300</td>
</tr>
<tr>
<td></td>
<td>175 MB</td>
<td>$3600</td>
</tr>
</tbody>
</table>

Manufacturing Advantages of Disk Arrays

Conventional: 4 disk designs

Disk Product Families

Low End → High End

Disk Array: 1 disk design

3.5"

Replace Small # of Large Disks with Large # of Small Disks! (1988 Disks)

<table>
<thead>
<tr>
<th></th>
<th>IBM 3390 (K)</th>
<th>IBM 3.5&quot; 0061</th>
<th>x70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Capacity</td>
<td>20 GBytes</td>
<td>320 MBytes</td>
<td>23 GBytes</td>
</tr>
<tr>
<td>Volume</td>
<td>97 cu. ft.</td>
<td>0.1 cu. ft.</td>
<td>11 cu. ft.</td>
</tr>
<tr>
<td>Power</td>
<td>3 KW</td>
<td>11 W</td>
<td>1 KW</td>
</tr>
<tr>
<td>Data Rate</td>
<td>15 MB/s</td>
<td>1.5 MB/s</td>
<td>120 MB/s</td>
</tr>
<tr>
<td>I/O Rate</td>
<td>600 I/Os/s</td>
<td>55 I/Os/s</td>
<td>3900 I/Os/s</td>
</tr>
<tr>
<td>MTTF</td>
<td>250 Khrs</td>
<td>50 Khrs</td>
<td>?? Khrs</td>
</tr>
<tr>
<td>Cost</td>
<td>$250K</td>
<td>$2K</td>
<td>$150K</td>
</tr>
</tbody>
</table>

Array Reliability

- Reliability of N disks = Reliability of 1 Disk + N
- 50,000 Hours + 70 disks = 700 hours
- Disk system MTTF: Drops from 6 years to 1 month!
- Arrays (without redundancy) too unreliable to be useful!

Hot spares support reconstruction in parallel with access: very high media availability can be achieved
Redundant Arrays of Disks

- Files are "striped" across multiple spindles
- Redundancy yields high data availability

Disks will fail
- Contents reconstructed from data redundantly stored in the array
  - Capacity penalty to store it
  - Bandwidth penalty to update

Techniques:
- Mirroring/Shadowing (high capacity cost)
- Horizontal Hamming Codes (overkill)
- Parity & Reed-Solomon Codes
- Failure Prediction (no capacity overhead!)
  VaxSimPlus — Technique is controversial

Redundant Arrays of Disks RAID 1: Disk Mirroring/Shadowing

- Each disk is fully duplicated onto its "shadow"
  - Very high availability can be achieved
  - Bandwidth sacrifice on write:
    Logical write = two physical writes
  - Reads may be optimized
  - Most expensive solution: 100% capacity overhead
  Targeted for high I/O rate, high availability environments

Redundant Arrays of Disks RAID 3: Parity Disk

- Parity computed across recovery group to protect against hard disk failures
  - 33% capacity cost for parity in this configuration
  - Wider arrays reduce capacity costs, decrease expected availability, increase reconstruction time
- Arms logically synchronized, spindles rotationally synchronized logically a single high capacity, high transfer rate disk

Targeted for high bandwidth applications: Scientific, Image Processing

Problems of Disk Arrays: Small Writes

RAID-5: Small Write Algorithm

- 1 Logical Write = 2 Physical Reads + 2 Physical Writes

Subsystem Organization

- Single board disk controller
- Single board disk controller
- Single board disk controller
- Single board disk controller
- Often piggy-backed in small format devices
System Availability: Orthogonal RAIDs

Data Recovery Group: unit of data redundancy
Redundant Support Components: fans, power supplies, controller, cables
End to End Data Integrity: internal parity protected data paths

System-Level Availability

Fully dual redundant

Goal: No Single Points of Failure

with duplicated paths, higher performance can be

Summary

- Disk industry growing rapidly, improves:
  - bandwidth 40%/yr
  - areal density 60%/year, $/MB faster?
- queue + controller + seek + rotate + transfer
- Advertised average seek time benchmark much greater than average seek time in practice
- Response time vs. Bandwidth tradeoffs
- Queueing theory: $W = \frac{1}{\lambda - \mu}$ or $W = \frac{\rho}{1 - \rho}$
- Value of faster response time:
  - 0.7 sec off response saves 4.9 sec and 2.0 sec (70%) total time per transaction => greater productivity
  - everyone gets more done with faster response, but novice with fast response = expert with slow