CMP 1203

LECTURE 8

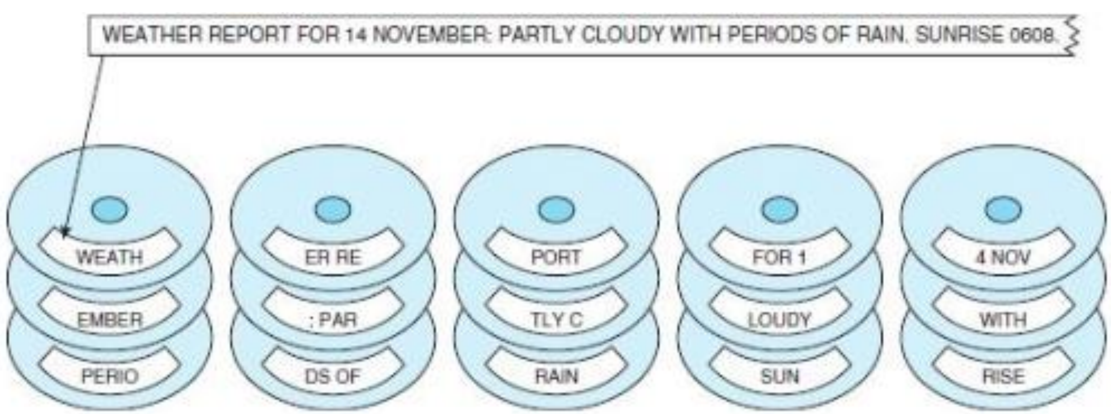
RAID : Redundant Arrays of Independent Disks

- Early disk drives large and costly
- Required highly controlled environments – heat – damaged circuitry, humidity- built up static
- Head crashes/ other failures could damage entire disks
- Needed more reliable disks
- Patterson, Katz and Gibson : improve reliability and performance using a number of “inexpensive” small disks ; Redundant Arrays of Inexpensive Disks
- Inexpensive is a relative term : “Independent”

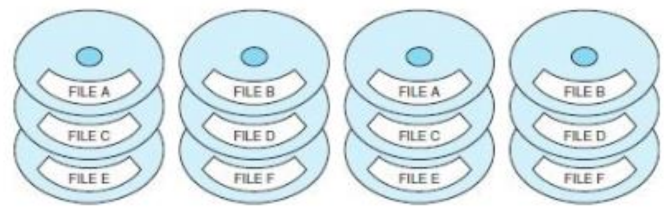
IBM RAMAC

RAID Level 0 (RAID-0)

- Data blocks placed in stripes across several disk surfaces
- One record occupies several sectors across many disk surfaces
- Also called drive spanning/ block interleave data striping or disk striping
- No redundancy ➔ best performance BUT very unreliable in case of faults
- Very inexpensive
- Very unreliable: as number of disks increases, probability of fault increases
- Recommended for non-critical data or data that doesn't change frequently and is backed up regularly, requires high-speed read/write and low cost e.g. video/image editing



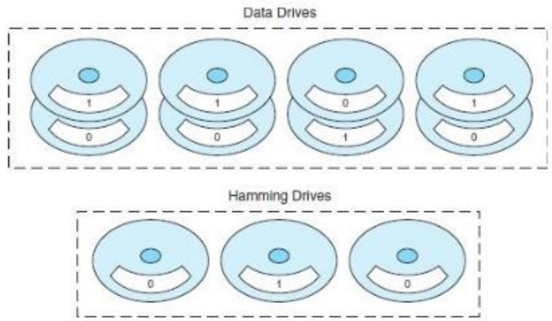
RAID -1



- Also disk mirroring
- Best failure protection
- Data is written on two sets of drives
- Second drive is mirror set/shadow set
- Acceptable performance
- Slower write performance than RAID-0 because data is written twice
- Faster read performance because can read from disk arm which is closest to target sector
- Good for transactions and applications that need high fault tolerance e.g. accounting, payroll
- Expensive because need twice as many disks to store given amount of information

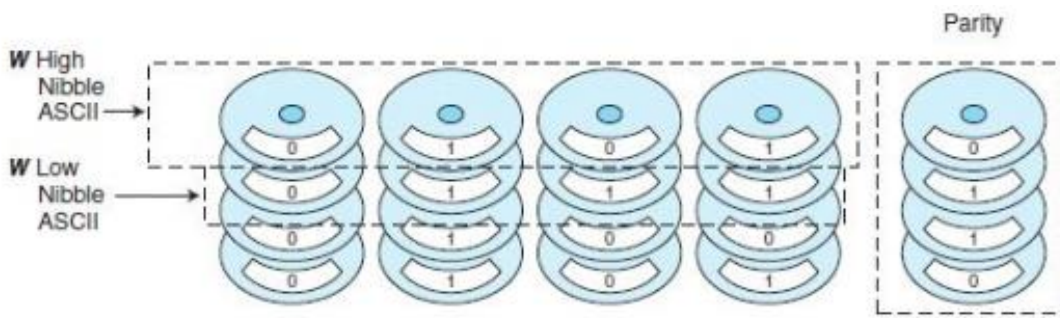
RAID -2

- Uses extreme data striping
- Writes one bit per strip (instead of blocks)
- Additional drives used for error correction using hamming codes
- Failed drive can be reconstructed from hamming drive and vice versa
- Acts as if it was one big drive since one data bit written per drive
- Need accurate synchronization else data becomes scrambled
- Hamming code generation is time consuming hence RAID-2 is too slow for commercial implementation
- Theoretical



RAID-3

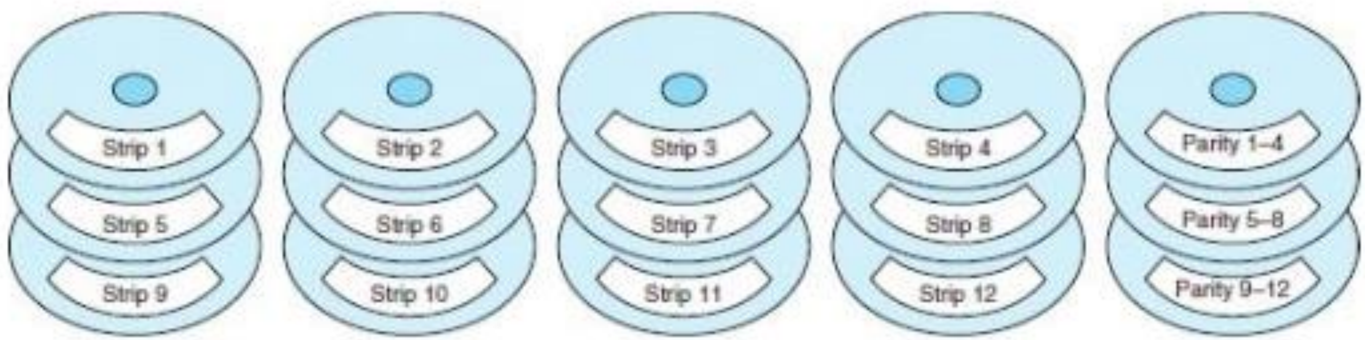
- Similar to RAID-2 but uses only one drive to hold a parity bit
- Same duplication and synchronization as RAID-2 but more economical because only 1 drive for protection



Letter	ASCII	Parity(even)	
		High Nibble	Low Nibble
W	0101 0111	0	1
E	0100 0101	1	0
A	0100 0001	1	1
T	0101 0100	0	1
H	0100 1000	1	1
E	0100 0101	1	0
R	0101 0010	0	1

RAID-4

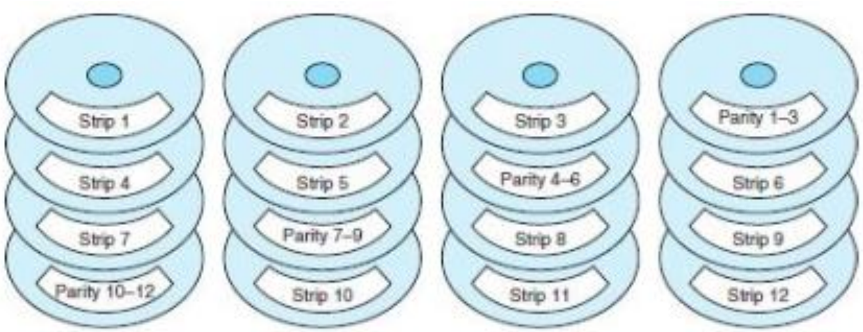
- Also theoretical level like RAID-2
- RAID-0 with parity
- Parity introduces performance bottle neck : need to write parity bit as well
- So cant service multiple writes concurrently e.g. write to strip 3, 4,1 because you need to update parity block



PARITY 1–4 = (Strip 1) XOR (Strip 2) XOR (Strip 3) XOR (Strip 4)

RAID-5

- RAID- 4 but spread parity disks across entire array
- Can service concurrent requests : provides best read throughput
- E.g. write to drive 4 strip 6 and drive 1 strip 7 because need different disk arms for both data and parity
- Most complex disk controller
- Best protection of all levels for least cost: commercially successful
- Used for file and application servers, email servers, database and web servers etc.

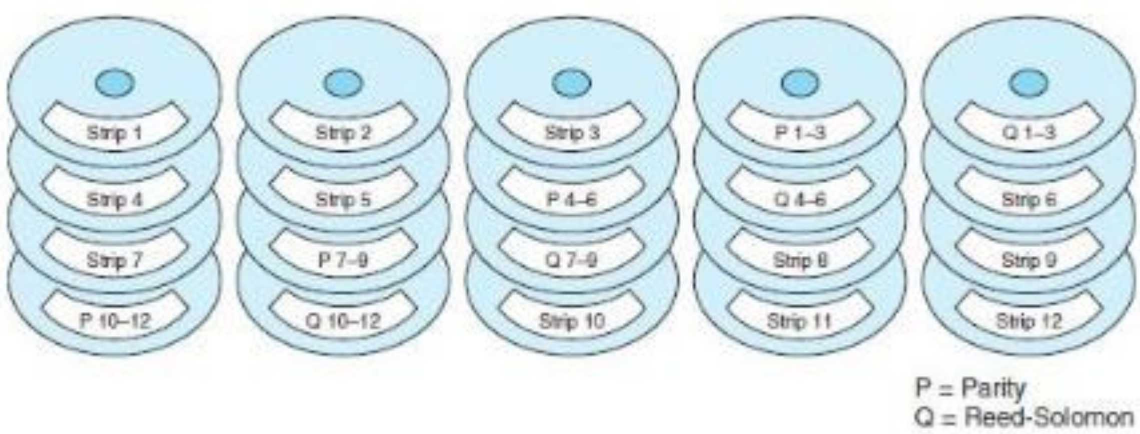


PARITY 1–3 = (Strip 1) XOR (Strip 2) XOR (Strip 3)

RAID-6

- Most raid systems can tolerate at most one disk failure
- BUT most disk failures occur in clusters. **Why?**

•Disks manufactured at same time have same end of life
•Disk failures: events that affect all drives at same time e.g. power surge
•RAID-1 can handle multiple disk failures if both drive and mirror not wiped out
•RAID-6 uses two sets of error correction strips for every horizontal row of drives :One is parity, second uses reed-solomon code
•This increases storage cost

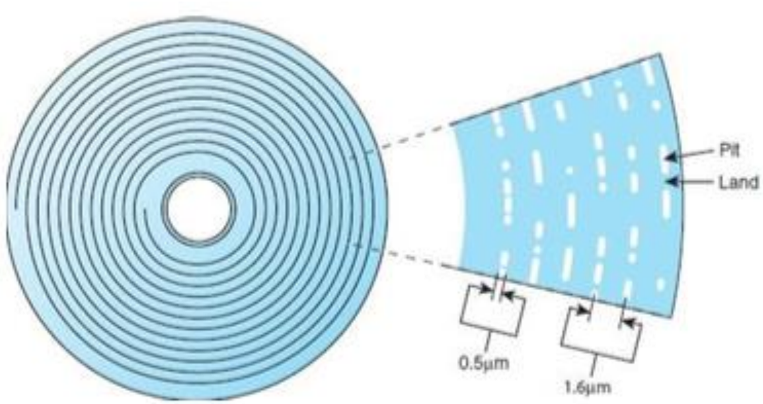


Homework

- RAID DP (double parity)
- Hybrid RAID systems

Optical Memory

- CD-ROM
- Polycarbonate disks covered with reflective aluminum film, then sealed with protective coating
- Compact discs written from center to outside edge using spiral track of bumps
- Bumps are called pits
- Spaced between pits called lands
- How do they work? https://www.youtube.com/watch?v=H-jxTzFrnpg



Self-Monitoring, Analysis and Reporting Technology (S.M.A.R.T)

- Monitoring system for computer HDDs to detect and report various indicators of reliability to predict failures
- It's an interface between BIOS and storage device
- If enabled, BIOS can process information from the storage device and send warning messages about potential failures
- Avails self-test or maintenance routines

Self-Monitoring, Analysis and Reporting Technology (S.M.A.R.T)

- Attributes to be monitored are set by manufacturer

SMART Attributes

Examples of SMART attributes, showing the

SMART attributes

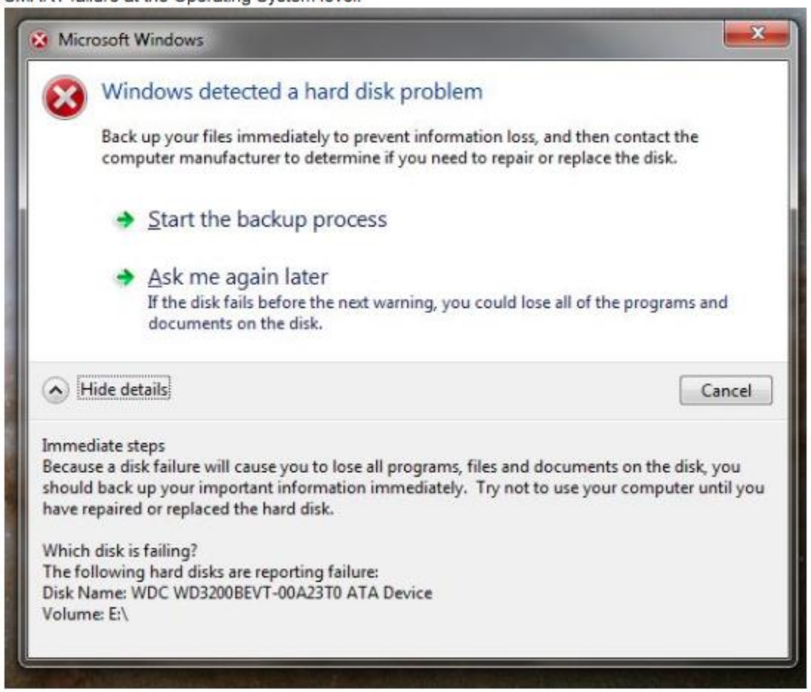
ID	Attribute name	Threshold	Value	Worst	Status	Raw Data	Raw Hex
01	Raw Read Error Rate	51	100	100	OK	0	000000000000
03	Spin Up Time	25	100	100	OK	3136	000000000C40
04	Start/Stop Count	0	99	99	OK	1653	000000000675
05	Reallocated Sector Count	11	100	100	OK	0	000000000000
07	Seek Error Rate	51	100	100	OK	0	000000000000
08	Seek Time Performance	15	100	100	OK	0	000000000000
09	Power-On Hours Count	0	100	100	OK	316913	000000004D5F1
0A	Spin-up Retry Count	51	100	100	OK	0	000000000000
0C	Power Cycle Count	0	100	100	OK	747	0000000002EB
BF	G-Sense Error Rate	0	93	93	OK	79827	0000000137D3
C2	HDA Temperature	0	124	82	OK	38	000000000026
C3	Hardware ECC Recovered	0	100	100	OK	5508	000000001584
C4	Reallocated Event Count	0	100	100	OK	0	000000000000
C5	Current Pending Sector Count	0	100	100	OK	0	000000000000
C6	Off-line Scan Uncorrectable Count	0	100	100	OK	0	000000000000
C7	UltraDMA CRC Error Rate	0	200	200	OK	17	000000000011
C8	Write Error Rate	51	100	100	OK	0	000000000000
C9	Soft Read Error Rate	0	100	100	OK	0	000000000000
DF	Load/Unload Retry Count	0	100	100	OK	2	000000000002
E1	Load/Unload Cycle Count	0	70	70	OK	305164	000000004A80C
FF		51	100	100	OK	0	000000000000

SMART reported good status.

Source: sandisk.com

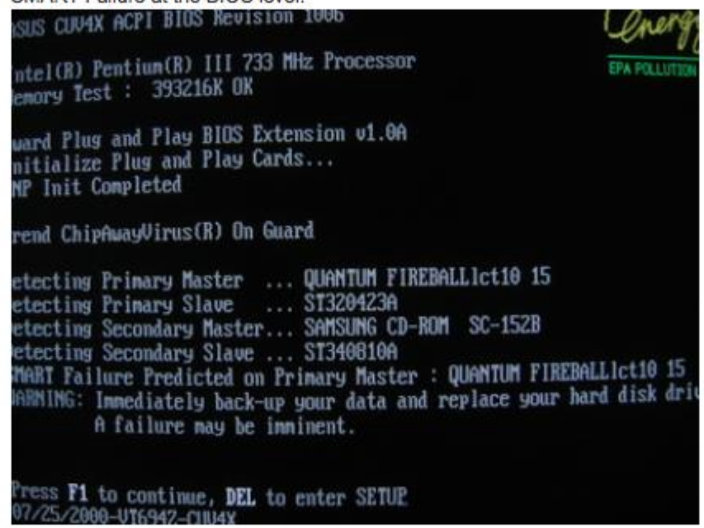
Self-Monitoring, Analysis and Reporting Technology (S.M.A.R.T)

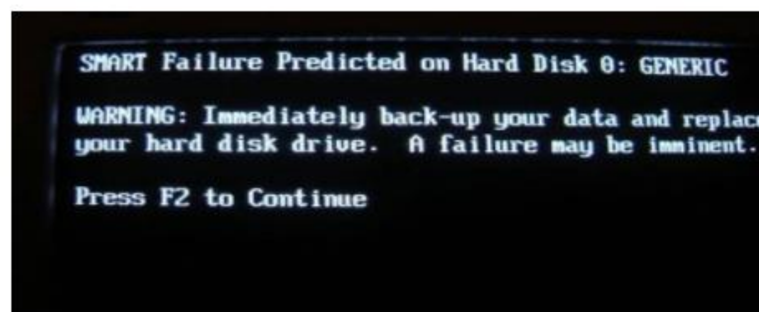
SMART failure at the Operating System level:



SMART Failure

SMART Failure at the BIOS level:





Source: sandisk.com

Error Detection and Correction

- No channel or storage medium is 100% error free
- The higher the transmission rate, higher bit timing
- The more bits stored per square mm of storage, magnetic flux densities increase
- Error rate increases proportional to transmission rate and number of bits per sq. mm of storage
- We cant create an error free medium and we cant detect 100% of all possible errors
- Trade-off: define an acceptable number of errors for system to work. As long as we can detect this and correct this “reasonable” number, all is ok.
- “reasonable” differs per application/ implementation

Error Detection: Parity method

- Uses an additional bit added to the code group →parity bit: either 0 or 1
- Even parity: choose bit so that the total number of 1's in the code group is even

e.g. **1 1 0 0 0 0 1 1 or 0 1 0 0 0 0 0 1**

- Odd parity : number of 1's is odd
- Can detect only single bit errors

Qn: Which is in error assuming odd parity?

1 1 0 0 0 0 0 1 or $\boxed{\text{1 1 0 0 0 0 0 0}}$

Qn: Which bit was in error?

Error Detection: Parity method

Qn: What if 2 bits are in error?

Ans: Doesn't change even or odd count of 1's so it wont identify that an error has occurred.

So when is it used?

- When probability of a single error very low and that of double errors is 0. For example? (Homework!)
- Transmitter and receiver have to agree before hand whether to use odd or even parity

Error Detection: CRC

- Modulo 2 addition: 0+0=0; 0+1=1; 1+0=1; 1+1=0.
- Modulo 2 division:

1. Write divisor directly beneath 1^{st} bit of dividend
2. Add them using modulo 2
3. Bring down bits from dividend until 1^{st} one of difference aligns with first 1 of divisor
4. Repeat until you reach number not divisible by divisor

quotient $\mathbf{1010}$

$$\mathbf{1011}\,\overline{\big)\,\mathbf{1001011}}$$

$$+\underline{\mathbf{1011}\downarrow}$$

$$\mathbf{0100}$$

$$+\ \underline{\mathbf{0000}\downarrow}$$

$$\mathbf{1001}$$

$$+\ \underline{\mathbf{1011}\downarrow}$$

$$\mathbf{0101}$$

$$+\ \underline{\mathbf{0000}}$$

remainder $\mathbf{101}$

Error Detection: CRC

- Operations have polynomial equivalents

$$\mathbf{1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0}$$

$$\textbf{Let}\quad \mathbf{X = 2;\quad 1 \times X^3 + 0 \times X^2 + 1 \times X^1 + 1 \times X^0}$$

- CRC uses such as generator polynomials

1. Let information be $\boldsymbol{I = 1001011_2}$
2. Sender and receiver agree on a binary pattern e.g.

$$\boldsymbol{P = 1011_2}$$

Error Detection: CRC

3. Shift I to left by one less than number of bits in P i.e.

$$I = \mathbf{1001011000}_2$$

4. Do modulo 2 division (work this out!) Remainder?? Ans=100_2 becomes CRC check sum.
5. Add remainder to I to give message M [1001011100_2]
6. At receiver, message is decoded by M÷P. If remainder is 0, no error. If remainder is another value, then there was an error.

Example

- Key: 1101
- Received word: 10000001
- Determine if there is an error.
- What word will be sent if the message is 100100?

Solution

Receiver Side

Let there be error in transmission media

Code word received at the receiver side - 100000001

```
          111010
      __________
1101 | 100000001
       1101
       ----
        1010
        1101
        ----
         1110
         1101
         ----
          0110
          0000
          ----
           1100
           1101
           ----
            0011
            0000
            ----
             011
            ----
```

Since the remainder is not all zeroes, the error is detected at the receiver side.

Data word to be sent - 100100

Key - 1101

Sender Side:

```
          111101
      __________
1101 | 100100000
       1101
       ----
        1000
        1101
        ----
         1010
         1101
         ----
          1110
          1101
          ----
           0110
           0000
           ----
            1100
            1101
            ----
             001
            ----
```

Therefore, the remainder is 001 and hence the code word sent is 100100001.

Hamming Code

- In data communication systems, error detection is enough.
- If an error is detected, simply ask sender to re transmit.
- This is impossible with storage systems and memory.
- We must recover the data.
- Hamming codes are an efficient way
- Based on Parity
- Use check bits/ redundant bits
- Code word has n bits; $n = m + r$: m data bits and r check bits

Hamming Code

- Define hamming distance as number of bit positions in which 2 code words differ.
- E.g. code words
- 10001001 and
- 10110001 have hamming distance of 3
- Hamming distance determines how many bit errors can be detected
- If distance is d, if d bits are flipped, error cannot be detected because it produces a valid code word
- Smallest hamming distance D(min) is the smallest distance between all pairs of code words
- To detect k errors, need minimum distance of k+1
- To correct k errors, minimum hamming distance must be at least 2k+1

Example

Data Word	Parity Bit	Code word
00	0	000
01	1	011
10	1	101
11	0	110
But 3 bits can have 8 possible combinations; only above 4 are valid words		
000	100	
001	101	
010	110	
011	111	

- Assume even parity
- If we receive 111; invalid code word hence error has occurred
- But we cant correct it
- Cant tell how many bits flipped and which in error
- What if 2 bits flipped? A valid code word is generated: D(min) is 2: So can detect only single bit errors

Example 2

0 0 0 0 0
0 1 0 1 1
1 0 1 1 0
1 1 1 0 1

- Consider code below
- D(min) = 3 (compare all code words)
- Can detect 2 errors and correct 1 bit error
- What if we receive 10000?
- Find hamming distance between each code word [1,4,2,3]
- Choose legal code word closest to received word
- It may not necessarily be correct. We have assumed 1 bit error occurred
- What if we received 11000 and we know 2 bits flipped? [2,3,3,2]
- We don't know closest code word!

Hamming Code Generation

- Use inequality: $$(m + r + 1) \le 2^r$$
- Specifies lower limit of how many check bits are needed. (For single bit errors)
- Homework: How do we derive this inequality?
- If data words have m = 4 bits, then

$$(4 + r + 1) \le 2^r \quad ; r \ge 3$$

- So we need 3 check bits to build a code word to correct single bit errors for a 4 bit data word

Hamming Code Generation

1. Determine number of check bits r required.
2. Number the n bits from right to left starting with 1
3. Each bit whose position is a power of 2 is a parity bit : rest are data bits
4. Parity bits check specific data bits such that: bit b is checked by the parity bits b1, b2, … bj such that b1+b2+….+bj=b (+ is modulo 2 sum)

Example

- Encode data word 01001011 using a hamming code
- Step 1: Determine code word
- m = 8; $(8+r+1) \leq 2^r \quad ; r \geq 4$
- Step 2: number bits from right to left

				X				X		X	X
12	11	10	9	8	7	6	5	4	3	2	1

- X are parity bit positions (power of 2 positions)

Hamming Code Generation

- Step 3: Assign parity bits to check bit positions. Write bit positions as sums of numbers which are powers of 2

1=1	5=1+4	9=1+8
2=2	6=2+4	10=2+8
3=1+2	7=1+2+4	11=1+2+8
4=4	8=8	12=4+8

- 1 appears in 1,3,5,7,9,11 so it will be show parity for those positions
- 2 will check 2,3,6,7,10,11 etc

Hamming Code Generation

- Write data words in blank positions and fill parity
- Assume even parity 01001011

0	1	0	0	X	1	0	1	X	1	X	0
12	11	10	9	8	7	6	5	4	3	2	1

- 1: 1,3,5,7,9,11
- 2:2,3,6,7,10,11
- 4:4,5,6,7,12
- 8: 8,9,10,11,12

Fill in the rest

Hamming Code Generation

- Write data words in blank positions and fill parity
- Assume even parity

0	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**0**
12	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**

- 1: 1,3,5,7,9,11
- 2:2,3,6,7,10,11
- 4:4,5,6,7,12
- 8: 8,9,10,11,12

Code word for K: 010011010110

Hamming Code Generation

- Assume error in position 9: 010111010110- received
- Even parity 010011010110 - codeword

0	1	0	1	1	1	0	1	0	1	1	0
12	11	10	9	8	7	6	5	4	3	2	1

- 1: 1,3,5,7,9,11 : error
- 2:2,3,6,7,10,11 : ok
- 4:4,5,6,7,12 :ok
- 8: 8,9,10,11,12 :error

Code word for K: 010011010110

Hamming Code Generation

- Assume error in position 9: 010111010110

0	1	0	1	1	1	0	1	0	1	1	0
12	11	10	9	8	7	6	5	4	3	2	1

•1: 1,3,5,7,9,11 : error

2:2,3,6,7,10,11 : ok

•4:4,5,6,7,12 :ok

8: 8,9,10,11,12 :error

•Parity bits 1 and 8 have errors: common bits are 9 and 11

•But 11 checked by parity 2 and is ok

•So error must be on 11

•We can also just sum parity bits in error (1+8) =9 to get bit position with error

Home work

- Reed Solomon codes

Handout

- Lobur and Null; Chapter 2 and 7
- Stallings, Chapter 6 , 5

- NOTE: CAT 2 - 15th April