CSCI 4717/5717 Computer Architecture

Topic: Error Detection & Correction

Reading: Stallings, Section 5.2

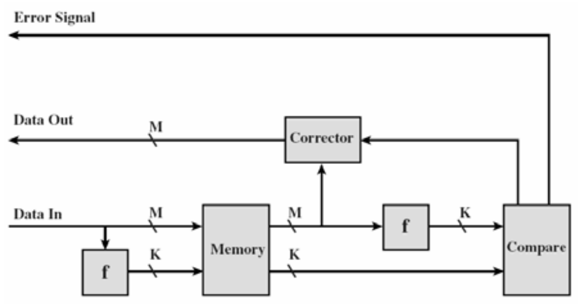
Error Correction in Memory

- Types of errors: hard or soft
- Hard Failure – Permanent defect caused by
  - Harsh environmental abuse (including static electricity)
  - Manufacturing defect
  - Wear such as trace erosion
- Soft Error
  - Random, non-destructive
  - Caused by electrical or EM/radioactive glitches
  - No permanent damage to memory

Error Detection & Correction

- Additional information must be stored to detect these errors
- When M bits of data are stored, they are run through function *f* where a K bit code is created
- M+K bits are then stored in memory
- When data is read out, it is once again run through function f and the resulting K bits of code are compared with the stored K bits of code
- In some cases, the code can be corrected (error correcting codes)
- In all cases, and error code is generated

Error Correcting Code Function



Hamming Error Correction Code

- One way to detect specific bit errors is to use multiple parity bits, each bit responsible for the parity of a smaller, overlapping portion of the data
- A flipped bit in the data would show up as a parity error in the overlapping groups of which it was a member and not in the other groups
- This would handle single-bit corrections

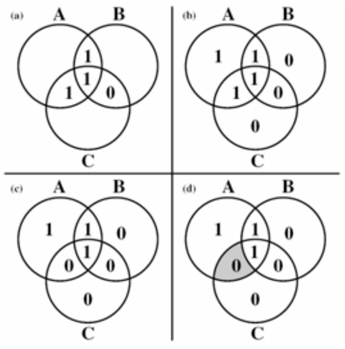
4-bit Hamming Code

- Below is an example of a 4-bit word broken into 3 groups; each group has a parity bit to generate even parity.
- D_n represent data bits while P_n represent parity bits

	D_3=1	D_2=0	D_1=1	D_0=1	P_0	P_1	P_2
Group A	1	0	1		0		
Group B	1		1	1		1	
Group C	1	0		1			0

4-bit Hamming Code (continued)

Can be represented graphically using three intersecting circles.



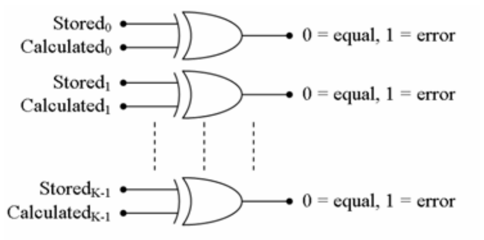
4-bit Hamming Code (continued)

- Areas are defined as:
  - A and B, but not C
  - A and C, but not B
  - B and C, but not A
  - A and B and C
- Each non-intersecting area contains a parity bit to make it and the three intersecting areas in a single circle have even parity.
- A change in only one area will make parity odd in 2 or all 3 of the circles indicating which intersection changed.

General Single-Bit Error Correction

- The mechanics of the typical error correction/detection system are created with XOR gates
  - Odd number of ones input to an XOR → 1 output
  - Even number of ones input to an XOR → 0 output
- Upon data retrieval, two K-bit values are generated:
  - The stored K-bit value
  - The K-bit value generated from the stored data
- A bit-by-bit comparison is performed on these two values generating a K-bit result
  - 0's in bit positions where there is no error
  - 1's in bit positions where two bits disagree
  - K-bit result is called a *syndrome word*

Generation of Syndrome Word



Syndrome Word

- All zeros means that the data was successfully retrieved
- For data with M bits and K code bits, then there are M+K possible single bit errors, i.e., there could be an error in the data OR the K-bit code
- For a K bit syndrome word, there are 2^K-1 (minus one for the no error case) possible values to represent single-bit errors
- Therefore, for the system to uniquely identify bit errors, $2^K-1 \geq M+K$

Single Error Correcting (SEC) Code Example

- Assume M=8
- First, how big does K have to be?
  K=3: $2^3-1 \geq 8+3$? (7 is not $\geq$ 11)
  K=4: $2^4-1 \geq 8+4$? (15 is $\geq$ 12)

SEC Code Example (continued)

- Next, decide what the values of the syndrome word represent
- 0 = no errors in syndrome word or data
- Only one bit of syndrome word set to one (1000, 0100, 0010, or 0001) = error was in syndrome word and data needs no correction
- Multiple bits of syndrome word set to one = digit represented by syndrome word identifies which bit of data was flipped and needs to be corrected

SEC Code Example (continued)

The table below is used to identify which bits of the M+K bits of the combined data and syndrome word are associated with which possible values of the syndrome word.

M+K Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bits	D8	D7	D6	D5		D4	D3	D2		D1		
Code bits					C8				C4		C2	C1

SEC Code Example (continued)

- We need a system such that the XOR-ing of the stored code or check bits with the code or check bits calculated identifies the position number from the table above.
- ***This means that when a bit changes in the data, then ones need to appear in the digits identifying that position.***
- Each code bit C8, C4, C2, and C1 is calculated by XOR-ing all of the bits in that position that have a 1.

SEC Code Example (continued)

- $C8 = D8 \oplus D7 \oplus D6 \oplus D5$
- $C4 = D8 \oplus D4 \oplus D3 \oplus D2$
- $C2 = D7 \oplus D6 \oplus D4 \oplus D3 \oplus D1$
- $C1 = D7 \oplus D5 \oplus D4 \oplus D2 \oplus D1$

Single Error Correcting, Double Error Detecting (SEC-DED) Code

- Double error detection will not correct double errors, but it will see if a double error has occurred.
- Adds additional bit for even parity to the M+K bits of the data and check code
- If one bit changed, the change caused parity to go from even to odd.
- Changing it back will restore parity
- If two bits changed, parity stayed even and a correction will force parity to go to odd indicating a double error.

SEC-DED Example