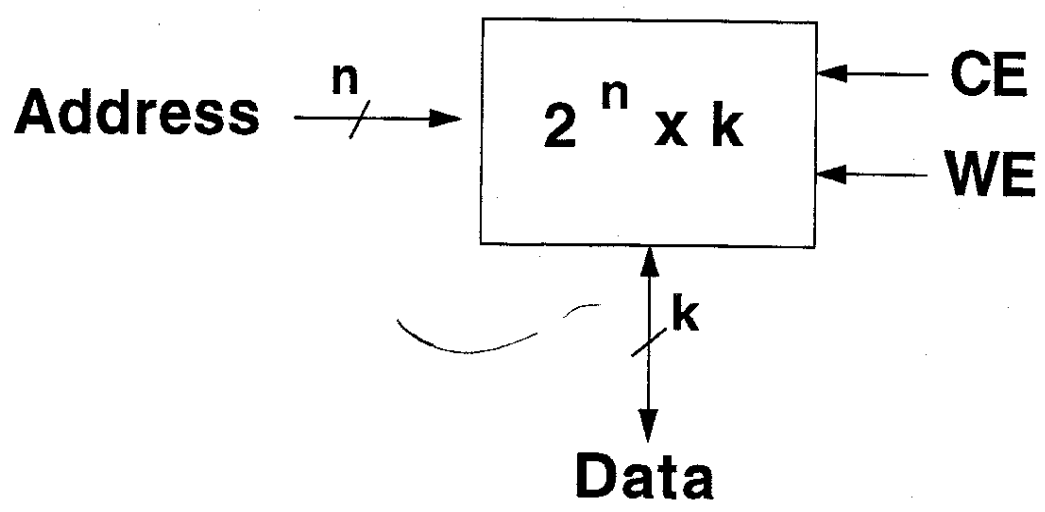


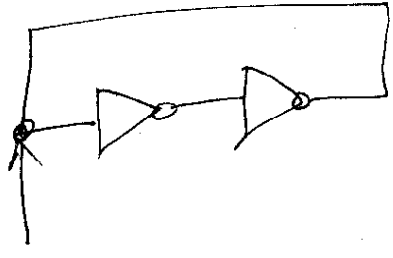
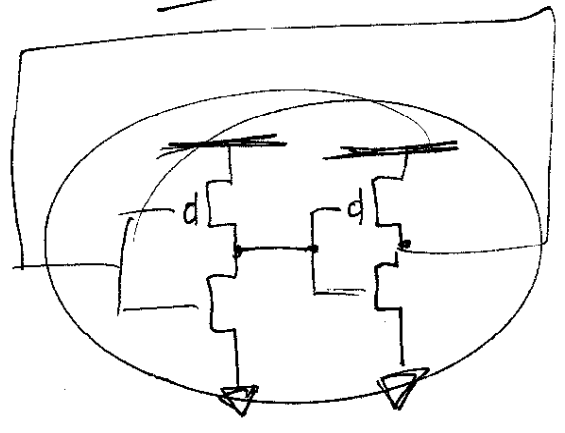
## The Memory Chip



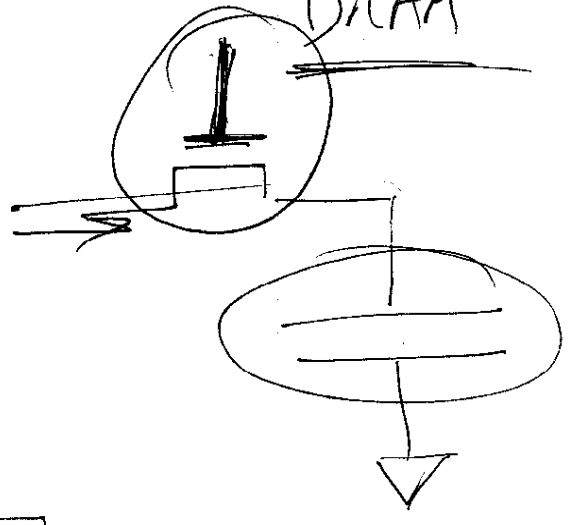
- \* **SRAM/DRAM**
  - **Size, Access Time**
  - **Refresh**

\* **Page Mode**

SRAM



DRAM



ISSUES

## \* ADDRESS SPACE

- MAXIMUM SIZE OF MEMORY

## \* ADDRESSABILITY

- GRANULARITY
- BYTE WRITES
- ECC BITS / PARITY / CHECK SUM  
(SECDED)

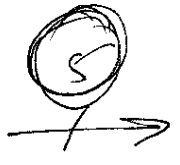
## \* ALIGNMENT

- SOFTWARE OR HARDWARE  
(WHO GETS THE CUSHY JOB)

## \* INTERLEAVING

- OLD DAYS : LATENCY
- TODAY : MULTIPLE CONCURRENT

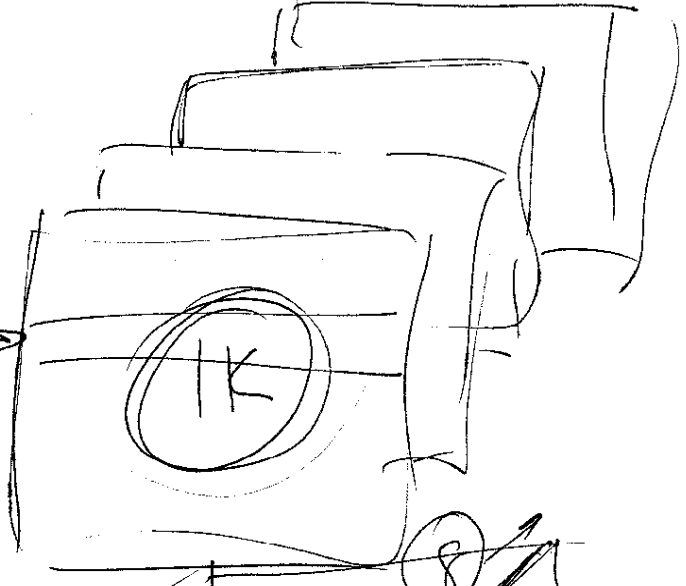
RAS



RA



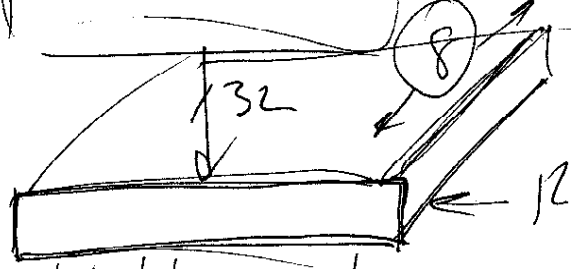
5



CA



32

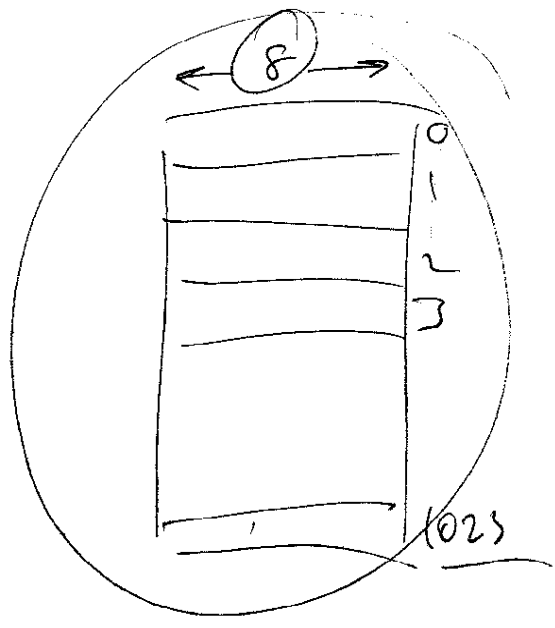


ROW BUFFER

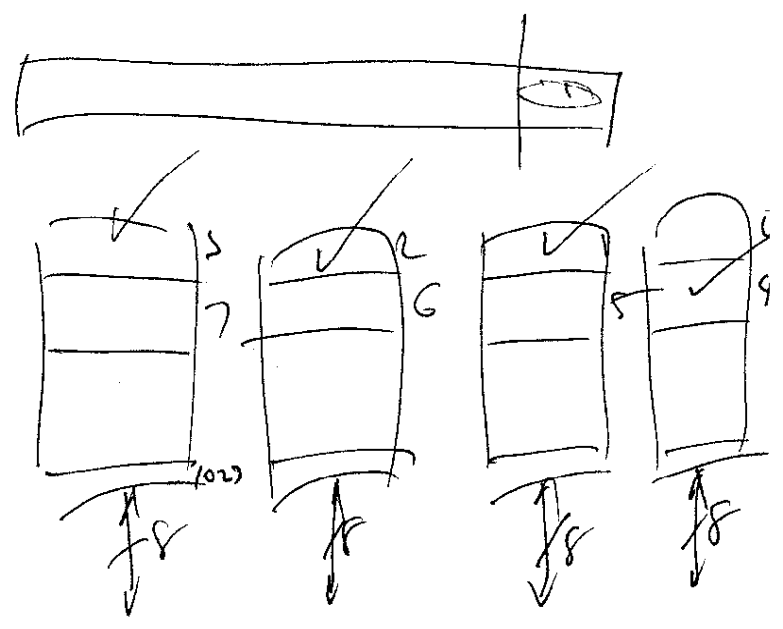
CAS

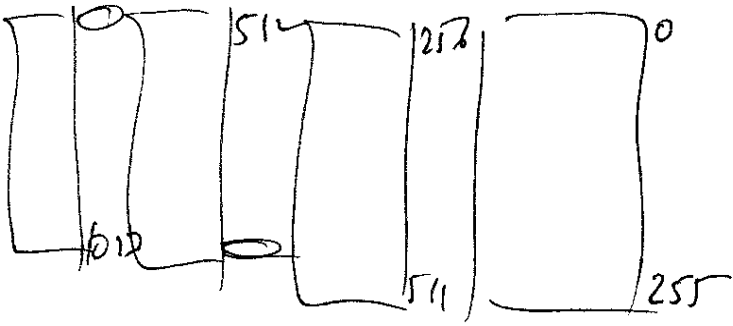


PAGE MODE

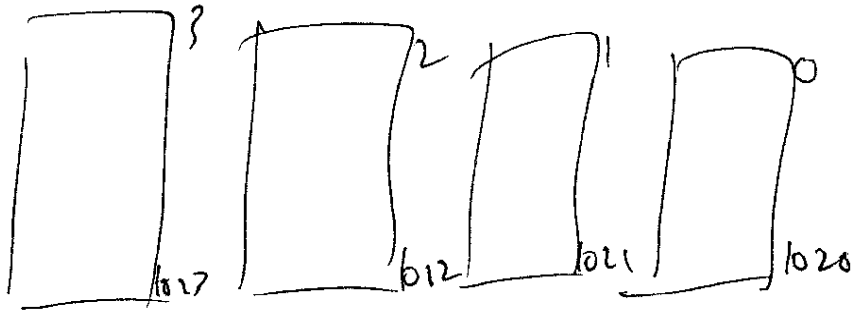


INTERLEAVING



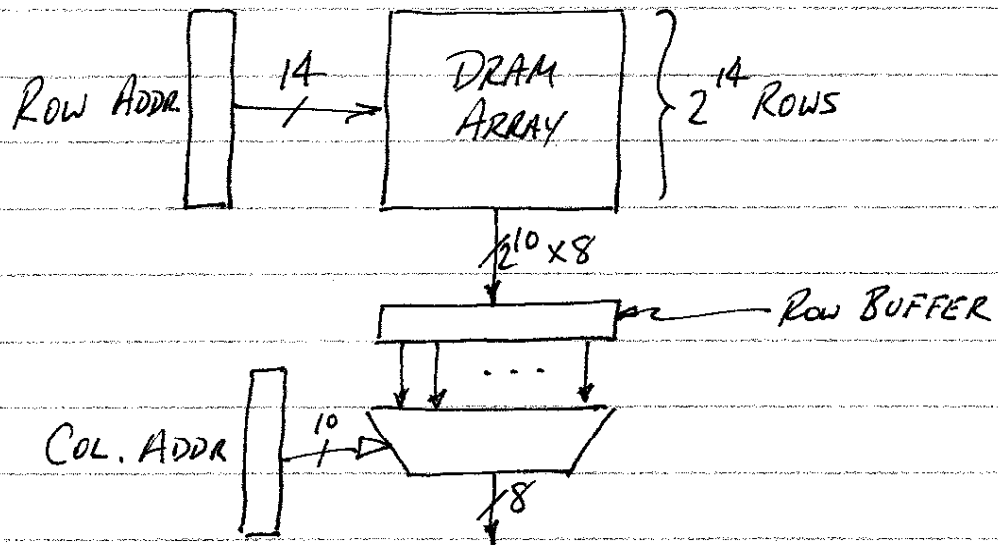


← Not interleaved  
Interleaved



# A 1 GB DRAM SYSTEM

## \* FIRST, THE DRAM STRUCTURE



- THE DRAM ARRAY STORES  $2^{24}$  BYTES OF INFORMATION  
i.e. A 16 MB ARRAY

- IF THE "NEXT" ACCESS IS TO THE SAME ROW AS THE PREVIOUS ACCESS, WE GET A ROW BUFFER HIT

ERGO, NO NEED TO TRANSFER ROW ADDR AGAIN.  
FASTER. CALLED PAGE MODE.

## \* THE DRAM CHIP

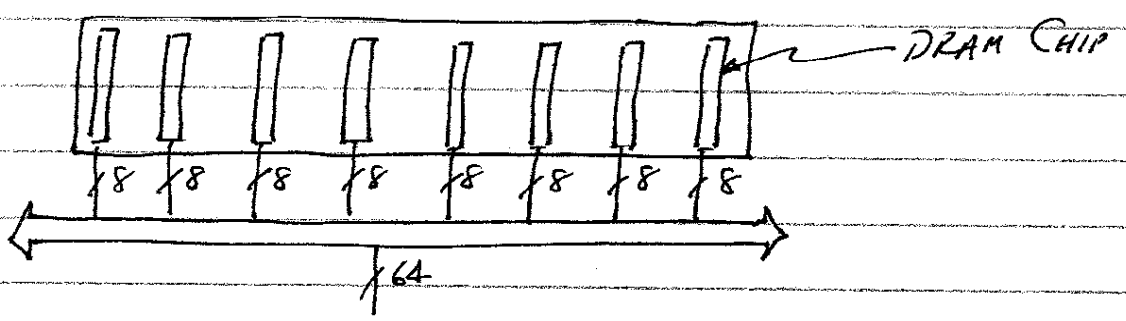
- THE DRAM CHIP CONSISTS OF 8 OF THESE DRAM STRUCTURES, EACH WITH ITS OWN ROW ADDR REG., COL. ADDR REG., ROW BUFFER.

# 1 GB DRAM (CONTINUED)

## \* THE DRAM CHIP (CONTINUED)

- THIS ALLOWS INTERLEAVING THE 8 ARRAYS ON THE CHIP INTO 8-WAY INTERLEAVED MEMORY
- TOTAL STORAGE ON THE CHIP:  $8 \times 16 \text{ MB}$   
(SILLY MARKETING CALLS IT 1 Gbit)  
I PREFER 128 MB SINCE WE HAVE  $2^{27}$  ADDRESSES, EACH OF WHICH STORES ONE BYTE

## \* A TYPICAL SYSTEM CONTAINS A MODULE CONSISTING OF 8 DRAM CHIPS, AS FOLLOWS



- THIS PROVIDES FOR A 64-bit ~~wide~~ wide TRANSFER BETWEEN THE PROCESSOR AND MEMORY
- ADDRESSES TO THE SAME BANK IN ALL CHIPS ARE IDENTICAL

\* IN SUMMARY, THE PHYSICAL ADDRESS:

k	14	3	10	3
ROW	BANK		COL	

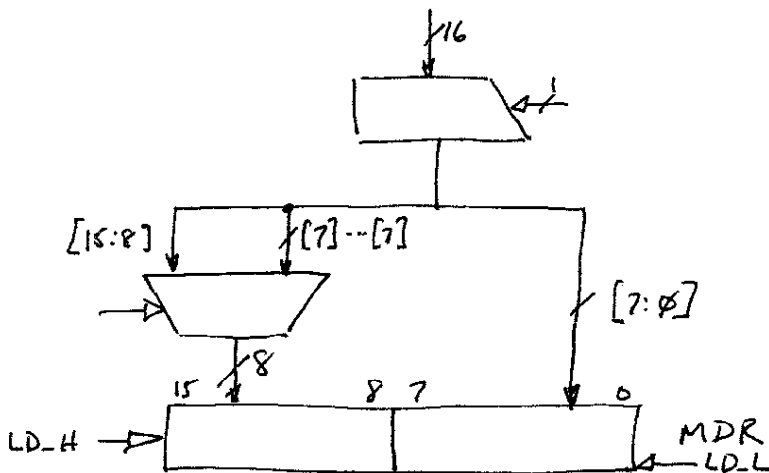
RANK BITS IF YOU NEED MORE THAN ONE MODULE (i.e., MORE THAN 1GB)

ADDR. BITS THAT DO NOT GO TO MEMORY

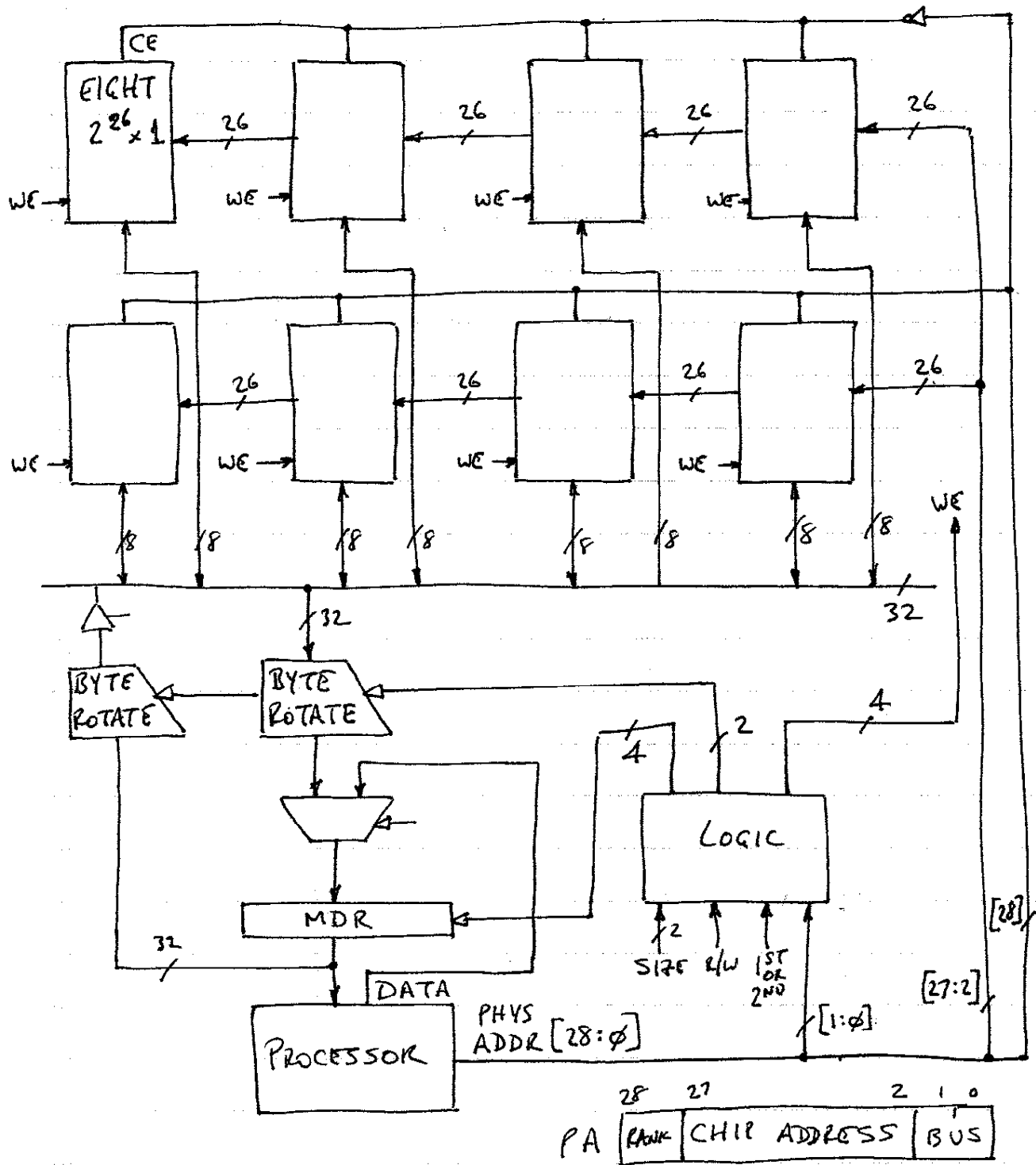
# UN-ALIGNED ACCESS

Mem/9

R/W	SIZE	MAR[0]	1 <sup>st</sup> / <sub>2<sup>nd</sup></sub>	SEN	LDH	LD.L	ROT	WE1	WE0
R	B	0	1	1	1	1	0	0	0
R	B	0	2	X	0	0	X	0	0
R	B	1	1	1	1	1	1	0	0
R	B	1	2	X	0	0	X	0	0
R	W	0	1	0	1	1	0	0	0
R	W	0	2	X	0	0	X	0	0
R	W	1	1	0	0	1	1	0	0
R	W	1	2	0	1	0	1	0	0
W	B	0	1	X	0	0	0	0	1
W	B	0	2	X	0	0	X	0	0
W	B	1	1	X	0	0	1	1	0
W	B	1	2	X	0	0	X	0	0
W	W	0	1	X	0	0	0	1	1
W	W	0	2	X	0	0	X	0	0
W	W	1	1	X	0	0	1	1	0
W	W	1	2	X	0	0	1	0	1

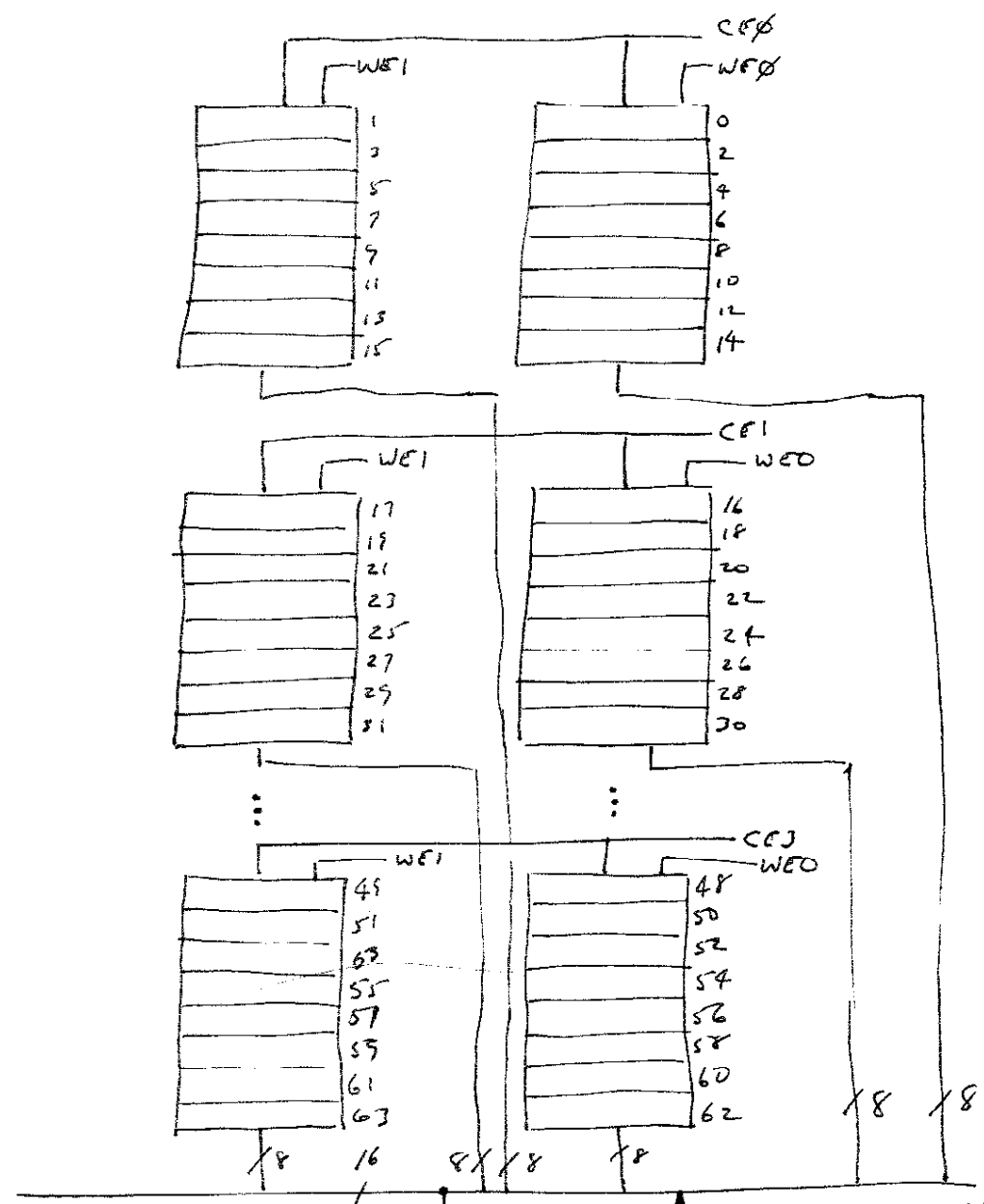




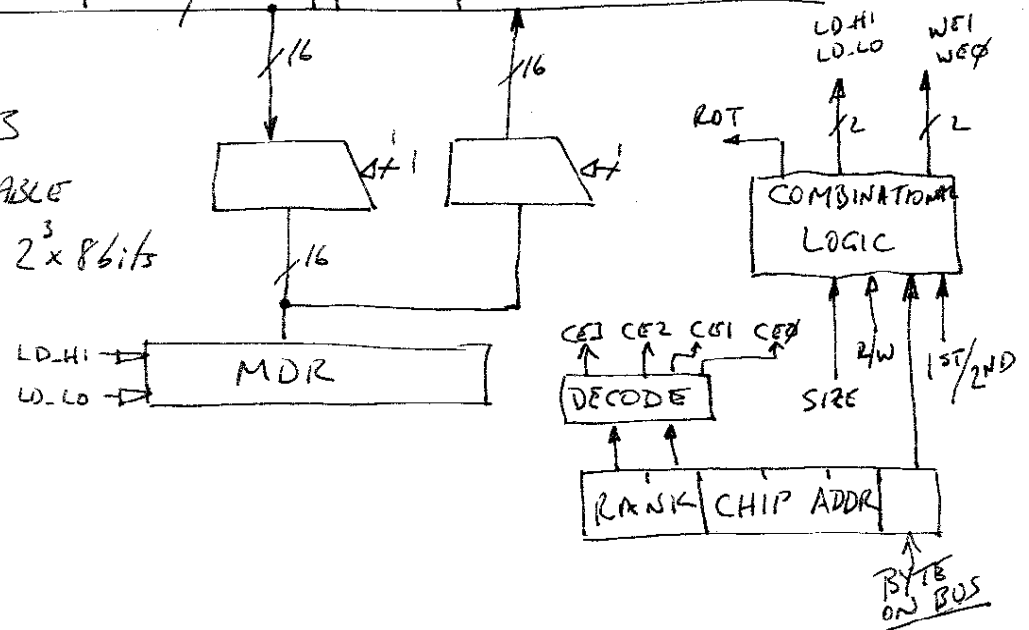


UNALIGNED ACCESSES  
(SIMPLIFIED BLOCK DIAGRAM)

# UN-ALIGNED ACCESS (NON-INTERLEAVED)

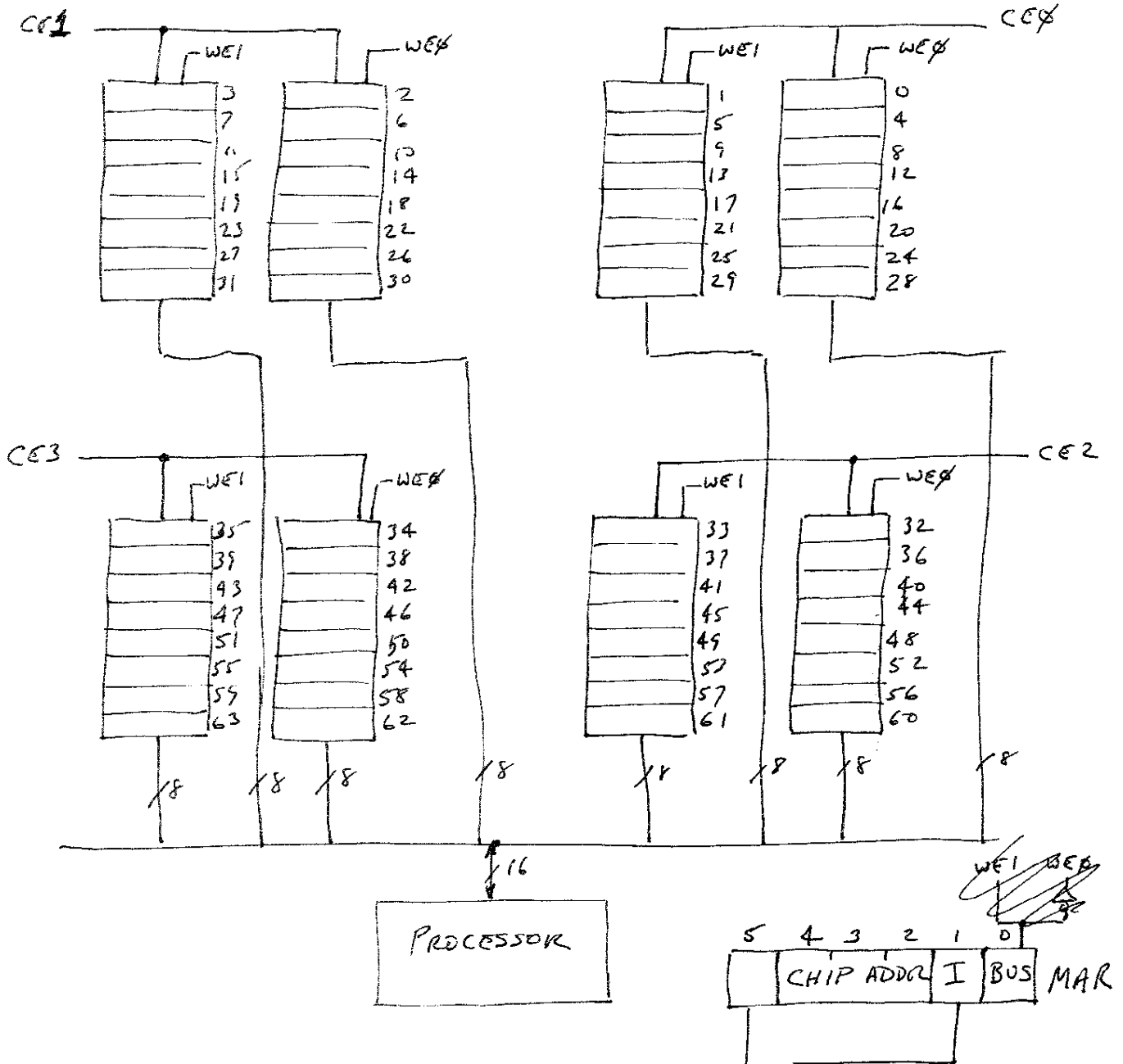


Memory : 64 B  
 BYTE ADDRESSABLE  
 EACH CHIP :  $2^3 \times 8 \text{ bits}$

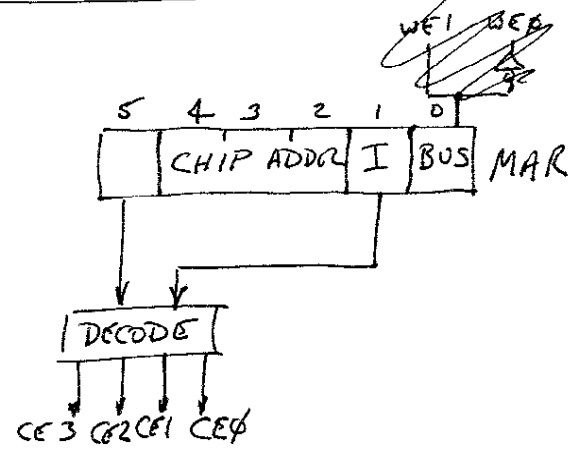


↑  
 BYTE ON BUS

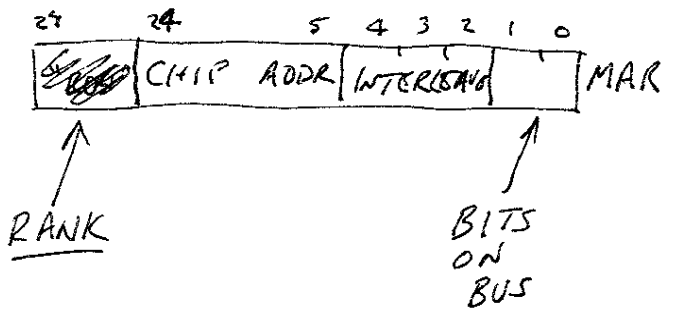
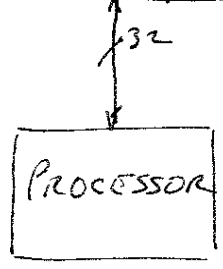
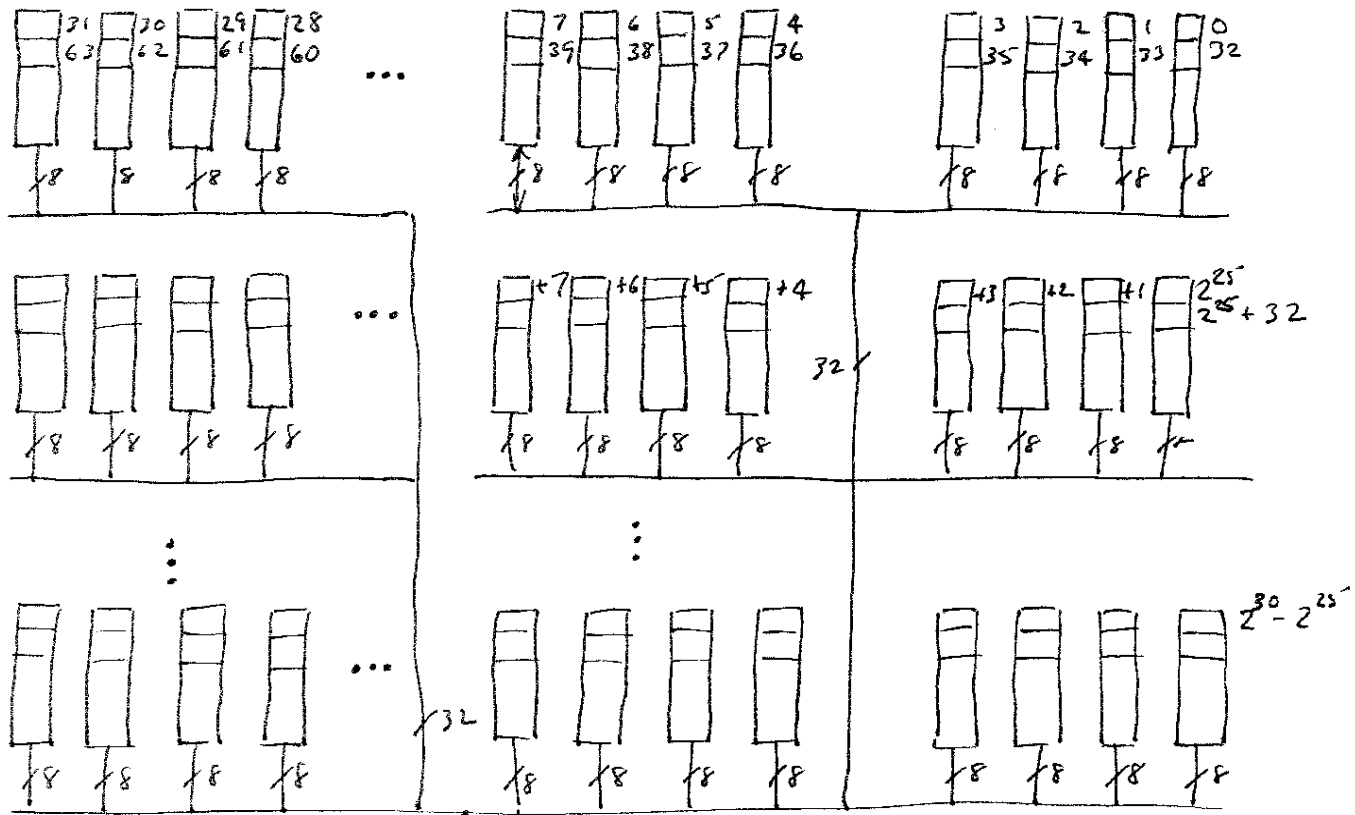
# INTERLEAVING (SHT 1)

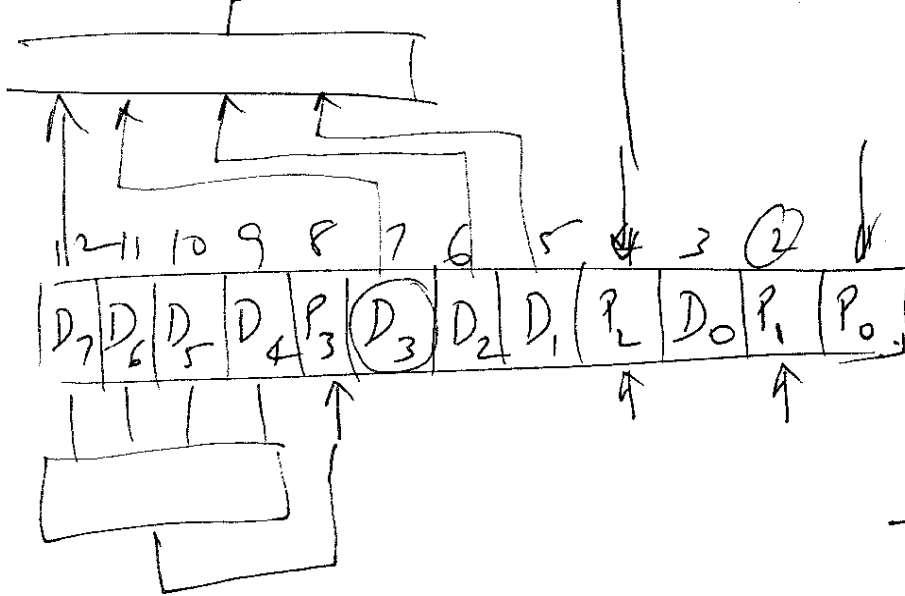


64 BYTES OF MEMORY  
 EACH CHIP :  $2^3 \times 8$  bits  
 BYTE - ADDRESSABLE MEMORY  
 16 BIT BUS  
 TWO-WAY INTERLEAVED



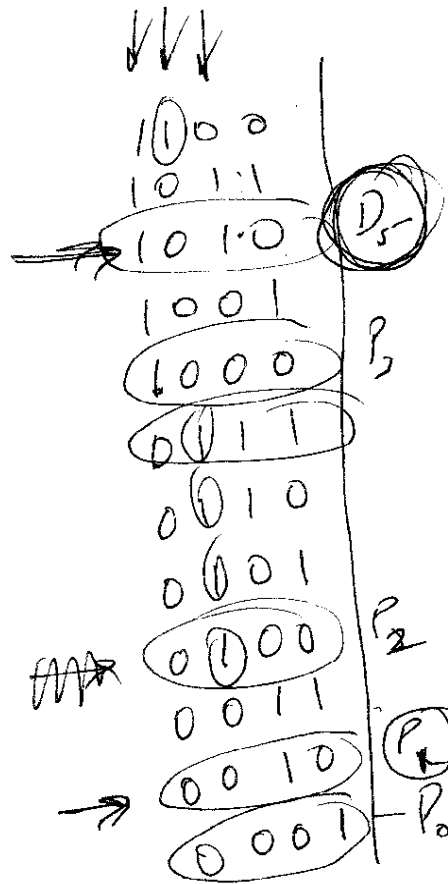
# INTERLEAVING (SHT 2)





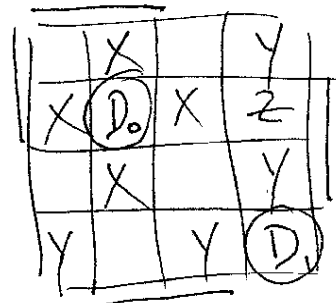
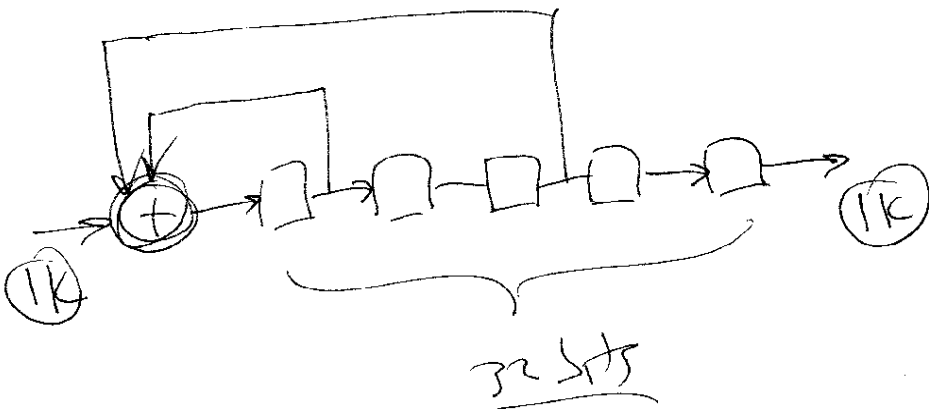
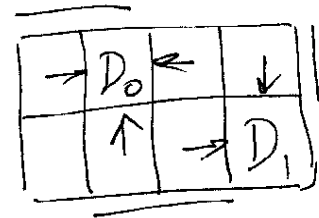
HAMMING

10<sup>-7</sup>  
10<sup>-14</sup>



ENEN

NEEE



## To Deal with Errors in Transmission

I also got email asking if I would include in the handout "Check Sum" or "CRC." So, sure.

① Simplest form. A single parity bit.

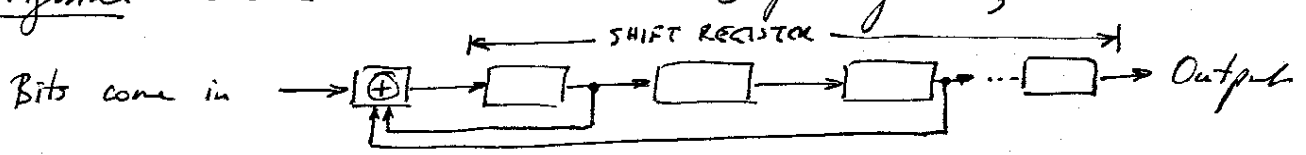
$$P = D_n \oplus D_{n-1} \oplus D_{n-2} \oplus \dots \oplus D_1$$

- \* We transmit  $n+1$  bits -  $n$  Data + 1 Parity.
- \* We can detect if exactly one bit transmitted in error. (What happens if two errors).
- \* Assumes: Each bit is statistically independent with respect to all the others, re: transmitting in error.

② ECC. Hamming Code. When detecting is not enough. We want to be able correct, so we need a code that allows us to identify the "bad bit."  
(The next two pages provide the detail I promised,

③ Check sum or CRC. - Cyclic Redundancy Check.

- \* When the probability of error of the bits are not independent.
- \* When you may get a short sequence clobbered.
- \* Original scheme was to use a Shift register, thus:



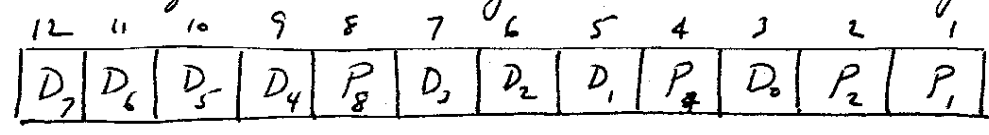
- \* Large no. of bits would come in and leave bit serial. BUT, they would be XOR'ed with selected older bits in the bit stream. After the last bit comes in you still have  $k$  bits as output to the Shift Register. These last bits referred to as CHECK SUM

ECC

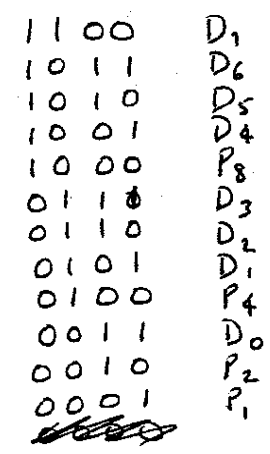
A little research uncovered a still better scheme, still due to Richard Hamming, and still based on XOR (or, parity functions).

Suppose we wish to transmit  $n$  bits. We can correct single errors by adding  $\log_2 n + 1$  bits as follows:

Simplest way to show is by means of an example. Let  $n = 8$ . Then we need  $\log_2 8 + 1 = 4$  parity bits. 12 bits altogether. We lay out the bits as follows



Note the "bit numbers" of each bit



We form a parity function of all bits having a 1 in the corresponding position of ~~each~~ the bit numbers.

$$\therefore P_8 = D_7 \oplus D_6 \oplus D_5 \oplus D_4$$

$$P_4 = D_7 \oplus D_3 \oplus D_2 \oplus D_1$$

$$P_2 = D_6 \oplus D_5 \oplus D_3 \oplus D_2 \oplus D_0$$

$$P_1 = D_6 \oplus D_4 \oplus D_3 \oplus D_1 \oplus D_0$$

# ECC (SHEET 2)

If we lay out with  $x$  the bits provide even parity, we have

	$D_7$	$D_6$	$D_5$	$D_4$	$P_8$	$D_3$	$D_2$	$D_1$	$P_4$	$D_0$	$P_2$	$P_1$
Row 4	x	x	x	x	x							
Row 3	x					x	x	x	x			
Row 2			x	x		x	x			x	x	
Row 1			x		x	x		x		x		x

When we receive the 12-bit value, we check the four even-parity functions. If <sup>(exactly one)</sup> ~~any~~ bit was received incorrectly, all rows having an  $x$  in that column will give an even-parity error message. Since each of the 12 columns provides a unique set of parity errors, we know immediately which bit was transmitted in error.

For example, suppose  $D_2$  is the culprit. We would get odd-parity for row 2 and row 3.  $D_2$  is the only bit that ~~bits~~ would give odd parity for only rows 2,3 if it were transmitted in error.

A nice by-product is that if you examine parity for the four rows (0 = even, 1 = odd), the results tell you if no error occurred 0000, or gives you the "bit number" of the error bit. In the case of  $D_2$ : 0110 which is the "bit number" of  $D_2$  in the 12 bit transmission