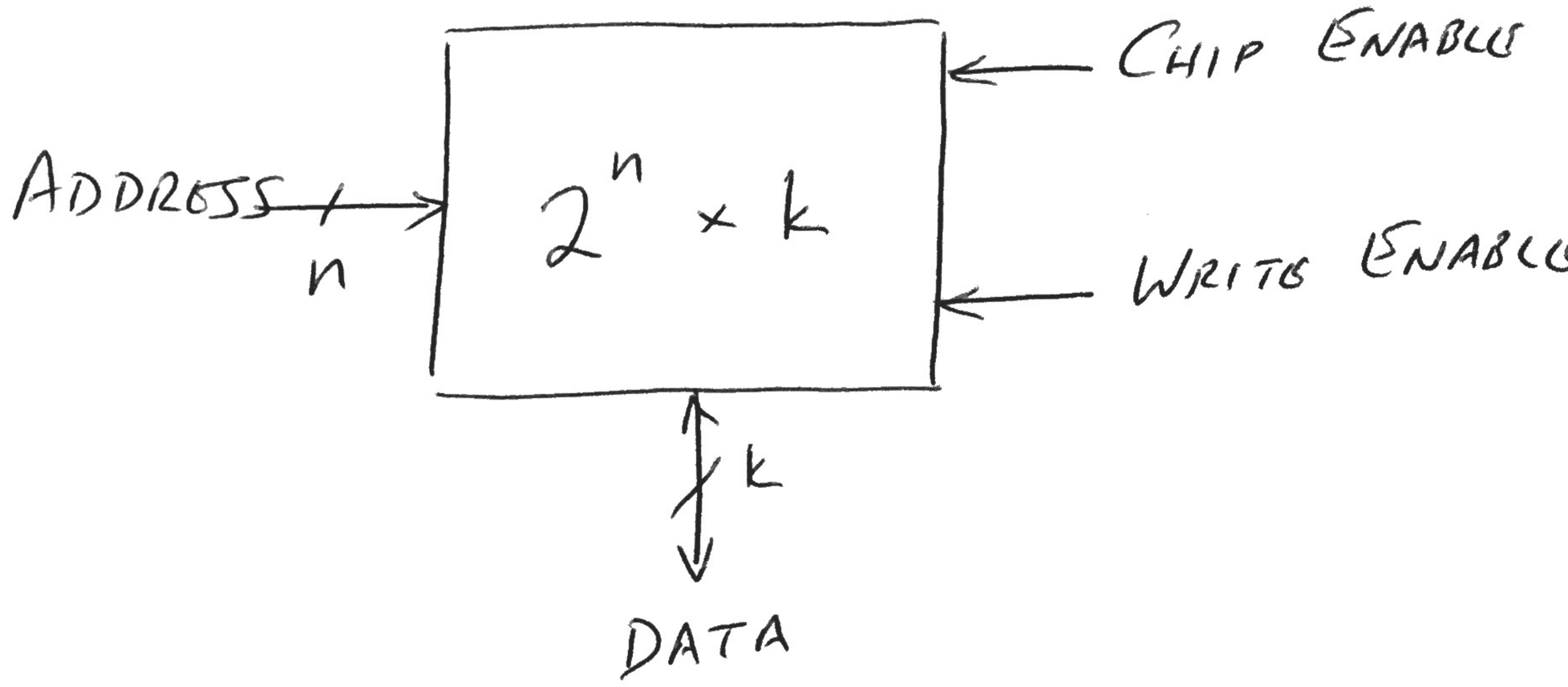
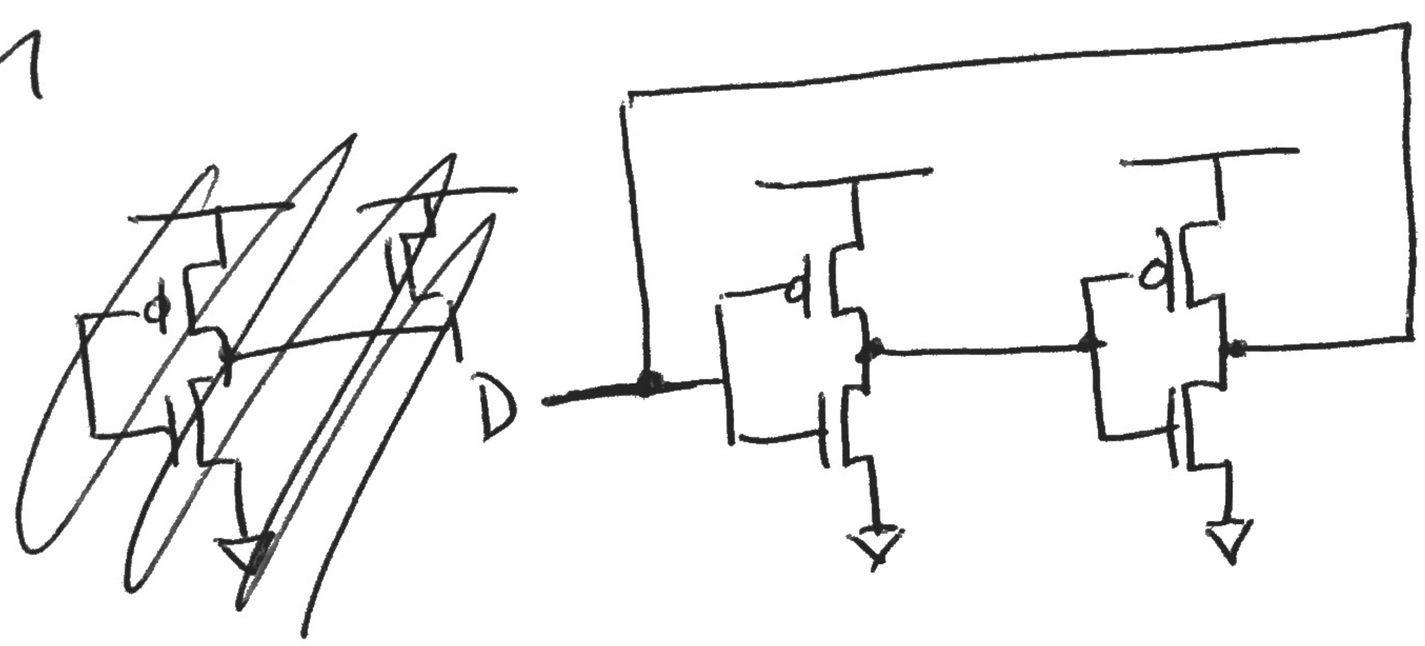
MEMORY BASICS

* A CHIP (BARE BONES)

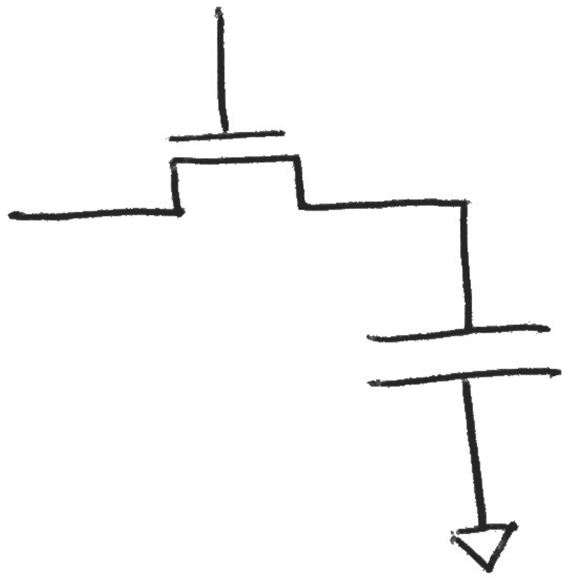


* SRAM / DRAM

SRAM
(4 TRANSISTOR)



DRAM
(1 TRANSISTOR)



CHARGE ON THE CAPACITOR

(REQUIRES REFRESH)

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

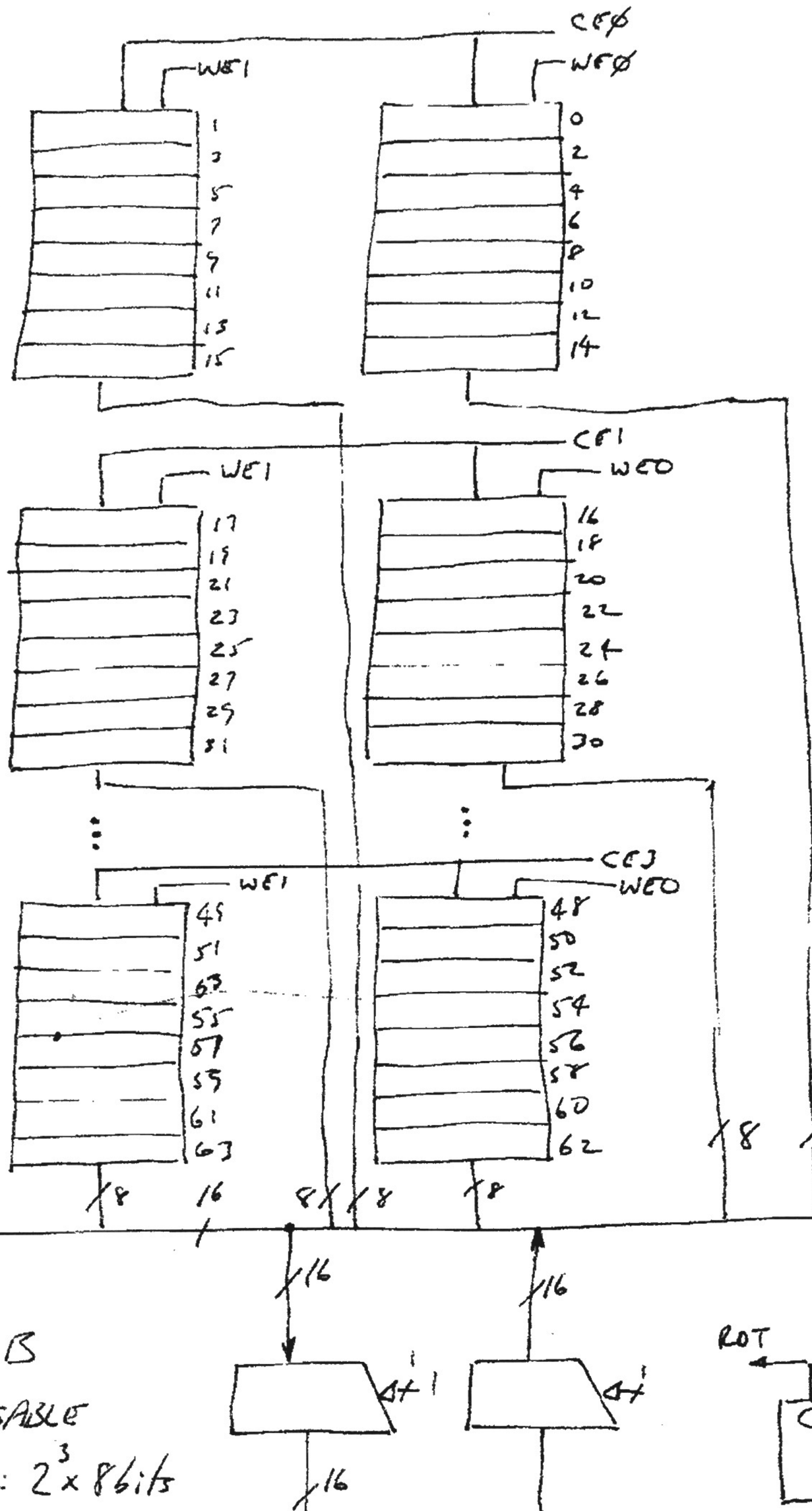
A SIMPLE EXAMPLE TO ILLUSTRATE UNALIGNED ACCESS

UN-ALIGNED ACCESS (NON-INTERLEAVED) (1)

* EIGHT CHIPS
* EACH CHIP STORES 8 BYTES

THAT IS 64 BYTES OF MEMORY

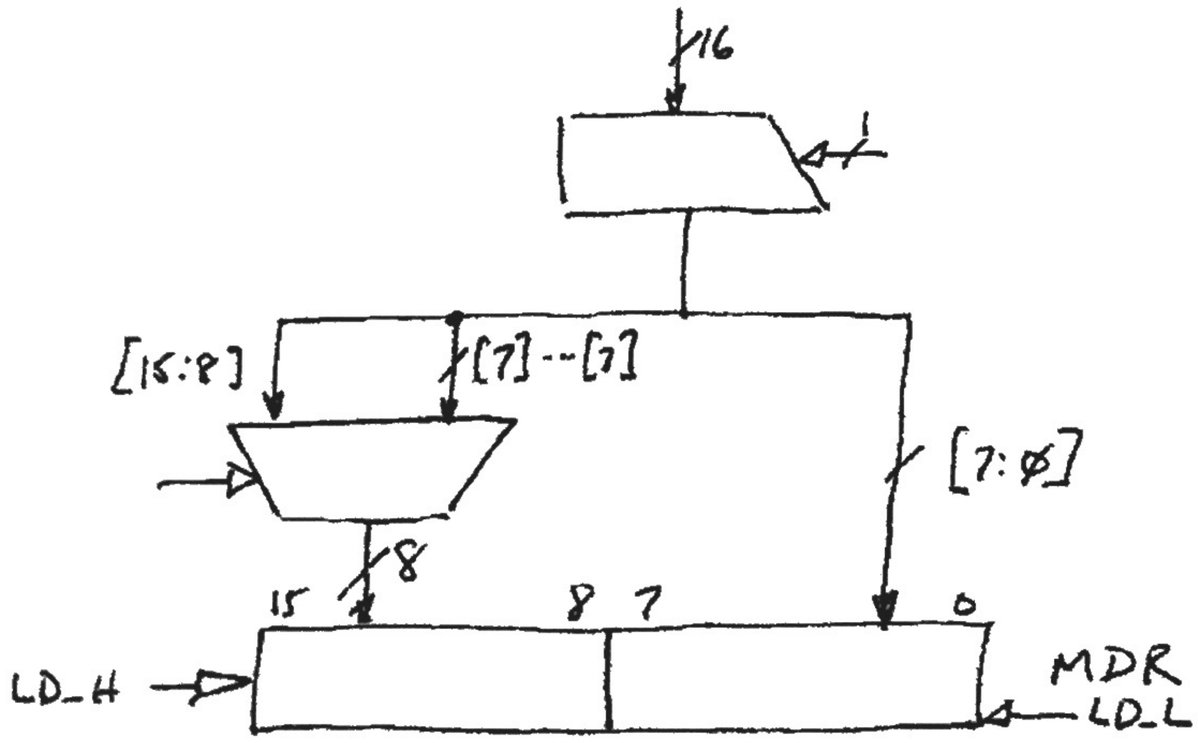
WE WANT TO DO BYTE/WORD, LOAD/STORE, AND ALLOW UNALIGNED ACCESS



UN-ALIGNED ACCESS (2)

R/W	SIZE	MAR[0]	1ST/2ND	SGN	LD_H	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

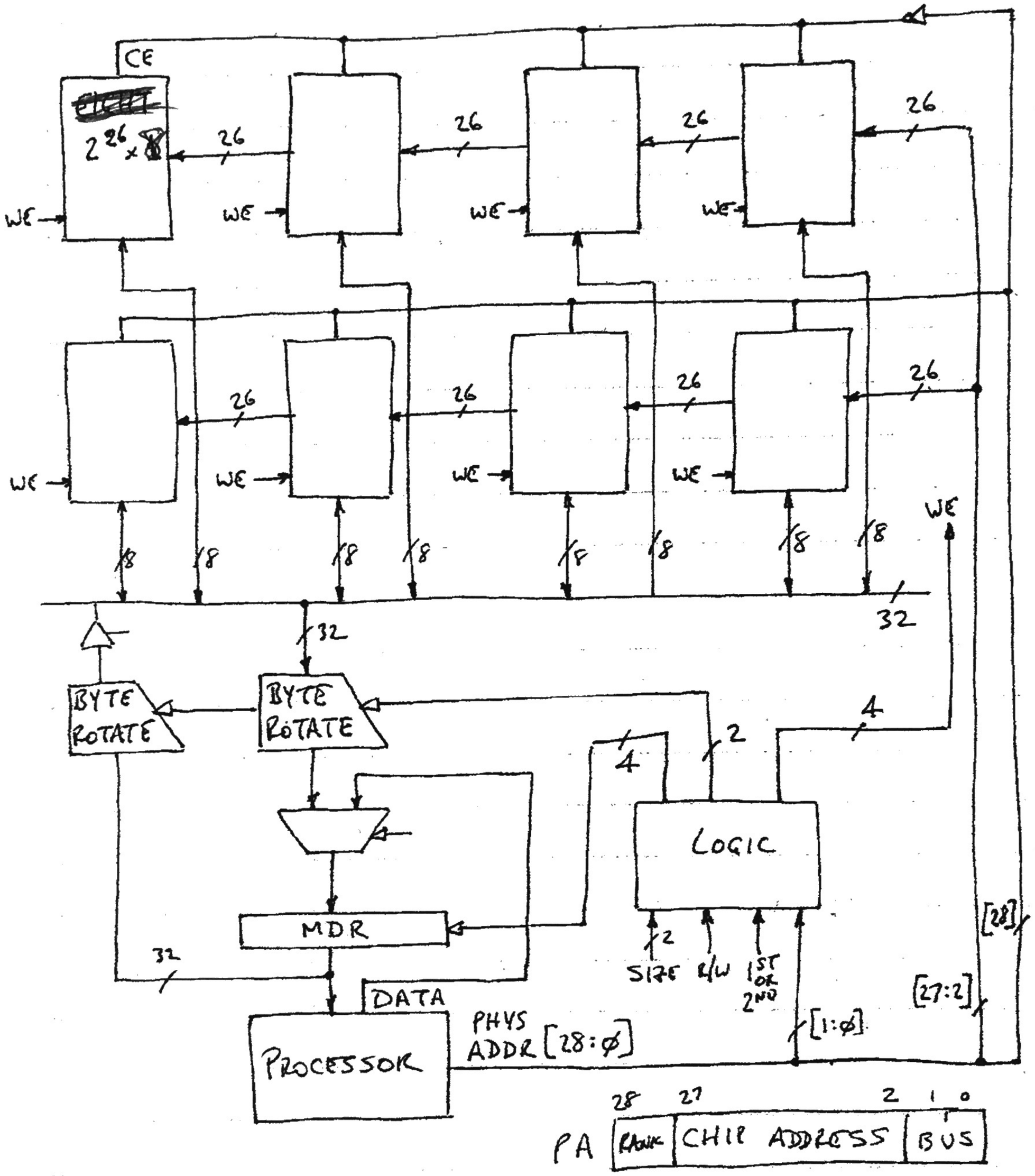
CONTROL SIGNALS NEED TO IMPLEMENT UNALIGNED LD/ST BYTE/WORD



SGN : SIGN-EXTOND

WE1, WE0 : WRITE ENABLE

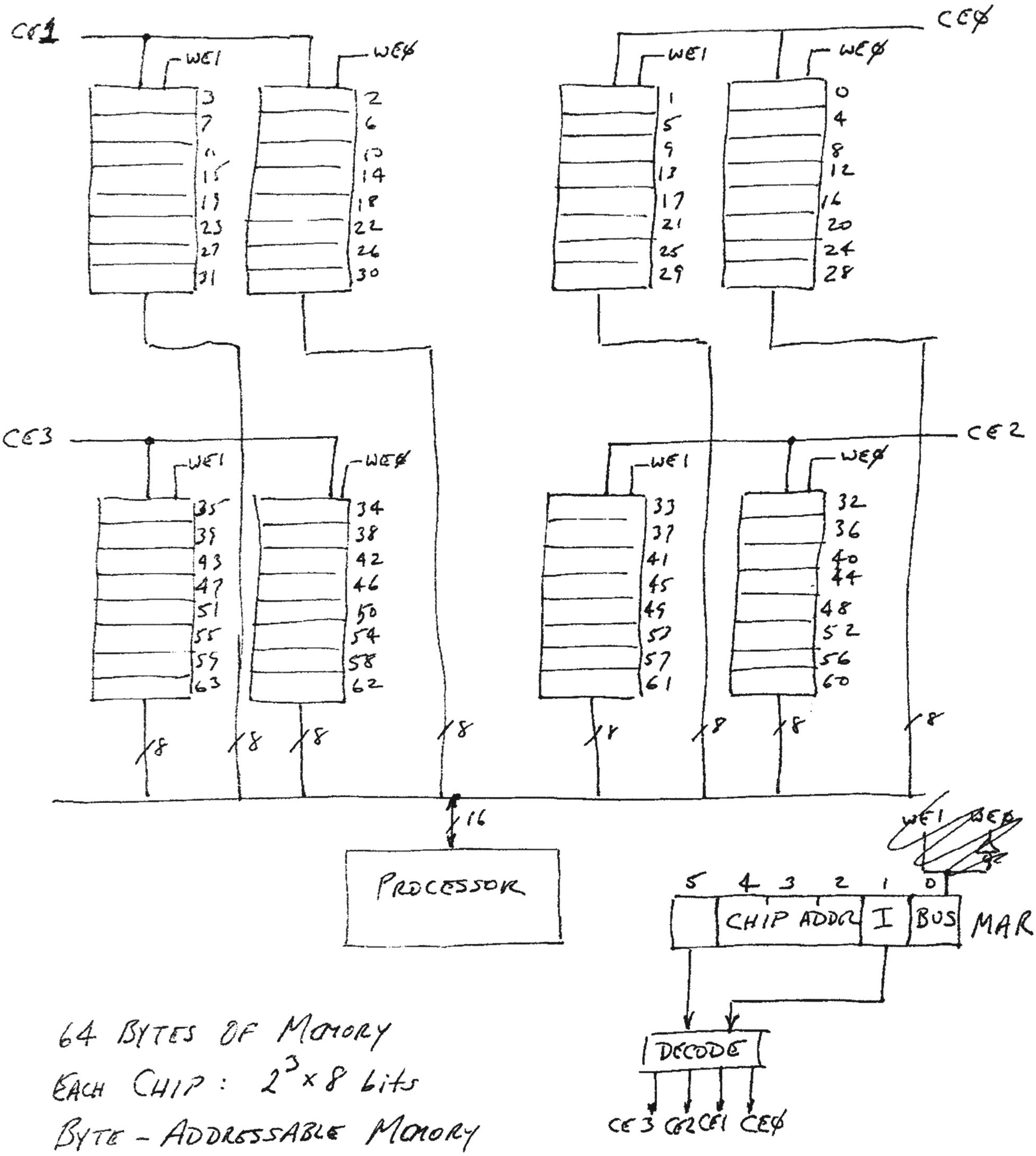
EXTEND UNALIGNED TO 2^{29} BYTE-ADDRESSABLE MEMORY
STILL NO INTERLEAVED
BUS WIDTH: 32 bits



UNALIGNED ACCESSES
(SIMPLIFIED BLOCK DIAGRAM)

A SIMPLE ILLUSTRATION OF INTERLOAVING

INTERLOAVING (SHT 1)



64 BYTES OF MEMORY

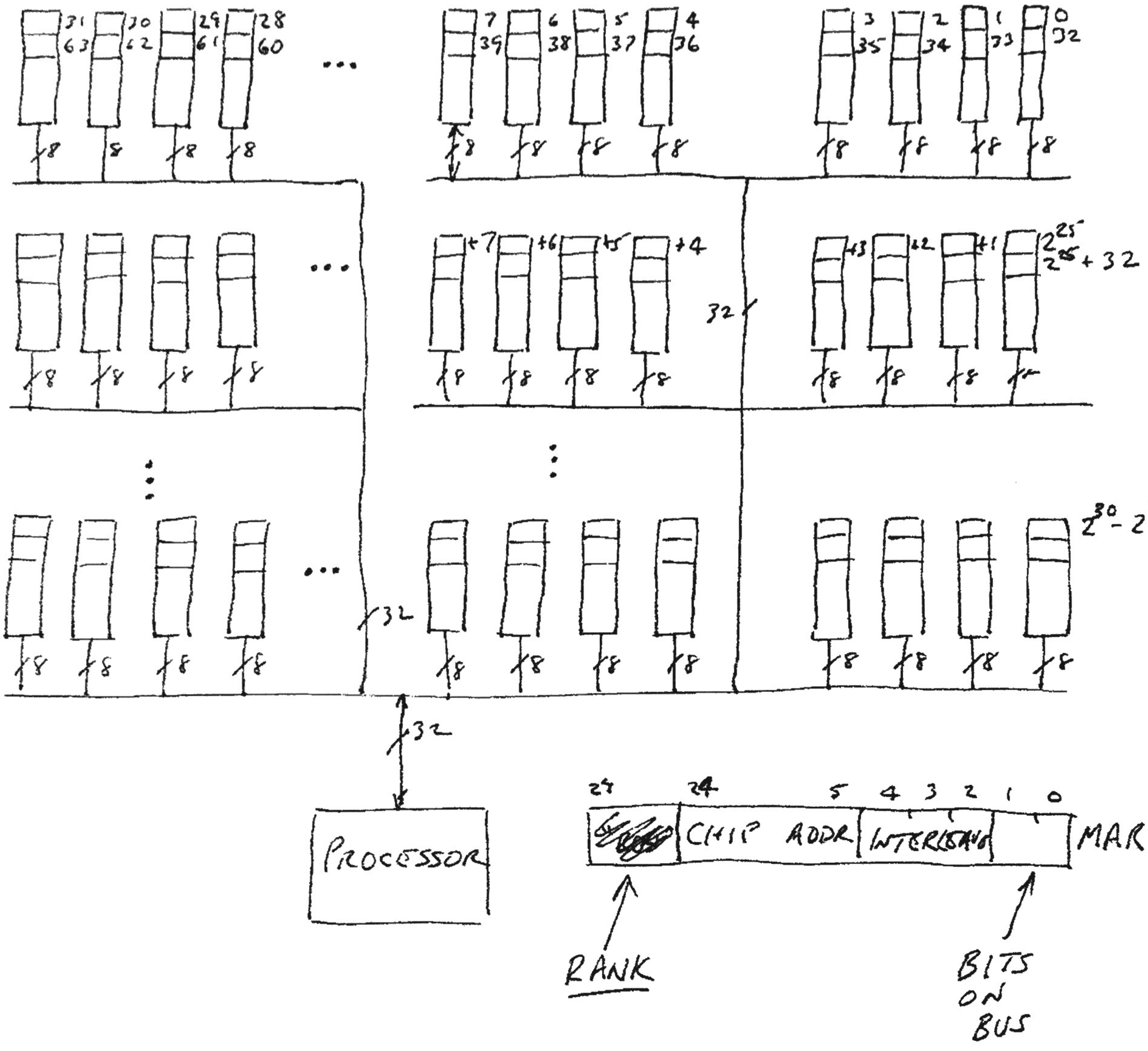
EACH CHIP: $2^3 \times 8$ bits

BYTE - ADDRESSABLE MEMORY

16 BIT BUS

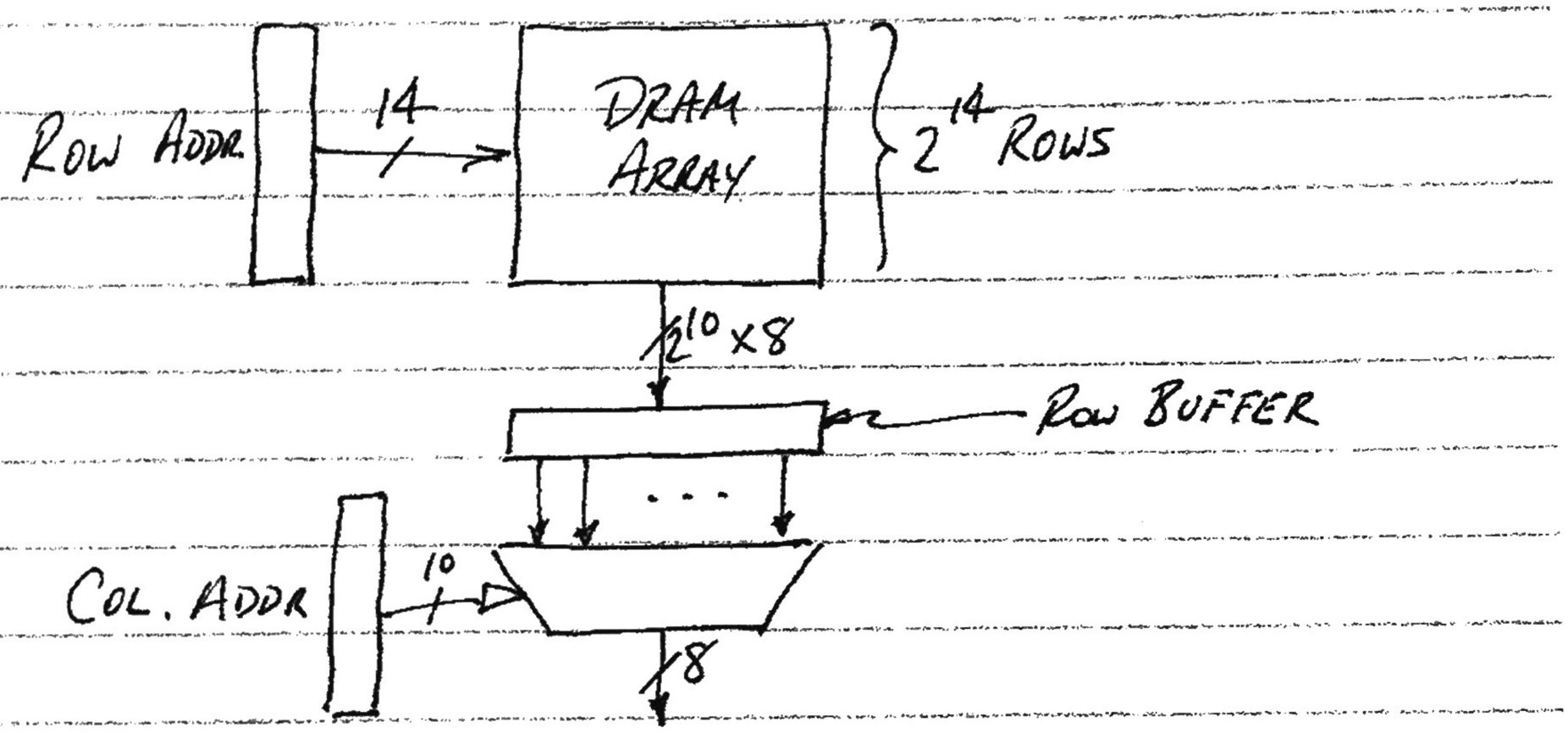
TWO - WAY INTERLEAVED

INTERLEAVING (SHT 2)



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.

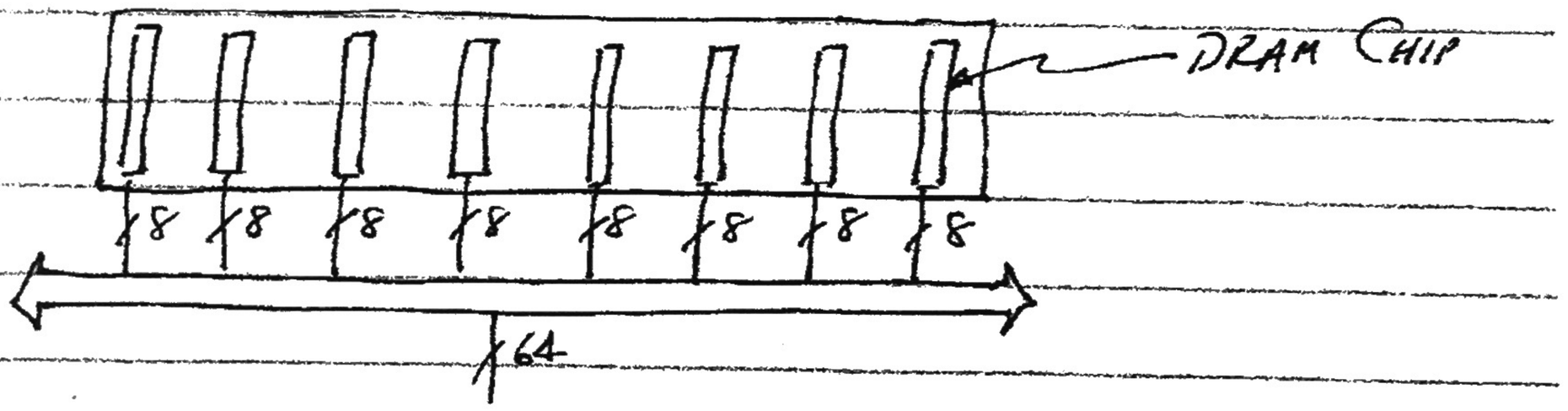
1 GB DRAM (CONTINUED)

* THE DRAM CHIP (CONTINUED)

  - THIS ALLOWS INTERLEAVING THE 8 ARRAYS ON THE CHIP INTO 8-WAY INTERLEAVED MEMORY

  - TOTAL STORAGE ON THE CHIP: 8 × 16 MB (SILLY MARKETING CALLS IT 1 Gbit)
    I PREFER 128 MB SINCE WE HAVE 2^{27} ADDRESSES, EACH OF WHICH STORES ONE BYTE

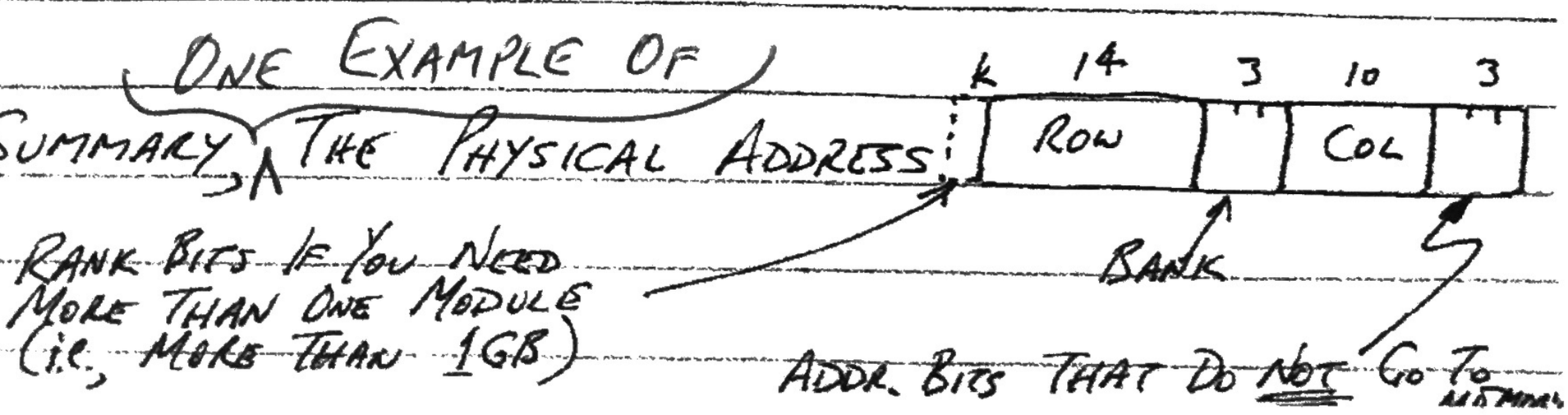
* A TYPICAL SYSTEM CONTAINS A MODULE CONSISTING OF 8 DRAM CHIPS, AS FOLLOWS



  - THIS PROVIDES FOR A 64-bit wide TRANSFER BETWEEN THE PROCESSOR AND MEMORY

  - ADDRESSES TO THE SAME BANK IN ALL CHIPS ARE IDENTICAL

* IN SUMMARY, A THE PHYSICAL ADDRESS (ONE EXAMPLE OF)



To Deal with Errors in Transmission

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

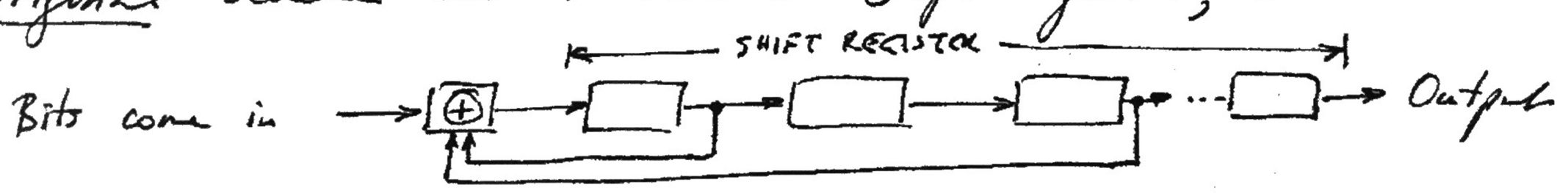
(1.) Simplest form. A single parity bit.

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

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

(2.) ECC. <u>Hamming Code</u>. When <u>detecting</u> is not enough. We want to be able <u>correct</u>, so we need a code that allows us to <u>identify</u> 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 <u>not</u> independent.
* When you may get a short sequence clobbered.
* <u>Original</u> scheme was to use a Shift register, thus:



* Large no. of bits would come in and leave <u>bit serial</u>. BUT, they would be XORed with <u>selected</u> older bits in the bit stream. After the last bit comes in, you still have k bits to output to the SHIFT Register. These last bits referred to as CHECK SUM

ECC

A little research uncovered a still Letter 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

12	11	10	9	8	7	6	5	4	3	2	1
D_7	D_6	D_5	D_4	P_8	D_3	D_2	D_1	P_4	D_0	P_2	P_1

Note the "bit number" of each bit

1100	D_7
1011	D_6
1010	D_5
1001	D_4
1000	P_8
0111	D_3
0110	D_2
0101	D_1
0100	P_4
0011	D_0
0010	P_2
0001	P_1

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

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

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

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

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

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 (exactly one) 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 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