520.492 Mixed-Signal VLSI Systems

*Week 3*

# Memory and Arithmetic

### References

1. Geiger, Allen and Strader: pp. 821-866.
2. Weste and Eshraghian (2nd Ed.), Chapter 8.

# MEMORY AND ARITHMETIC

ref: Weste & Eshraghian, Chapter 8
Geiger, Allen & Strader, pp 821-866

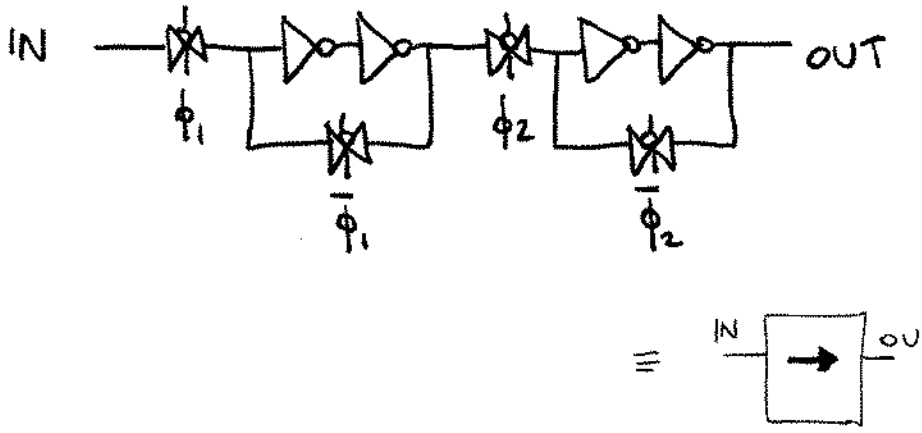
## MEMORIES (and the like)

- Registers
  - Shift registers
  - Stack registers
- Random Access Memories (RAM)
  - Static (sRAM)
  - Dynamic (dRAM)
- Read-Only Memories (ROM)
  - Hardwired
  - Programmable / Erasable (EPROM; $E^2$PROM)
- Programmable Logical Arrays (PLA)

## ARITHMETIC

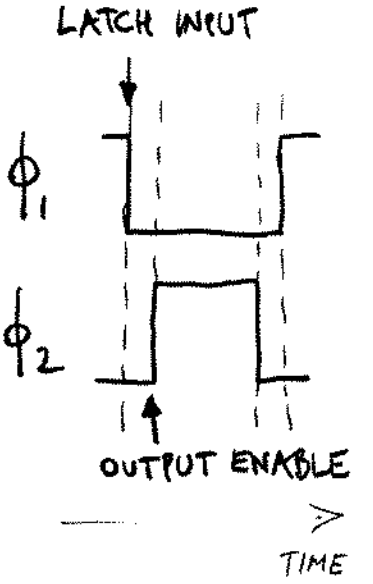
- Addition and Subtraction
  - Bit-Parallel Adders (carry look-ahead, Manchester, ...)
  - Bit-Serial Adders
- Multiplication
  - Array Multiplication (Booth recoding, ...)
  - Wallace Tree Multipliers
  - Serial Multipliers

# REGISTERS

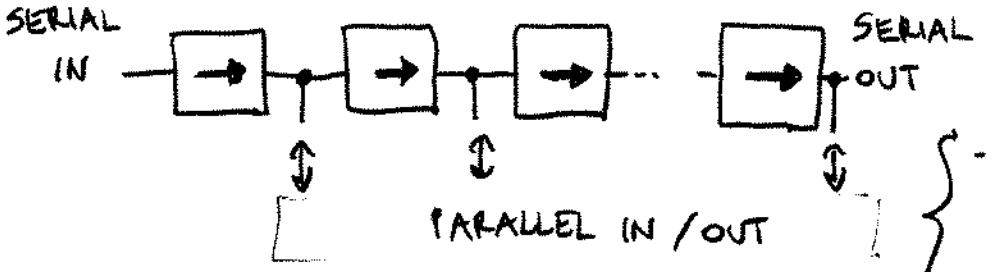






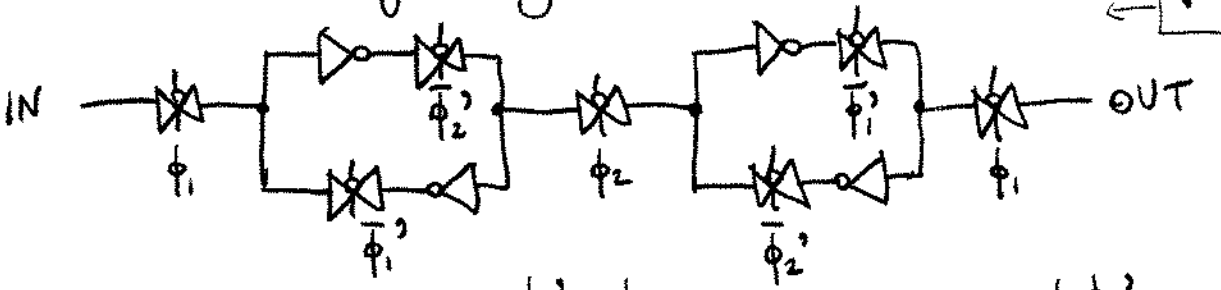



- **Shift Register:**




- parallel in / serial out
- serial in / parallel out

Bidirectional Shift Register Cell:

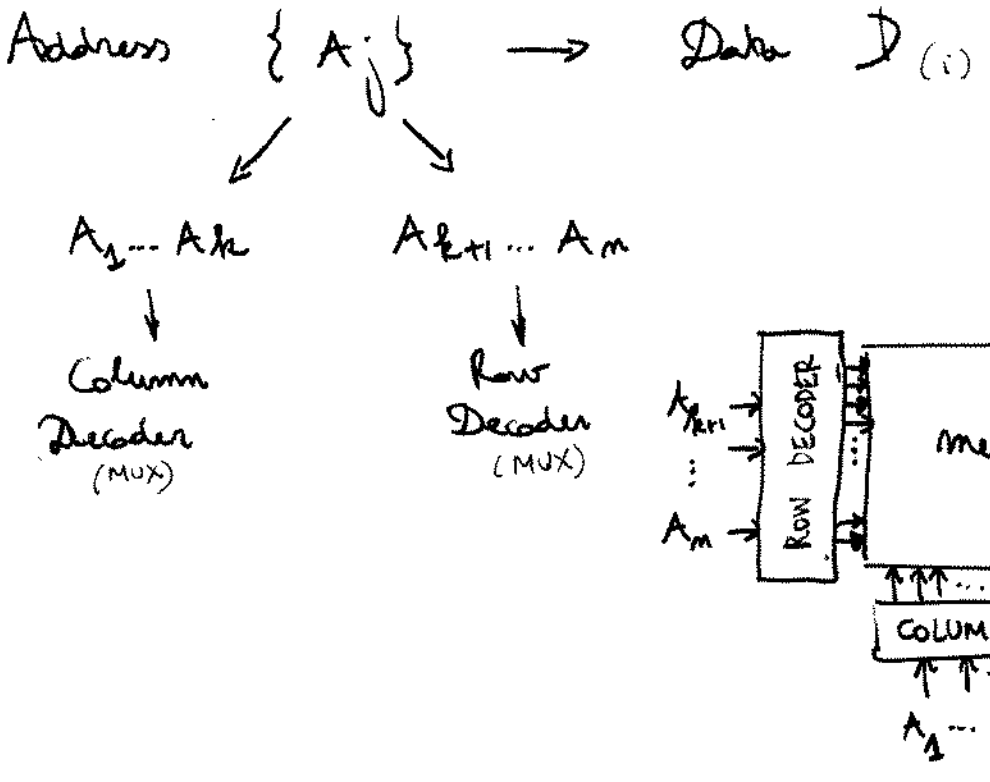




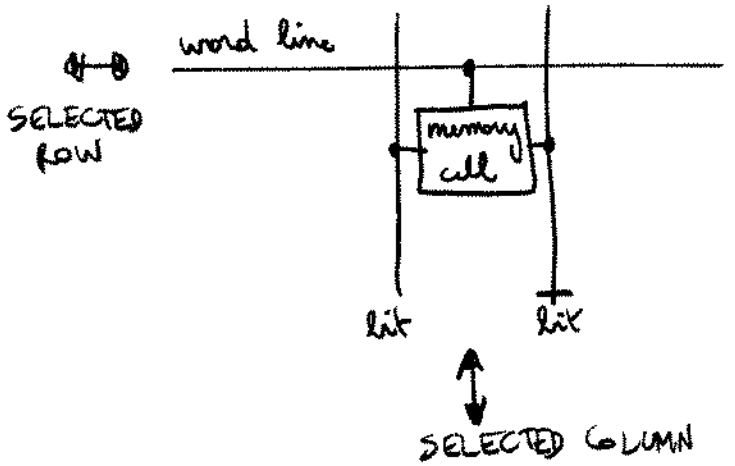
SHIFT RIGHT: $\begin{cases} \phi_1' = \phi_1 \\ \phi_2' \equiv 0 \end{cases}$ ; SHIFT LEFT: $\begin{cases} \phi_1' \equiv 0 \\ \phi_2' = \phi_2 \end{cases}$

- **Stack Register:**

# ADDRESS - ENCODED MEMORIES

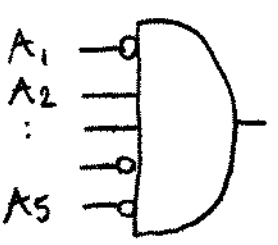



- row/column configuration in memory:

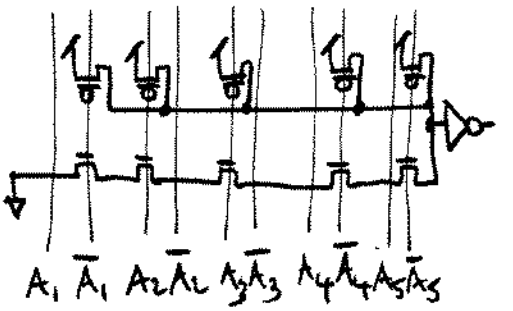



- decoder:

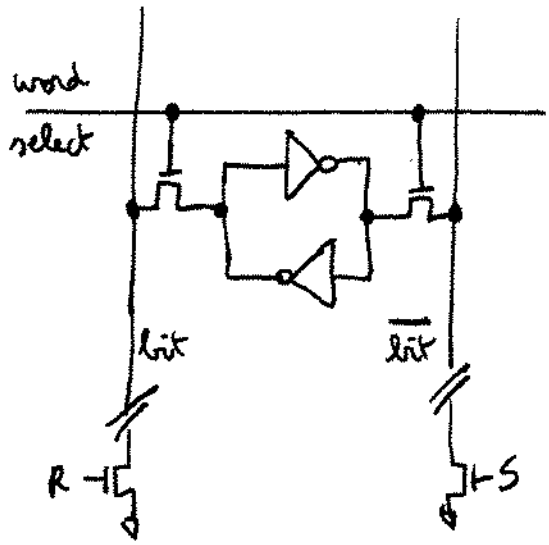
say, "01100":



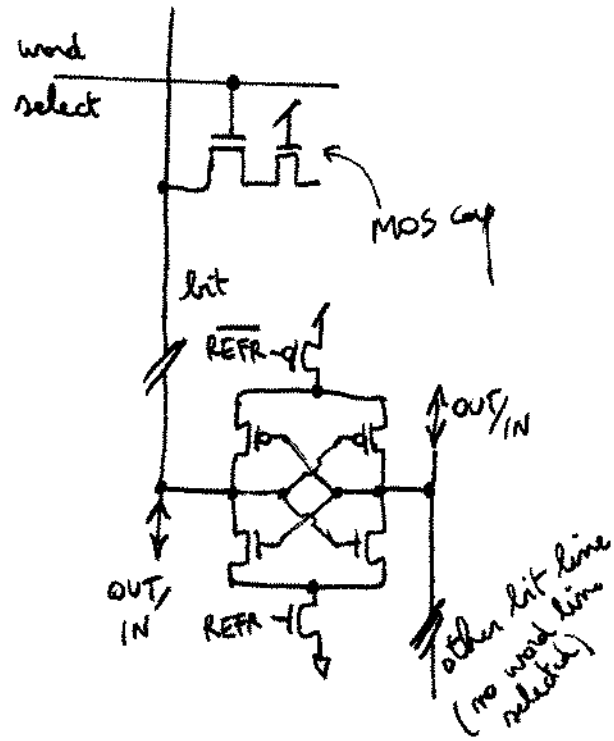

⇒



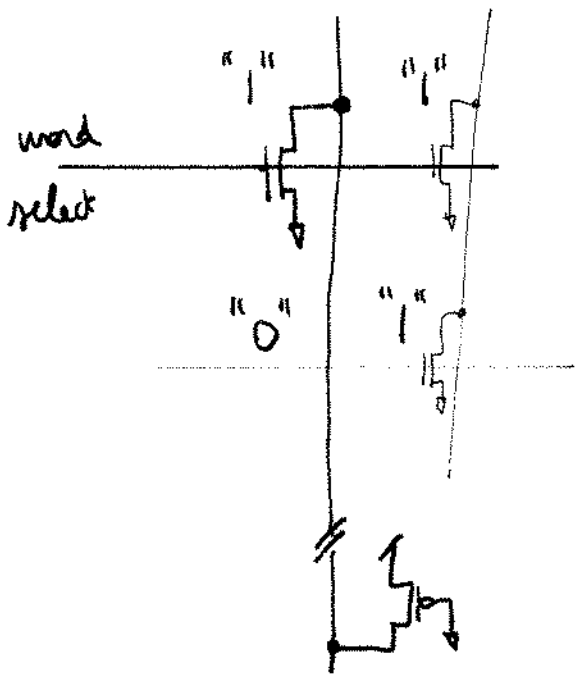
## S-RAM CELL:

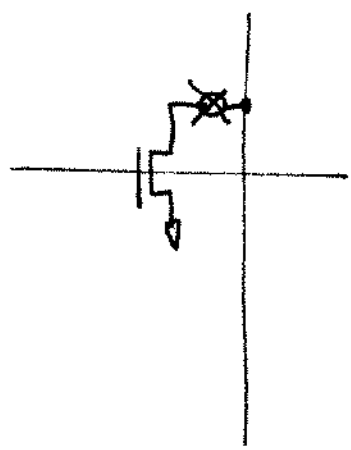



## D-RAM CELL:

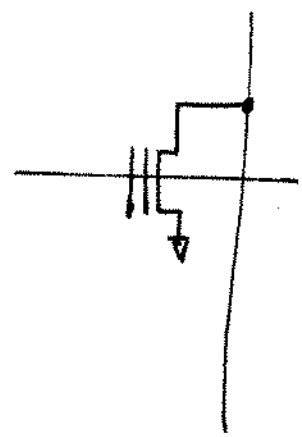



## (P) ROM CELL:



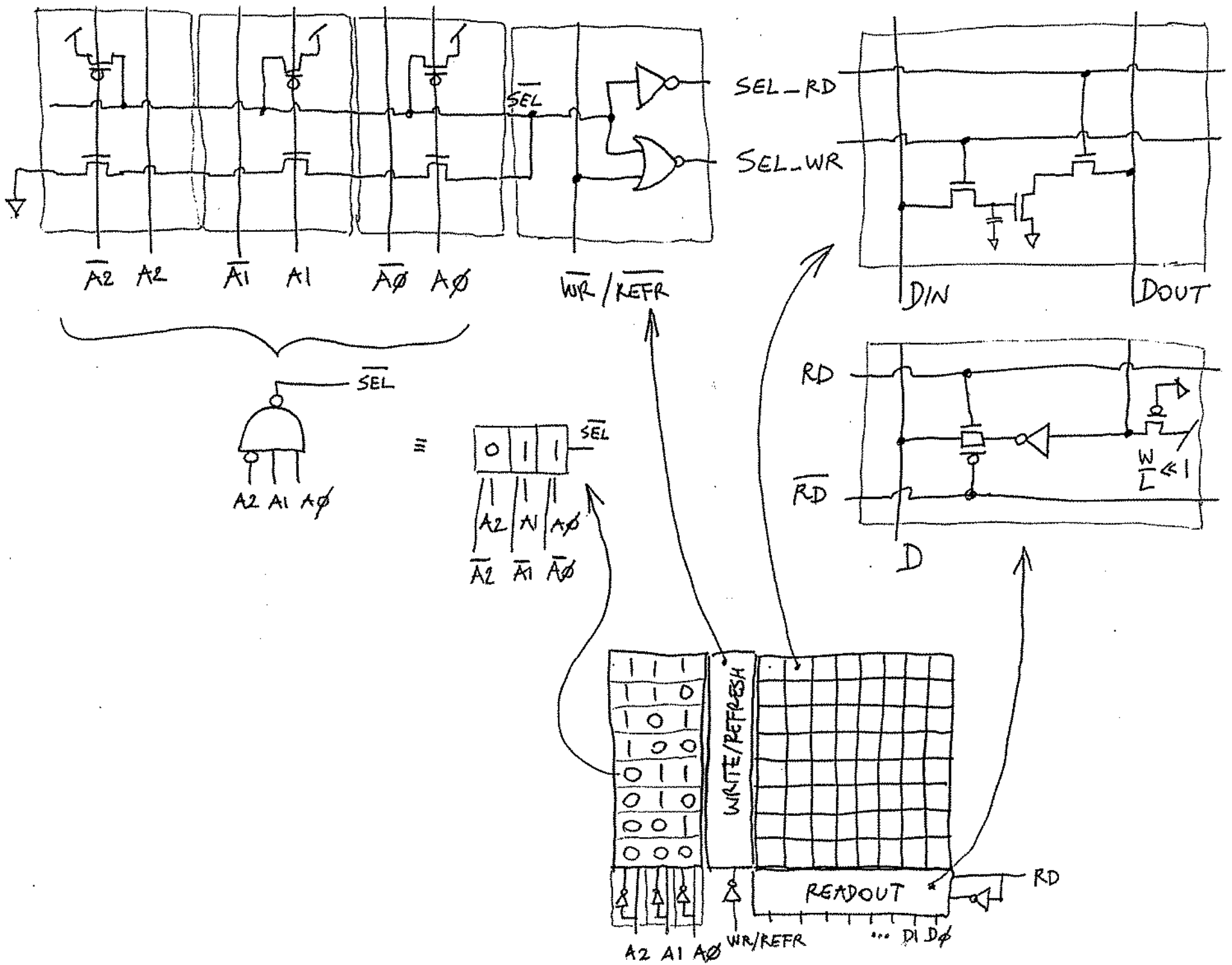



PROM



EPROM
$E^2$PROM

# 3T DRAM


SEL
A2 A2 A1 A1 AØ AØ
WR/REFR
SEL_RD
SEL_WR
DIN
DOUT
RD
RD
W/L ≪1
D
SEL
A2 A1 AØ
≡
SEL
A2 A1 AØ
A2 A1 AØ
WRITE/REFRESH
READOUT
RD
A2 A1 AØ
WR/REFR
... D1 DØ

# PROGRAMMABLE LOGICAL ARRAYS (PLA)

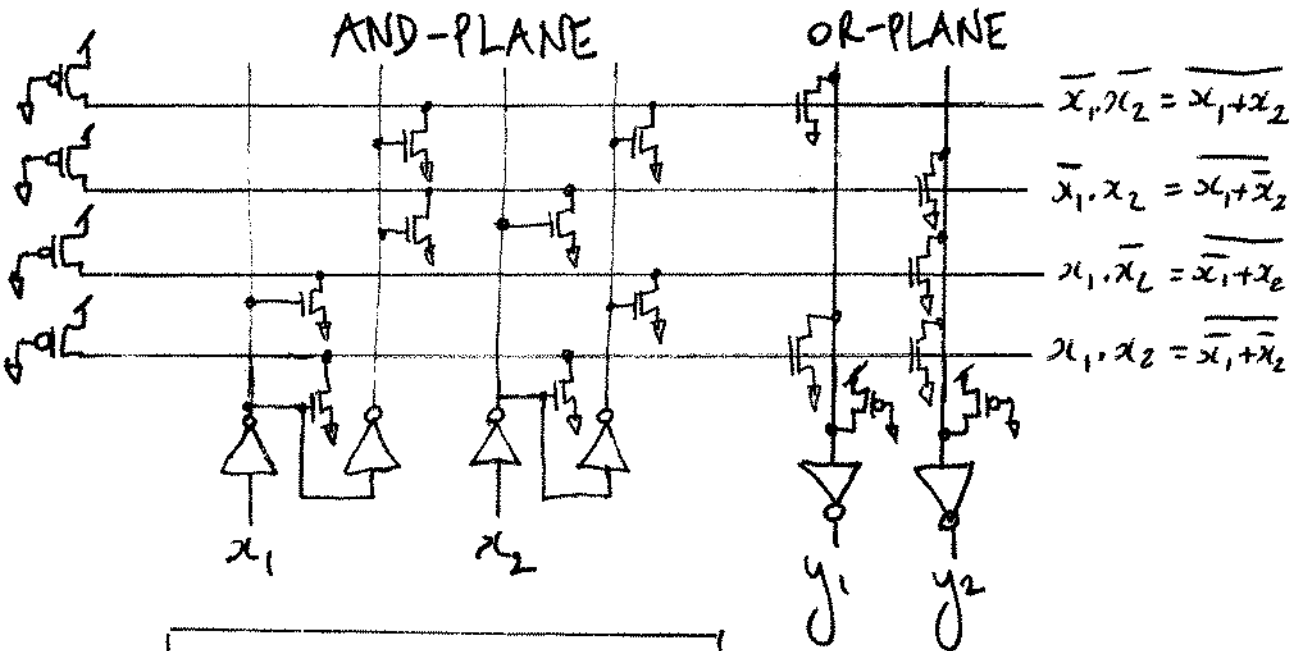
$$Y = f(X)$$

as a sum (canonical sum) of product combinations of $x_j$ and $\bar{x}_j$ terms

e.g.:

| X | | Y | |
|---|---|---|---|
| $x_1$ | $x_2$ | $y_1$ | $y_2$ |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$\Rightarrow \begin{cases} y_1 = \bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot x_2 \\ y_2 = \bar{x}_1 \cdot x_2 + x_1 \cdot \bar{x}_2 + x_1 \cdot x_2 \end{cases}$$




Extensions: $Y_i = f(X_i, Y_{i-1})$ sequential logic

finite state machines

PROS: structured logic
scalable, fast turnaround

CONS: efficiency low (size, power, ... vs. minimalist approach)

## ADDITION (unsigned)

$$\left.\begin{aligned} A &= \sum_{i=0}^{n-1} A_i\, 2^i \\ B &= \sum_{i=0}^{n-1} B_i\, 2^i \end{aligned}\right\} \quad S = A + B = \sum_{i=0}^{n-1} (A_i + B_i)\, 2^i = \sum_{i=0}^{n-1} S_i\, 2^i$$

- $S_i \neq A_i + B_i$ : problem $\begin{cases} A_i = 1 \\ B_i = 1 \end{cases}$
- if bit-sum $\geq 2 \Rightarrow$ subtract 2 and carry a 1 to the next bit level $(i+1)$

$$\begin{cases} S_i = (A_i + B_i + C_i) \bmod 2 \\ C_{i+1} = (A_i + B_i + C_i) \operatorname{div} 2 \end{cases} \qquad (C_0 = 0)$$

(arithmetic)

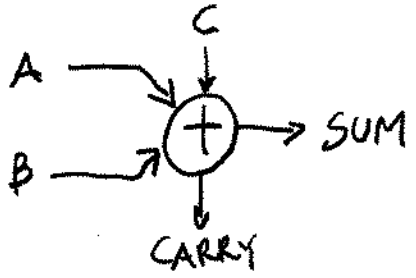
| $A_i$ | $B_i$ | $C_i$ | $(A_i+B_i+C_i)$ | $S_i$ | $C_{i+1}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 2 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 0 | 1 |
| 1 | 1 | 0 | 2 | 0 | 1 |
| 1 | 1 | 1 | 3 | 1 | 1 |

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$$

(logic)

FULL ADDER :

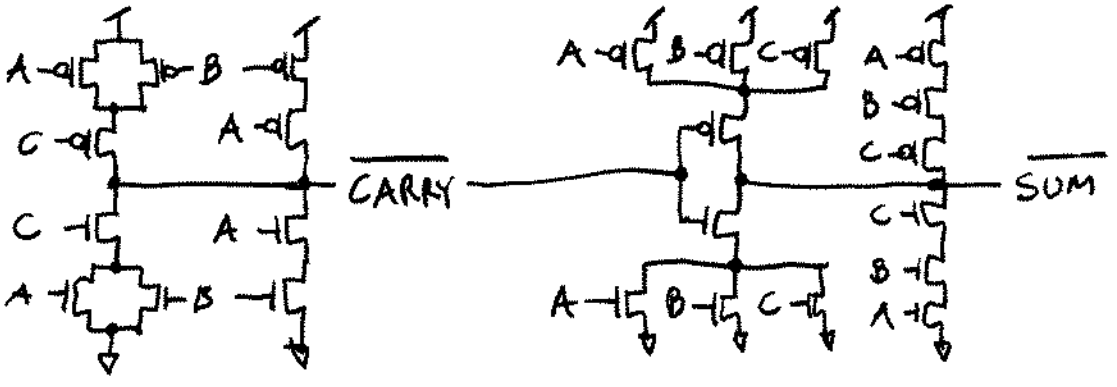



(1) $CARRY = A.B + C.(A+B)$

generate (G) propagate (P)

$SUM = ABC + A\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} =$

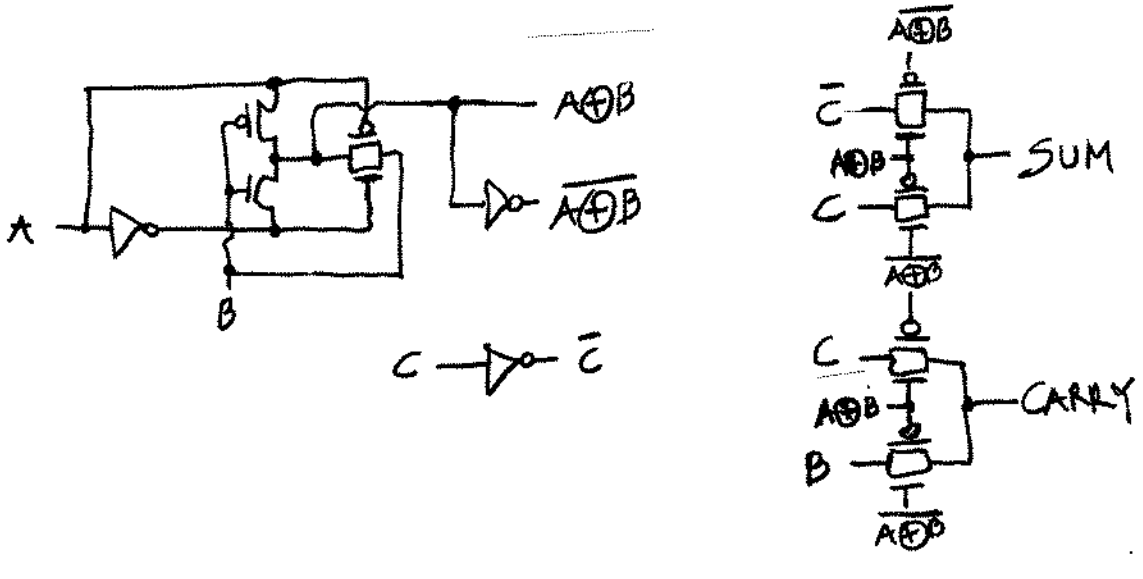
$A.B.C + (A+B+C).\overline{CARRY}$




(2) $CARRY = (A \oplus B).C + \overline{(A \oplus B)}.B$

$SUM = (A \oplus B).\bar{C} + \overline{(A \oplus B)}.C$

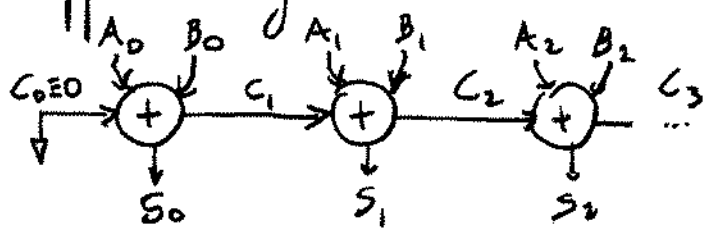
where $A \oplus B \equiv A.\bar{B} + \bar{A}.B$



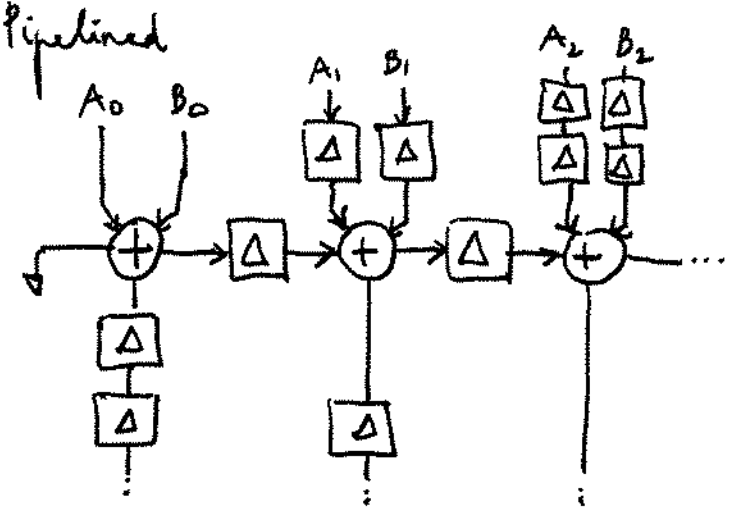
# CONFIGURATION

## Bit-Parallel:

- Ripple Carry

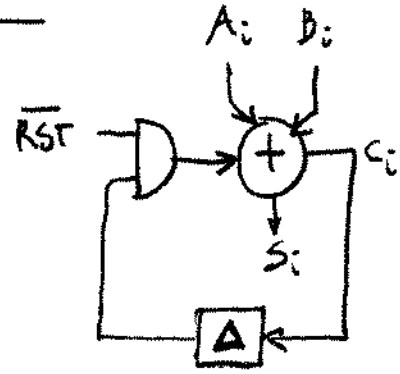



- Pipelined




- Carry Look-Ahead (Manchester); Carry Select; Conditional Sum ...
  → techniques to reduce critical carry path propagation delay

## Bit-Serial:



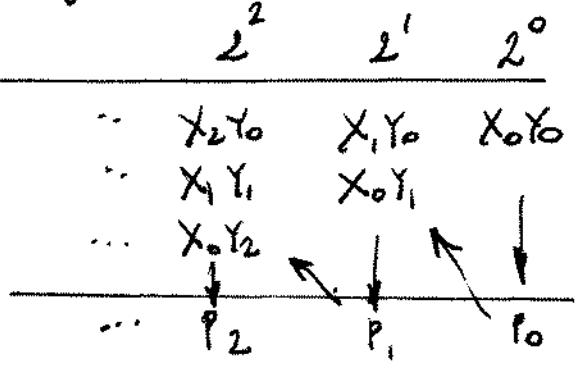
# MULTIPLICATION

$$X = \sum_{i=0}^{n-1} X_i 2^i \qquad Y = \sum_{j=0}^{m-1} Y_j 2^j$$

$$P = X \cdot Y = \sum_{i=0}^{n-1} X_i 2^i \cdot \sum_{j=0}^{m-1} Y_j 2^j =$$

$$\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (X_i Y_j) 2^{i+j} = \sum_{k=0}^{n+m-2} P_k 2^k \qquad (k = i+j)$$

Again, $P_k \neq \sum_{j=0}^{m-1} (X_{k-j} \cdot Y_j)$



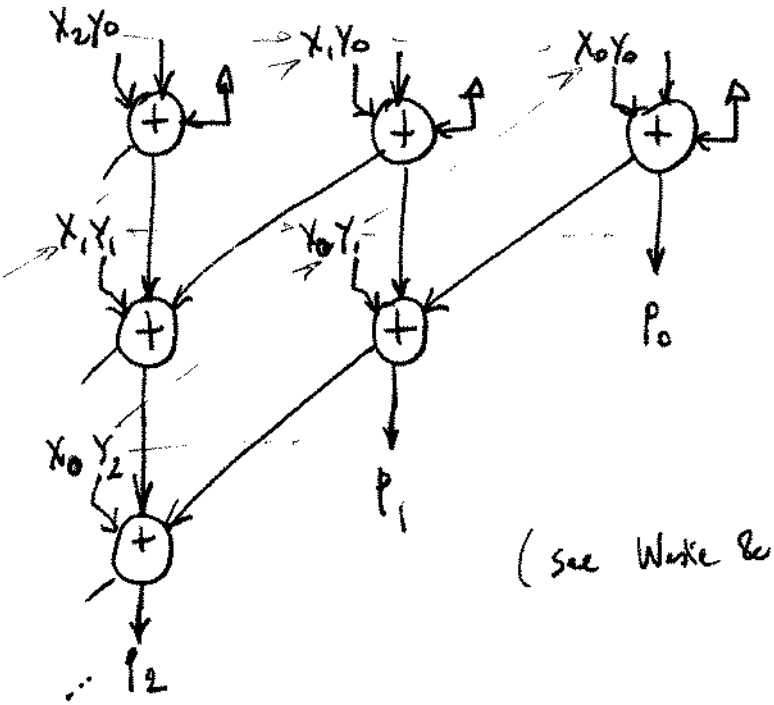

- Parallel Summation of $(X_{k-j} \cdot Y_j)$ contributions:
  - Linear parallel implementation ( $O(n)$ depth )
    - array multiplier
    - Booth redecoded multiplier
  - Tree structure implementation ( $O(\log_2 n$ depth)
    - Wallace tree multiplier
- Serial Summation

  Serial / Parallel multiplier

## Parallel Summation of $X_{kj} \cdot Y_j$ product terms:

| | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|
| | $X_2 Y_0$ | $X_1 Y_0$ | $X_0 Y_0$ |
| ... | $X_1 Y_1$ | $X_0 Y_1$ | |
| | $X_0 Y_2$ | | |
| ... | $P_2$ | $P_1$ | $P_0$ |

- array:




(see Weste & Eshraghian, Fig. 8.37)

- Booth recoding
- Tree structure (linear, $\log_2(n)$ latency)

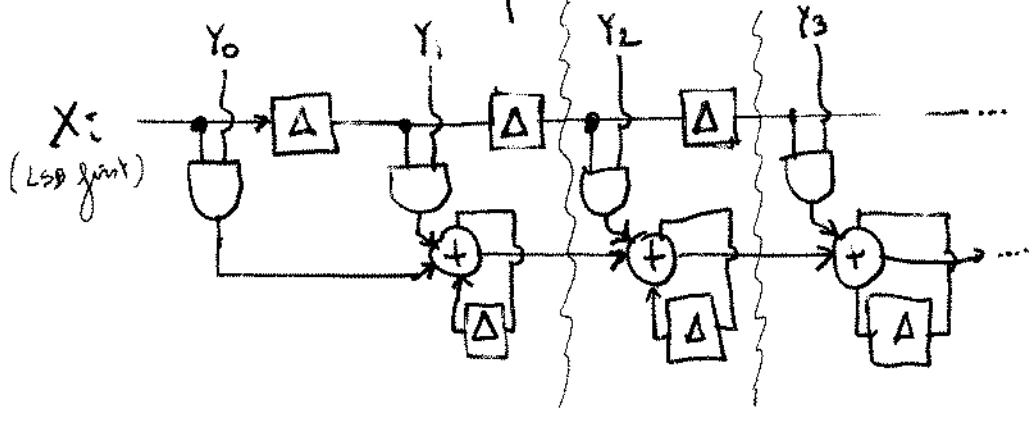
## Serial Summation of $X_{k-j} \cdot Y_j$ product terms:

$$\sum_i P_i 2^i = \sum_i \left( \sum_j X_{i-j} Y_j \right) 2^i$$

$X_{i-j}$: SHIFT, $Y_j$: FIXED

| | $2^2$ | $2^1$ | $2^0$ | |
|---|---|---|---|---|
| | $X_2Y_0$ | $X_1Y_0$ | $X_0Y_0$ | → $i=0$ |
| ... | $X_1Y_1$ | $X_0Y_1$ | | → $i=1$ |
| | $X_0Y_2$ | | | → $i=2$ |
| ... | $P_2$ | $P_1$ | $P_0$ | |

- Serial/Parallel Multiplier




- Serial/Parallel Pipelined Multiplier