Memory design

Top-level organization of a memory

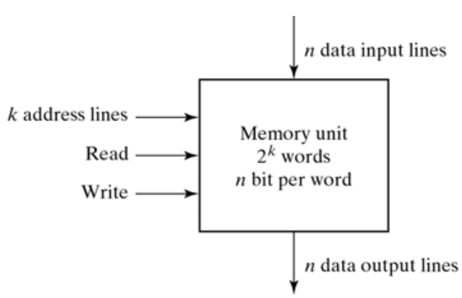


Fig. 7-2 Block Diagram of a Memory Unit

Example of 1K x 16 memory addressing and contents

Memory address: Binary	Memory address: decimal	Memory contest
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

Fig. 7-3 Content of a 1024 × 16 Memory

Some Types

- Random Access Memory (RAM)
  - Data can Read and Written
- Read-Only Memory (ROM)
  - In normal operation, only reading is possible
  - Writing takes a special kind of operation

- RAM technologies:
  - Static ("SRAM")
    - Holds stored data as long as power is applied
    - Fast
    - Expensive, high power, low density
  - Dynamic ("DRAM")
    - Stored data fades with time – must be refreshed every few milliseconds
    - Slow (comparatively)
    - Cheap, low power, high density

Typical memory timing

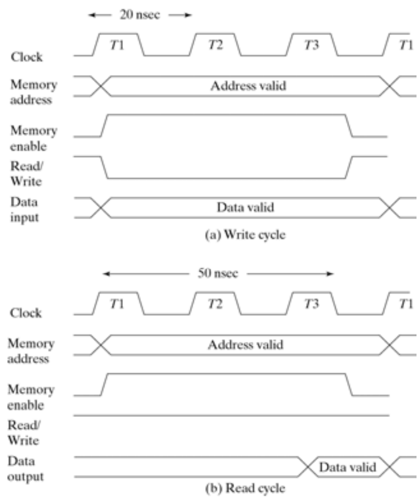


Fig. 7-4 Memory Cycle Timing Waveforms

The University Of Alabama in Huntsville Computer Science

Static Memory cell (using S-R FF)

cs309

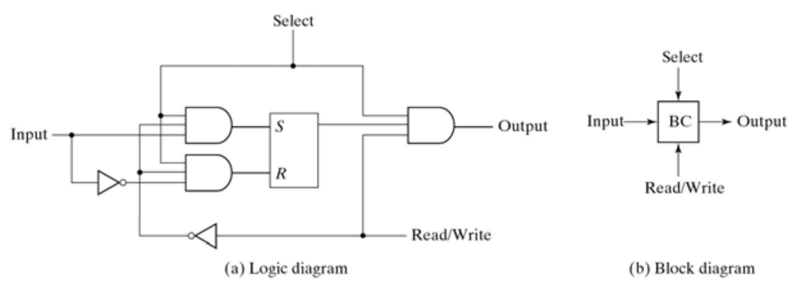


Fig. 7-5 Memory Cell

Combining cells to form a RAM

The University Of Alabama in Huntsville Computer Science

cs309

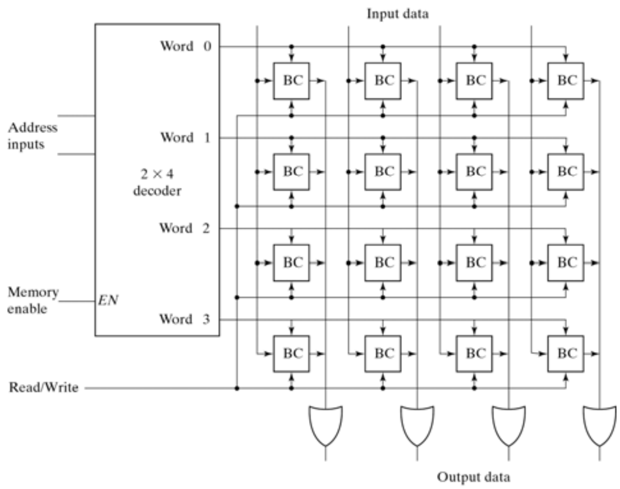


Fig. 7-6 Diagram of a 4 × 4 RAM

Reading

The University Of Alabama in Huntsville Computer Science

cs309

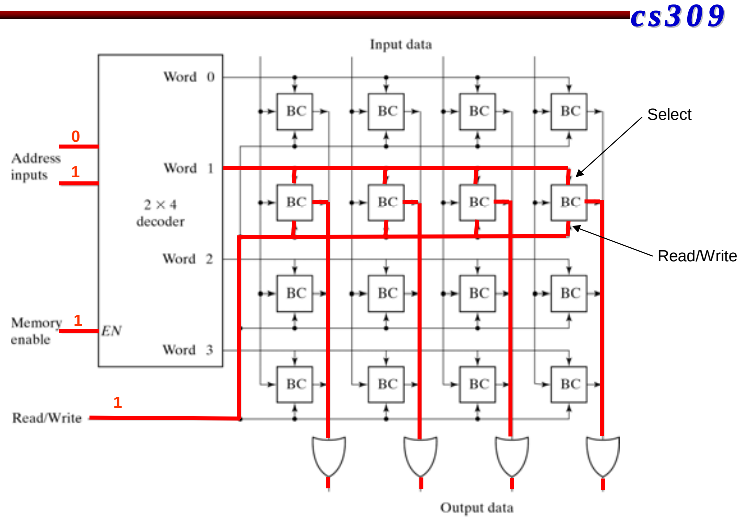


Fig. 7-6 Diagram of a 4 × 4 RAM

A decoding problem

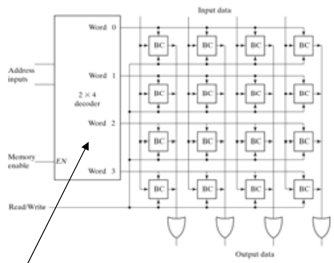


Fig. 7-6 Diagram of a 4 × 4 RAM

- A problem:
  - A k x 2^k decoder requires 2^k, k-input AND gates
  - As memory size grows, the number and size of the AND gates explodes

2D decoding

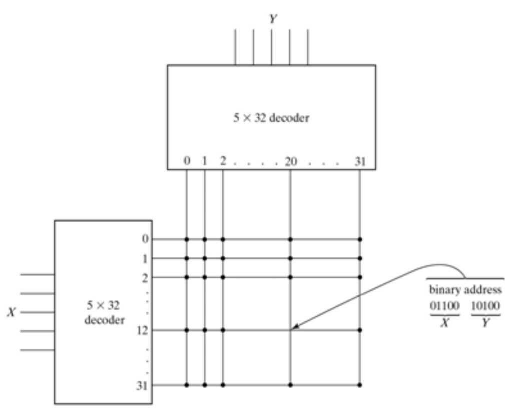


Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

- Add 2nd enable line to cells
- Instead of a k x 2^k decoder, use two k/2 x $2^{k/2}$
  - k x 2^k:
    - # ANDS = 2^k
    - # inputs = k
  - k/2 x $2^{k/2}$:
    - # ANDS = 2($2^{k/2}$) = $2^{k/2+1}$
    - # inputs = k/2
- Example: 1K x ?? RAM:
  - 1024 10-input ANDS versus
  - 64 5-input ANDS

Address multiplexing

cs309

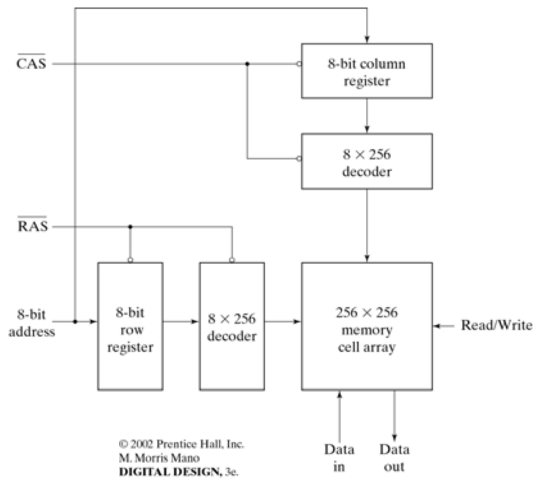


Fig. 7-8 Address Multiplexing for a 64K DRAM

Reduces # pins on chip and wires in buss

The University Of Alabama in Huntsville Computer Science

G. W. Cox – Spring 2010

cs309

Error detection and correction

The University Of Alabama in Huntsville Computer Science

G. W. Cox – Spring 2010

Error detection

cs309

- Can determine (within limits) that there is an error(s) in a binary data item, but cannot tell which bit(s) is wrong
- Example: Simple parity check will detect any odd number of bits in error (review)
  - Odd parity -- add a parity bit to the item that makes the total # 1's odd
    - stored = 10110000
    - read = 10010000 → error
  - Even parity – same, except make # 1's even
  - Recal that we can use XOR to implement this

The University Of Alabama in Huntsville Computer Science

Error correction

cs309

- Can determine (within limits) that there is an error(s) in a binary data item, and which bit(s) is wrong
- Example: 2D parity check

Stored	Read
1011 0	1011 0
0001 0	0101 0 x
1111 1	1111 1
0000 1	0000 1
1010 1	1010 1
	x

The University Of Alabama in Huntsville Computer Science

Hamming Code

The University Of Alabama in Huntsville Computer Science

cs309

- Basic Hamming Code can detect and correct any single error
- Can be extended to any length data
- Can be extended to more errors
- Efficient (low number of parity bits per data bit)

For 8-bit data:

```
p1 p2 d1 p4 d2 d3 d4 p8 d5 d6 d7 d8

p1 = XOR (d1, d2, d4, d5, d7)
p2 = XOR (d1, d3, d4, d6, d7)
p4 = XOR (d2, d3, d4, d8)
p8 = XOR (d5, d6, d7, d8)

to check:
c1 = XOR (p1, d1, d2, d4, d5, d7)
c2 = XOR (p2, d1, d3, d4, d6, d7)
c4 = XOR (p4, d2, d3, d4, d8)
c8 = XOR (p8, d5, d6, d7, d8)

"c8 c4 c2 c1" read as binary gives the error position
```


The University Of Alabama in Huntsville Computer Science

cs309

ROMs and Array Logic

G. W. Cox – Spring 2010

ROM organization

The University Of Alabama in Huntsville Computer Science

cs309

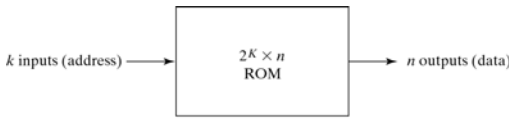


Fig. 7-9 ROM Block Diagram

Crosspoint is connected for “1”, not for “0”.
(do this by blowing fuse where you want a “0”)

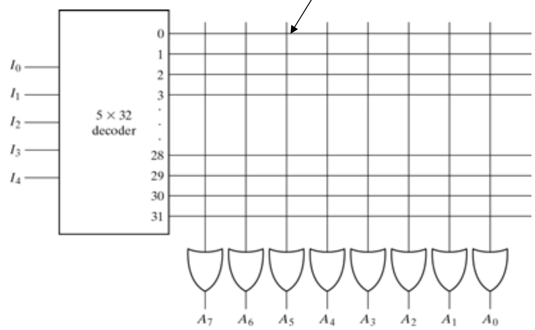


Fig. 7-10 Internal Logic of a 32 × 8 ROM

ROM types

The University Of Alabama in Huntsville Computer Science

cs309

- Mask Programmable ROM
  - Programmed at factory
- Programmable ROM (PROM)
  - Shipped with all fuses intact.
  - Special eqpt blows fuses where “0” is wanted.
  - Cannot be changed
- Erasable PROM (EPROM)
  - Blown fuses can be restored by exposure to UV light
- Electrically Erasable PROM (EEPROM or E^2PROM)
  - Blown fuses can be restored electrically (in-socket)
  - Subject to fatigue
- Flash Memory
  - Variant of EEPROM
  - Can selectively erase parts of memory
  - Subject to fatigue

ROM programming

I4	I3	I2	I1	I0	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
	⋮				⋮					⋮		

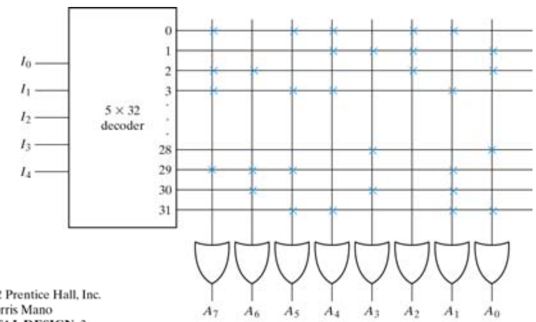
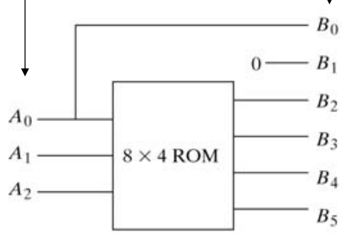


Fig. 7-11 Programming the ROM According to Table 7-3

A ROM is a minterm generator: we can implement combinational circuits

3-bit number

Square of number



(a) Block diagram

A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

(b) ROM truth table

Fig. 7-12 ROM Implementation of Example 7-1

Programmable Logic Devices (PLDs)

PLD Types

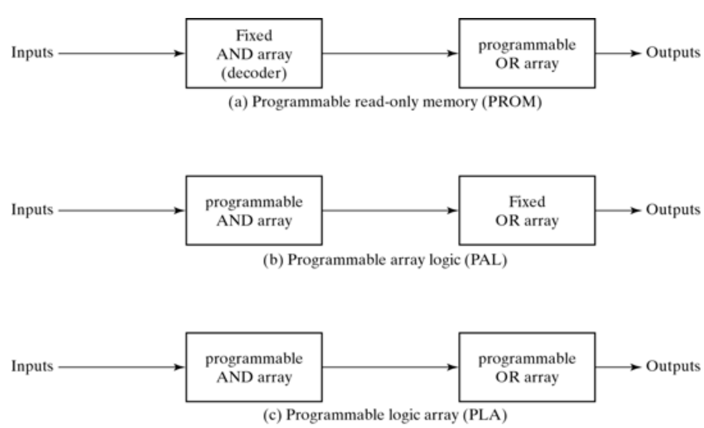


Fig. 7-13 Basic Configuration of Three PLDs

PLA

cs309

The University Of Alabama in Huntsville Computer Science

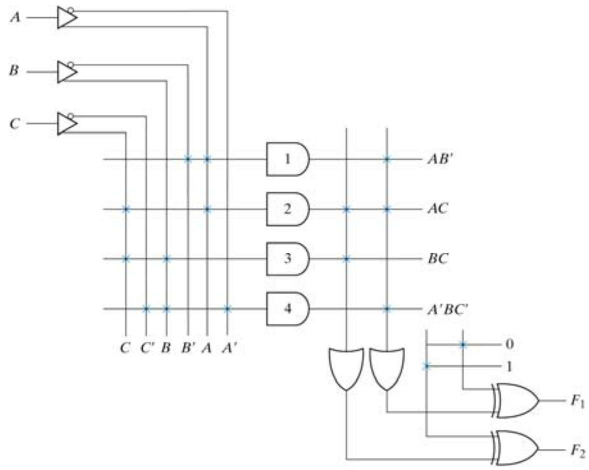


Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

G. W. Cox – Spring 2010

PAL

cs309

The University Of Alabama in Huntsville Computer Science

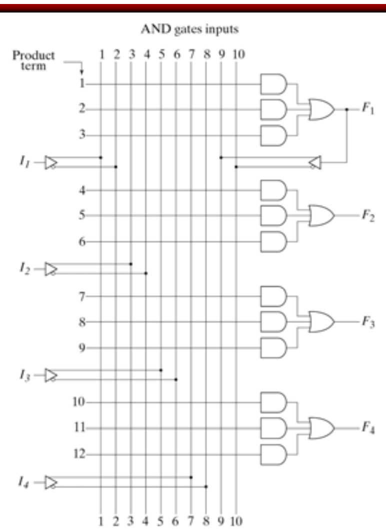
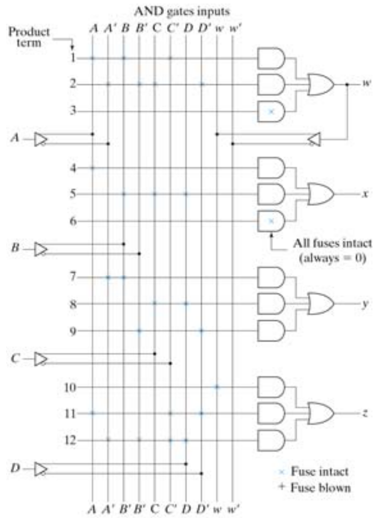


Fig. 7-16 PAL with Four Inputs, Four Outputs, and Three-Wide AND-OR Structure

G. W. Cox – Spring 2010

A Programmed PAL



w = ABC' + A'B'CD'

x = A + BCD

y = A'B + CD + B'D'

z = w + AC'D' + A'B'C'D

Fig. 7-17 Fuse Map for PAL as Specified in Table 7-6

Sequential PLDs

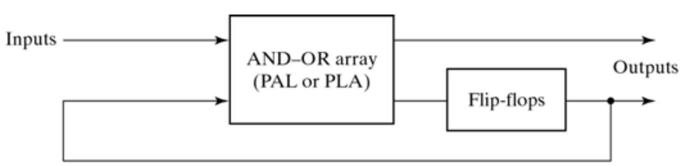


Fig. 7-18 Sequential Programmable Logic Device

The University Of Alabama in Huntsville Computer Science

Macrocell

cs309

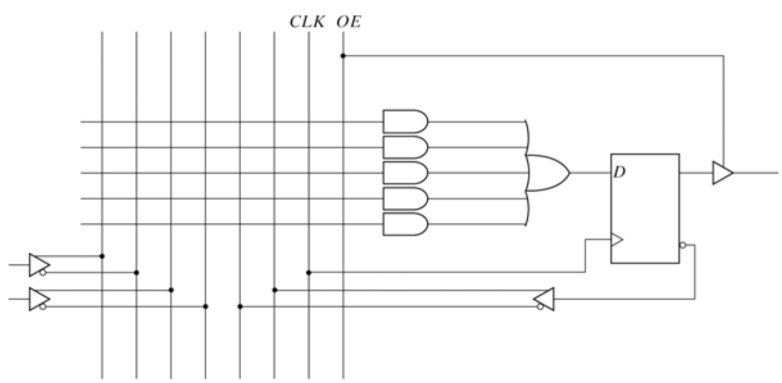


Fig. 7-19 Basic Macrocell Logic

The University Of Alabama in Huntsville Computer Science

Complex PLD (CPLD)

cs309

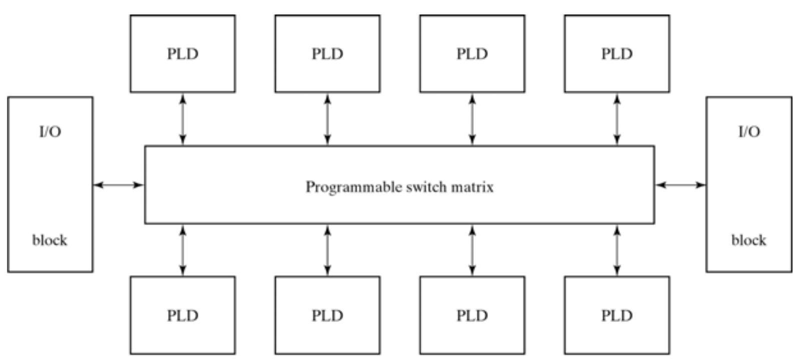


Fig. 7-20 General CPLD Configuration