University of Massachusetts Amherst

Engin112 – Lectures 35-36

Memory - RAM, ROM

Maciej Ciesielski
Department of Electrical and Computer Engineering
11/30-12/02/2011

Recap from last lectures

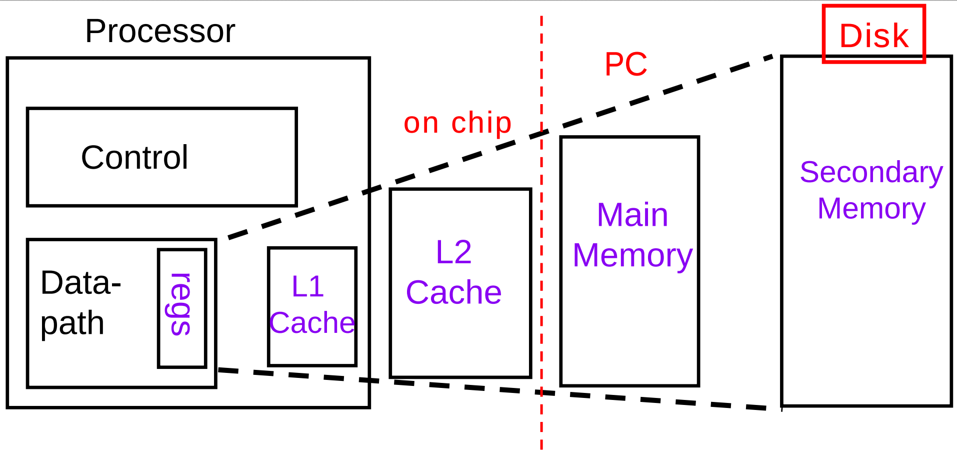
- Sequential logic
  - FSMs
  - Fliplfops, latches
  - Registers

- Today's lecture:
  Computer Memory
  - Random Access Memory (RAM)
  - Read Only Memory (ROM)

Memories

- Where is a memory used?
  - Memory hierarchy (next slide)
- What are the differences between memories?
  - Technology: magnetic, voltage, charge, fuse
  - Volatility: volatile, non-volatile
  - Access sequence: random, sequential
  - Speed: access time and throughput
  - Access type:
    - » Random Access Memory (RAM)
    - » Read Only memory (ROM)
- Is memory combinatorial or sequential logic?
  - RAM: sequential (similar to register)
  - ROM: combinatorial (programmable logic)

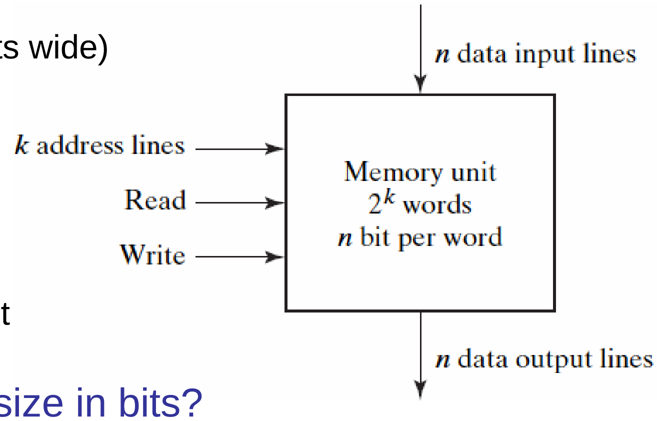
Memory Hierarchy



Speed (ns):	0.1ns	0.3ns	1ns	10ns	8,000,000ns
Size (MB):	0.001	0.5	2-6	4,000-8,000	500,000+
Cost ($/MB):	--	--	$10	$0.01	$0.0001
Technology:	Regs	SRAM	SRAM	DRAM	Disk

Random-access Memory (RAM)

- Memory terminology:
  - Smallest unit is "word" (*n* bits wide)
  - Identified by "address"
- Block diagram:
  - *k* address lines
    - » At most 2^k words storage
  - n data lines input and output



- What is the total memory size in bits?
  - Size = $2^k \times n$ bits
- Example:
  - 1024-bit memory with 16-bit words
  - How many address lines?

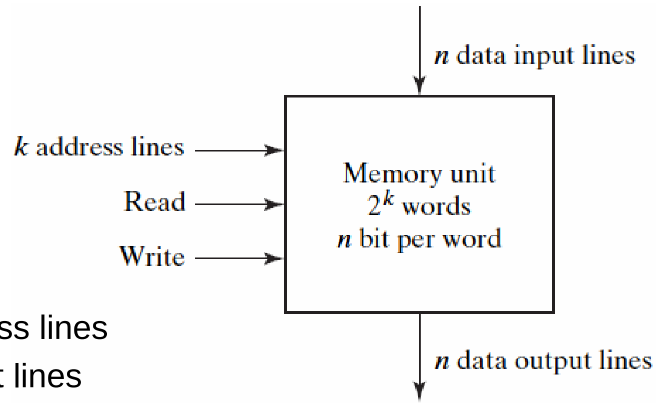
Memory Layout

- Example:
  - 1024 16-bit word storage
  - 10 bits identify address
  - 16 bits stored at each location
- Why not make memory bit-wise addressable?
- Remember:
  - k (Kilo) = 2^{10}
  - M (Mega) = 2^{20}
  - G (Giga) = 2^{30}

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

RAM Read and Write

- Control inputs:
  - Address
  - Read
  - Write



- Write operation
  1. Apply address to address lines
  2. Apply data to data input lines
  3. Activate write input

- Read operation
  1. Apply address to address line
  2. Activate read input
  3. Read value from data output lines

RAM Read and Write

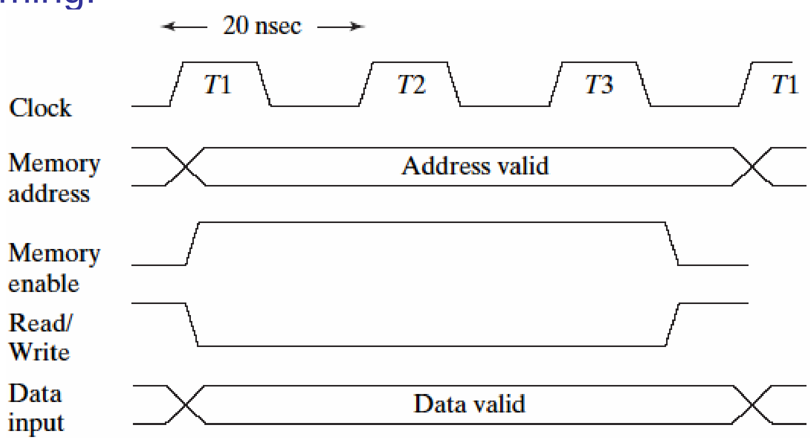
- Control inputs on commercial memories
  - Memory enable
  - Read and write (often $\overline{read/write}$)

Memory enable	Read / write	Memory operation
0	X	none
1	0	write to selected word
1	1	read from selected word

- What timing constraints need to be considered?
  - *Access time* = time to read a word
  - *Cycle time* = time to write a word
  - Also: *setup* and *hold time* on address

RAM Timing

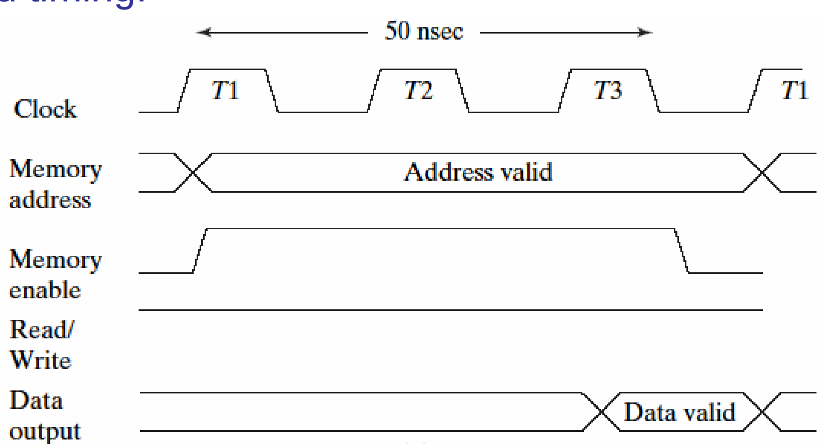
- Example:
  - Processor clock f = 50MHz (20*ns*)
  - RAM access time = RAM cycle time = 50 *ns*
- Write timing:



  - Address and data needs to be valid before memory enable

RAM Timing

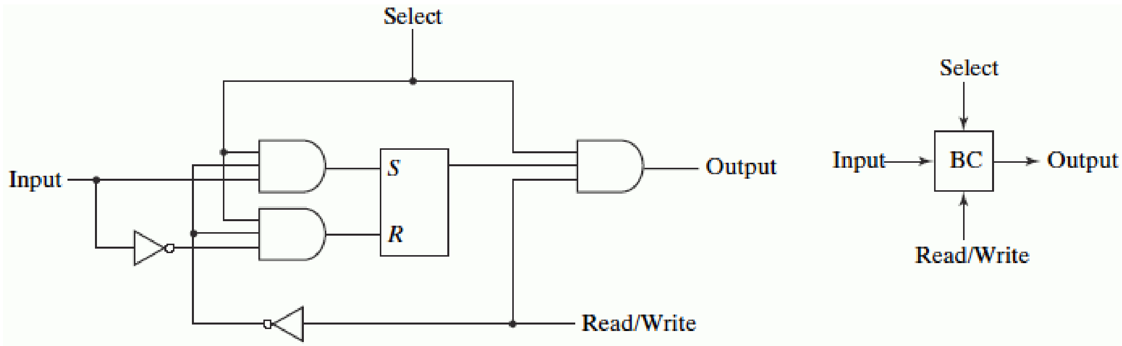
- Read timing:



  - Address needs to be valid before memory enable
  - Data becomes valid before access time expires
- Processor and memory often not “in sync”
  - Memory access can take multiple clock cycles

Memory Cell Design

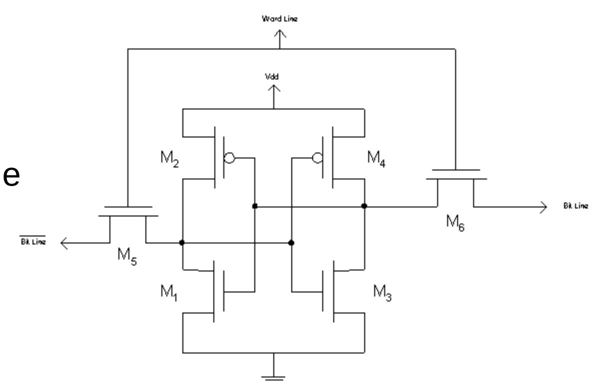
- Binary memory cell implemented with SR latch:



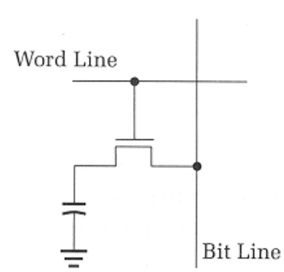
  - Note: memory is not clocked

RAM Types

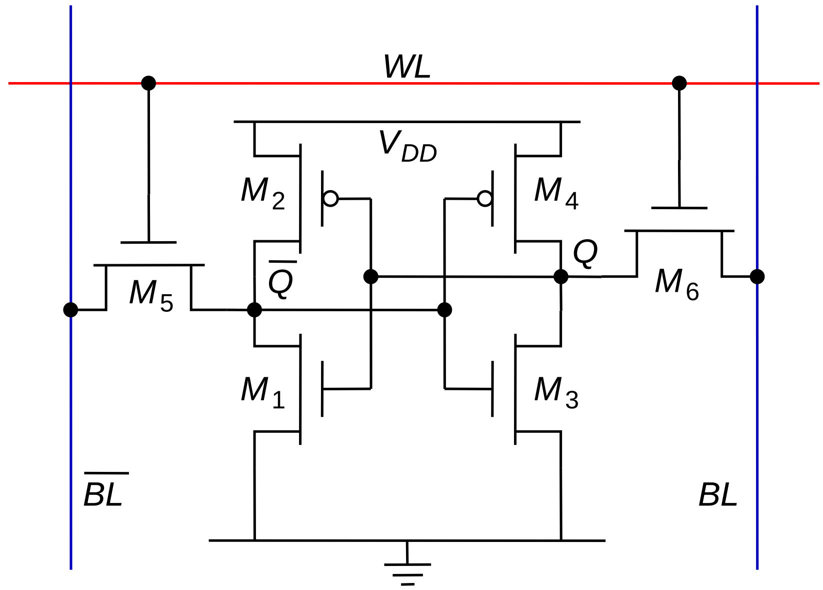
- Static random access memory (SRAM)
  - Operates like a latch
  - Implemented with 6 transistors
  - Fast operation
  - Memory retains information while power is applied



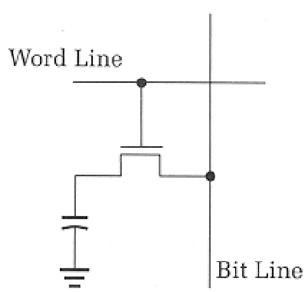
- Dynamic random access memory (DRAM)
  - Data stored as charge in capacitance
  - Implemented with 1 transistor
  - Slower operation
  - Memory needs to be “refreshed”

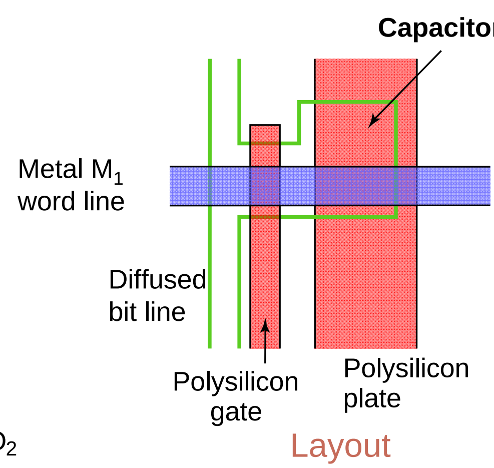


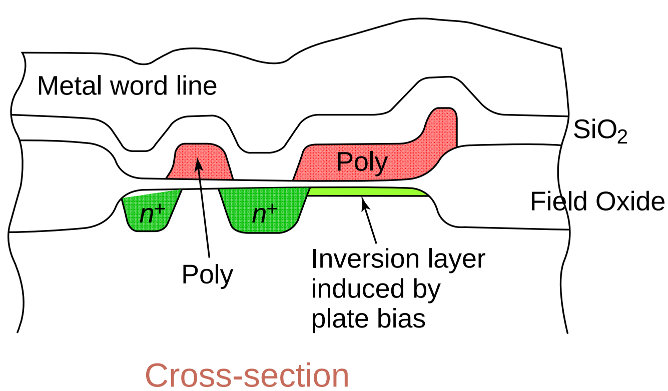
6-transistor CMOS SRAM Cell



1-Transistor DRAM Cell Design



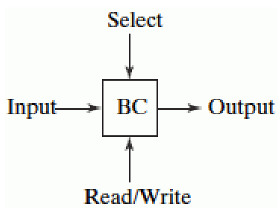




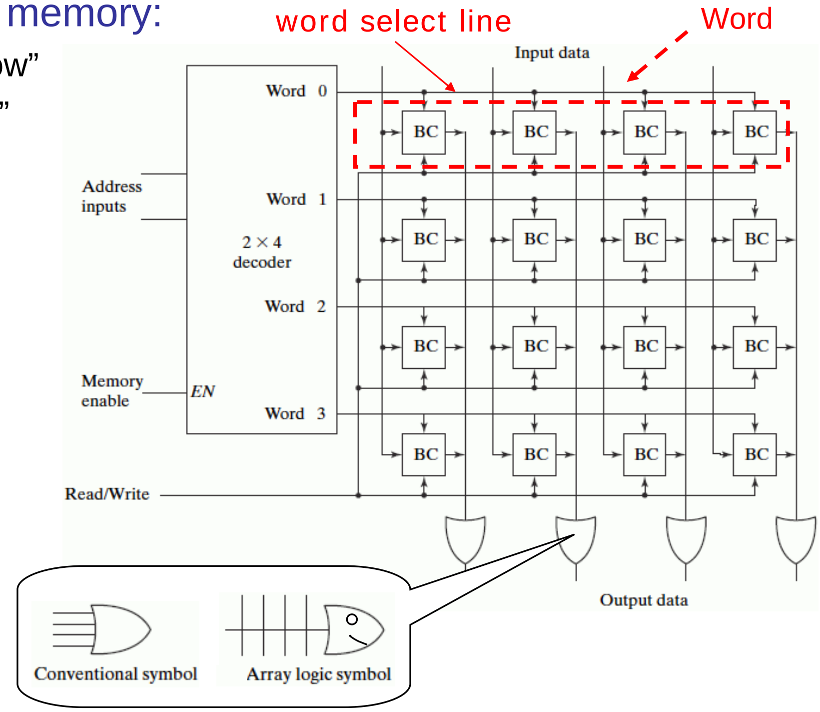
Uses Polysilicon-Diffusion Capacitance

Memory Organization

- Internal layout of memory:
  - One word per "row"
  - Each "binary cell" stores one bit
- Memory cell:

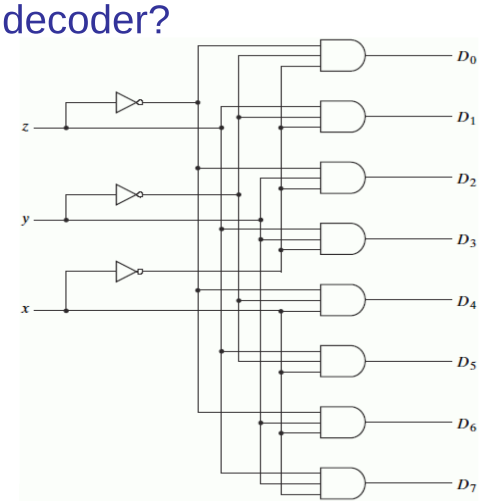


- Decoder:
  - Selects word



Larger Memories – Address Decoding

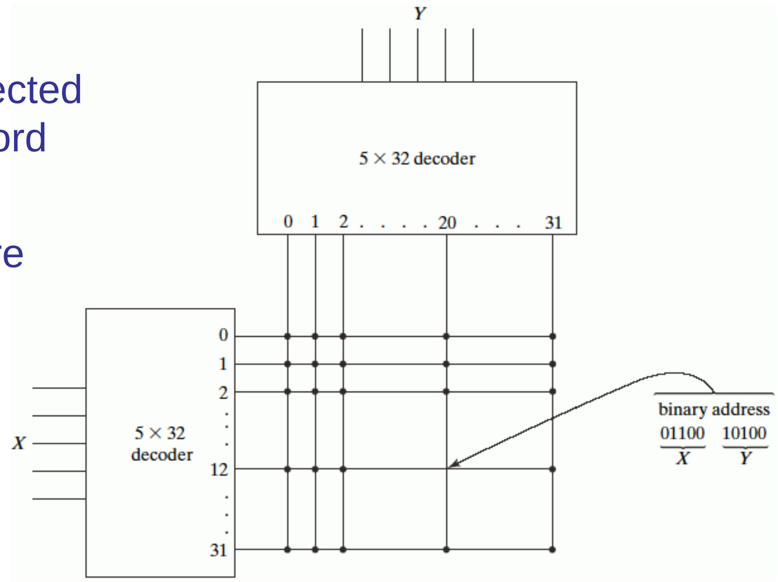
- What is the problem with larger memories?
  - Number of binary cells increases (that's expected)
  - Decoder becomes really large!
- How many gates in a $k \times 2^k$ line decoder?
  - 2^k AND gates with k inputs each!



- Example:
  - 1024-word memory decoder requires 1024 AND gates with 10 inputs
- Solution?
  - address decoding

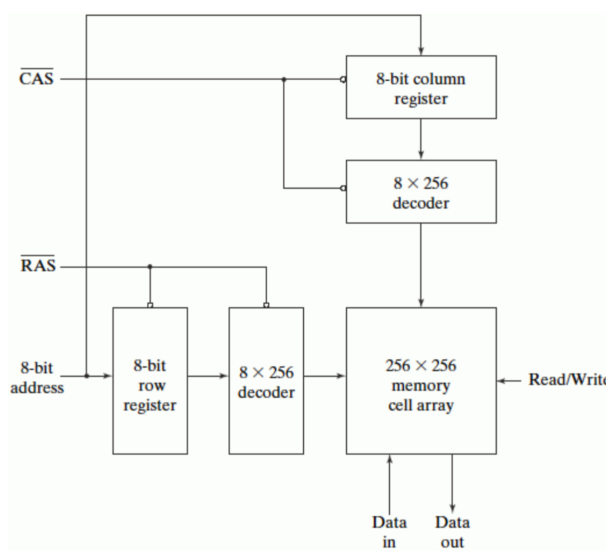
Coincident Decoding – Address Splitting

- Split decoders into two dimensions:
  - $k/2 \times 2^{k/2}$ decoder in each dimension
- Coincidence of selected lines determines word
- How many gates are required?
  - 32 + 32 = 64 ANDs with 5 inputs



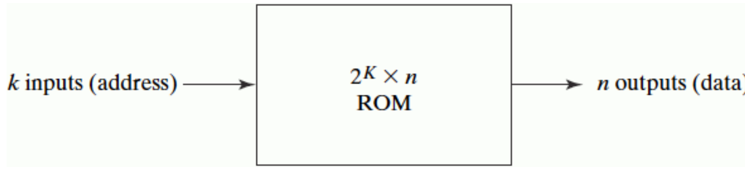
Address Multiplexing - DRAM

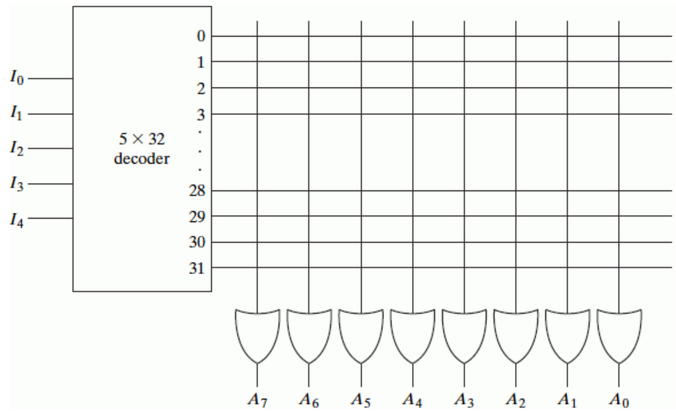
- Number of pins on memory chip impacts cost
  - How can we reduce the number of pins necessary?
- Address multiplexing
  - Address is transmitted in parts
- Example: 64k word DRAM
  - CAS
    - » "Column Address Strobe"
  - RAS
    - » "Row Address Strobe"
- What does timing diagram look like?



Read-Only Memory (ROM)

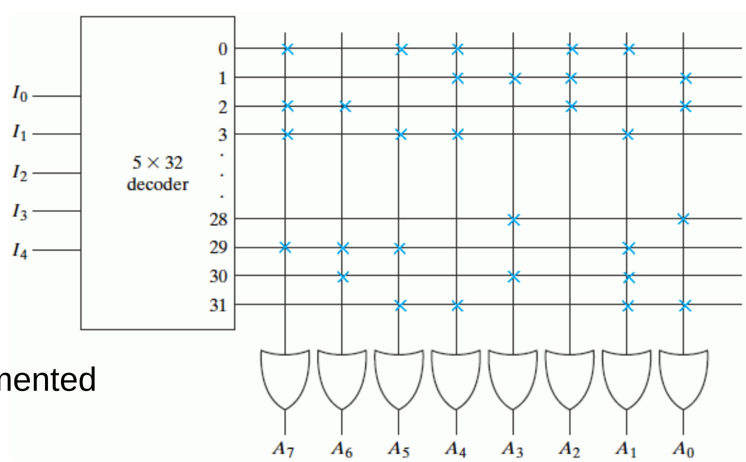
- Storage for memory that does not need to change
- Block diagram:
  - Address
  - Outputs
  - No data inputs
- Internal design
  - 32 × 8 ROM
  - 256 internal connections
- Connections:
  - No connection = 0
  - Connection = 1





Read-Only Memory

- Example of "programmed" ROM:
  - "x" indicates connection



- Programming of ROM
  - Connections implemented as "fuses" (logic 1)
  - Blown fuse (open) indicates 0
  - Higher voltage can blow fuse

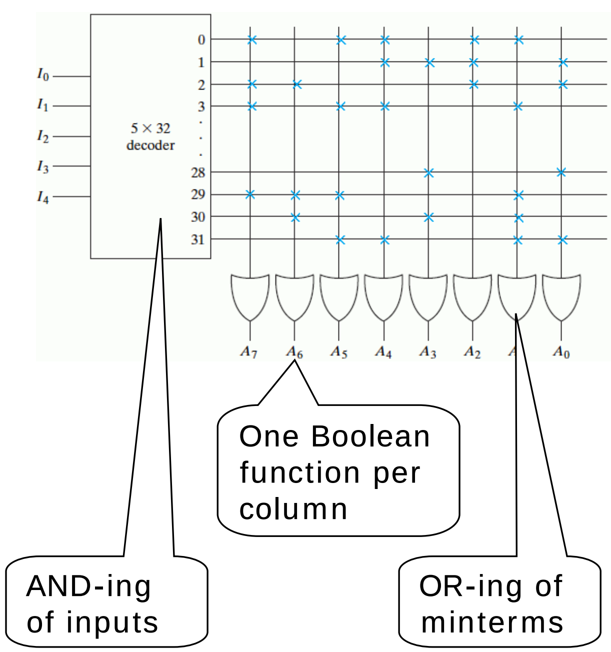
Types of ROM

- Mask Programmed ROM
  - During fabrication of ROM
  - Only economical in large quantities
- Programmable ROM (PROM)
  - ROM with all fuses intact
  - High-voltage pulse on special pin can irreversibly blow fuses
  - Note: hardware procedure despite “programmable”
- Erasable PROM (EPROM)
  - Connections can be restored
  - Exposure to UV light resets EPROM

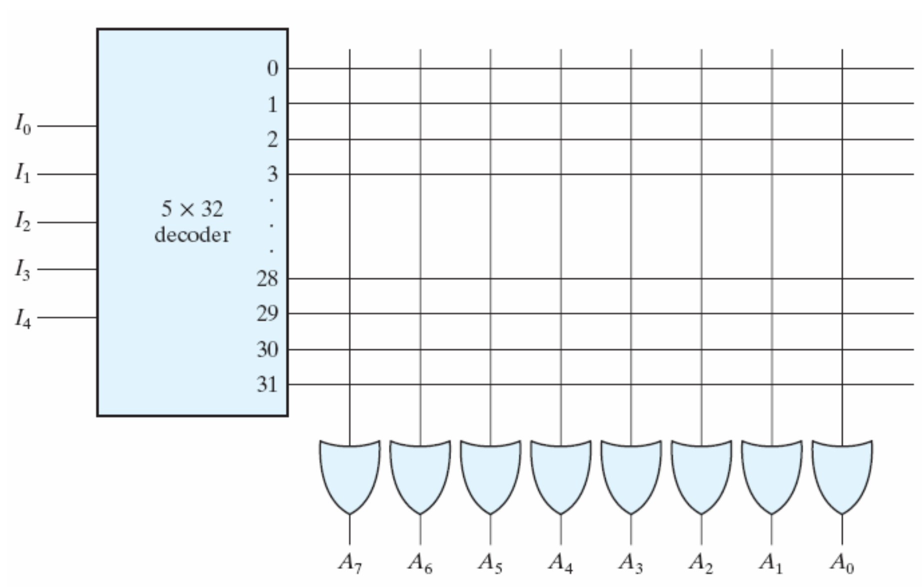
- Electrically-erasable PROM (EEPROM)
  - Electric signal can reset connections (no UV required)

Programmable ROM

- Any Boolean function can be implemented with a decoder and an OR gate
  - Decoder produces minterms
  - OR gate makes sum
- How many Boolean function can be implemented on 32×8 ROM?
  - 8 Boolean functions
  - 5 inputs each
- AND-OR structure
- The biggest part of ROM:
  - Decoder

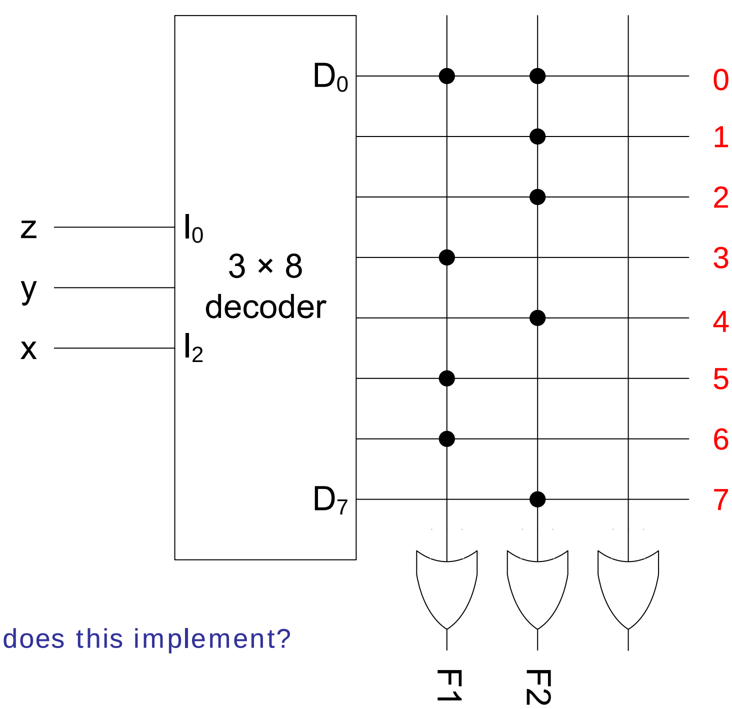


Programmable ROM - core



Programmable ROM - example

- Implement
  - *F1(x,y,z)*
    = Σ(0,3,5,6)
    = x'y'z' + x'yz
    +xy'z + xyz'
  - *F2(x,y,z)*
    = Σ(0,1,2,4,7)
    = x'y'z' + x'y'z
    +x'yz' + xy'z' + xyz



What (arithmetic) function does this implement?