Read Only Memory (ROM)

Overview

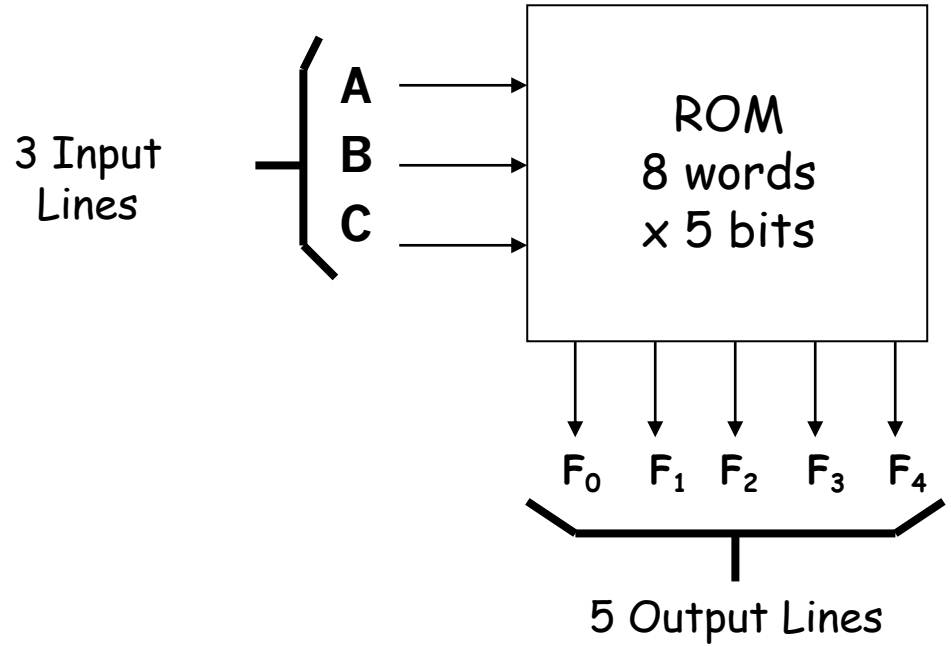
- Read-only memory can normally only be read
- Internal organization similar to SRAM
- ROMs are effective at implementing truth tables
  - Any logic function can be implemented using ROMs
- Multiple single-bit functions embedded in a single ROM
- Also used in computer systems for initialization
  - ROM doesn’t lose storage value when power is removed
- Very useful for implementing FSMs

Read-Only Memory (ROM)

- 2^N words by M bits
- Data can be read but not changed
  - (normal operating conditions)

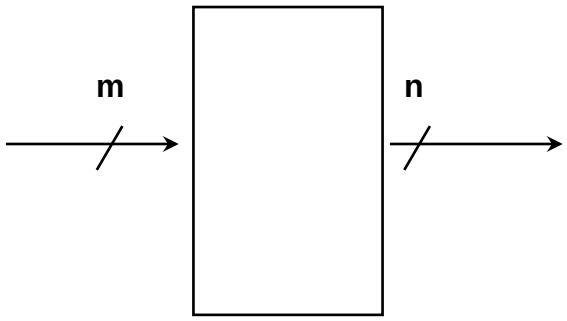
Read-Only Memory (ROM)

- N input bits
- 2^N words by M bits
- Implement M arbitrary functions of N variables
  - Example 8 words by 5 bits:



ROM Implementation

- **ROM = "Read Only Memory"**
  - **values of memory locations are fixed ahead of time**
- **A ROM can be used to implement a truth table**
  - **if the address is m-bits, we can address 2^m entries in the ROM.**
  - **our outputs are the bits of data that the address points to.**



0	0	0	0	0	1	1
0	0	1	1	1	0	0
0	1	0	1	1	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	1
1	1	0	0	1	1	0
1	1	1	0	1	1	1

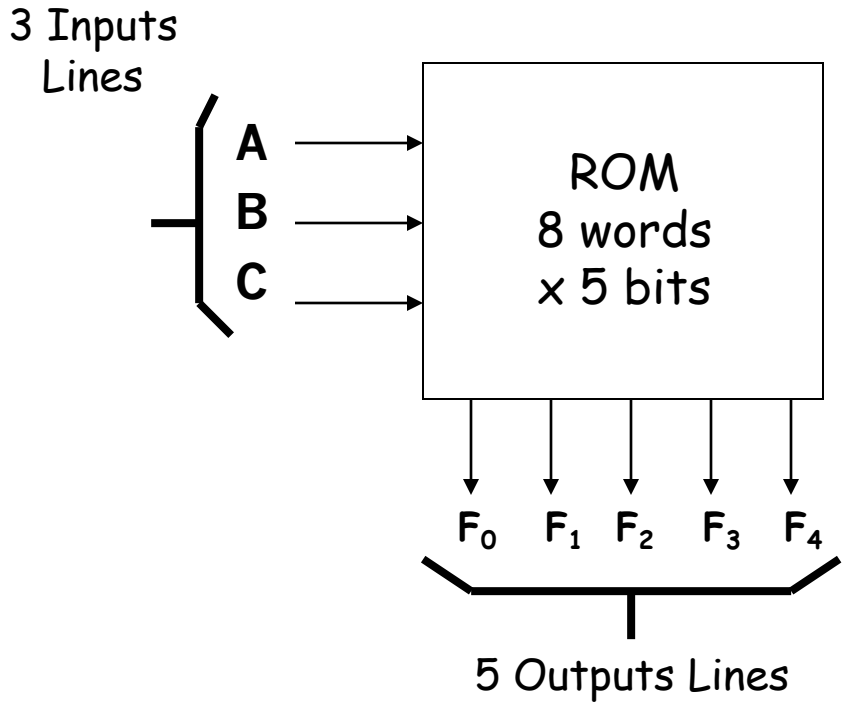
- **m is the "height", and n is the "width"**

ROM Implementation

- **Suppose there are 10 inputs**
  10 address lines (i.e., 2^{10} = 1024 different addresses)
- **Suppose there are 20 outputs**
- **ROM is 2^{10} x 20 = 20K bits (and a rather unusual size)**
- **Rather wasteful, since lots of storage bits**
  - **For functions, doesn't take advantage of K-maps, other minimization**

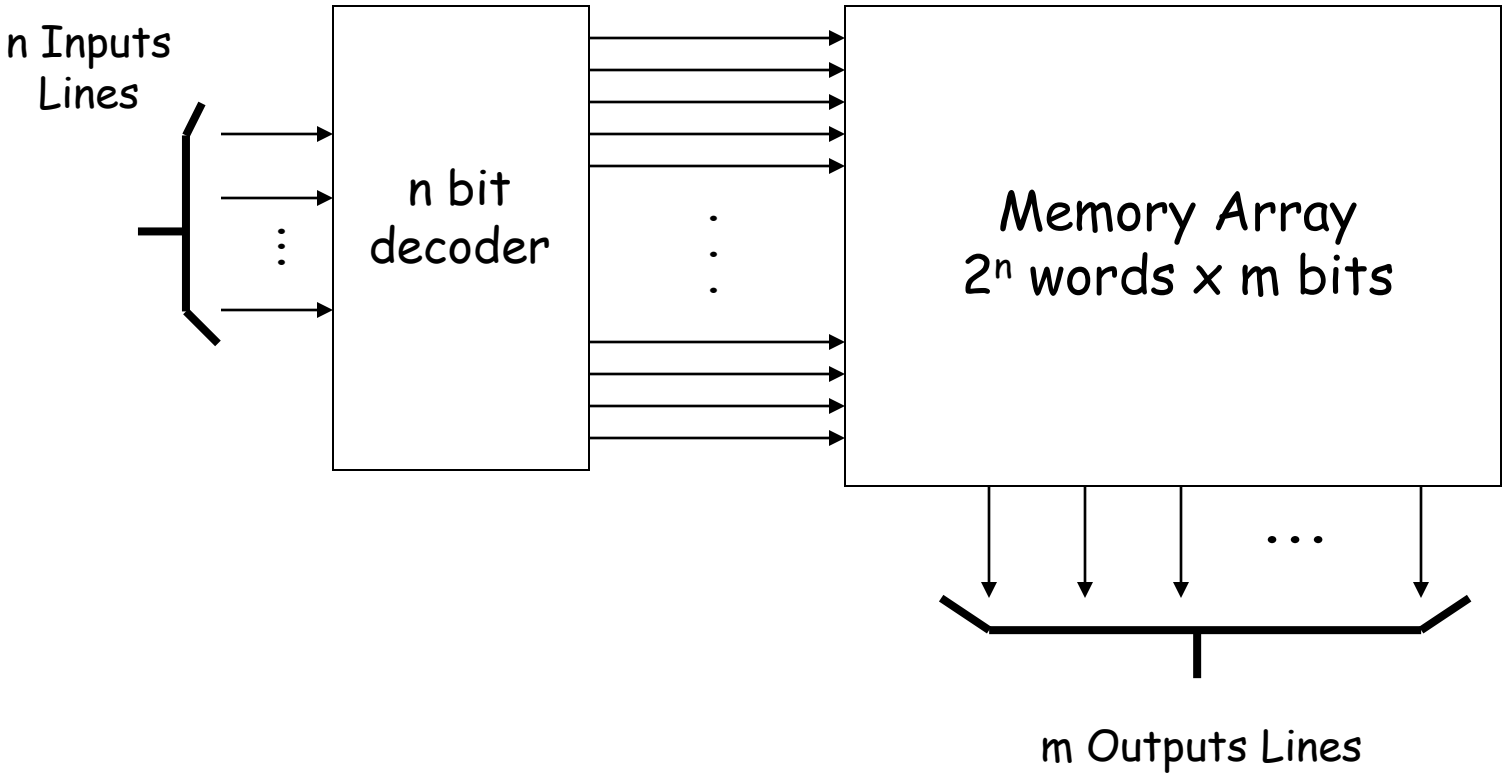
Read-Only Memory (ROM)

Each minterm of each function can be specified

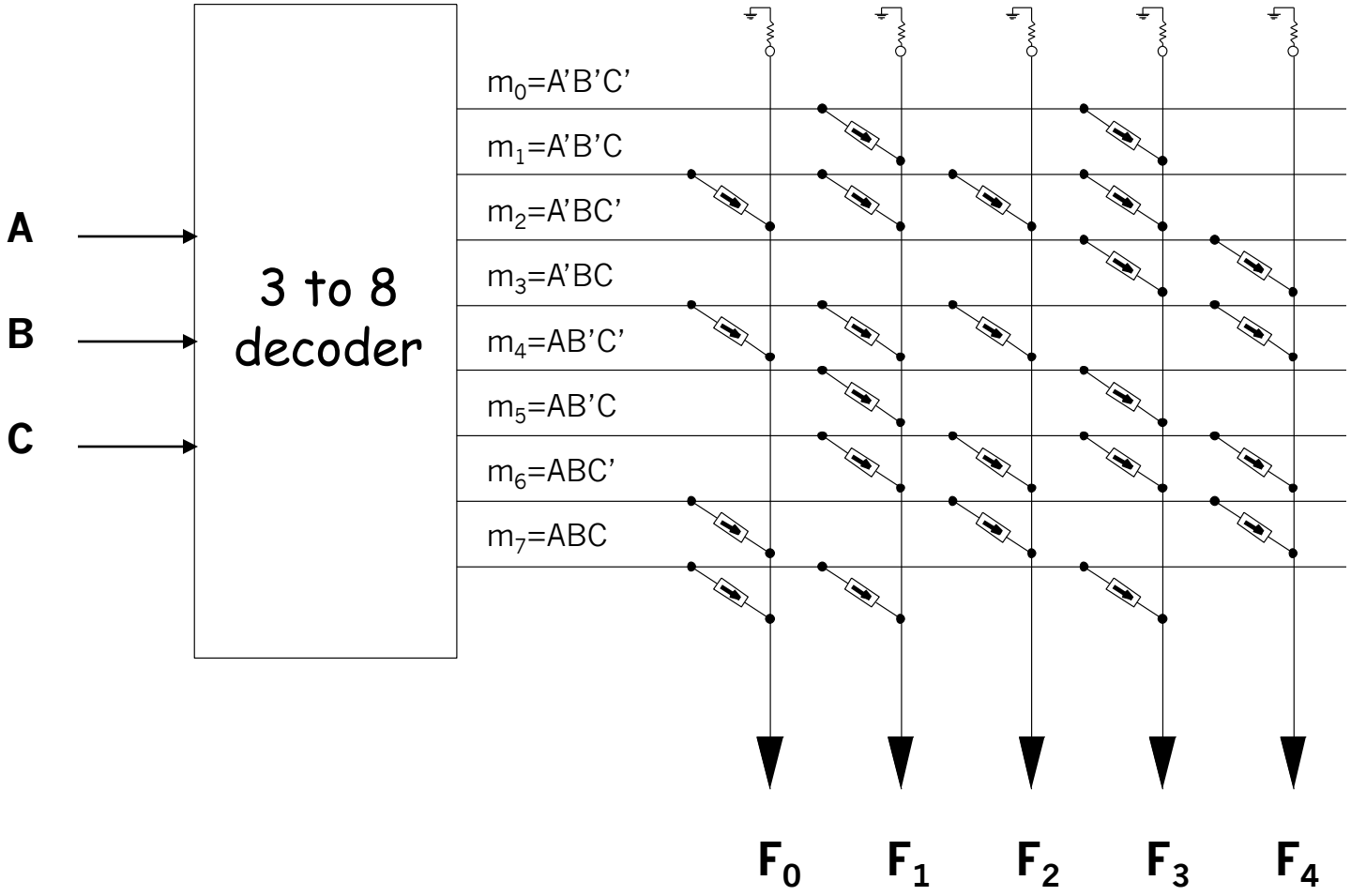


A	B	C	F_0	F_1	F_2	F_3	F_4
0	0	0	0	1	0	1	0
0	0	1	1	1	1	1	0
0	1	0	0	0	0	1	1
0	1	1	1	1	1	0	1
1	0	0	0	1	0	1	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	0	1
1	1	1	1	1	0	1	0

ROM Internal Structure



ROM Memory Array



Inside the ROM

- **Alternate view**
  - **Each possible horizontal/vertical intersection indicates a possible connection**
- **Or gates at bottom output the word selected by the decoder (32 x 8)**

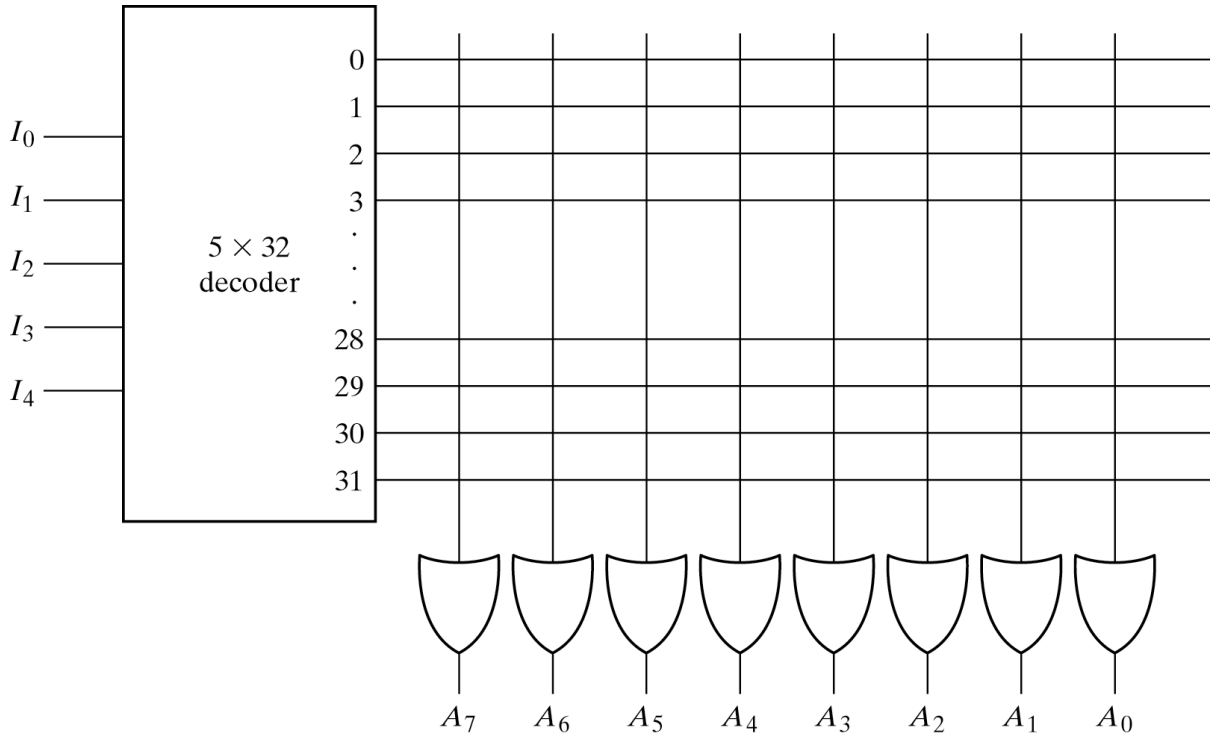


Fig. 7-10 Internal Logic of a 32 × 8 ROM

ROM Example

Specify a truth table for a ROM which implements:

F = AB + A'BC'

G = A'B'C + C'

H = AB'C' + ABC' + A'B'C

A	B	C	F	G	H
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

ROM Example

Specify a truth table for a ROM which implements:

F = AB + A'BC'

G = A'B'C + C'

H = AB'C' + ABC' + A'B'C

A	B	C	F	G	H
0	0	0	0		
0	0	1	0		
0	1	0	1		
0	1	1	0		
1	0	0	0		
1	0	1	0		
1	1	0	1		
1	1	1	1		

ROM Example

Specify a truth table for a ROM which implements:

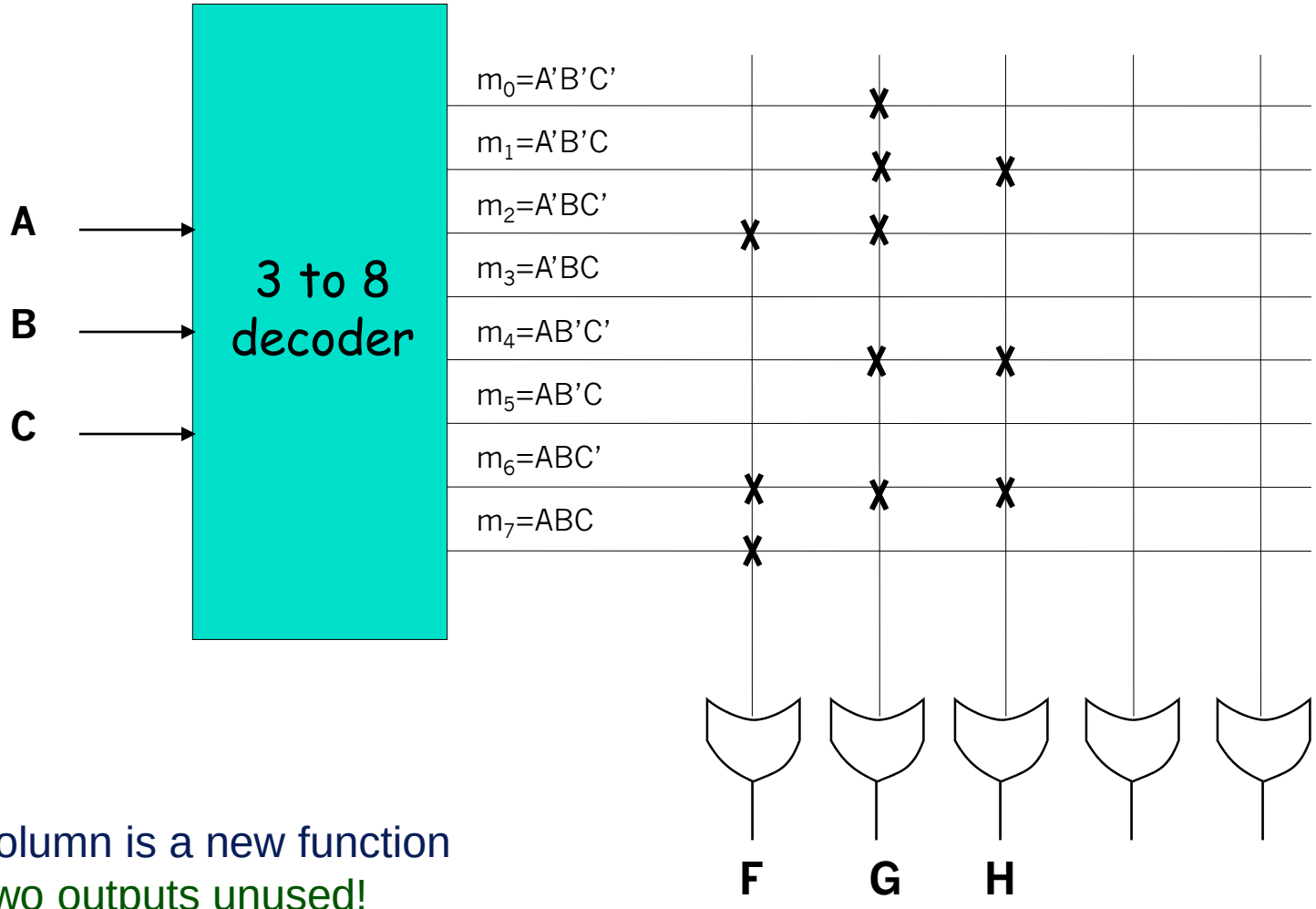
F = AB + A'BC'

G = A'B'C + C'

H = AB'C' + ABC' + A'B'C

A	B	C	F	G	H
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	0

Function Implementation



Each column is a new function

Note: two outputs unused!

Summary

- ROMs provide stable storage for data
- ROMs have address inputs and data outputs
  - ROMs directly implement truth tables
- ROMs can be used effectively in Mealy and Moore machines to implement combinational logic
- In normal use ROMs are read-only
  - They are only read, not written
- ROMs are often used by computers to store critical information
  - Unlike SRAM, they maintain their storage after the power is turned off