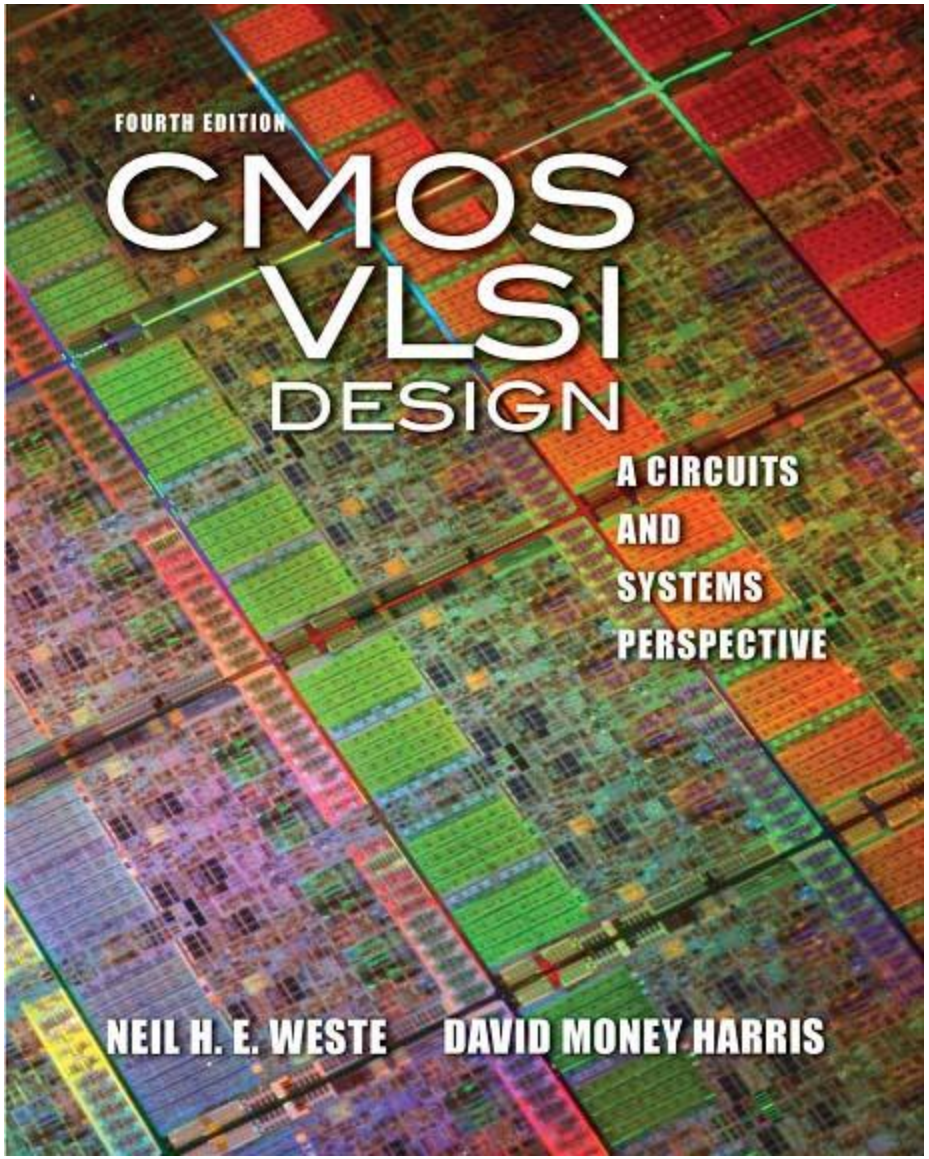
CPE 110408423
VLSI Design
Chapter 12: Array Subsystems

Bassam Jamil

[Computer Engineering Department,
Hashemite University]



Lecture 4: ROMs, SAMs, CAMs, PLAs

Outline

- Read-Only Memories
- Serial Access Memories
- Content-Addressable Memories
- Programmable Logic Arrays

12.4 Read-Only Memories

- Read-Only Memory (ROM) cells can be built with only **one transistor per bit** of storage.
- A ROM is a nonvolatile memory structure in that the state is **retained** indefinitely—even **without power**.
- A ROM array is commonly implemented:
  - As a **single-ended NOR array**.
  - Commercial ROMs are normally dynamic, although pseudo-nMOS is simple and suffices for small structures.
- As in SRAM cells and other footless dynamic gates, the wordline input must be low during precharge on dynamic NOR gates.
- In situations where DC power dissipation is acceptable and the speed is sufficient, the **pseudo-nMOS ROM is the easiest to design**, requiring no timing.

ROM Example

- Below figure shows **4-word x 6-bit Pseudo-nMOS ROM**
  - Represented with dot diagram
  - Dots indicate 1's in ROM
- The dots correspond to nMOS pulldown transistors connected to the bitlines, but the outputs are inverted.

```
Word 0: 010101
Word 1: 011001
Word 2: 100101
Word 3: 101010
```

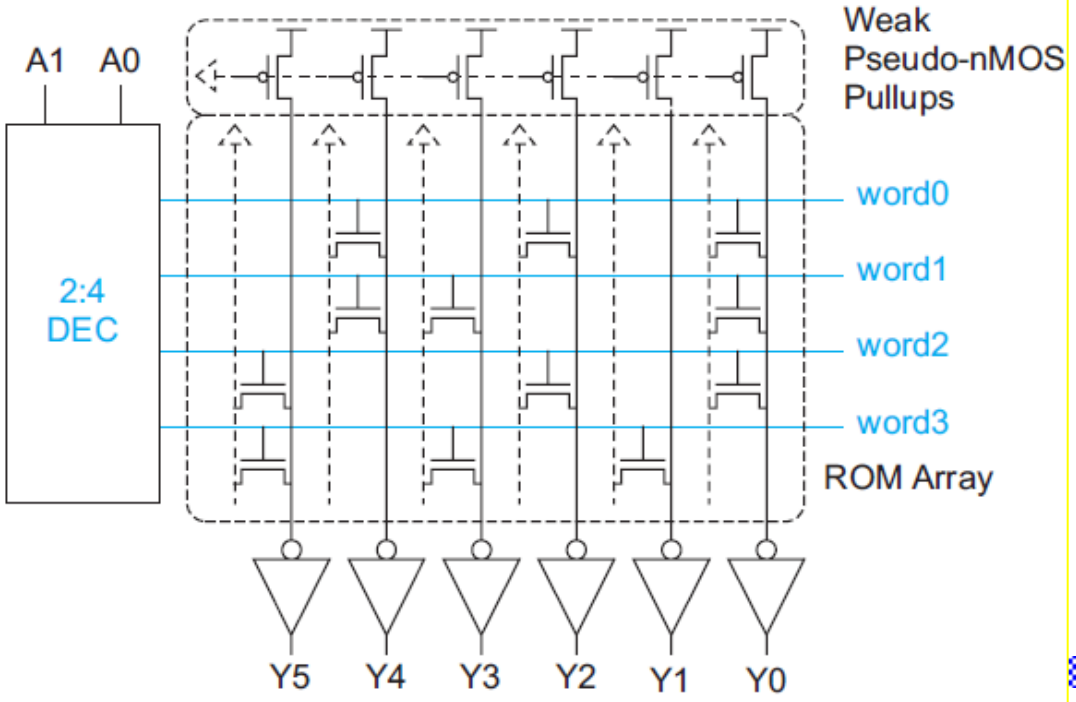
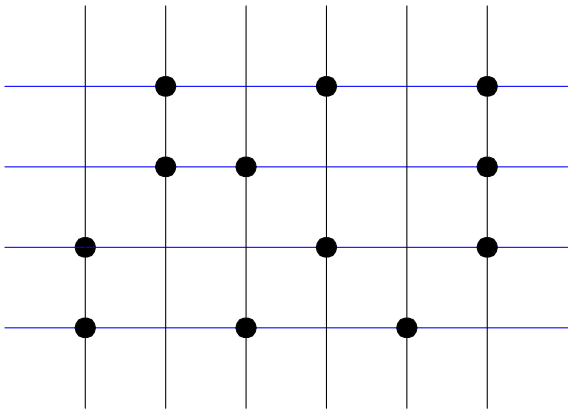


FIGURE 12.52 Pseudo-nMOS ROM



Configuring Mask-Programmed ROMs

- Mask-programmed ROMs can be **configured** by:
  - The **presence** or **absence** of a **transistor** or **contact**, OR
  - **Threshold implant** that turns a transistor permanently OFF where it is not needed.
- Omitting transistors has the advantage of reducing capacitance on the wordlines and power consumption.
- Programming with metal contacts was once popular because such ROMs could be completely manufactured except for the metal layer, and then programmed according to customer requirements through a metallization step.
- The advent of EEPROM and Flash memory chips has reduced demand for such mask programmed ROMs.

14.4.1 Programmable ROM

- Programming/writing speeds are generally slower than read speeds for ROMs.
- Four types of nonvolatile memories :
  - Programmable ROMs (PROMs),
  - Erasable Programmable ROMs (EPROMs),
  - Electrically Erasable Programmable ROMs (EEPROMs),
  - Flash memories.
- All of these memories require some enhancements to a standard CMOS process:
  - PROMs use fuses
  - EPROMs, EEPROMs, and Flash use charge stored on a floating gate.

PROMs

- Programmable ROMs can be fabricated as ordinary **ROMs** fully populated with pulldown transistors in every position.
  - Each transistor is placed in series with a **fuse** that can be **burned** out by applying **a high current**.
- The user typically configures the ROM in a specialized PROM programmer.
- As there is no way to repair a blown fuse, PROMs are also referred to as **one-time programmable memories**.
- As technology has improved, **reprogrammable nonvolatile memory** has largely displaced PROMs. These memories:
  - include: EPROM, EEPROM, Flash.
  - use a second layer of polysilicon to form a floating gate between the control gate and the channel

Floating Gate Transistor

- The floating gate is a good conductor, but it is **not attached** to anything.
- Applying a high voltage to the control gate causes **electrons to jump through the thin oxide onto the floating gate** through the processes called Fowler-Nordheim (FN) tunneling.
- Injecting the electrons **induces a negative voltage on the floating gate**, effectively **increasing the threshold** voltage of the transistor to the point that it is **always OFF**.

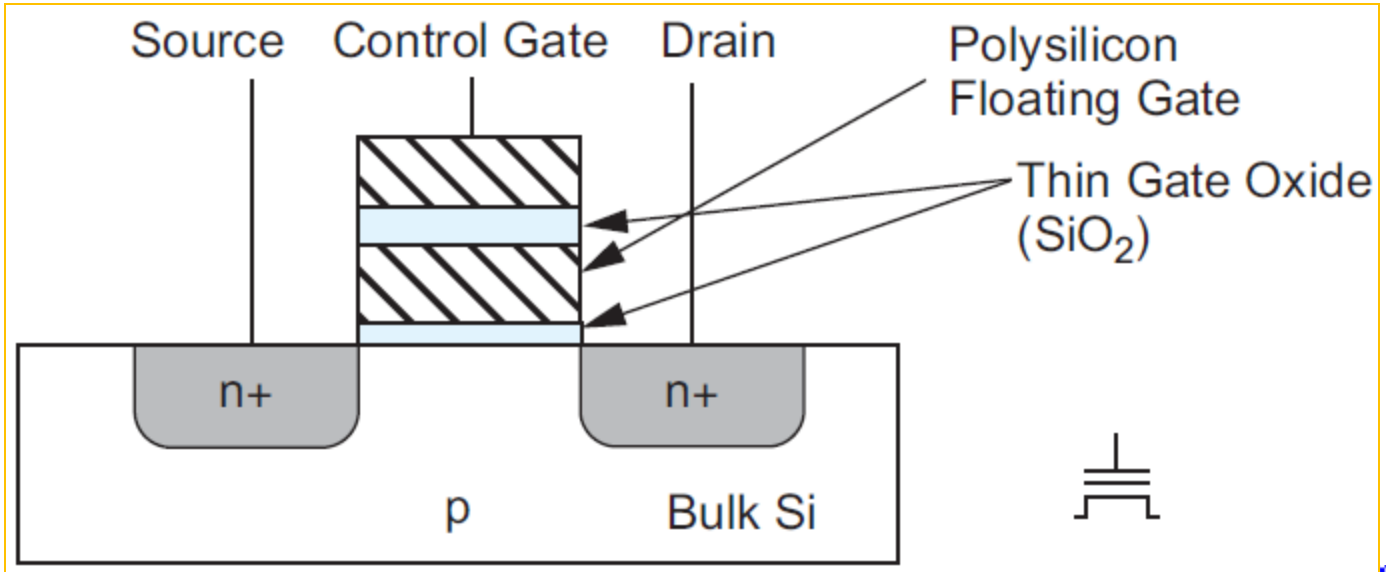


FIGURE 12.57 Cross-section of floating gate nMOS transistor

EPROM, EEPROM and Flash

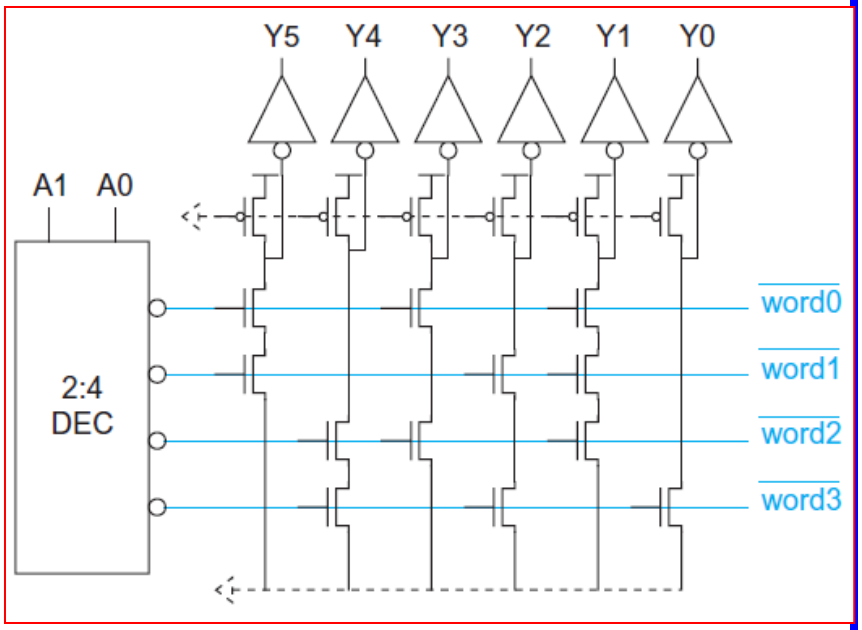
- The difference between EPROM and EEPROM lies in the way that the memory programs and erases.
  - EPROMs:
    - EPROM is programmed electrically, but it is erased through exposure to ultraviolet light that knocks the electrons off the floating gate.
    - EPROMs are encapsulated in plastic that is opaque to UV light, making them "one-time programmable".
    - It offers a dense cell.
  - EEPROM can be programmed and erased electrically.

EEPROM and Flash

- Both EEPROM and Flash can be erased electrically without being removed from the system.
  - EEPROM offers fine-grained control over which bits are erased, while Flash is erased in bulk.
  - EEPROM cells are larger, so Flash has become the most economical form of convenient nonvolatile storage.
- Most NOR Flash memory is a hybrid style—programming is through hot carrier injection and erase is through Fowler–Nordheim tunneling.

NAND ROMs

- The main issue of NOR ROM:
  - The size of the cell is limited by the ground line.
- NAND ROM
  - Does not require the ground line .
    - → Smaller area
  - Uses active-low wordlines.
  - Transistors are placed in series and the transistors on the nonselected rows are ON.
  - If no transistor is associated with the selected word, the bitline will pull down. If a transistor is present, the bitline will remain high.



- Disadvantage of the NAND ROM is that the delay grows quadratically with the number of series transistors discharging the bitline. NAND structures with more than 8–16 series transistors become extremely slow.
- NAND structures are attractive for Flash memories in which density and cost are more important than access time.

12.4.3 Flash

- **Blocks** of memory were erased all at once.
- Nonvolatile, high density and low cost per bit.
- Most stand-alone Flash memory uses the NAND architecture to minimize bit cell size and cost.
- NAND Flash memories are divided into blocks, which in turn are made of pages.
- The memory is written one **page** at a time and erased one **block** at a time.
- The charge on the floating gate determines the threshold of the transistor and indicates the state of the cell. A negative threshold represents a logic '1' and a positive threshold represents a logic 0.'.

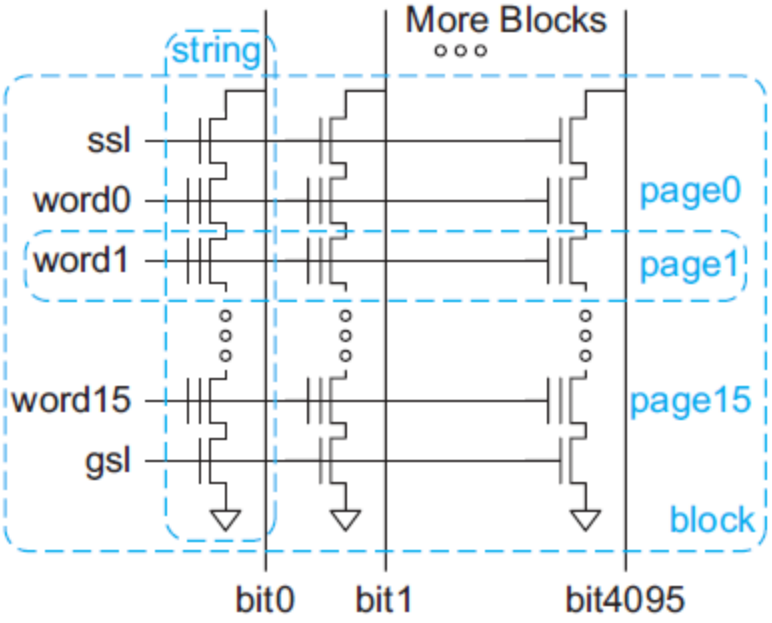


FIGURE 12.60 NAND Flash string

NAND Flash

- Floating gate transistors are connected in series to form strings.
- The Figure shows the organization of a string, page, and block.
- Each string consists of :**16 cells**, **a string select** transistor, and a **ground select** transistor all connected in series and attached to the bitline.
- The control gate of each cell is connected to a wordline.

- The array contains one column for each bit in a page.
- Each column contains one string per block.
- The number of cells in the string determines the number of pages per block.

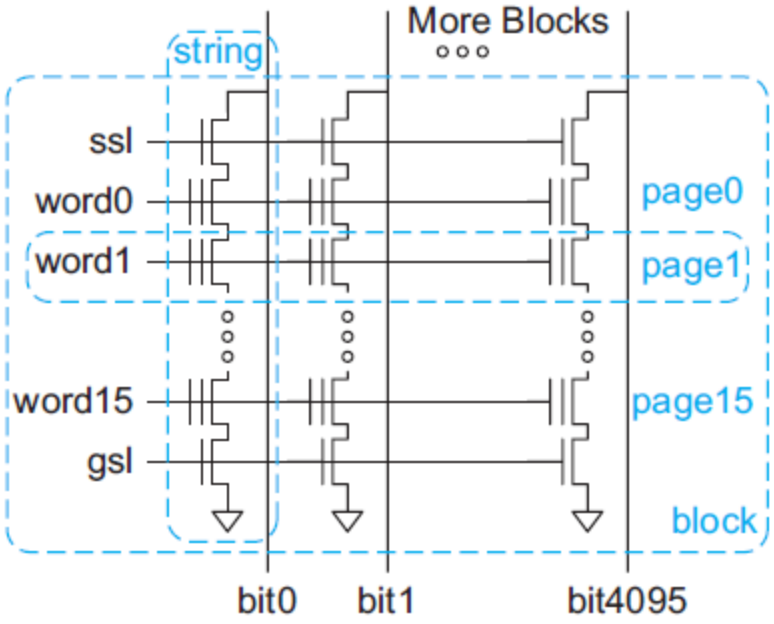


FIGURE 12.60 NAND Flash string

Flash Size

- Flash Size = Number of Blocks × Block Size
- Block Size = Page Size × String Size × Bits/Trans
- Page Size = # of columns
- String Size= # of pages
- Bits/Trans:
  - Default = 1
  - Multi-level: 2 or 4 ….
- Example (1):
  - Flash has 8 blocks, 16 pages, each page is 512 B (4 Kb).
  - Flash size = Number of Blocks × Page Size × String Size × Bits/Trans
    = 8 × 512B × 16 × 1 = 64KB
- Example (2): flash has
  - 2 blocks, Number of columns =64K, String size = 64 transistors
  - Flash uses 16 levels to store 4 bits per transistor.
  - Flash size = Number of Blocks × Page Size × String Size × Bits/Trans
    = 2 × 64K × 64 × 4
    = 32 Gb = 4 GB

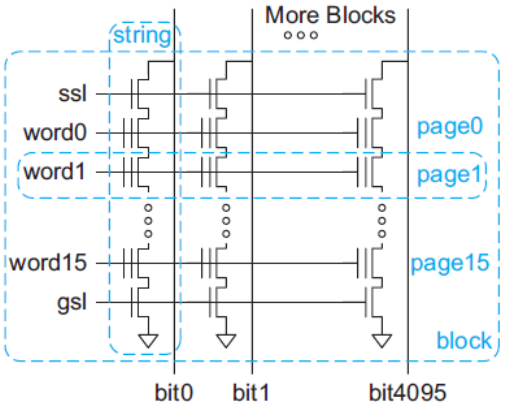
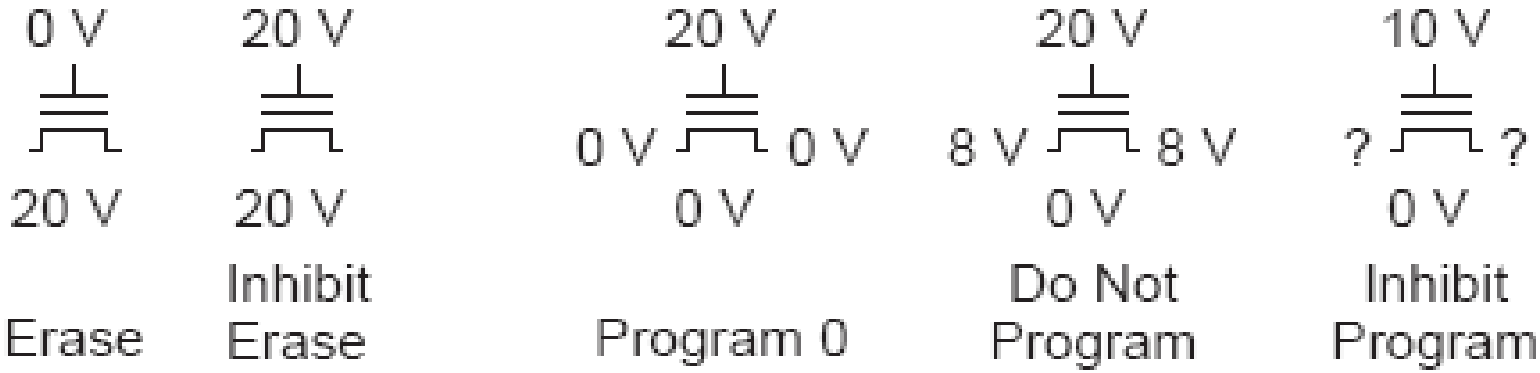


FIGURE 12.60 NAND Flash string

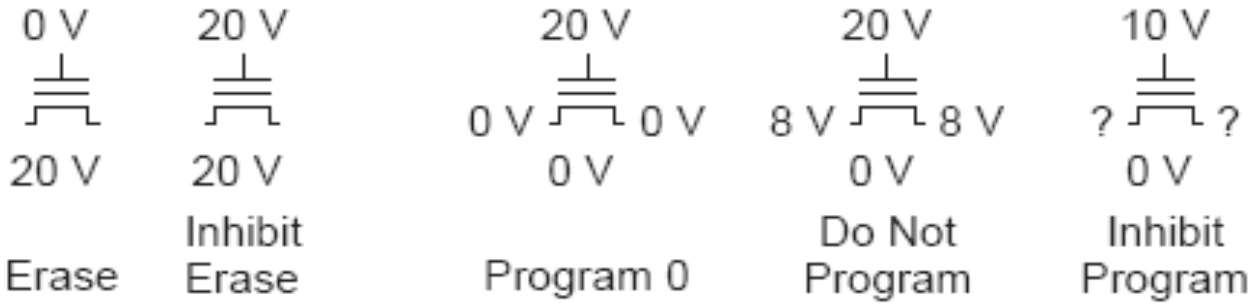
Flash Programming

- Charge on floating gate determines Vt
- Logic 1: negative Vt
- Logic 0: positive Vt
- Cells erased to 1 by applying a high body voltage so that electrons tunnel off floating gate into substrate
- Programmed to 0 by applying high gate voltage



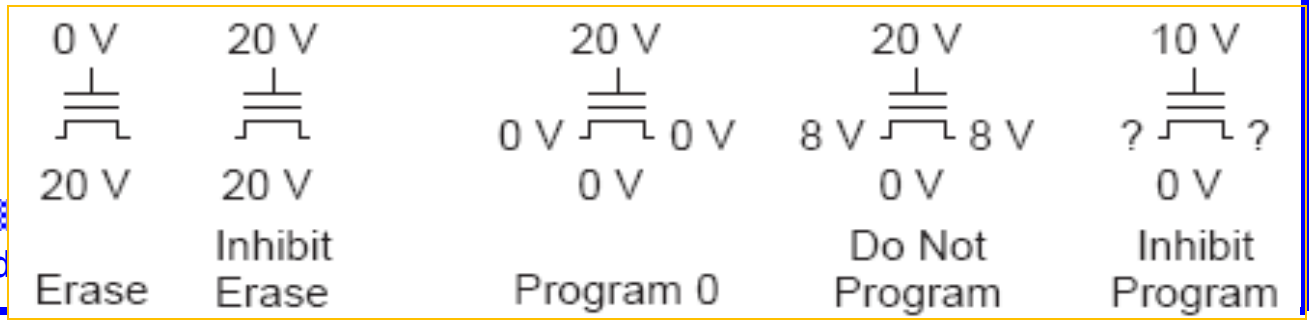
Flash Programming: Block Erase

- Block is erased by:
  - Setting all of the control **gates to GND and raising the substrate to 20 V**.
  - The high voltage across the gate oxide **induces FN tunneling**, causing the electrons to flow from the floating gate to the substrate.
  - At the end of the erase step, all the floating gate transistors have a negative Vt and thus represent 1.
- Tunneling is a **slow** process, so block erase takes on the order of a millisecond.
- The wordlines for other blocks on the chip are set to the same voltage as the substrate to inhibit erasing.
- An on-chip charge pump is used to generate the high voltages.



Flash Programming: page programming

- A cell is programmed (written) to 0 by tunneling electrons onto the floating gate.
- The programming cannot restore 1 values, so the block must be erased before any cell is reprogrammed.
- An entire page is programmed at once.
- To program a page:
  - bitlines are driven with the data values: 0 V for a logic 0 and 8 V for a logic 1.
  - substrate is held at ground.
  - wordline is set to 20 V for the page being programmed and 10 V for the other pages in the block.
  - ground select line (gsl) is left OFF but the string select line (ssl) for the block is turned ON, passing the voltage on the bitline to the channels of all the transistors being programmed.
  - Thus, cells being programmed to 0 see 20 V on the control gate and 0 V on the channel.
- This high voltage difference induces FN tunneling that drives electrons onto the floating gate, raising Vt to a positive voltage. The other cells see a smaller voltage that is insufficient to cause tunneling.

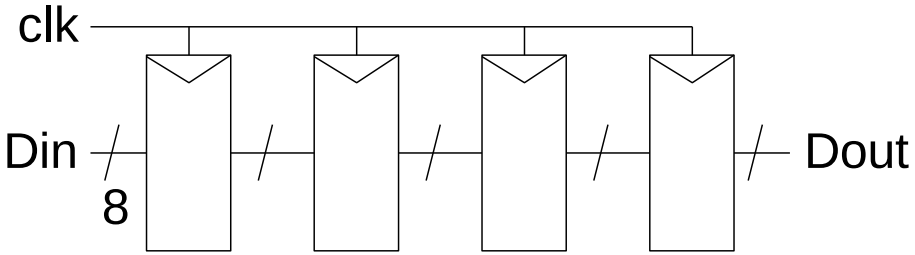


12.5 Serial Access Memories

- Serial access memories do not use an address
  - Shift Registers
  - Tapped Delay Lines
  - Serial In Parallel Out (SIPO)
  - Parallel In Serial Out (PISO)
  - Queues (FIFO, LIFO)

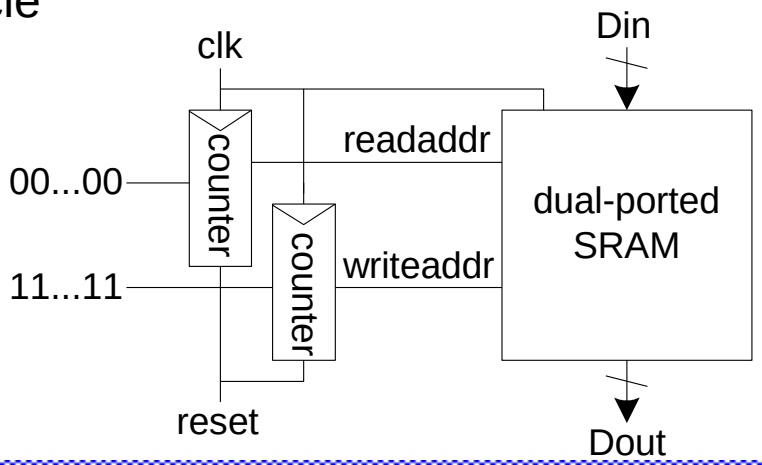
12.5.1 Shift Register

- *Shift registers* store and delay data
  - commonly used in signal-processing applications to store and delay data.
- Simple design:
  - a simple 4-stage 8-bit shift register constructed from 32 flip-flops.
  - No logic between the registers Watch your hold times!



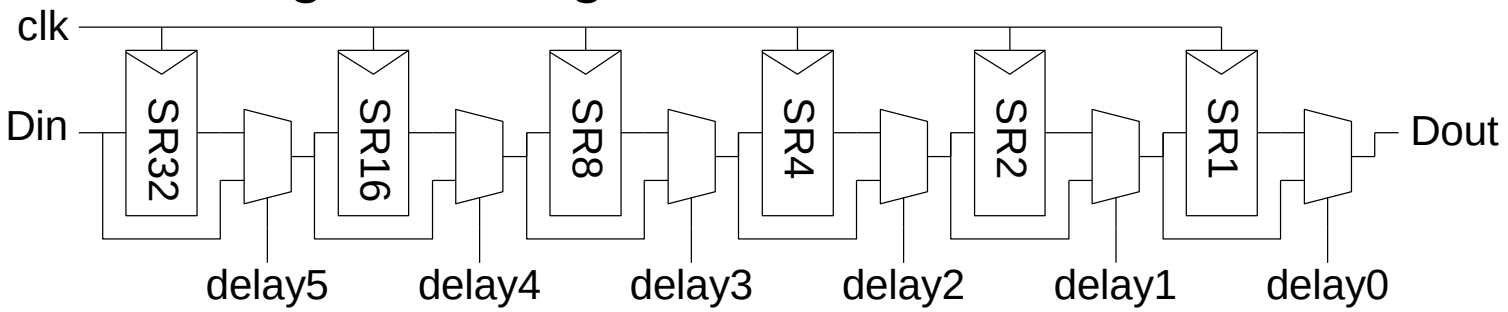
Denser Shift Registers

- Flip-flops aren't very area-efficient
- For large shift registers, keep data in SRAM instead
- The RAM is configured as a circular buffer with a pair of counters specifying where the data is read and written.
- Move read/write pointers to RAM rather than data
  - The read counter is initialized to the first entry and the write counter to the last entry on reset
  - Increment address on each cycle



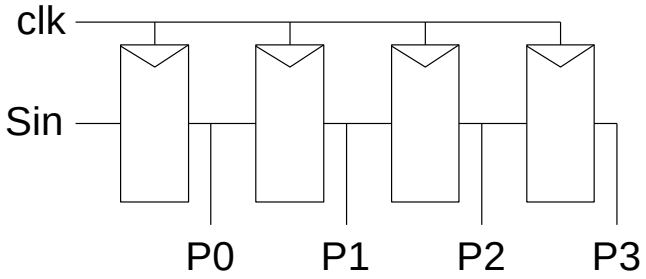
Tapped Delay Line

- A *tapped delay line* is a shift register with a programmable number of delay stages
- Multiplexers control pass-around of the delay blocks to provide the appropriate total delay.
- The Figure shows a 64-stage tapped delay line .
  - Delay blocks are built from 32-, 16-, 8-, 4-, 2-, and 1-stage shift registers.

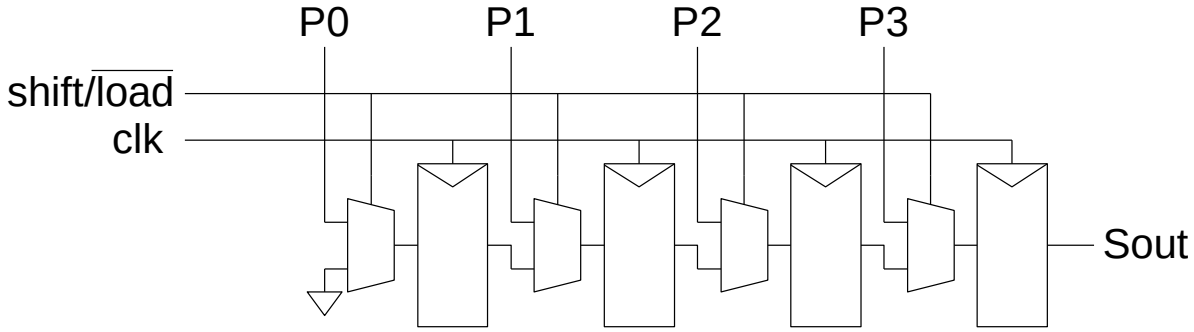


Serial In Parallel Out & Parallel In Serial Out

- **Serial In Parallel Out**: 1-bit shift register reads in serial data
  - After N steps, presents N-bit parallel output

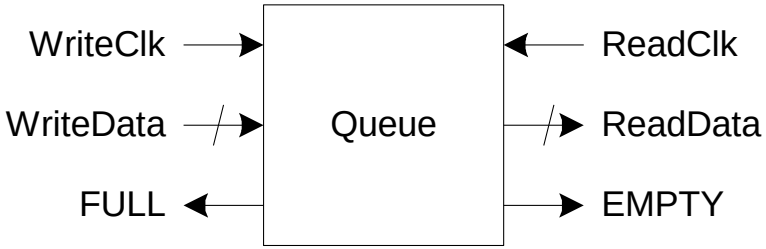


- **Parallel In Serial Out**: Load all N bits in parallel when shift = 0
  - Then shift one bit out per cycle



12.5.2 Queues

- *Queues* allow data to be read and written at different rates.
- Below figure shows an interface to a queue:
  - Read and write operations each are controlled by their own clocks that may be **asynchronous**
  - **FULL flag** means there is no room remaining to write data
  - **EMPTY flag** means there is no data to read.
- Build with SRAM and read/write counters (pointers)
  - The queue internally maintains read and write pointers indicating which data should be accessed next.
  - The queue internally maintains read and write pointers indicating which data should be accessed next.
- Some queues also provide **ALMOST-FULL** and **ALMOST-EMPTY** flags to communicate the impending state and halt write or read requests.



FIFO, LIFO Queues

First In First Out **(FIFO)**

Used to buffer data between two asynchronous streams.
Organized as a circular buffer.

On reset,

the read and write pointers are both initialized to the first element
FIFO is EMPTY.

On a write,

write pointer advances to the next element.
If it is about to catch the read pointer, the FIFO is FULL.

On a read,

the read pointer advances to the next element.
If it catches the write pointer, the FIFO is EMPTY again.

Last In First Out **(LIFO)**

- Known as stacks, are used in applications such as subroutine or interrupt stacks in microcontrollers.
- Uses a single pointer for both read and write.

On reset,

the pointer is initialized to the first element
LIFO is EMPTY.

On a write,

the pointer is incremented.
If it reaches the last element, the LIFO is FULL.

On a read,

the pointer is decremented.
If it reaches the first element, the LIFO is EMPTY again.

12.6 Content-Addressable Memory (CAM)

- The CAM is an extention of SRAM
  - Can be read or written given *adr and data*
  - But also performs matching operations.
- Matching asserts a matchline output for each word of the CAM that contains a specified key.
- A common application of CAMs is translation lookaside buffers (TLBs)
  - The virtual address is given as the key to the TLB CAM.
  - If this address is in the CAM, the corresponding matchline is asserted.
  - This matchline can serve as the wordline to access a RAM containing the associated physical address, as shown in Figure 12.68.
  - A NOR gate processing all of the matchlines generates a miss signal for the CAM.

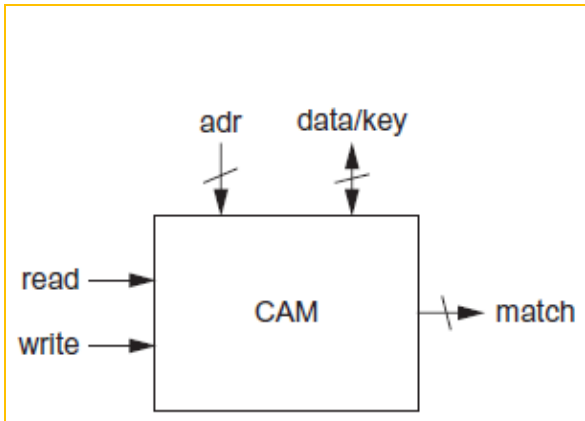


FIGURE 12.67 Content-addressable memory

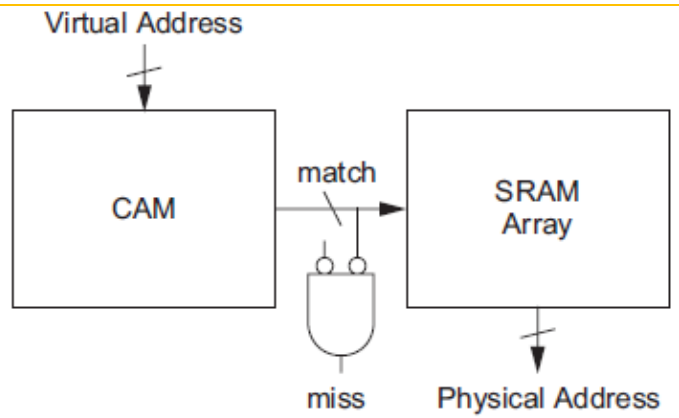
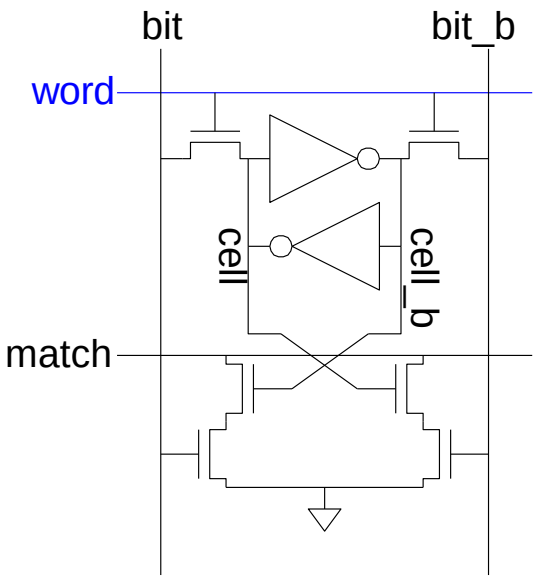
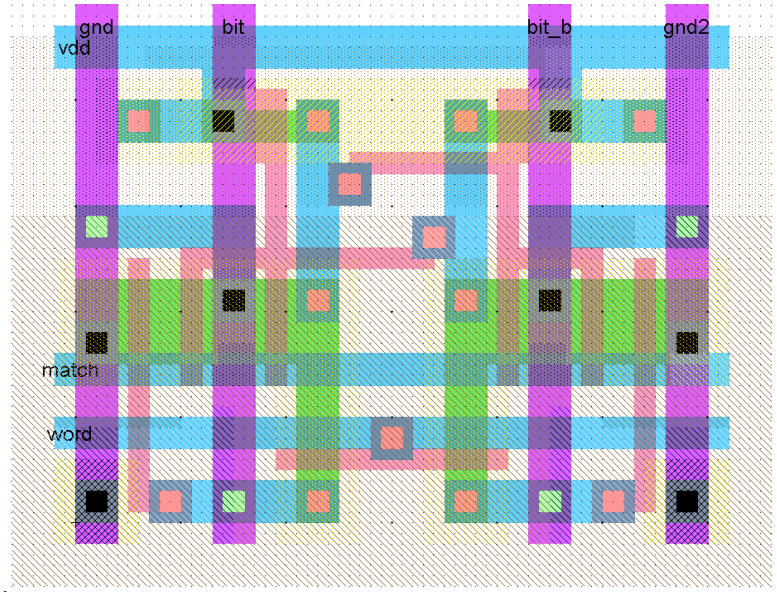


FIGURE 12.68 Translation Lookaside Buffer (TLB)

10T CAM Cell (read)

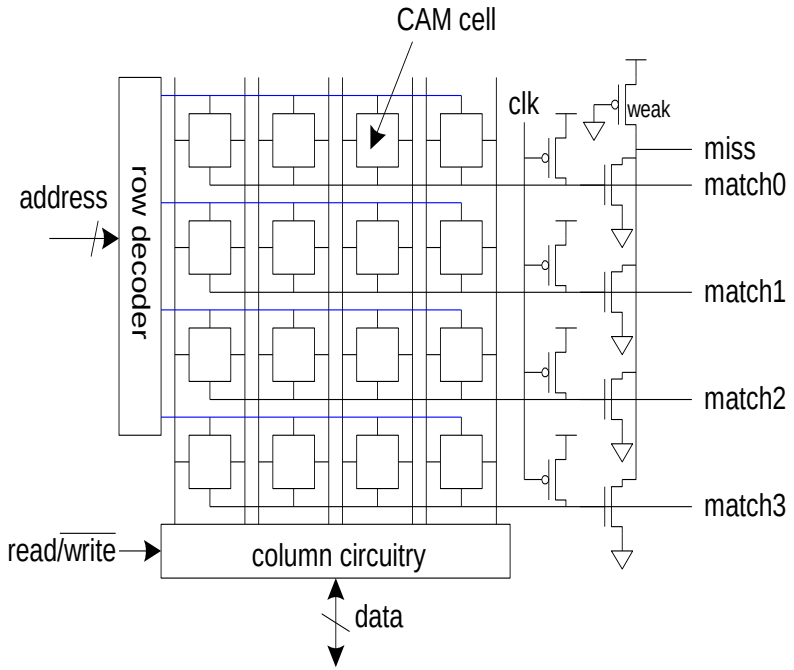
- Add four match transistors to 10T SRAM
  - 56 x 43 λ unit cell





CAM Cell Operation (read)

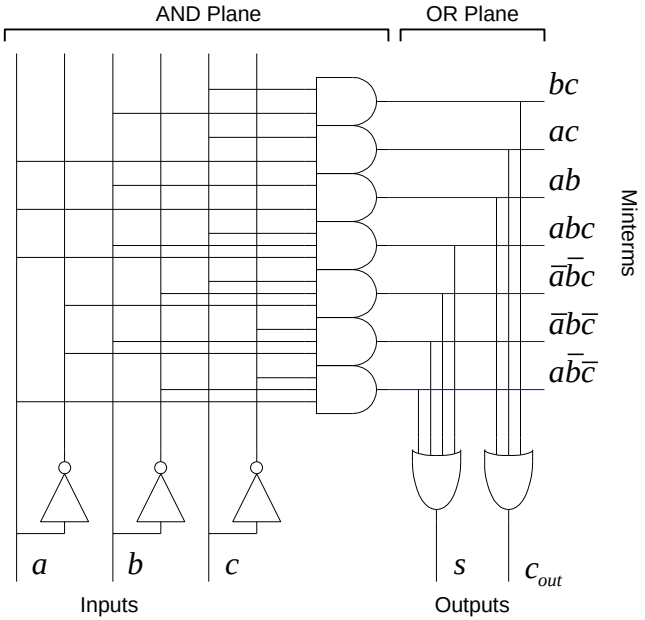
- Read and write like ordinary SRAM
- For matching:
  - Leave wordline low
  - Precharge matchlines
  - Place key on bitlines
  - Matchlines evaluate



- Miss line
  - Pseudo-nMOS NOR of match lines
  - Goes high if no words match

12.7 Programmable Logic Array

- A *PLA* performs any function in sum-of-products form (combinational logic circuit).
- *Literals*: inputs & complements
- *Products / Minterms*: AND of literals
- *Outputs*: OR of Minterms



- Example: Full Adder

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc$$

$$c_{\text{out}} = ab + bc + ac$$

NOR-NOR PLAs

- ANDs and ORs are not very efficient in CMOS
- Dynamic or Pseudo-nMOS NORs are very efficient
- Use DeMorgan's Law to convert to all NORs

PLA Example

Example 12.5

Write the equations for a full adder in sum-of-products form. Sketch a 3-input, 2-output PLA implementing this logic.

SOLUTION: Figure 12.75 shows the PLA. The logic equations are

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc \qquad (12.9)$$
$$c_{out} = ab + bc + ac$$

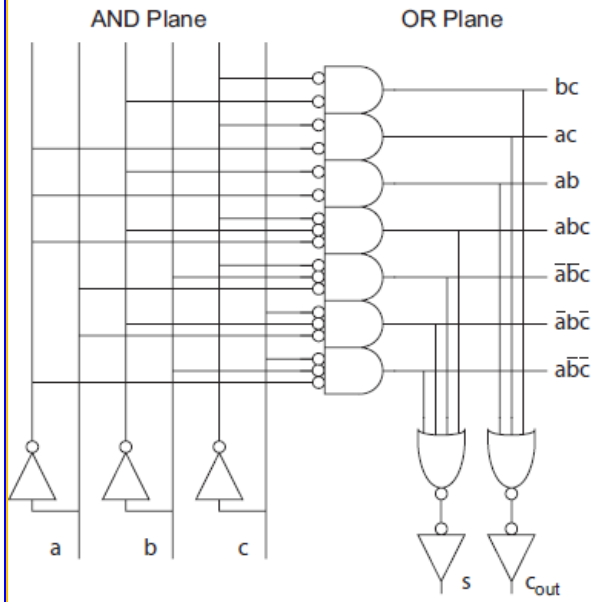


FIGURE 12.75 AND/OR representation of PLA

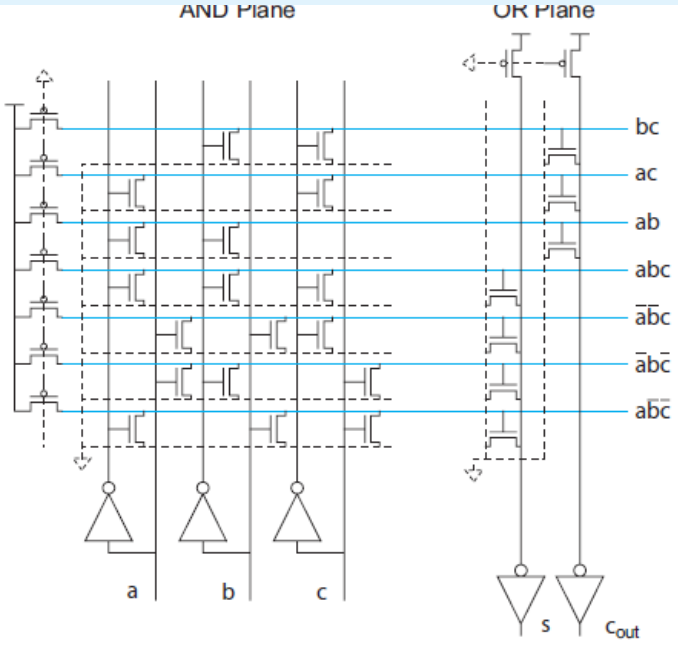


FIGURE 12.76 Pseudo-nMOS PLA schematic

PLA Example: Dot Diagram

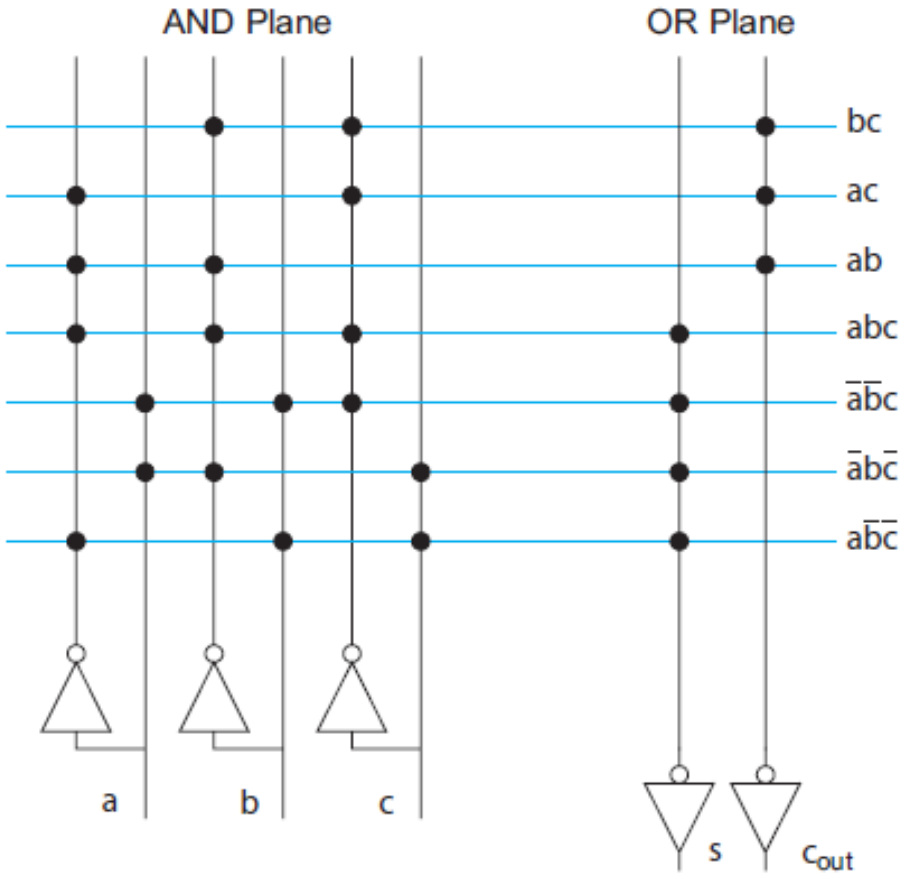


FIGURE 12.74 Dot diagram representation of PLA

PLAs vs. ROMs

- The OR plane of the PLA is like the ROM array
- The AND plane of the PLA is like the ROM decoder
- PLAs are more flexible than ROMs
  - No need to have 2^n rows for n inputs
  - Only generate the minterms that are needed
  - Take advantage of logic simplification