



# CS 3530 Operating Systems

## Lecture Subject/Title

CS 3530 Operating Systems. Dr. Ken Hoganson, Copyright © 2008

## Memory Page Replacement

- Objective is to approximate Least Recently Used, as a predictor of future page use.
- Useful ideas for virtual memory and caching – both replace pages

### Two Example Strategies

- Reference Bits
  - use a set of bits to track the usage of a page, then make page replacement decisions based on the tracking data
- Second Chance Algorithm
  - use a single bit to track usage

- Implemented in hardware
- Uses a single bit to track recent access (per page)
- Uses a set of reference bits to store access “history” (per page)
  1. Every time a page is accessed, its accessed bit is “set”
  2. Periodically, the access bit is merged with the reference bits
    1. Shift reference bits to the right
    2. Add access bit in high-order bit position
    3. Reset the access bit
  3. The reference bits can be used to decide which page has been least recently accessed.
    - High number is more recently access than low number

- Pages are listed in a FIFO queue
- Each page has a single “accessed” bit
- When need to swap out a page – To find the page to replace
  - Start at top of list
  - If page’s reference bit is set, skip this page (but reset the bit to zero)
  - If page’s reference bit is reset, swap this page
  - If reached the bottom of the list
    - (all bits were set, but have now been reset),
    - start over at the top (and replace that one)

- Optimal
- FIFO
- LRU
- Ref bits
- Second Chance FIFO
- Belady's Anomaly – possible to get unusual results with particular sequences of page references for a particular method





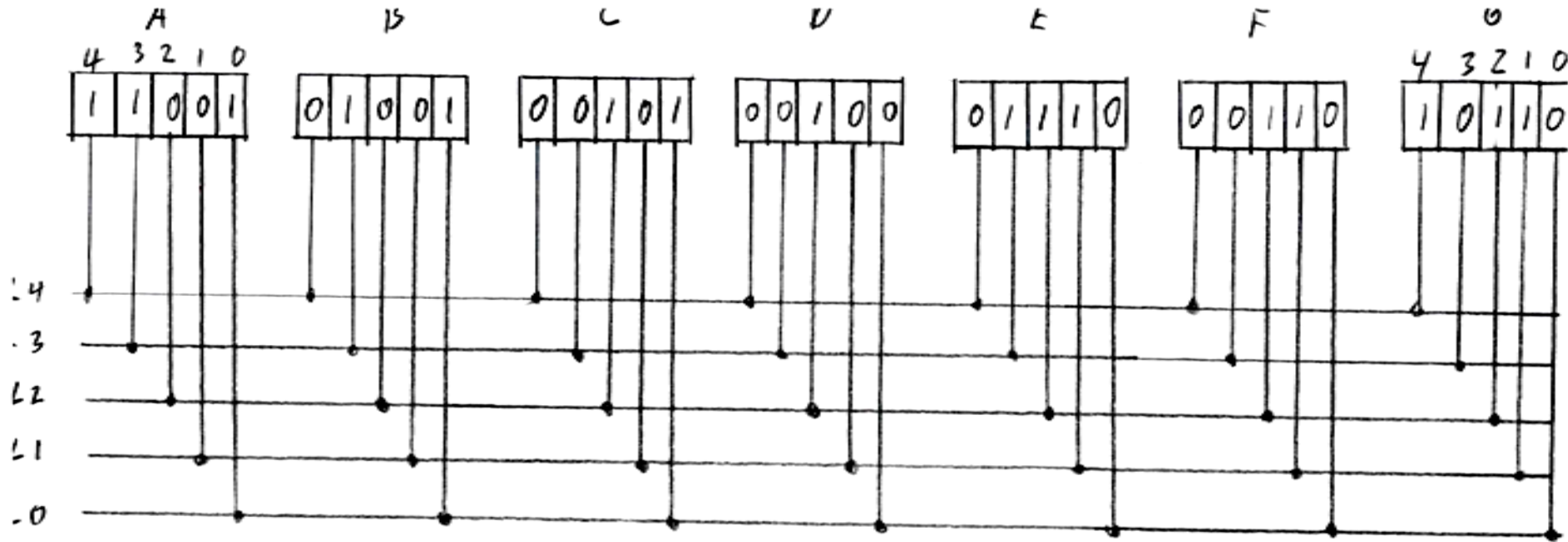


- Implemented in hardware
- Uses a single bit to track recent access (per page)
- Uses a set of reference bits to store access “history” (per page)
  1. Every time a page is accessed, its accessed bit is “set”
  2. Periodically, the access bit is merged with the reference bits
    1. Shift reference bits to the right
    2. Add access bit in high-order bit position
    3. Reset the access bit
  3. The reference bits can be used to decide which page has been least recently accessed.
    - High number is more recently access than low number



- Pages are listed in a FIFO queue
- Each page has a single “accessed” bit
- When need to swap out a page – To find the page to replace
  - Start at top of list
  - If page’s reference bit is set, skip this page (but reset the bit to zero)
  - If page’s reference bit is reset, swap this page
  - If reached the bottom of the list
    - (all bits were set, but have now been reset),
    - start over at the top (and replace that one)



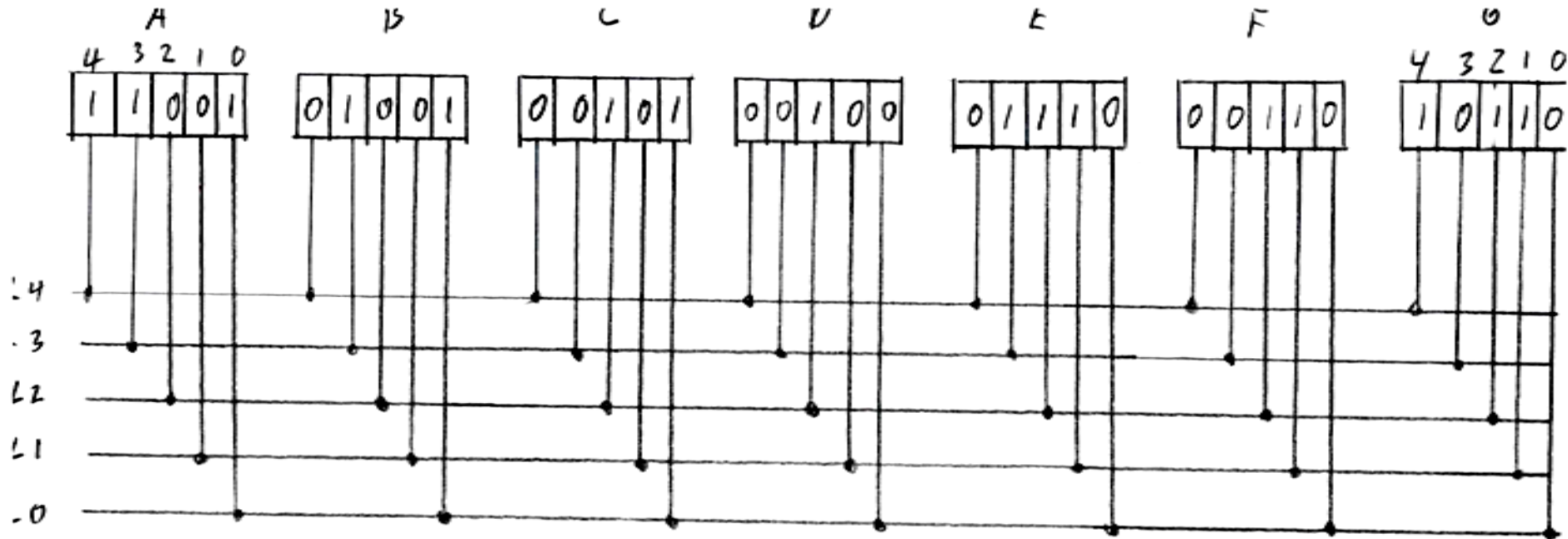


Illustrates 8 values to compare.

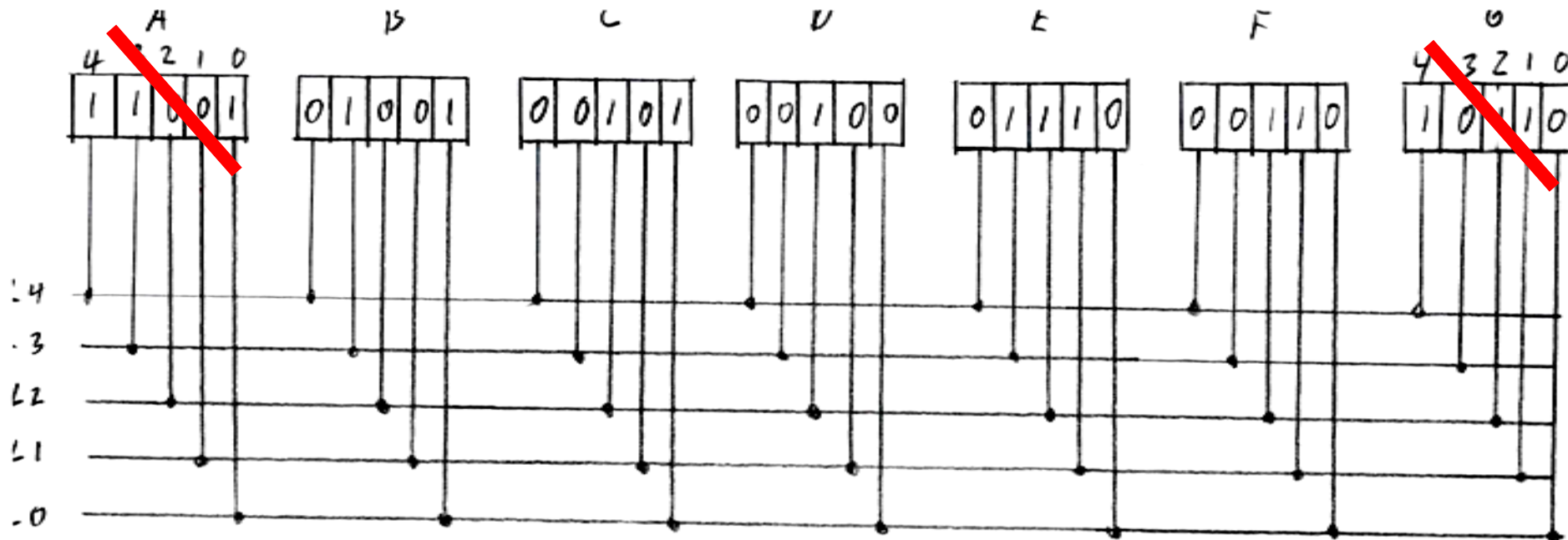
A set of extra lines/bus L0-L4 (5 in this case)

Algorithm is iterative, one iteration per bit (5 bits)

- Parallel LRU approximation, reference bit comparison by self-selection
- Each bit is asserted (set to 1) to a line if it is a 1
- Each object, concurrently does:
  1. Check L4: If L4=1, then drop the assertion of your Lines. If ANY other line (L0-L4) is 1, then drop out (someone is lower than you).
  2. Check L3: If L3 = 1, then drop the assertion of your lines. If ANY other line (L0-L4) is 1, then drop out (someone is lower than you).
  3. Repeat through L0

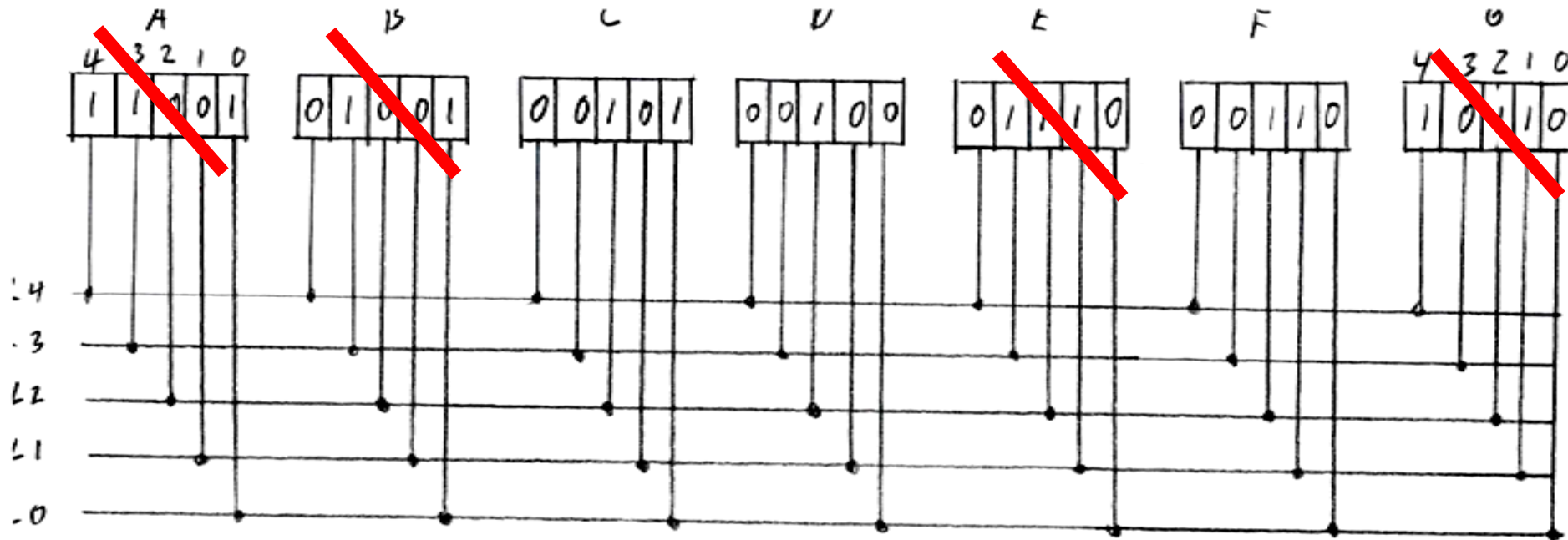


Line 4: A and G are asserting, they drop L4, detect other lines being asserted, so drop out of competition



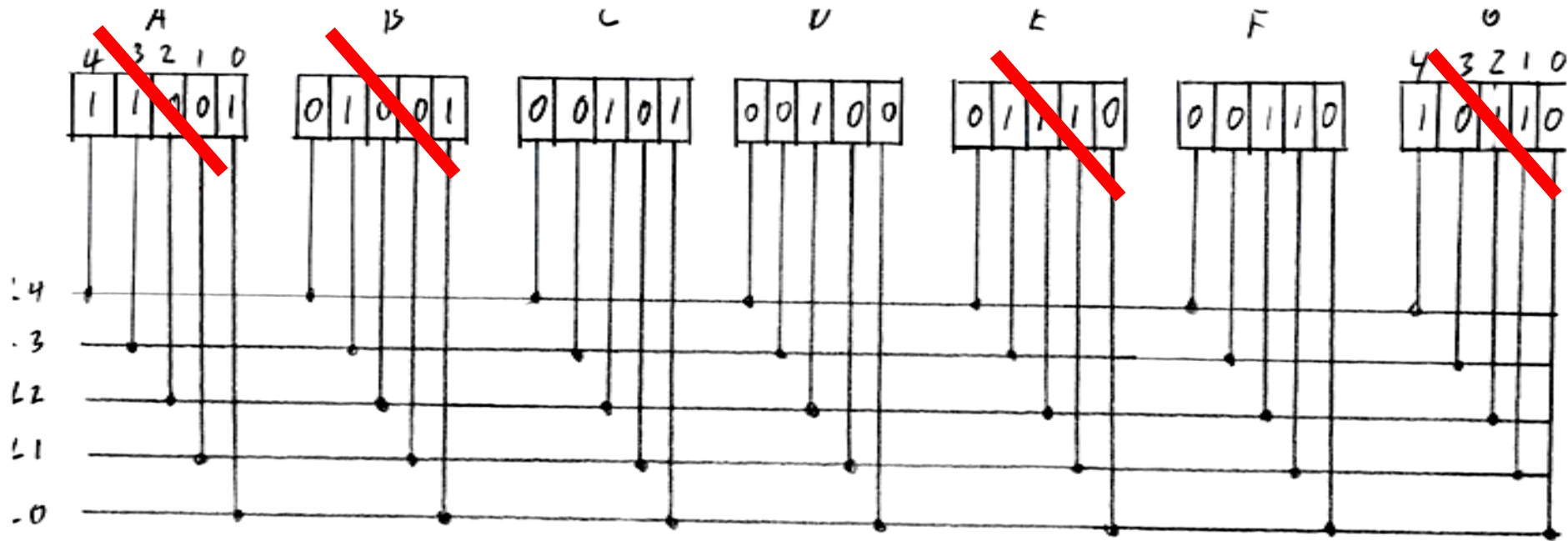
Line 4: A and G are asserting, they drop L4, detect other lines being asserted, so drop out of competition

Line 3: B and E are asserting, they drop all lines, detect other lines being asserted, so drop out of competition



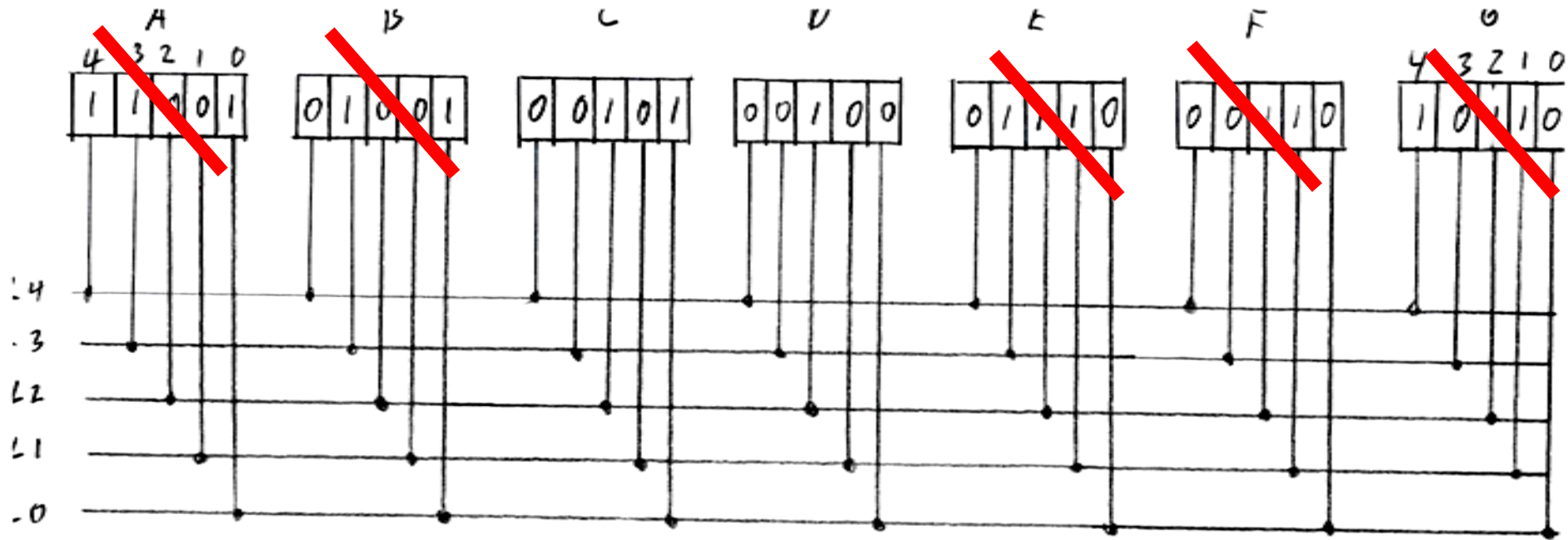
Line 3: B and E are asserting, they drop all lines, detect other lines being asserted, so drop out of competition

Line 2: C, D, and F are asserting. They drop all lines, no other lines are asserted, so they do NOT drop out.



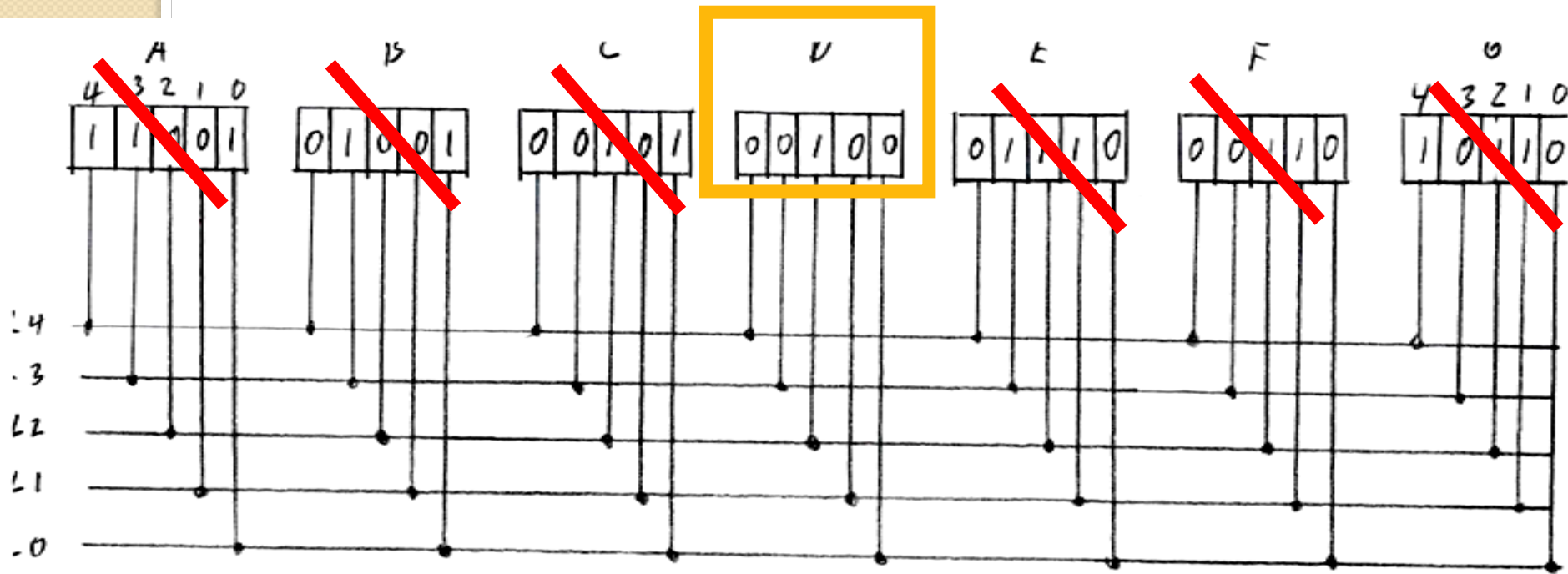
Line 2: Line 2 = 0, no-one drops out

Line 1: F is asserting, it drops its lines, detects other lines being asserted, so drops out of competition.



Line 1: F is asserting, it drops L1, detects other lines being asserted, so drops out of competition

Line 0: C is asserting, it drops it lines, detects other lines being asserted, so drops out of competition



Line 0: C is asserting, it drops L0, detects other lines being asserted, so drops out of competition

Last one still standing is D, which is correct, is the lowest number in reference bits.

- If  $n$  is the number of pages or objects we need to select from (8 in this case),
- If  $m$  is the # of reference bits (5 in this case),
- Then the number of iterations is related to  $m$  (order ( $m$ ) work) rather than  $n$ .
- Efficient compared to alternatives.
  
- Still need mechanism to break ties (perhaps arbitrary order).



**End  
Of  
Today's  
Lecture.**



This slide intentionally left blank