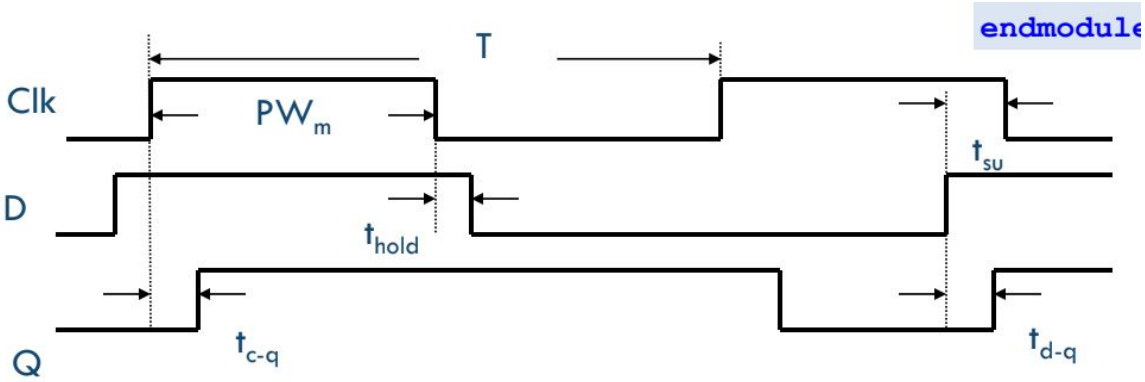
Discussion 13

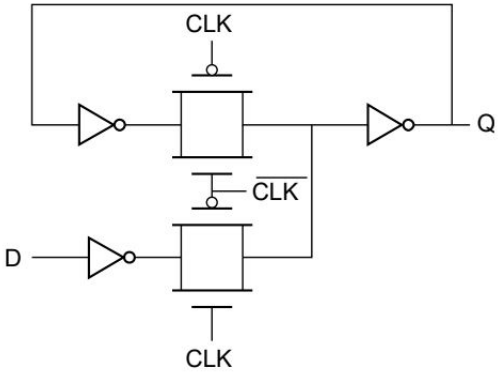
Timing, FF/Latch Design, SRAMs, Caches

Latch Timing

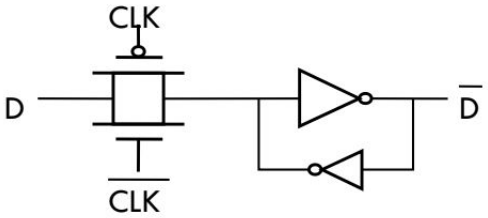


- A positive latch is transparent (q = d) when the clock is high and opaque (q = d, during negedge clock) when the clock is low
- t_{d->q} is the delay from d to q when the latch is transparent
- t_{clk->q} is the delay from the rising clock edge to d propagating to q

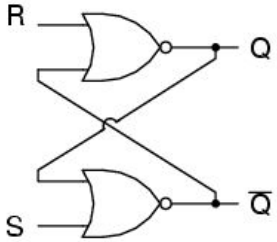
Latch Circuits



‘Feedback-breaking’ latch
Transparent high



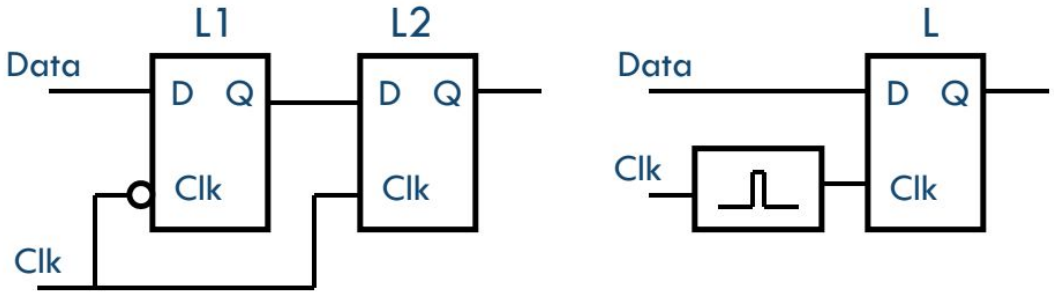
‘State-forcing’ latch
Transparent low



S	R	Q	$\overline{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

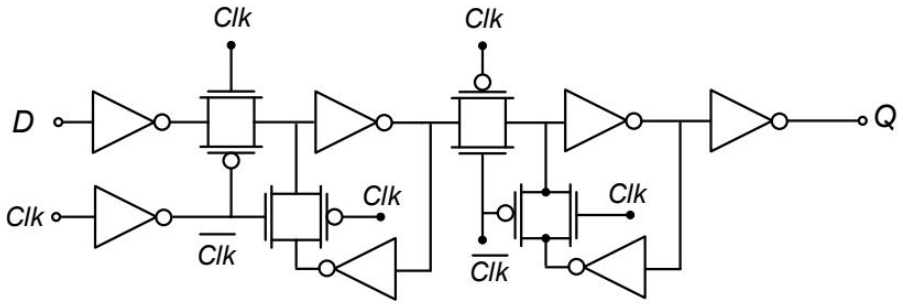
SR latch
Common interview question

Building a Flip-Flop from Latches



- Clock pulsed latch
  - Latch becomes transparent for a short time and holds the value it received on the pulse
  - Not common anymore, sometimes used in high performance circuits
  - Positive hold time
- Master-slave latches
  - Commonly used technique. L2 holds output data stable when clock is high. L2
  - Negative hold time

Flip-Flop Hold/Setup/clk->q Time



- This is a negative edge flip-flop as drawn
- We'll consider the positive edge case

- Hold time = the amount of time after a clock edge that the data input needs to be stable for (can be negative)
- Setup time = the amount of time before a clock edge that the data input needs to be stable to be properly latched internally
- Clk-q time = delay from a clock edge to q being updated with the new value written

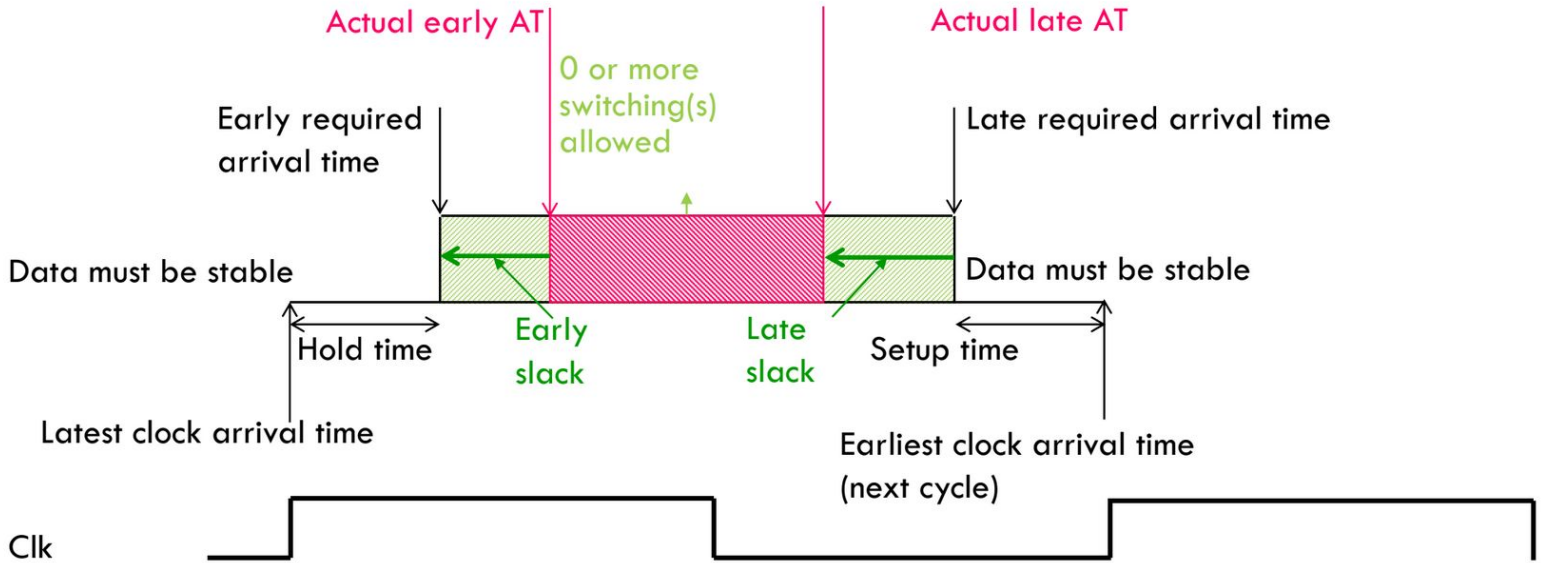
Path Timing Constraints

Setup constraint: T_{clk} > t_{clk->q} + t_{logic,max} + t_{setup}

- The clock period must be greater than the delay of the critical path

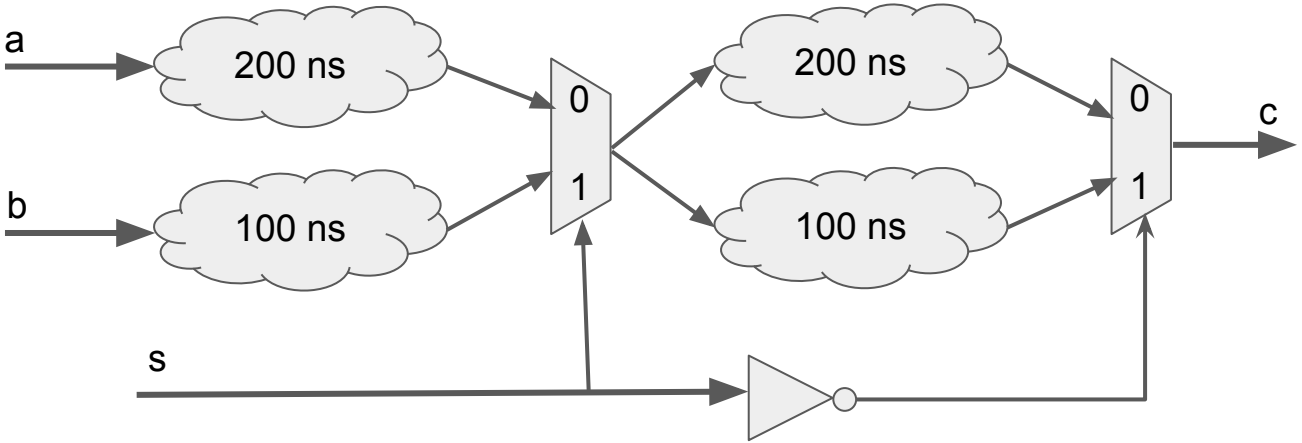
Hold constraint: t_{hold} < t_{clk->q} + t_{logic,min}

- The minimum logic delay must be greater than the hold time



False Paths

- Be careful about finding the critical path by statically adding up delays
- Some paths may not be exercised based on logic expressions
- Here, the critical path is 300ns, not 400ns



Clock Skew

Setup constraint: $T_{clk} > t_{clk->q} + t_{logic,max} + t_{setup}$
Hold constraint: $t_{hold} < t_{clk->q} + t_{logic,min}$

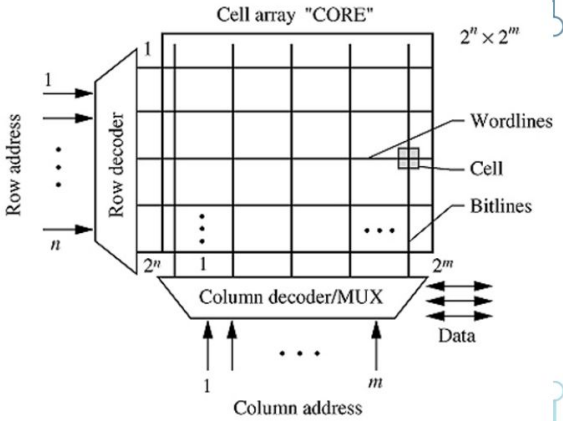
- Skew is the deterministic clock arrival time difference between 2 flops
- Positive skew = receiving edge arrives later than nominal
- Negative skew = receiving edge arrives earlier than nominal
- New timing equations:
  - $T_{clk} > t_{clk->q} + t_{logic,max} + t_{setup} - t_{skew}$
    - Note positive skew can improve clock frequency
    - Negative skew hurts setup margin
  - $t_{hold} + t_{skew} < t_{clk->q} + t_{logic,min}$
    - Note positive skew hurts hold margin

Clock Jitter

- Jitter is the non-deterministic difference in clock arrival times
  - Can be treated like skew in timing calculations
  - Assuming worst case jitter in the unfavorable direction for each timing calculation
  - Can lump the jitter of both the launching and receiving FFs into an equivalent skew

SRAM Architecture

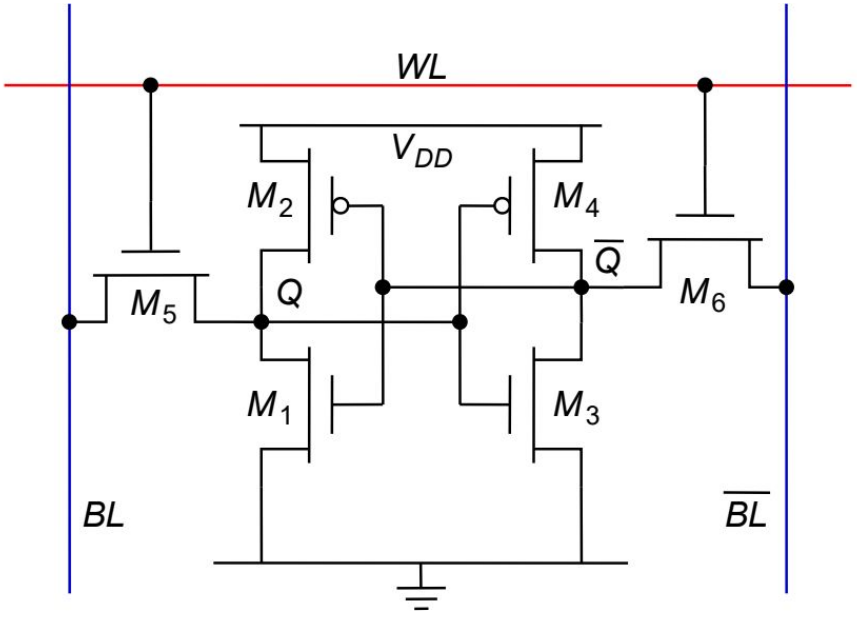
- CORE
  - Wordlines to access rows
  - Bitlines to access columns
  - Data multiplexed onto columns
- Decoders
  - Addresses are binary
  - Row/column MUXes are 'one-hot' - only one is active at a time



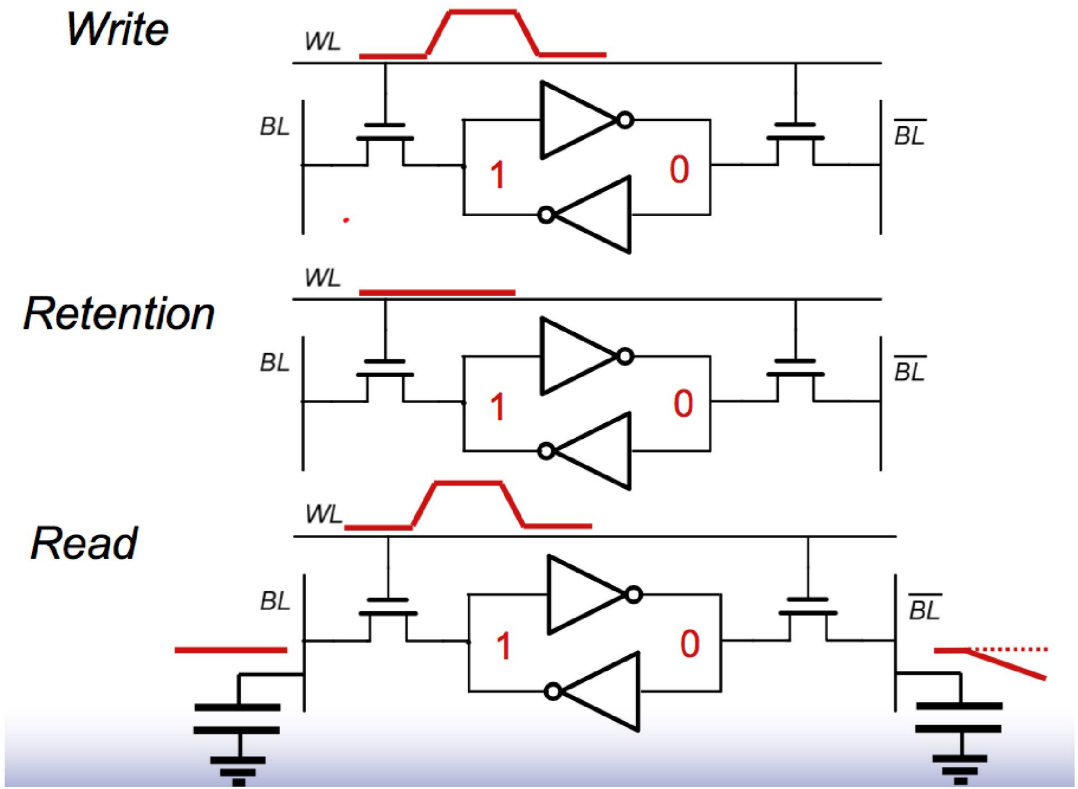
- SRAM bitcells arranged in a grid
- Bitlines (vertical) are shared across cells in a column, they are often long wires with a large capacitive load (connect to drains of access transistors)
- Wordlines (horizontal) are shared across cells in a row, they connect to the gates of access transistors
- Peripheral circuitry (bitline drivers, sense amp, decoders)

6T SRAM Cell

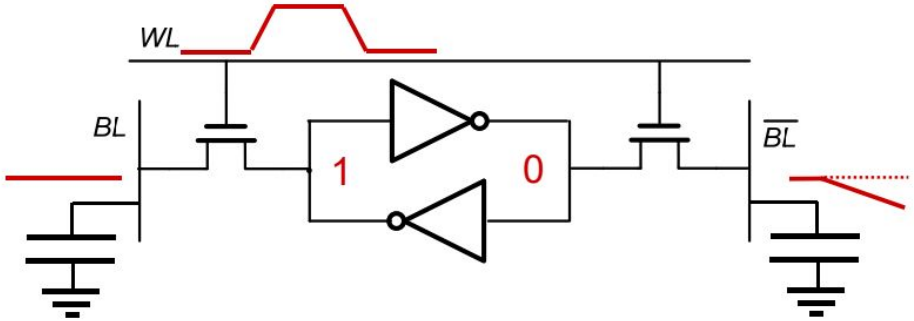
- Inverters in positive feedback form the memory element (similar to a latch)
- M5/M6 are the access transistors; they allow the bitlines to access the memory nodes (Q, Qbar) when WL = 1
- Only 1 WL in an SRAM array is active at a time and it addresses an entire row of SRAM cells
- Bitlines are controlled differently for read and write



6T SRAM Cell - 3 Modes of Operation

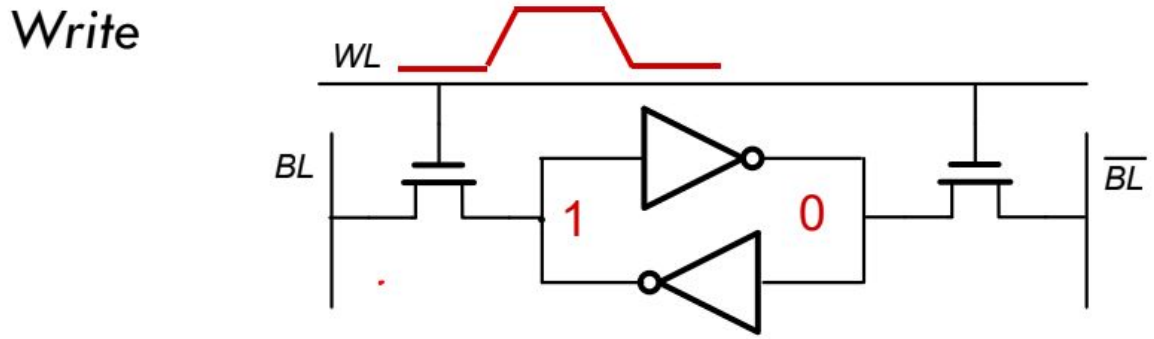


SRAM Read



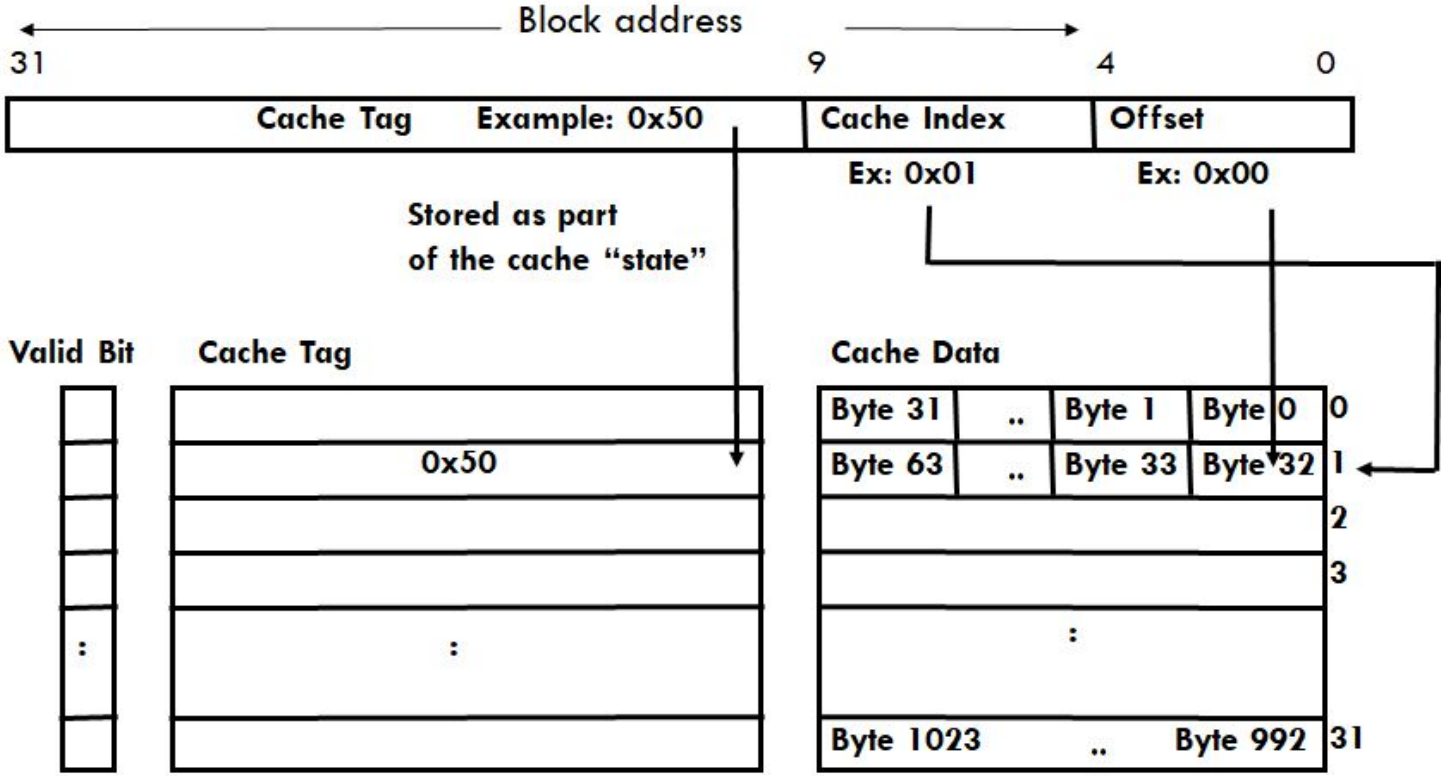
- 1) precharge BL and BLbar to VDD, 2) raise WL, 3) sense dip on one bitline with sense amp, 4) lower WL, 5) discharge bitlines
- Read stability = reading doesn't corrupt the value stored in Q and Qbar
  - The pass transistor shouldn't overpower the node storing a '0' and flip its state (consider voltage divider from bitline to Q)
- We choose to make the NMOSes in the inverters stronger than the pass transistor = (Wn > Wpass) to prevent read corruption

SRAM Write

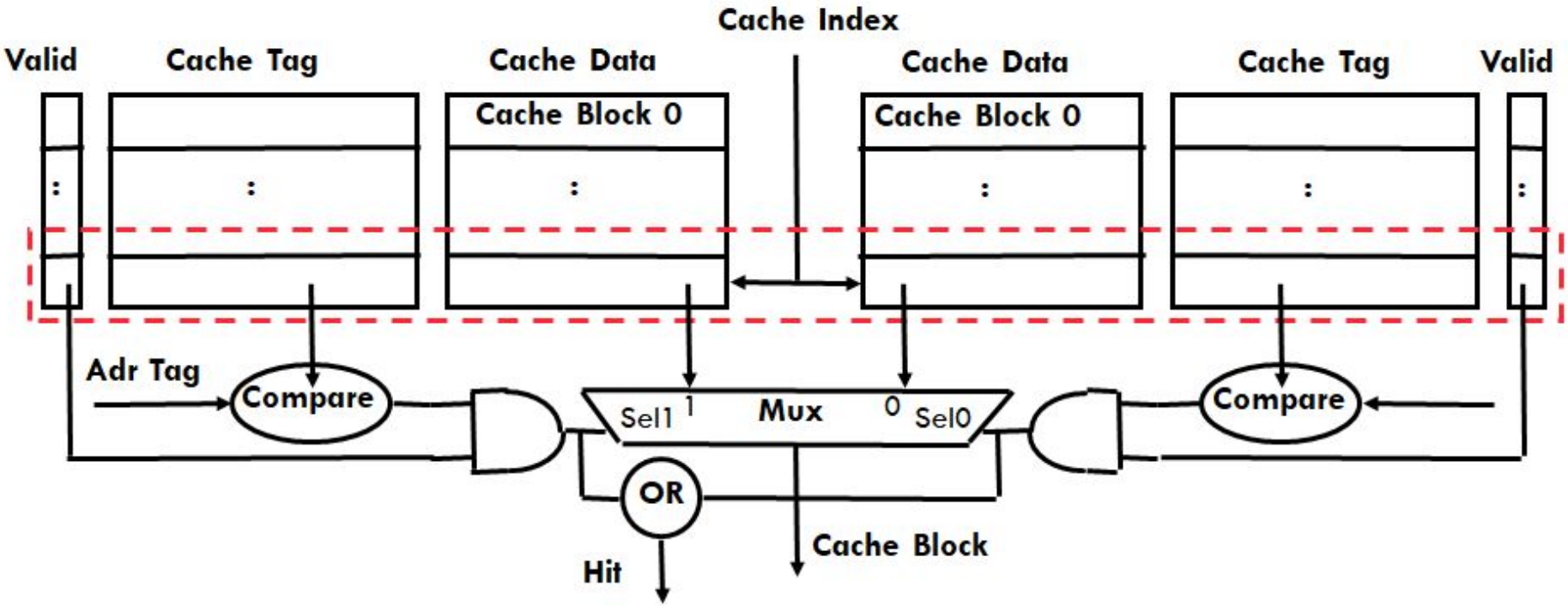


- 1) drive BL and BLbar with new values, 2) raise WL, 3) wait some time (write time), 4) lower WL, 5) discharge bitlines
- Write-ability = the cell's memory value can be changed
  - requires the pass transistor overpower one of the data nodes
  - If we assume the cell is read stable, the inverter NMOS is stronger than the pass transistor. This means the node with '0' can't be overpowered => *so we must overpower PMOS*.
- Pass transistor strength > PMOS pullup strength = (Wpass > Wp)
  - Voltage divider on '1' node must be strong enough to cause inverters to switch

Caches - Direct Mapped



Caches - N-Way Set Associative



Caches - Fully Associative