
Computer Science 141

Computing Hardware

Fall 2009

Harvard University

Instructor: Prof. David Brooks

dbrooks@eecs.harvard.edu

Sequential Circuits

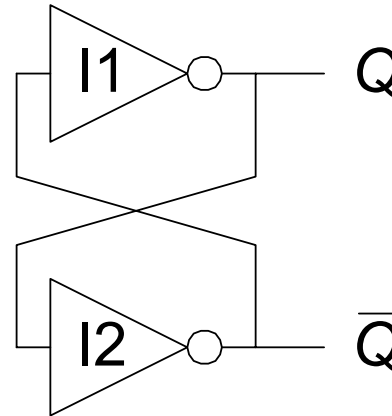
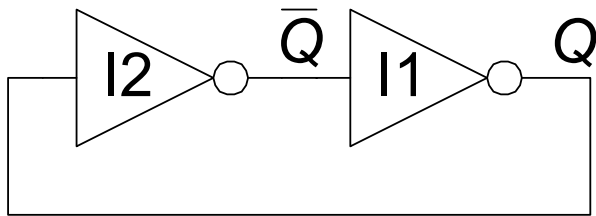
- Next 3-4 lectures on sequential logic and FSM design
 - Today
 - Motivate sequential circuits
 - Discuss state elements
-

Sequential Logic

- circuits with feedback
 - outputs = $f(\text{current inputs} + \text{entire input history})$
 - input history summarized by state information
 - state information stored in memory elements
 - Bistable circuits
 - SR Latch
 - JK Latch
 - D Latch
 - D Flip-Flop
-

Bistable Circuit

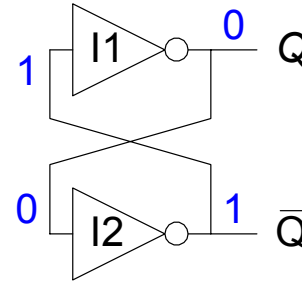
- Fundamental building block of other state elements
- Two outputs: Q, \overline{Q}
- No inputs



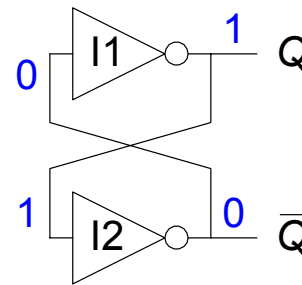
Bistable Circuit Analysis

- Consider the two possible cases:

- $Q = 0$: then $\bar{Q} = 1$ and $Q = 0$ (consistent)



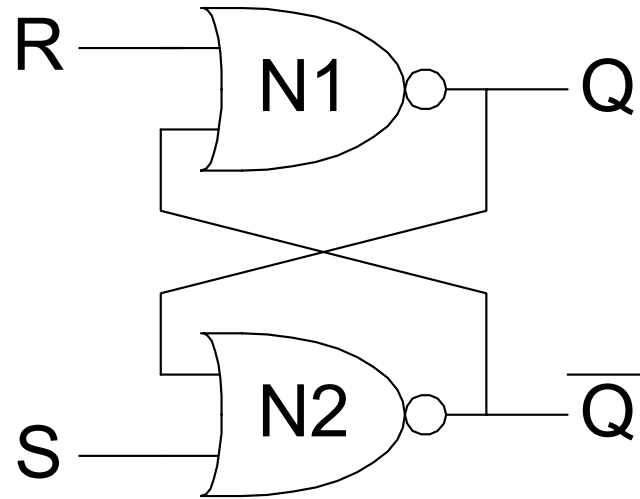
- $Q = 1$: then $\bar{Q} = 0$ and $Q = 1$ (consistent)



- Bistable circuit stores 1 bit of state in the state variable, Q (or \bar{Q})
- But there are **no inputs to control the state**

SR (Set/Reset) Latch

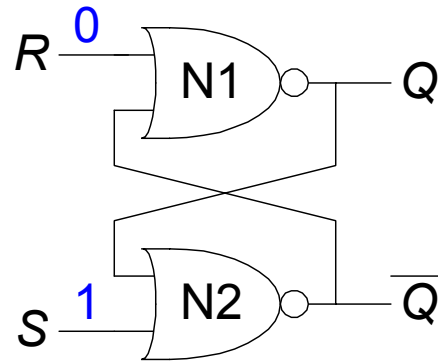
- SR Latch



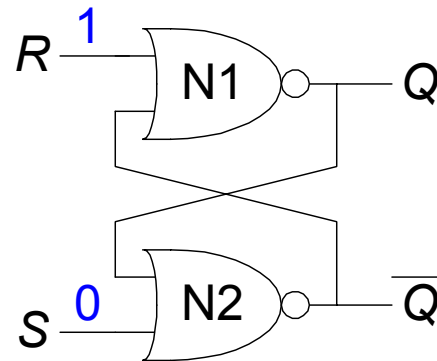
- Consider the four possible cases:
 - $S = 1, R = 0$
 - $S = 0, R = 1$
 - $S = 0, R = 0$
 - $S = 1, R = 1$
-

SR Latch Analysis

– $S = 1, R = 0$: then $Q = 1$ and $\overline{Q} = 0$

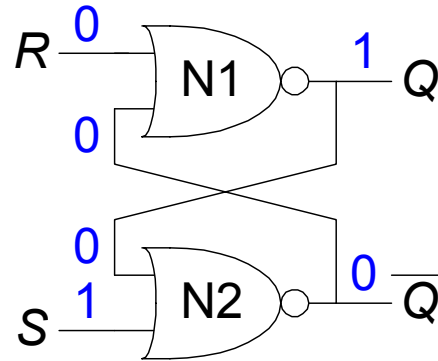


– $S = 0, R = 1$: then $Q = 0$ and $\overline{Q} = 1$

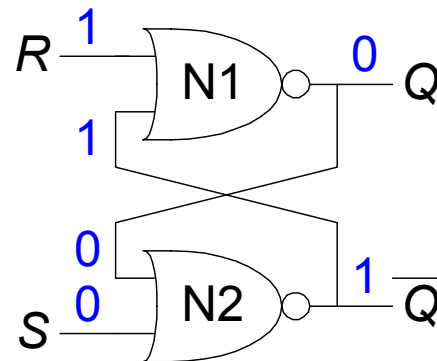


SR Latch Analysis

– $S = 1, R = 0$: then $Q = 1$ and $\overline{Q} = 0$



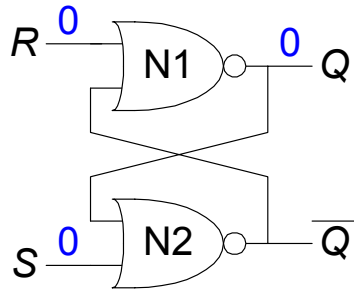
– $S = 0, R = 1$: then $Q = 0$ and $\overline{Q} = 1$



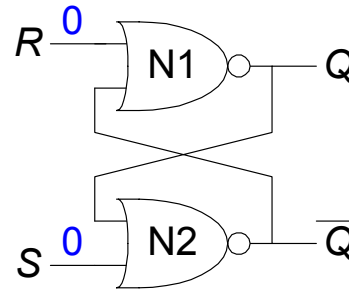
SR Latch Analysis

– $S = 0, R = 0$: then $Q = Q_{prev}$

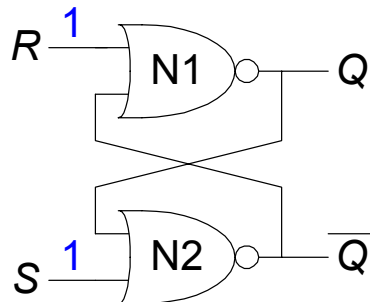
$Q_{prev} = 0$



$Q_{prev} = 1$

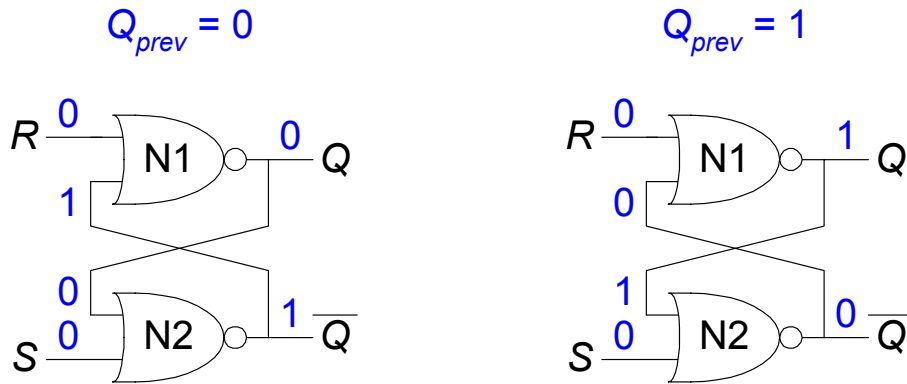


– $S = 1, R = 1$: then $Q = 0$ and $\bar{Q} = 0$

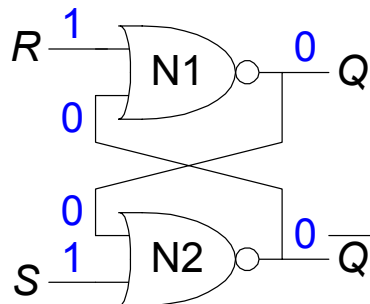


SR Latch Analysis

- $S = 0, R = 0$: then $Q = Q_{prev}$ and $\bar{Q} = \bar{Q}_{prev}$ (**memory!**)

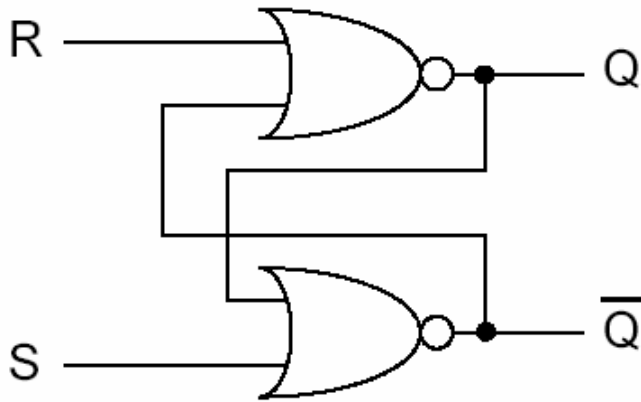


- $S = 1, R = 1$: then $Q = 0$ and $\bar{Q} = 0$ (**invalid state: $\bar{Q} \neq \text{NOT } Q$**)



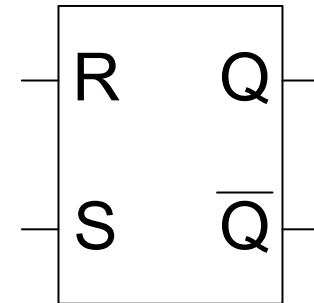
SR Latch

- Set/Reset Latch: Stores one bit of state (Q)



| S | R | Q |
|---|---|----------|
| 0 | 0 | hold |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | unstable |

SR Latch Symbol



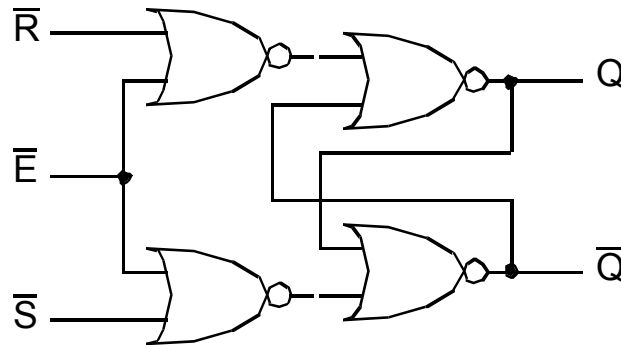
Both Q and Q' driven to 0,
thus $S=R=1$ is forbidden

$$Q^+ = S + \bar{R}Q$$

- characteristic equation:
- state diagram can also describe behavior

Transparent Latch

- transparent latch = output changes immediately to reflect input
 - unclocked latch = latch only has data inputs (e.g. R and S)
 - level-sensitive latch = latch has additional enable (clock) input



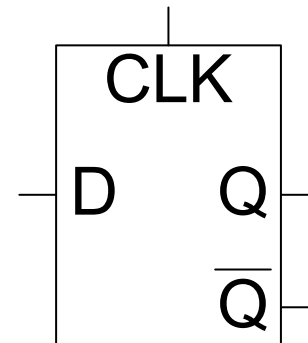
Enable ($E=0$), first set of NOR's act as inverters

!Enable ($E=1$), first set of NOR's output 0's, holding SR latch

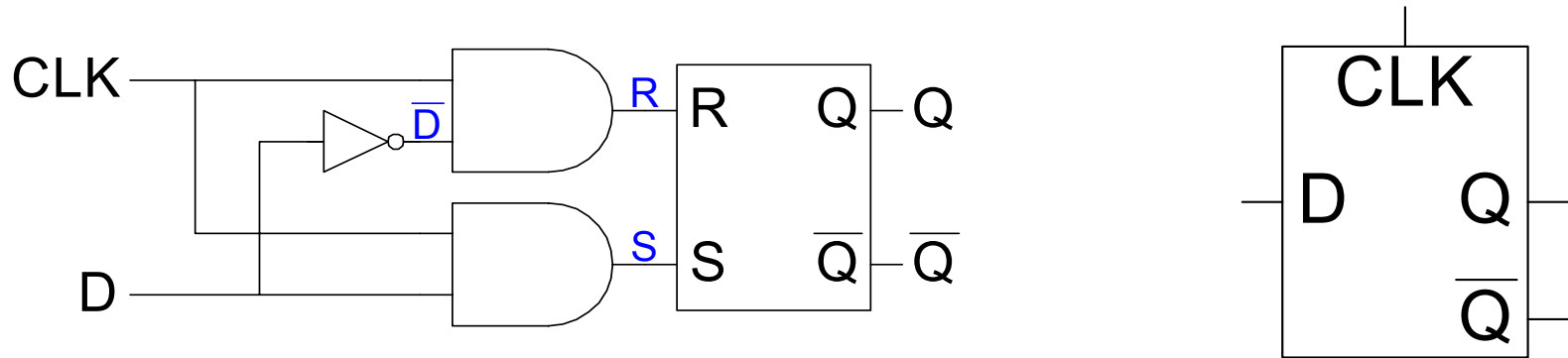
D Latch

- Two inputs: CLK , D
 - CLK : controls *when* the output changes
 - D (the data input): controls *what* the output changes to
- Function
 - When $CLK = 1$, D passes through to Q (the latch is *transparent*)
 - When $CLK = 0$, Q holds its previous value (the latch is *opaque*)
- Avoids invalid case when $Q \neq \text{NOT } \bar{Q}$

D Latch Symbol

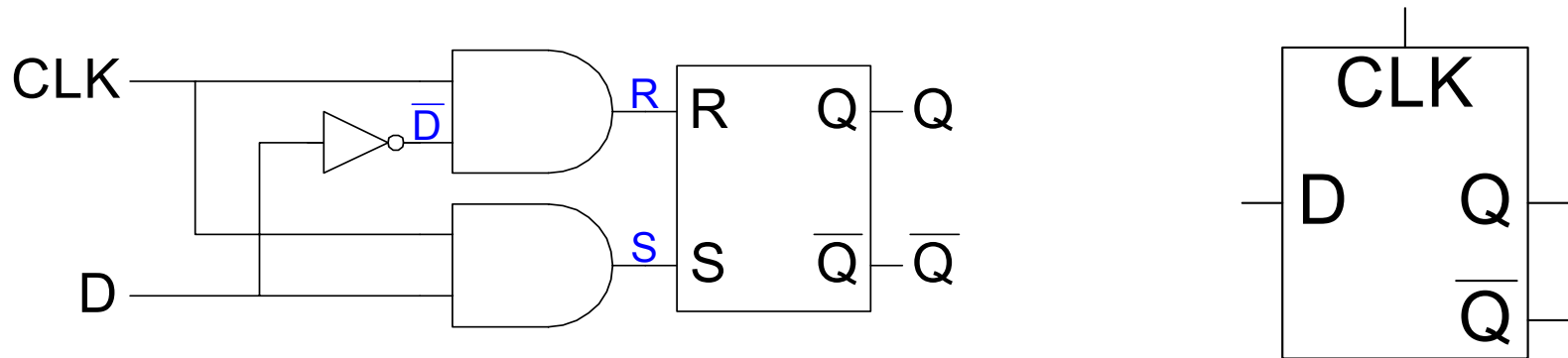


D Latch Internal Circuit



| <i>CLK</i> | <i>D</i> | \bar{D} | <i>S</i> | <i>R</i> | <i>Q</i> | \bar{Q} |
|------------|----------|-----------|----------|----------|----------|-----------|
| 0 | X | | | | | |
| 1 | 0 | | | | | |
| 1 | 1 | | | | | |

D Latch Internal Circuit



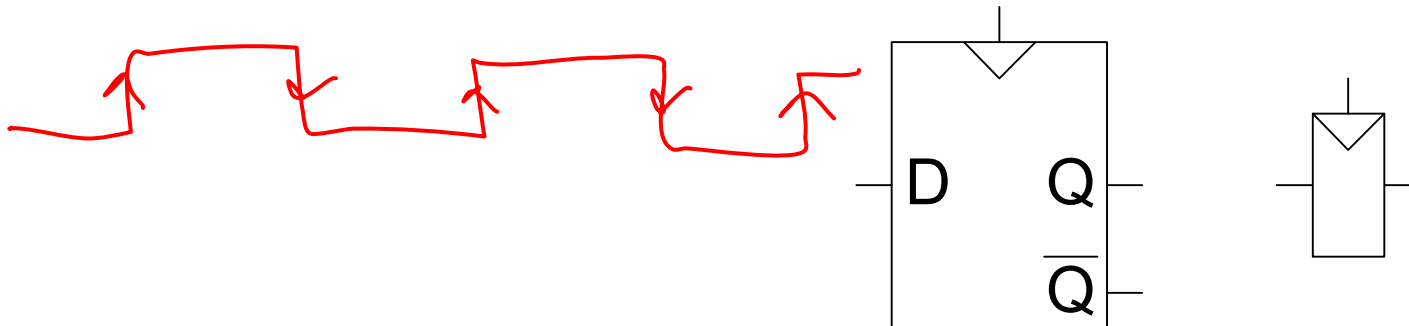
| CLK | D | \bar{D} | S | R | Q | \bar{Q} |
|-------|-----|-----------|-----|-----|------------|------------------|
| 0 | X | X | 0 | 0 | Q_{prev} | \bar{Q}_{prev} |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |

characteristic eqn.: $Q^+ = D$

D Flip-Flop

- Two inputs: CLK , D
- *Function*
 - The flip-flop “samples” D on the rising edge of CLK
 - When CLK rises from 0 to 1, D passes through to Q
 - Otherwise, Q holds its previous value
 - Q changes only on the rising edge of CLK
- A flip-flop is called an *edge-triggered* device because it is activated on the clock edge

D Flip-Flop Symbols



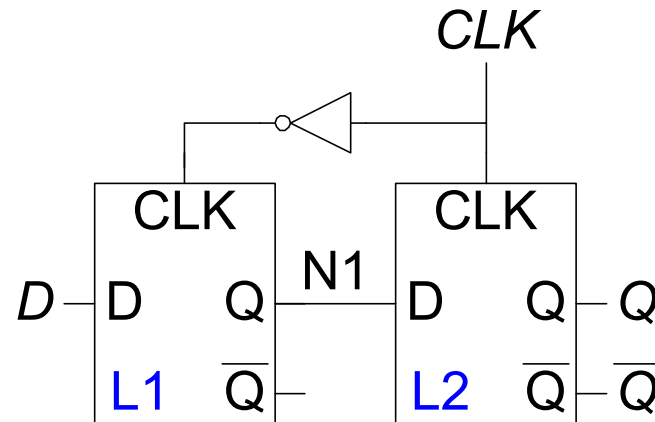
D Flip-Flop Internal Circuit

- Two back-to-back latches (L1 and L2) controlled by complementary clocks



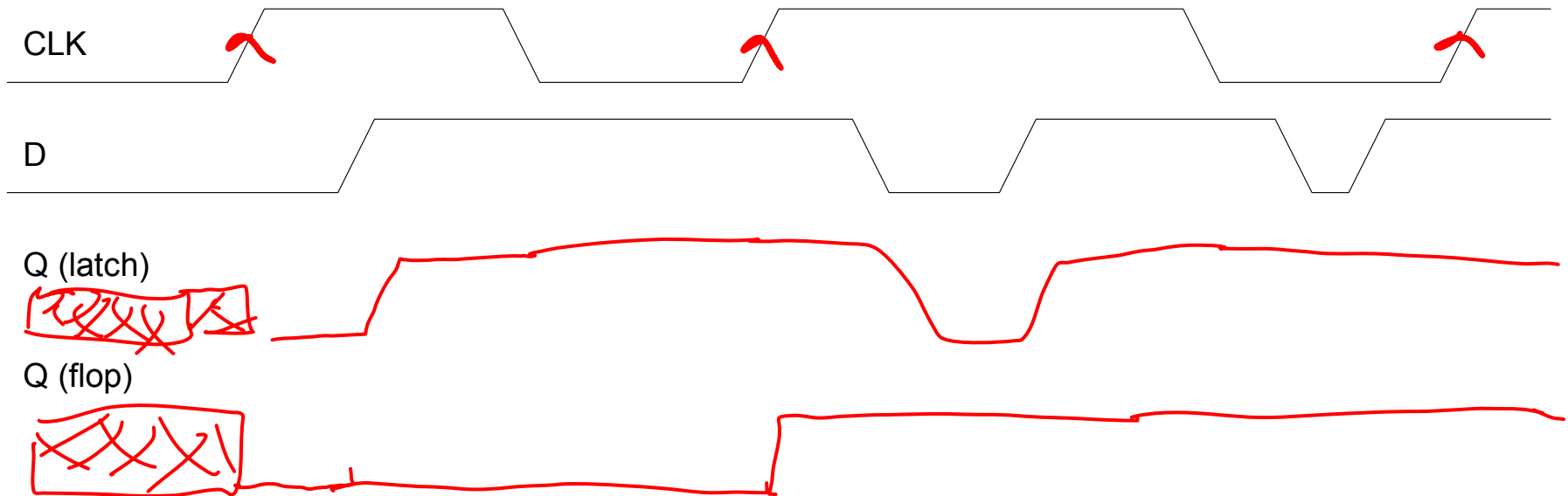
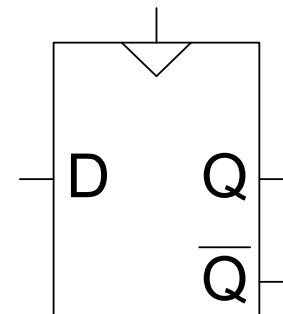
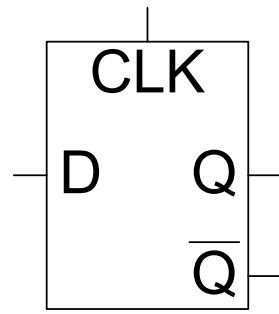
- When $CLK = 0$
 - L1 is transparent
 - L2 is opaque
 - D passes through to N1

- When $CLK = 1$
 - L2 is transparent
 - L1 is opaque
 - N1 passes through to Q

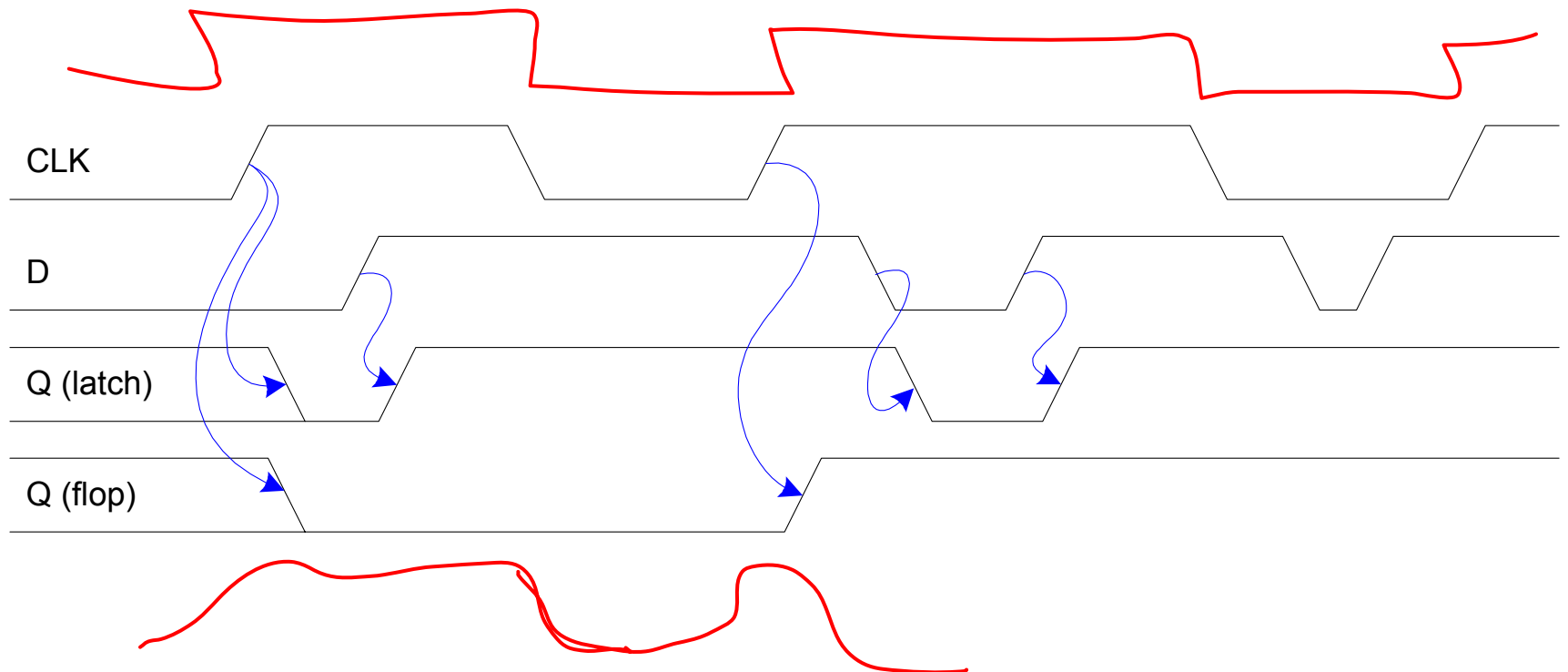


- Thus, on the edge of the clock (when CLK rises from 0→1)
 - D passes through to Q
-

D Flip-Flop vs. D Latch

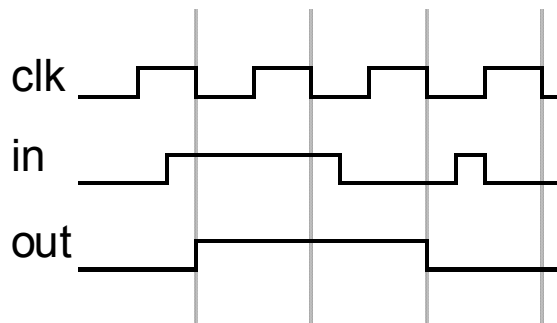
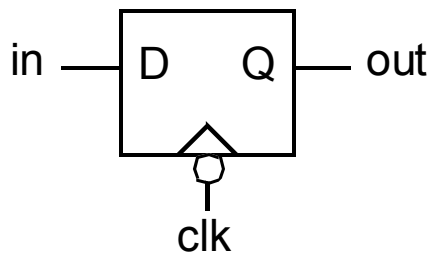


D Flip-Flop vs. D Latch



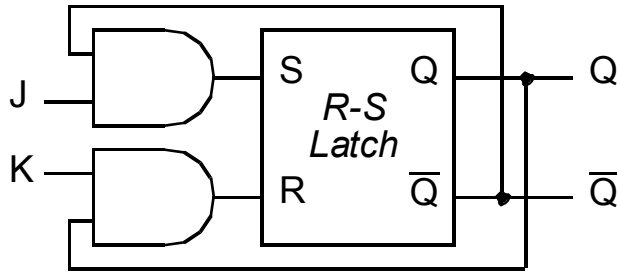
More Flip-Flop Terminology

- flip-flop = output only changes when clock changes
 - positive-edge triggered FF = changes on LH clock transition
 - negative-edge triggered FF = changes on HL clock transition
 - master-slave FF = changes on HL clock transition



Two other kinds of latches

- J-K latch



| J | K | Q | Q ⁺ | Action |
|---|---|---|----------------|--------|
| 0 | 0 | 0 | 0 | hold |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | reset |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | set |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 1 | toggle |
| 1 | 1 | 1 | 0 | |

- characteristic eqn.: $Q^+ = Q\bar{K} + \bar{Q}J$
 - problem: toggle forever due to immediate feedback

Toggle Flip-Flop (TFF)

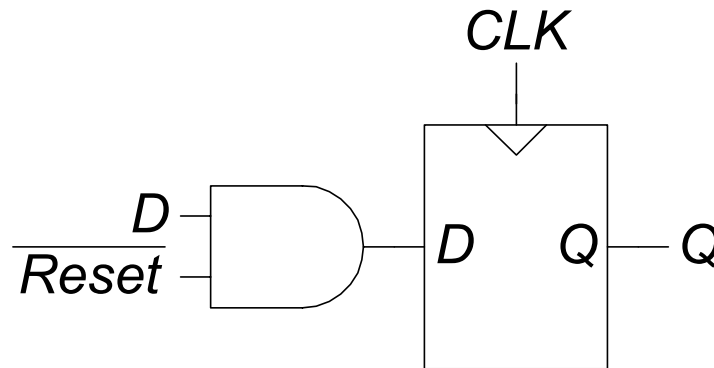
- Toggle FF (TFF)
 - single input FF
 - when input == 1, output complemented
 - characteristic equation: $Q^+ = T\bar{Q} + \bar{T}Q$

 - Can convert between FF types (D from JKs, JK from D)
-

Resettable Flip-Flops

- Two types:
 - Synchronous: resets at the clock edge only
 - Asynchronous: resets immediately when $Reset = 1$
- Asynchronously resettable flip-flop requires changing the internal circuitry of the flip-flop (see Exercise 3.10)
- Synchronously resettable flip-flop?

Internal
Circuit



Choosing a Flip-Flop type

- R-S latch
 - limited stand-alone utility
 - J-K FF
 - along with DFF, most commonly used FF
 - often requires fewest gates to determine next state
 - 2 inputs though, often creates more wiring than DFF design
 - DFF
 - also extremely common
 - especially in VLSI circuits where wiring expensive
 - TFF
 - rarely available in TTL package since easily created from J-K FF by tying J and K together
 - great building blocks for counters
-

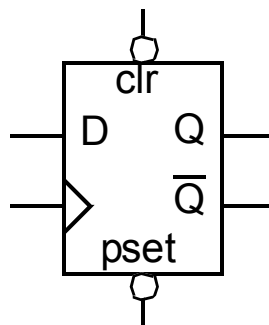
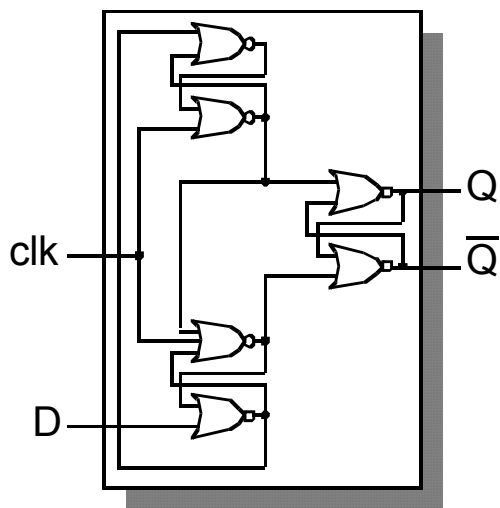
Flip Flop

- Excitation tables - input values that cause particular state change

| Q | Q ⁺ | S | R | J | K | D | T |
|---|----------------|---|---|---|---|---|---|
| 0 | 0 | 0 | X | 0 | X | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | X | 1 | 1 |
| 1 | 0 | 0 | 1 | X | 1 | 0 | 1 |
| 1 | 1 | X | 0 | X | 0 | 1 | 0 |

Larger sequential logic elements

- Until now, used gates with feedback to build memory elements
 - e.g. R-S latch, J-K FF, D-FF \Rightarrow storage for 1-bit



| $\overline{\text{pset}}$ | $\overline{\text{clr}}$ | D | clk | Q ⁺ |
|--------------------------|-------------------------|---|-----|----------------|
| L | X | X | X | H |
| X | L | X | X | L |
| H | H | L | ↑ | L |
| H | H | H | ↑ | H |

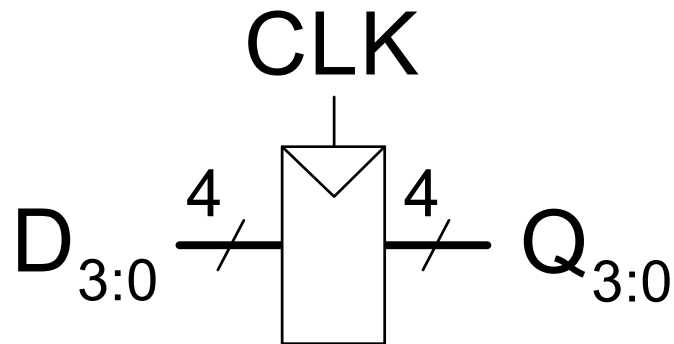
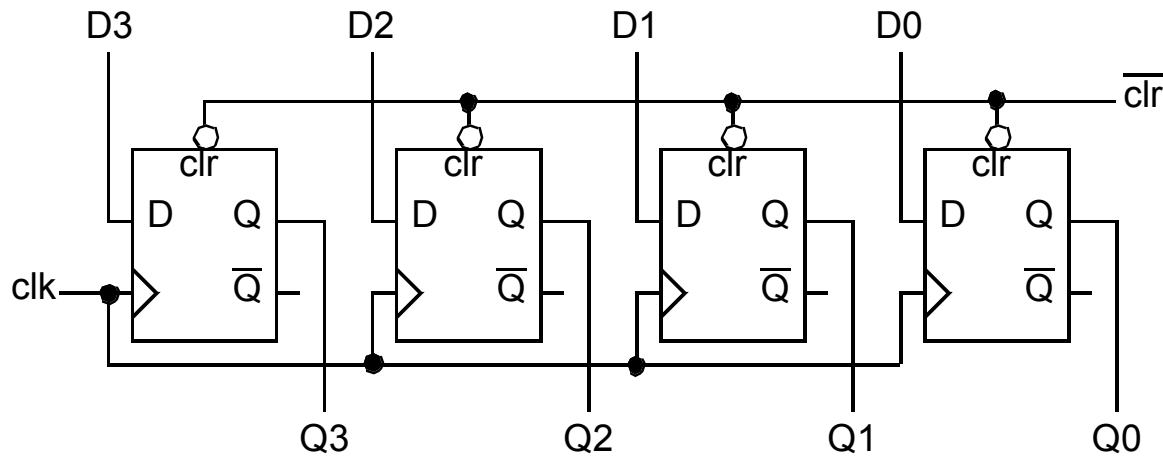
- from now on, mostly deal with D-FF abstraction
 - (similar to what we did with transistor-to-gate abstraction)

Basic Storage Elements

- 4 important kinds of basic storage elements
 - registers (and register files)
 - Random Access Memories (RAMs)
 - shift registers
 - counters
 - counters lead into general Finite State Machine (FSM) design
-

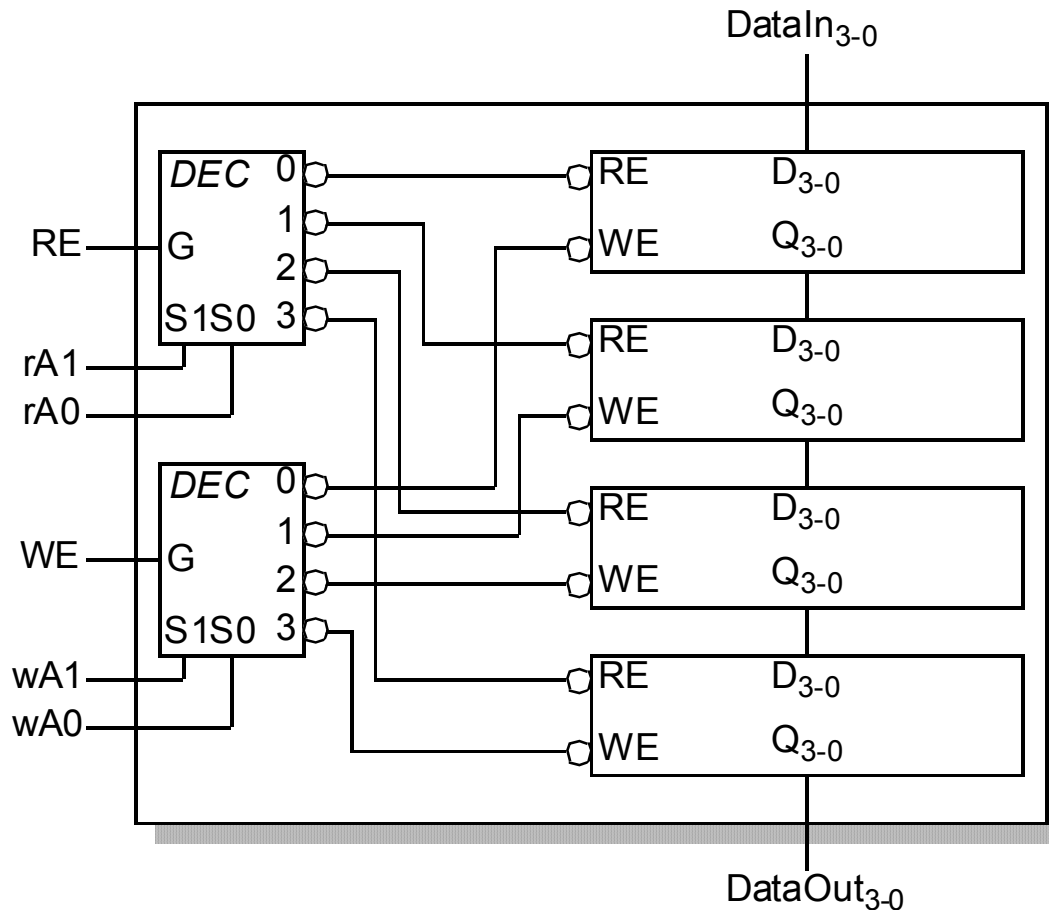
Registers

- register = group of storage elements read/written as a unit



Registers

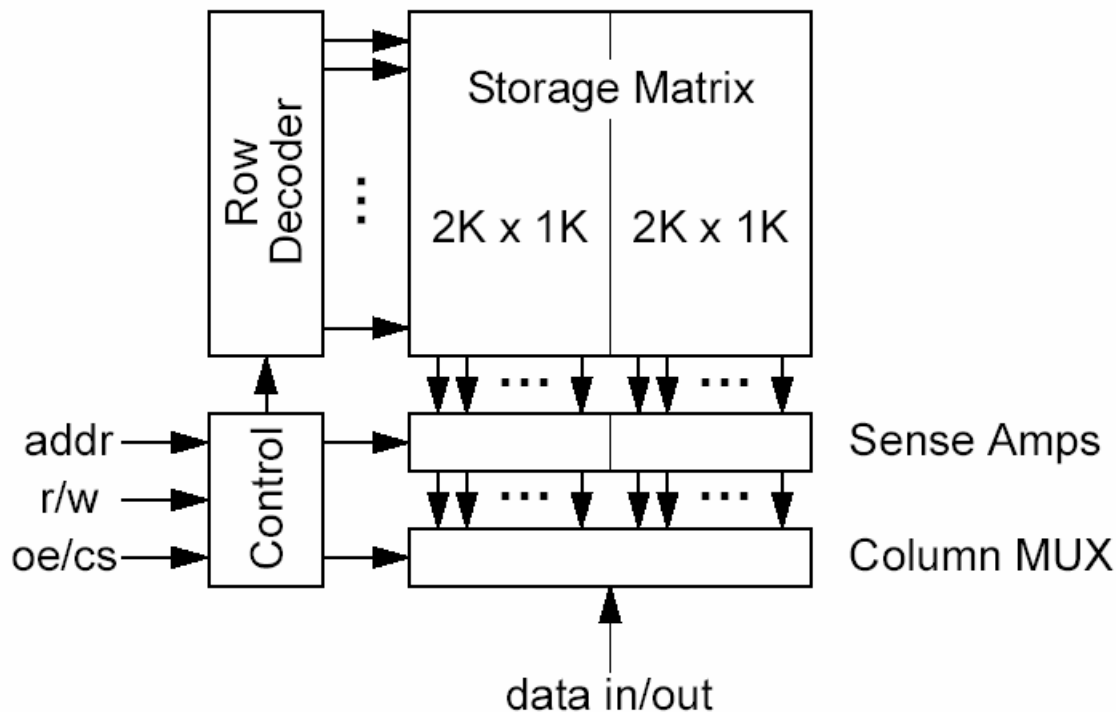
- register file = single unit comprised of multiple registers



- No explicit clock, WE acts as clock signal and RE acts as OE!

Random Access Memories (RAMs)

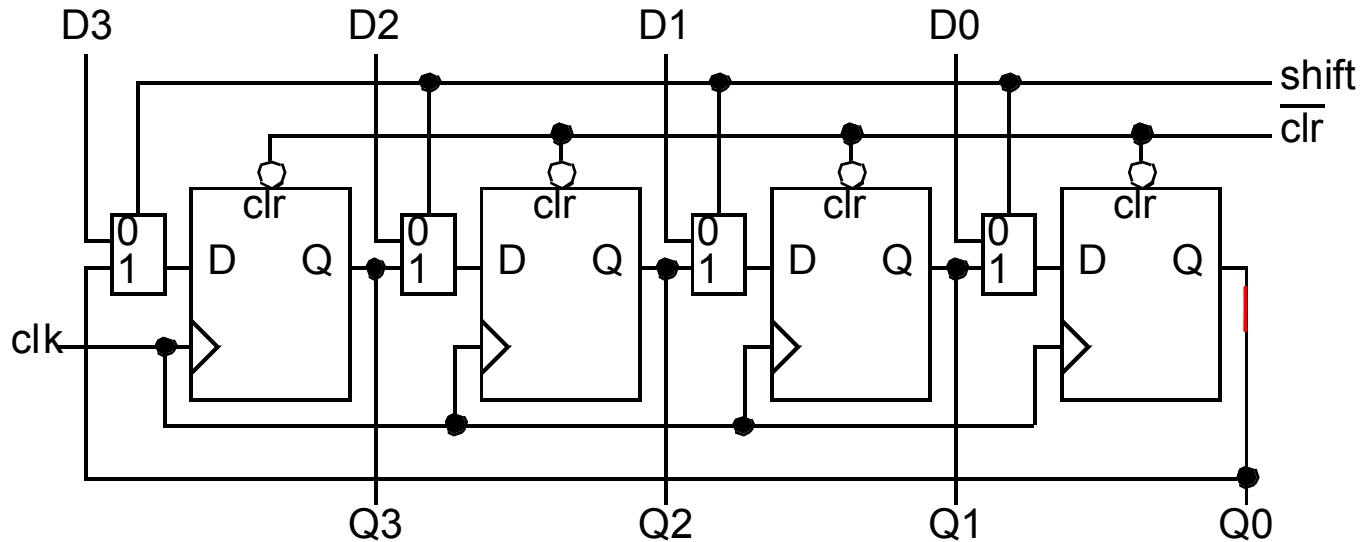
- generalizes the regfile concept; array of storage elements
- state of the art: 64M to 512M bits in a single DRAM chip



Memories are VERY important: we'll return to array structures later in the course

Shift registers

- register that can circulate as well as store bits



- Standard part: 74194 “4-bit Bidirectional Universal Shift Register”
hold, shift right, shift left, parallel load, asynchronous clear
-

Counters

- sequential logic circuits that proceed through:
 - a well-defined sequence of states
 - types
 - up-counter vs. down-counter
 - binary vs. decade vs. Gray code vs. ring vs. Johnson
 - simplest possible finite-state machines (FSMs)
 - single input (typically clock saying “count!”)
 - outputs simply the current state
-