Lecture 9 -- Dynamic Logic

Outline

- Not going to talk about logical effort today -- realized that the planned schedule wouldn’t cover the stuff you need for MP2 in time

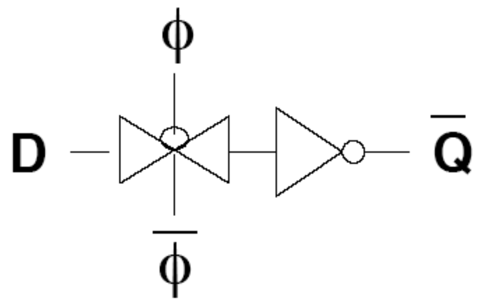
- Instead, talk about dynamic logic
  - Concepts
  - Basic Implementations
  - Staging dynamic gates
  - Larger-scale systems

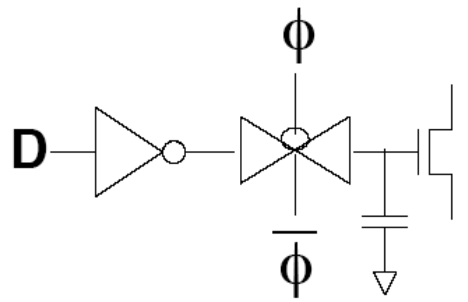
Dynamic Vs. Static Logic

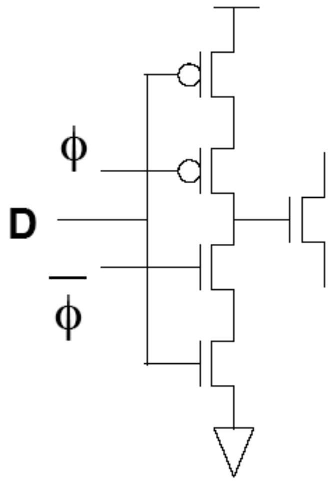
- Static Logic always has a path from power or ground to the output
  - Stable over long periods
  - Simple clocking schemes

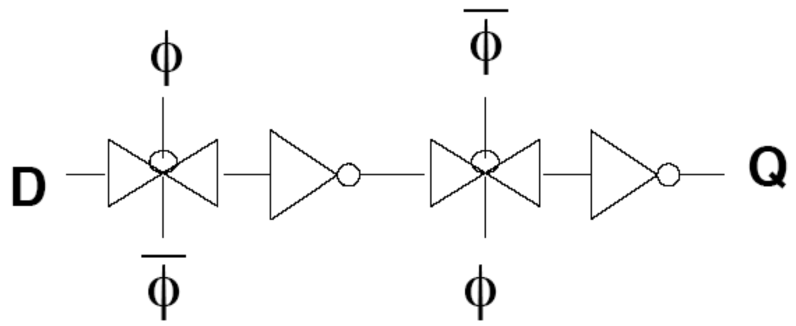
- Dynamic Logic relies on capacitance of gates or other structures to hold state
  - Can be faster than static logic
  - Has <u>minimum</u> operating frequency
  - More restrictions about how/when inputs can change
    - Consequence of how we implement dynamic logic

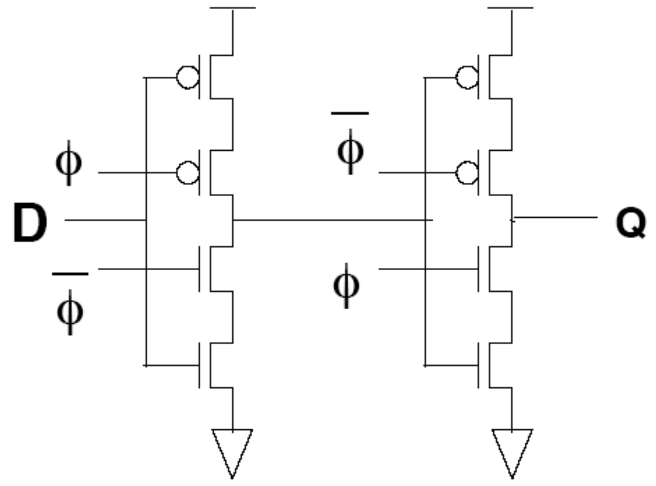
Dynamic Latch





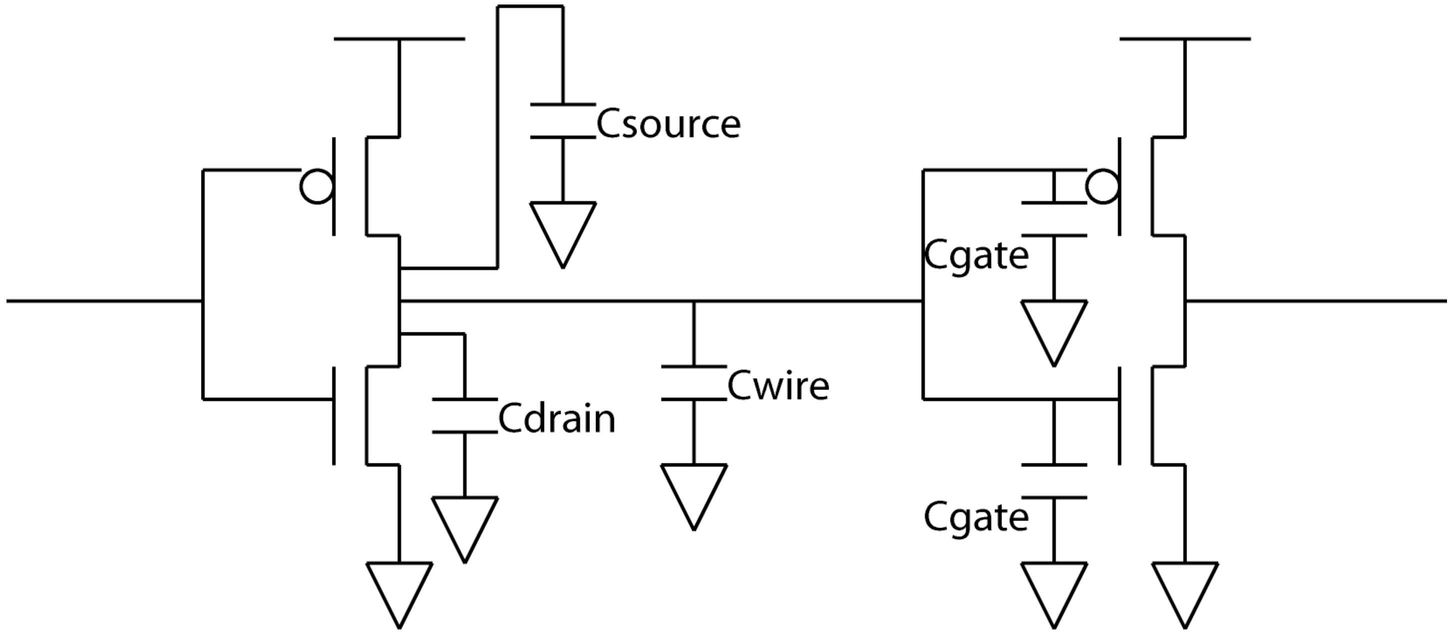






Bad Points of Static CMOS

- Capacitance on input
  - Two transistors, and P-FET generally 2x N-FET width
- Conflicting drive while input switching
  - During input swing, will have paths from both power and ground to output, increasing delay



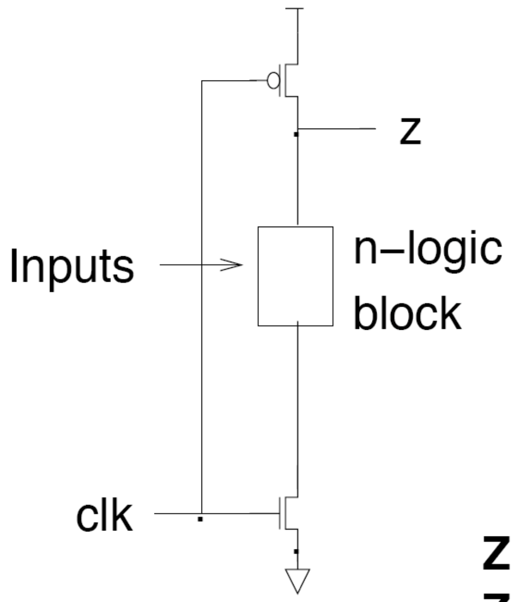
Dynamic Logic Concepts

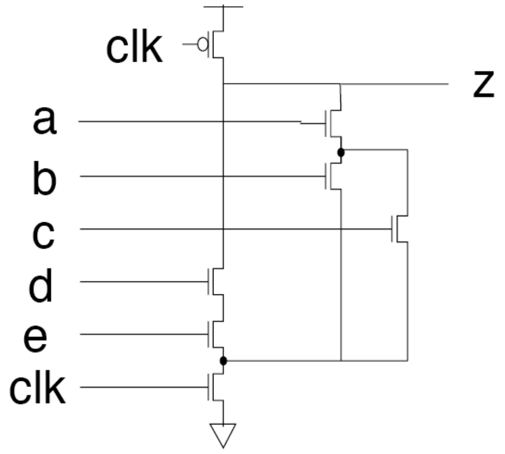
- All dynamic logic we'll look at is clocked
- In one phase of the clock, we'll *precharge* the gate to either 0 or 1.
- In another phase, we'll *evaluate* whether the gate's output should stay at the precharged value or change.

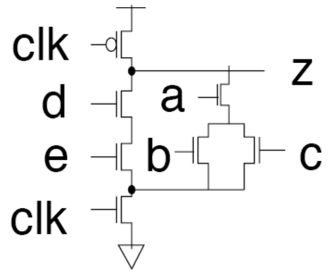
- Doing this will allow us to eliminate either the p- or n-network from each gate
  - Reduces load capacitance
  - Timing of precharge/evaluate phases eliminates delay due to connecting output to both power and ground during input swing time

Example Dynamic Logic Gates

- These are what your book calls *footed* logic gates
- Some designs don't include the clocked N-FET
  - Reduces load on clock signal
  - Potential for static power dissipation when clock low

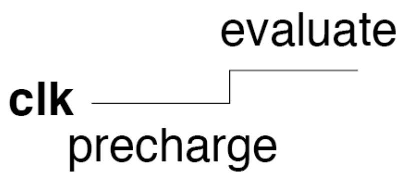






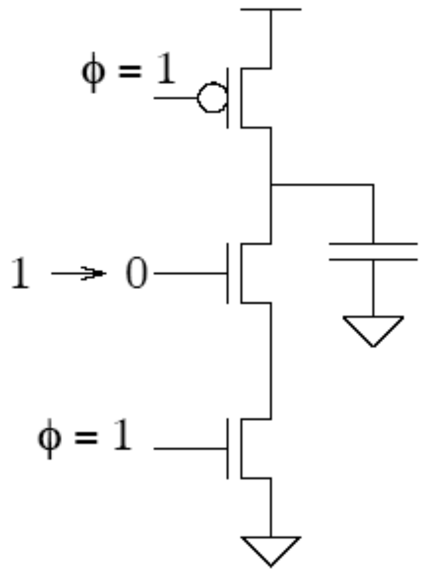
$Z = \overline{A.(B+C) + (D.E)}$ **clk = 1**

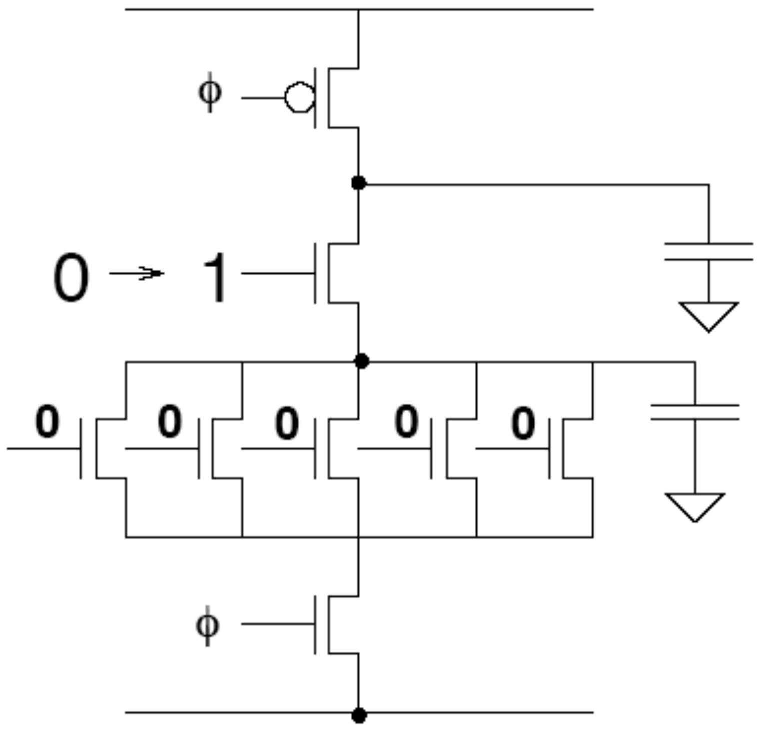
Z = HIGH **clk = 0**



Using Dynamic Logic

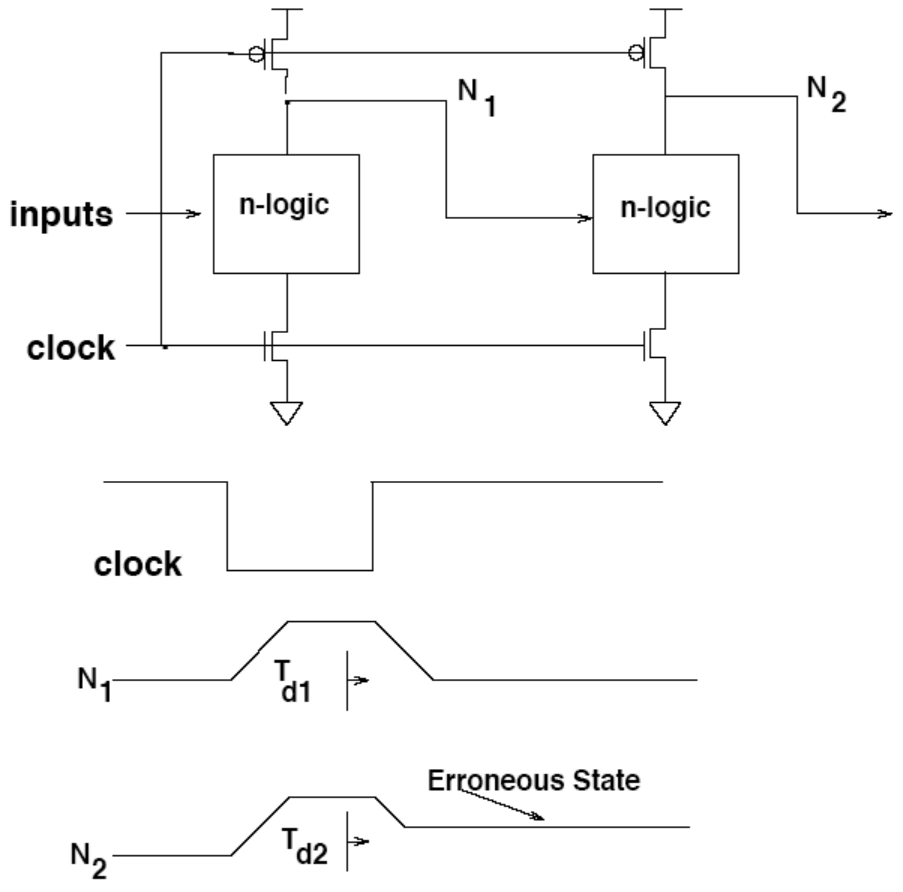
- Inputs may change freely during precharge phase
- Input transitions during evaluate phase may cause problems





Cascading Dynamic Gates

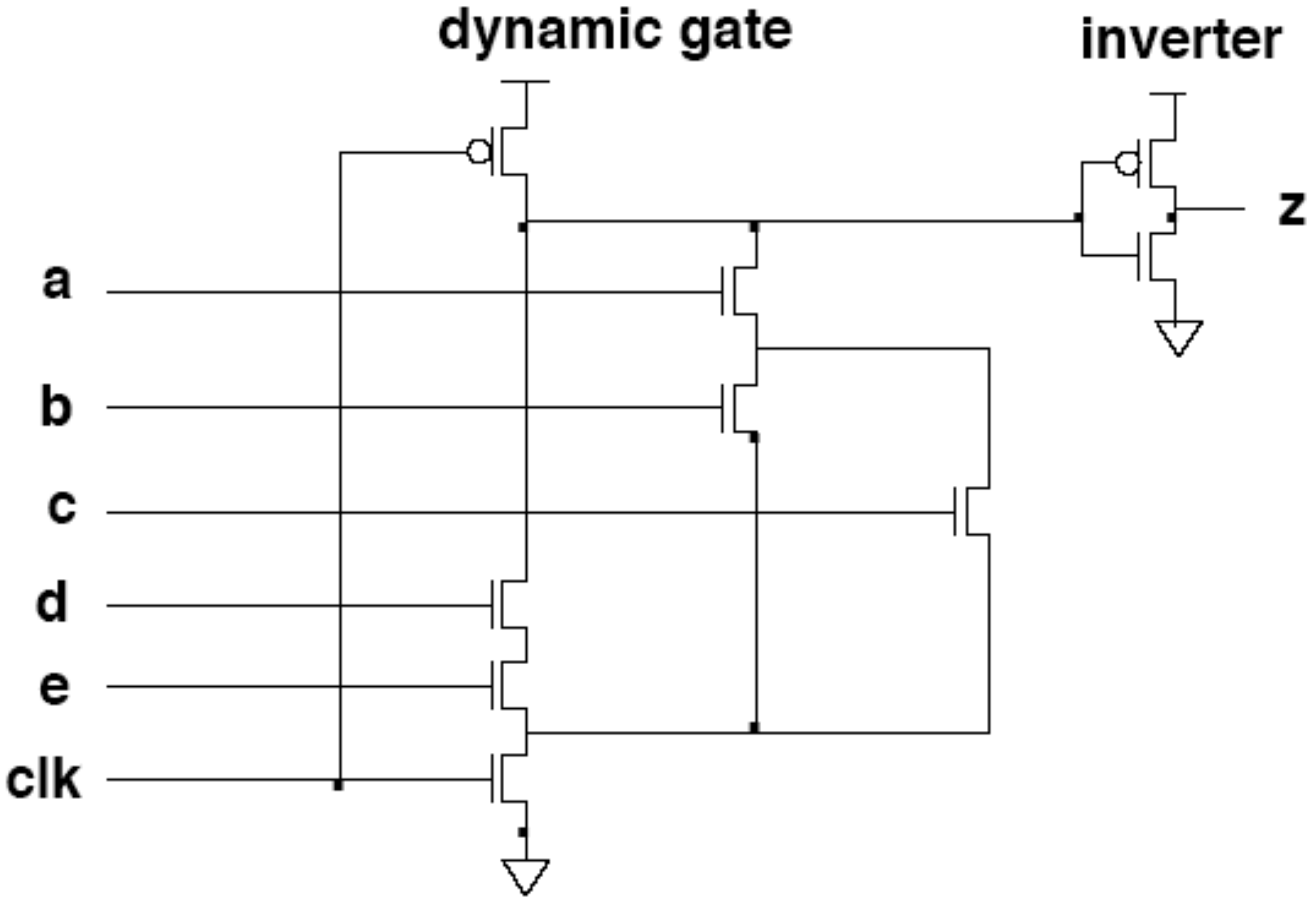
- Just plain doesn't work with the designs we've shown



Domino CMOS

- Add inverter to output of each gate
  - Can now connect gates in sequence
  - As one gate evaluates, it triggers the gates it drives, similar to a set of falling dominoes
- Domino gates can make a single 0-->1 output transition during the evaluate phase
  - No glitches on output
- Still have potential charge-sharing problems
- Can only make non-inverting structures
  - Means that there are logic functions we can't implement with this form of Domino CMOS

Example Domino Gate

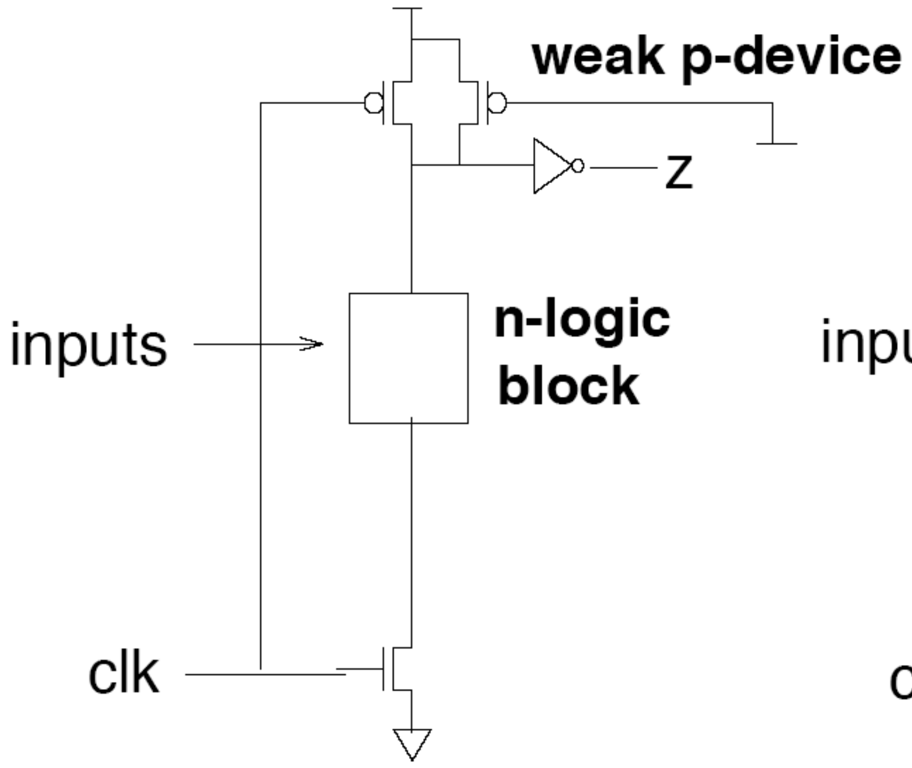


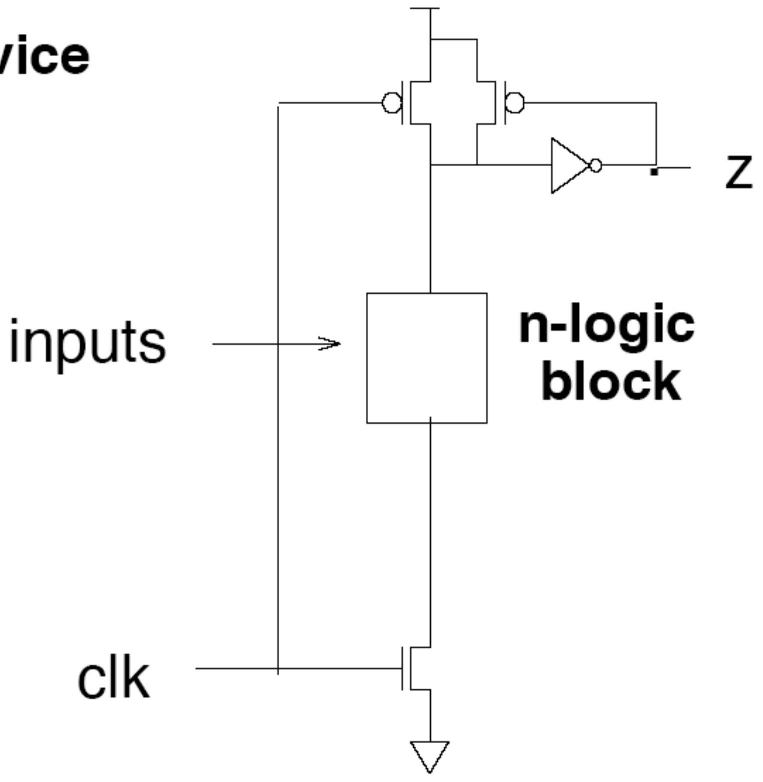
Staticizing Gates

- Often, it’s hard to guarantee that a circuit will be clocked fast enough to avoid charge leakage problems on dynamic logic
  - Example: may want to support low-speed testing
- Domino gates are static when outputting a “1”, but rely on charge storage to maintain outputs of “0”
- Can add staticizing logic to domino gate to allow it to drive “0” outputs for long periods of time
  - Some cost in performance, but still faster than static CMOS

Staticized Gates

- Make pull-up devices very weak to minimize performance impact
  - Just need to flow enough current to counter leakage



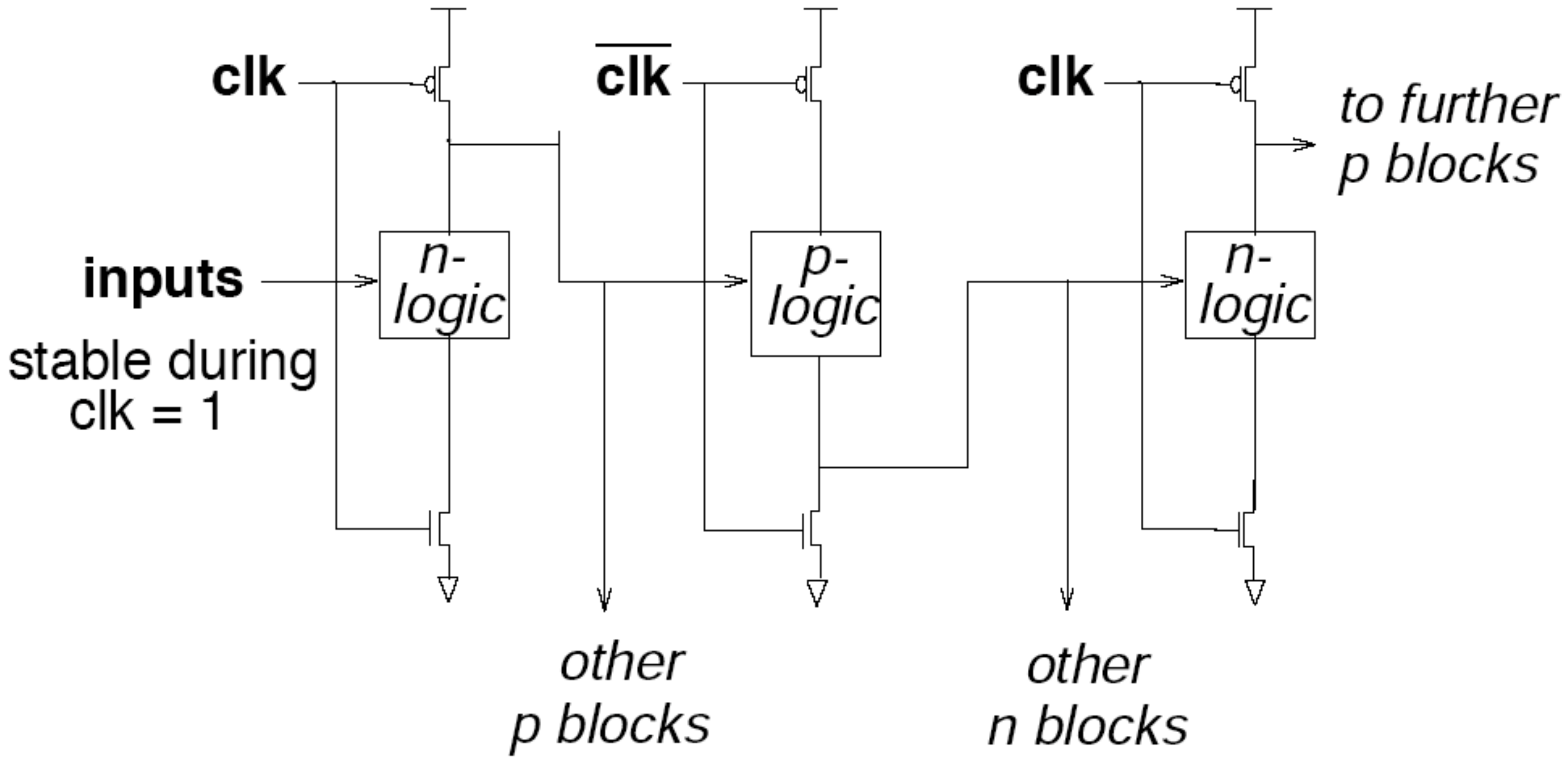


N-P Domino Logic

- Need inverters at output of domino gates because output transition of n-network has the wrong polarity for n-network inputs
  - Decreases performance somewhat
  - Can’t build inverting structures

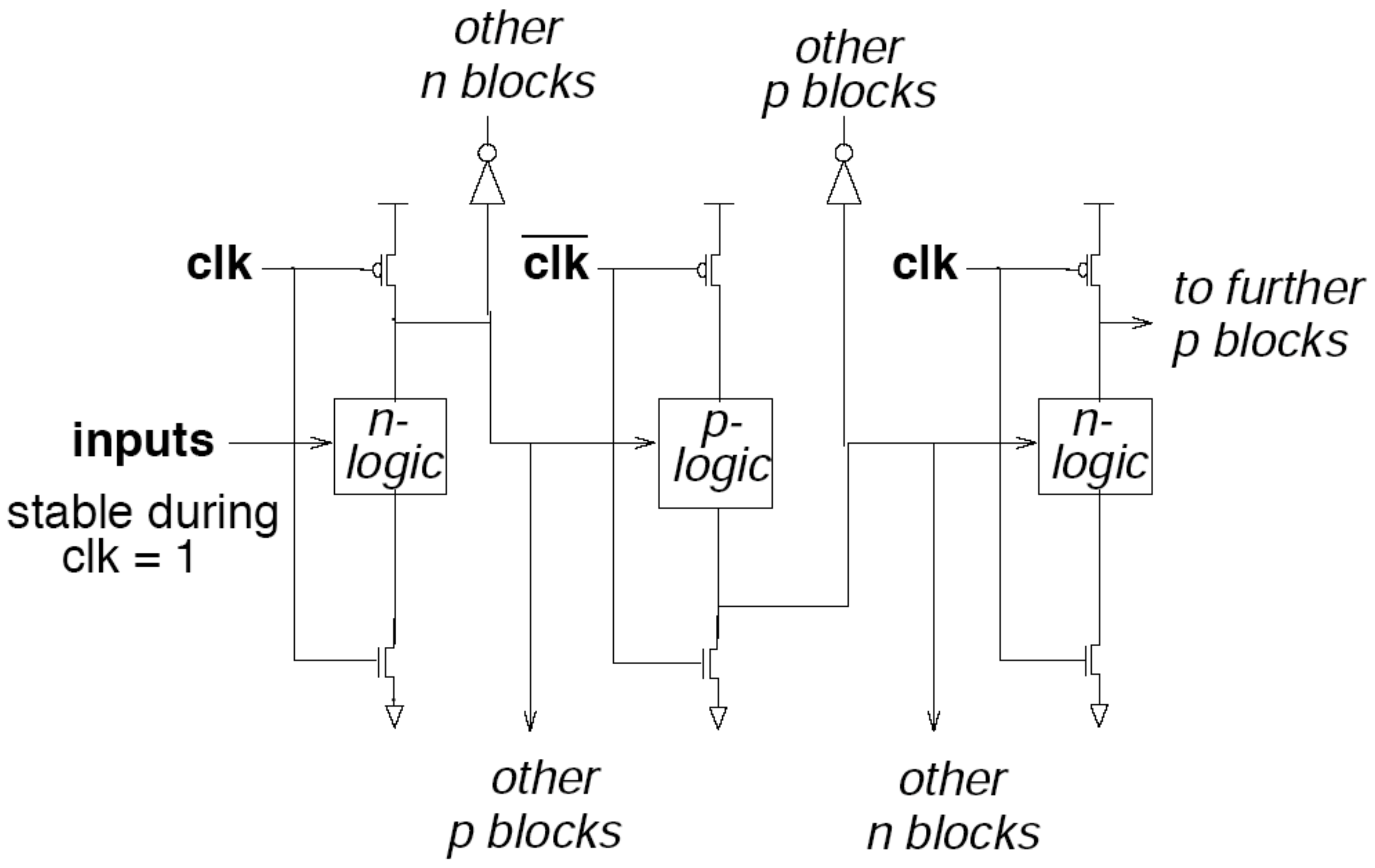
- Alternative: Alternate dynamic gates with n- and p-networks
  - Provides inversion
  - P-networks can take n-network output directly
  - P-FETs are slower than N-FETs for same size
    - Widening P-FETs adds load capacitance

N-P Domino Logic



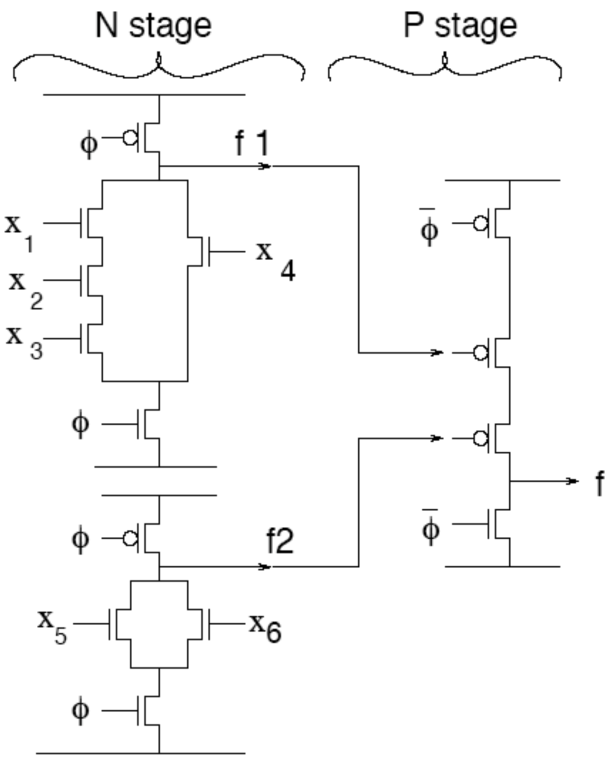
N-P Domino Logic

- Can add inverters to allow gates to drive both n- and p-gates



Example

- Implement $f = (x_1x_2x_3 + x_4)(x_5 + x_6)$ in N-P CMOS
- Define subfunctions $f_1 = \overline{x_1x_2x_3 + x_4}$, $f_2 = \overline{x_5 + x_6}$
- Then $f = \bar{f_1}\bar{f_2}$ can be implemented as follows:

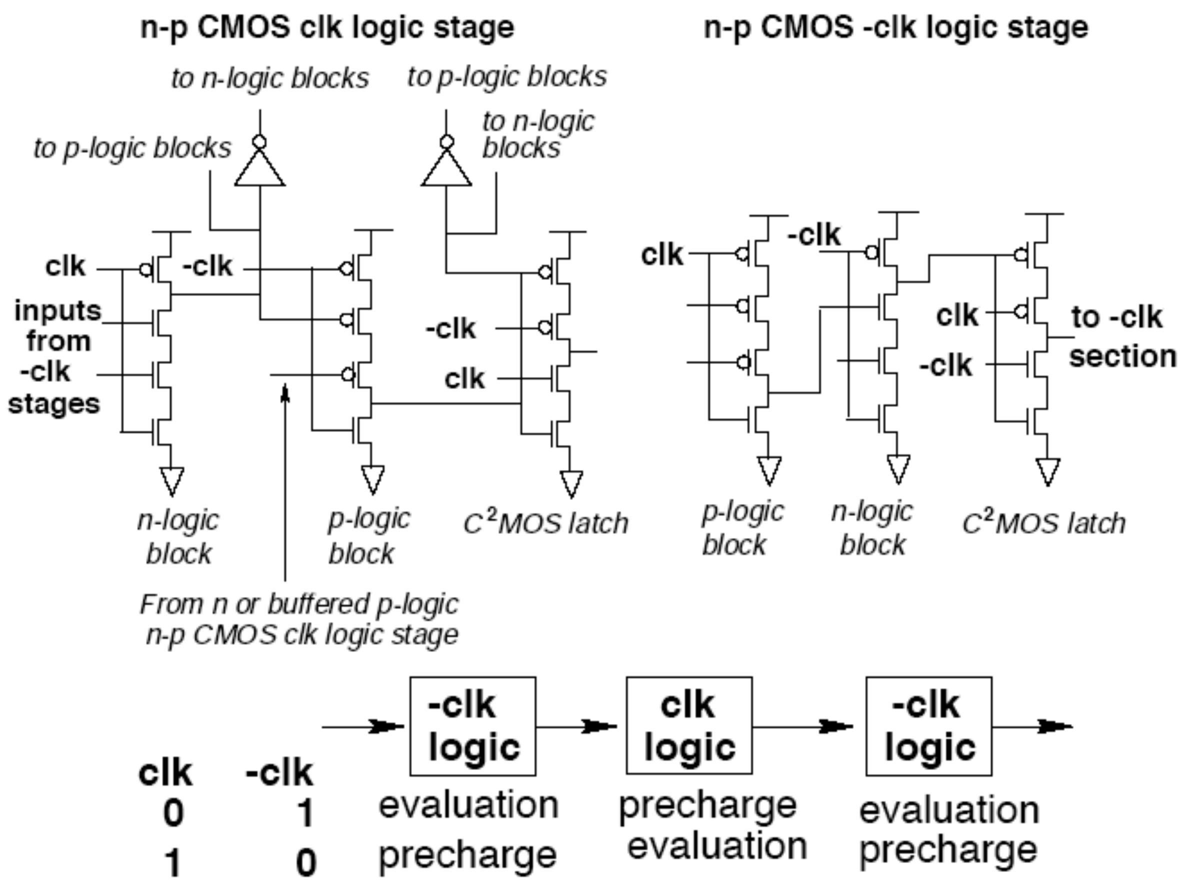


Clocking Dynamic Logic

- Dynamic logic needs to be clocked to generate precharge, evaluate phases
  - Length of precharge period determined by precharge time of single gate
  - Length of evaluate period determined by sum of gate evaluate times
- Generally don't implement entire chips in dynamic logic
  - Greatly increased design time/gate
  - Most gates not performance-critical
- Typically want latches at dynamic/static boundaries
  - Pipelining
  - Control glitches

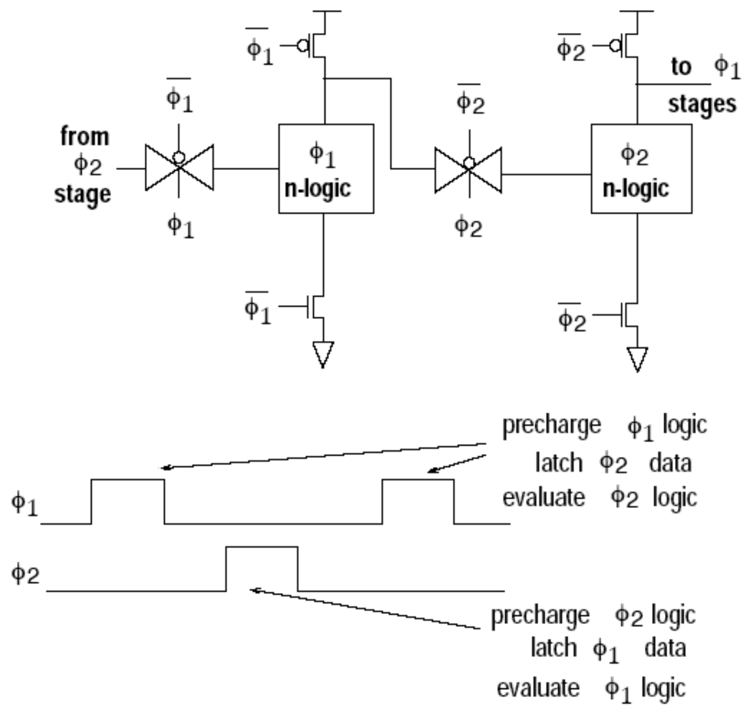
Single-Phase Clocking

- Add clocked latch at the output of each set of dynamic gates



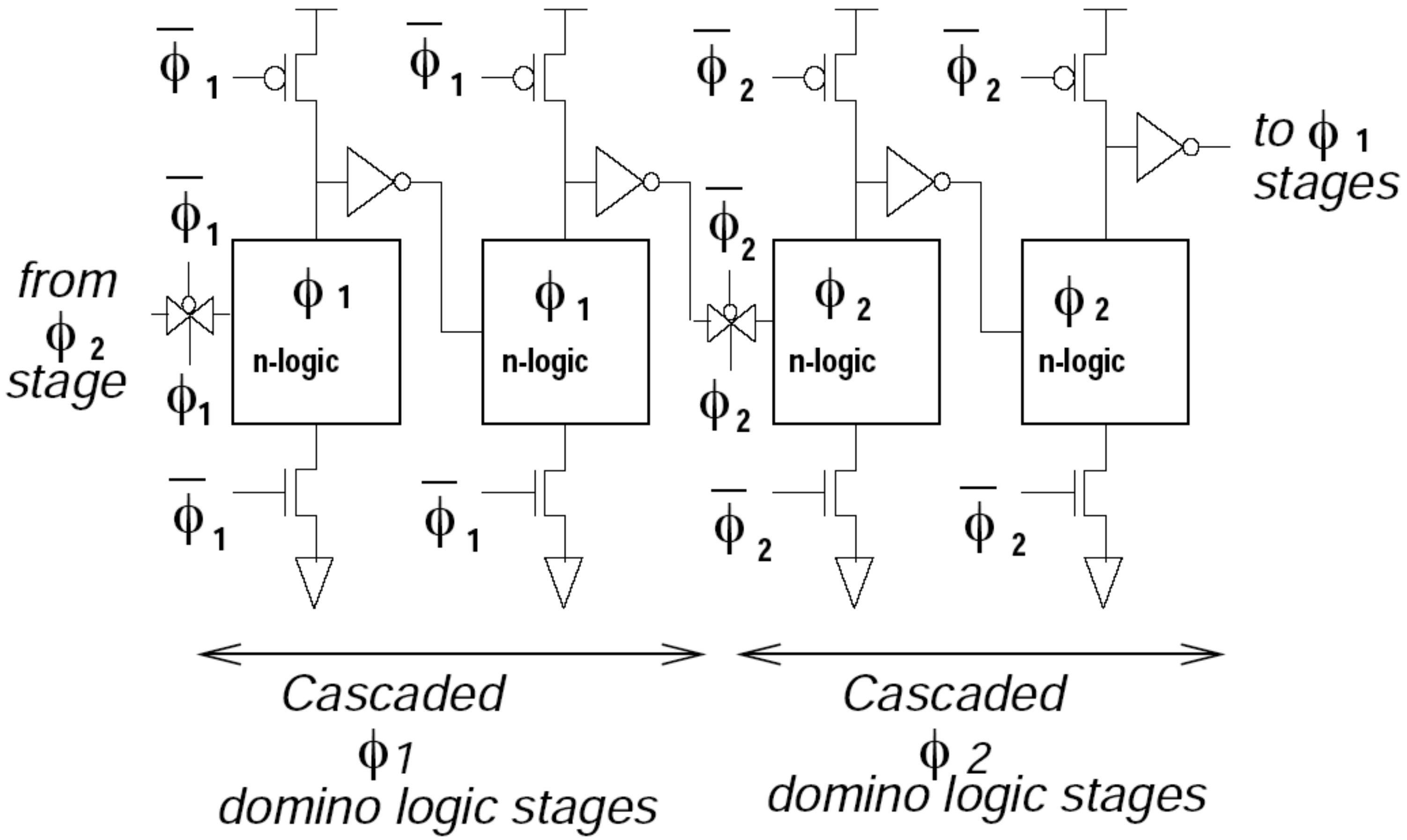
Two-Phase Clocking

- First stage is precharged during phi_1 and evaluated during phi_2
- While first is evaluated, second is precharged, and first stage outputs stored on second-stage inputs.



Two-Phase Domino Logic

- Cascade number of domino logic stages before latching the result



Reading

- Sections 6.2.4 and 6.3 in your book
  - 6.3 in particular covers things that can go wrong with your circuits