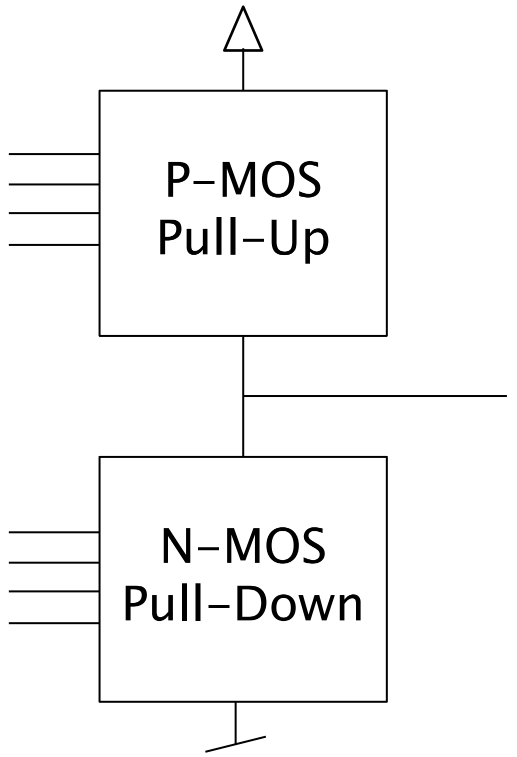
ADVANCED VLSI DESIGN

Introduction to Dynamic Logic

18/02/2025

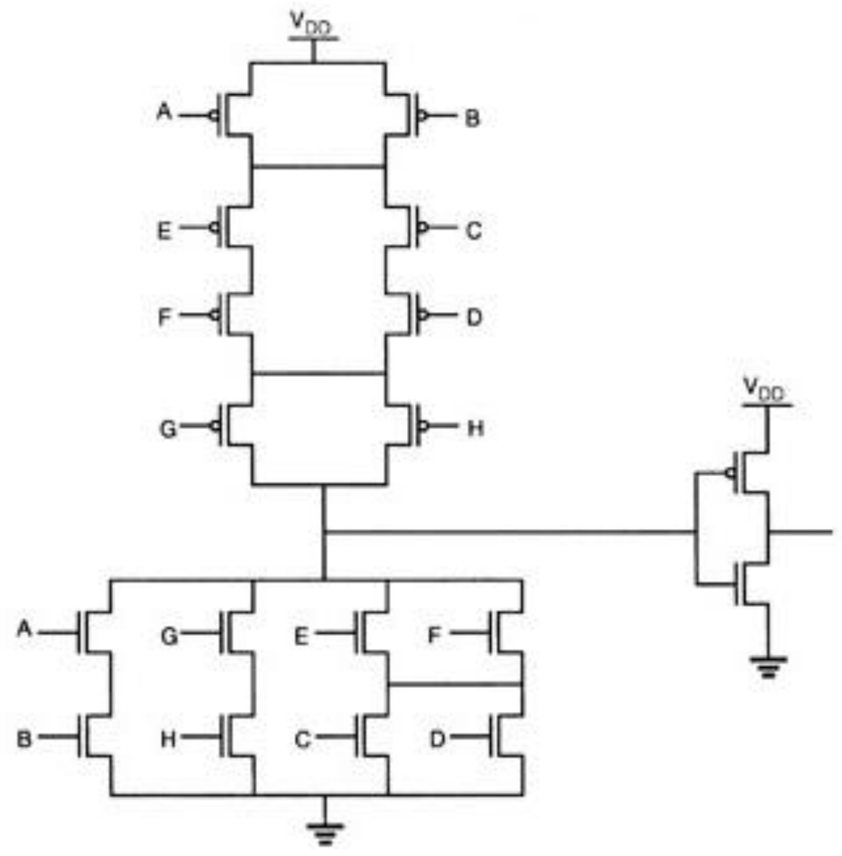
Static CMOS Logic



- Complementary Pull-Up and Pull-Down network of P-MOS and N-MOS transistors
- Full rail-to-rail operation
- Good noise margin
- No static currents

- N input gate requires 2*N transistors

Static CMOS Logic

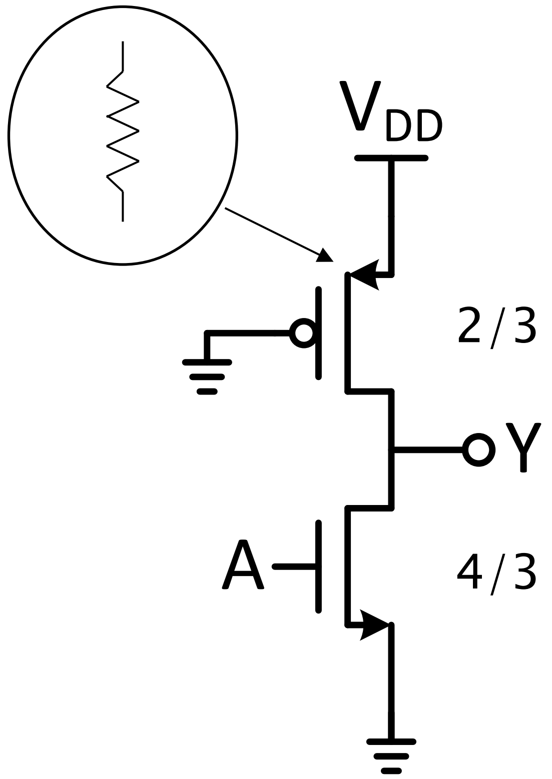


- Complementary Pull-Up and Pull-Down network of P-MOS and N-MOS transistors
- Full rail-to-rail operation
- Good noise margin
- No static currents

- One branch (NMOS or PMOS) always suffers from many series transistors
- P-MOS network is slow
- P-MOS upsining adds to load

How can we remove the “redundant” P-MOS network?

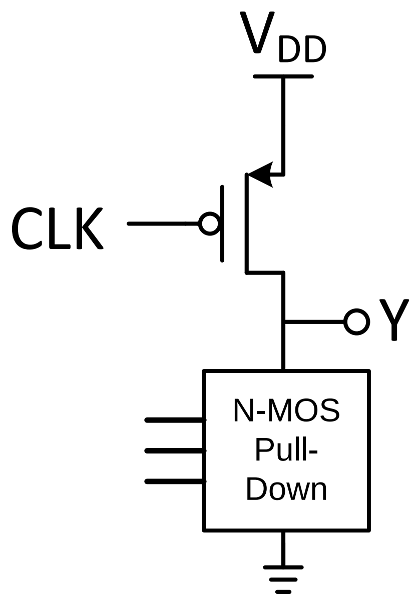
Pseudo N-MOS Logic

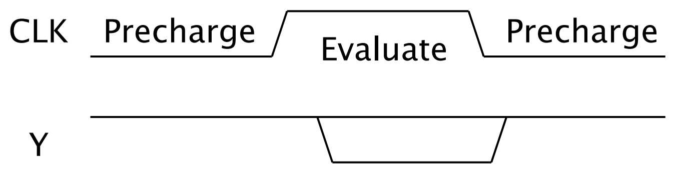


- **Idea: avoid the *redundant* P-MOS network**
- **Replace P-MOS network by a load with a relatively high impedance**
  - Ratioed circuit
- **Reduces transistor count (and load) to only N+1 (N)**
- **But has several issues:**
  - Slow Pull-Up
  - Static Power Consumption
  - Non-zero V_{OL} (resistive divider)

How can we avoid contention between Pull-Up and Pull-Down

Dynamic (N-MOS) Logic

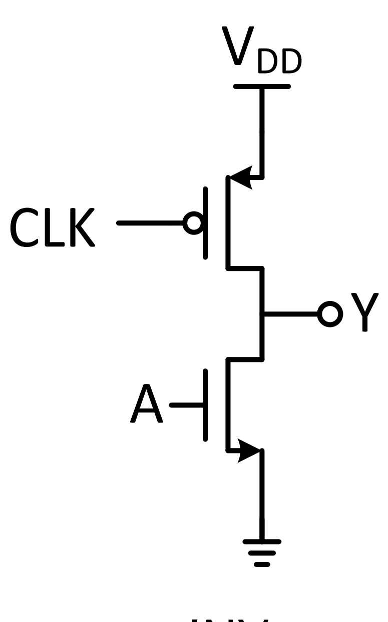




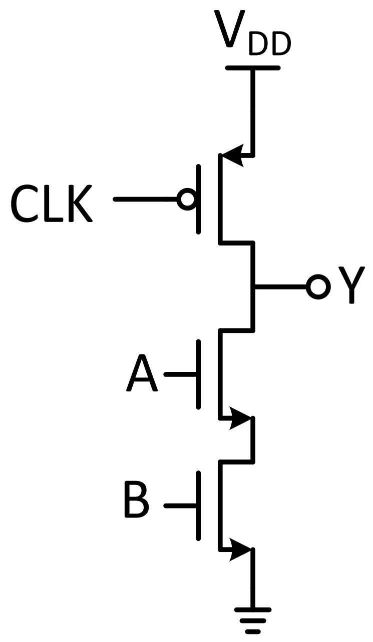
- **Replace the static load with a clocked “Precharge” transistor**
- **Operation in two phases**
  - Precharge (CLK=“0”): initialize output node to “1”
  - Evaluate (CLK=“1”): N-MOS network conditionally pulls down the output as needed
- **Several advantages:**
  - Only N+1 transistors (load=N)
  - Speed determined only by N-MOS

Dynamic (N-MOS) Logic Examples

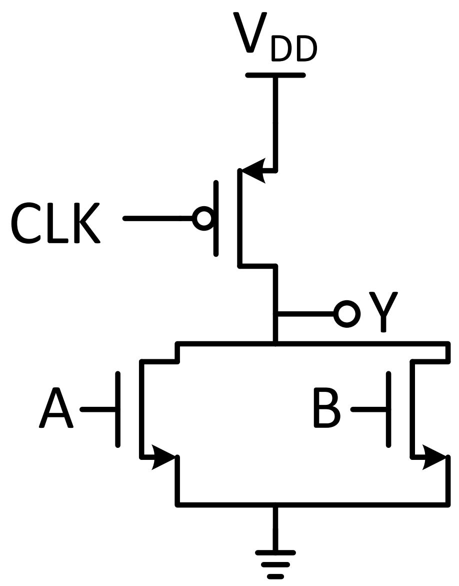
- **Only N-MOS network determines the number of stacked transistors**
- **Good for realizing high fan-in gates which in CMOS would have a large P-MOS stack (wide NORs)**



INV



NAND2



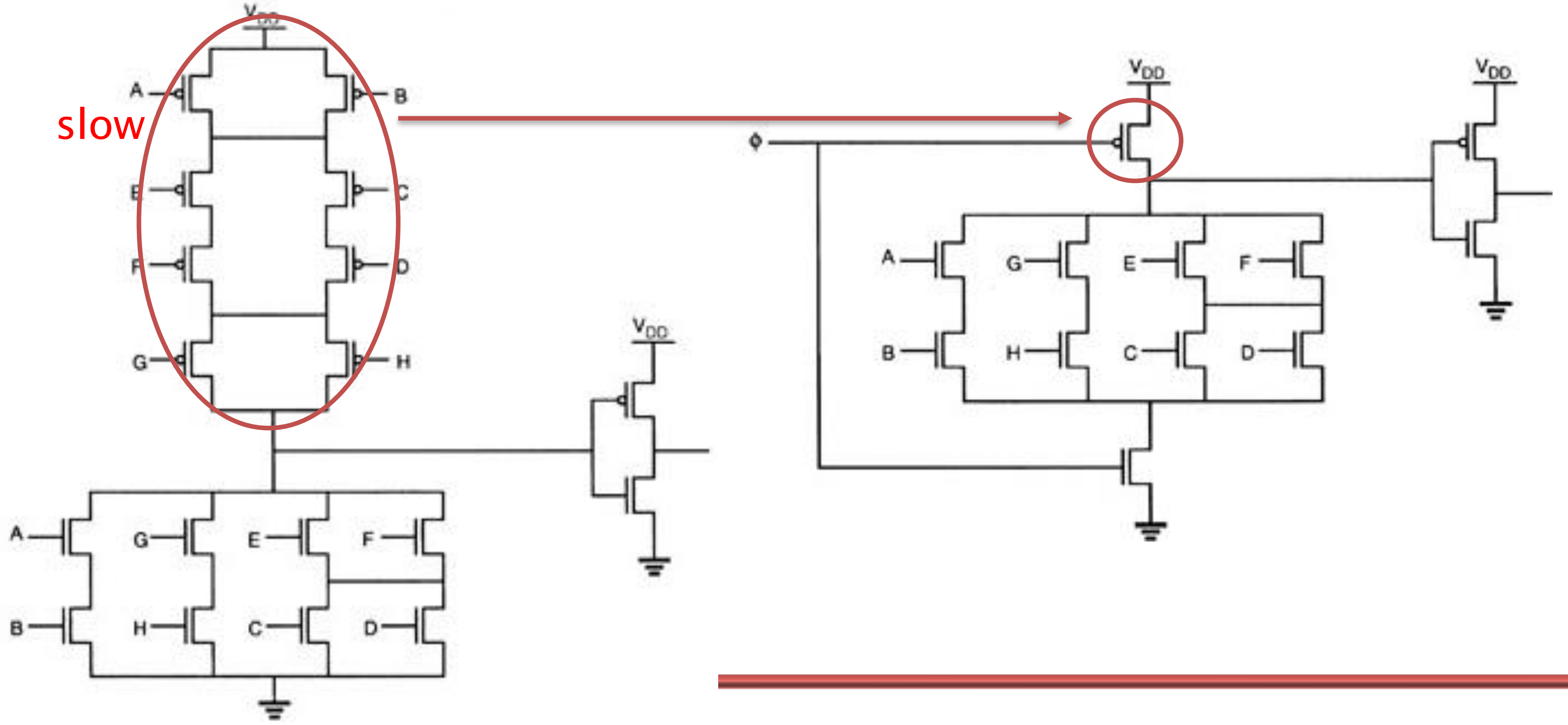
NOR2

Dynamic (N-MOS) Logic Examples

- **Dynamic (N-MOS) logic is particularly advantageous for high fan-in gates with large P-MOS stacks**

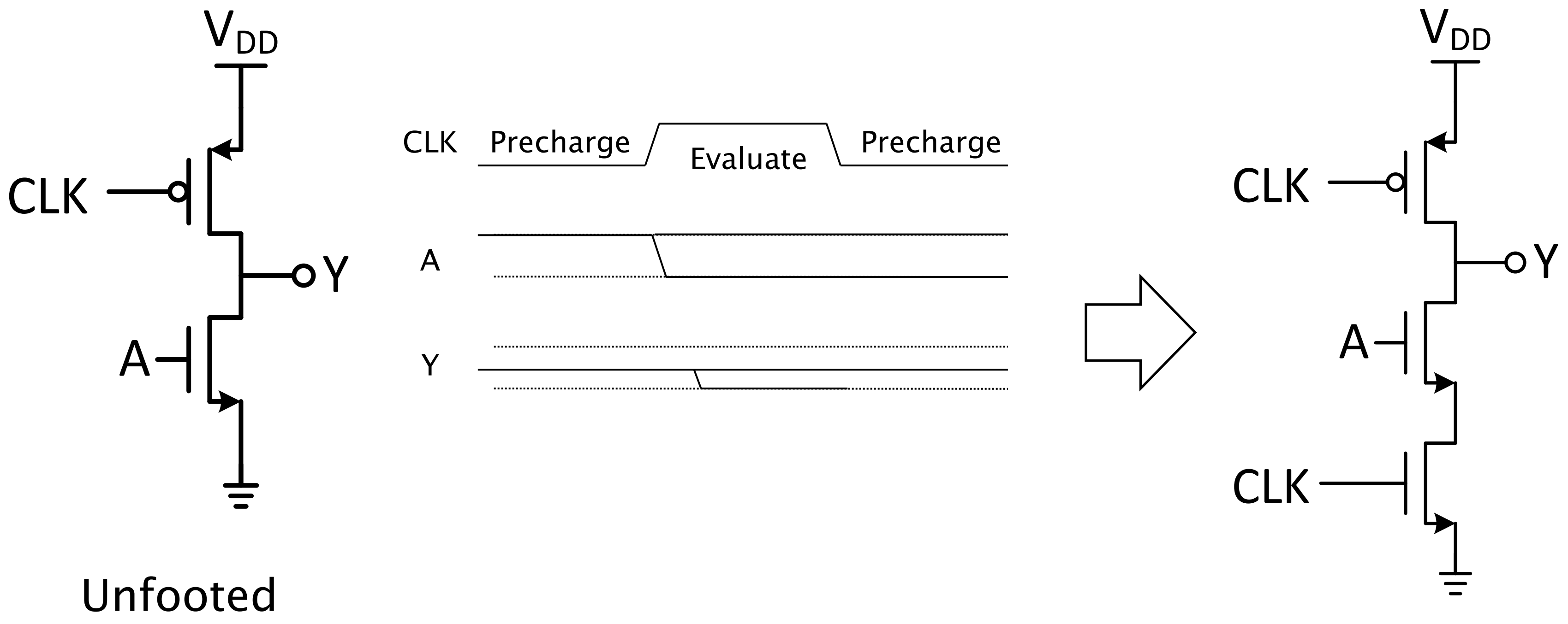
Using conventional CMOS logic

Domino CMOS logic



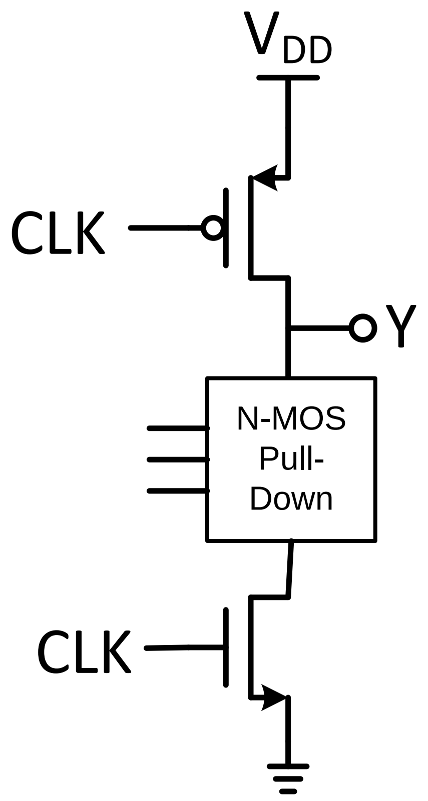
Footed Dynamic Logic

- **Problem of dynamic logic: contention during precharge when inputs can not guarantee that N-MOS network is OFF**
  - **Caused by the negative-unate function** of “CMOS” gates



Footed

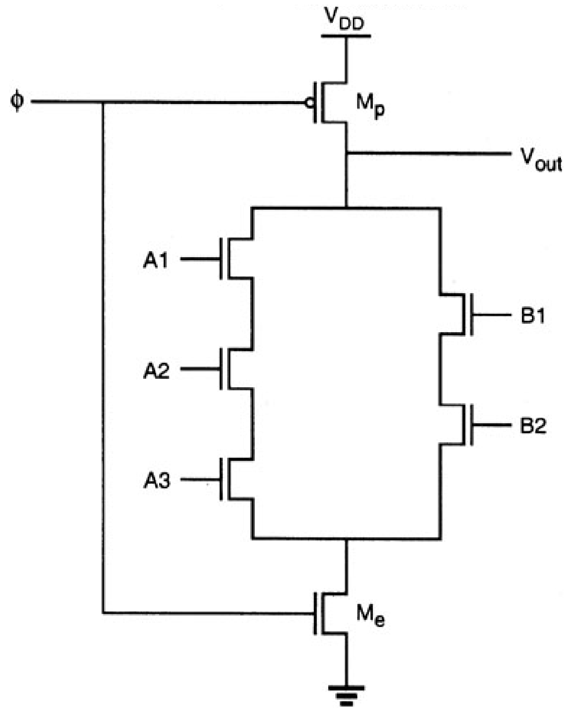
Footed Dynamic Logic

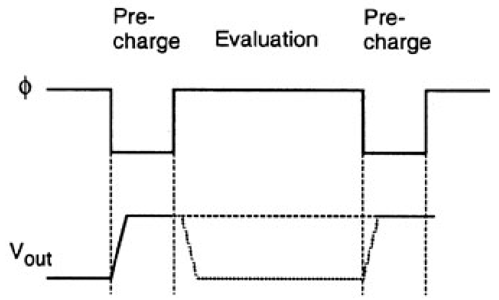


- **Footed Dynamic logic avoids contention during precharge (independent of the input)**
- **Adds one extra transistor**
  - N+1 => N+2
  - No impact on the input load
  - But, additional series resistance in pull-down path
  - Additional load on clock network
- **Necessary also always when cascading CMOS with dynamic logic**

Footed Dynamic Logic (Example)

- **Operation based on precharging and evaluating**
  - Clock = low : The pMOS charges the output node
  - Clock = high : Logic high -> V_{out} remains
    Logic low -> V_{out} drops

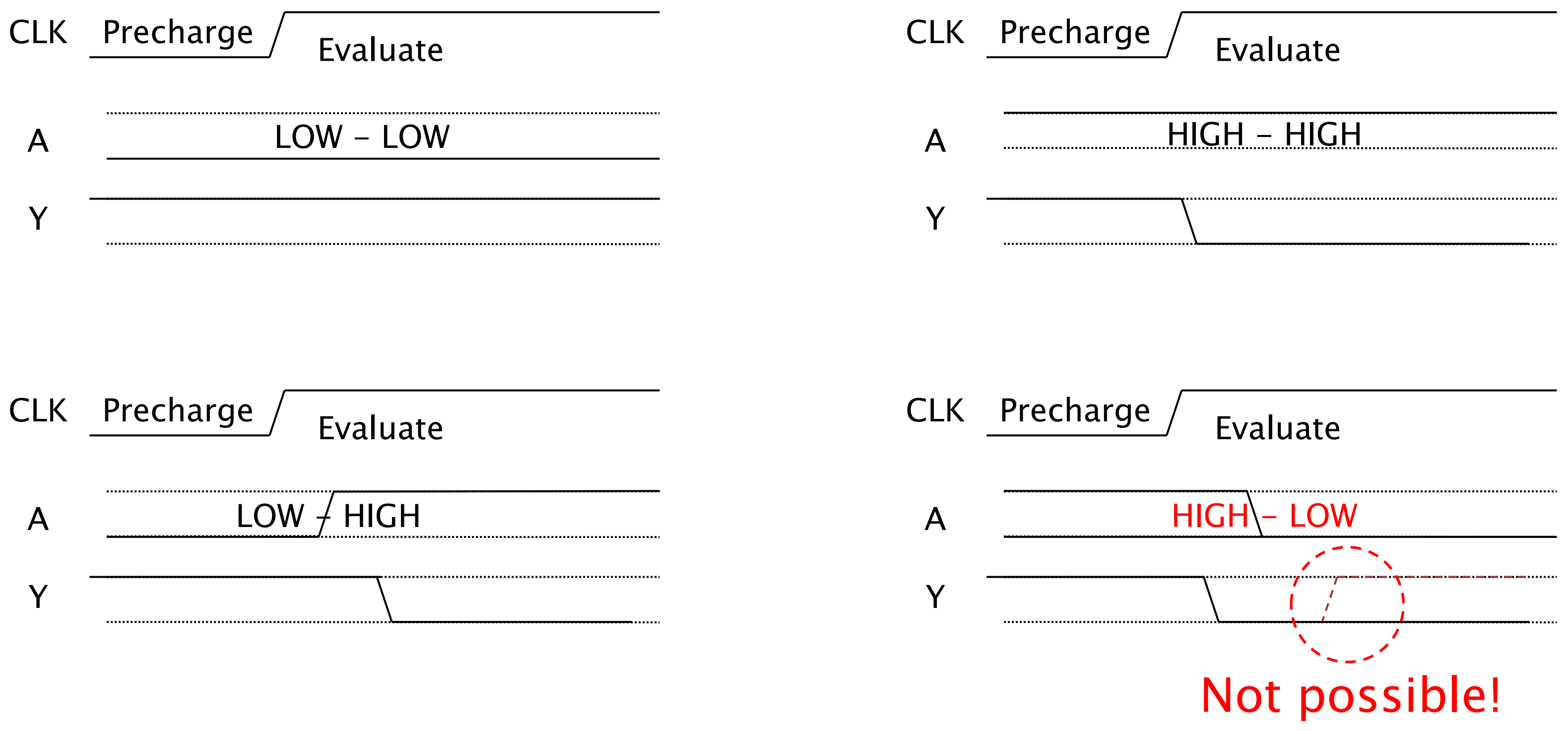




$$F = \overline{\left(A_1 A_2 A_3 + B_1 B_2\right)}$$

Monotonicity Problem

- **Footed dynamic logic does not solve all the issues!**
- **During evaluation, inputs must be monotonically rising**
  - Output can only have a single falling transition (no P-MOS network for pull-up)



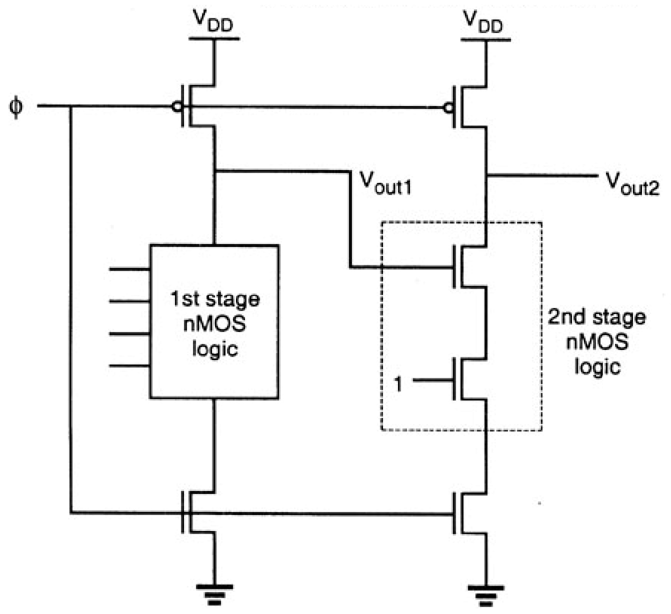
Cascading Dynamic Logic

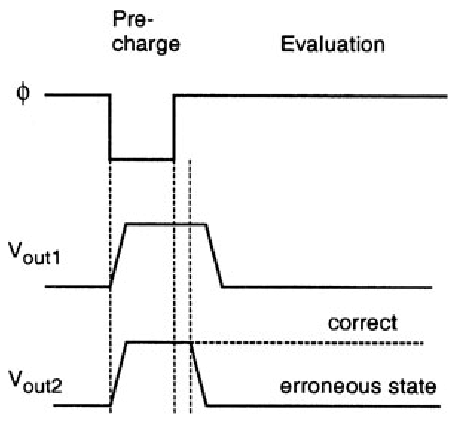
- **Output of any (also footed) dynamic (N-MOS) gate starts HIGH and conditionally falls (monotonically FALLING), while inputs must be monotonically RISING**
- **Example: 1st and 2nd stages evaluate at the same time**

  → Precharge output V_{out1} discharges V_{out2}

  → When V_{out1} evaluates to "0", V_{out2} can not rise anymore

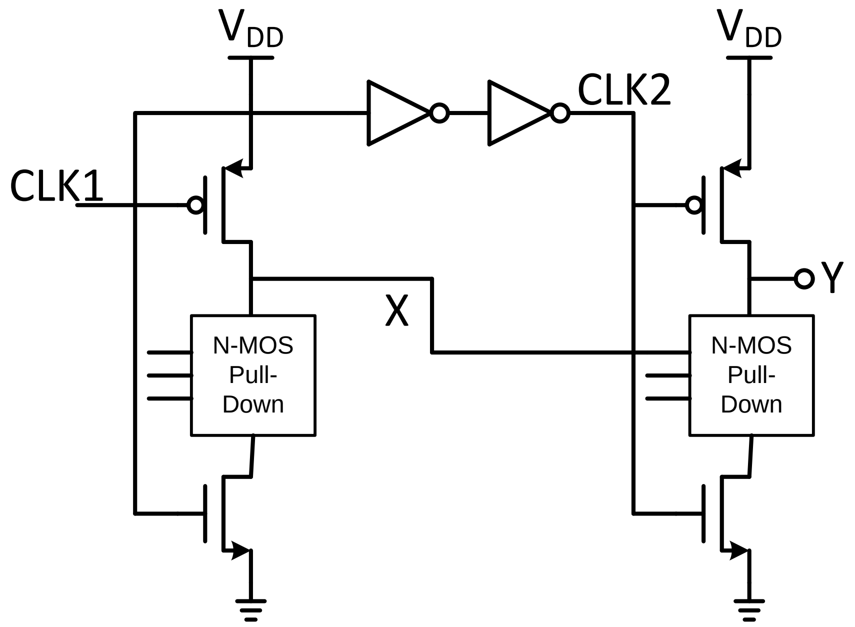
  → V_{out2} at the end of the evaluation phase will be erroneous

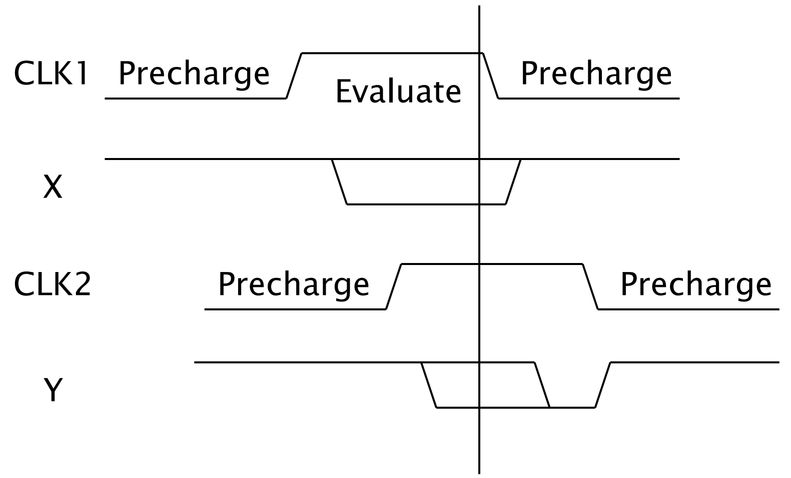




Cascading Dynamic Logic

- **Problem: Unexpected discharge in Vout due to the propagation delay of the previous stage**
- **Solution: Delay precharge and evaluation until output of previous stage has settled**

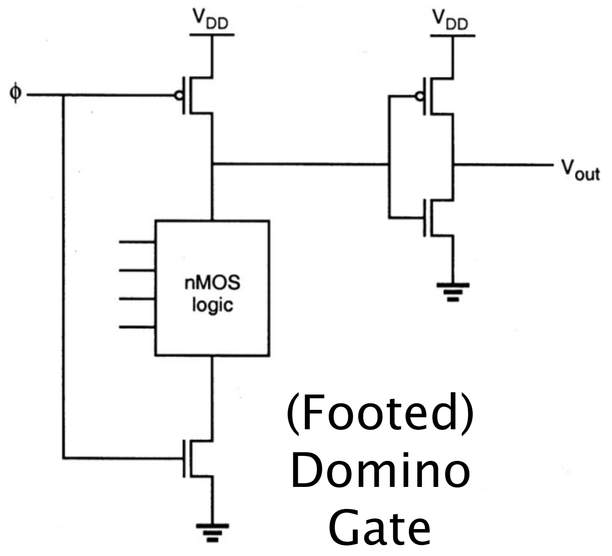


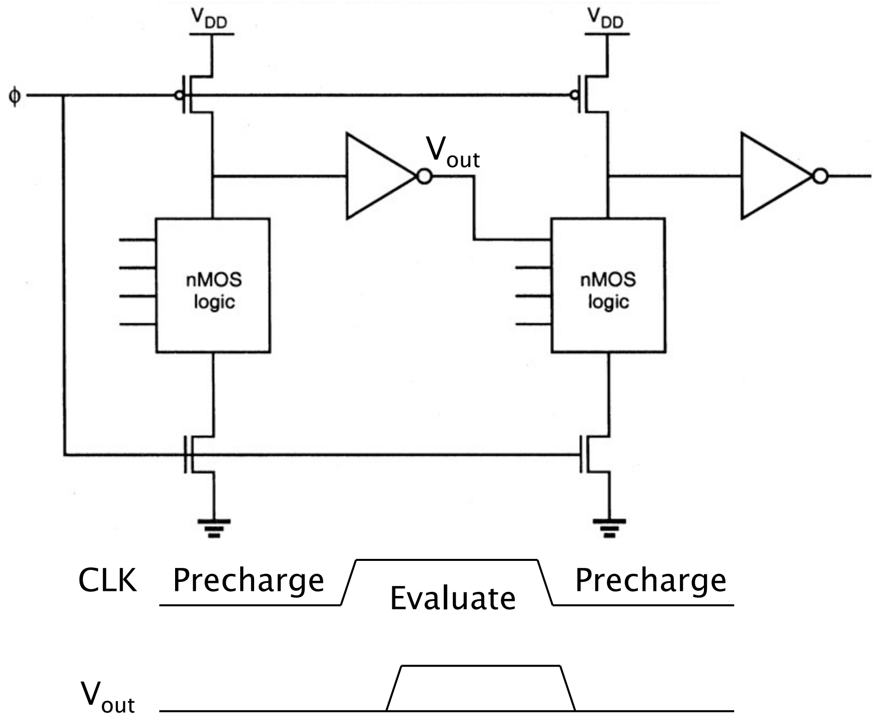


End of first gate precharge cycle ends the safe data valid window

Footed Domino Logic

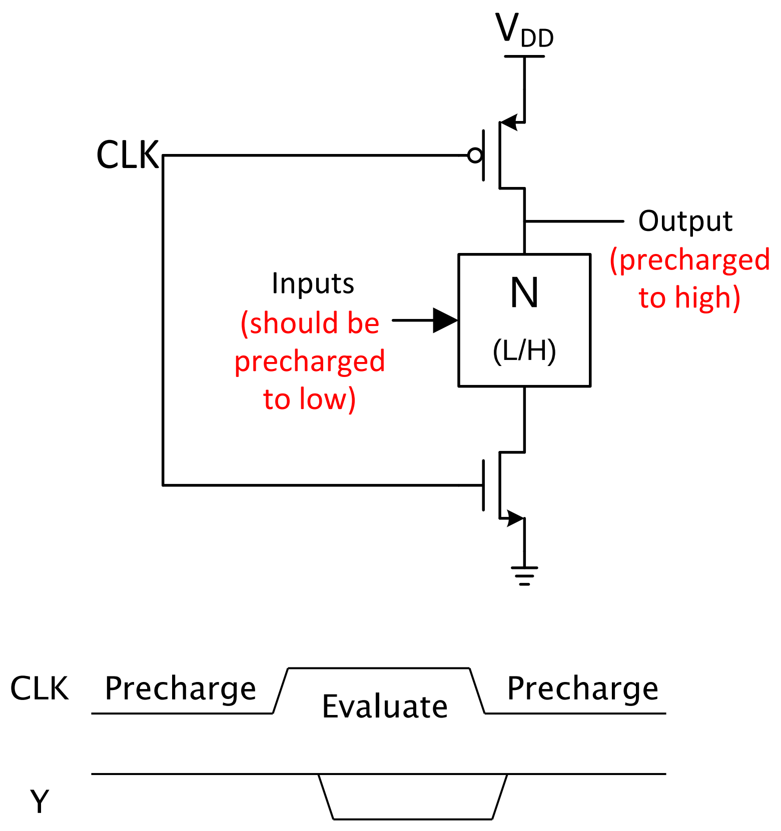
- **Problem: Unexpected discharge in V_{out} due to the propagation delay of the previous stage**
- **Solution: ensure that the output of each gate resets to "0" during precharge**
  - Insert a static INVERTER: V_{out} monotonically rising
  - Effectively work with positive-UNATE gates (instead of negative-UNATE)

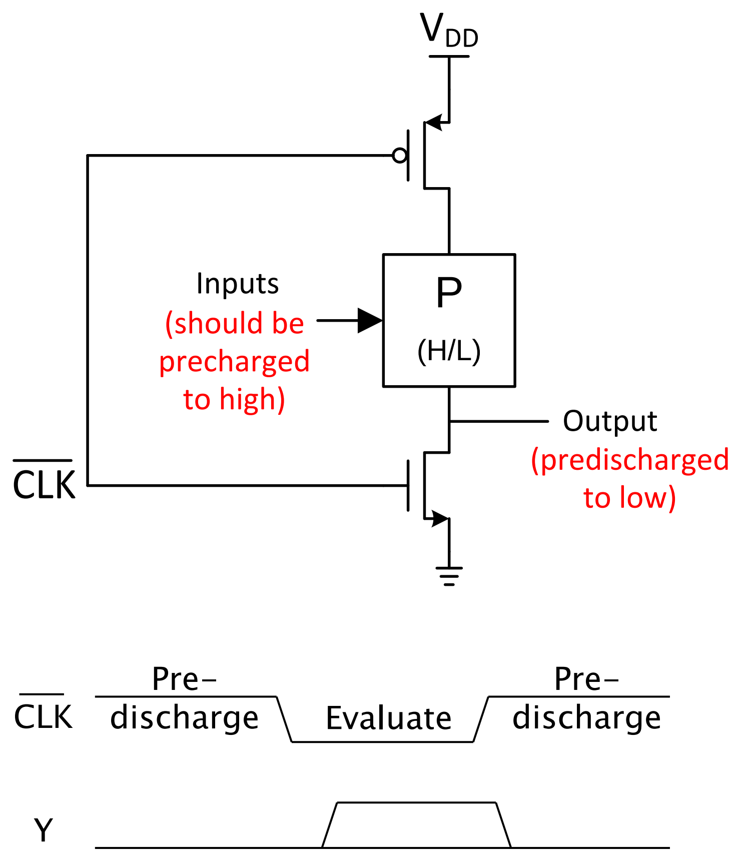




N- and P-Type Dynamic Logic

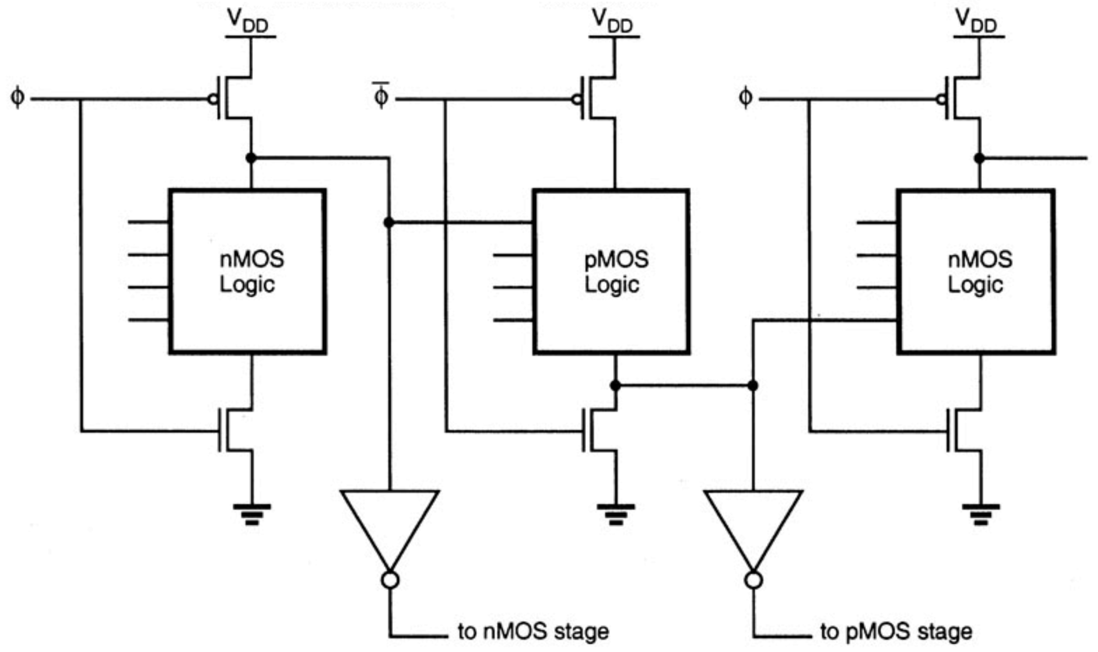
- **Idea: Dynamic logic gates with monotonically RISING output**

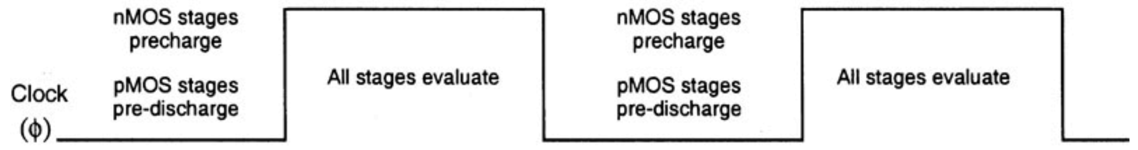




NP-Domino Logic

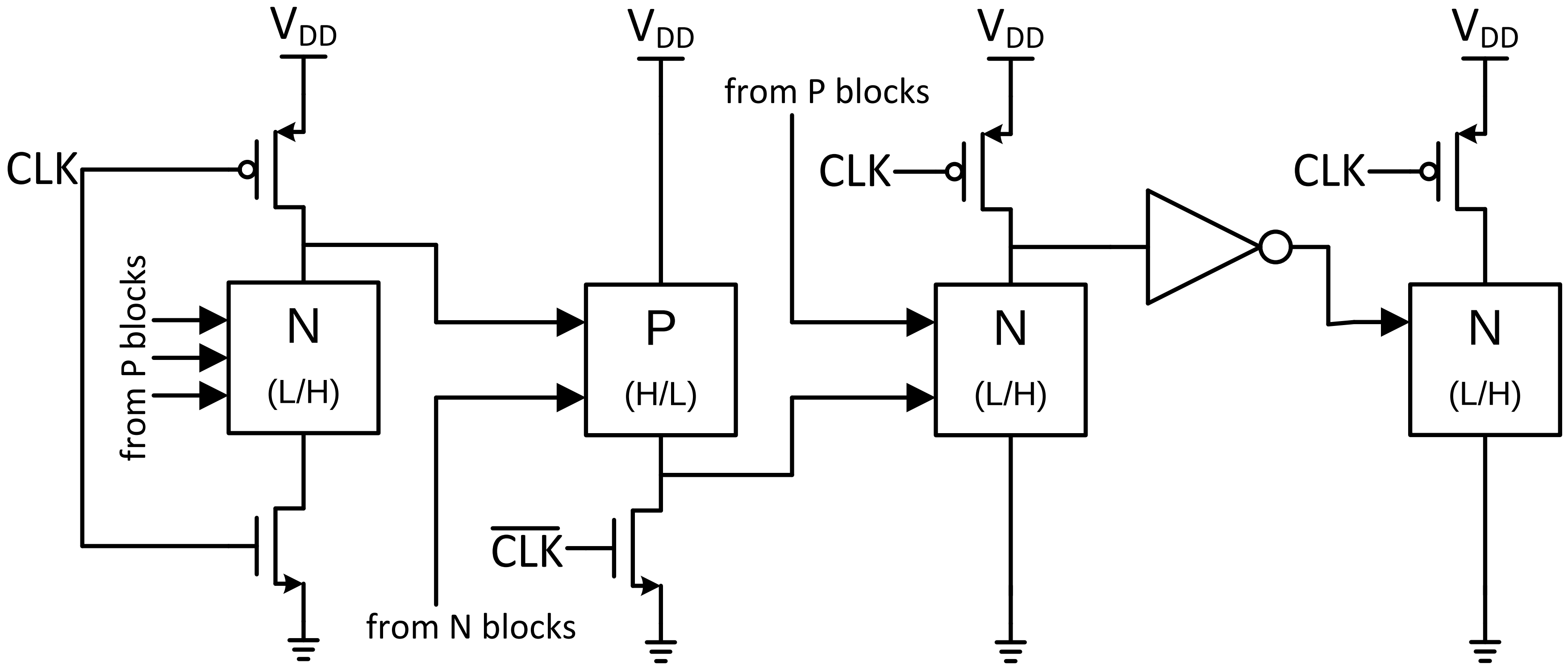
- **Dynamic logic stages using both nMOS and pMOS**
  - During Φ ='0' and $\overline{\Phi}$ ='1': —
    - nMOS logic pre-charge
    - pMOS logic pre-discharge
  - During Φ ='1' and $\overline{\Phi}$ ='0': —
    - nMOS logic evaluate (falling)
    - pMOS logic evaluate (rising)





Footed vs. Unfooted Logic

- **Footer is only needed to avoid contention during pre-(dis)charge**
- **Domino Logic and NP-Domino Logic can be unfooted**
  - Exception: 1st stage which interfaces to conventional logic

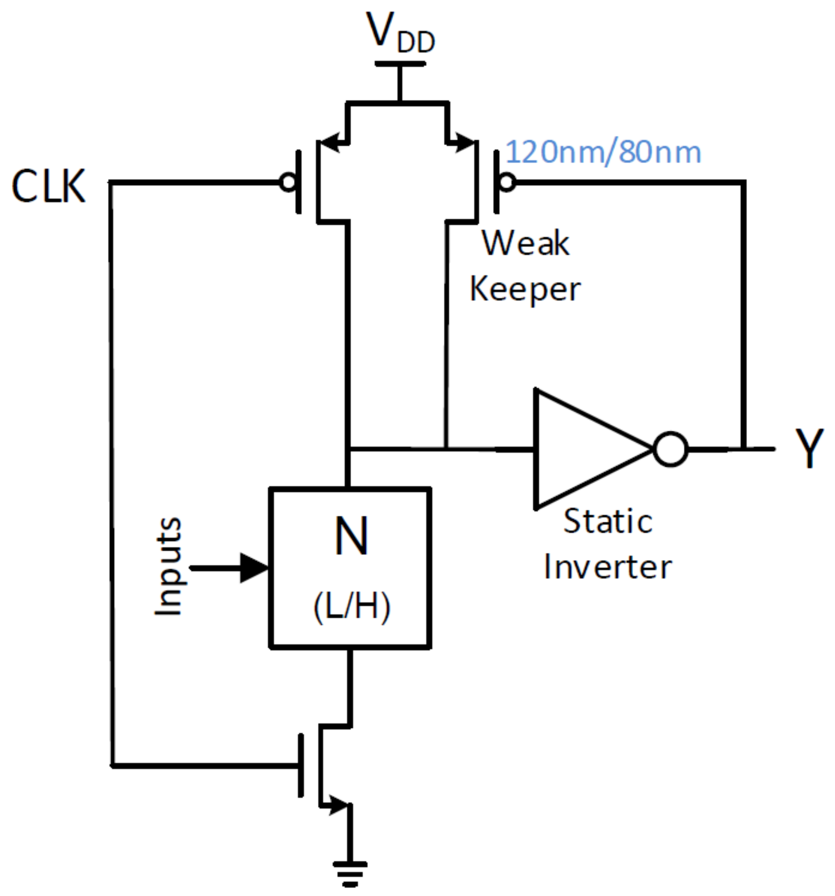


Leakage Currents

- **Problem: output is floating during evaluation for a “1”**
  - Sensitive to any kind of coupling
  - Leakage through OFF N-MOS network may deteriorate the logic “1”, especially for slow circuits

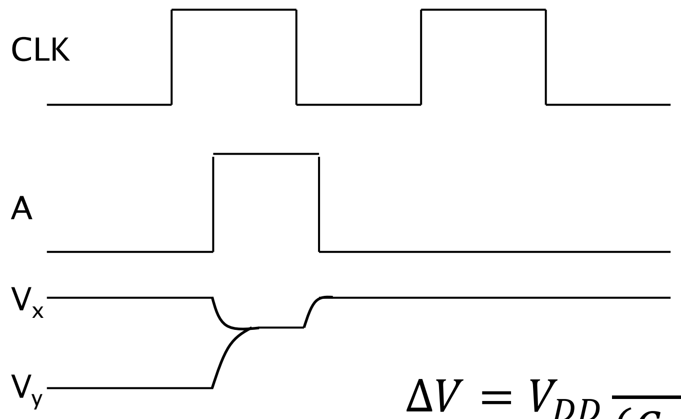
- **Solution: add a keeper to compensate for leakage and to restore logic “1” level**

- **Only small overhead for Domino gates**

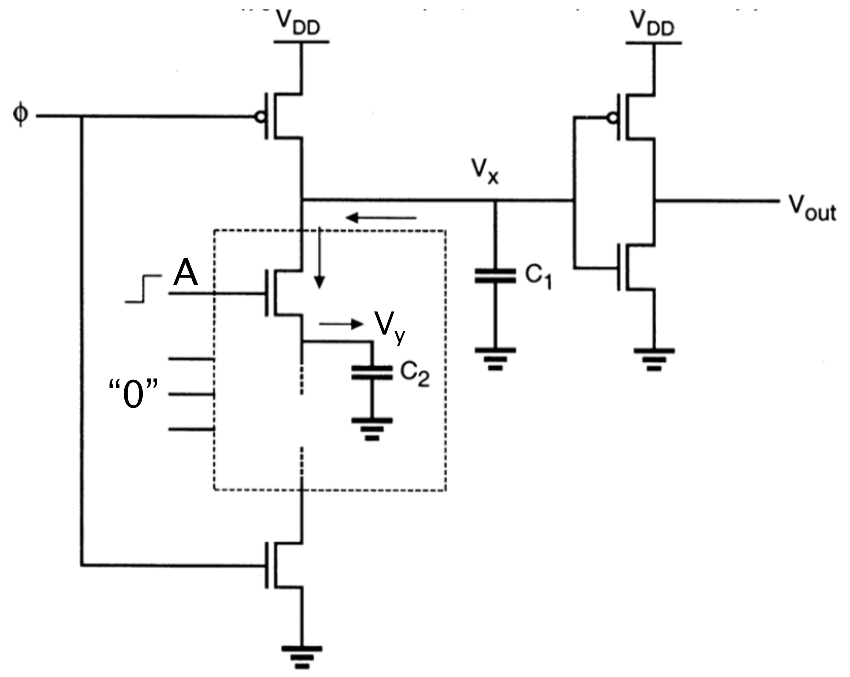


Charge Sharing Problem

- **Suppose output (C1) is precharged to "1", but in previous cycles, C2 was discharged to "0"**
  - During evaluation, input A rises
  - Other inputs remain "0", keeping the gate OFF
- **Charge sharing between C1 and C2 causes a drop in V_{out}**

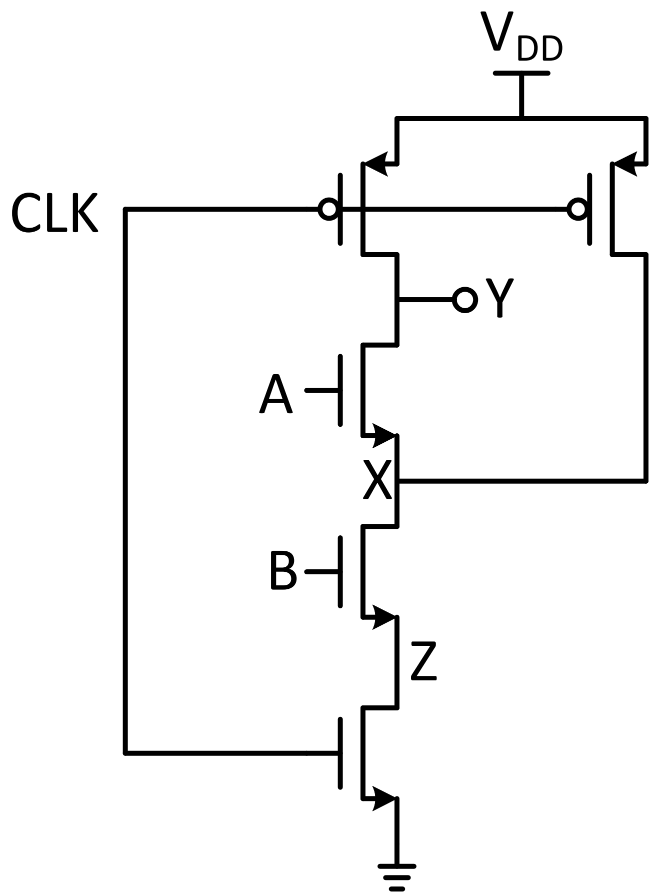


$$\Delta V = V_{DD}\frac{C_2}{(C_1 + C_2)}$$



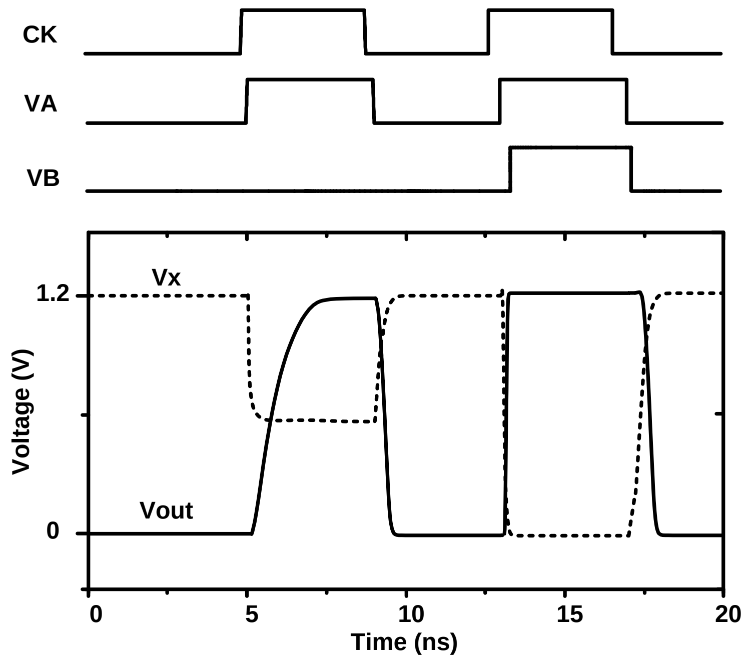
Charge Sharing Problem

- **Solution: precharge also critical (highly capacitive) internal nodes**
  - More transistors
  - Increases the clock load
  - Increases the intrinsic output load (additional delay)

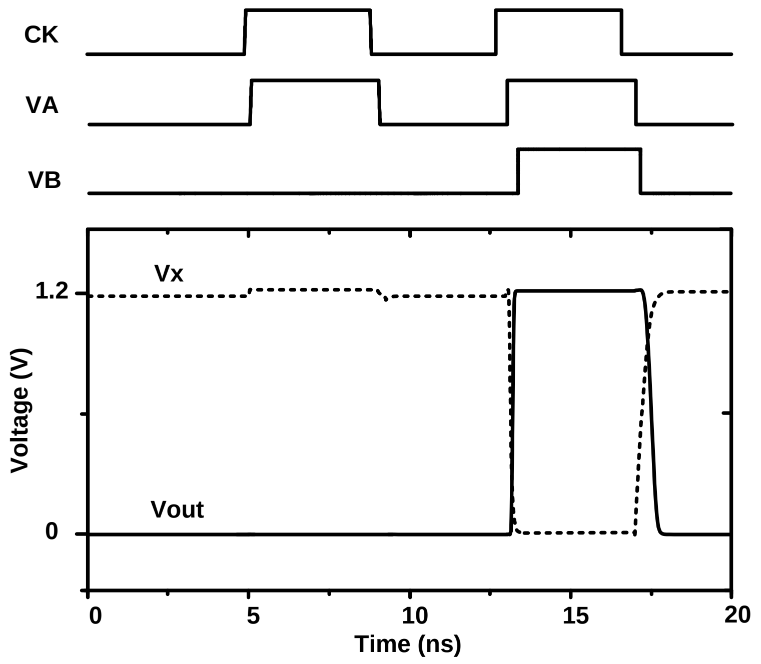


Charge Sharing Problem

- **Simulation results**



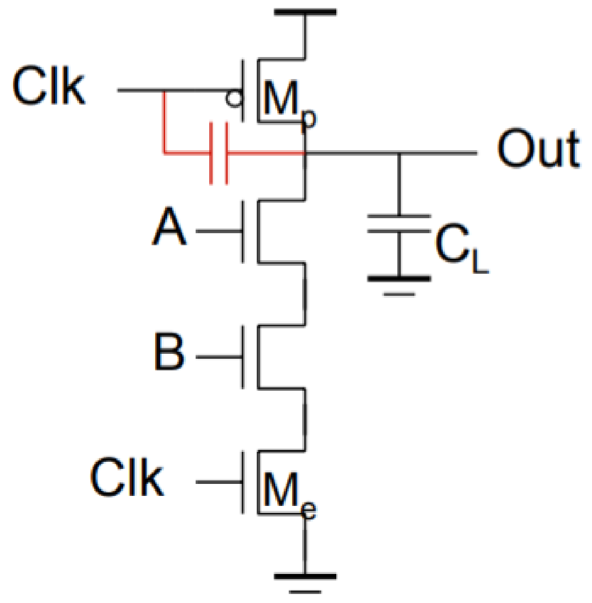
(a) w/o additional pMOS

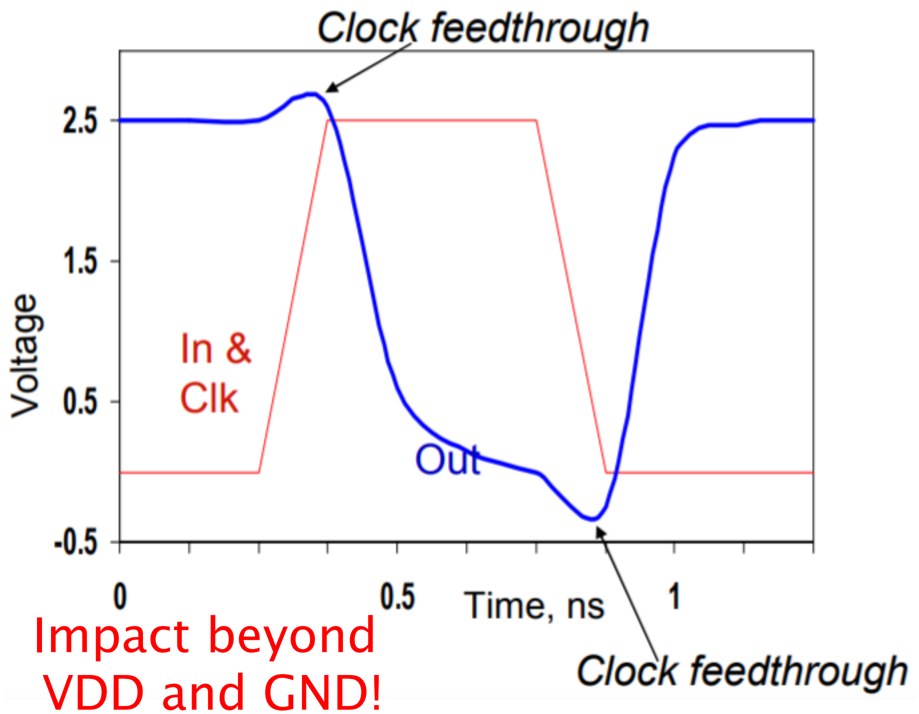


(b) w/ additional pMOS precharge Tr.

Clock Feedthrough

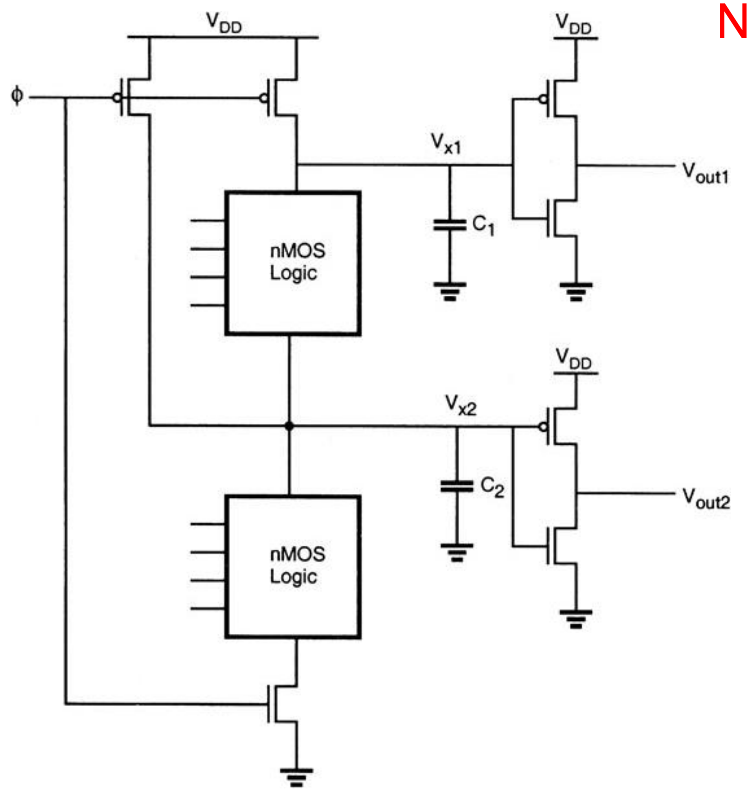
- **Capacitive coupling from clock to floating (output) node => negative impact on speed**
  - Rising clock turns off precharge transistor and raises floating node through capacitive coupling
  - Falling clock edge lowers floating node through coupling while precharge transistor is not yet conducting



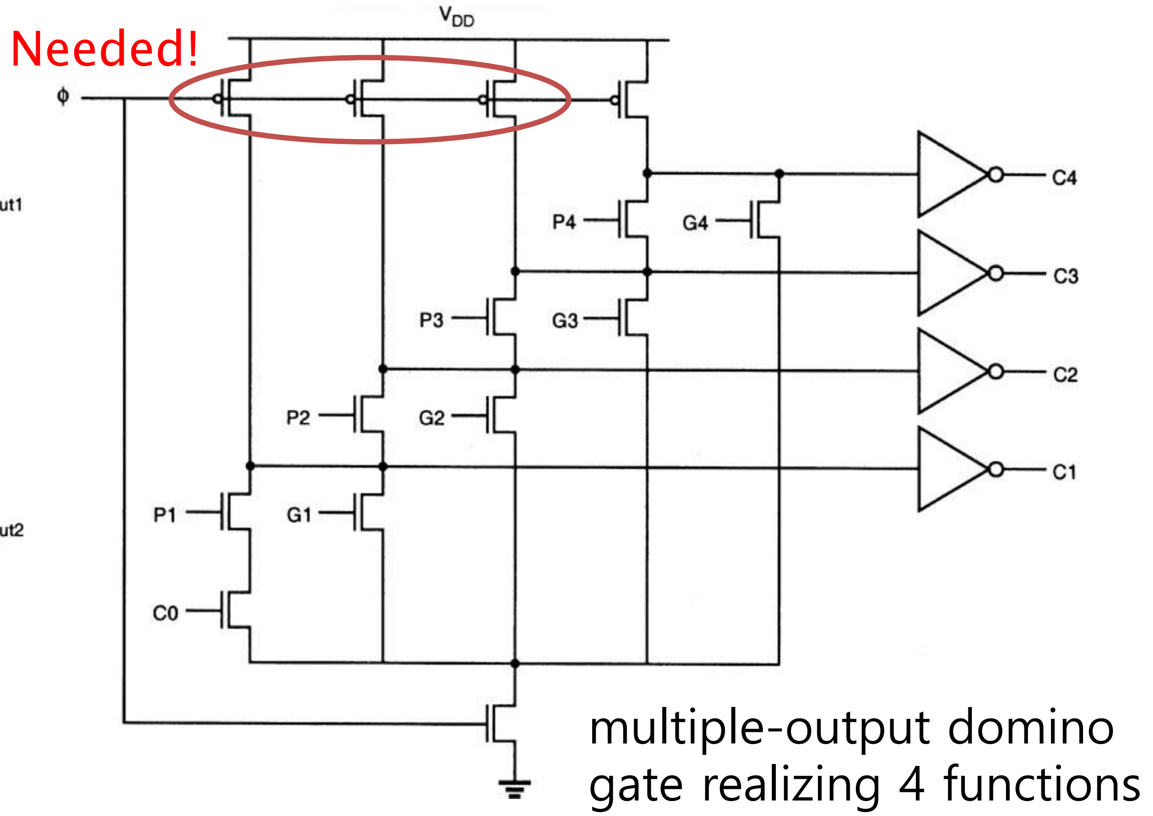


Impact beyond VDD and GND!

Multiple-Output Domino Structure



Multiple-output domino CMOS structures



multiple-output domino gate realizing 4 functions

$\overline{C_1} = \overline{G_1 + P_1 C_0}$ $\overline{C_2} = \overline{G_2 + P_2 C_1}$

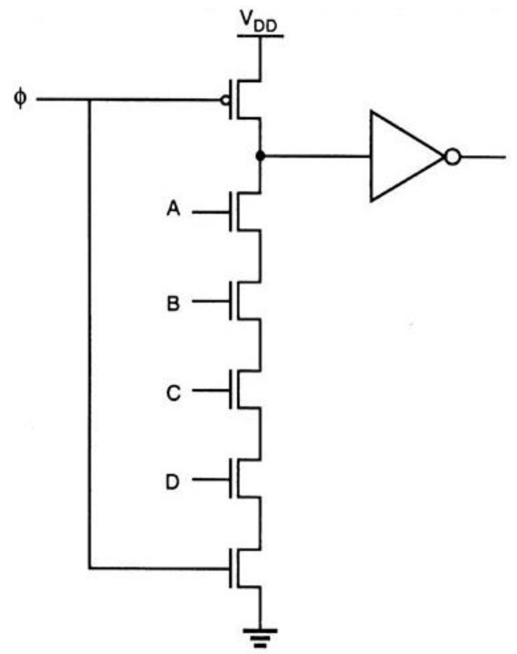
$C_1 = G_1 + P_1 C_0$ $C_2 = G_2 + P_2 G_1 + P_2 P_1 C_0$

$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$

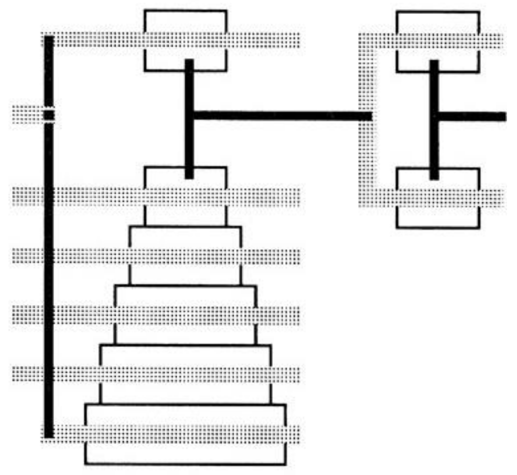
$C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_0$

Sizing

- **Precharge transistor (P-MOS): size to complete precharge in time (generally small) before the evaluation phase**
- **nMOS sizes should be increased from top to bottom   to lead a better transient performance**



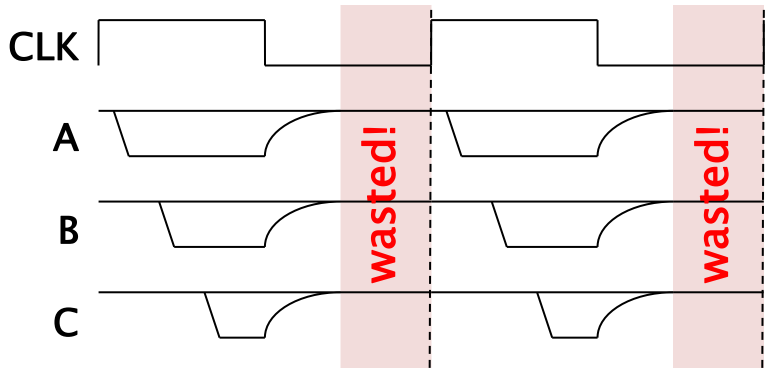
(a) Four-input domino CMOS NAND gate

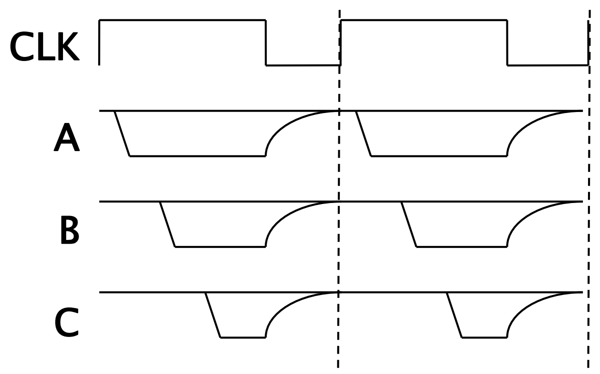


(b) Stick-diagram layout

Clocking

- **Most clocks have a 50/50 duty cycle**
  - Dynamic logic: 50% of cycle wasted for precharge
- **Solution: changing the duty cycle (pulsed clk)**
  - Allocate only the necessary time for precharge
  - **Tradeoff**: shorter precharge -> larger precharge transistors -> more power, stronger clock drivers, more intrinsic capacitance on output node
  - **Caveat** of pulse generation: quality of short pulses, propagating pulses through clock network, control over pulse width (variation)

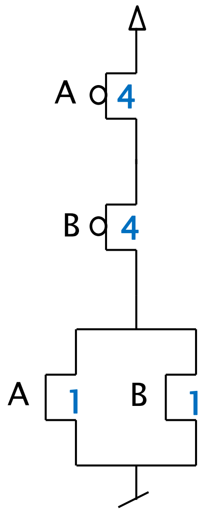




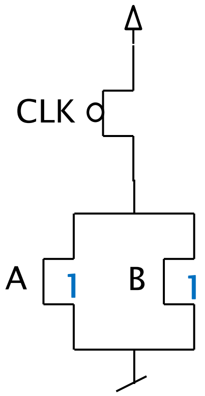
Logic Effort with NP Domino

- **Logical Effort Model can be applied roughly**
  - Remember: Logical-effort $g = \frac{C_{IN}^{gate}}{C_{IN}^{inv}}$

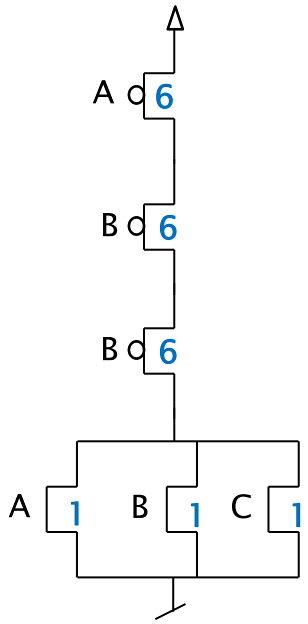
NOR in N−Domino



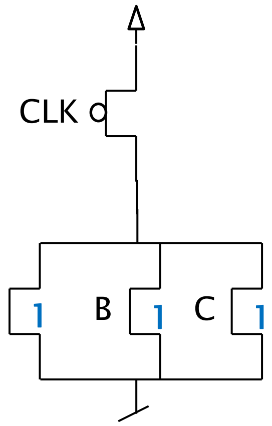
$g = 5/3$



$g = 1/3$



$g = 7/3$

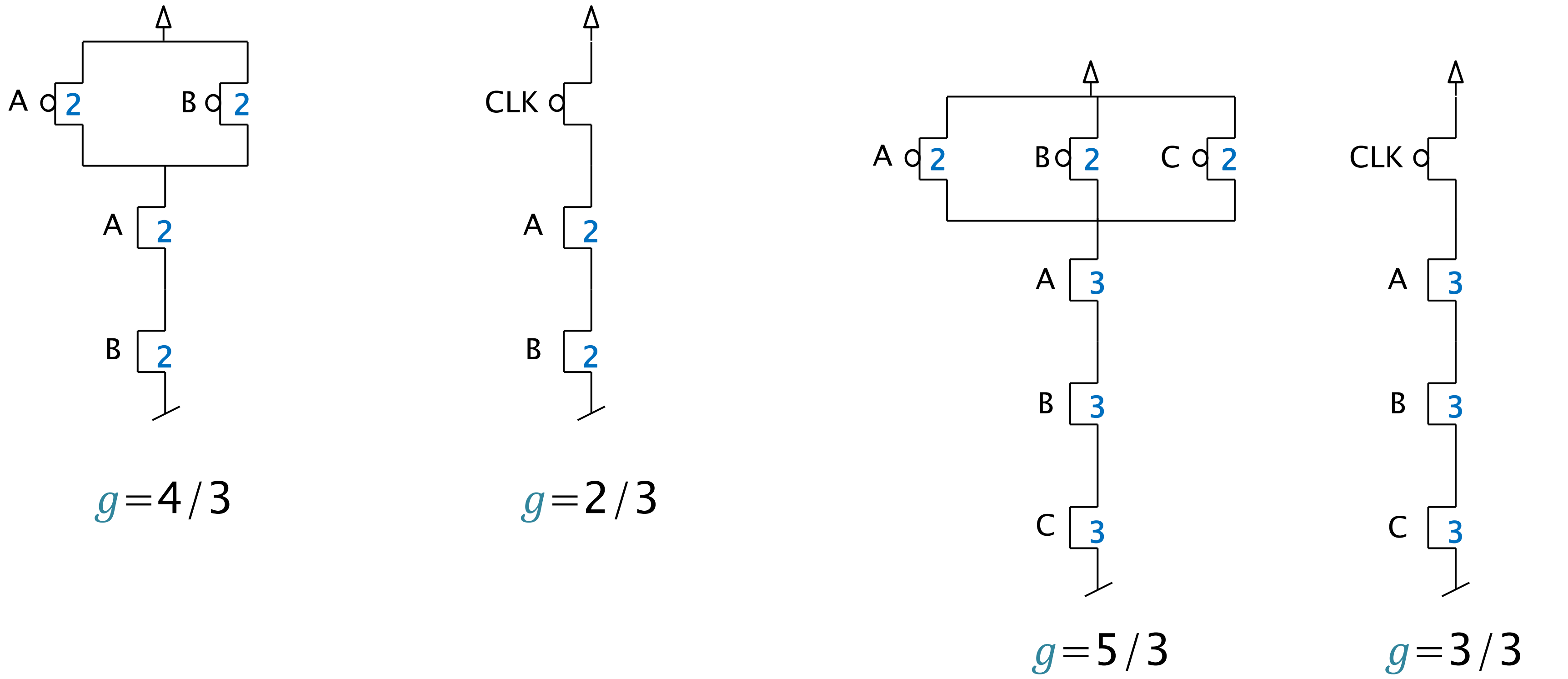


$g = 1/3$

Logic Effort with NP Domino

- **Logical Effort Model can be applied roughly**
  - Remember: Logical-effort $g = \frac{C_{IN}^{gate}}{C_{IN}^{inv}}$

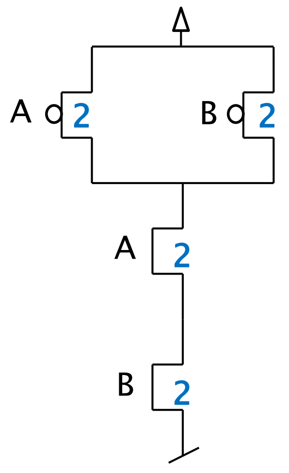
NAND in N−Domino



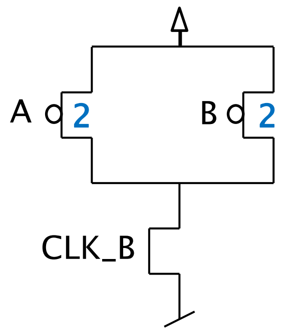
Logic Effort with NP Domino

- **Logical Effort Model can be applied roughly**
  - Remember: Logical-effort $g = \frac{C_{IN}^{gate}}{C_{IN}^{inv}}$

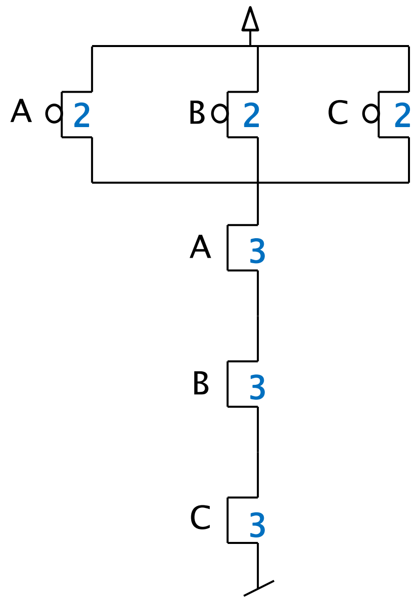
NAND in P–Domino



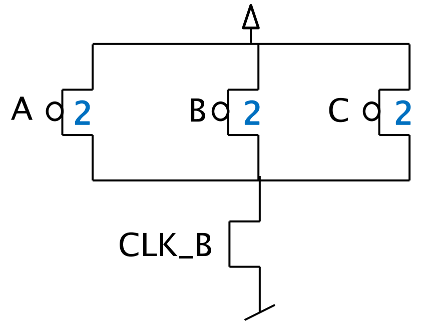
$g = 4/3$



$g = 2/3$



$g = 5/3$



$g = 2/3$