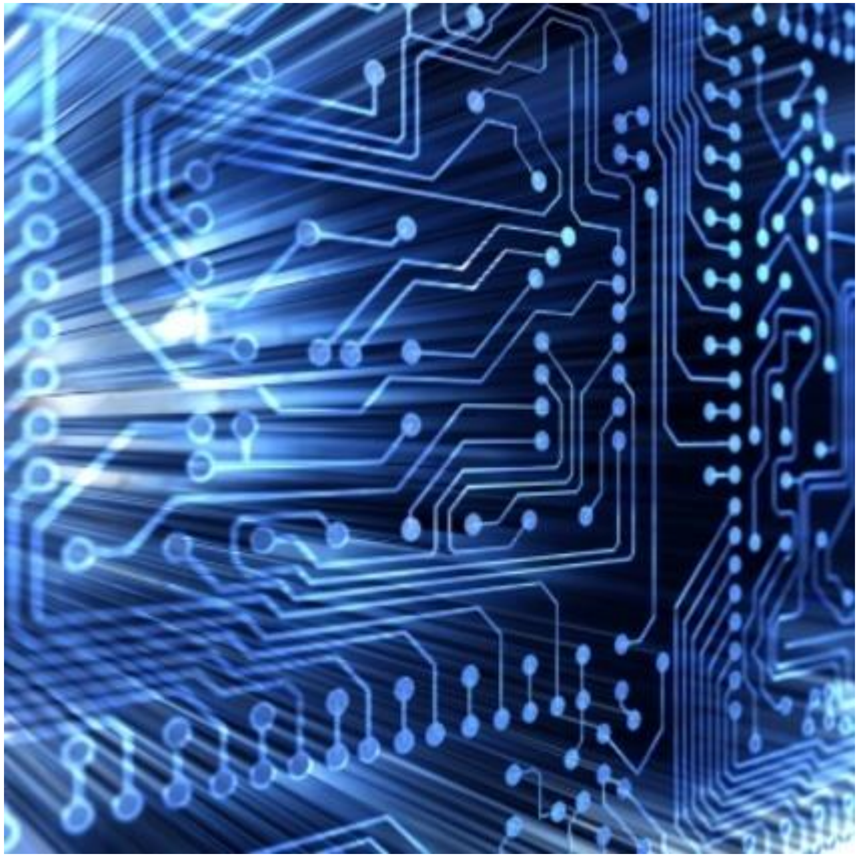
Digital Microelectronic Circuits (361-1-3021)

Presented by: Adam Teman



Lecture 10:

Dynamic Logic

Motivation

- Last lecture, we learned about *Pass Transistor Logic*.
- Using this technique (i.e. passing a signal through a diffusion input in addition to the gate input), we were able to reduce the number of transistors needed to implement several logic gates.
- However, pass transistors presented several disadvantages, such as V_T *drop*, static power dissipation, loss of regenerative property, and in certain situations, slow transitions.
- In this lecture, we will discuss another concept – d*ynamic logic* – and its implementation into an efficient and very fast logic family.

What will we learn today?

10.1 Dynamic CMOS

10.2 Charge Loss

10.3 Charge Sharing

10.4 Cascading Dynamic Gates

10.5 Logical Effort of Domino

10.1

10.1 Dynamic CMOS

10.2 Charge Loss

10.3 Charge Sharing

10.4 Domino Logic

10.5 LE of Domino

So now we’d like to reduce some gates and speed up our calculations. It’s time for

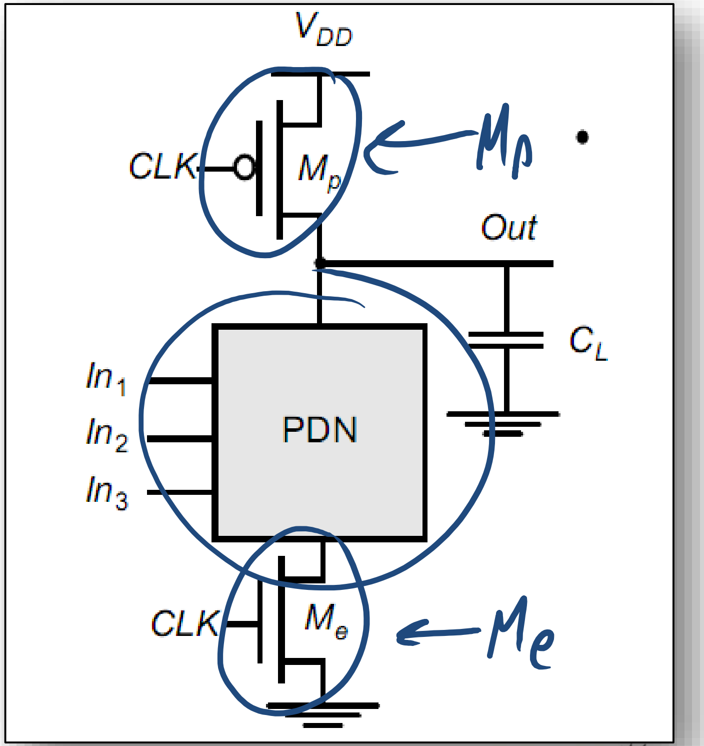
DYNAMIC LOGIC

Introduction

- So far, we've seen that:
  - » ***Standard CMOS*** require ***2N transistors*** for ***N inputs***.
  - » ***Pseudo nMOS*** requires only ***N+1 transistors***, but has ***high static power consumption***.
  - » ***PTL*** is only efficient for certain functions
- An alternative logic style called ***Dynamic Logic*** provides:
  - » ***N+2 Transistors*** for ***N Inputs***
  - » ***Low Static Power Consumption***
  - » ***High Operation Speed***
- ***Dynamic Logic*** provides a very different approach to gate implementation, defining it as a completely different style than the previously discussed ***Static*** families.

Dynamic Logic Concept

- ***Dynamic Circuits*** operate in ***two phases***:
  - » ***Precharge***: set an ***initial output state***
  - » ***Evaluation***: change the precharged output to the ***legal state***.
- This is done with a basic architecture that includes:
  - » A standard ***PDN*** network
  - » Complementary ***precharge switches***
- This is an "***n-type***" network. The same can be accomplished using a "***p-type***" ***Pull-Up*** network.

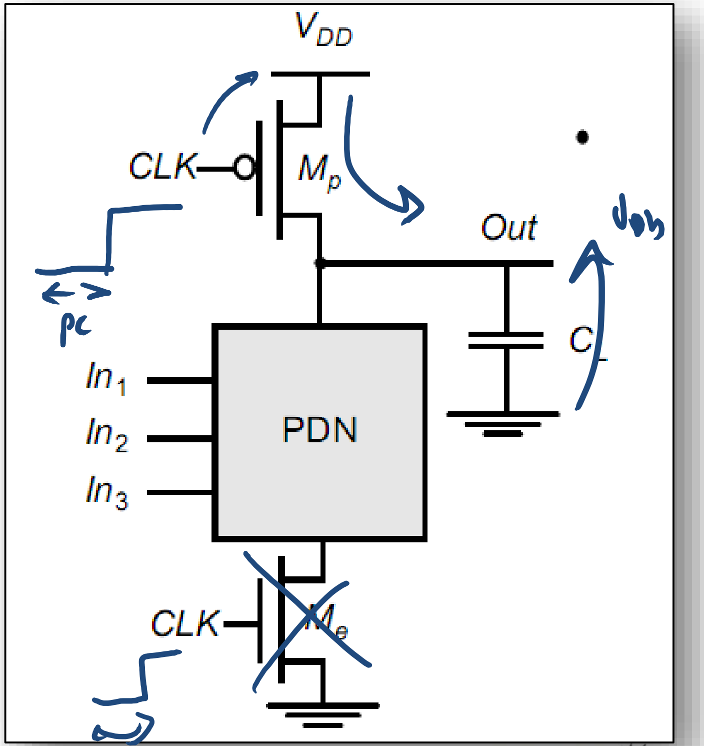


Dynamic Logic Concept - Precharge

- ***Precharge*** occurs when the ***clock is low***, ***blocking*** the ***discharge path*** and ***enabling*** the ***pull up path***.

- The ***output capacitance*** is charged to ***'1'*** through the ***top pMOS*** (the ***Precharge Transistor***).

- The ***bottom nMOS*** eliminates ***static current*** and ***ratioed behavior.***



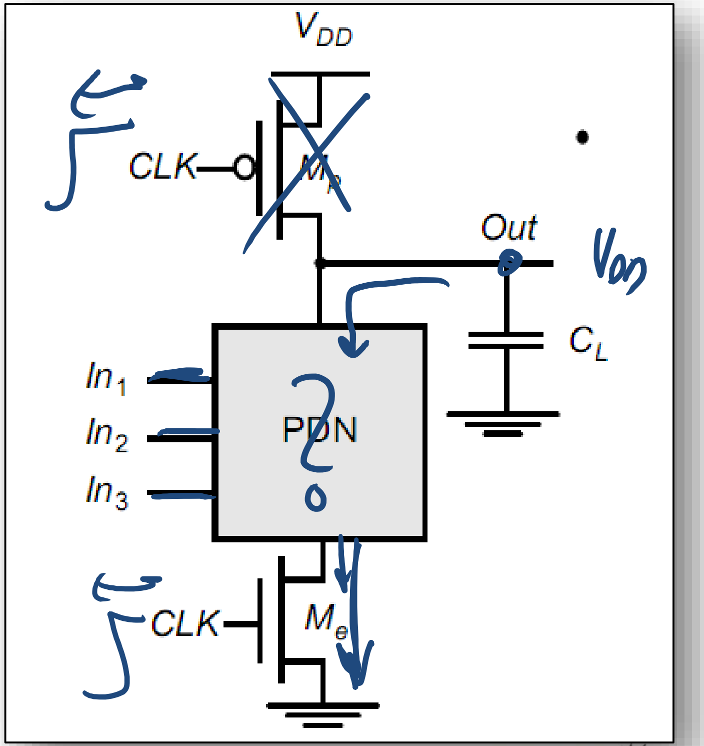
Dynamic Logic Concept - Evaluation

- ***Evaluation*** occurs when the ***clock is high***.

- The ***Precharge Transistor*** is turned ***off***, blocking any additional charge from ***flowing to the output capacitance***.

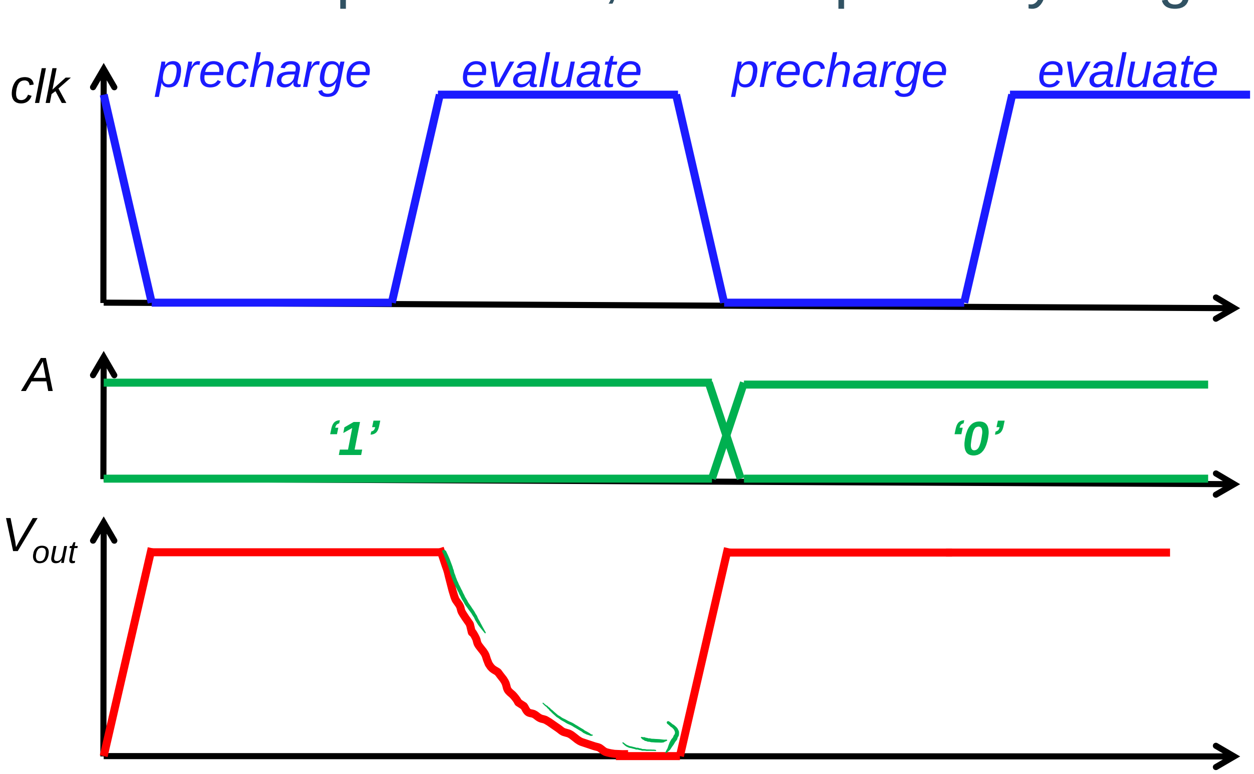
- The ***bottom nMOS*** (the ***Evaluation Transistor***) is turned ***on***, enabling a ***conditional path to ground***.

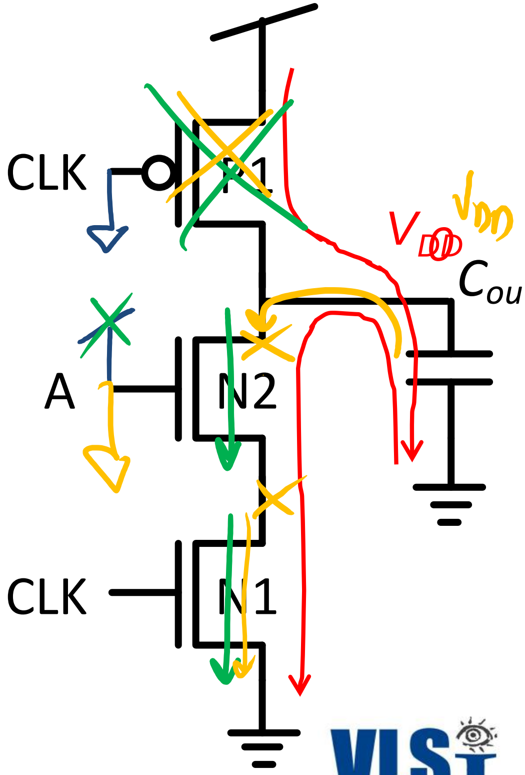
- The ***output is discharged***, depending on the ***input values*** and the ***combinational function*** of the ***PDN***, similar to ***static logic*** families.



Example – Dynamic Inverter

- When the clock is low, the capacitance is charged.
- When the clock goes high, the output is discharged if the input is high.
- If the input is low, the output stays high.

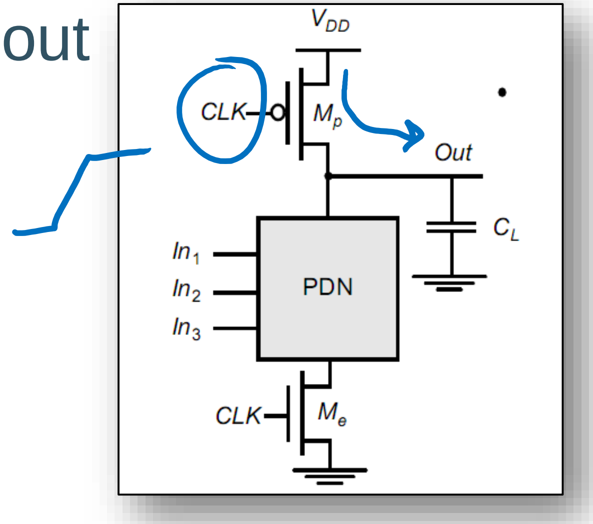




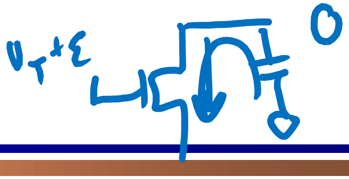
Dynamic Logic Properties

$P_{dyn} = \alpha f C V_{DD}^2$

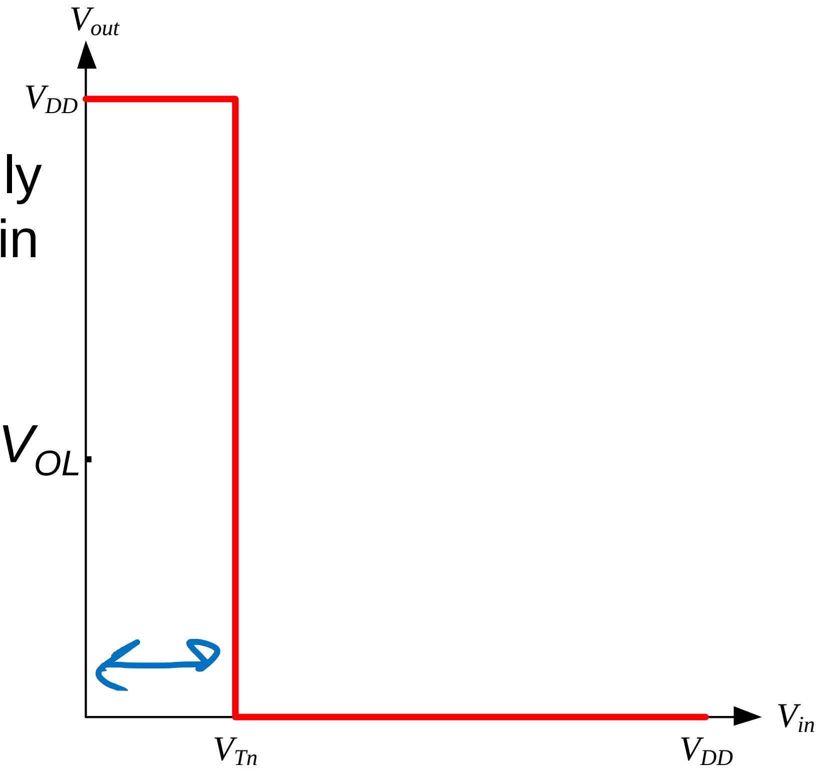
- The operation of ***Dynamic Logic*** brings about a number of important properties:
  - ***N+2 Transistors*** are required for gate implementation.
  - The logic is ***Non-Ratioed*** (i.e. ***sizing*** doesn't affect ***functionality***).
  - ***Static Power Consumption*** is ***low***, but ***Dynamic Power Consumption*** is significantly ***higher*** than ***Standard CMOS***.
  - The ***Switching Speeds*** are ***higher*** than ***Standard CMOS*** due to ***Reduced Load Capacitance***, ***Zero Short-Circuit Current*** and ability to ***Optimize Only One Swing*** (t_{pHL}).



Dynamic Logic VTC

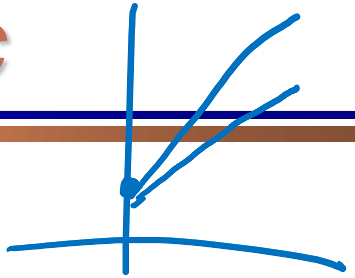


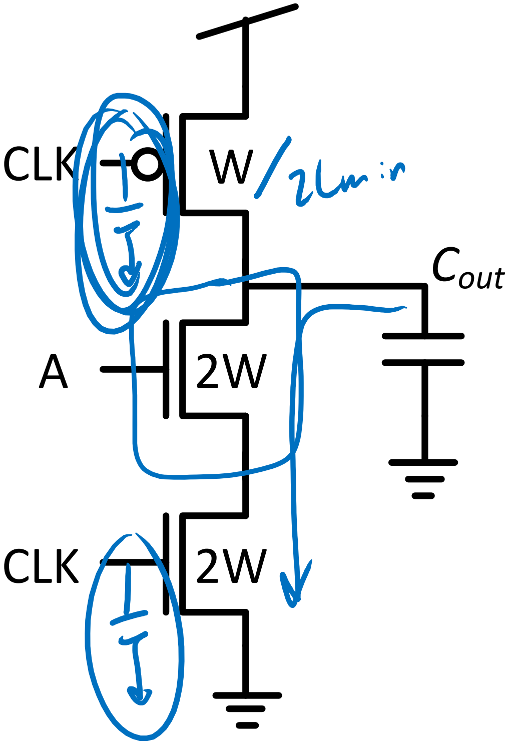
- An Interesting (Bizarre) VTC
  - Remember – assume DC…
  - Until V_{Tn}, the output is V_{OH}.
  - Once we pass V_{Tn}, there is no partially open pMOS combating the PDN, so in a DC perspective, output will fully discharge.
  - The VTC drops STRAIGHT down to V_{OL}.
  - (If not DC, but bounded with time, the VTC will be more gradual…)



- $NM_L = V_{Tn}$. This is very low!

Logical Effort of Dynamic Logic





$\underline{\mathrm{t_{pHL}}}$:

$$R_{eq} = 2\,{}^{R_n}\!/_{2} = R_{inv}$$

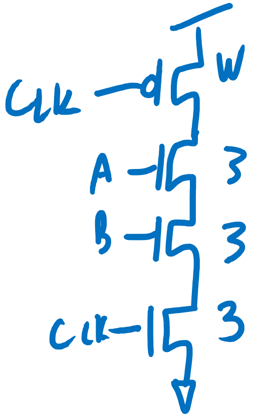
$$C_g\left(A\right) = 2C_{g\min}$$

$$C_d = 3C_{d\min}$$

$$p = \frac{R_{gate}}{R_{inv}} \cdot \frac{C_{d,gate}}{C_{d,\min}} = \frac{3}{3} = 1$$

$$LE = \frac{R_{gate}}{R_{inv}} \cdot \frac{C_{g,gate}}{C_{g,\min}} = \frac{2}{3}$$

$\underline{\mathrm{t_{pLH}}}$: $p = 0,\ LE = 0$



$R_{eq} = R_{min}$

$C_{gA} = 3$

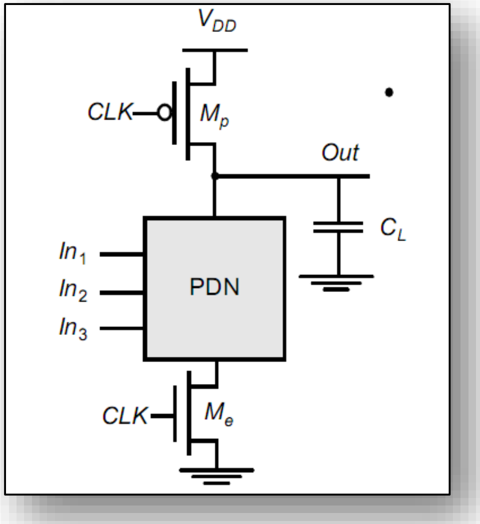
$C_{gB} = 3$

$C_d = 4$

- Therefore, we get a very fast gate!
- For a 2-input *NAND*, we get:

$$\boxed{p_{NAND} = \frac{4}{3},\ \ LE_{NAND} = 1}$$

Dynamic Logic Problems



- ❑ The basic consideration of using ***Dynamic*** vs. ***Static Logic*** is ***Speed*** vs. ***Power***.

- ❑ However, ***High Output Degradation*** can occur in ***Dynamic Logic*** in the following cases:
  - » Input Glitches ✓
  - » Leakage Currents ✓
  - » Charge Sharing ✓
  - » Cascading dynamic gates ✓

10.2

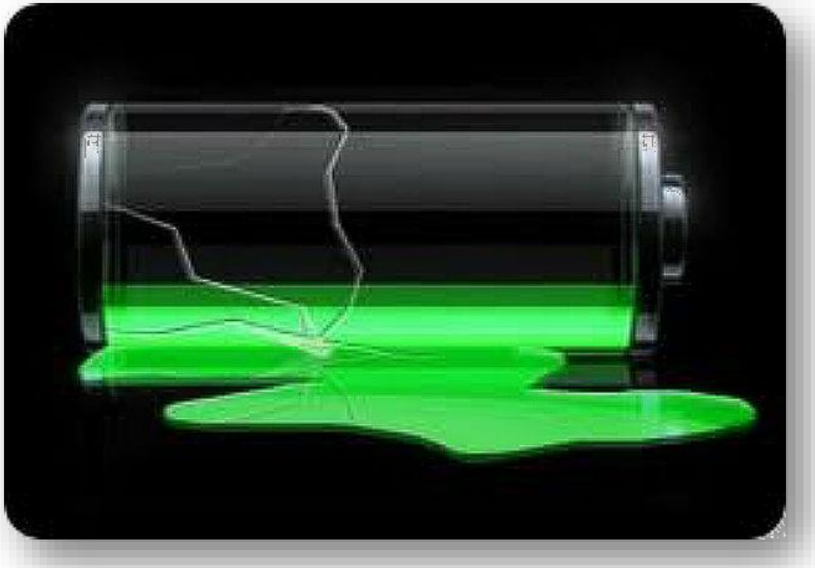
10.1 Dynamic CMOS

10.2 Charge Loss

10.3 Charge Sharing

10.4 Domino Logic

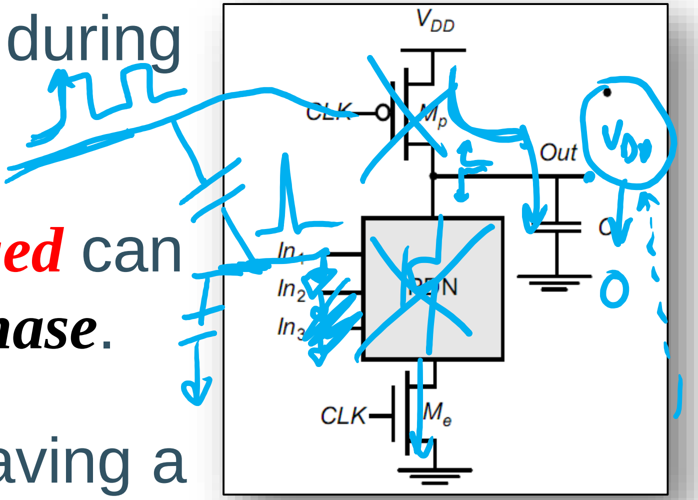
10.5 LE of Domino



The Big Problem with Dynamic gates is their

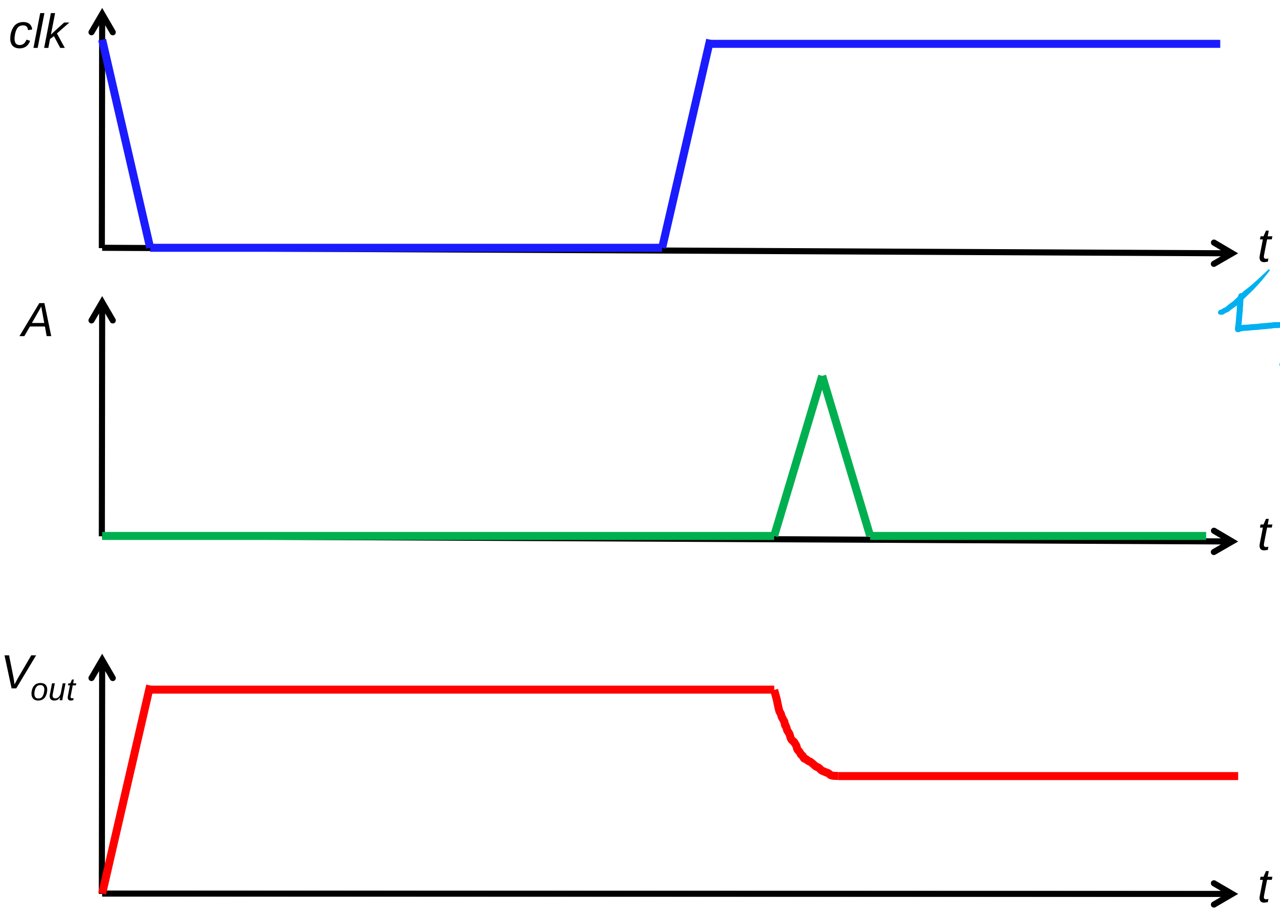
CHARGE LOSS

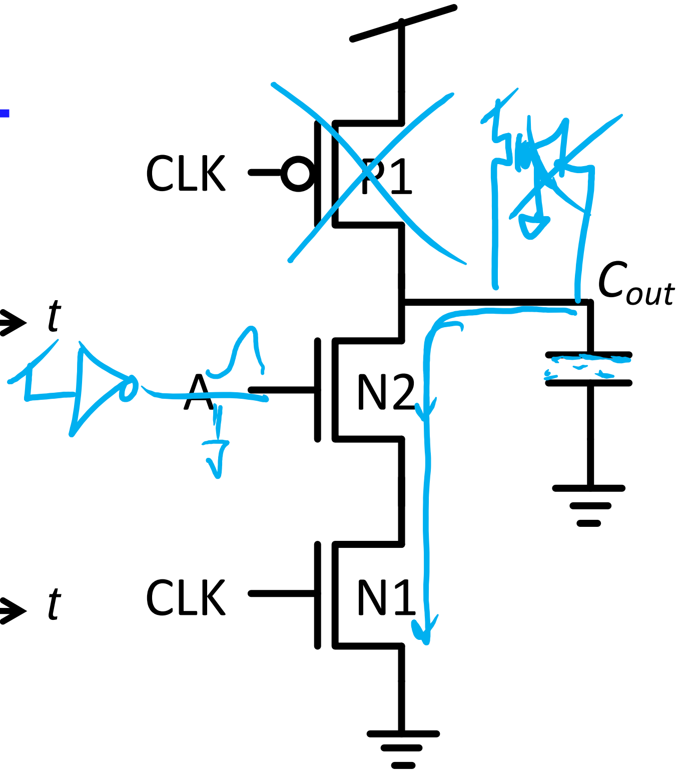
Problem #1: Charge Loss



- ❑ ***Dynamic Logic*** only ***pulls up the output*** during the ***Precharge Phase***.
- ❑ Therefore, any current that is ***discharged*** can only be ***replenished*** at the ***next clock phase***.
- ❑ If a ***low input*** (that ***cuts off the PDN***, leaving a ***high output***) "***glitches***" from ***'0'*** to ***'1'***, a ***path to ground*** temporarily opens, ***discharging*** some of the output.
- ❑ In addition, ***Static Leakage Current*** through the ***large nMOS*** transistors ***degrades the output level***.

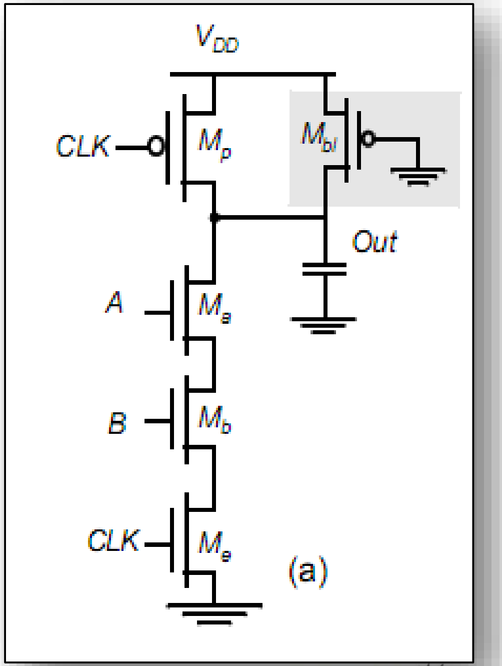
Input Glitch Example





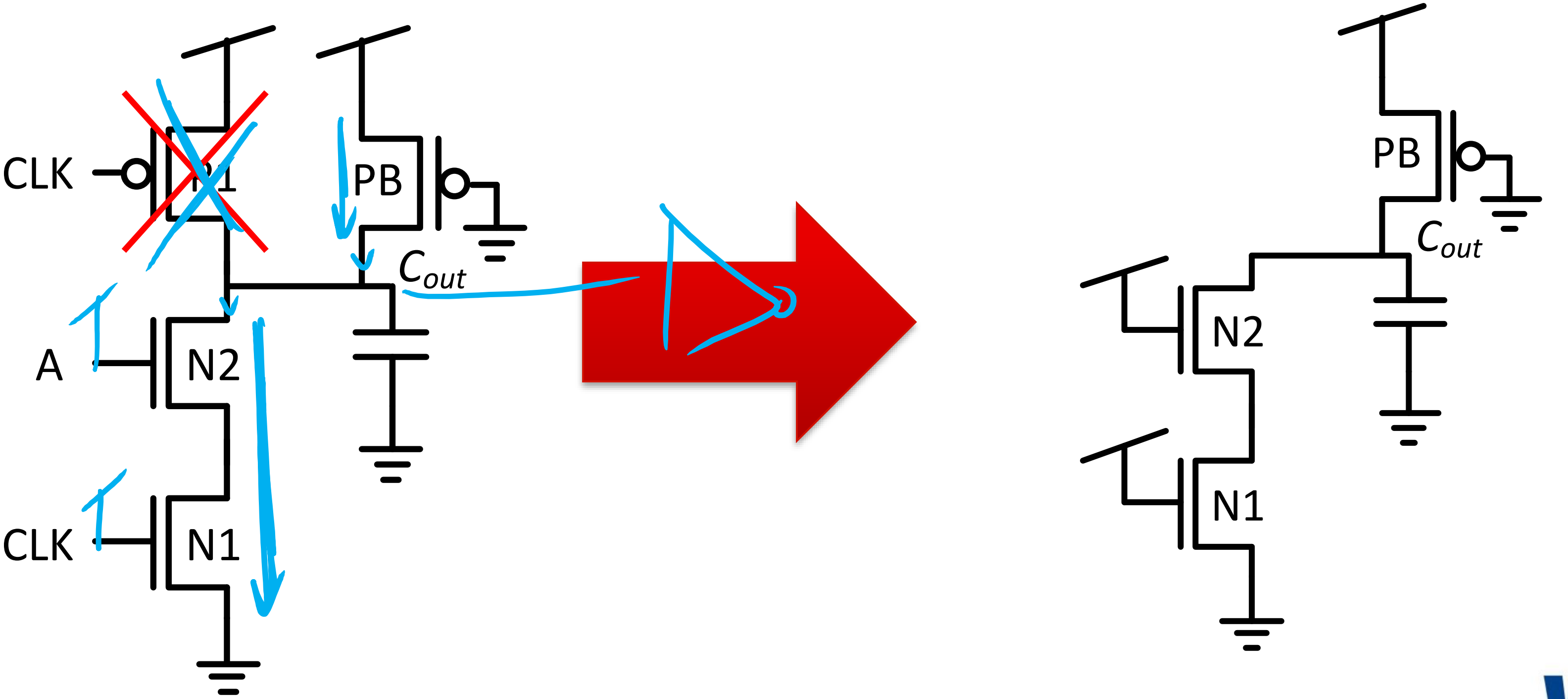
Solution: Bleed Devices

- To fight the ***Charge Loss*** in an output node, a ***Bleed Device*** can be added:
- A small ***Pseudo-nMOS*** style ***pMOS*** is the basic implementation
  - » This causes $V_{OLmin}>0$.
  - » Static Current Dissipation.



Example – Bleed Device

- We have to carefully size the bleed device to trade off level compensation and static current.
- To find the static current, assume the gate is in *evaluation* with *A='1'*.

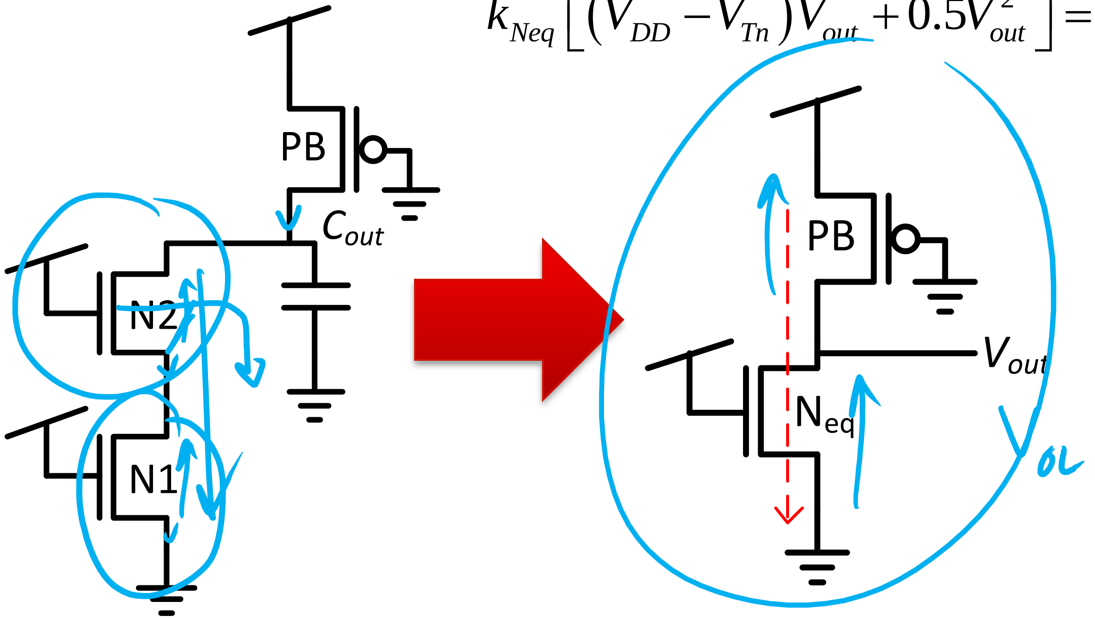


Example – Bleed Device

- We now find the minimum output level.
- We can replace the two nMOS transistors with a single nMOS with $L_{eq}=L_1+L_2$ (assuming they were sized with an equivalent W)

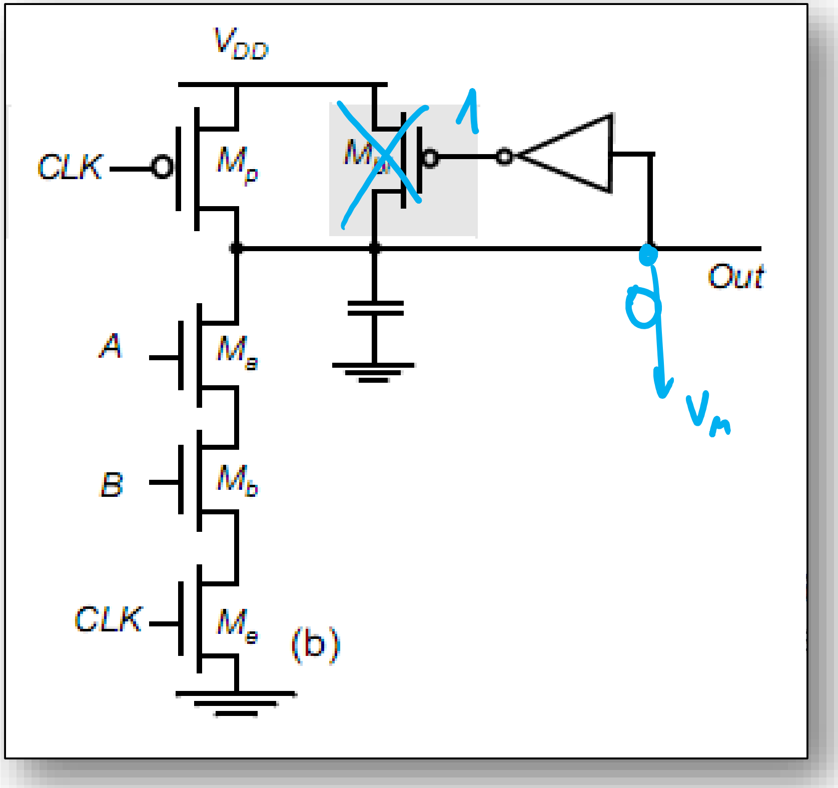
$$I_{Neq}(lin)=I_{PB}(vel.sat)$$

$$k_{Neq}\left[(V_{DD}-V_{Tn})V_{out}+0.5V_{out}^2\right]=k_{PB}\left[(V_{DD}-V_{Tp})V_{Dsat,p}+0.5V_{Dsat,p}^2\right]$$



Feedback Bleed Device

- ❑ A better way to attach a bleed device is using feedback.
- ❑ In this way the bleed device is cut off when V_{out}='0'
- ❑ We get:
  - » Rail to Rail Swing
  - » No Static Current
  - » No glitching problem
- ❑ But:
  - » We still have to make sure the pull down network is strong enough to flip the inverter.
  - » We need an extra 3 transistors*



*In a few minutes, we'll see that this is almost "free"

10.3

10.1 Dynamic CMOS

10.2 Charge Loss

10.3 Charge Sharing

10.4 Domino Logic

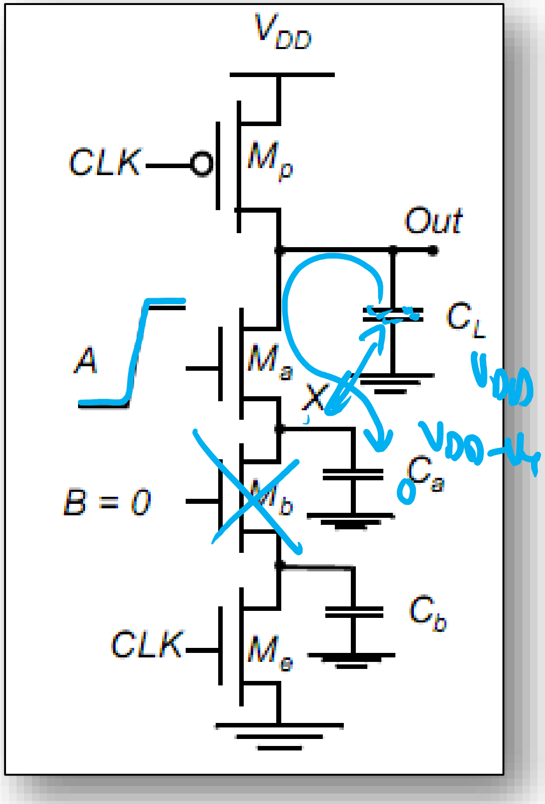
10.5 LE of Domino

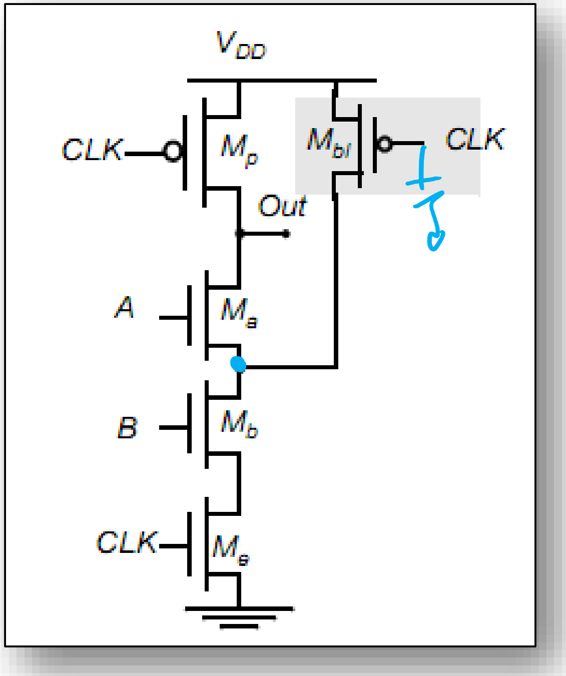
A secondary problem of Dynamic Gates is

CHARGE SHARING

Problem #2: Charge Sharing

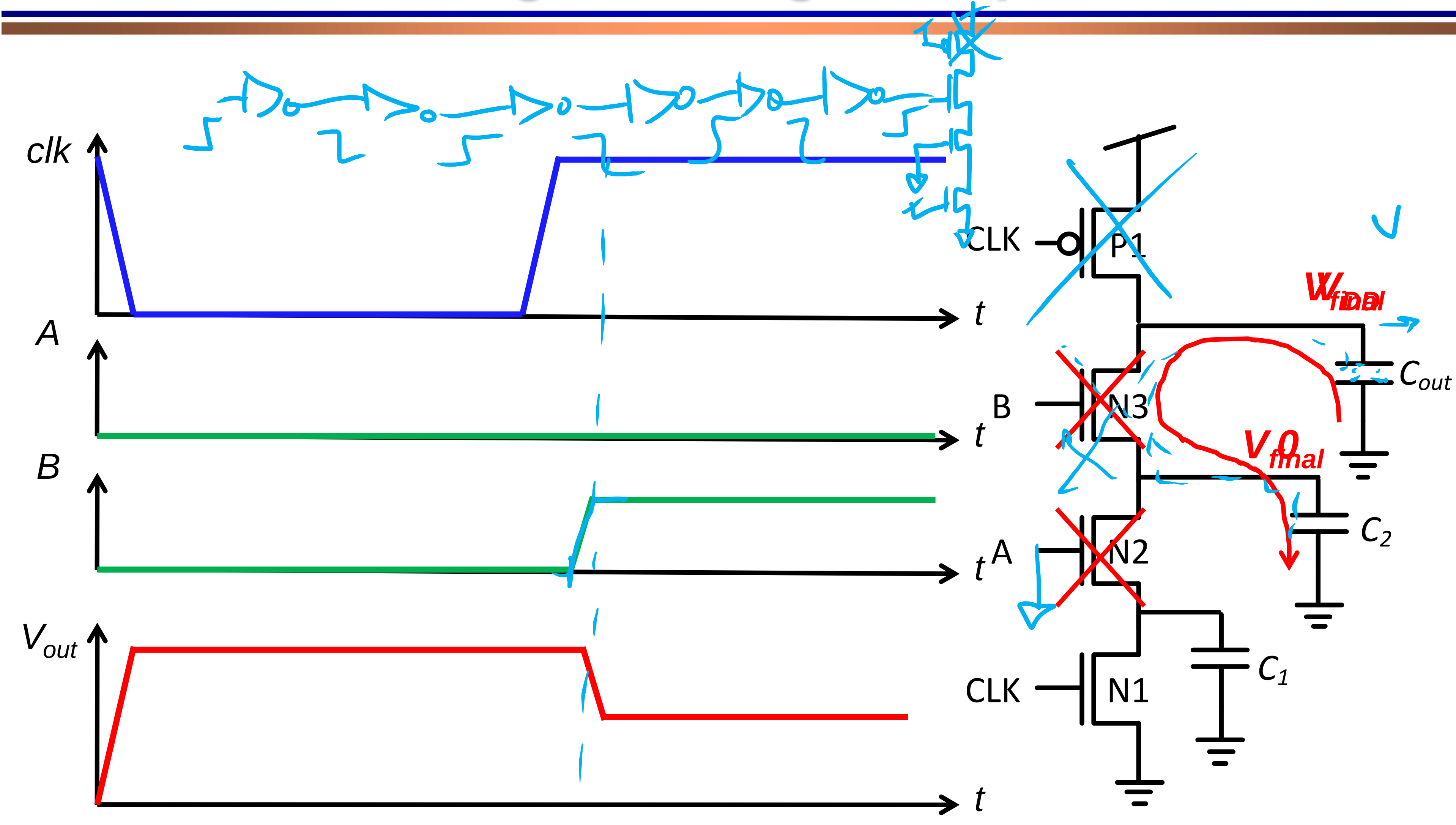
- ***Charge Sharing*** occurs when the ***PDN*** is ***closed***, but one or more ***stacked transistors*** next to the output are ***open***.
  - » The charge is shared between the ***output capacitance*** and the ***diffusion capacitance*** of the ***conducting transistor***.





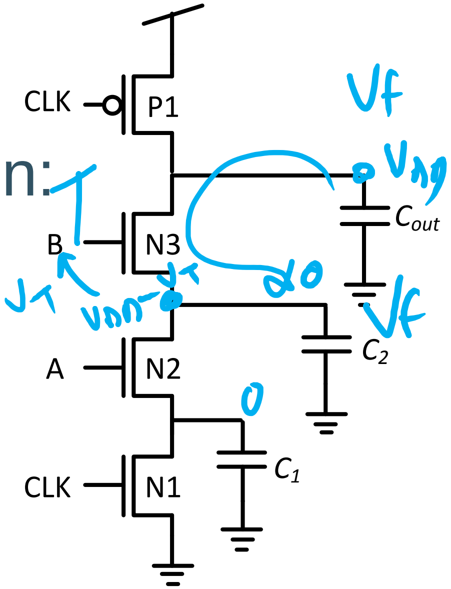
- One way to fix this is to ***Precharge*** these capacitances.

Charge Sharing Example



Charge Sharing Example

- How much charge is lost?
- We have to use Charge Conservation Equation:



$$Q_{initial} = Q_{final}$$

$$Q_{initial} = C_{out} \cdot V_{DD} \qquad Q_{final} = C_{out} \cdot V_{final} + C_2 \cdot V_{final}$$

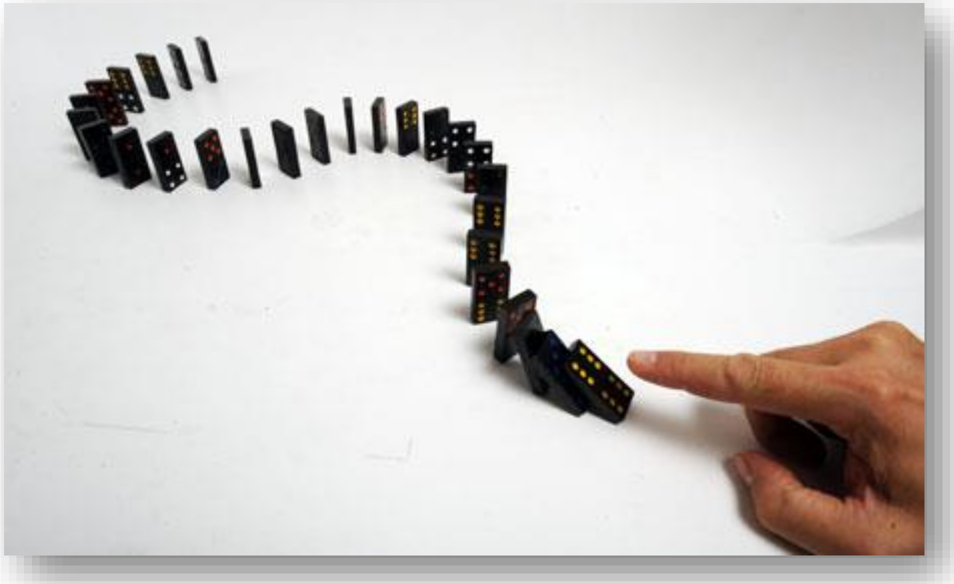
- Just don't forget to check if $V_{final} > V_{DD} - V_T$...

$C_{out} \cdot V_{final}$

$C_2 (V_{DD} - V_T)$

10.4

- 10.1 Dynamic CMOS
- 10.2 Charge Loss
- 10.3 Charge Sharing
- 10.4 Domino Logic
- 10.5 LE of Domino

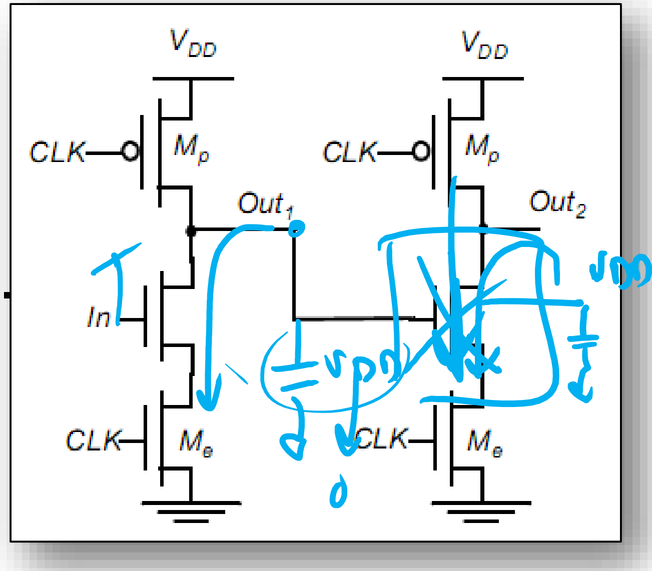


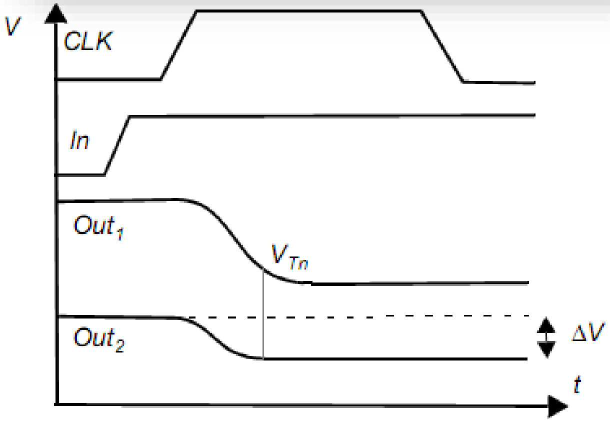
But the most interesting problem and solution comes with the need to hook up dynamic gates in a logic network, which brings us:

DOMINO LOGIC

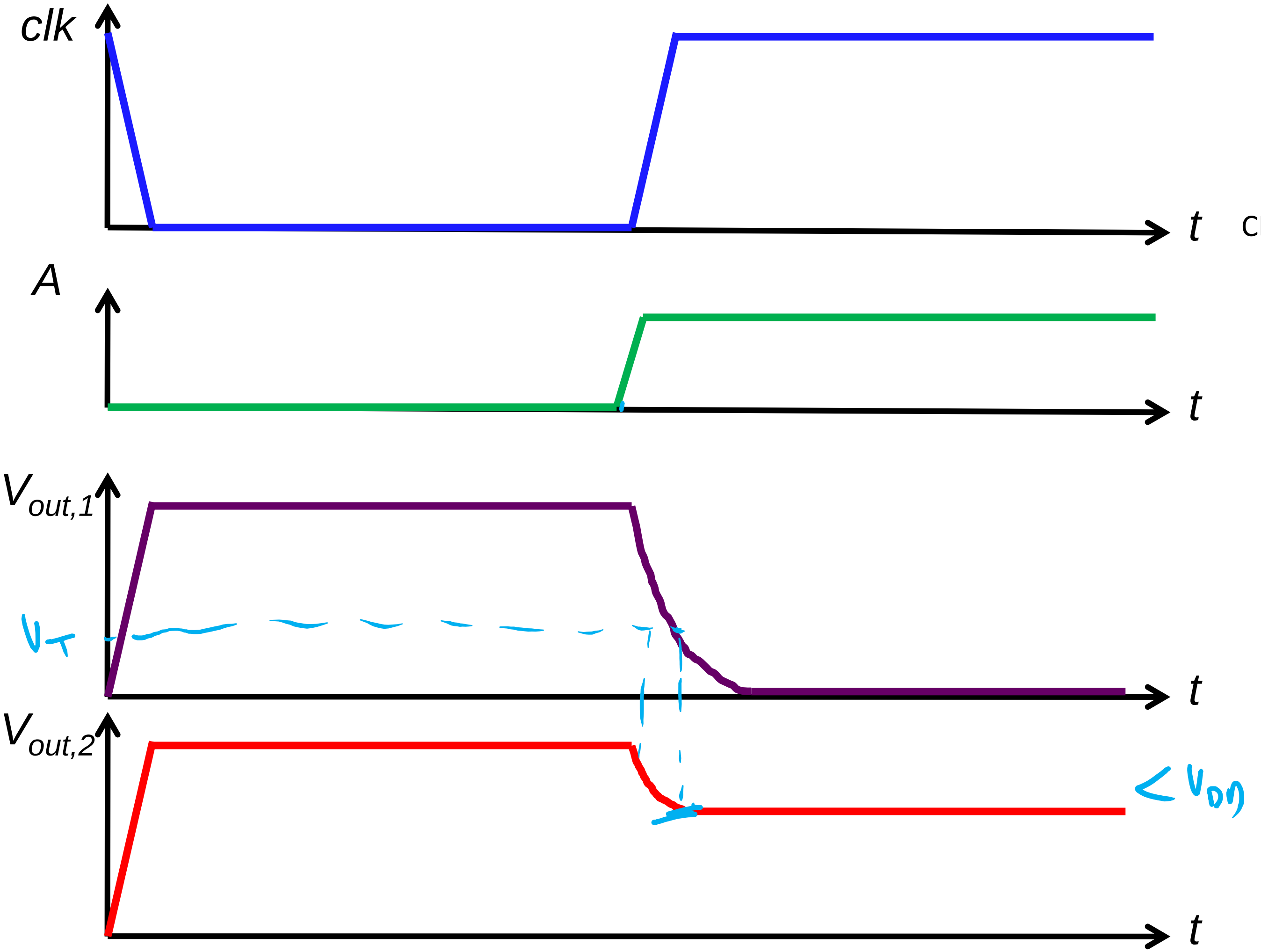
Problem #3: Cascading Dynamic Gates

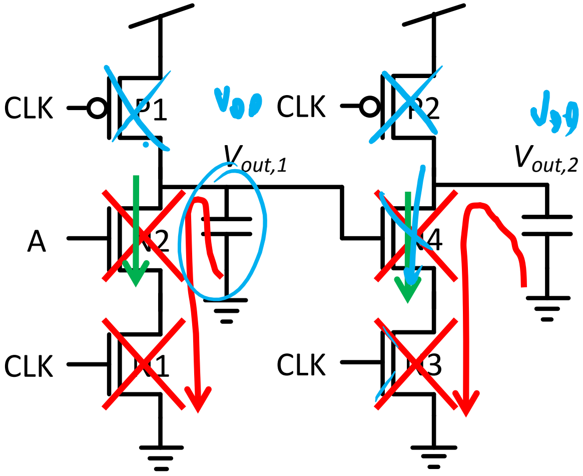
- The biggest drawback of ***Dynamic Logic*** is that ***Dynamic Gates cannot be Cascaded***:
  - During ***Precharge***, the output of the ***Driving Gate*** is charged to V_{DD}, ***turning on*** the ***PDN*** of the ***Cascaded Gate***.
  - During the ***Evaluation***, it takes time to ***discharge the output*** of the ***Driving Gate*** (considering it ***should be Low***).
  - During this time, the ***PDN*** of the ***Cascaded Gate*** is ***incorrectly conducting***, ***discharging*** its ***output***.





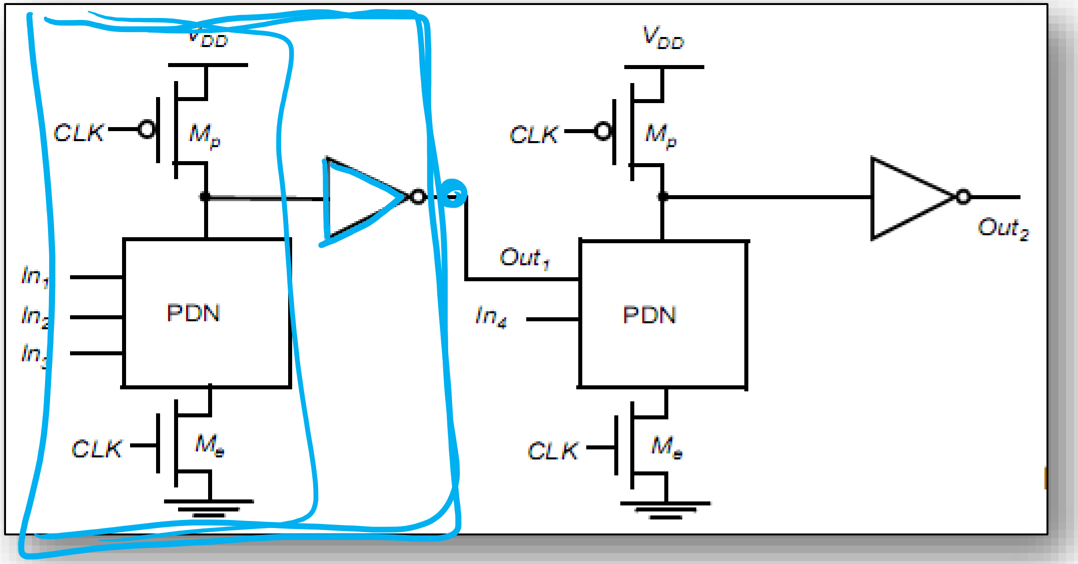
Cascading Gates Example





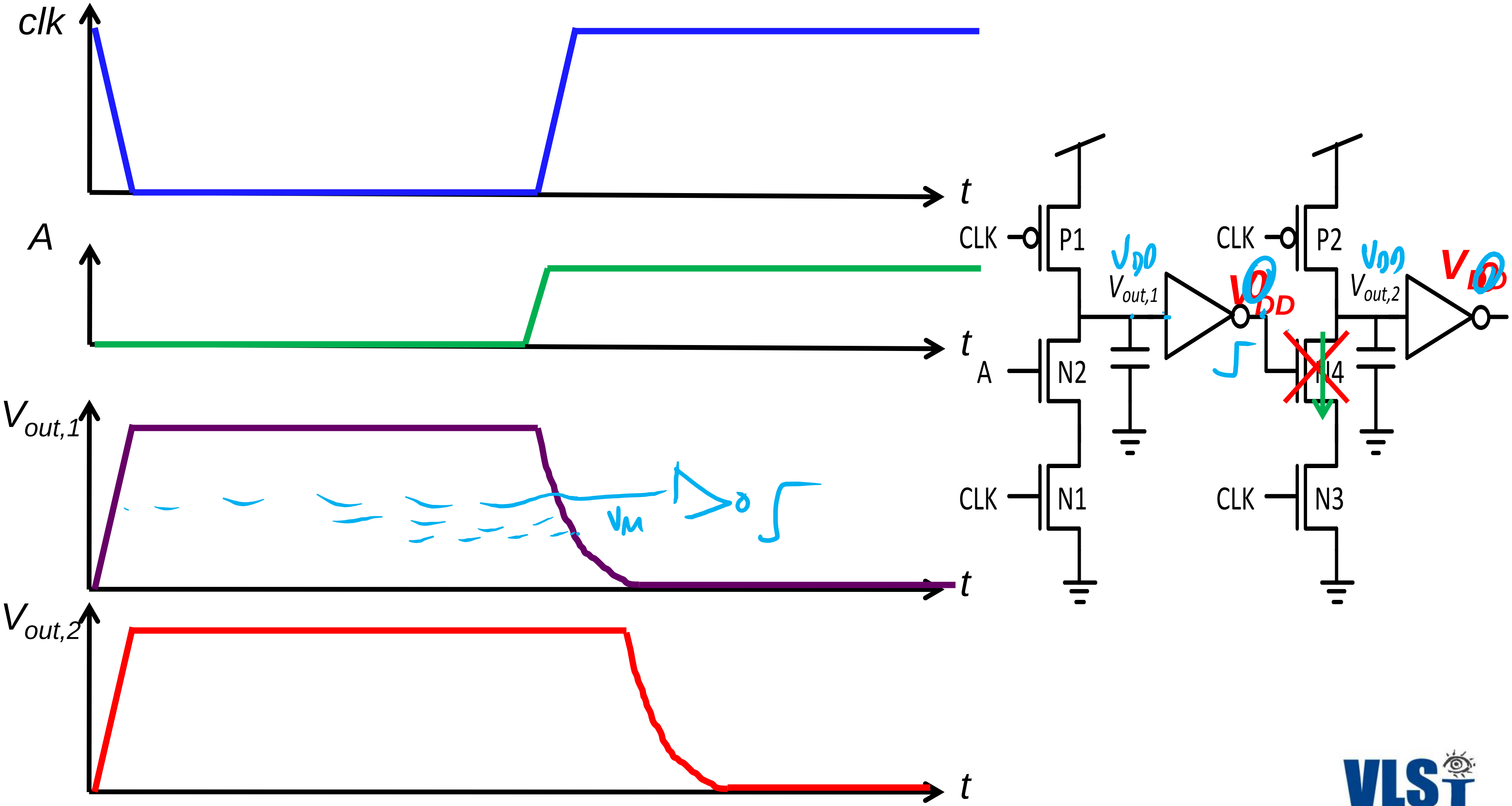
Solution: Domino Logic

- ❑ One solution to the cascading problem is using ***"Domino Logic"***.
- ❑ Each ***Dynamic Gate*** is connected to an ***Inverter***, causing the ***input of the next gate*** to be ***Low after Precharge***.
- ❑ This ***solves the cascading problem*** and ***buffering gates*** provides several other ***advantages***.



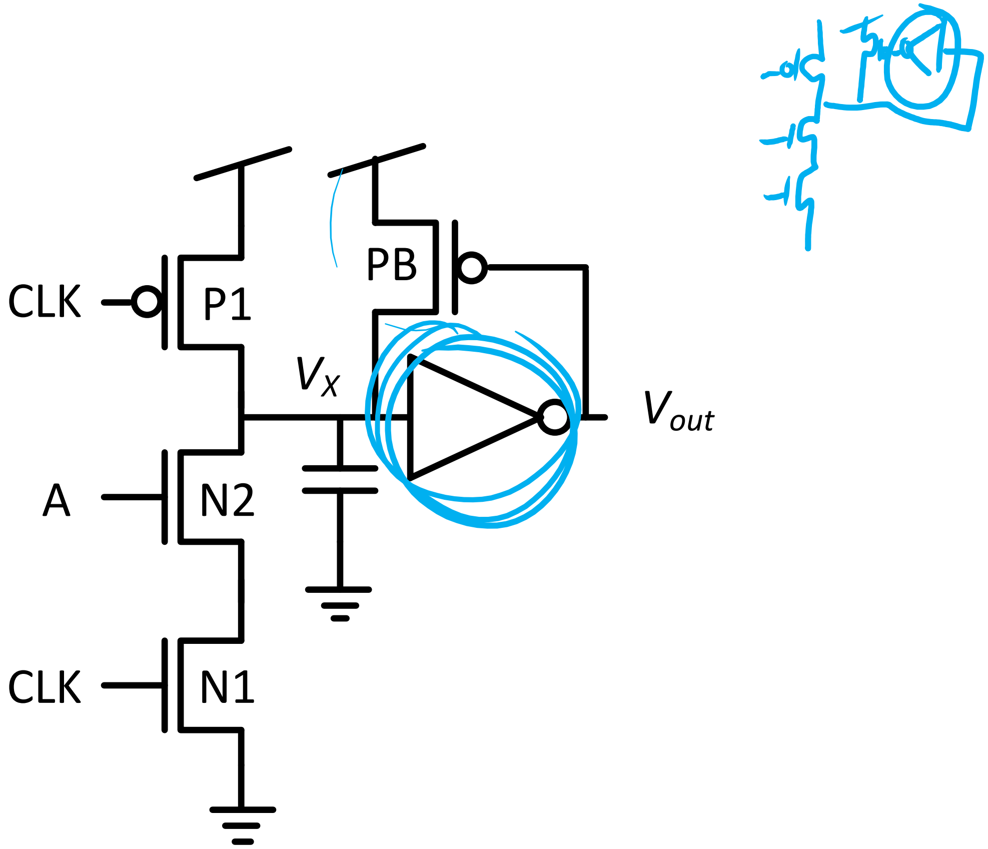
- ❑ One ***major disadvantage*** occurs, though.
- ❑ **Can you guess what it is**? (Hint: ***Universality***)

Domino Logic Example



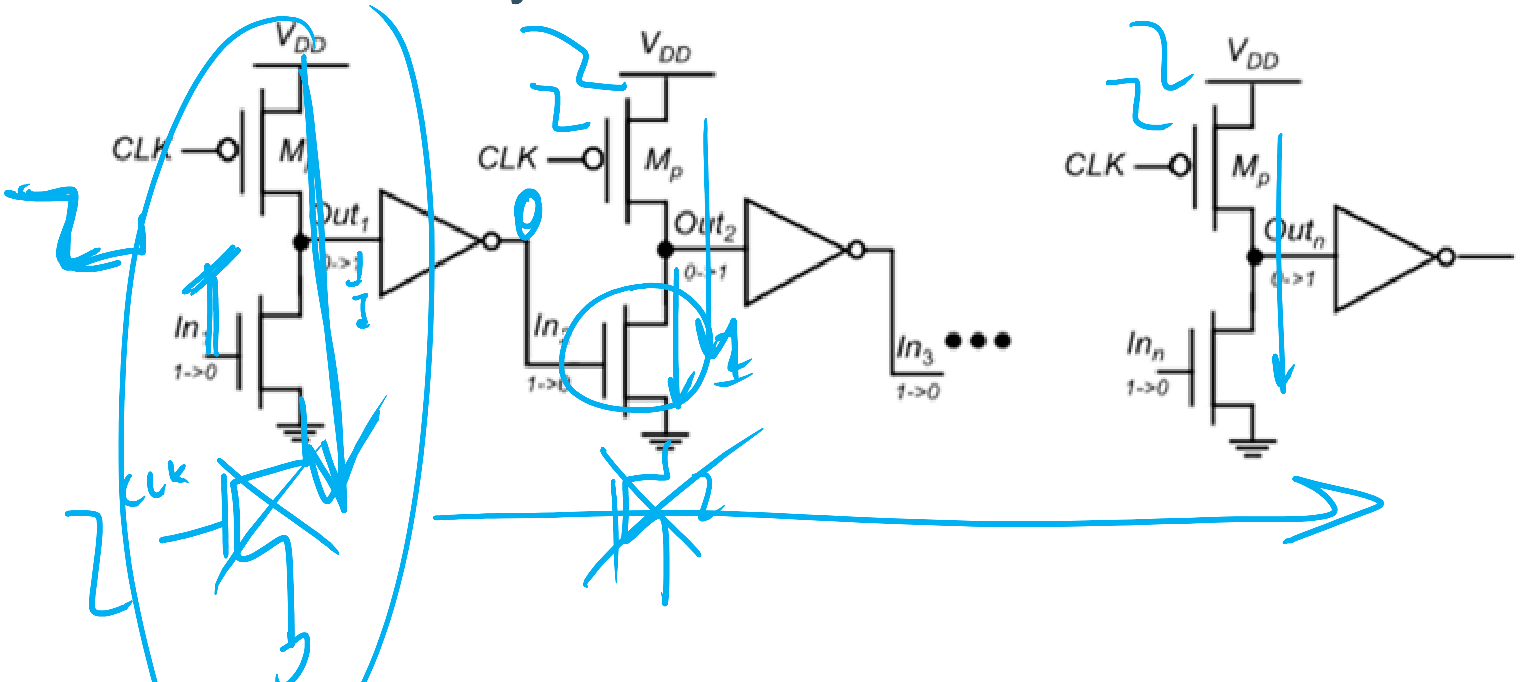
Solving the Rest of Our Problems…

- Now we can freely add our feedback bleed transistor…



Can we reduce the transistor count?

- Since the output of domino is always low during precharge, we don't actually need the evaluation transistor!

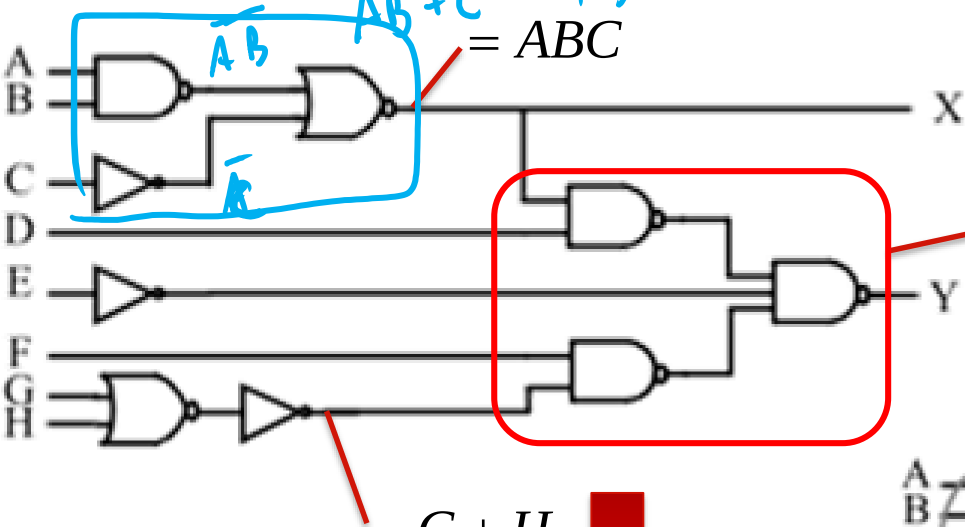


- However, we better be careful, because the *precharge* must now *propagate*, creating a constraint on the precharge time!

What about universality?

- Sometimes, logic restructuring works:

$\overline{\overline{AB}+\bar{C}} = ABC$

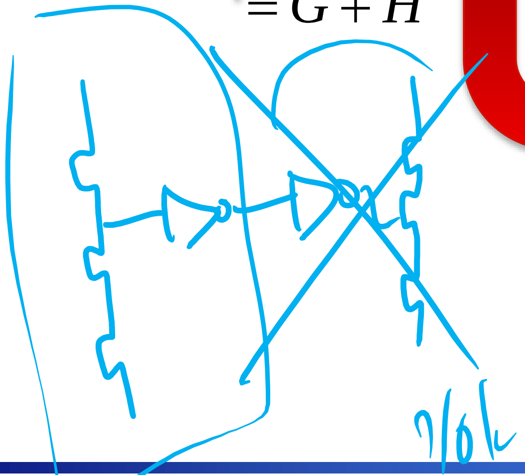


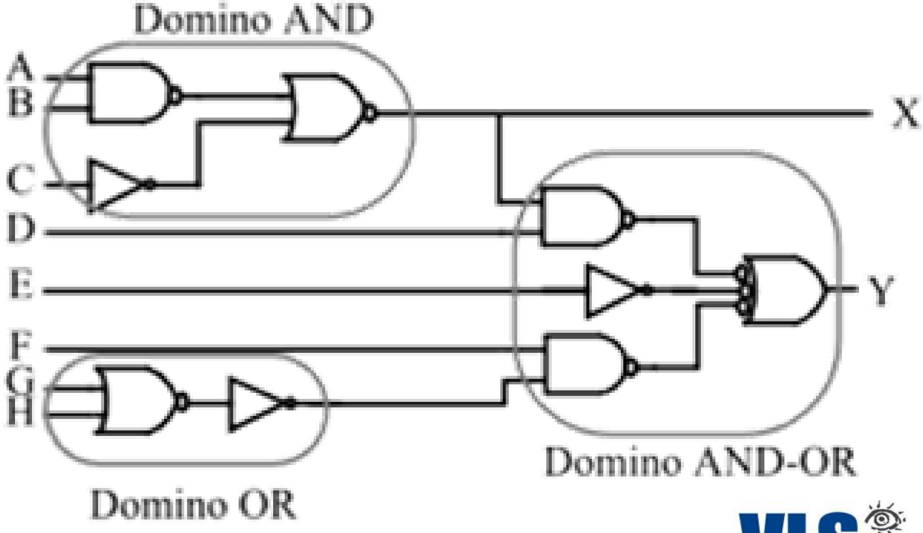
$= ABC$

$= G + H$

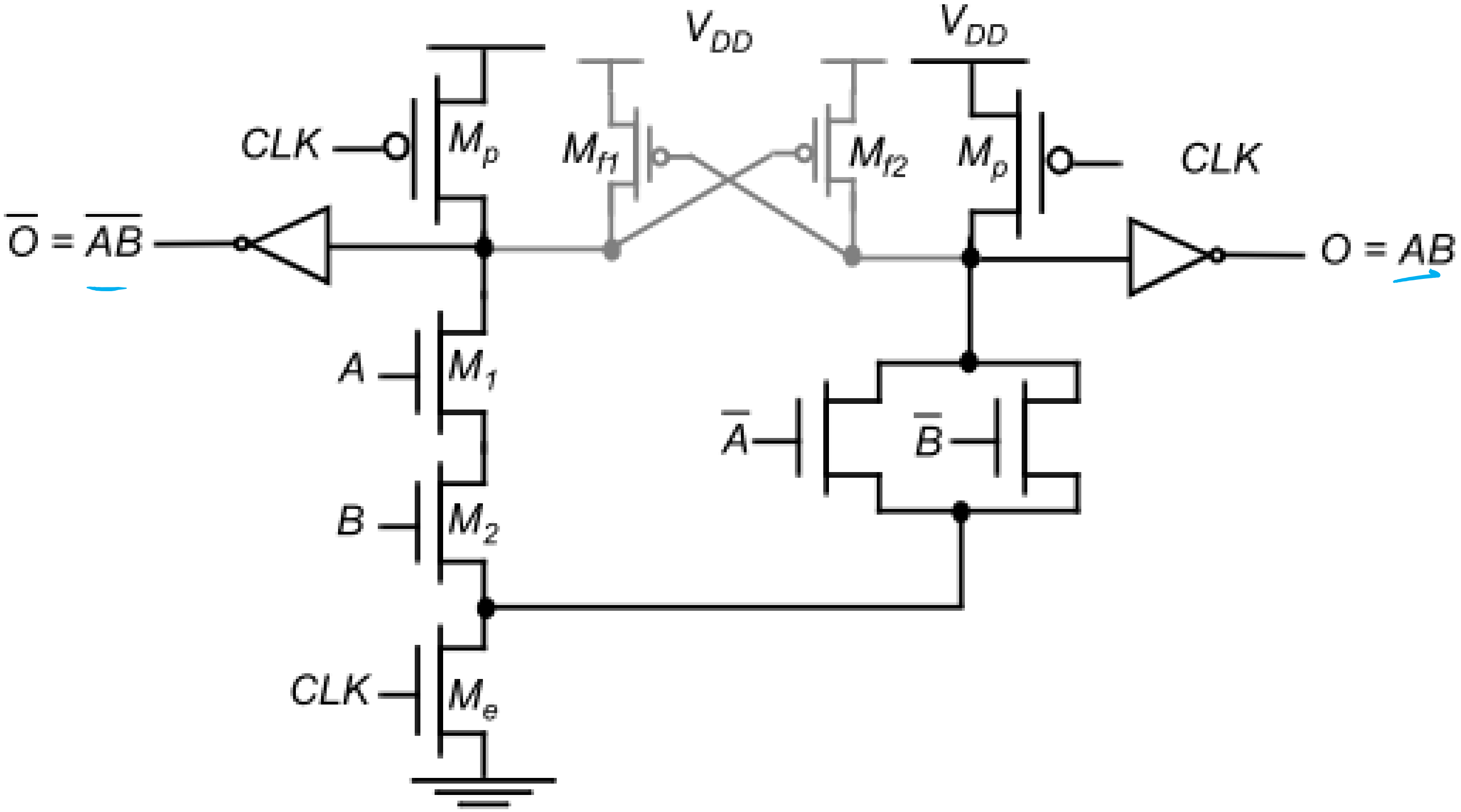
$$= \overline{\overline{I_1 I_2} + \overline{I_3} + \overline{I_4 I_5}} =$$

$$= I_1 I_2 + I_3 + I_4 I_5$$

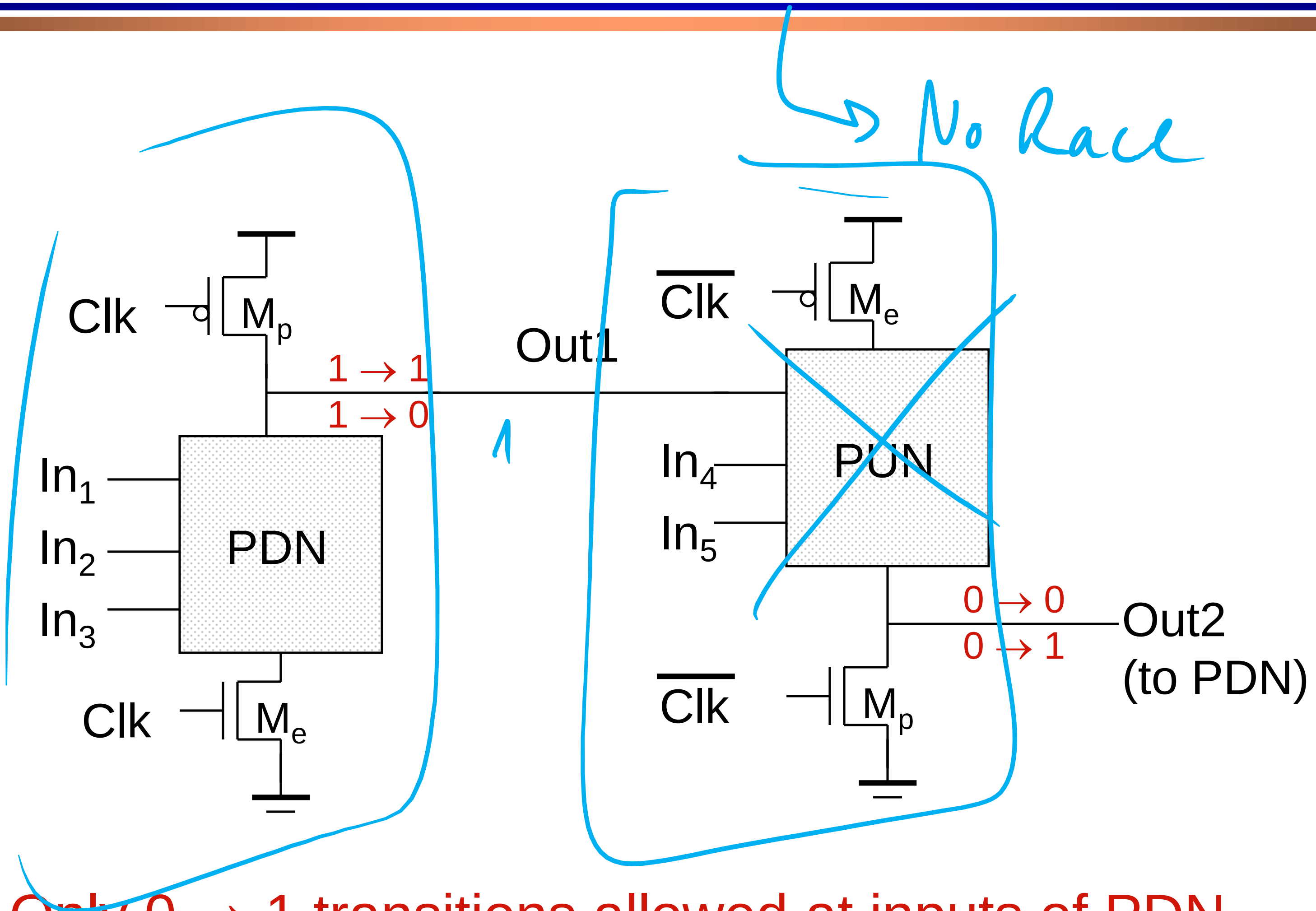




Dual Rail Domino



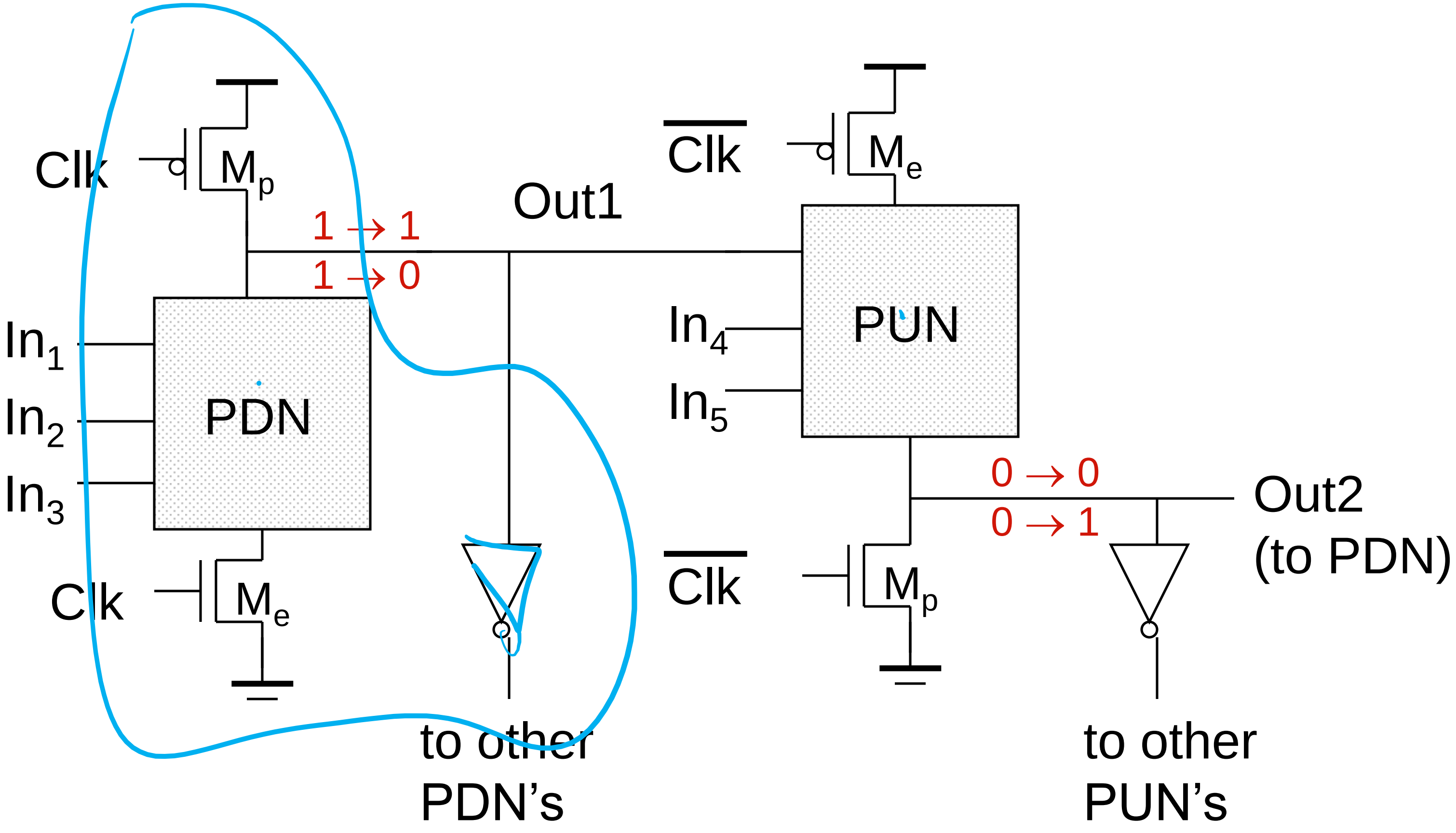
np-CMOS (NORA)



Only 0 → 1 transitions allowed at inputs of PDN

Only 1 → 0 transitions allowed at inputs of PUN

np-CMOS



Clk
M$_p$
1 → 1
1 → 0
Out1
In$_1$
In$_2$
In$_3$
PDN
Clk
M$_e$
to other PDN's
$\overline{\text{Clk}}$
M$_e$
In$_4$
In$_5$
PUN
0 → 0
0 → 1
Out2 (to PDN)
$\overline{\text{Clk}}$
M$_p$
to other PUN's

10.5

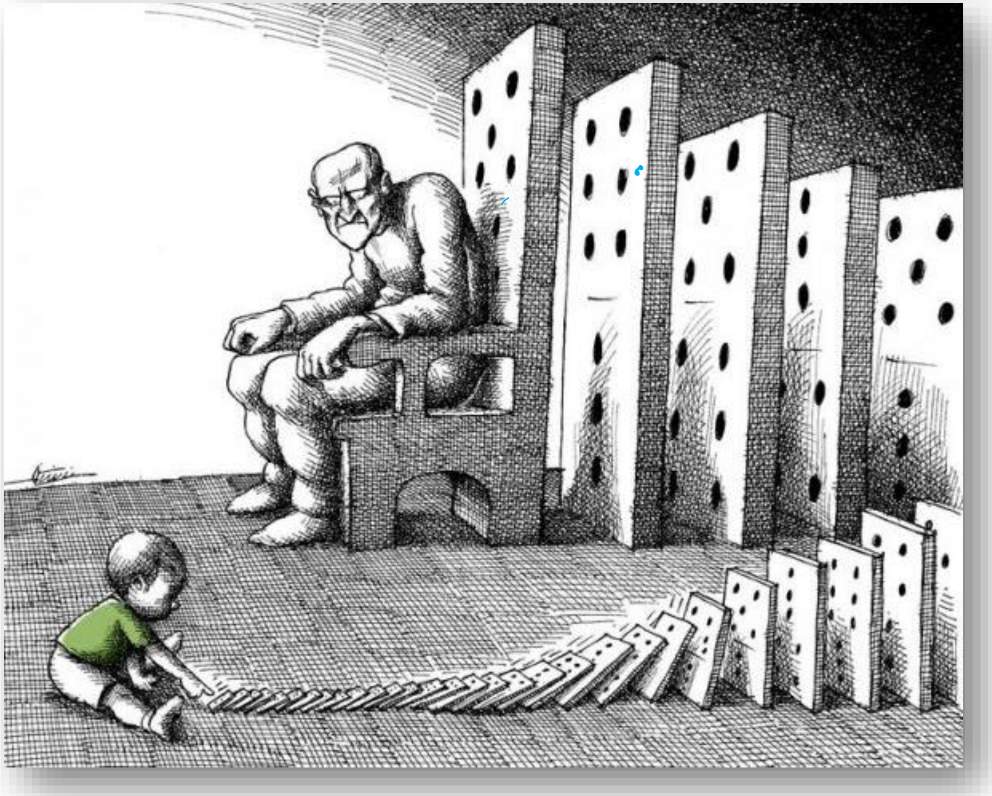
10.1 Dynamic CMOS

10.2 Charge Loss

10.3 Charge Sharing

10.4 Domino Logic

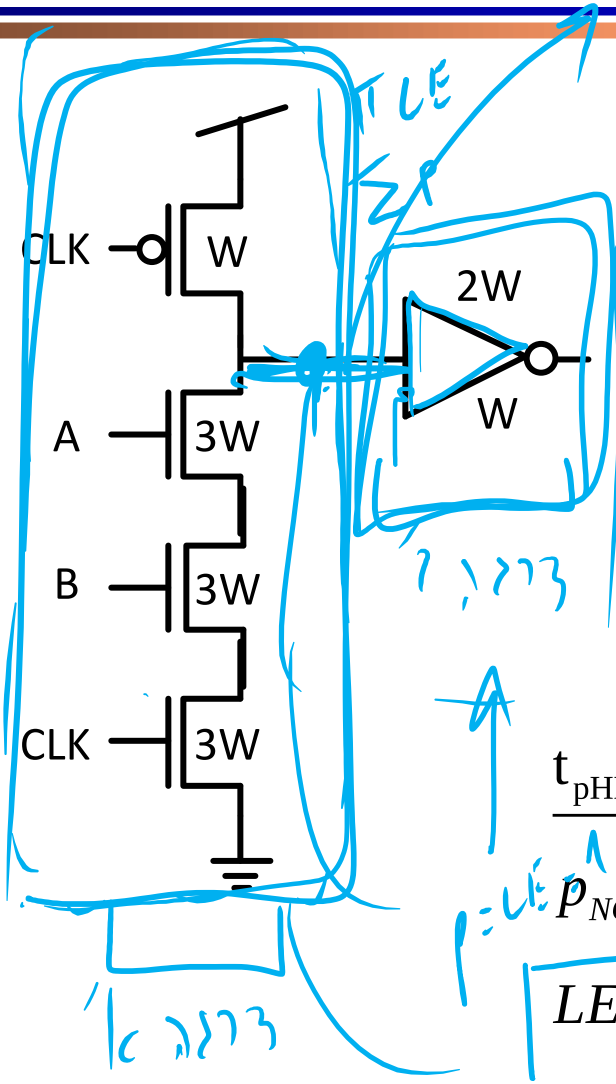
10.5 LE of Domino



Finally, let's see how domino gates behave in a logic network:

LOGICAL EFFORT OF DOMINO LOGIC

Domino Logic Logical Effort



t_{pHL} (NAND):

$$\boxed{\begin{aligned} R_{eq} &= R_{\min} \\ C_g\left(A,B\right) &= 3C_{g\min} \\ C_d &= 4C_{d\min} \end{aligned}}$$

$$p_{NAND} = {}^{4}\!/\!_{3}$$

$$LE_{NAND} = 1$$

$$\underline{\Pi LE_{NAND}} = \underline{1}\cdot 1 = 1$$

$$\underline{\Sigma p_{NAND}} = {}^{4}\!/\!_{3} + 1 = \underline{{}^{7}\!/\!_{3}}$$

t_{pHL} (NOR):

$$p_{NOR} = {}^{5}\!/\!_{3}$$

$$\boxed{LE_{NOR} = {}^{2}\!/\!_{3}}$$

$$\Sigma p_{NOR} = {}^{5}\!/\!_{3} + 1 = \boxed{{}^{8}\!/\!_{3}}$$

$$\Pi LE_{NOR} = {}^{2}\!/\!_{3}\cdot 1 = {}^{2}\!/\!_{3}$$

Domino Logic Logical Effort

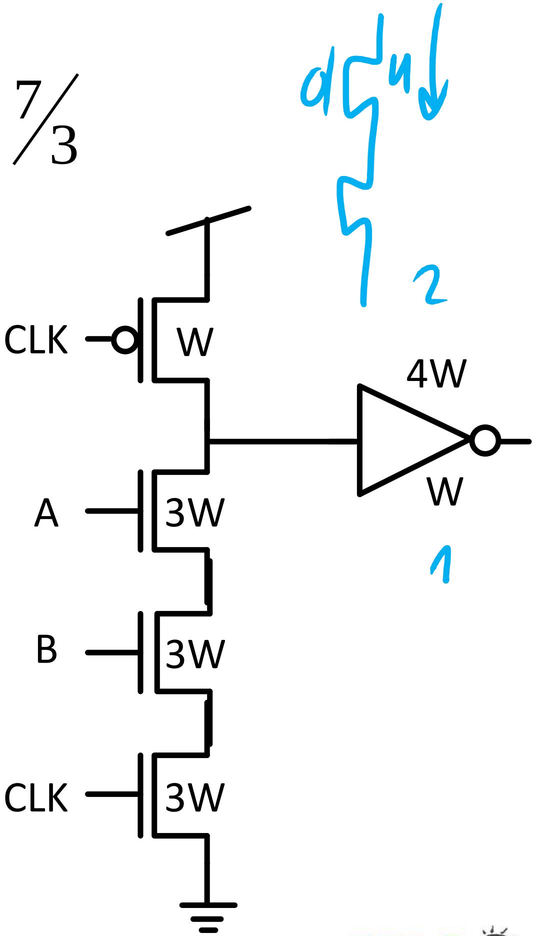
- ❑ We can improve this by using a “skewed inverter”

$\frac{1}{2} R_{min}$

- ❑ For a standard Domino NAND, we got:

$$p_{NAND} = 4/3 \quad LE_{NAND} = 1 \quad \Pi LE_{\cancel{N}AND} = 1 \quad \Sigma p_{\cancel{N}AND} = 7/3$$

- ❑ But if we enlarge the pMOS of the inverter, we get:



$$\underline{t_{pLH}}:$$

$$p_{INV*} = LE_{INV*} = 1/2 \cdot 5/3 = \boxed{5/6}$$

$$\Pi LE_{NAND*} = 1 \cdot 5/6 = \boxed{5/6} \quad \Sigma p_{NAND} = 4/3 + 5/6 = \boxed{13/6}$$

- ❑ The “Average” LE (~stage effort) is:

$$\sqrt{\Pi LE_{NAND*}} = \sqrt{5/6} = \underline{0.91}$$