CMOS Logic Gate Design

- To achieve correct operation of any integrated logic gate, both functional and timing constraints have to be satisfied.
- The following effects can result in incorrect functioning
  - Incorrect or insufficient power supplies, or power supply noise
  - Noise on gate inputs
  - Faulty transistors
  - Faulty connections to transistors
  - Incorrect ratios in ratioed logic
  - Charge sharing or incorrect clocking in dynamic gates
- Critical paths
  - slowest paths in a logic design
  - speed optimization is required
  - can be affected at four main levels
    1. architecture level
    2. RTL/logic gate level
    3. circuit level
    4. layout level
  - The most leverage is achieved by completing a good architecture, i.e. designing the overall function in the most efficient manner at the highest level possible. (No amount of skillful logic design can overcome a poor architecture)
  - Once the logic level has been decided, the circuit level of design can be used to optimize a critical speed path. This may be done by sizing transistors or using other styles of CMOS logic
  - Finally, one can affect the speed of a set of logic by rearranging the physical layout.

Fan-In and Fan-Out

- Fan-in affects gate speed

  e.g. If two identical transistors are connected in series, the rise (or fall)time will be approximately double that for a single transistor with the same capacitive load

- Fan-out

  The capacitive load of a gate is usually determined by the number of its fan-out

- For an m-input NAND gate

  1. Equal-sized gate (The gate sizes of p and n-transistors are the same)

     — worst-case rise time t_r(only one p-device is on)

$$t_r = \frac{R_p}{n}(mnC_d + C_r + C_g)$$

R_p = the effective resistance of p-device in a minimum-sized inverter
n = width multiplier for p-device in this gate
k = the fan-out(in units of minimum-sized inverters)
m = fan-in of gate
C_g = gate capacitance of a minimum-sized inverter
C_d = source/drain capacitance of a minimum-sized inverter
C_r = routing capacitance

— This can be reformulated as

$$t_r = \frac{R_p}{n}(mnrC_g + q(k)C_g + kC_g)$$

$$= \frac{R_pC_g}{n}(mnr + q(k) + k)$$

$$= R_pC_g mr + \frac{R_pC_g}{n}q(k) + \frac{R_pC_g}{n}k$$

where

$t_r = C_d/ C_g$, the ratio of the intrinsic drain capacitance of an inverter to the gate capacitance,q(k) = a function of the fan-out representing the routing capacitance as a multiplier times the gate capacitance.

※ Larger n => 1. The latter two terms become smaller
2. Higher cost (I.e. larger size)
3. Heavier capacitive load for the anterior stage

The above equation is of the form

$$t_r = t_{\text{internal-r}} + k * t_{\text{output-r}}$$

where $t_{\text{internal-r}} = R_p C_g\, mr$

$$t_{\text{output-r}} = \frac{R_p C_g}{n} \left(1 + \frac{q(k)}{k}\right)$$

Similarly, the fall time, t_f, is approximated by

$$t_f = m \frac{R_n}{n} (mnrC_g + q(k)C_g + kC_g)$$

$$= R_n C_g m^2 r + mk \frac{R_n C_g}{n} \left(1 + \frac{q(k)}{k}\right)$$

$$= t_{\text{internal-f}} + k * t_{\text{output-f}}$$

where R_n=the effective resistance of
a minimum-sized n-device

2. Equal-delay method (where rise and fall times are equalized)

$$t_r = t_f$$

$$\Longrightarrow \frac{R_p}{n}(mnrC_g + q(k)C_g + kC_g)$$

$$= m \frac{R_n}{n} (mnrC_g + q(k)C_g + kC_g)$$

$$\Longrightarrow R_p = mR_n$$

$$\Longrightarrow [(\frac{W}{L})_p \mu_p]^{-1} = m [(\frac{W}{L})_n \mu_n]^{-1}$$

$$\Longrightarrow W_p = \frac{W_n}{m} \left(\frac{\mu_n}{\mu_p}\right)$$

- For an m-input NOR gate

—e.g. equal-sized method

$$t_r = m\frac{R_p}{n}(mnrC_g + q(k)C_g + kC_g)$$

$$t_f = \frac{R_n}{n}(mnrC_g + q(k)C_g + kC_g) \cdots\cdots$$ only one n-device is on

Driving Large Fan-out

- Certain logic circuits can have signals that have large capacitive loads due to large fan-out

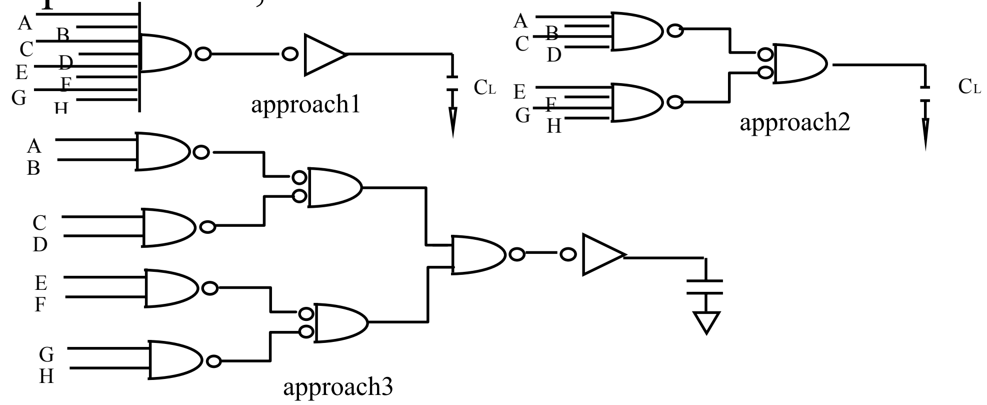
  clocks and reset signals are common examples

- Example

  As an example of a simple logic decision consider the implement of an 8-input AND gate driving a 1***p***F load (for instance, a row decoder in a RAM or ROM), we may use the following(Fig. 5.3):

  - Approach 1--An 8-input NAND and an inverter
  - Approach 2--Two 4-input NANDs and a 2-input NOR
  - Approach 3--Four 2-input NANDs, two 2-input NORs

    a 2-input NAND, and an inverter



APPROACH	DELAY(ns) STAGE 1	DELAY(ns) STAGE 2	DELAY(ns) STAGE 3	DELAY(ns) STAGE 4	Total Delay(ns) (SPICE)
1	2.82 ND8 falling	3.37 INV rising			6.2 (6.5)
2	0.88 ND4 falling	4.36 NR2 rising			5.24 (5.26)
3	0.31 ND2 falling	0.4 NR2 rising	0.31 ND2 falling	2.17 INV rising	3.19 (3.46)

- The shortest delay is achieved by approach 3
- Large stage ratio may be required for driving a large output capacitive load(for all 3 approaches)

Guidelines for High-Speed Logic Design

- Use NAND structures where possible.
- Place inverters (or at worst, small fan-in NAND gates) at high fan-out nodes, if possible.
- Avoid the use of NOR structures in high-speed circuits, especially with a fan-in greater than four and where the fan-out is large.
- Use a fan-out below 5-10.
- Use minimum-sized gates on high fan-out nodes to minimize the load presented to the driving gate.
- Keep rising and falling edges sharp.
- When designing with power or area as a constraint , remember that large fan-in complementary gates will always work given enough time.

CMOS Logic Structures

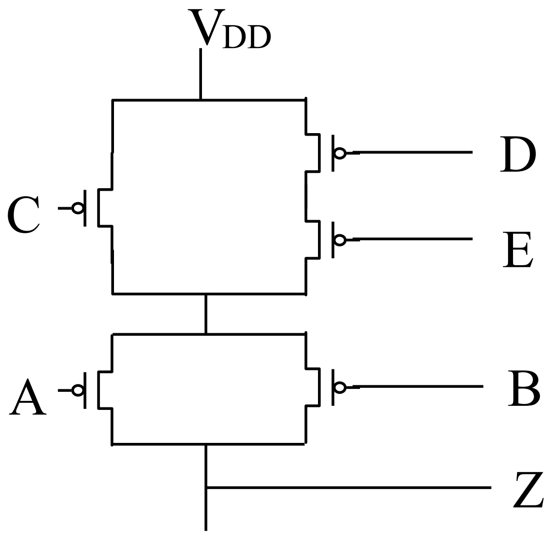
- CMOS complementary Logic

  e.g. $F = A\bullet B+C\bullet(D+E)$

— TRANSMIT " 1's "

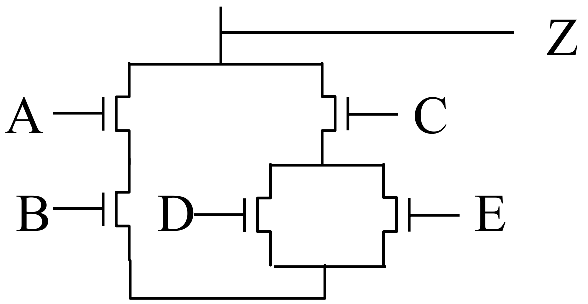
$$\begin{tikzcd} & F = (\overline{A\bullet B})\bullet(\overline{C\bullet(D+E)}) \arrow[dl] \arrow[dr] & & \\ \overline{A}+\overline{B} & & \overline{C}+(\overline{D+E}) \arrow[dr] & \\ & & & \overline{D}\bullet\overline{E} \end{tikzcd}$$

$$F = (\overline{A}+\overline{B})\bullet(\overline{C}+(\overline{D}\bullet\overline{E}))$$

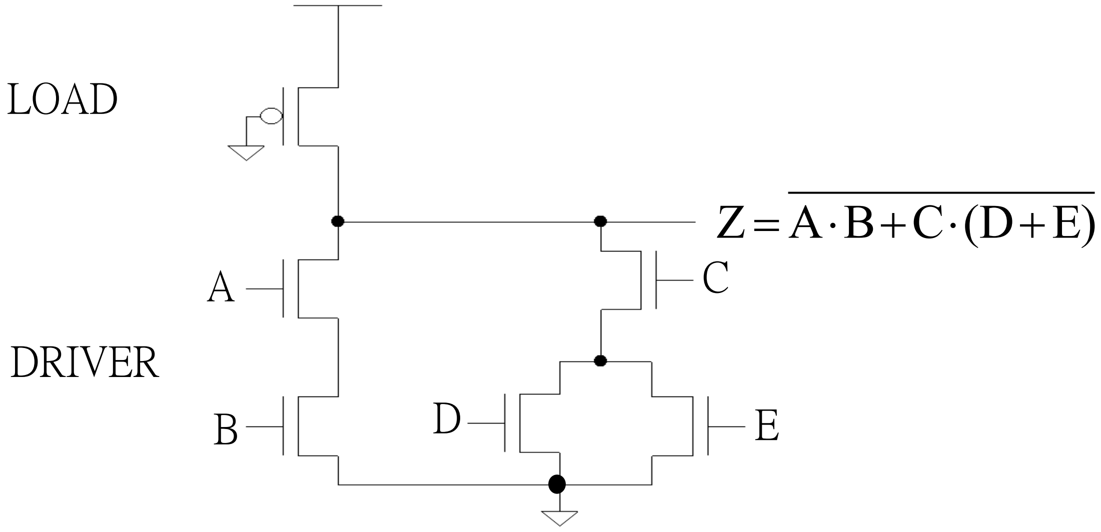


— TRANSMIT " 0's " :

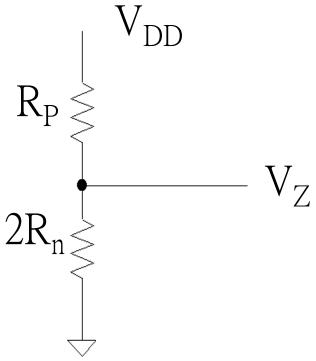
$\overline{F} = A\bullet B+C\bullet(D+E)$



Pseudo-NMOS Logic



- PMOS transistor is always ON
  - — If NMOS network is OFF
    Z is pulled up ; $Z = V_{DD}$
  - — If NMOS network is ON
    Z is pulled down ; $Z \rightarrow V_{SS}$



$$V_Z = V_{DD}\frac{2R_n}{2R_n + R_P} \qquad R\ \alpha\ \beta^{-1}$$

$$\frac{V_Z}{V_{DD}} = \frac{\frac{2}{\beta_N}}{\frac{2}{\beta_N}+\frac{1}{\beta_P}} = \frac{\frac{2}{\beta_N}}{\frac{2\beta_P+\beta_N}{\beta_N\beta_P}}$$

$$= \frac{2}{\beta_N}\frac{\beta_N\beta_P}{2\beta_P+\beta_N} = \frac{2\beta_P}{2\beta_P+\beta_N}$$

For $V_Z \rightarrow V_{SS}$, $\beta_n >> \beta_p$

$\Rightarrow$ ratioed transistors $\frac{w_n}{w_p} >> 1$

$\frac{\beta_n}{\beta_p} \approx 6$ (refer to page 75 of textbook)

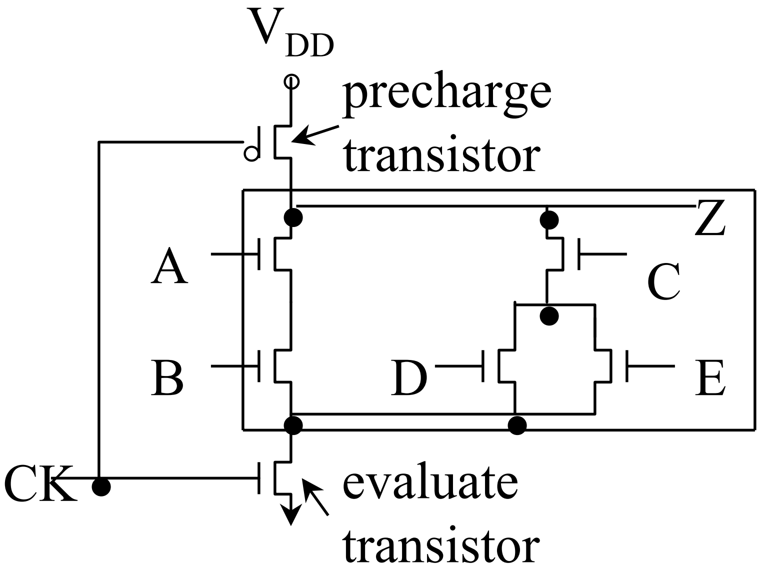
- Static P_D when pull-dpwn network is on
- Capacitance load for any input =C_g
- If min. size driver transistors are used, L_p>W_p to give proper β ratio ==> slower t_r
- Emulates NMOS circuits ==> use of existing base
- Density better than full CMOS (n+1 transistors VS. 2n)

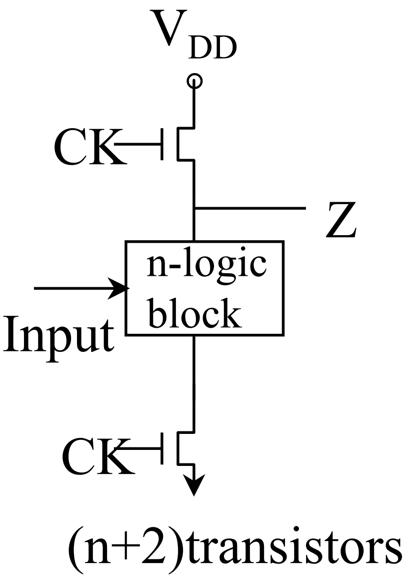
Comparsions for N-input CkT

	Full CMOS	pseudo NMOS
C_{IN}	$2C_g$	C_g
#T's	2N	N+1
t_f	t_f	t_f
t_r	t_r	$>t_r^*$

* depends on CkT

Dynamic CMOS Logic





WHEN ∅ = 0 : Z=HIGH, precharge

WHEN ∅ = 1 : stage of n-network is evaluated

if n-network is ON, Z=LOW

if n-network is OFF , Z remains HIGH

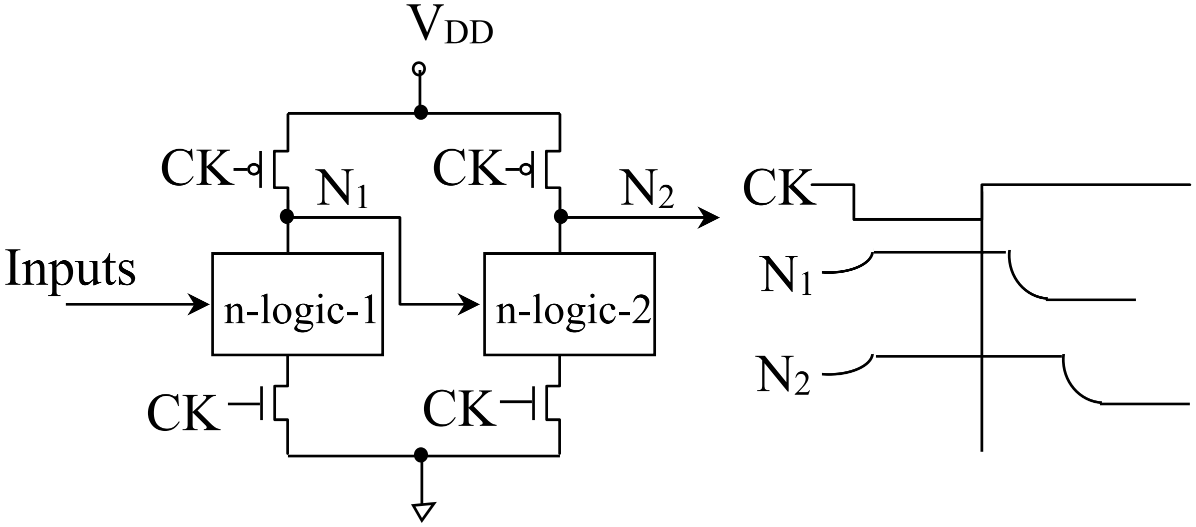
=> $Z = \overline{AB + C(D+E)}$

- For any input , $C_{in} = 1C_g$
- Pull-up time ~ same as pseudo N-MOS
- Pull-down time ↑ due to evaluate time

● Restriction on inputs to charge only during precharge due to charge redistribution

Simple, Single Phase Dynamic CMOS Gates

- Cannot be cascaded



－ When the gates are precharged, N_1 and N_2 are charged to V_{DD}

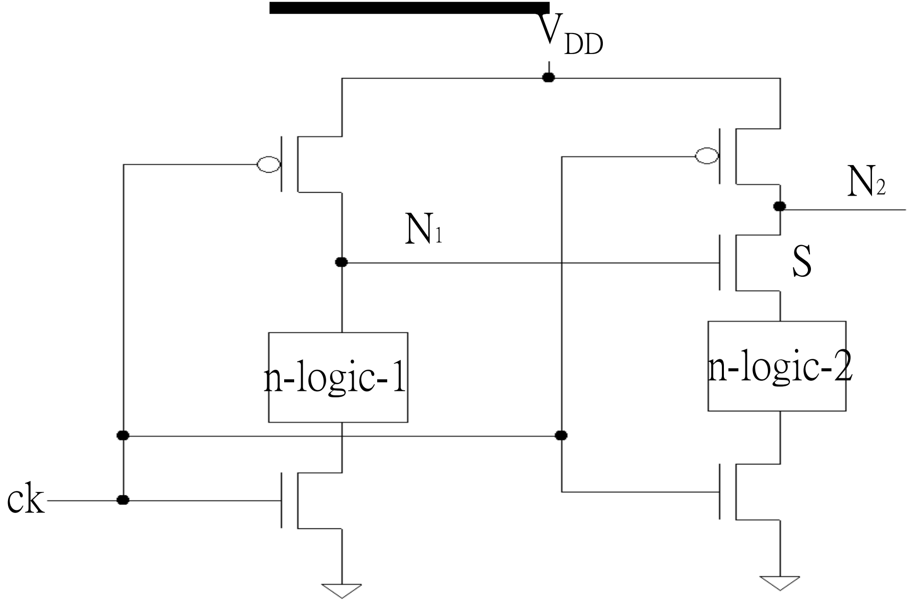
－ During evaluate, n-logic-1 will conditionally discharge with some delay due to t_p

－ N_1 must be fully evaluated before N_2 is evaluated; othermise the output N_2 will be in error

This cannot be assured with single-phase clocks

－ These problems can be overcome by using multi-phase clocks and a sample and hold circuit to isolate cascaded stages and control evaluate timing

Example



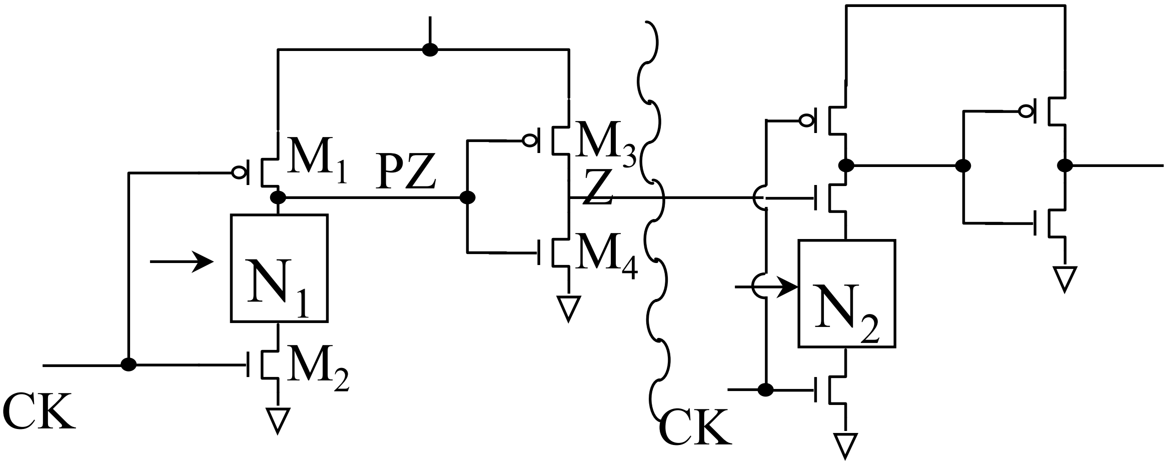
- During precharge, N_1 is automatically high and S is automatically ON

- Suppose that n-logic-1 is ON and n-logic-2 is ON
  —Immediately after precharge ends, N_2 will tend to be pulled down until N_1 is pulled down by n-logic-1 thereby turning S OFF

- To make cascaded stages (single phase clock) ok, S should be OFF until it is conditionally turned ON (by N_1)
  —Can do this by inserting an inverter between N_1 and S => domino logic

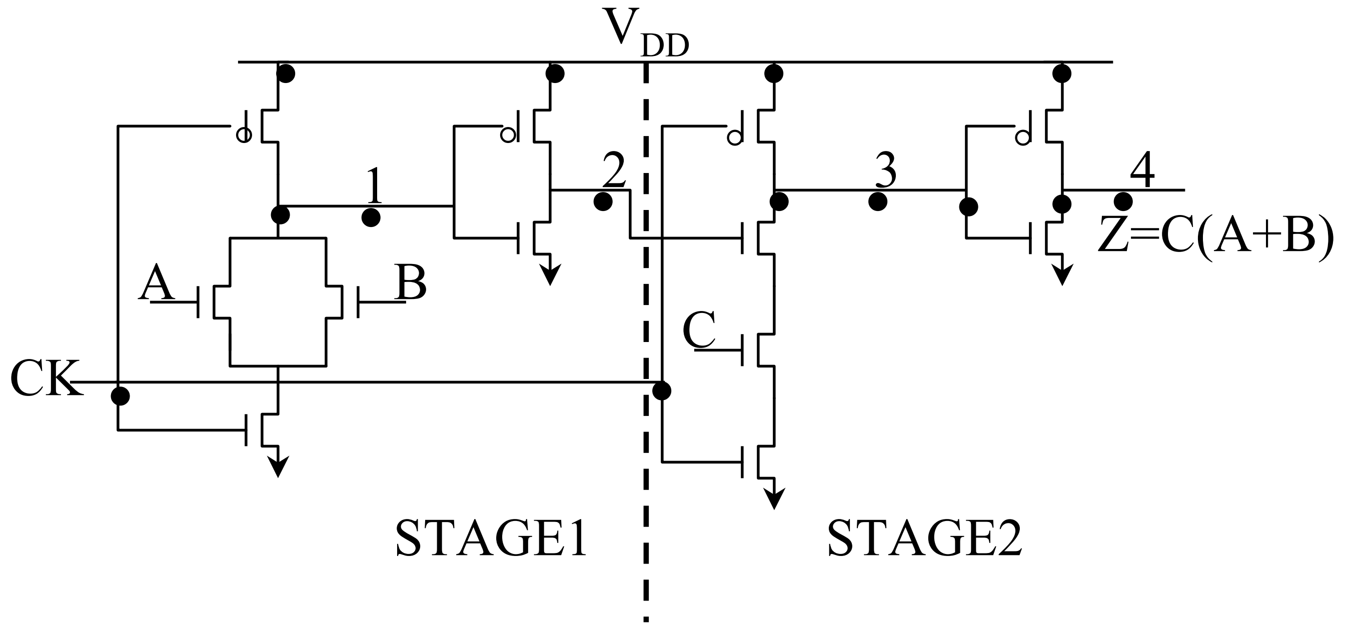
CMOS DOMINO LOGIC

- lncorporate a static (non-clocked) buffer(inverter) into each dynamic logic gate



CK=0:	M_1 on M_2 off	M_3 off M_4 on	PZ precharged HIGH Z precharged LOW
CK=1:	M_1 off M_2 on	? ?	evaluate Z=PZ ==> causes next stage to evaluate

Example



- Precharge: CK ⟶ LOW
  - A,B,C, don,t care

NODE 1 ⟶ HIGH

2 ⟶ LOW

3 ⟶ HIGH

4 ⟶ LOW

↓ PROPOGATION “DOMINOS”

- Evaluate: CK ⟶ HIGH
  - Assume A=B=1 and C=0

1 ⟶ LOW

2 ⟶ HIGH

3 ⟶ HIGH (C=0)

4 ⟶ LOW

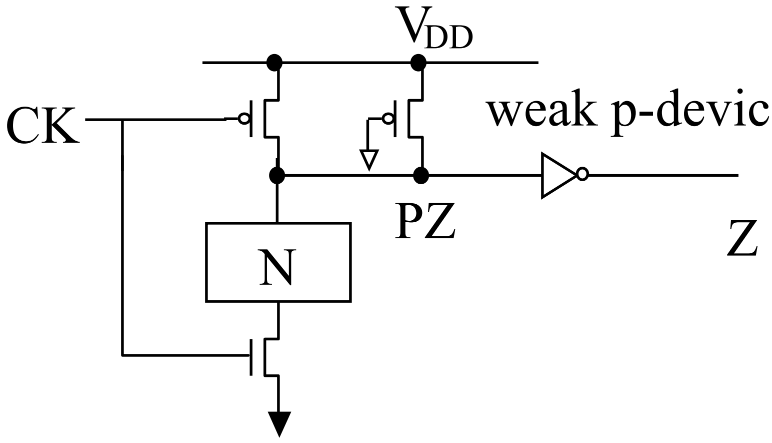
— ASSUME A=B=1 and C=1

3 ⟶ LOW

4 ⟶ HIGH

CMOS Domino Logic (Cont.)

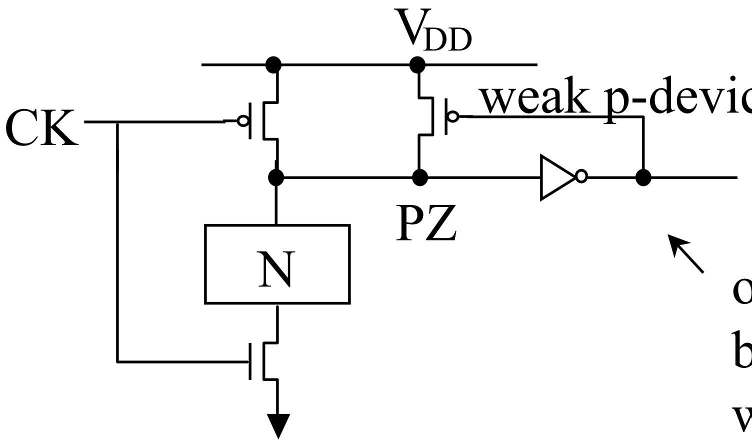
- Static version



weak p-device (small $\frac{W}{L}$)

To balance effects of leakage but not interfere with pull down

- Latched version



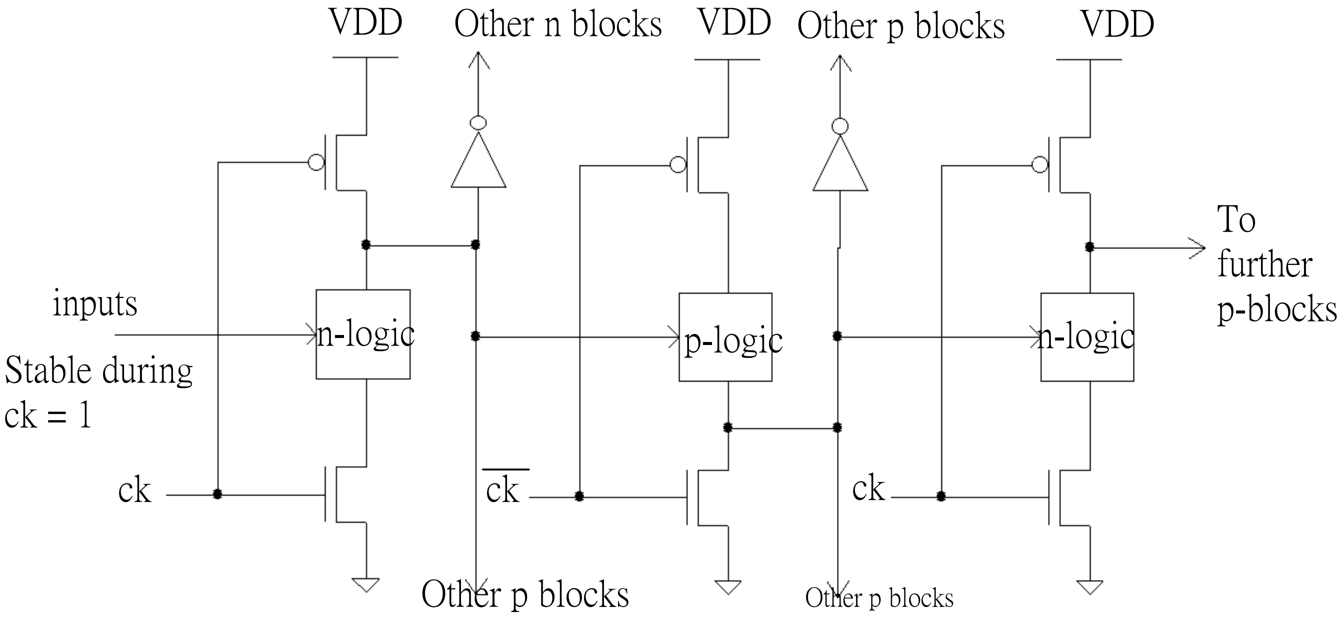
weak p-device feedback

only need to balance leakage when input to inverter is high

$\therefore$ feedback only turns on weak p-device when inverter output is 0, and input is high

NP Dominio Logic (Zipper CMOS)

- Buffer removed
- Cascaded logic blocks are alternate, p-logic and n-logic



	PRECHARGED	EVALUALED
ck = 0	n-logic state	p-logic stage
ck = 1	p-logic state	n-logic stage

- Advantages
  1. One clock
  2. No buffers

- Disadvantages
  1. Speed of p-blocks
  2. Charge sharing
     (decreased noise margin)