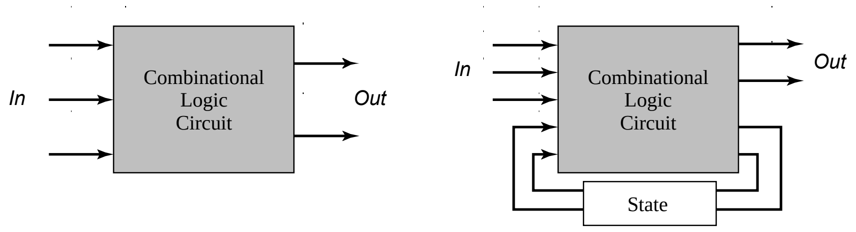
Chapter 6. Designing Combinational Logic Gates in CMOS

[Adapted from Rabaey's *Digital Integrated Circuits*, ©2002, J. Rabaey et al.]

Combinational vs. Sequential Logic



Combinational

Sequential

Output = *f*(*In*)

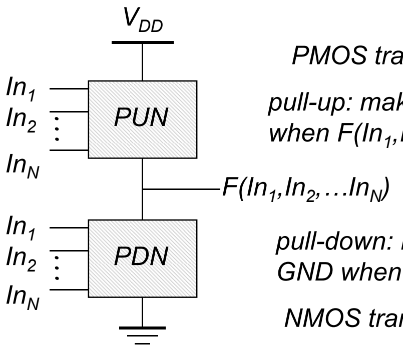
Output = *f*(*In, Previous In*)

CMOS Circuit Styles

- Static complementary CMOS - except during switching, output connected to either V_{DD} or GND via a low-resistance path
  - high noise margins
    - full rail to rail swing
    - V_{OH} and V_{OL} are at V_{DD} and GND, respectively
  - low output impedance, high input impedance
  - no steady state path between V_{DD} and GND (no static power consumption)
  - delay a function of load capacitance and transistor resistance
  - comparable rise and fall times (under the appropriate transistor sizing conditions)
- Dynamic CMOS - relies on temporary storage of signal values on the capacitance of high-impedance circuit nodes
  - simpler, faster gates
  - increased sensitivity to noise

Static Complementary CMOS

- *Pull-up network (PUN) and pull-down network (PDN)*



PMOS transistors only

pull-up: make a connection from V_{DD} to F when $F(In_1,In_2,\ldots In_N) = 1$

pull-down: make a connection from F to GND when $F(In_1,In_2,\ldots In_N) = 0$

NMOS transistors only

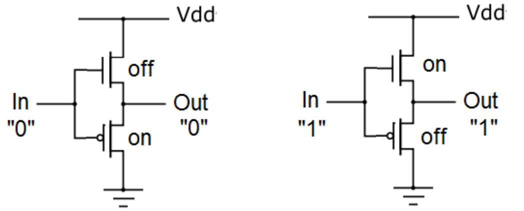
PUN and PDN are dual logic networks

Threshold Drops

❑ In a CMOS circuit, PUN is always constructed with PMOS transistors, PDN is always constructed with NMOS transistors. Why? Can we do it in an opposite way?

❑ Ex: If we exchange the NMOS and PMOS transistors in a CMOS inverter, can we use it as a buffer?

→ Not a good design due to threshold drops of NMOS and PMOS transistors.

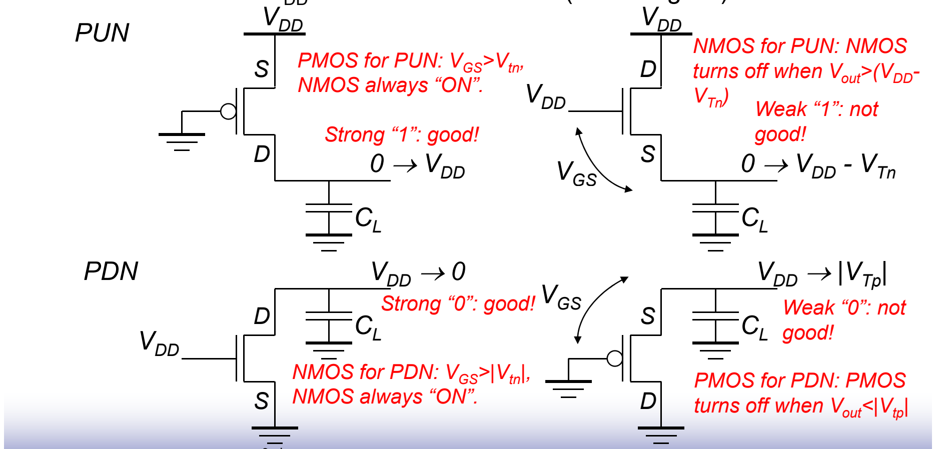


CMOS inverter: exchange NMOS and PMOS
Out=In, a buffer?

Threshold Drops

❑ Fact: NMOS transistors can pass a strong “0”, but a weak “1”. PMOS transistor can pass a strong “1”, but a weak “0”.

❑ In order to get strong “0” (Gnd) and strong “1” (V_{DD}) at output, output should connect to Gnd via NMOS transistors (for strong “0”), and should connect to V_{DD} via PMOS transistors (for strong “1”).



PMOS for PUN: V_{GS}>V_{tn}, NMOS always “ON”.

Strong “1”: good!

NMOS for PUN: NMOS turns off when V_{out}>(V_{DD}-V_{Tn})

Weak “1”: not good!

Strong “0”: good!

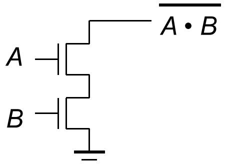
NMOS for PDN: V_{GS}>|V_{tn}|, NMOS always “ON”.

Weak “0”: not good!

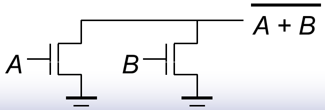
PMOS for PDN: PMOS turns off when V_{out}<|V_{tp}|

Construction of PDN

- NMOS devices connected in series implement “AND” function. However, in a CMOS circuit, when NMOS portion is turned “ON”, it connects output to Gnd (instead of Vdd). Thus the actual function implemented by NMOS devices in series in a CMOS circuit is a “NAND” function.

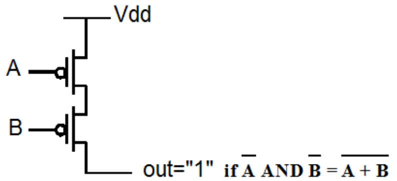


- Similarly, NMOS devices in parallel implement “OR” function. Since NMOS connects output to Gnd, the actual function implemented by NMOS devices in parallel in a CMOS circuit is a “NOR” function.

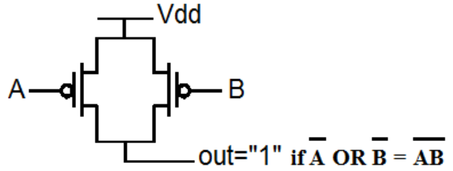


Construction of PUN

- PMOS transistors are used in PUN of CMOS circuits. PMOS transistor is turned “ON” when Vin=“0”.
- PMOS devices connected in series implement “NOR” function.


Vdd
A
B
out="1" if $\overline{A}$ AND $\overline{B}$ = $\overline{A + B}$

- PMOS devices in parallel implement “NAND” function.


Vdd
A
B
out="1" if $\overline{A}$ OR $\overline{B}$ = $\overline{AB}$

Dual PUN and PDN

- PUN and PDN are dual networks
  - DeMorgan's theorems

$\overline{A + B} = \overline{A} \bullet \overline{B}$ [!(A + B) = !A • !B or !(A | B) = !A & !B]

$\overline{A \bullet B} = \overline{A} + \overline{B}$ [!(A • B) = !A + !B or !(A & B) = !A | !B]

  - a parallel connection of transistors in the PUN corresponds to a series connection of the PDN
- Complementary gate is naturally inverting (NAND, NOR, AOI, OAI). If we want to implement a non-inverting function (e.g. AND, OR), we should cascade CMOS gate with an inverter.
- Number of transistors for an N-input logic gate is 2N

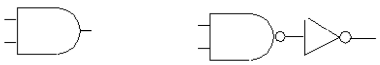
Complementary CMOS Logic Style

- PUP is the <u>DUAL</u> of PDN

  (can be shown using DeMorgan's Theorem's)

$$\overline{A+B} = \overline{A}\,\overline{B}$$
$$\overline{AB} = \overline{A} + \overline{B}$$

- The complementary gate is inverting

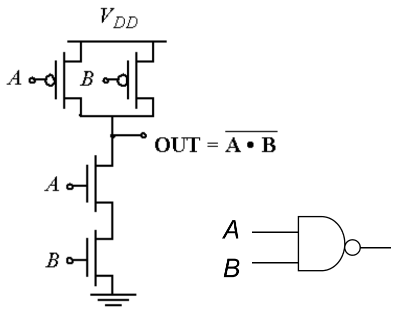


AND = NAND + INV

Example Gate: NAND

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table of a 2 input NAND gate

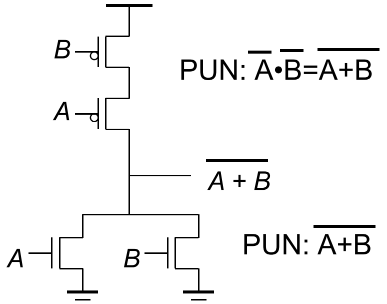


PDN: G = A B ⇒ Conduction to GND ⇒ G = $\overline{AB}$

PUN: F= $\overline{A} + \overline{B}$ = $\overline{AB}$ ⇒ Conduction to V_{DD} ⇒ G = $\overline{AB}$

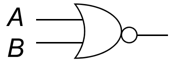
$$\overline{G(In_1, In_2, In_3, \ldots)} \equiv F(\overline{In_1}, \overline{In_2}, \overline{In_3}, \ldots)$$

CMOS NOR Gate



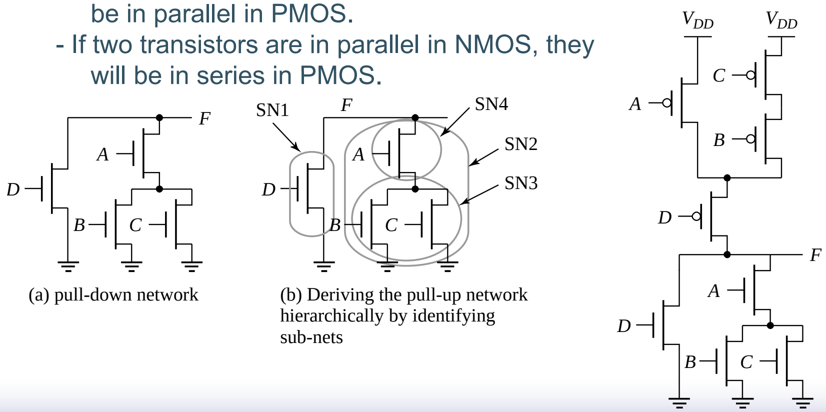
Truth Table for NOR Gate

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



Constructing a Complex Gate

- ❑ Constructing a complex CMOS gate:
- ✓ First construct PDN according to the non-inverting function.
  "•" : NMOS in serial, "+": NMOS in parallel.
- ✓ Then construct PUN according to DeMorgan theorem:
  - If two transistors are in series in NMOS, they will be in parallel in PMOS.
  - If two transistors are in parallel in NMOS, they will be in series in PMOS.

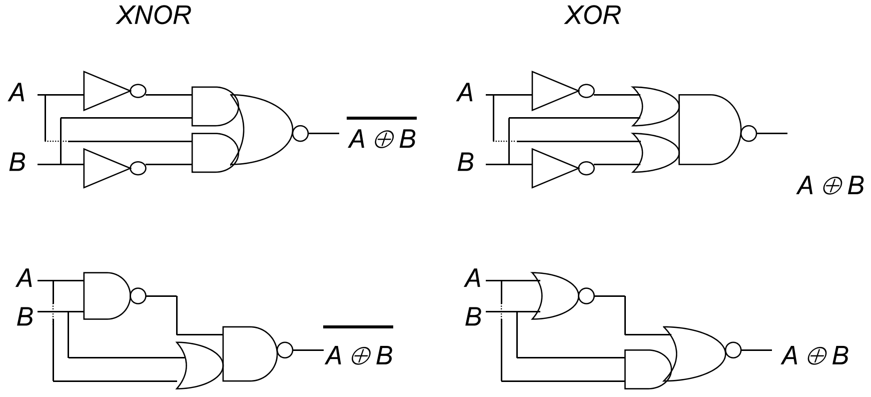


(a) pull-down network

(b) Deriving the pull-up network hierarchically by identifying sub-nets

(c) complete gate

XNOR/XOR Implementation

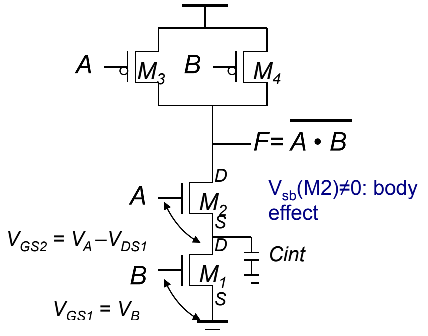


- ❑ *How many transistors in each?*
- ❑ *Can you create the stick transistor layout for the lower left circuit?*

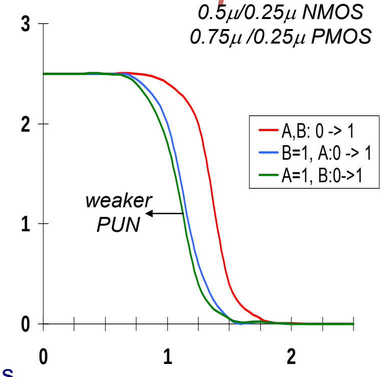
CMOS Properties

- ❑ Full rail-to-rail swing; (V_{OH}=V_{DD}, V_{OL}=Gnd). high noise margins
- ❑ Logic levels not dependent upon the relative device sizes; ratioless
- ❑ Always a path to Vdd or Gnd in steady state; low output impedance
- ❑ Extremely high input resistance; nearly zero steady-state input current
- ❑ No direct path steady state between power and ground; no static power dissipation
- ❑ Propagation delay function of load capacitance and resistance of transistors
- ❑ Comparable output rise and fall times (under appropriate sizing conditions)

CMOS Static Property: VTC is Data-Dependent

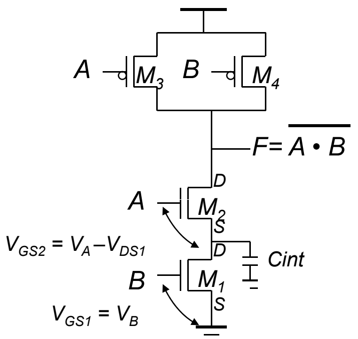


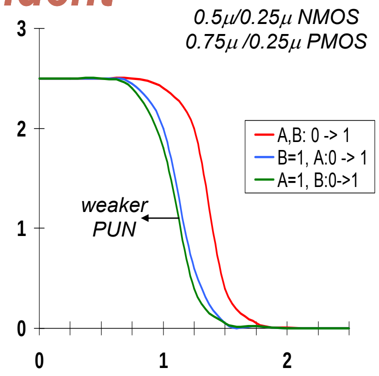
A
M_3
B
M_4
$F= \overline{A \bullet B}$
D
A
M_2
S
V_{sb}(M2)≠0: body effect
$V_{GS2} = V_A - V_{DS1}$
D
Cint
B
M_1
S
$V_{GS1} = V_B$



- ❑ VTC depends on data input patterns.
- ❑ NAND gate: 3 possible patterns can switch output from high to low: (a). A=B=0→1; (b). A=1, B=0→1; (c). B=1, A=0→1. Switching threshold: V_M(a)>V_M(c)>V_M(b). Why?
- ❑ In (a), both PMOS are ON for A=B=0, representing a strong pull-up. Thus it needs higher V_M to pull output down.
- ❑ For (b) and (c), V_{tn}(M2)>V_{tn}(M1) due to body effect. Needs higher V_A to switch M2 off.

VTC is Data-Dependent





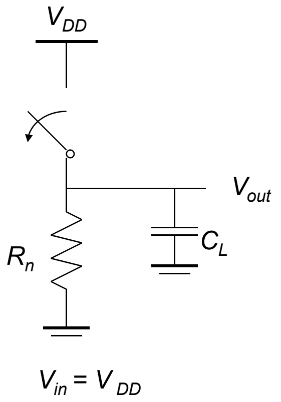
- The threshold voltage of M_2 is higher than M_1 due to the body effect (γ)

  $V_{Tn1} = V_{Tn0}$

  $V_{Tn2} = V_{Tn0} + \gamma(\sqrt{|2\phi_F| + V_{int}} - \sqrt{|2\phi_F|})$

  since V_{SB} of M_2 is not zero (when $V_B = 0$) due to the presence of Cint

CMOS Dynamic Property: Review: CMOS Inverter: Dynamic



$t_{pHL} = f(R_n, C_L)$

$t_{pHL} = 0.69\ R_{eqn}\ C_L$

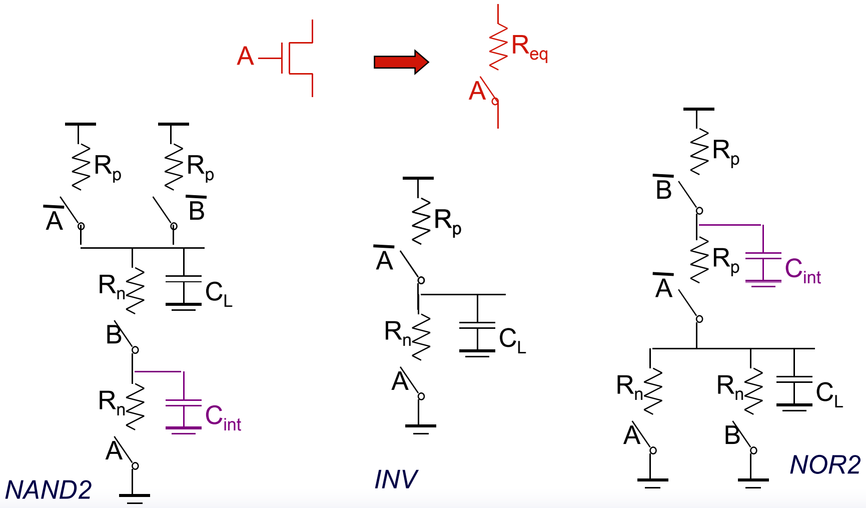
$t_{pHL} = 0.69\ (3/4\ (C_L\ V_{DD})/\ I_{DSATn})$

$= 0.52\ C_L\ /\ (W/L_n\ k'_n\ V_{DSATn})$

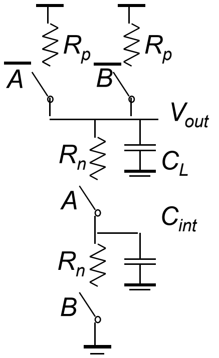
Review: Designing Inverters for Performance

- Reduce C_L
  - internal diffusion capacitance of the gate itself
  - interconnect capacitance
  - fanout
- Increase W/L ratio of the transistor
  - the most powerful and effective performance optimization tool in the hands of the designer
  - watch out for self-loading!
- Increase V_{DD}
  - only minimal improvement in performance at the cost of increased energy dissipation
- Slope engineering - keeping signal rise and fall times smaller than or equal to the gate propagation delays and of approximately equal values
  - good for performance
  - good for power consumption

Switch Delay Model



Input Pattern Effects on Delay



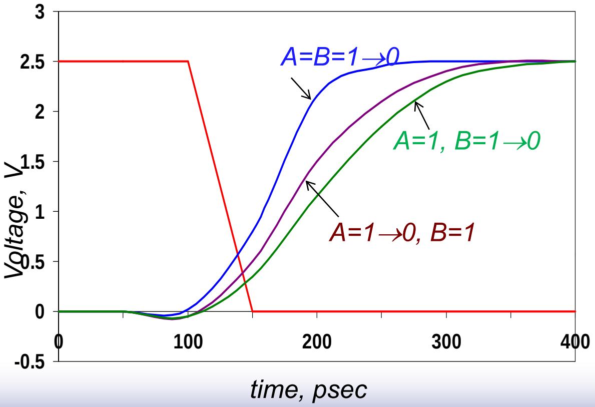
2-input NAND

C_{int}: internal node capacitance

- Delay is dependent on the pattern of inputs
- Low to high transition (t_{pLH}): Vout=0→1
  - both inputs go low (A=B=1→0)
    - delay is $0.69(R_p/2)C_L$ since two p-resistors are on in parallel. Vout is charged to V_{dd} via both PMOS transistors in parallel.
  - one input goes low (A=1, B=1→0; or B=1, A=1→0)
    - delay is $0.69R_pC_L$. V_{out} is discharged to Gnd via 2 NMOS transistors in series.
- High to low transition (t_{pHL}): V_{out}=1→0
  - both inputs go high: V_{out} must be discharged via both NMOS transistors in series.
    - delay is $0.69(2R_n)C_L$
- When we consider delays (t_{pLH}, t_{pHL}), generally we consider the worst case scenario.
- Adding transistors in series (without sizing) slows down the circuit

Delay Dependence on Input Patterns

- A=B=1→0: V_{dd} charges V_{out} via both PMOS in parallel: faster.
- A=1→0, B=1: V_{dd} needs to charge C_L via 1 PMOS transistor: medium.
- A=1, B=1→0: V_{dd} needs to charge both C_L and C_{int} via 1 PMOS transistor: slow.



2-input NAND with
NMOS = 0.5μm/0.25μm
PMOS = 0.75μm/0.25μm
C_L = 10 fF

Input Data Pattern	Delay (psec)
A=B=0→1	69
A=1, B=0→1	62
A= 0→1, B=1	50
A=B=1→0	35
A=1, B=1→0	76
A= 1→0, B=1	57

Transistor Sizing for Performance

- Transistor sizing: Adjust the size (generally the width W) of each individual transistor.
- Fact: Transistor on-resistance R_{eq} (hence the RC delay) is reversely proportional to transistor size (W/L). Thus if we want R_{eq} to be reduced to half, we should increase the transistor width (W) to two times (2W).
- Transistor sizing: generally transistor length L is fixed, only width W is changed.

  Reason: To ensure fast speed, the length of all the transistors are generally set to minimum value (2λ) and cannot be further reduced. As a result, to further improve the speed, we should enlarge transistor width (W).
- Sometimes to synchronize the signal flow to reduce glitches, we may need to match the propagation delay (t_{pLH}, t_{pHL}) to another gate (e.g. inverter, etc.)
- Delay of CMOS gate is pattern-dependent. For propagation delay, generally we consider worst-case scenario.

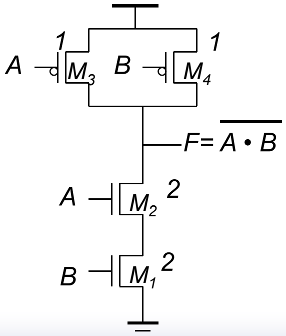
Transistor Sizing: NAND Gate

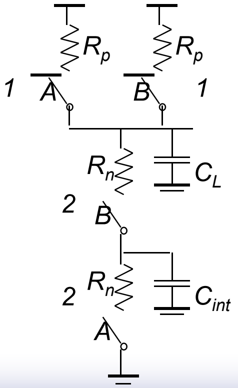
- Determin the sizes of transistors of NAND gate such that it has approximately the same t_{plh} and t_{phl} (worst-case scenario) as minimum size inverter: (W_p/L_p=9λ/2λ, W_n/L_n=3λ/2λ).
- Solution: 1). For t_{plh}, worst case scenario: Vdd is charging F only via 1 PMOS transistor (either one). Thus:

  $R_{p\_nand} \cdot C_L = R_{p\_inv} \cdot C_L \rightarrow R_{p\_nand} = R_{p\_inv}$
  $\rightarrow W_{p\_nand} = W_{p\_inv}$
  → Each PMOS size: 1

2). For t_{phl}: F has to be discharged via both NMOS transistors in series:

$2R_{n\_nand} \cdot C_L = R_{n\_inv} \cdot C_L$
$\rightarrow R_{n\_nand} = (1/2)R_{n\_inv}$
$\rightarrow W_{n\_nand} = 2W_{n\_inv}$
→ Each NMOS size: 2





Transistor Sizing: NOR Gate

- Determin the sizes of transistors of NOR gate such that it has approximately the same t_{plh} and t_{phl} (worst-case scenario) as minimum size inverter: ($W_p/L_p=9\lambda/2\lambda$, $W_n/L_n=3\lambda/2\lambda$).
- Solution: 1). For t_{plh}, Vdd has to charge F via 2 PMOS transistors in series. Thus:

  $2R_{p\_nor} \cdot C_L = R_{p\_inv} \cdot C_L \rightarrow R_{p\_nor}=(1/2)R_{p\_inv}$

  $\rightarrow W_{p\_nor}=2W_{p\_inv}$

  → Each PMOS size: 2

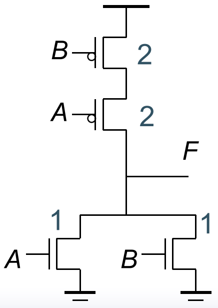
2). For t_{phl}: worst case scenario: F is discharged via 1 NMOS transistor (either one):

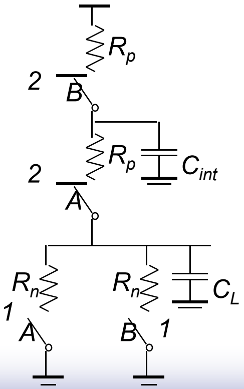
$R_{n\_nor} \cdot C_L = R_{n\_inv} \cdot C_L$

$\rightarrow R_{n\_nor}=R_{n\_inv}$

$\rightarrow W_{n\_nor}=W_{n\_inv}$

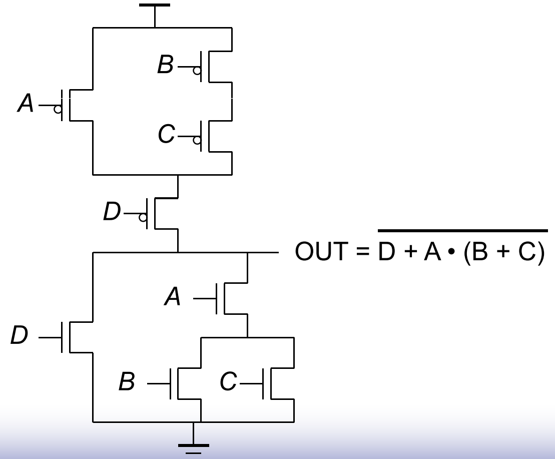
→ Each NMOS size: 1





Transistor Sizing a Complex CMOS Gate

- Determin the sizes of transistors of following gate such that it has approximately the same t_{plh} and t_{phl} (worst-case scenario) as minimum size inverter: ($W_p/L_p=9\lambda/2\lambda$, $W_n/L_n=3\lambda/2\lambda$).



Transistor Sizing a Complex CMOS Gate

- Solution: 1). For t_{pLH}, worst case: Out is charged either via PMOS transistors A-D or B-C-D.

i). If via PMOS A-D: $(R_{p\_A}+R_{p\_D})C_L=(R_{p\_inv})C_L$

$\rightarrow R_{p\_A}=R_{p\_D}=(1/2)R_{p\_inv} \rightarrow W_{p\_A}=W_{p\_D}=2W_{p\_inv}$.

Sizes for PMOS A, D: 2, 2 (in unit of W_{p_inv})

ii). If via PMOS B-C-D:

$(R_{p\_B}+R_{p\_C}+R_{p\_D})C_L=(Rp_{\_inv})C_L$,

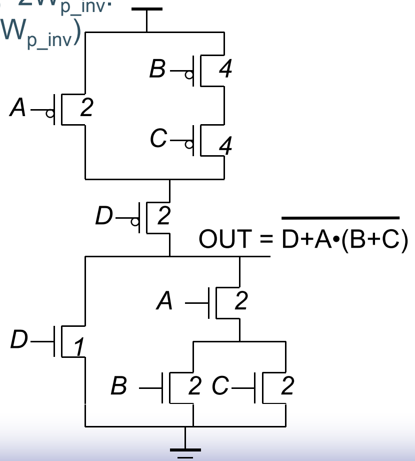
$R_{p\_D}=(1/2)R_{p\_inv}$,

$\rightarrow (R_{p\_B}+R_{p\_C})=(1/2)R_{p\_inv}$

$\rightarrow R_{p\_B}=R_{p\_C}=(1/4)R_{p\_inv}$

$\rightarrow W_{p\_B}=W_{p\_C}=4W_{p\_inv}$,

Sizes for PMOS B,C: 4,4 (in unit of W_{p_inv})



Transistor Sizing a Complex CMOS Gate

- Solution: 2). For t_{pHL}, worst case: Out is discharged either via NMOS transistors D, or A-B, or A-C.

i). If via NMOS D: $R_{n\_D}C_L=(R_{n\_inv})C_L$

$\rightarrow R_{n\_D}=R_{n\_inv} \rightarrow W_{n\_D}=W_{n\_inv}$.

Sizes for NMOS D: 1 (in unit of W_{n_inv})

ii). If via NMOS A-B:

$(R_{n\_A}+R_{n\_B})C_L=(R_{n\_inv})C_L$,

$\rightarrow (R_{n\_A}+R_{n\_B})=R_{n\_inv}$

$\rightarrow R_{n\_A}=R_{n\_B}=(1/2)R_{n\_inv}$

$\rightarrow W_{n\_A}=W_{n\_B}=2W_{n\_inv}$,

Sizes for NMOS A,B: 2,2 (in unit of W_{n_inv}).

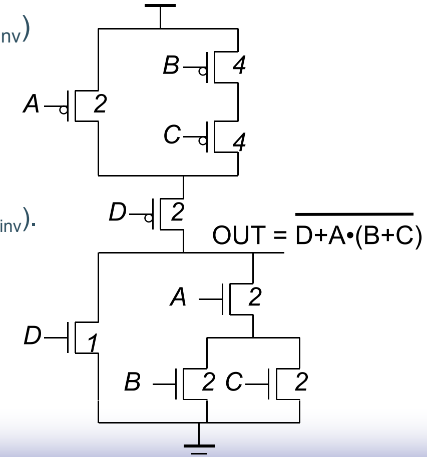
iii). Similarly, if via NMOS A-C:

$(R_{n\_A}+R_{n\_C})C_L=(R_{n\_inv})C_L$,

$\rightarrow (R_{n\_A}+R_{n\_C})=R_{n\_inv}$, $R_{n\_A}=(1/2)R_{n\_inv}$,

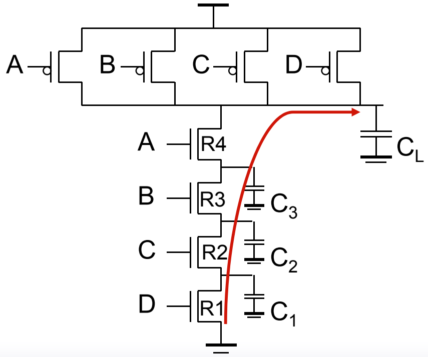
$\rightarrow R_{n\_C}=(1/2)R_{n\_inv} \rightarrow W_{n\_C}=2W_{n\_inv}$,

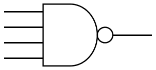
Sizes for NMOS C: 2 (in unit of W_{n_inv}).



Fan-In Considerations

- ❑ For gate with large fan-ins: internal node capacitance become significant.
- ❑ Consider tpHL, Gnd (0V) needs to propagate to output via four NMOS transistors.





❑ Distributed RC model (Elmore delay):
t_{pHL}=069[R1C1+(R1+R2)C2+(R1+R2+R3)C3+(R1+R2+R3+R4)CL]

✓ R1 appears in all the terms: transistor M(D) is important to minimize delay.

✓ If all NMOS have equal size:
$t_{pHL} = 0.69\ R_{eqn}(C_1+2C_2+3C_3+4C_L)$

❑ Propagation delay deteriorates rapidly as a function of fan-in – quadratically in the worst case.

T_p a Function of Fan-In

❑ CMOS NAND gate:

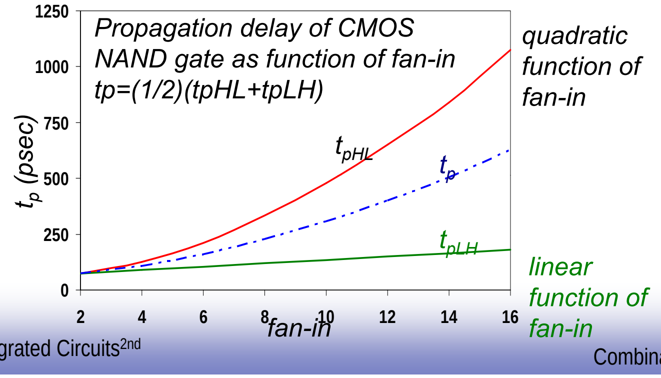
✓ t_{pLH}: increases linearly with fan-in (N).

Reason: For NAND gate, PMOS transistors are in parallel. Cint at output increases linearly with N, → t_{pLH} increase linearly with N,

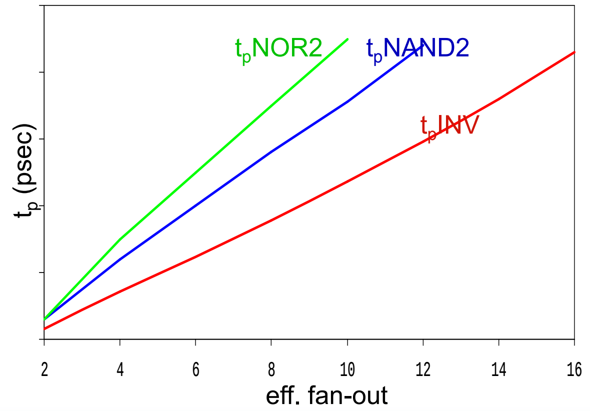
✓ t_{pHL}: increase quadratically with fan-in (N).

Reason: See Elmore delay model of distributed RC network. NMOS transistors in NAND gate are connected in series.

❑ Gates with a fan-in greater than 4 should be avoided.



t_p as a Function of Fan-Out



All gates have the same drive current.

Slope is a function of “driving strength”

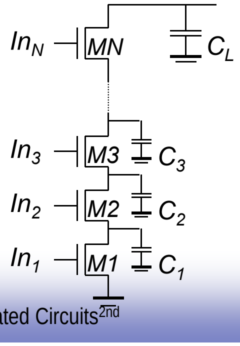
t_p as a Function of Fan-In and Fan-Out

- ❑ Fan-in: quadratic due to increasing resistance and capacitance
- ❑ Fan-out: each additional fan-out gate adds two gate capacitances to C_L

$$t_p = a_1FI + a_2FI^2 + a_3FO$$

Fast Complex Gates: Design Technique 1

- Transistor sizing
  - as long as fan-out capacitance dominates
- Progressive sizing: If N transistors are connected in series, according to distributed RC line delay model, R(M1) appears N times in delay equation, R(M2) appears (N-1) times in delay equation… → R(M1) is most important, should be sized the largest to reduce dominant resistance.
- ✓ Progressive sizing: Graduately increase transistor size along transistors in series when moving away from output.



Distributed RC line

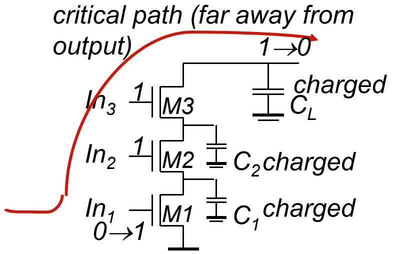
M1 > M2 > M3 > … > MN

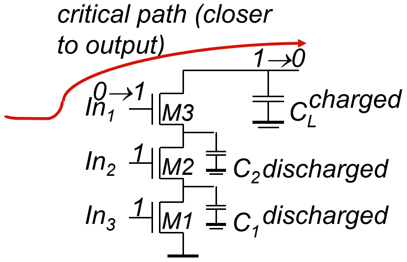
(the fet closest to the output should be the smallest)

Can reduce delay by more than 20%; decreasing gains as technology shrinks

Fast Complex Gates: Design Technique 2

- Input re-ordering
  - when not all inputs arrive at the same time, the input become stable the last is called critical input. The path through the logic which determine the ultimate speed of the circuit is called critical path.
  - Putting critical-path transistors closer to gate output can speed up the circuit.

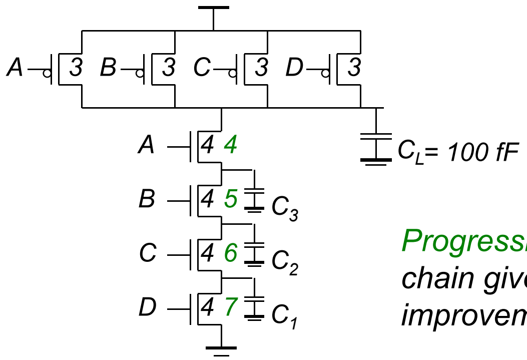


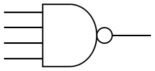


delay determined by time to discharge C_L, C_1 and C_2: slow

delay determined by time to discharge C_L only. (C1, C2 are previously already discharged.): fast

Sizing and Ordering Effects





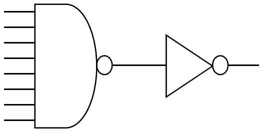
Progressive sizing in pull-down chain gives up to a 23% improvement.

Input ordering saves 5%
critical path A – 23%
critical path D – 17%

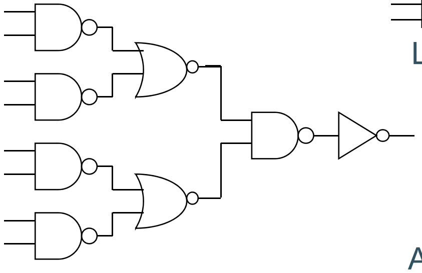
Fast Complex Gates: Design Technique 3

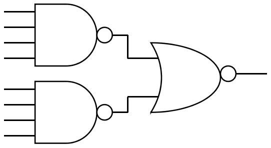
- Logic Restructuring: Manipulating logi equations can reduce fan-in requirements and thus reduce the gate delay. Alternative logic structures

F = ABCDEFGH



Large fan-in: slow



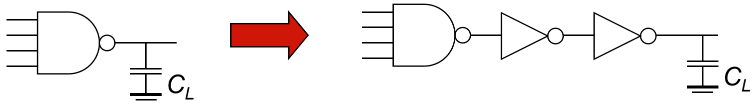


Alternative logic structure: less fan-in, faster

Alternative logic structure: less fan-in, faster

Fast Complex Gates: Design Technique 4

- Isolating fan-in from fan-out using buffer insertion



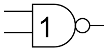
- *Real lesson is that optimizing the propagation delay of a gate in isolation is misguided.*

Fast Complex Gates: Design Technique 5: Sizing Logic Path for Speed

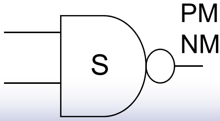
- Frequently, input capacitance of a logic path is constrained
- Logic also has to drive some capacitance
- Example: ALU load in an Intel's microprocessor is 0.5pF
- How do we size the ALU datapath to achieve maximum speed?
- We have already solved this for the inverter chain – can we generalize it for any type of logic?

Transistor Sizing and Gate Sizing

- Transistor/gate sizing: generally transistor length L is fixed, only width W is changed.
- Transistor sizing: Adjust the size (generally the width W) of each individual transistor.
- Gate sizing: Enlarge or shrink the size (generally the width W) of all the PMOS and NMOS transistors in a gate simultaneously by factor S. That is,
  - ✓ Original gate (size "1"): PMOS: (W_{pi}/L_{pi}), NMOS: (W_{ni}/L_{ni}),
  - ✓ The same gate with size S: PMOS: $(W_{pi}/L_{pi}) \rightarrow (S \cdot W_{pi}/L_{pi})$, NMOS: $(W_{ni}/L_{ni}) \rightarrow (S \cdot W_{ni}/L_{ni})$.
- Sizing a gate from size "1" to size "S" does not change the size ratio between PMOS and NMOS transistors, hence it does not change the (t_{pHL}/t_{pLH}) ratio of the gate.



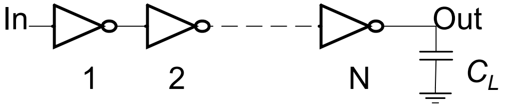
PMOS: (W_{pi}/L_{pi})
NMOS: (W_{ni}/L_{ni})



PMOS: $(S \cdot W_{pi}/L_{pi})$
NMOS: $(S \cdot W_{ni}/L_{ni})$

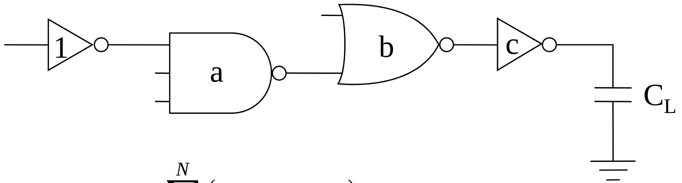
Buffer Example

- The delay of an inverter chain can be minimized by proper sizing:



For given N: $C_{i+1}/C_i = C_i/C_{i-1}$, $f=(F)^{1/N}$.
To find N: $f=C_{i+1}/C_i \sim 4$, $N=\ln(F)/\ln(f)$.

- How to generalize this strategy to any logic path?



$$Delay = t_{p0}\sum_{i=1}^{N}\left(p_i + g_i \cdot f_i\right)$$ (t_{p0}: delay of an inverter)

Sizing the Gates base on Logical Effort

- The optimum fan-out for a chain of N inverters driving a load C_L is

$$f = (C_L/C_{in})^{1/N}$$

  - so, if we can, keep the fan-out per stage around 4 (f_{opt}=4 for γ=1).
- Can the same approach (logical effort) be used for any combinational circuit?
  - For a complex gate, we expand the inverter equation

$$t_p = t_{p0}\,(1 + C_{ext}/\ \gamma C_g) = t_{p0}\,(1 + f/\gamma)$$

to

$$\boxed{t_p = t_{p0}\,(p\ +\ g\ f/\gamma)}$$

- – t_{p0} is the intrinsic delay of an inverter
- – f is the effective fan-out (C_{ext}/C_g) – also called electrical effort
- – p is the ratio of the instrinsic (unloaded) delay of the complex gate and a simple inverter (a function of the gate topology and layout style). It reflects the fact that the intrinsic delay of a complex CMOS gate is larger than that of an inverter.
- – g is logical effort. It reflects the fact that a complex gate causes more delay than an inverter when it is used as fanout (load).

Intrinsic Delay Term, p

- The more involved the structure of the complex gate, the higher the intrinsic delay compared to an inverter

Gate Type	p
Inverter	1
n-input NAND	n
n-input NOR	n
n-way mux	2n
XOR, XNOR	n 2^{n-1}

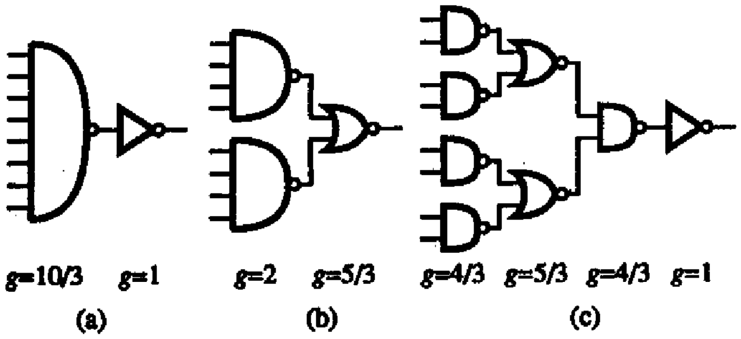
Ignoring second order effects such as internal node capacitances

Logical Effort Term, g

- g represents the fact that, for a given load, complex gates have to work harder than an inverter to produce a similar (speed) response
  - the logical effort of a gate tells how much worse it is at producing an output current than an inverter (how much more input capacitance a gate presents to deliver it same output current)

Gate Type	g (for 1 to n input gates)			
	1	2	3	n
Inverter	1			
NAND		4/3	5/3	(n+2)/3
NOR		5/3	7/3	(2n+1)/3
mux		2	2	2
XOR		4	12	

Example – 8-input AND

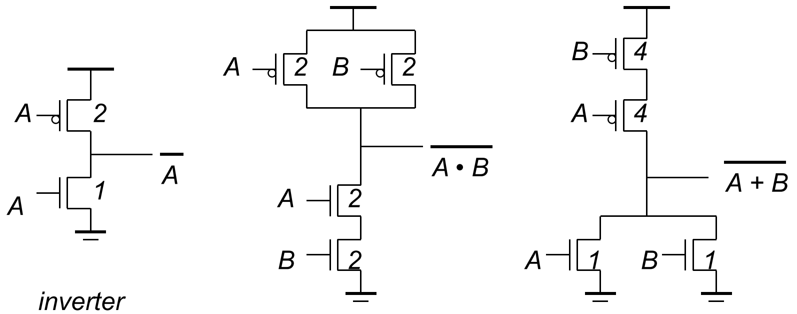


Logical Effort

- ❑ Inverter has the smallest logical effort and intrinsic delay of all static CMOS gates
- ❑ Logical effort of a gate presents the ratio of its input capacitance to the input capacitance of an inverter when the gate is sized to deliver the same output current
- ❑ Logical effort increases with the gate complexity

Example of Logical Effort

- ❑ Assuming a pmos/nmos ratio of 2, the input capacitance of a minimum-sized inverter is three times the gate capacitance of a minimum-sized nmos (C_{unit})
- ❑ Logical effort is the ratio of input capacitance of a gate to the input capacitance of an inverter with the same output current.



inverter
$C_{unit} = 3$
g = 1

2-input NAND
$C_{unit} = 4$
g = 4/3

2-input NOR
$C_{unit} = 5$
g = 5/3

Delay in a Logic Gate

Gate delay:

$$d = p + h$$

intrinsic delay **effort delay**

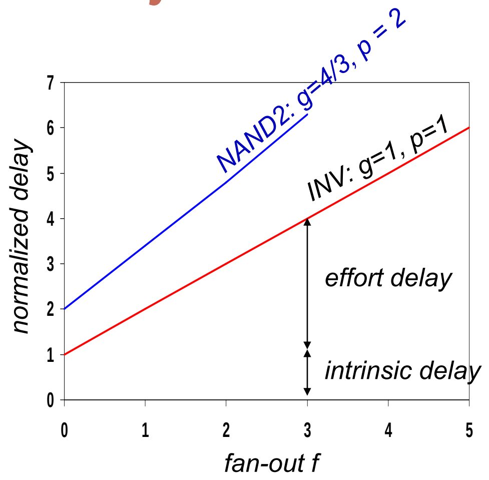
Effort delay:

$$h = g\,f$$

logical effort **effective fanout = C_{out}/C_{in}**

Logical effort is a function of topology, independent of sizing
Effective fanout (electrical effort) is a function of load/gate size

Delay as a Function of Fan-Out



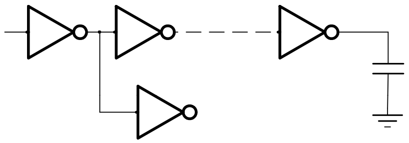
- ❑ The slope of the line is the logical effort of the gate
- ❑ The y-axis intercept is the intrinsic delay

- ❑ *Can adjust the delay by adjusting the effective fan-out (by sizing) or by choosing a gate with a different logical effort*
- ❑ *Gate effort: h = fg*

Add Branching Effort

Branching effort:

$$b = \frac{C_{on-path} + C_{off-path}}{C_{on-path}}$$



Multistage Networks

$$Delay = \sum_{i=1}^{N} (p_i + g_i \cdot f_i)$$

Stage effort (or gate effort): $h_i = g_i f_i$

Path electrical effort: $F = C_{out}/C_{in}$

Path logical effort: $G = g_1 g_2 \dots g_N$

Branching effort: $B = b_1 b_2 \dots b_N$

Path effort: $H = GFB$

Path delay $D = \Sigma d_i = \Sigma p_i + \Sigma h_i$

Optimum Effort per Stage

❑To minimize the total delay through the path, each stage should bear the same effort:

$$h^N = H$$

$$h = \sqrt[N]{H}$$

Stage efforts: $g_1 f_1 = g_2 f_2 = \dots = g_N f_N = h$

Effective fanout of each stage: $f_i = h/g_i$

Minimum path delay (Generally γ=1)

$$D = t_{p0}\left(\sum_{j=1}^{N} p_j + \frac{N\sqrt[N]{H}}{\gamma}\right) = t_{p0}(P + NH^{1/N}/\gamma)$$

Path Delay of Complex Logic Gate Network

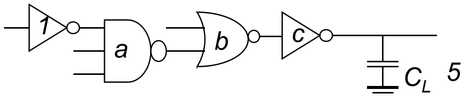
- ❑ Total path delay through a combinational logic block

$$t_p = \Sigma\, t_{p,j} = t_{p0}\, \Sigma(p_j + (f_j\, g_j)/\gamma\,)$$

- ❑ So, the minimum delay through the path determines that each stage should bear the same gate effort

$$f_1 g_1 = f_2 g_2 = \dots = f_N g_N = h = (H)^{1/N}$$

- ❑ Consider optimizing the delay through the logic network



how do we determine a, b, and c sizes?

Steps for Path Delay Optimization using Logical Efforts

- ❑ *The path logical effort, $G = \prod g_i$*
- ❑ *And the path effective fan-out (path electrical effort) is $F = C_L/g_1$*
- ❑ *The branching effort accounts for fan-out to other gates in the network*

$$b = (C_{on\text{-}path} + C_{off\text{-}path})/C_{on\text{-}path}$$

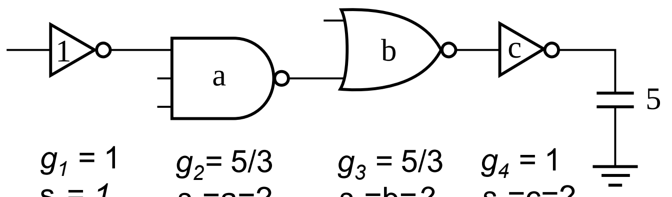
- ❑ *The path branching effort is then $B = \prod b_i$*
- ❑ *And the total path effort is then H = GFB*
- ❑ *Optimum gate effort for each stage:*

$$h = (H)^{1/N} = f_1g_1 = f_2g_2 = . . . = f_Ng_N$$

- ❑ *Effective fanout (electrical effort) of each stage: $f_i=h/g_i$*
- ❑ *Optimum size of each gate:* $s_i = \left(\frac{g_1s_1}{g_i}\right)\prod_{j=1}^{i-1}\left(\frac{f_j}{b_j}\right)$
- ❑ *So, the minimum delay through the path is*

$$D = t_{p0} (\Sigma p_j + (N\cdot H^{1/N})/ \gamma)$$

Example: Path Delay Optimization using Logical Efforts



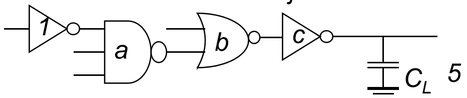
$g_1 = 1$, $s_1= 1$
$g_2= 5/3$, $s_2=a=?$
$g_3 = 5/3$, $s_3=b=?$
$g_4 = 1$, $s_4=c=?$

Effective fanout, F =?
G = ?
H = ?
h = ?
f_i= ?
a = ?
b = ?
c= ?

Example: Path Delay Optimization using Logical Efforts

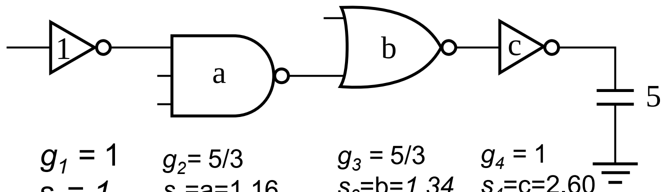
- ❑ For gate i in the chain, its size is determined by

$$s_i = [(g_1\ s_1)/g_i]\times \prod_{j=1}^{i-1} (f_j/b_j)$$



- ❑ *Following the steps as discussed before:*
  - *$F = C_L/C_{g1} = 5$*
  - *$G = g_1g_2g_3g_4$ = 1 x 5/3 x 5/3 x 1 = 25/9*
  - *$B = b_1b_2b_3b_4$= 1 x 1 x 1 x 1 =1 (no branching)*
  - *H = GFB = 125/9, so the optimal stage effort is $h=(H)^{1/N}=(H)^{1/4}$= 1.93*
    - *Fan-out factors are f_1=h/g1=1.93/1=1.93, $f_2=h/g_2$=1.93/(5/3)=1.16, $f_3=h/g_3$=1.93/(5/3)=1.16, $f_4=(h/g_4)$=1.93/1=1.93*
  - *So the gate sizes are: s_1=1, $s_2=a=(g_1s_1f_1)/(g_2b_1)$=1.16, $s_3=b=(g_1s_1/g_3)\cdot[(f_1f_2)/(b_1b_2)]$=1.34 $s_4=c=(g_1s_1/g_4)\cdot[(f_1f_2f_3)/(b_1b_2b_3)]$=2.60*

Example: Path Delay Optimization using Logical Efforts



$g_1 = 1$, $s_1= 1$
$g_2= 5/3$, $s_2=a=1.16$
$g_3 = 5/3$, $s_3=b=1.34$
$g_4 = 1$, $s_4=c=2.60$

Effective fanout, F = 5
G = 25/9
H = 125/9 = 13.9
h = 1.93
$a = (g_1s_1f_1)/(g_2b_1)=1.16$
$b = (g_1s_1/g_3)\cdot[(f_1f_2)/(b_1b_2)]$ =1.34
$c = (g_1s_1/g_4)\cdot[(f_1f_2f_3)/(b_1b_2b_3)]$ =2.60

Summary

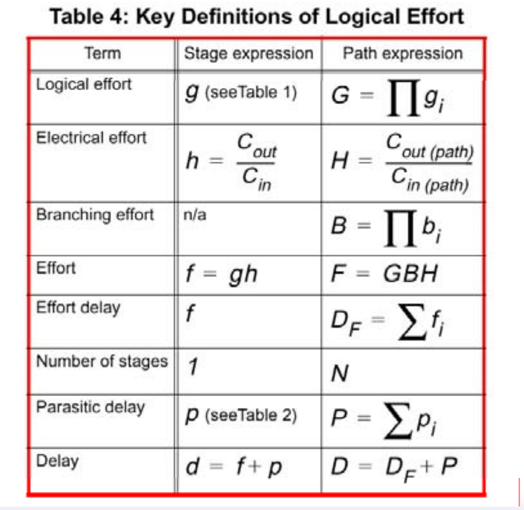


Table 4: Key Definitions of Logical Effort

Term	Stage expression	Path expression
Logical effort	g (seeTable 1)	$G = \prod g_i$
Electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out\ (path)}}{C_{in\ (path)}}$
Branching effort	n/a	$B = \prod b_i$
Effort	$f = gh$	$F = GBH$
Effort delay	f	$D_F = \sum f_i$
Number of stages	1	N
Parasitic delay	p (seeTable 2)	$P = \sum p_i$
Delay	$d = f + p$	$D = D_F + P$

Sutherland,
Sproull
Harris

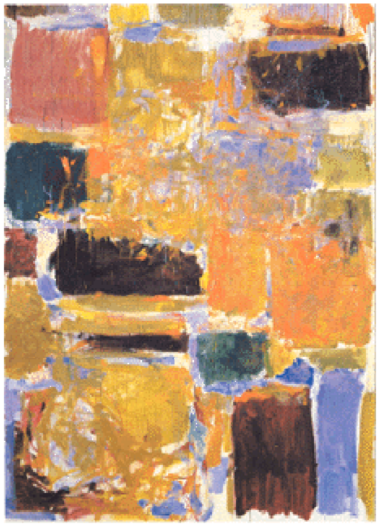
Fast Complex Gates: Design Technique 6

- Reducing the voltage swing

$$t_{pHL} = 0.69\ (3/4\ (C_L\ V_{DD}) /\ I_{DSATn})$$

$$= 0.69\ (3/4\ (C_L\ V_{swing}) /\ I_{DSATn})$$

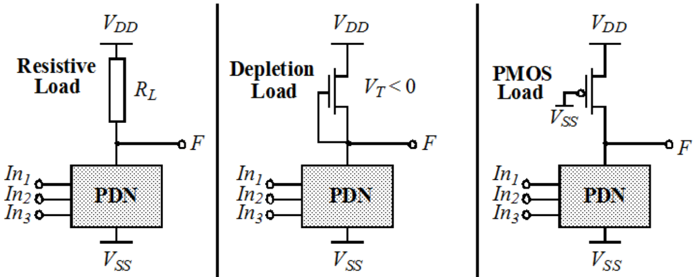
- linear reduction in delay
- also reduces power consumption
- requires use of "sense amplifiers" on the receiving end to restore the signal level (will look at their design when covering memory design)



Ratioed Logic

Ratioed Logic

- Purpose: to reduce number of transistors required to implement a given logic function, often at the cost of reduced robustness and extra power.
- Implementation: Replace PUN with a single unconditional load device.

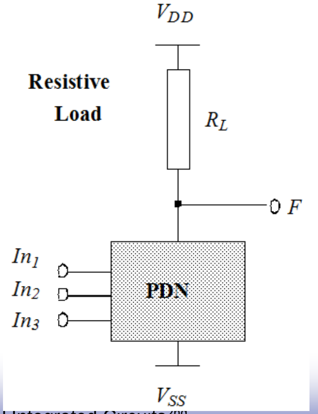


(a) resistive load (b) depletion load NMOS (c) pseudo-NMOS

Goal: to reduce the number of devices over complementary CMOS

Ratioed Logic with Resistive Load

- ❑ Ratioed logic with resistive load: $V_{OH}=V_{DD}$ (good), but $V_{OL}=V_{DD}\cdot R_{PDN}/(R_{PDN}+R_L)\neq 0$.
- ❑ In order for $V_{OL}\rightarrow 0$, we should set: $R_L>>R_{PDN}$. That is, output voltage swing and functionality of the gate depends on size ratio of PDN and R_l → ratioed logic. (Static CMOS: ratioless)



- ❑ N transistors + Load
- ❑ PDN is OFF: $V_{OH}=V_{DD}$
- ❑ PDN is ON: $V_{OL}=V_{DD}\cdot R_{PDN}/(R_{PDN}+R_L)\neq 0$
- ❑ Asymmetrical response
- ✓ $t_{pLH}\approx 0.69R_LC_L$
- ✓ $t_{pHL:}$ pattern-dependent.
- ❑ Static power consumption (when F=0)

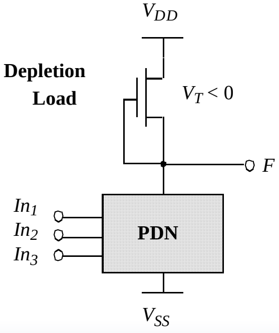
Ratioed Logic with Active Loads

❑ Depletion load NMOS: use depletion mode NMOS transistor as load.

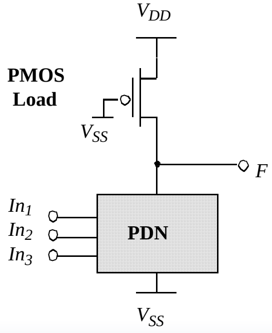
✓ Depletion mode NMOS: a conductive channel already exists when $V_{GS}=0$ (normally "ON"). In order to turn it off, a negative V_{GS} should be applied (i.e. $V_{Tn}<0$). It is commonly used as resistor instead of switch.

❑ Pseudo-NMOS: a grounded PMOS load.

✓ Question: Can we use a NMOS load with Gate tied to Vdd? Why?



depletion load NMOS



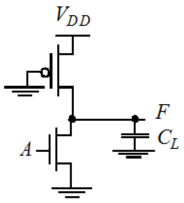
pseudo-NMOS

Pseudo-NMOS Inverter: VTC

❑ Pseudo-NMOS inverter:

✓ $V_{OH}=V_{DD}$ (when A=0)

✓ V_{OL}=? Set $V_A=V_{DD}$, NMOS in linear mode, PMOS in saturation mode, current $I_n=I_p$:



$$k_n\left[\left(V_{DD}-V_{Tn}\right)V_{OL}-V_{OL}^2/2\right]+k_p\left[\left(-V_{DD}-V_{Tp}\right)\cdot V_{DSATp}-V_{DSATp}^2/2\right]=0$$

$$V_{OL}\approx\frac{\mu_p\cdot W_p}{\mu_n\cdot W_n}\cdot V_{DSATp}$$

❑ To make $V_{OL}\rightarrow 0$, we should set: $W_p<<W_n$. But reducing W_p increases t_{pLH}, hence degrading circuit speed.

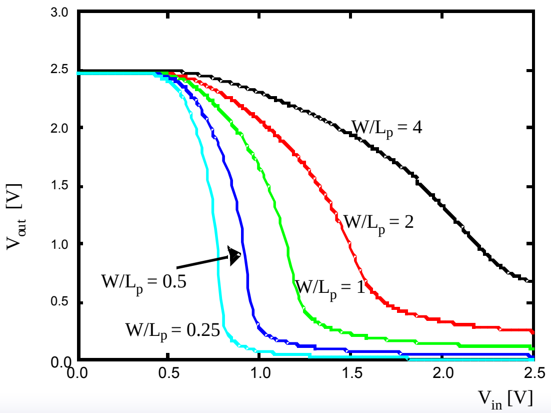
❑ Static power consumption when $V_{out}=0$:

$$P_{low}=V_{DD}I_{low}\approx V_{DD}\cdot\left|k_p\left[\left(-V_{DD}-V_{Tp}\right)\cdot V_{DSATp}-V_{DSATp}^2/2\right]\right|$$

Pseudo-NMOS Inverter VTC

❑ Pseudo-NMOS inverter VTC:

✓ As $(W/L)_p$ increases, V_{OL} increases, eventually it may cause the inverter to malfunction → ratioed logic.



Pseudo-NMOS Gates: 4-input NOR/NAND

❑ Question: Design 4-input pseudo-NMOS NOR gate and NAND gate?

NOR4

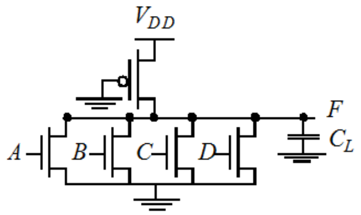
NAND4

Pseudo-NMOS Gates: 4-input NOR/NAND

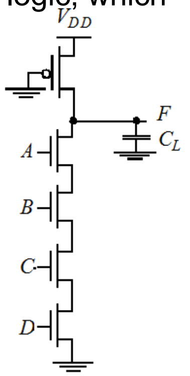
❑ Question: Design 4-input pseudo-NMOS NOR gate and NAND gate?

❑ Question: For circuit implementation in pseudo-NMOS, given the choice between NOR and NAND logic, which one would you prefer? Why?

(Hint: Consider from performance perspective)



NOR4



NAND4

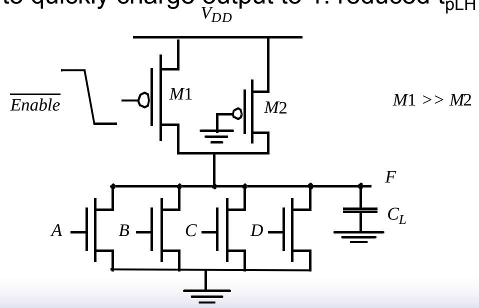
Improved Loads (1): Adaptive Load

❑ In order for V_{OL}→0, M_2 should be small (large R_{on}), this increases t_{pLH} delay → slow speed.

❑ Solution: Add a wide PMOS enable transistor M_1 ($W(M_1)>>W(M_2)$), used in address decoder of memory.

✓ In standby mode, Enable=0, Enable'=1, M1 is off: will not affect the circuit,

✓ When address change is detected, Enable=0→1, Enable'=1→0, M1 is turned ON to quickly charge output to 1: reduced t_{pLH} delay.



Adaptive Load

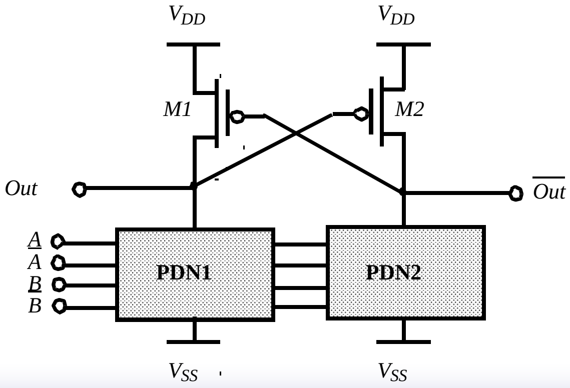
Improved Loads (2): DCVSL

❑ Differential Cascode Voltage Switch Logic (DCVSL): eliminates static currents and provides rail-to-rail voltage swing.

✓ Each input is provided in complementary format, it produces complementary outputs in turn.

✓ Feedback mechanism ensures load device is turned off when not needed.

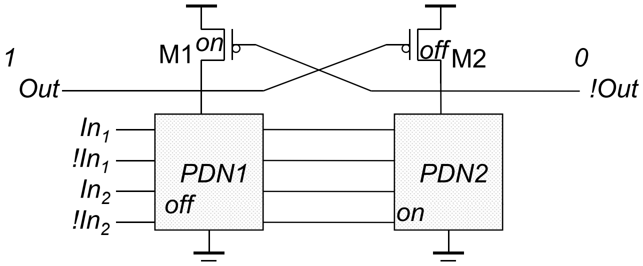
✓ Pull-down networks PDN1 and PDN2 are mutually exclusive: When PDN1 is ON, PDN2 is off; when PDN1 is OFF, PDN2 is ON.



Differential Cascode Voltage Switch Logic (DCVSL)

DCVS Logic: Working Principle

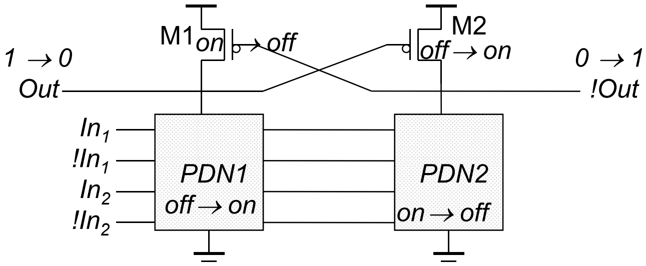
❑ DCVSL working principle: For example, if initially PDN1 is off, PDN2 is on, Out=1, Out'=0, M1: on, M2: off.



PDN1 and PDN2 are mutually exclusive

DCVS Logic: Working Principle

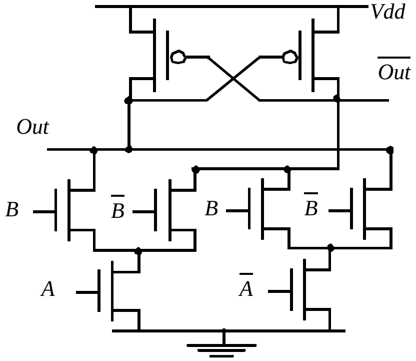
❑ If input pattern changes so that PDN1: off→on, PDN2: on→off. Now Out' is floating, M1 and PDN1 are both on. PDN1 must be strong enough to bring "Out" below V_{DD}-$|V_{Tp}|$, so that M2 switches off→on and starts charging Out' to V_{DD} (0→1). This further turns M1 on→off, so that "Out" can be fully pulled-down to Gnd (1→0).



PDN1 and PDN2 are mutually exclusive

DCVSL Example

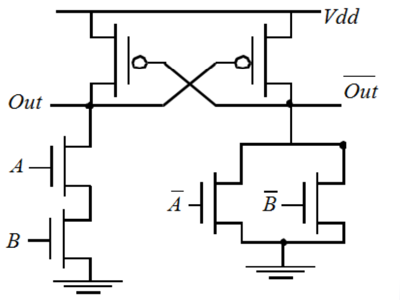
❑ DCVSL XOR-XNOR gate (It's possible to share transistors among two pull-down networks to reduce circuit area).
Question: Out=? Out'=?

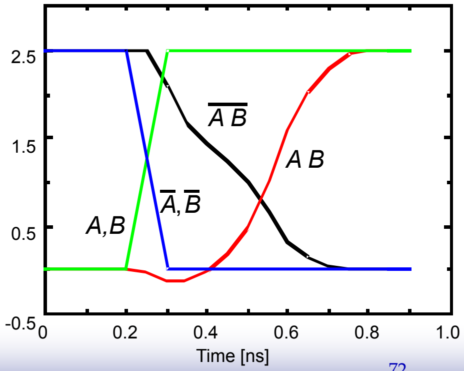


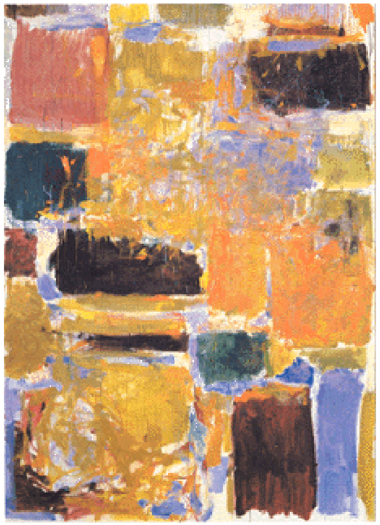
XOR-XNOR gate

DCVSL Transient Response

❑ DCVSL AND/NAND gate: Out=? Out'=?
❑ Transient response: at t=0.2ns, AB=00→11.
❑ Advantage of DCVSL: Both Out and Out' are obtained simultaneously. (In static CMOS, Out is generated from Out' using an inverter: extra inverter delay between Out' and Out)





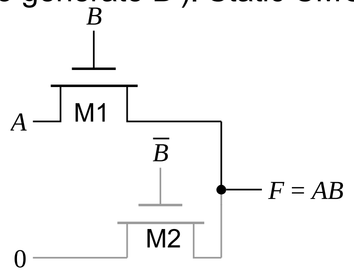


Pass-Transistor Logic

Pass-Transistor Logic (PTL)

❑ Pass-transistor logic (PTL): to reduce No. of transistors required to implement logic by allowing the primary inputs to drive gate terminals as well as source-drain terminals. (Static CMOS: inputs only drive gate terminals).

❑ Example: AND gate in PTL: 4 transistors (2 for inverter to generate B'). Static CMOS: 6 transistors.



❑ Output $F=A\cdot B+B'\cdot 0=AB$

❑ Verify:

✓ AB=00: M1 off, M2 on, F=0.

✓ AB=01: M1 on, M2 off, F=A=0.

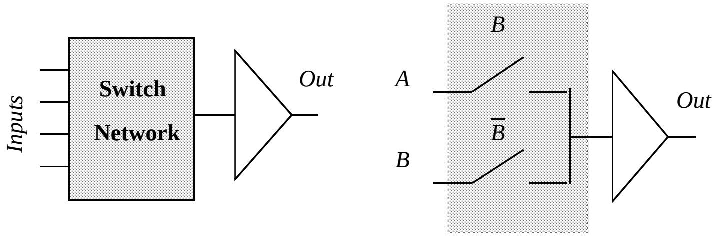
✓ AB=10: M1 off, M2 on, F=0

✓ AB=11: M1 on, M2 off, F=A=1.

→ correct function of AND gate.

❑ Question: can we remove M2? → No. Otherwise, when B=0, F is floating. A low-impedance path must exist to supply rails under all circumstances.

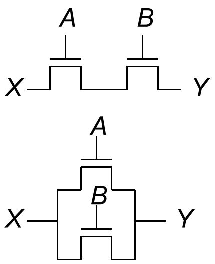
Pass-Transistor Logic



- N transistors
- No static consumption

NMOS Transistors in Series/Parallel

- Primary inputs drive both gate and source/drain terminals
- NMOS switch closes when the gate input is high



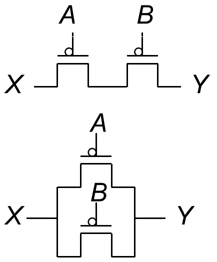
X = Y if A and B

X = Y if A or B

- Remember - NMOS transistors pass a strong 0 but a weak 1

PMOS Transistors in Series/Parallel

- ❑ Primary inputs drive both gate and source/drain terminals
- ❑ PMOS switch closes when the gate input is low

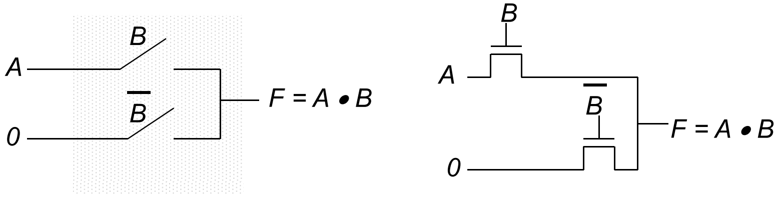


$X = Y$ if $\overline{A}$ and $\overline{B} = \overline{A + B}$

$X = Y$ if $\overline{A}$ or $\overline{B} = \overline{A \bullet B}$

- ❑ Remember - PMOS transistors pass a strong 1 but a weak 0

Pass Transistor (PT) Logic

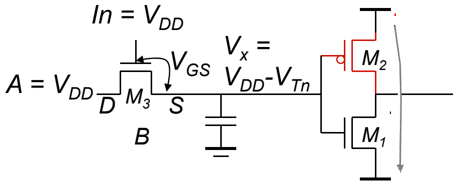


F=A·B+B'·0=A·B

- ❑ *Gate is static – a low-impedance path exists to both supply rails under all circumstances*
- ❑ *N transistors instead of 2N*
- ❑ *No static power consumption*
- ❑ *Ratioless*
- ❑ *Bidirectional (versus undirectional in static CMOS)*

Voltage Swing for Pass-Transistor Circuits

❑ Example: NMOS Only PT Driving an Inverter

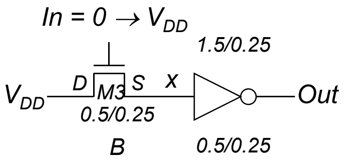


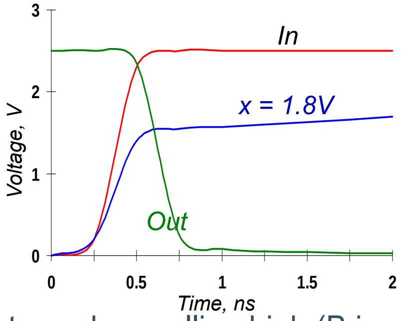
❑ V_x does not pull up to V_{DD}, but $V_{DD} – V_{Tn}$

- ❑ *Threshold voltage drop causes static power consumption (M_1 is turned on by V_x, but V_x may not be high enough to turn off M_2. M_2 may be weakly conducting forming a path from V_{DD} to GND via M_2, M_1)*
- ❑ *Notice V_{Tn} increases of pass transistor due to body effect (V_{SB}) (Bulk of NMOS is connected to Gnd, $V_{SB}(M_3)=V_x≠0$).*

Voltage Swing of PT Driving an Inverter

❑ Due to threshold drop of M_3, V_x goes up at most to 1.8V, instead of V_{DD} (2.5V).

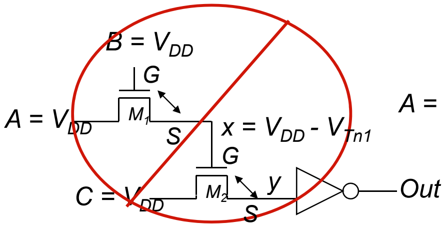


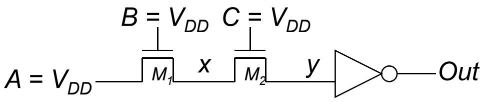


- ❑ Body effect – large V_{SB} at x - when pulling high (B is tied to GND and S charged up close to V_{DD})
- ❑ So the voltage drop is even worse

$$V_x = V_{DD} - (V_{Tn0} + \gamma(\sqrt{(|2\phi_f| + V_x)} - \sqrt{|2\phi_f|}))$$

Cascaded NMOS Only PTs





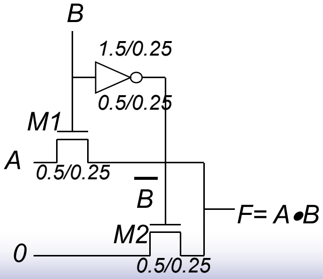
Improper way of cascading pass gates: Swing on y = V_{DD} - V_{Tn1} - V_{Tn2}

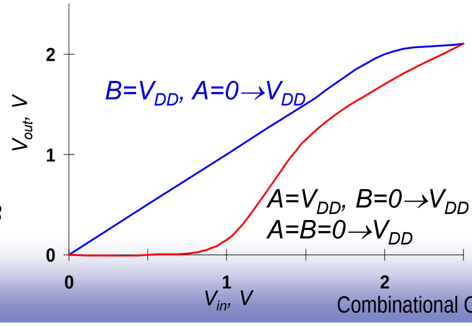
Proper way of cascading pass gates: Swing on y = V_{DD} - V_{Tn1} (only 1 threshold drop)

- Pass-transistor gates cannot be cascaded by connecting the output of a pass gate to the gate input of another pass-transistor (see left).
- Logic on the right suffers from static power dissipation and reduced noise margins

VTC of PT AND Gate

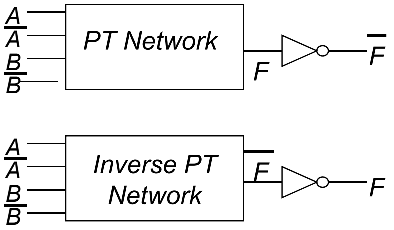
- VTC of pass-transistor gate: data dependent:
  - ✓ $V_B=V_{DD}$, $V_A=0\rightarrow V_{DD}$: M1 on, M2 off, F follows VA till M1 is turned off ($V_A=V_{DD}-V_{Tn}$).
  - ✓ $V_A=V_{DD}$, $V_B=0\rightarrow V_{DD}$: Since inverter $V_M=V_{DD}/2$, when $V_{in}<V_{DD}/2$, M_2 remains on and $V_F\approx 0$. When $V_{in}>V_{DD}/2$, M_2 is off, V_F follows V_B minus a threshold drop (V_B-V_{Tn}).
- Pure PT logic is not regenerative - the signal gradually degrades after passing through a number of PTs (solution: occasional insertion of static CMOS inverter)

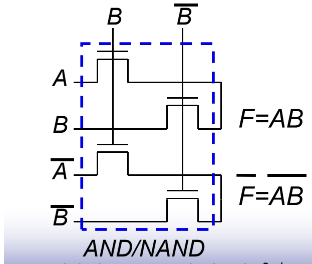




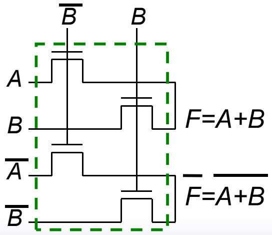
Differential PT Logic (CPL)

❑ Differential pass-transistor logic (called CPL or DPL): accept true and complementary inputs, produce true and complementary outputs. (similar to DCVSL)

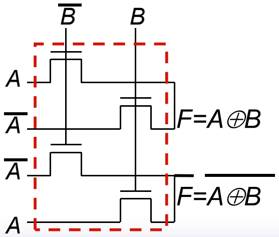




AND/NAND

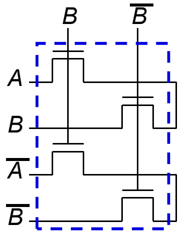


OR/NOR



XOR/XNOR

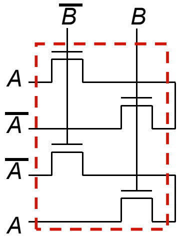
Differential PT Logic (CPL)



F=A·B+B·B’ =A·B+0=A·B

F’=A’·B+B’·B’=A’B+ B’=A’B+B’+A’B’=A’ +B’=(AB)’

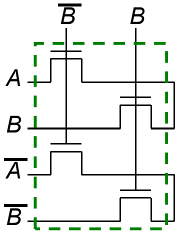
AND/NAND



F=AB’+A’B =A⊕B

F’=A’B’+AB=(A⊕B)’

XOR/XNOR



F=AB’+B·B=AB’+(AB+B) =A(B’+B)+B=A+B

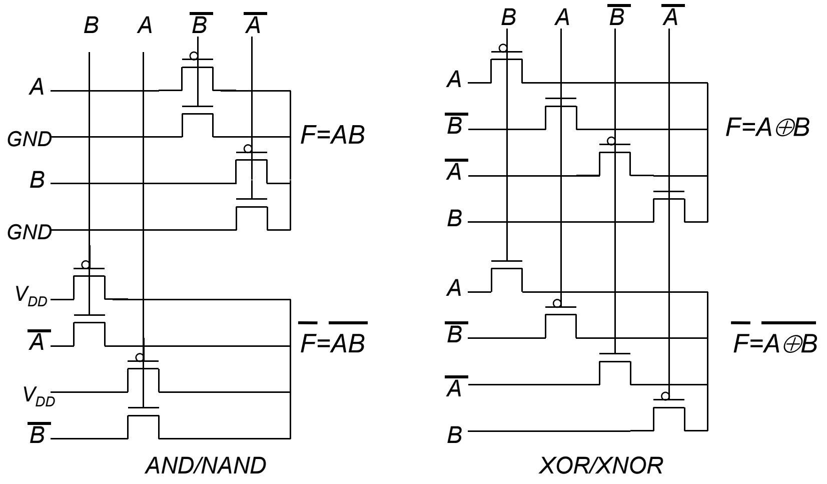
F’=A’B’+B’·B=A’B’=(A+B)’

OR/NOR

CPL Properties

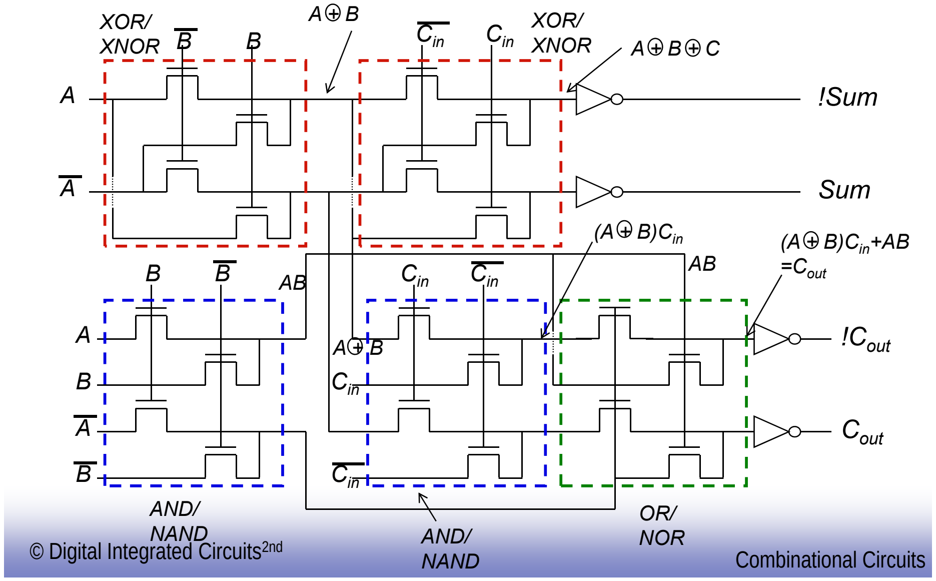
- Differential so complementary data inputs and outputs are always available (so don't need extra inverters)
- Still static, since the output defining nodes are always tied to V_{DD} or GND through a low resistance path
- Design is modular; all gates use the same topology, only the inputs are permuted.
- Simple XOR makes it attractive for structures like adders
- Fast (assuming number of transistors in series is small)
- Additional routing overhead for complementary signals
- Still have static power dissipation problems

Differential TG Logic (DPL): Another Design



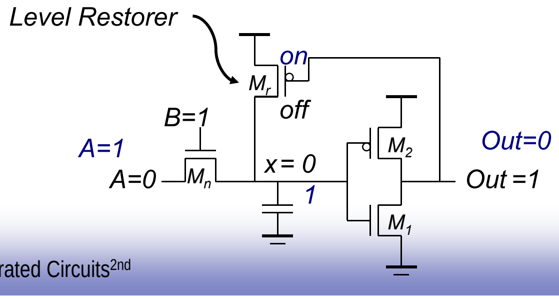
CPL Full Adder

- CPL full adder: $Sum=A\oplus B\oplus C_{in}$, $C_{out}=AB+BC_{in}+Ac_{in.}$ Verify?



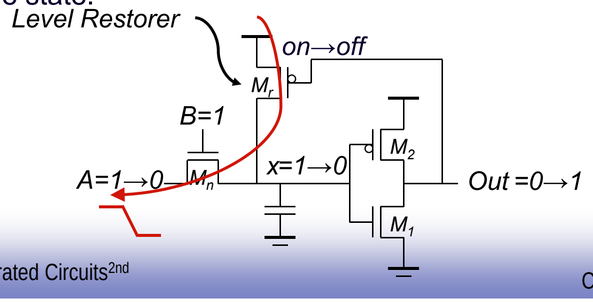
Threshold Drop in PTL - Solution 1: Level Restorer

- Threshold drop in PTL leads to reduced voltage swing, reduced noise margin and static power dissipation.
- Solution 1: use level restorer (a single feedback PMOS)
- ✓ Full swing on x (due to Level Restorer) so no static power consumption by inverter
- ✓ No static backward current path through Level Restorer and PT since Restorer is only active when A is high (If $A=1=V_{DD}$, no current flows from V_{DD} to A; If A=0, M_r is off, again no current flowing from V_{DD} to A).



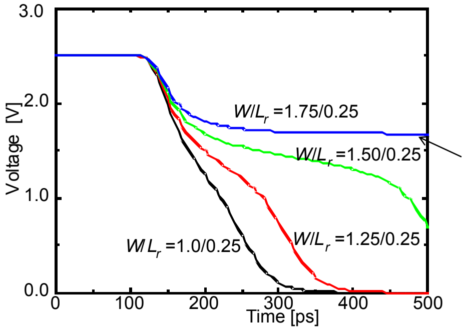
Threshold Drop in PTL - Solution 1: Level Restorer

- For correct operation M_r must be sized correctly (ratioed)
- ✓ When A=1→0, B=1, M_n tries to pull down X, while level restorer M_r pulls X to V_{DD}. M_n and M_r should be properly sized so that X drops below inverter threshold V_M, hence Out changes 0→1, Mr is turned off and X is fully pulled down by A.
- ✓ Transistor size of Mr: $(W/L)_r$ should be small, so that resistance $R(M_r)$ is large to ensure X can be pulled down while M_r and M_n are simultaneously ON. Otherwise, V_x may never be able to switch the inverter, and gate is locked in a single state.



Restorer Sizing: Simulation Result

- Simulation result: For $(W/L)_r$>1.5μm/0.25μm, node X cannot be brought below V_M of inverter, and can't switch the output.



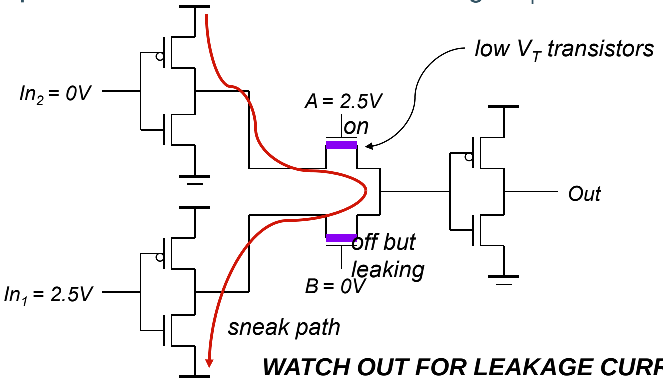
- Upper limit on restorer size
- Pass-transistor pull-down can have several transistors in stack

node x never goes below V_M of inverter so output never switches

- *Restorer has speed and power impacts: increases the capacitance at x, slowing down the gate; increases t_r (but decreases t_f) at "Out".*

Solution 2: Multiple V_T Transistors

- Technology solution: Use (near) zero V_T devices for the NMOS PTs to eliminate *most* of the threshold drop (body effect still in force preventing full swing to V_{DD}). All the devices other than pass transistors still use standard high V_T devices.

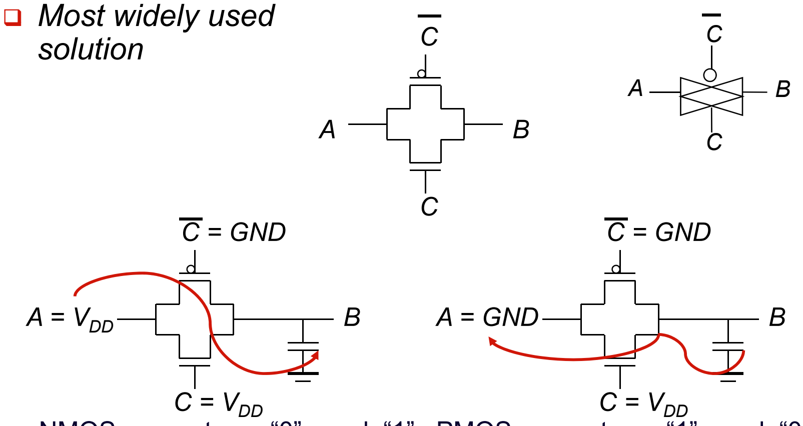


WATCH OUT FOR LEAKAGE CURRENTS

- *Impacts static power consumption due to subthreshold currents flowing through the PTs (even if V_{GS} is below V_T)*

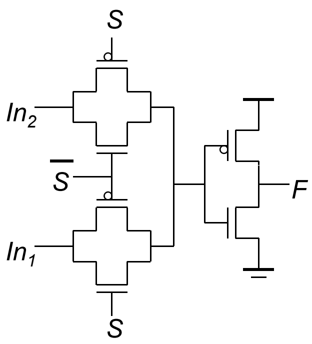
Solution 3: Transmission Gate (TG) Logic

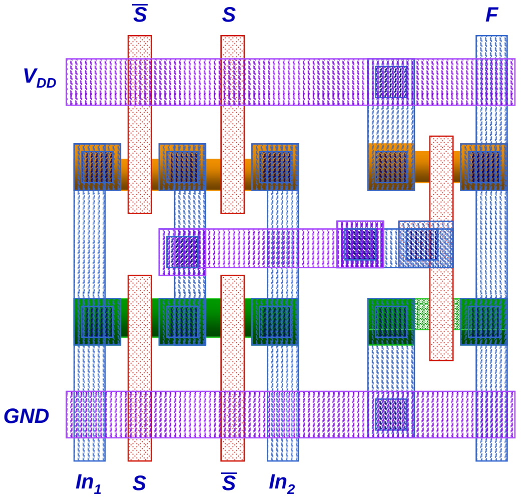
- *Most widely used solution*



- NMOS: pass strong "0", weak "1"; PMOS: pass strong "1", weak "0". Transmission gate (TG): NMOS and PMOS in parallel → pass both strong "0" and strong "1", no threshold drop.
- Full swing *bidirectional* switch controlled by the gate signal C, A = B if C = 1

TG Multiplexer

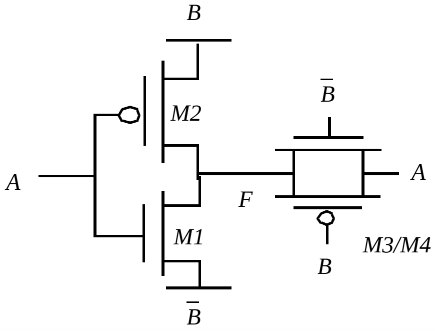




$F = !(In_1 \bullet S + In_2 \bullet \overline{S})$
No. of transistors: 6
(assume S' is directly available).
Static CMOS: 8 transistors

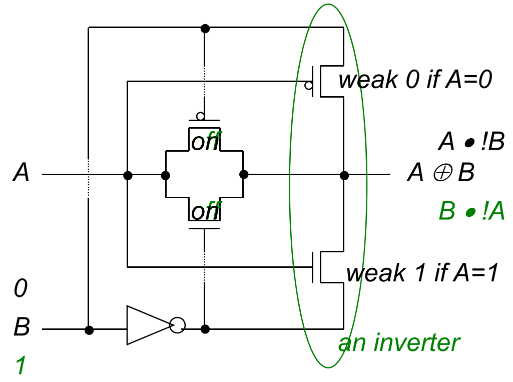
Transmission Gate XOR

- ❑ Transmissoin-gate based XOR circuit: 6 transistors (including inverter used for B')
- ❑ Static CMOS XOR: 12 transistors (4 transistors for inverters used to generate A', B').
- ❑ Working principle:
- ✓ When B=1, B'=0, M2 and M1 act as inverter, M3 and M4 are off, F=A'B.
- ✓ When B=0, B'=1, M2 and M1 are disabled, TG (M3/M4) is on, F=AB'.
- ✓ Thus:
  F=A'B+AB'=A XOR B



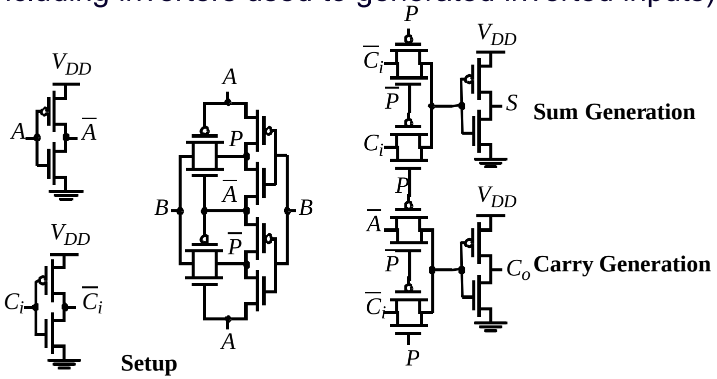
Transmission Gate XOR

- ❑ Another drawing of transmission gate based XOR circuit:



Transmission Gate Full Adder

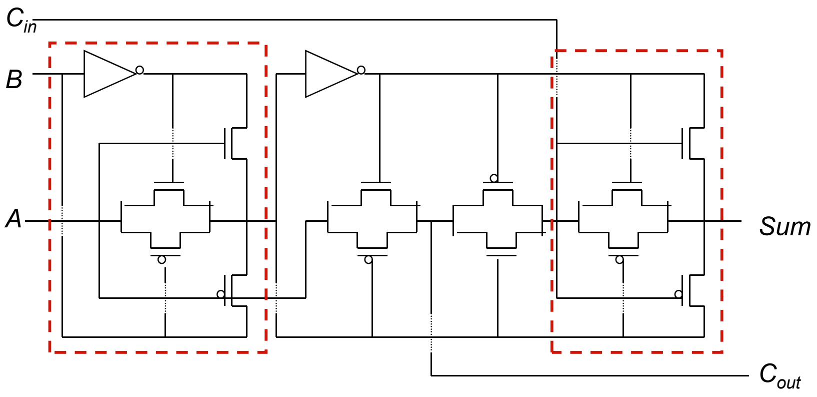
- ❑ Transmission-gate based full adder circuit: 24 transistors (including inverters used to generated inverted inputs)



Similar delays for sum and carry

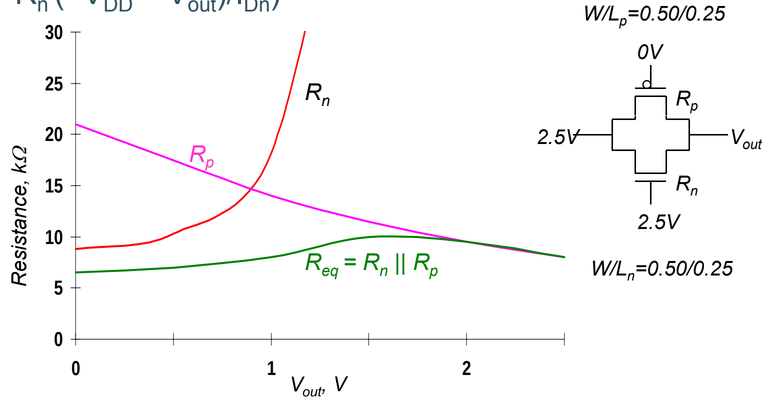
TG Full Adder

- Transmission-gate based full adder circuit: another drawing



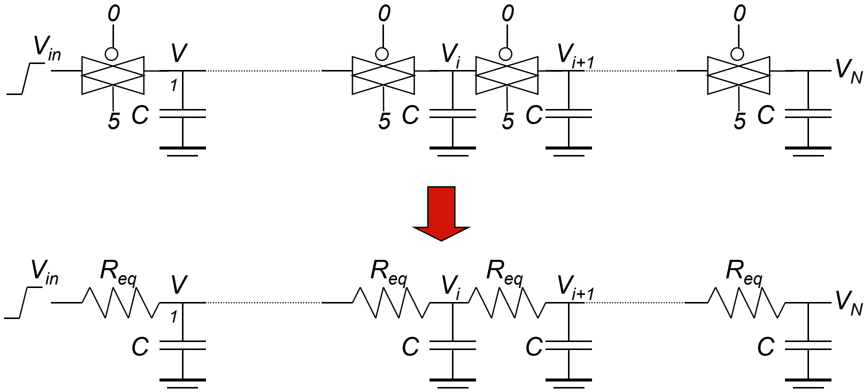
TG Logic Performance

- Effective resistance of the TG is modeled as a parallel connection of R_p (= $(V_{DD} – V_{out})/(-I_{Dp})$) and R_n (=$V_{DD} – V_{out})/I_{Dn}$)



- *So, the assumption that the TG switch has a constant resistive value, R_{eq}, is acceptable*

Delay of a TG Chain

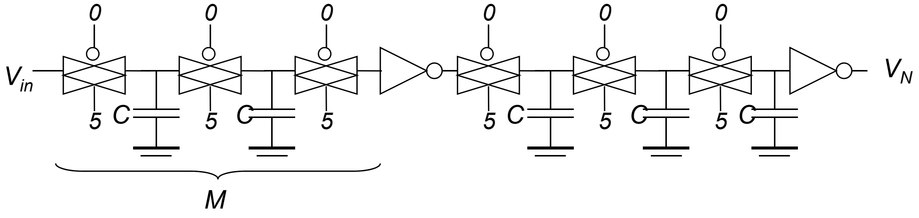


- *Delay of the RC chain (N TG's in series) is*

$$t_p(V_n) = 0.69 \sum_{k=1}^{N} kCR_{eq} = 0.69\, CR_{eq}\, (N(N+1))/2 \approx 0.35\, CR_{eq}N^2$$

TG Delay Optimization

- Can speed it up by inserting buffers every M switches



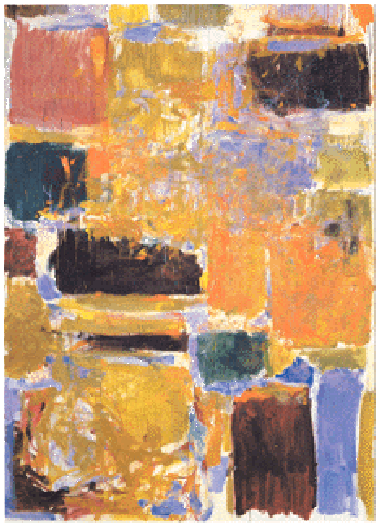
- *Delay of buffered chain (M TG's between buffer)*

$$t_p = 0.69 \lfloor N/M\, CR_{eq}\, (M(M+1))/2 \rfloor + (N/M - 1)\, t_{pbuf}$$

- linear dependence on no. of TGs (N). (without buffer insertion: quadratic dependence (N^2) → slow)

- *Optimal no. of TGs between buffers:*

$$M_{opt} = 1.7 \sqrt{(t_{pbuf}/CR_{eq})} \approx 3 \text{ or } 4 \text{ (typically)}$$



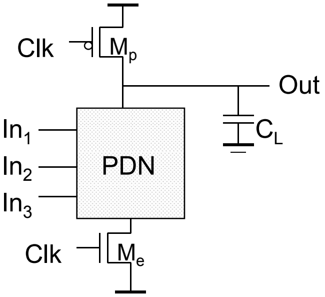
Dynamic Logic

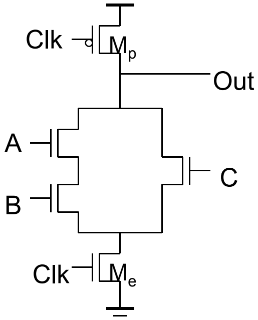
Dynamic CMOS

- ❑ In static circuits at every point in time (except when switching) the output is connected to either GND or V_{DD} via a low resistance path.
  - fan-in of N requires 2N devices (N NMOS + N PMOS)
- ❑ Dynamic circuits rely on the temporary storage of signal values on the capacitance of high impedance nodes.
  - requires only N + 2 transistors (N+1 NMOS + 1 PMOS)
  - takes a sequence of precharge and conditional evaluation phases to realize logic functions
  - no static power consumption (this is better than pseudo-NMOS)

Dynamic Gate: Φn Network

- ❑ Dynamic gate (Φn network): PDN + precharge transistor M_p + evaluation transistor M_e.
- ❑ Construction of Φn network: Starting from static CMOS circuit, replace PUN with a PMOS precharge transistor (M_p), and insert a NMOS evaluation transistor (M_e) between bottom of PDN and Gnd.





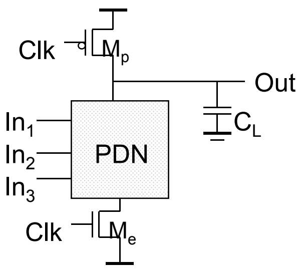
Two phase operation

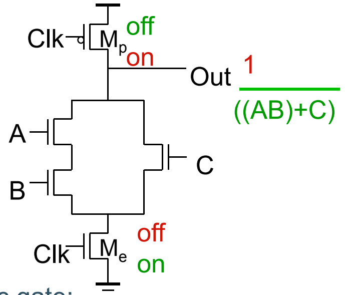
Precharge (CLK = 0)

Evaluate (CLK = 1)

Dynamic Gate: Working Principle

- ❑ Precharge phase: Clk=0, M_p is on, M_e is off, output is prechaged to V_{DD} regardless input values (Out=1).
- ❑ Evaluation phase: Clk=1, M_p is off, M_e is on, output is conditionally discharged based on input values to PDN. For given input pattern,
- ✓ If PDN is off: Out=1, but “Out” is not connected to V_{dd}. It’s floating and relies on charge previously stored in capacitor C_L to hold “Out” to be 1;
- ✓ If PDN is on: Out=0, “Out” is discharged to 0 via PDN.

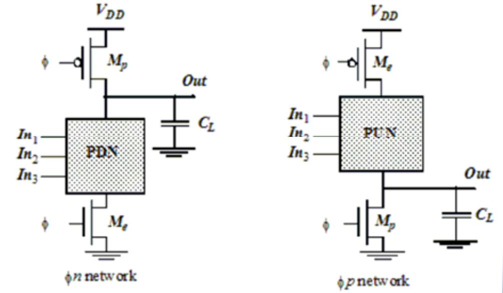




- ❑ Function implemented by above dynamic gate:
  Out=$\overline{CLK}$+$\overline{(A\cdot B+C)}$·CLK   If clk=0, Out=1; If clk=1, Out=(A·B+C)’

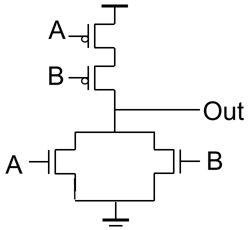
Dynamic Logic: Φn/Φp Network

- ❑ Φn network: PDN + precharge M_p (PMOS) + evaluation M_e (NMOS)
- ✓ Φ=0, precharge phase, Mp on, Me off, Out is pre-charged to 1: Out=1
- ✓ Φ=1, evaluation phase, Mp off, Me on, Out depends on PDN
- ❑ Φp network: PUN + predischarge M_p (NMOS) + evaluation M_e (PMOS)
- ✓ Φ=1, predischarge phase, M_p on, M_e off, Out is predischarged to 0: Out=0
- ✓ Φ=0, evaluation phase, M_p off, M_e on, Out depends on PUN
  - If PUN is off, Out remains 0, but is floating,
  - If PUN is on, Out is connected to V_{dd} via PUN: Out=1.
- ✓ Φp network is less popular because PMOS is slower than NMOS.

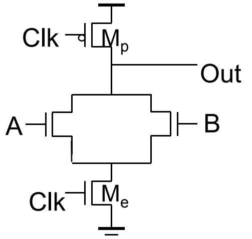


Dynamic Gate: Working Principle

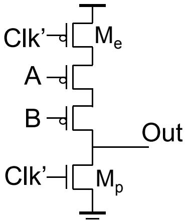
- ❑ Question: Implement 2-input NOR gate with static CMOS, Φn CMOS, Φp CMOS separately: Out=(A+B)'? Rules:
  - ✓ Φn network: Starting from static CMOS circuit, replace PUN with a PMOS precharge transistor (M_p), and insert a NMOS evaluation transistor (M_e) between bottom of PDN and Gnd.
  - ✓ Φp network: Starting from static CMOS circuit, replace PDN with a NMOS predischarge transistor (M_p), and insert a PMOS evaluation transistor (M_e) between top of PUN and Vdd.



Static CMOS NOR2



Φn CMOS NOR2

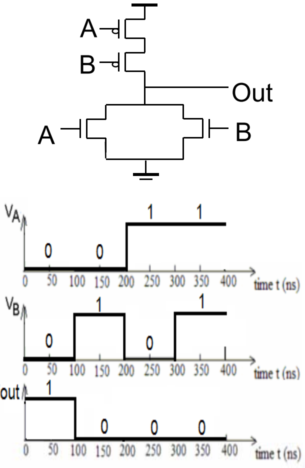


Φp CMOS NOR2

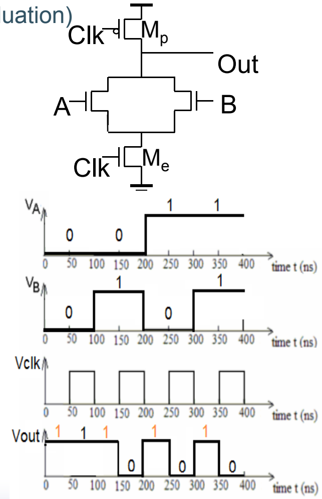
Dynamic Gate: Working Principle

- ❑ Question: Compare static CMOS and Φn NOR2 gate, if AB=00→01→10→11, sketch waveforms of Out?

(Within each clock cycle: precharge→evaluation)



Static CMOS NOR2

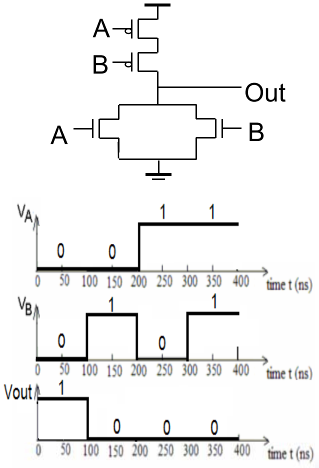


Φn CMOS NOR2

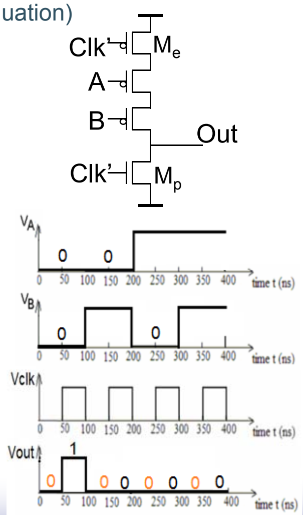
Dynamic Gate: Working Principle

- ❑ Question: Compare static CMOS and Φp NOR2 gates, if AB=00→01→10→11, sketch waveforms of Out?

(Within each clock cycle: precharge→evaluation)



Static CMOS NOR2



Φp CMOS NOR2

Φn Network: Conditions on Output

- ❑ Dynamic logic works in two phases: precharge and evaluation.
- ✓ In precharge phase, output is always 1 regardless input patterns.
- ✓ In evaluation phase, output is the valid function of inputs. If output=1, output is floating (high impedance state) and it relies on the charge previously stored in capacitor to hold output to be "1". (Instead, static CMOS always has a low resistance path between output and one of the power rails.)
- ❑ Once the output of a dynamic Φn gate is discharged, it cannot be charged again until the next precharge operation.
- ❑ Output can make at most one transition (1→0) during evaluation.
- ❑ Output is valid only in evaluation phase. In precharge phase, output is always "1" and cannot be utilized.
- ❑ Digital system can be designed in such a way that precharge time coincides with other system functions. E.g. the precharge of arithmetic unit in a microprocessor could coincide with instruction decoding.

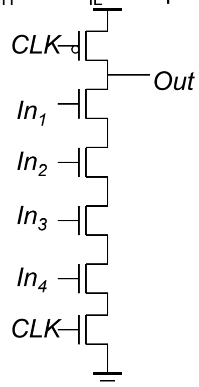
Properties of Dynamic Gates

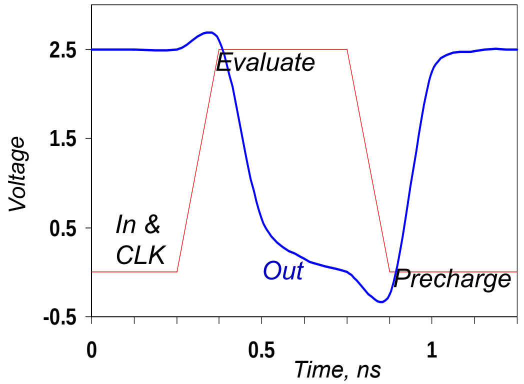
- ❑ Logic function is implemented by the PDN only
  - ▪ number of transistors is N + 2 (versus 2N for static complementary CMOS)
  - ▪ should be smaller in area than static complementary CMOS
- ❑ Full swing outputs (V_{OL} = GND and $V_{OH} = V_{DD}$)
- ❑ Nonratioed - sizing of the devices is not important for proper functioning (sizing only affects performance)
- ❑ Faster switching speeds
  - ▪ reduced load capacitance due to lower number of transistors per gate (C_{int}) so a reduced logical effort
  - ▪ reduced load capacitance due to smaller fan-out (C_{ext})
  - ▪ Ignoring the influence of precharge time on the switching speed of the gate, $t_{pLH} = 0$ (why?) but the presence of the evaluation transistor slows down the t_{pHL}
  - ▪ no I_{sc}, so all the current provided by PDN goes into discharging C_L

Dynamic Behavior

❑ Dynamic behavior of dynamic 4-input NAND:

✓ PDN starts to work as soon as the input signals exceed V_{Tn}, so set V_M, V_{IH} and V_{IL} all equal to V_{Tn}: low noise margin (NM_L)

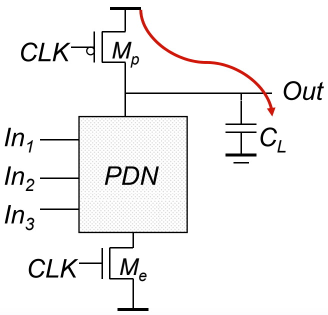




#Trns	V_{OH}	V_{OL}	V_M	NM_H	NM_L	t_{pHL}	t_{pLH}	t_p
6	2.5V	0V	V_{Tn}	2.5-V_{Tn}	V_{Tn}	110ps	0ns	83ps

Power Consumption of Dynamic Gate

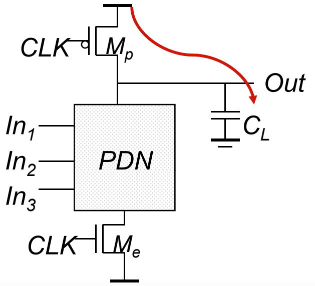
- ❑ Dynamic logic seems to consume less power because:
- ✓ Physical capacitance is lower (it uses fewer transistors)
- ✓ Load capacitance is small (load for each fanout is 1 instead of 2 transistors)
- ✓ No glitching power: Output at most have 1 transition per cycle.
- ✓ No short-circuit power: M_p is off when PDN is evaluating.



Power only dissipated when previous Out = 0

Power Consumption of Dynamic Gate

- However, dynamic logic generally consumes more power than static CMOS. Why?
- ✓ Its clock power can be significant: clock has a guaranteed transition on every clock cycle.
- ✓ Short-circuit power may exist when leakage-combating devices are added.
- ✓ It has higher switching activity ($\alpha_{0\to1}$) due to periodic precharge and discharge operation. (switching power: $P_{sw}=\alpha_{0\to1}C_LV_{dd}^2f$)



Power only dissipated when previous Out = 0

Dynamic Power Consumption: Data Dependent

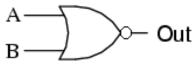
- 0→1 switching probability of static CMOS gate:
  $\alpha_{0\to1}(static)=p_0p_1=p_0(1-p_0)$
- Dynamic Φn gate: output makes 0→1 transition during precharge only if output was 0 in preceding evaluation phase. Thus $\alpha_{0\to1}$ only depends on signal probability p_0:
  $\alpha_{0\to1}(dynamic)=p_0 \geq p_0(1-p_0) =\alpha_{0\to1}(static)$
- ❑ Example: Static CMOS and Φn dynamic 2-input NOR Gates

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Assume signal probabilities

$P_{A=1} = 1/2$

$P_{B=1} = 1/2$



Transition probability of Φn NOR2:

$\alpha_{0\to1}(dynamic) = P_{out=0} = 3/4$

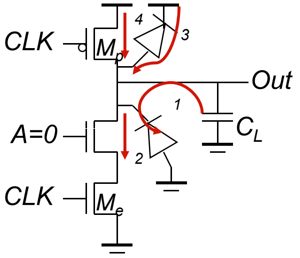
Static CMOS NOR2 gate:

$\alpha_{0\to1}(static)=p_0p_1=(3/4)\times(1/4)=3/16$

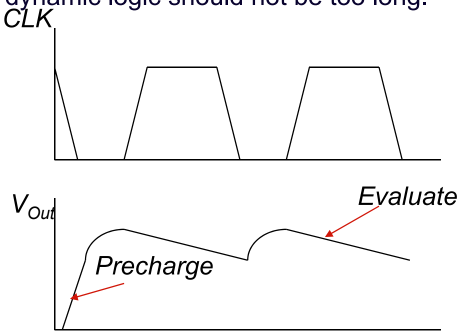
- ❑ Switching activity can be higher in dynamic gates!
  Dynamic gates: $\alpha_{0\to1} = P_{out=0}$
- ❑ Question: Find $\alpha_{0\to1}$ for dynamic Φp NOR2 gate? ($\alpha_{0\to1}=P_{out=1}$)

Issues in Dynamic Design 1: Charge Leakage

- Dynamic gate relies on dynamic storage of output value on a capacitor. In evaluation, if PDN is off, Out=1. It relies on charge previously stored in capacitor C_L during precharge to maintain it at V_{DD}. However, due to leakage currents, the charge gradually leaks away, V_{out} drops gradually, eventually resulting in malfunctioning of the gate. → clock period of dynamic logic should not be too long.



Leakage sources

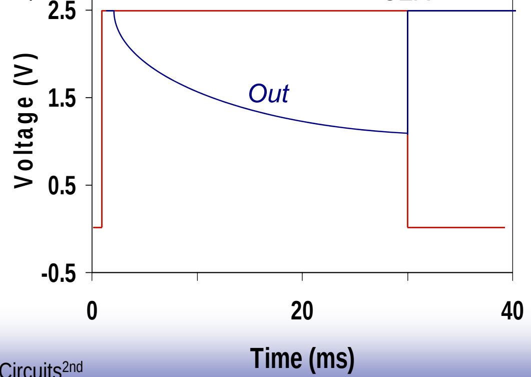


Dominant component is subthreshold current

- ❑ *Dynamic logic requires minimum clock rate of a few kHz.*

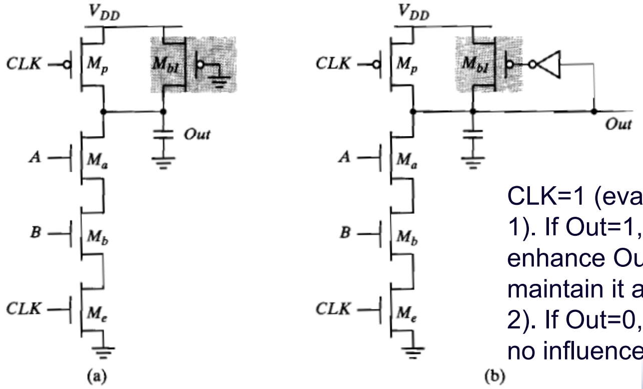
Impact of Charge Leakage

- Output settles to an intermediate voltage determined by a resistive divider of the pull-up and pull-down networks
  - Once the output drops below the switching threshold of the fan-out logic gate, the output is interpreted as a low voltage. (Question: Does static CMOS have this issue?)



Dynamic Logic: Charge Leakage Solution

- ❑ Solution to charge leakage: add a bleeder transistor between Vdd and output.
- ✓ Using grounded PMOS as bleeder: ratioed, not good.
- ✓ Bleeder in feedback configuration: no static power dissipation.

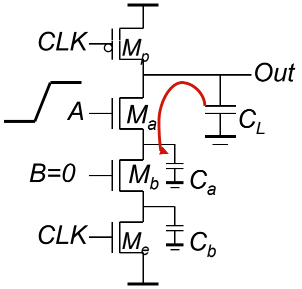


Static bleeders compensate for the charge leakage.

CLK=1 (evaluation):
1). If Out=1, M_{bl} is on, enhance Out1 and maintain it at V_{dd};
2). If Out=0, M_{bl} is off, no influence.

Issues in Dynamic Design 2: Charge Sharing

❑ Assume AB=00 in precharge, and C_a is discharged. Now in evaluation, B=0, but A=0→1, M_a is on. Out should remain 1, but the charge stored originally on C_L is redistributed over C_L and C_a. This causes a drop in V_{out}, which cannot be recovered due to dynamic nature of circuit. → charge sharing!

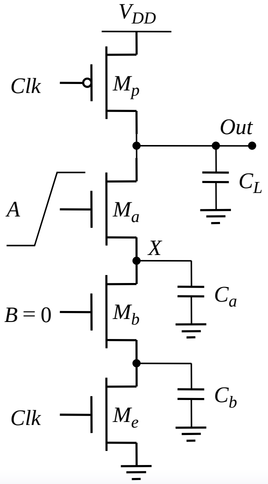


Charge stored originally on C_L is redistributed (shared) over C_L and C_a leading to static power consumption by downstream gates and possible circuit malfunction.

When $\Delta V_{out} = - V_{DD} (C_a / (C_a + C_L))$ the drop in V_{out} is large enough to be below the switching threshold of the gate it drives, this may cause a malfunction.

Charge Sharing in Dynamic Gate

❑ Due to charge sharing, what's the final value of V_{out}?



case 1) if $\Delta V_{out} < V_{Tn}$
(this happens when C_a is small, $(C_a/C_L)<V_{tn}/(V_{DD}-V_{Tn})$, V_x is charged up to $(V_{DD}-V_{tn})$, and M_a is off.

$$C_L V_{DD} = C_L V_{out}(t) + C_a(V_{DD} - V_{Tn}(V_X))$$

or

$$\Delta V_{out} = V_{out}(t) - V_{DD} = -\frac{C_a}{C_L}(V_{DD} - V_{Tn}(V_X))$$

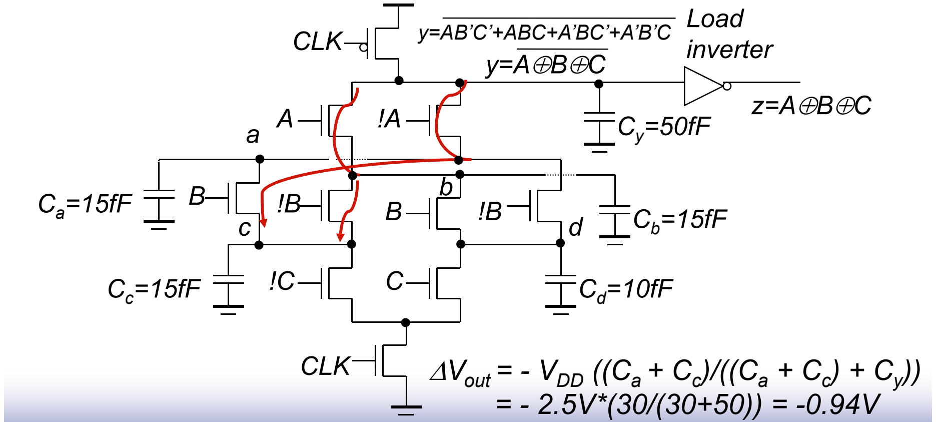
case 2) if $\Delta V_{out} > V_{Tn}$ *(this happens when $(C_a/C_L)>V_{tn}/(V_{DD}-V_{Tn})$, M_a is always on,*

$$\Delta V_{out} = -V_{DD}\left(\frac{C_a}{C_a + C_L}\right)$$

charge on C_L is shared between C_a and C_L.

Charge Sharing Example

❑ *3-input EXOR gate: y=A⊕ B⊕ C. What is the worst case voltage drop on y? (Assume all inputs are low during precharge and that all internal nodes are initially at 0V.) Since C_c>C_d, C_a=C_b, worst cases: charge on C_y is shared with C_a+C_c or C_b+C_c. This happens when C' is off, but A,B' or A',B are on, i.e. for patterns ABC=011 or 101.*

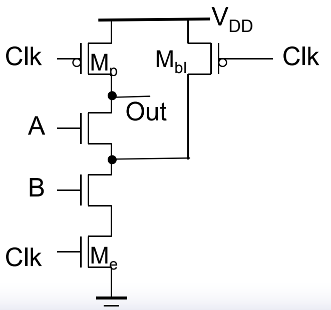


$\Delta V_{out} = - V_{DD} ((C_a + C_c)/((C_a + C_c) + C_y))$
$= - 2.5V*(30/(30+50)) = -0.94V$

Solution to Charge Redistribution

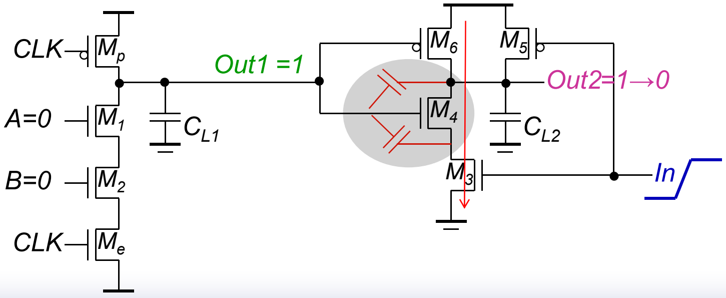
❑ Solution to charge redistribution: Precharge critical internal nodes using a clock-driven transistor (at the cost of increased area and power)

❑ Since internal nodes are charged to V_{DD} during precharge, charge sharing does not occur.



Issues in Dynamic Design 3: Backgate Coupling

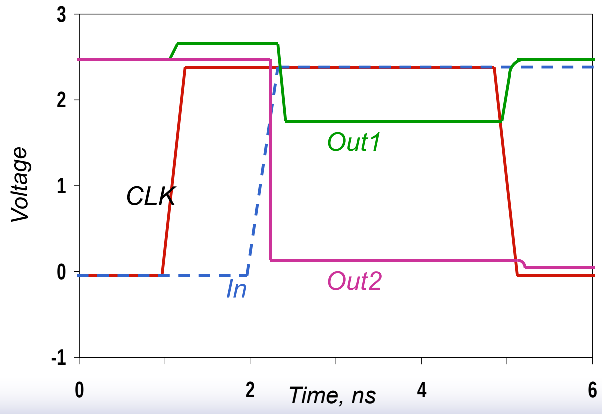
- Output of dynamic logic is susceptible to crosstalk due to 1) high impedance of the output node, 2) capacitive coupling.
- Backgate (output-to-input) coupling: In following circuit, Out2=(In·Out1)'. Out2 capacitively couples with Out1 through the gate-source and gate-drain capacitances of M4.
- ✓ If Out1=1, In=0→1, Out2=1→0, this output transition capacitively couples to Out1, Out1 drops significantly and cause M6 to be on. → leakage current via M6-M4-M3, Out2 is not fully pulled down to 0V.



Dynamic NAND Static NAND

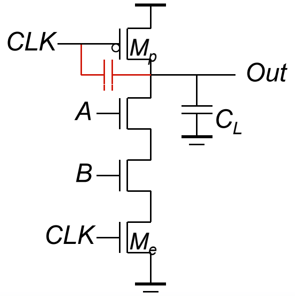
Backgate Coupling Effect

- Simulation result: Capacitive coupling means Out1 drops significantly so Out2 doesn't go all the way to ground



Issues in Dynamic Design 4: Clock Feedthrough

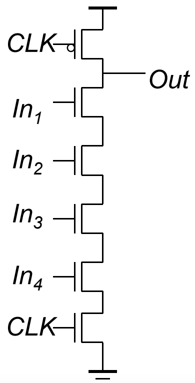
- Clock feedthrough: a special case of capacitive coupling between the clock input of the precharge transistor and the dynamic output node.
- When CLK=0→1, rising transition of CLK is capacitively coupled to Out and causes the output (dynamic node) to rise above VDD. Similarly, when CLK=1→0, falling transition of CLK is coupled to Out and causes it to temporarily fall below 0V.

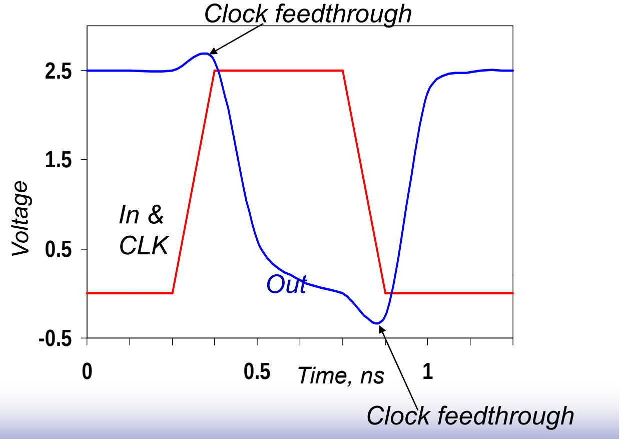


Coupling between Out and CLK input of the precharge device due to the gate-drain capacitance. So voltage of Out can rise above V_{DD}. The fast rising (and falling edges) of the clock couple to Out.

Clock Feedthrough

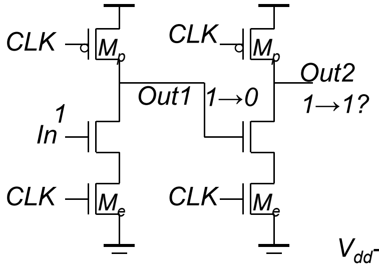
- Clock feedthrough may cause normally reverse-biased junction diode to become forward-biased. → electron injection to substrate and collected by a nearby high-impedance node in "1" state → eventually may cause malfunction!

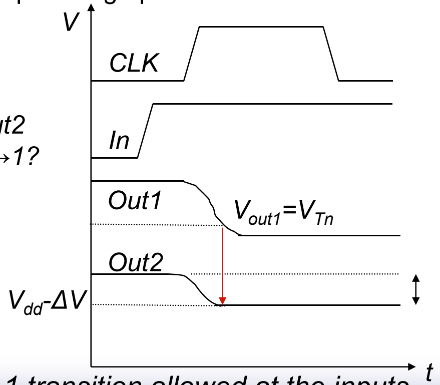




Cascading Dynamic Gates

- Straightforward cascading of dynamic gates does not work!
- Ex: cascading 2 dynamic Φn inverters. When CLK=0→1 (evaluation), In=1, Out2=(In')' should remain 1. But Out1=1→0 with propagation delay. As long as Out1>V_{Tn}, it turns on NMOS in next inverter, Out2 is mis-discharged till V_{out1}<V_{Tn}. Once Out2 is mis-discharged, it cannot recover back to V_{DD} till next precharge phase.

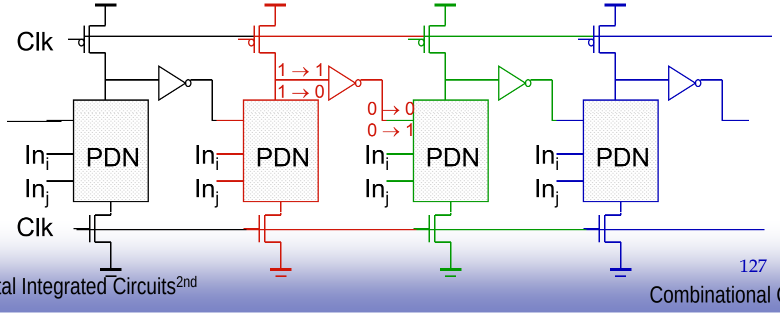




Only a single 0 → 1 transition allowed at the inputs during the evaluation period!

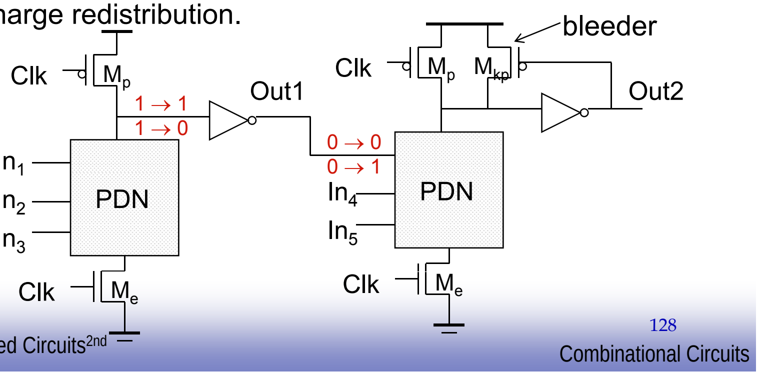
Cascading Dynamic Gates #1: Domino Logic

- Cascading problem is because outputs of each gate (thus the inputs to next stages) are precharged to 1. This may turn on NMOS transistors in PDN of next stage and cause inadvertent discharge in evaluation.
- Solution: 1). Domino logic: add inverter to set the inputs to next stage to 0 instead of 1 during precharge. → inverter outputs turn off the transistors in PDN of next stage after precharge. Transistors are turned on only when needed - and at most, once per cycle. → no inadvertent discharging in evaluation.



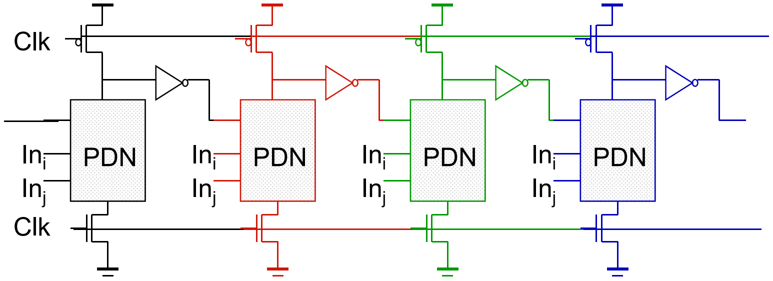
Domino Logic

- Domino logic: N-type dynamic logic block followed by inverter.
- In precharge, output of PDN is precharged to "1", Out1="0". NMOS transistor in next stage connected to Out1 is off.
- In evaluation, NMOS transistors in next Φn stage either remains off, or turn on as needed: No 1→0 transition, no inadvertent discharging!
- Inverter can also be used to drive a bleeder to combat leakage and charge redistribution.



Why Called Domino?

- ❑ Why called Domino? For chain of domino gates: In precharge, output of all stages are set to 0 simultaneously (parallel process). In evaluation, output of 1st domino block either stays at 0 or makes 0→1 transition, affecting the 2nd gate. This effect ripples through the whole chain one after the other, like a line of falling dominoes (serial process). – hence the name.

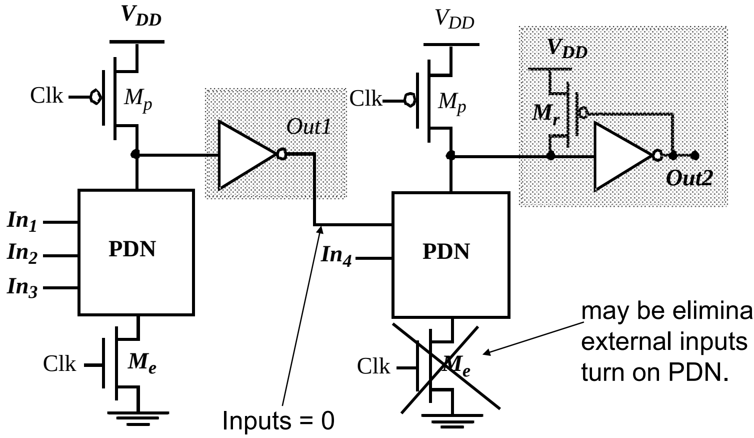


Like falling dominos!

Properties of Domino Logic

- ❑ Only non-inverting logic can be implemented, fixes include
  - can reorganize the logic using Boolean transformations
  - use differential logic (dual rail)
  - use np-CMOS (zipper)
- ❑ Very high speed
  - t_{pHL} = 0 (only t_{pLH} exists)
  - static inverter can be optimized to match fan-out (separation of fan-in and fan-out capacitances)
  - Input capacitance reduced – smaller logical effort

Designing with Domino Logic

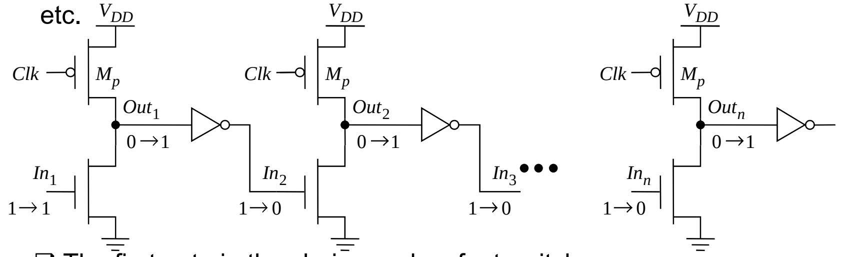


may be eliminated if external inputs do not turn on PDN.

Inputs = 0 during precharge

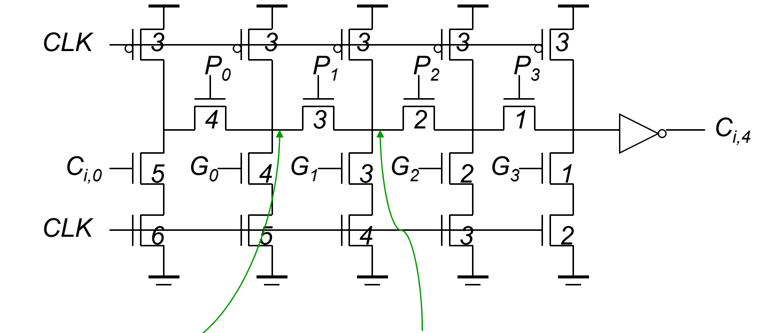
Footless Domino

- ❑ Footless Domino: Domino logic with foot transistor (evaluation transistor M_e) removed.
- ❑ However, removing M_e extends the precharge cycle – precharge now has to ripple through the logic network as well.
- ❑ Ex: Footless Domino inverter chain, in evaluation, In_1=1, then Out_i=0, In_i=1. When clk=1→0, In_1=1, In_2=1→0 after 2 gate delays, before that, Out_2 cannot be precharged because In_2 is still on. Similarly, 3rd gate has to wait until 2nd gate precharges before it can start precharging, etc.



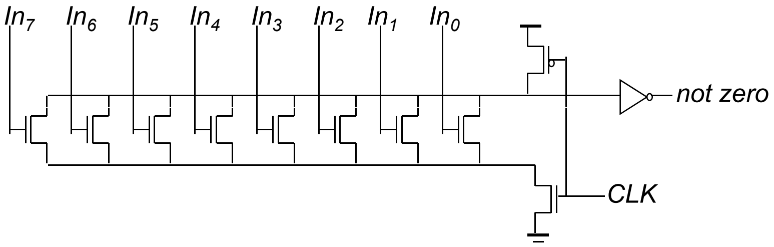
- ❑ The first gate in the chain needs a foot switch
- ❑ Precharge is rippling (not a parallel process anymore) – short-circuit current. A solution is to delay the clock for each stage

Domino Manchester Carry Chain

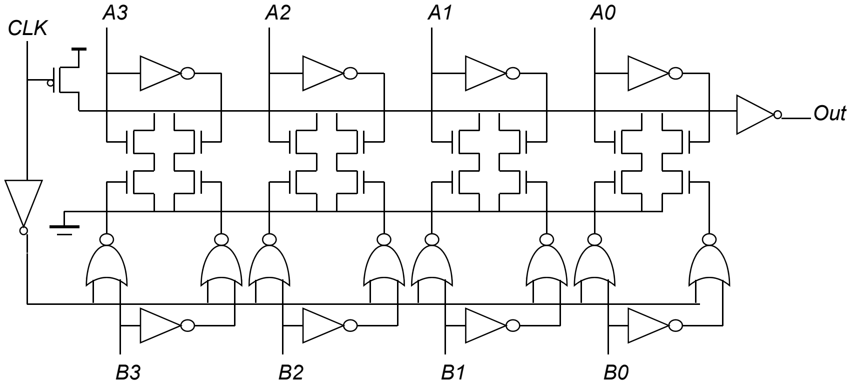


$!(G_0 + P_0 C_{i,0})$ $!(G_1 + P_1G_0 + P_1P_0 C_{i,0})$

Domino Zero Detector

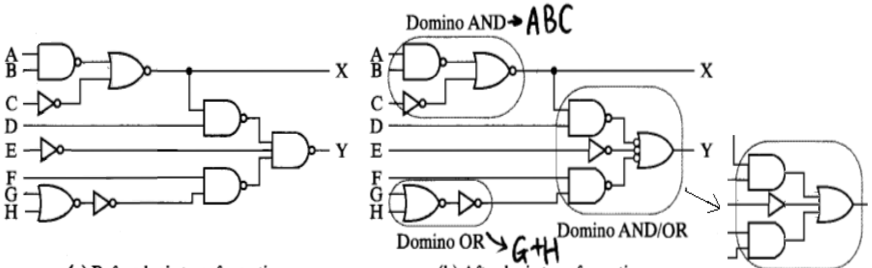


Domino Comparator



Dealing with Noninverting Property of Domino: #1. De Morgan's law

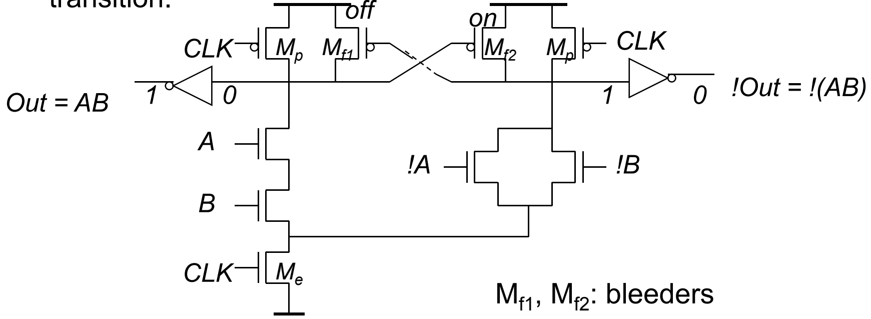
- Domino logic: only non-inverting logic can be implemented. This has limited usage of pure domino logic.
- Solution: 1). Reorganizing the logic using simple Boolean transforms (e.g. De Morgan's law). But it may not be always possible.



(a) Before logic transformation (b) After logic transformation

Dealing with non-inverting property of Domino: #2. Differential (Dual Rail) Domino

- Solution 2: Differential (dual-rail) Domino. Similar to DCVSL, can be used to implement any arbitrary function.
- ✓ More power due to guaranteed transition every clock cycle regardless input patterns - either out or out' must make a 0→1 transition.



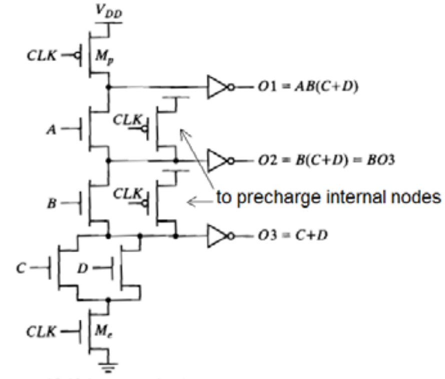
M_{f1}, M_{f2}: bleeders

AND/NAND differential Domino: Solves the problem of non-inverting logic

Due to its high-performance, differential domino is very popular and is used in several commercial microprocessors!

Variations: Multiple-output Domino

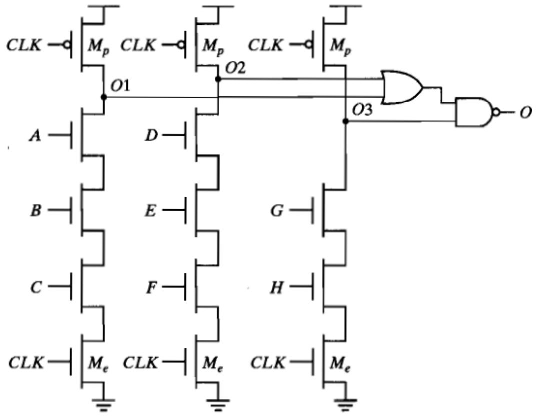
- Multiple-output domino logic: to reduce transistor count. It exploits the fact that certain outputs are subsets of other outputs to generate a number of logic functions in a single gate.
- E.g. In following circuit, O3=C+D is used in all 3 outputs, so it is implemented at the bottom of PDN. O2 and O1 reuse O3 without re-implementing it.



Multiple-output domino.

Variations: Compound Domino

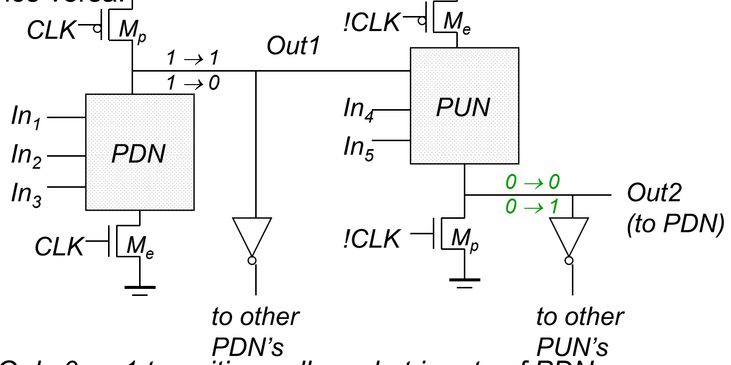
- Compound domino: to reduced transistor count. It combines outputs of multiple dynamic gates with a complex static CMOS gate.
- Ex: In following circuit, O1=(ABC)', O2=(DEF)', O3=(GH)', Final output: O=[(O1+O2)O3]'=ABCDEF+GH → reduced fan-in, faster speed.



Compound domino logic uses complex static gates at the output of the dynamic gates.

Cascading Dynamic Gates #2: np-CMOS (Zipper)

- Cascading dynamic gates solution 2: np-CMOS – exploits the duality between Φn and Φp networks to eliminate cascading problem. If Φn gates are controlled by CLK, Φp gates are controlled by CLK', then Φn gates can directly drive Φp gates, and vice versa.

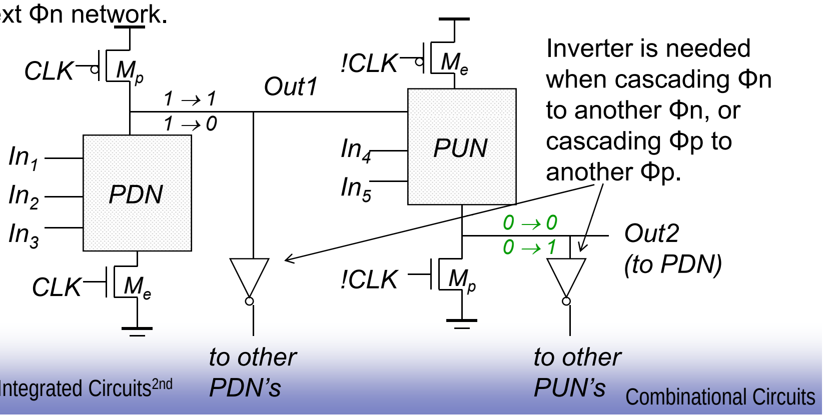


Only 0 → 1 transitions allowed at inputs of PDN

Only 1 → 0 transitions allowed at inputs of PUN

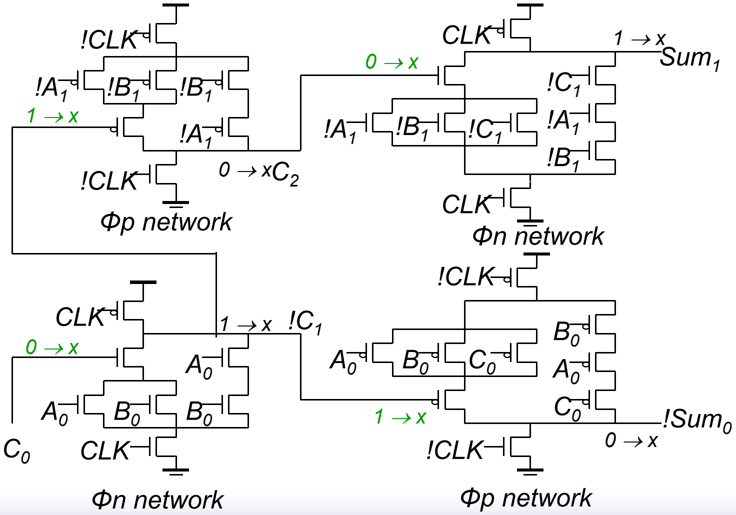
Cascading Dynamic Gates: np-CMOS (Zipper)

- np-CMOS: Cascade Φn and Φp networks alternately without extra inverters
  - ✓ Precharge: CLK=0, Out1 from Φn is precharged to Vdd, it turns off PMOS in next Φp network; Out2 from Φp is predischarged to Gnd, it turns off NMOS in next Φn network.
  - ✓ Evaluation: CLK=1, PMOS in next Φp network either remains off or turned on as needed → no inadvertent charging. Same for NMOS in next Φn network.



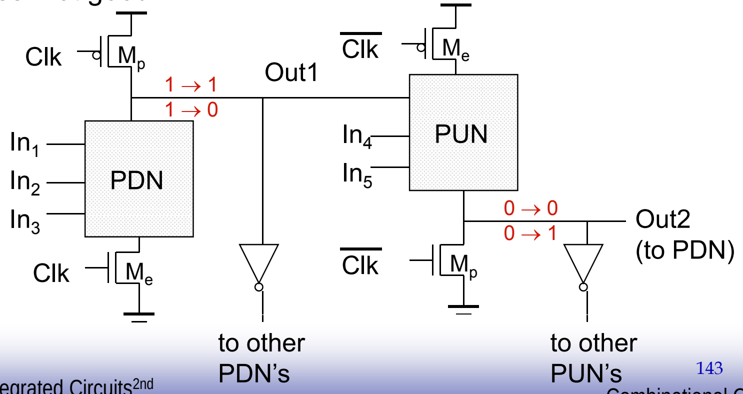
Inverter is needed when cascading Φn to another Φn, or cascading Φp to another Φp.

np-CMOS Adder Circuit



np-CMOS (Zipper)

- Limitation of np-CMOS:
  - ✓ P-tree blocks are slower than n-tree modules (PMOS is slower than NMOS given the same size). Equalizing propagation delay requires extra area.
  - ✓ Lack of buffers requires dynamic nodes are routed between gates: not good.



How to Choose a Logic Style

- Must consider ease of design, robustness (noise immunity), area, speed, power, system clocking requirements, fan-out, functionality, ease of testing

4-input NAND

Style	# Trans	Difficulty	Ratioed?	Delay	Power
Static CMOS	8	1	no	3	1
CPL*	12 + 2	2	no	4	3
domino	6 + 2	4	no	2	2 + clk
DCVSL*	10	3	yes	1	4

** Dual Rail*

- *Current trend is towards an increased use of complementary static CMOS: design support through DA tools, robust, more amenable to voltage scaling.*