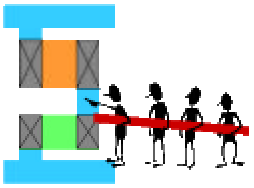
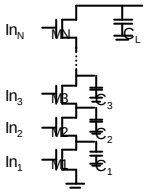
COMBINATIONAL LOGIC DYNAMICS



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

Fast Complex Gates: Design Technique 1

- Transistor sizing
  - » as long as fan-out capacitance dominates
- Progressive sizing



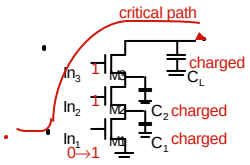
Distributed RC line

M1 > M2 > M3 > … > MN
(the fet closest to the output is the smallest)

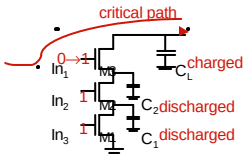
Can reduce delay by more than 20%; decreasing gains as technology shrinks

Fast Complex Gates: Design Technique 2

- Transistor ordering



delay determined by time to discharge C_L, C_1 and C_2

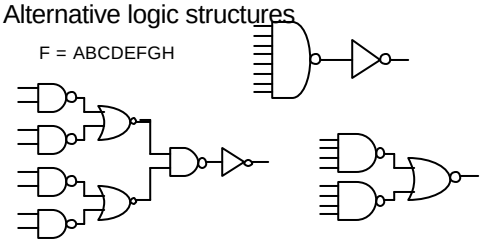


delay determined by time to discharge C_L

Fast Complex Gates: Design Technique 3

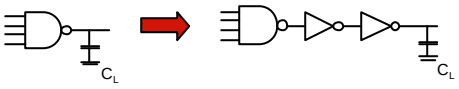
- Alternative logic structures

F = ABCDEFGH



Fast Complex Gates: Design Technique 4

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



Fast Complex Gates: Design Technique 5

- Reducing the voltage swing

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

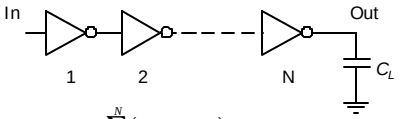
$$= 0.69\ (3/4\ (C_L V_{swing})/ I_{DSATn})$$

  - » linear reduction in delay
  - » also reduces power consumption
- But the following gate is much slower!
- Or requires use of "sense amplifiers" on the receiving end to restore the signal level (memory design)

Sizing Logic Paths 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?

Buffer Example



$$Delay = \sum_{i=1}^{N} (p_i + g_i \cdot f_i) \quad \text{(in units of } \tau_{inv})$$

For given N: $C_{i+1}/C_i = C_i/C_{i-1}$
To find N: $C_{i+1}/C_i \sim 4$
How to generalize this to any logic path?

Logical Effort

$$Delay = k \cdot R_{unit} C_{unit} \left(1 + \frac{C_L}{g C_{in}}\right)$$
$$= t\,(p + g \cdot f)$$

p – intrinsic delay ($3kR_{unit}C_{unit}\gamma$) - gate parameter ≠ f(W)
g – logical effort ($kR_{unit}C_{unit}$) – gate parameter ≠ f(W)
f – effective fanout

Normalize everything to an inverter:
$g_{inv} = 1$, $p_{inv} = 1$

Divide everything by τ_{inv}
(everything is measured in unit delays τ_{inv})
Assume $\gamma = 1$.

Delay in a Logic Gate

Gate delay:

$$d = h + p$$

effort delay **intrinsic 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

Logical Effort

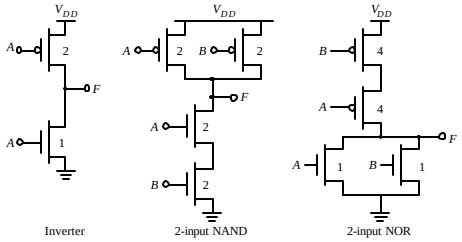
- Inverter has the smallest logical effort and intrinsic delay of all static CMOS gates
- Logical effort of a gate presents the ratio of its input capacitance to the inverter capacitance when sized to deliver the same current
- Logical effort increases with the gate complexity

Intrinsic Delay

- Inverter has the smallest intrinsic delay and of all static CMOS gates
- Intrinsic delay of a gate presents the ratio of its output capacitance to the inverter output capacitance when sized to deliver the same current
- Intrinsic delay increases with the gate complexity

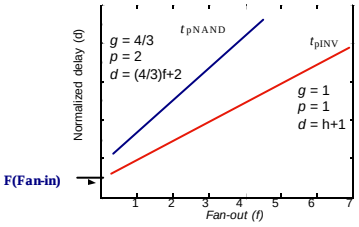
Logical Effort

Logical effort is the ratio of input capacitance of a gate to the input capacitance of an inverter with the same output current



Inverter $g = p = 1$ | 2-input NAND $g = 4/3$, $p = 2$ | 2-input NOR $g = 5/3$, $p = 2$

Logical Effort of Gates

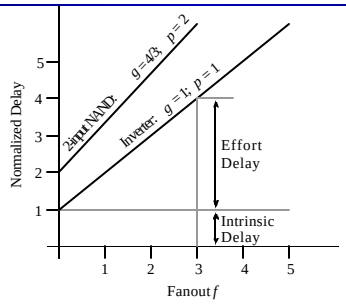


h = g f

d = h+p

g – logical effort
f - effective fan out
p – intrinsic delay

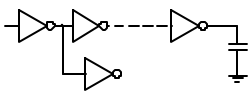
Logical Effort of Gates



Add Branching Effort

Branching effort: $C_{off\text{-}path}$ is the branch capacitance

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



Multistage Networks

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

Stage effort: $h_i = gf_i$

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

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

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

Path effort: $H = GFB$

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

Optimum Effort per Stage

When each stage bears the same effort:

$$h^N = H$$

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

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

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

Minimum path delay

$$\hat{D} = \sum (g_i f_i + p_i) = NH^{1/N} + P$$

Optimal Number of Stages

For a given load,
and given input capacitance of the first gate
Find optimal number of stages and optimal sizing

$$D = NH^{1/N} + Np_{inv}$$

$$\frac{\partial D}{\partial N} = -H^{1/N}\ln\left(H^{1/N}\right) + H^{1/N} + p_{inv} = 0$$

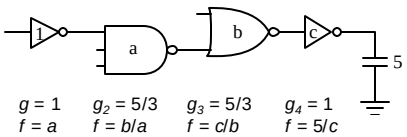
Substitute 'best stage effort' $h = H^{1/\hat{N}}$

Logical Effort

Gate Type	Number of Inputs 1	2	3	n
Inverter	1			
NAND		4/3	5/3	(n + 2)/3
NOR		5/3	7/3	(2n + 1)/3
Multiplexer		2	2	2
XOR		4	12	

From Sutherland, Sproull

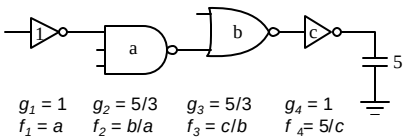
Example: Optimize Path



$g = 1$ $g_2 = 5/3$ $g_3 = 5/3$ $g_4 = 1$
$f = a$ $f = b/a$ $f = c/b$ $f = 5/c$

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

Example: Optimize Path

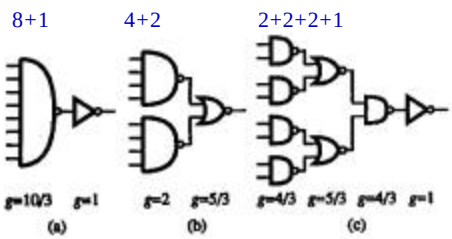


$g_1 = 1$ $g_2 = 5/3$ $g_3 = 5/3$ $g_4 = 1$
$f_1 = a$ $f_2 = b/a$ $f_3 = c/b$ $f_4 = 5/c$

Effective fanout, F = 5
G = 25/9
H = 125/9 = 13.9 (since no branching here then B=1, H=GFB)
h = 1.93 (this is the optimum effort for each gate $h=H^{1/4}$)
a = h/g_2=1.16 (from $h=f_1g_2$=1.93 and f_1=a)
b = ha/g_3 = 1.34 (same as $h=f_2g_3$=1.93 and f_2=b/a)
c = hb/g_4 = 2.59 (same as $h=f_3g_4$=1.93 and f_3=c / b)

Example – 8-input AND

Intrinsic delays



Fan out is not known here

Method of Logical Effort

- Compute the path effort: H = *GBF*
- Find the best number of stages $N \sim \log_4 H$
- Compute the stage effort $h = H^{1/N}$
- Sketch the path with this number of stages
- Work either from either end, find sizes: $C_{in} = C_{out} * g/h$

Reference: Sutherland, Sproull, Harris, "Logical Effort, Morgan-Kaufmann 1999.

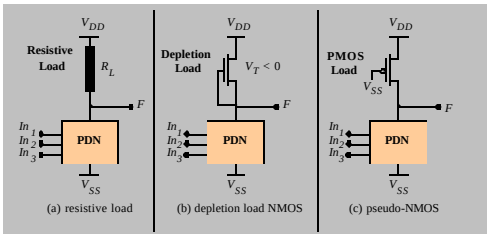
Summary

Table 4: Key Definitions of Logical Effort

Term	Stage expression	Path expression
Logical effort	g (see Table 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 (see Table 2)	$P = \sum p_i$
Delay	$d = f + p$	$D = D_F + P$

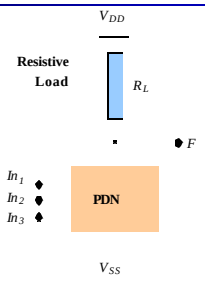
Sutherland, Sproull Harris

Ratio Based Logic



Goal: to reduce the number of devices over complementary CMOS

Ratio Based Logic



- N transistors + Load
- $V_{OH} = V_{DD}$
- $V_{OL} = \frac{R_{DN}}{R_{DN} + R_L} V_{DD}$
- Asymmetrical response
- Static power consumption
- $t_{pLH} = 0.69\, R_L\, C_L$

$$t_{pHL} = 0.69(R_L \| R_{PDN})C_L$$

Ratio Based Logic Problems

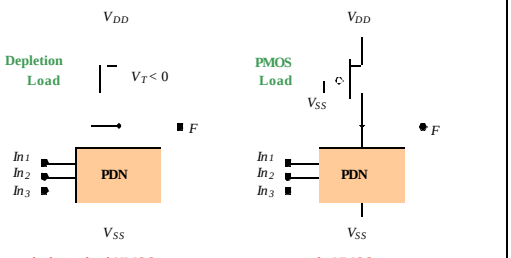
Problems with Resistive Load

- $I_L = (V_{DD} - V_{out})/ R_L$
- **Charging current drops rapidly once V_{out} starts to rise**

Solution: Use a current source!

- **Available current is independent of voltage**
- **Reduces t_{pLH} by 25%**

Active Loads



Active Loads

Depletion mode NMOS load

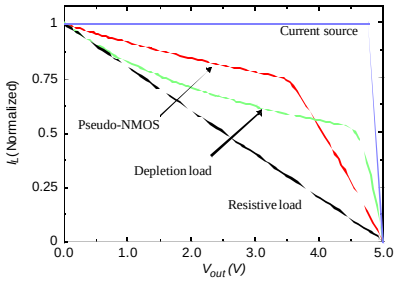
- $V_{GS} = 0$
- $I_L \sim (k_{n,\ load} / 2)(|V_{Tn}|)^2$
- Deviates from ideal current source
  - Channel length modulation
  - Body effect
    - $V_{SB} != V_{DD}$
    - varies with V_{out}
    - reduces $|V_{Tn}|$, hence I_L gets smaller for increasing V_{out}

Active Loads

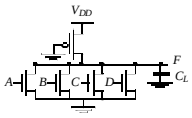
Pseudo-NMOS load

- **No body effect, $V_{SB} = 0V$**
- **$V_{GS} = -V_{DD}$, higher load current**
- **$I_L = (k_p / 2)(V_{DD} - |V_{Tn}|)^2$**
- Larger V_{GS} causes pseudo-NMOS load to leave saturation mode sooner than NMOS

Load Lines of Ratioed Gates



Pseudo-NMOS



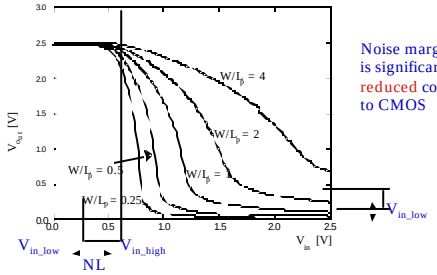
$V_{OH} = V_{DD}$ (similar to complementary CMOS)

$$k_n\left[(V_{DD} - V_{Tn})V_{OL} - \frac{V_{OL}^2}{2}\right] = \frac{k_p}{2}(V_{DD} - |V_{Tp}|)^2$$

$$V_{OL} = (V_{DD} - V_T)\left[1 - \sqrt{1 - \frac{k_p}{k_n}}\right] \quad (\text{assuming that } V_T = V_{Tn} = |V_{Tp}|)$$

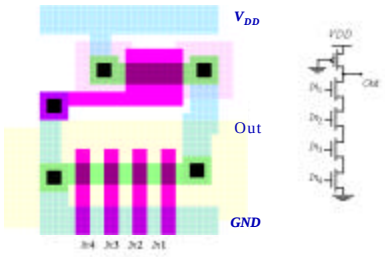
SMALLER AREA & LOAD BUT STATIC POWER DISSIPATION!!!

Pseudo-NMOS VTC



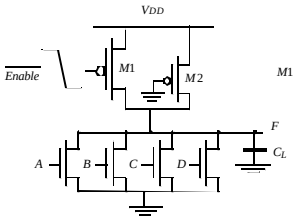
Noise margin low is significantly reduced comparing to CMOS

Pseudo-NMOS NAND Gate



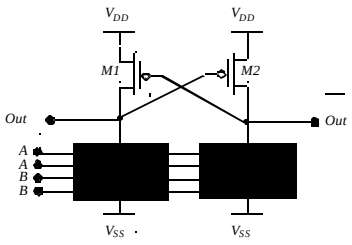
$C_{L,pseudo} = 0.5\ C_{L,CMOS}$ (Fan-out of 1)

Improved Loads



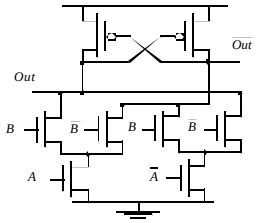
Adaptive Load

Improved Loads (2)



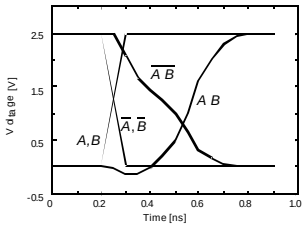
Differential Cascode Voltage Switch Logic (DCVSL)

DCVSL Example

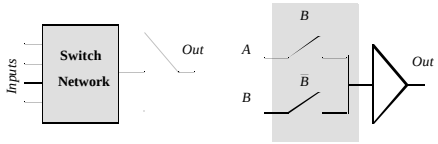


XOR-NXOR gate

DCVSL Transient Response

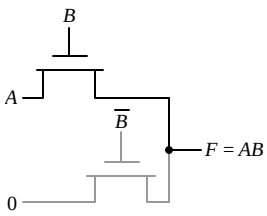


Pass-Transistor Logic

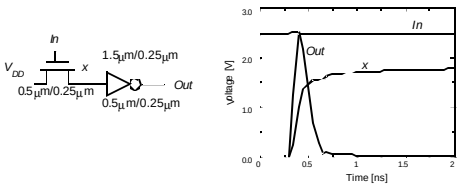


- **N transistors**
- **No static consumption**

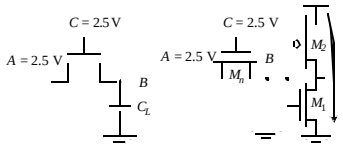
Example: AND Gate



NMOS-Only Logic



NMOS-only Switch

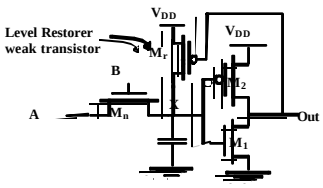


V_B does not pull up to 2.5V, but 2.5V -V_{TN}

Threshold voltage loss causes static power consumption

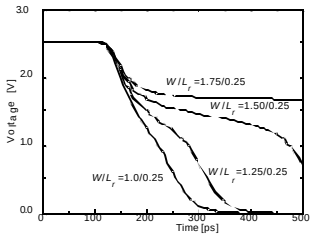
NMOS has higher threshold than PMOS (body effect)

NMOS Only Logic: Level Restoring Transistor



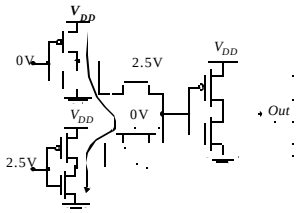
- Advantages: Full Swing, No static power dissipation
- Restorer adds capacitance, takes away pull down current at X
- Ratio problem

Restorer Sizing



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

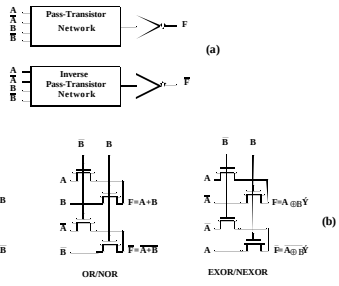
Solution 2: Single Transistor Pass Gate with V_T=0



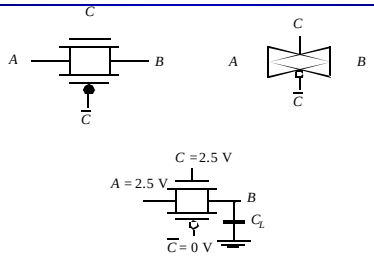
If pass transistors have V_T=0 the output Does not require level restorer but there is a leakage current

WATCH OUT FOR LEAKAGE CURRENTS

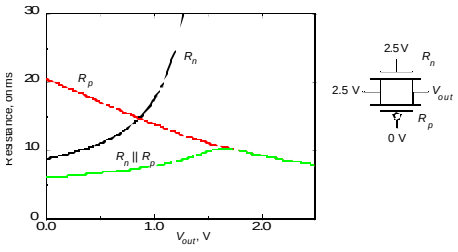
Complementary Pass Transistor Logic



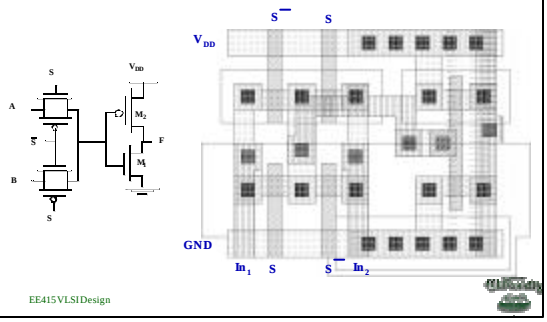
Solution 3: Transmission Gate



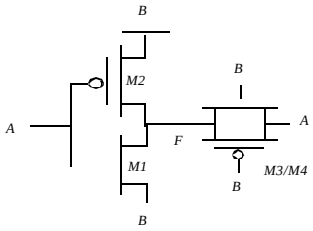
Resistance of Transmission Gate



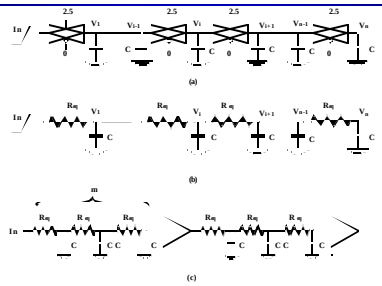
Pass-Transistor Based Multiplexer



Transmission Gate XOR



Delay in Transmission Gate Networks



Delay Optimization

- **Delay of RC chain**

$$t_p = 0.69 \sum_{k=0}^{n} CR_{eq} k = 0.69 CR_{eq} \frac{n(n+1)}{2}$$

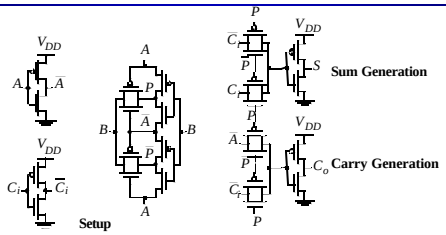
- **Delay of Buffered Chain**

$$t_p = 0.69\left[\frac{n}{m} CR_{eq} \frac{m(m+1)}{2}\right] + \left(\frac{n}{m} - 1\right) t_{buf}$$

$$= 0.69\left[CR_{eq} \frac{n(m+1)}{2}\right] + \left(\frac{n}{m} - 1\right) t_{buf}$$

$$w_{opt} = 1.7 \sqrt{\frac{t_{pbuf}}{CR_{eq}}}$$

Transmission Gate Full Adder

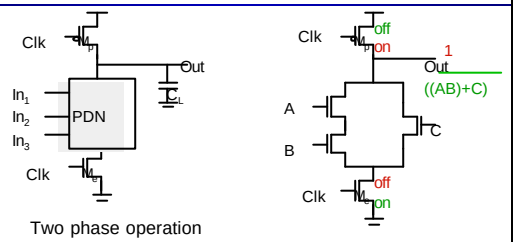


Similar delays for sum and carry

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 2*n* (*n* N-type + *n* P-type) devices
- Dynamic circuits rely on the temporary storage of signal values on the capacitance of high impedance nodes.
  - » requires on *n* + 2 (*n*+1 N-type + 1 P-type) transistors

Dynamic Gate



Two phase operation

Precharge (Clk = 0)

Evaluate (Clk = 1)

Conditions on Output

- Once the output of a dynamic gate is discharged, it cannot be charged again until the next precharge operation.
- Inputs to the gate can make at most one transition during evaluation.
- Output can be in the high impedance state during and after evaluation (PDN off), state is stored on C_L

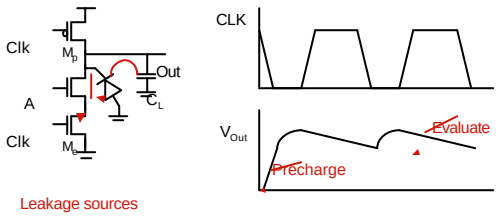
Properties of Dynamic Gates

- Logic function is implemented by the PDN only
  - » number of transistors is N + 2 (versus 2N for static complementary CMOS)
- Full swing outputs (V_{OL} = GND and V_{OH} = V_{DD})
- Non-ratioed - sizing of the devices does not affect the logic levels
- Faster switching speeds
  - » reduced load capacitance due to lower input capacitance (C_{in})
  - » reduced load capacitance due to smaller output loading (Cout)
  - » no I_{sc}, so all the current provided by PDN goes into discharging C_L

Properties of Dynamic Gates

- Overall power dissipation usually higher than static CMOS
  - » no static current path ever exists between V_{DD} and GND (including P_{sc})
  - » no glitching
  - » higher transition probabilities
  - » extra load on Clk
- PDN starts to work as soon as the input signals exceed V_{Tn}, so V_M, V_{IH} and V_{IL} equal to V_{Tn}
  - » low noise margin (NM_L)
- Needs a precharge/evaluate clock

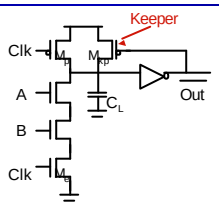
Issues in Dynamic Design 1: Charge Leakage



Leakage sources

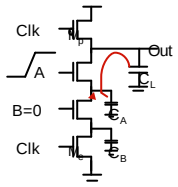
Dominant component is subthreshold current

Solution to Charge Leakage



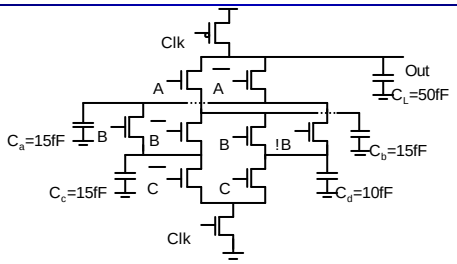
Same approach as level restorer for pass-transistor logic

Issues in Dynamic Design 2: Charge Sharing

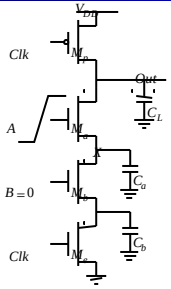


Charge stored originally on C_L is redistributed (shared) over C_L and C_A leading to reduced robustness

Charge Sharing Example



Charge Sharing



case 1) if $\Delta V_{out} < V_{Tn}$

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

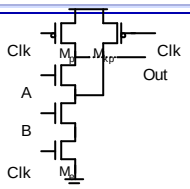
or

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

case 2) if $\Delta V_{out} > V_{Tn}$

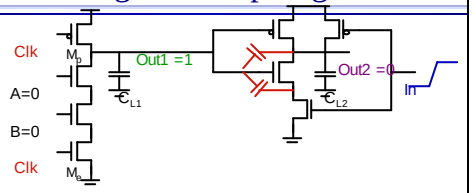
$$\Delta V_{out} = -V_{DD}\left(\frac{C_a}{C_a + C_L}\right)$$

Solution to Charge Redistribution



Precharge internal nodes using a clock-driven transistor (at the cost of increased area and power)

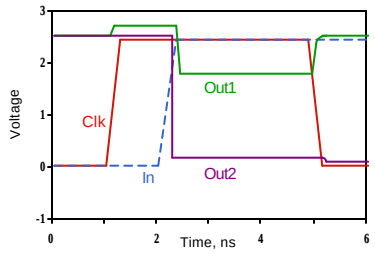
Issues in Dynamic Design 3: Backgate Coupling



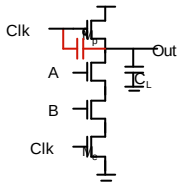
Dynamic NAND Static NAND

Output connected to a NAND gate and Out2 pulls down Out1 through capacitive coupling

Backgate Coupling Effect

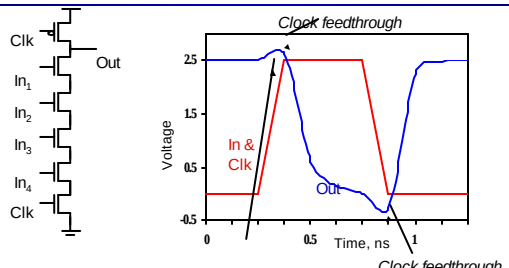


Issues in Dynamic Design 4: Clock Feedthrough



Coupling between Out and Clk input of the precharge device due to the gate to 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

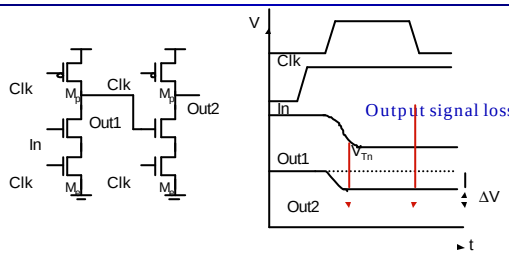


More noise is generated as a result of clock coupling

Other Effects

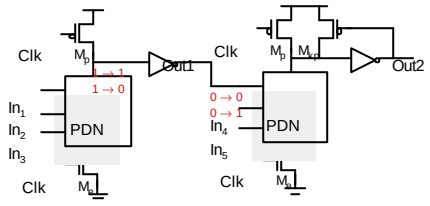
- Capacitive coupling between output wires pulls down prestored charges
- Substrate coupling
- Minority charge injection
- Supply noise (negative ground bounce may discharge the output)

Cascading Dynamic Gates



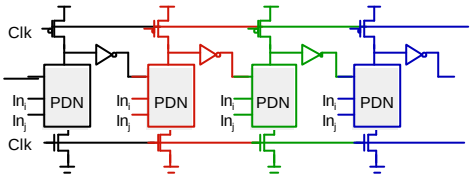
Only 0 → 1 transitions allowed at inputs!
So do not connect these gates directly

Domino Logic



Here we guarantee proper 0 to 1 transitions between gates

Why Domino?

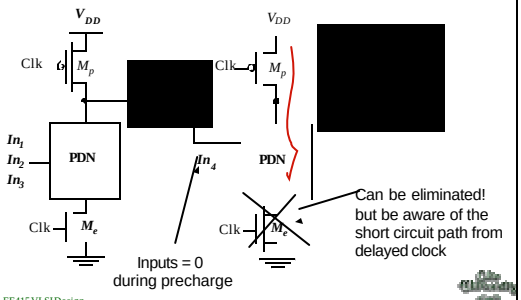


Like falling dominos!

Properties of Domino Logic

- Only non-inverting logic can be implemented
- Very high speed
  - » static inverter can be skewed, only L-H transition so make PMOS of inverter stronger
  - » Input capacitance reduced – smaller logical effort

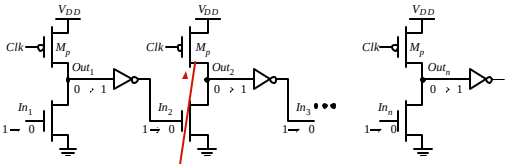
Designing with Domino Logic



Inputs = 0 during precharge

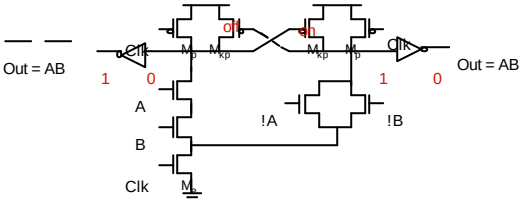
Can be eliminated! but be aware of the short circuit path from delayed clock

Footless Domino



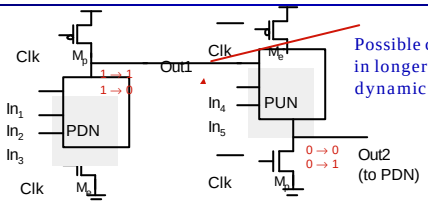
The first gate in the chain needs a foot switch
Precharge is rippling – short-circuit current
A solution is to delay the clock for each stage

Differential (Dual Rail) Domino



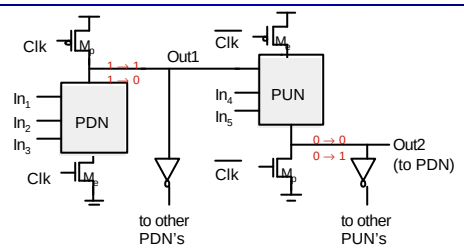
Solves the problem of non-inverting logic

np-CMOS



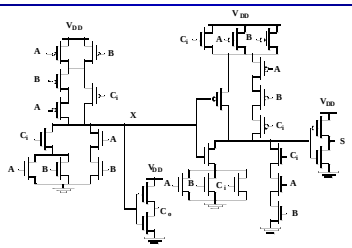
Only 0 → 1 transitions allowed at inputs of PDN
Only 1 → 0 transitions allowed at inputs of PUN

NORA Logic



WARNING: Very sensitive to noise!

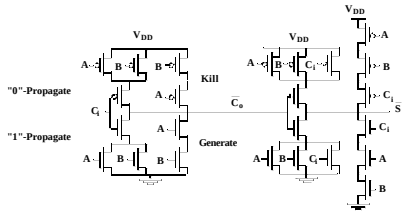
Example: Full Adder



$C_o = AB + C_i(A+B)$

28 transistors

A Revised Adder Circuit



24 transistors