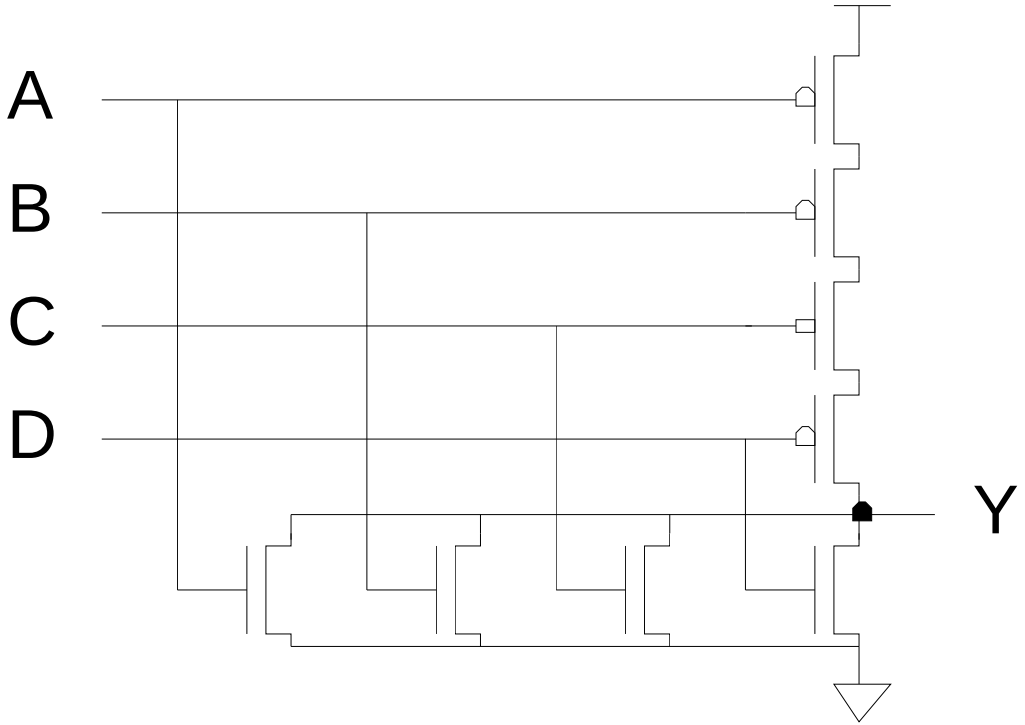
Subsystem Design and Layout

Architectural issues

- Define the requirements (properly and carefully).
- Partition the overall architecture into appropriate subsystems.
- Consider communication paths carefully in order to develop sensible interrelationships between subsystems.
- Draw a floor plan of how the system is to map onto the silicon (and alternate between 2, 3 and 4 as necessary).
- Aim for regular structures so that design is largely a matter of replication.
- Draw suitable (stick or symbolic) diagrams of the leaf-cells of the subsystems.
- Convert each cell to a layout.
- Carefully and thoroughly carry out ' a design rule check on each cell.
- Simulate the performance of each cell/subsystem.

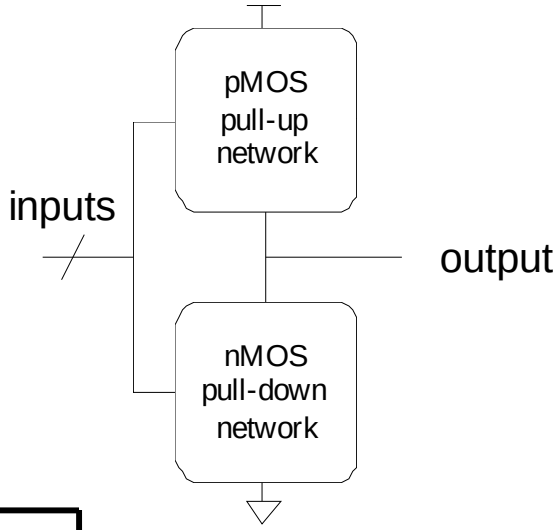
CMOS Gate Design

- A 4-input CMOS NOR gate



Complementary CMOS

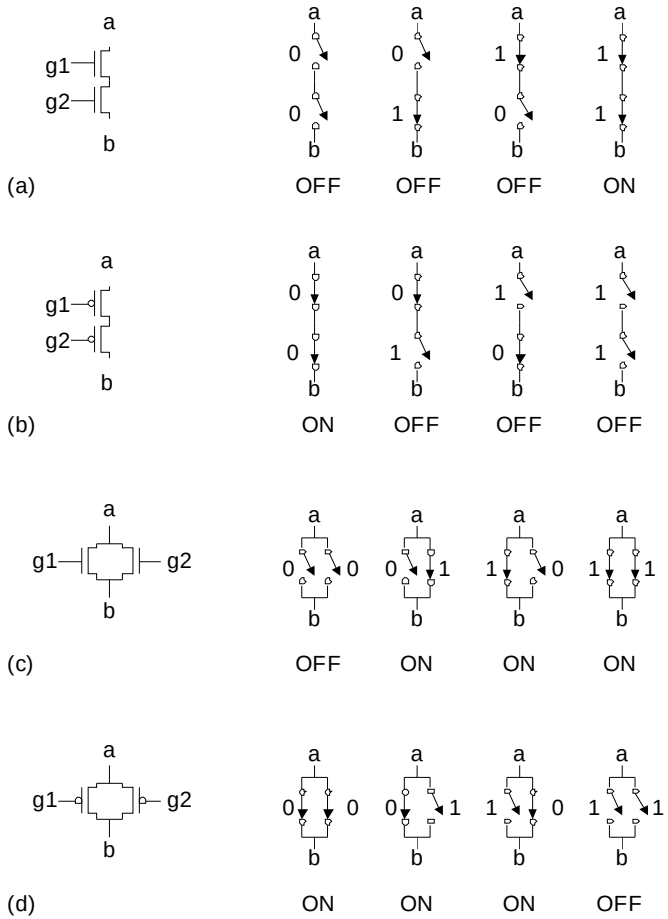
- Complementary CMOS logic gates
  - nMOS *pull-down network*
  - pMOS *pull-up network*
  - a.k.a. static CMOS



	Pull-up OFF	Pull-up ON
Pull-down OFF	Z (float)	1
Pull-down ON	0	X (crowbar)

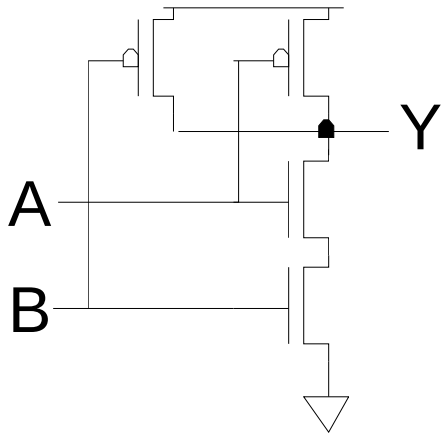
Series and Parallel

- nMOS: 1 = ON
- pMOS: 0 = ON
- *Series*: both must be ON
- *Parallel*: either can be ON



Conduction Complement

- Complementary CMOS gates always produce 0 or 1
- Ex: NAND gate
  - Series nMOS: Y=0 when both inputs are 1
  - Thus Y=1 when either input is 0
  - Requires parallel pMOS

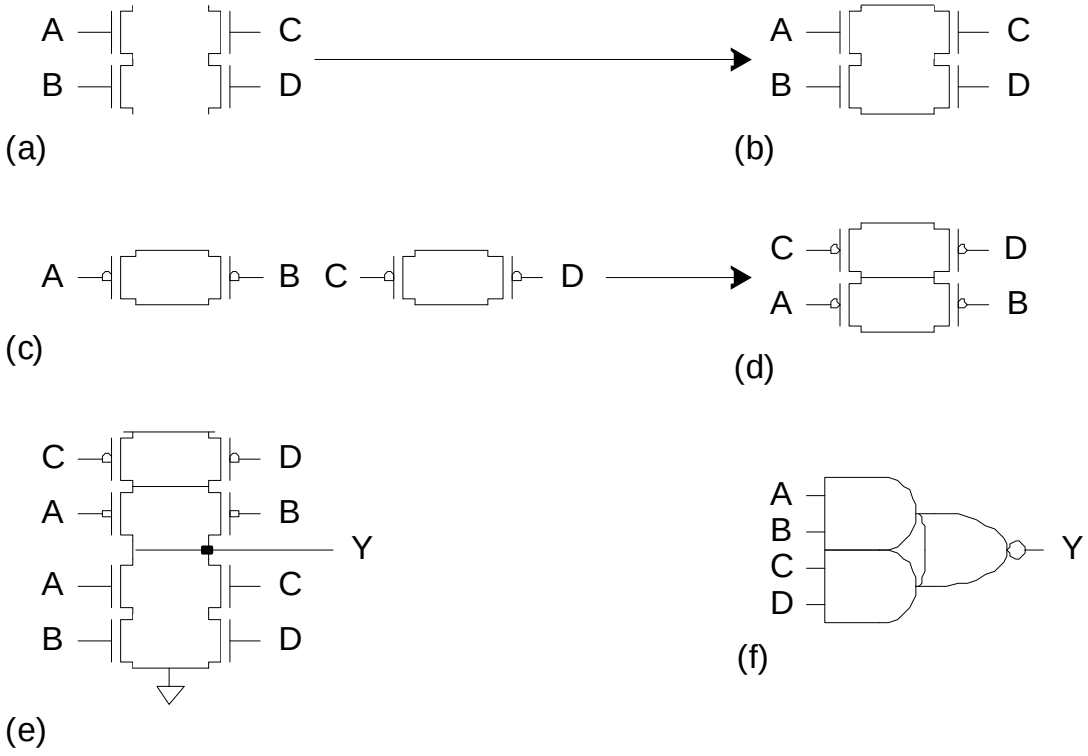


- Rule of *Conduction Complements*
  - Pull-up network is complement of pull-down
  - Parallel -> series, series -> parallel

Compound Gates

- *Compound gates* can do any inverting function
- Ex: AND-AND-OR-INV (AOI22)

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

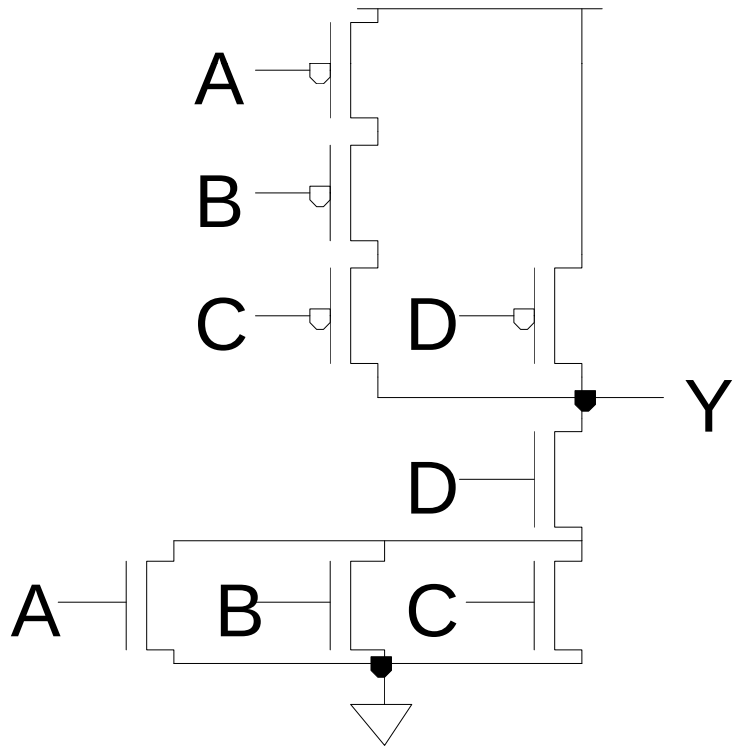


Example: O3AI

- $Y = \overline{(A + B + C) \bullet D}$

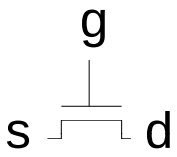
Example: O3AI

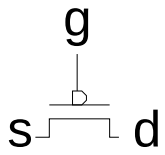
- $$Y = \overline{(A + B + C) \bullet D}$$



Pass Transistors

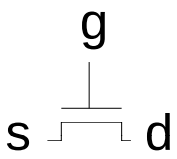
- Transistors can be used as switches

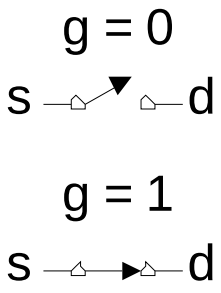


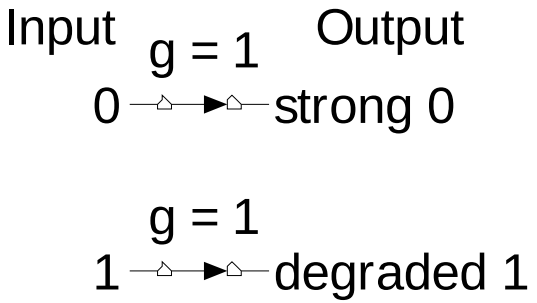


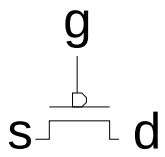
Pass Transistors

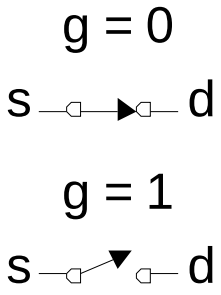
- Transistors can be used as switches

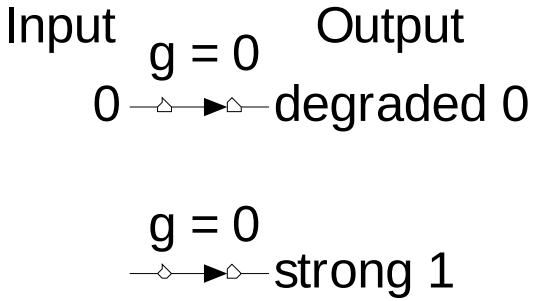












Signal Strength

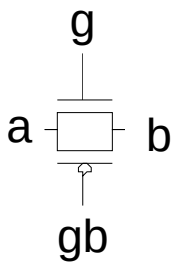
- *Strength* of signal
  - How close it approximates ideal voltage source
- V_{DD} and GND rails are strongest 1 and 0
- nMOS pass strong 0
  - But degraded or weak 1
- pMOS pass strong 1
  - But degraded or weak 0
- Thus NMOS are best for pull-down network
- Thus PMOS are best for pull-up network

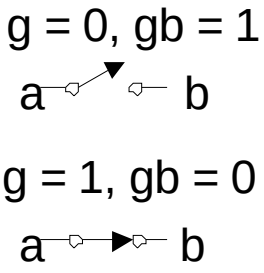
Transmission Gates

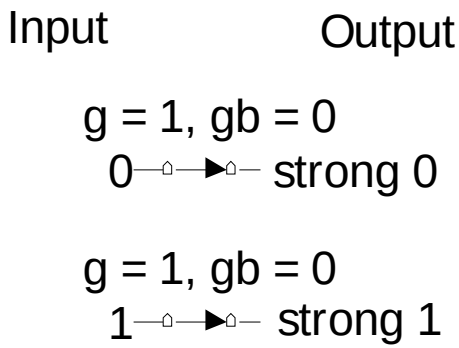
- Pass transistors produce degraded outputs
- *Transmission gates* pass both 0 and 1 well

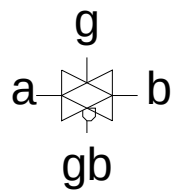
Transmission Gates

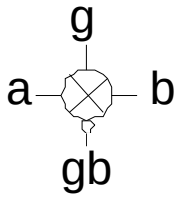
- Pass transistors produce degraded outputs
- *Transmission gates* pass both 0 and 1 well

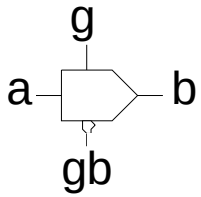








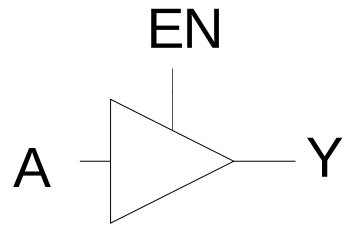


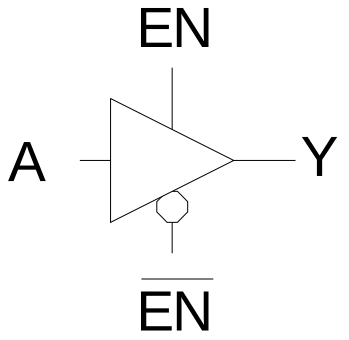


Tristates

- *Tristate buffer* produces Z when not enabled

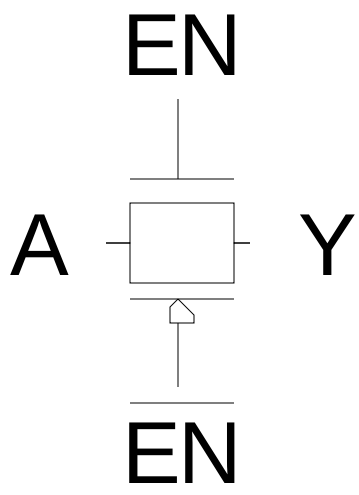
EN	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1





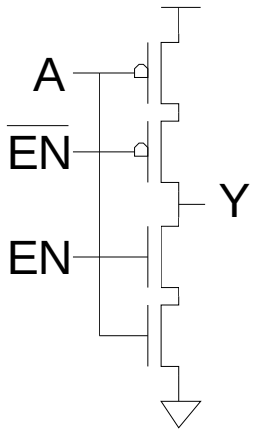
Nonrestoring Tristate

- Transmission gate acts as tristate buffer
  - Only two transistors
  - But *nonrestoring*
    - Noise on A is passed on to Y (after several stages, the noise may degrade the signal beyond recognition)



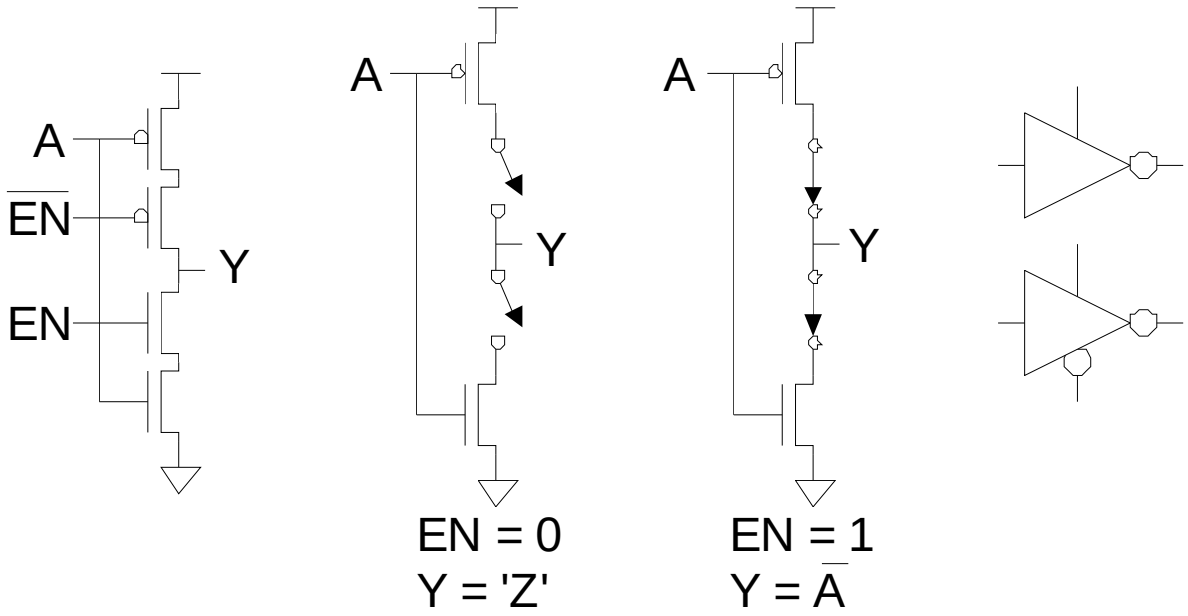
Tristate Inverter

- Tristate inverter produces restored output
- Note however that the Tristate buffer
  - ignores the conduction complement rule because we want a Z output



Tristate Inverter

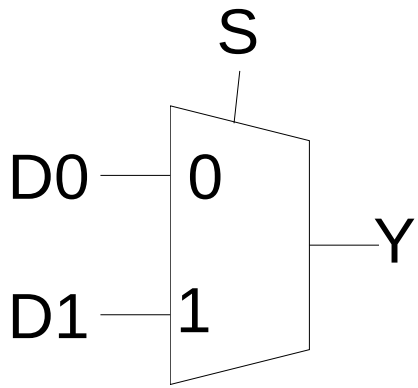
- Tristate inverter produces restored output
- Note however that the Tristate buffer
  - ignores the conduction complement rule because we want a Z output



Multiplexers

- 2:1 *multiplexer* chooses between two inputs

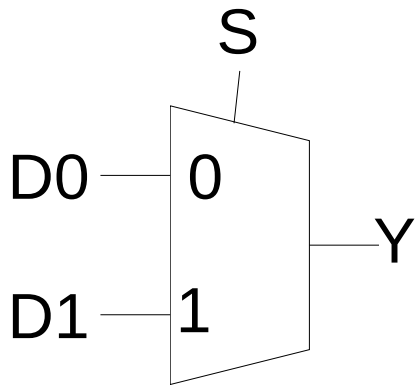
S	D1	D0	Y
0	X	0	
0	X	1	
1	0	X	
1	1	X	



Multiplexers

- 2:1 multiplexer chooses between two inputs

S	D1	D0	Y
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1

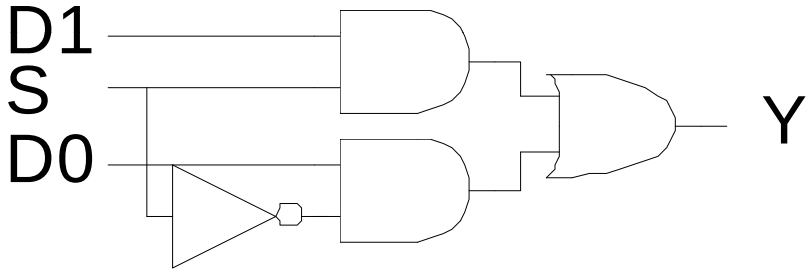


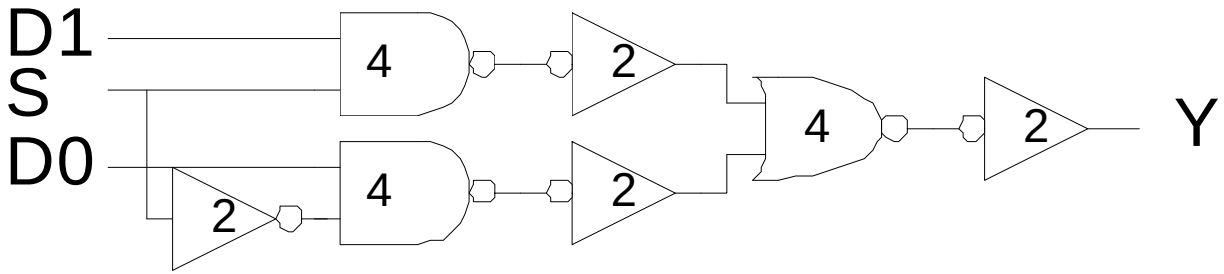
Gate-Level Mux Design

- $Y = SD_1 + \overline{S}D_0$ (too many transistors)
- How many transistors are needed?

Gate-Level Mux Design

- $Y = SD_1 + \overline{S}D_0$ (too many transistors)
- How many transistors are needed? 20



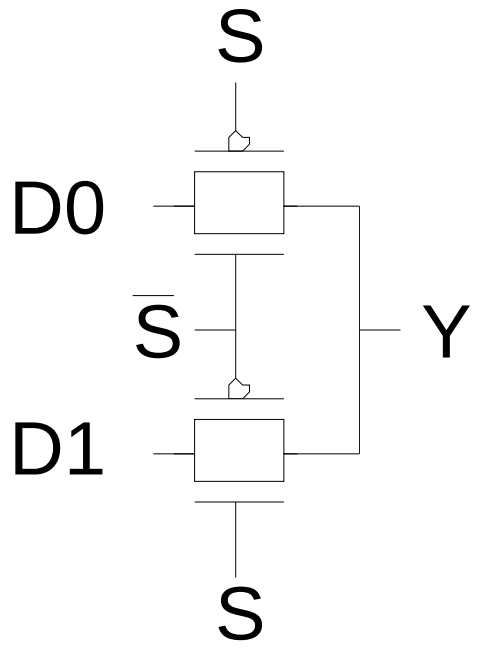


Transmission Gate Mux

- Nonrestoring mux uses two transmission gates

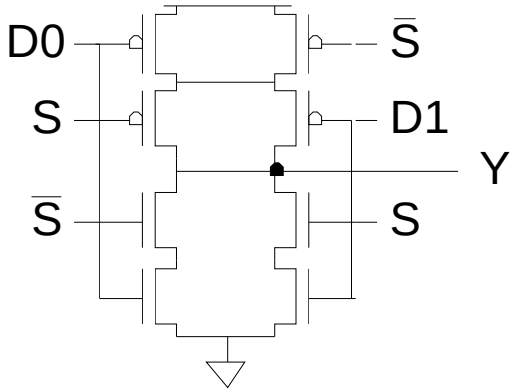
Transmission Gate Mux

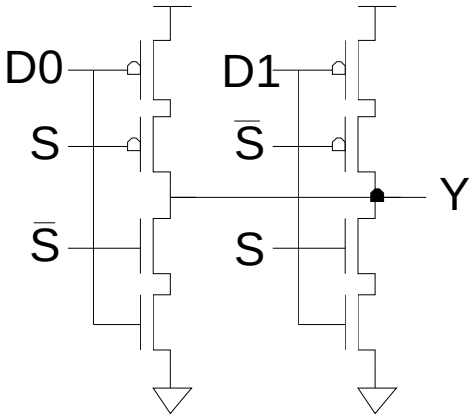
- Nonrestoring mux uses two transmission gates
  - Only 4 transistors

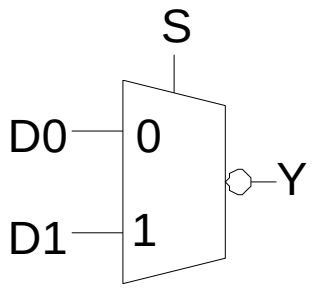


Inverting Mux

- Inverting multiplexer
  - Use compound AOI22
  - Or pair of tristate inverters
  - Essentially the same thing
- Noninverting multiplexer adds an inverter





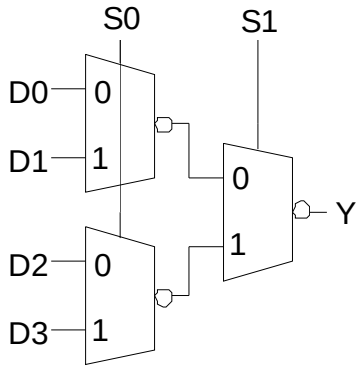


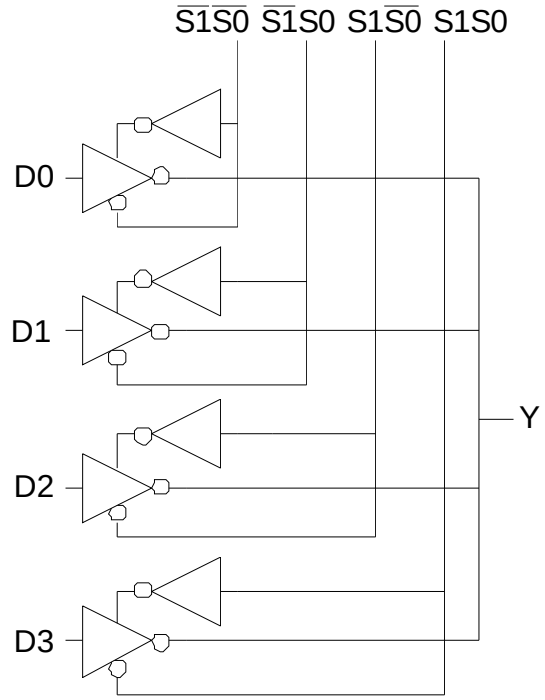
4:1 Multiplexer

- 4:1 mux chooses one of 4 inputs using two selects

4:1 Multiplexer

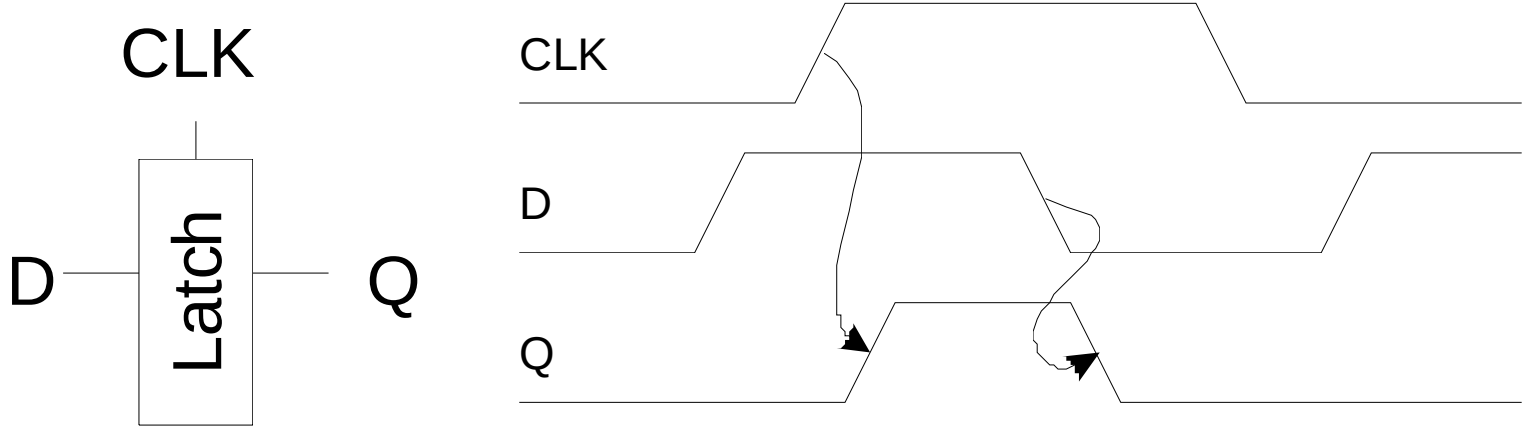
- 4:1 mux chooses one of 4 inputs using two selects
  - Two levels of 2:1 muxes
  - Or four tristates





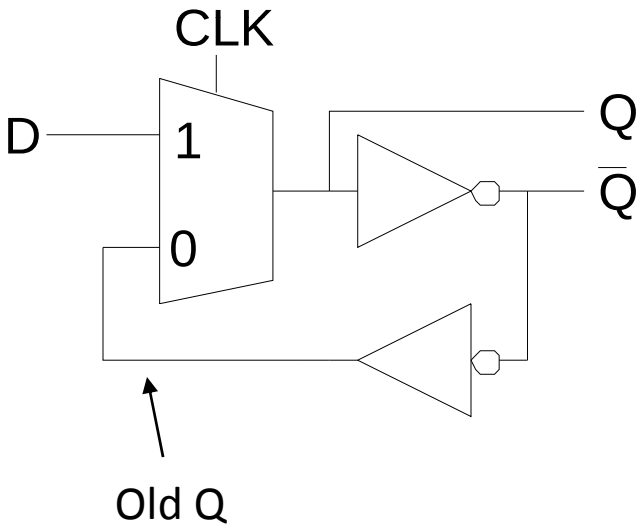
D Latch

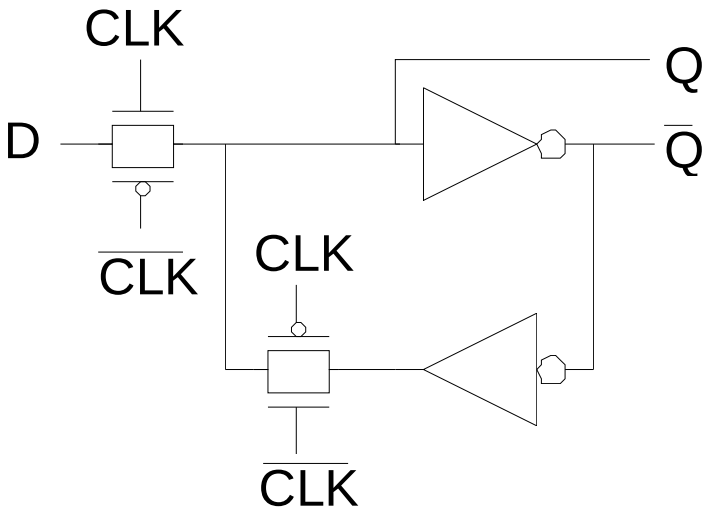
- When CLK = 1, latch is *transparent*
  - Q follows D (a buffer with a Delay)
- When CLK = 0, the latch is *opaque*
  - Q holds its last value independent of D
- a.k.a. *transparent latch* or *level-sensitive latch*



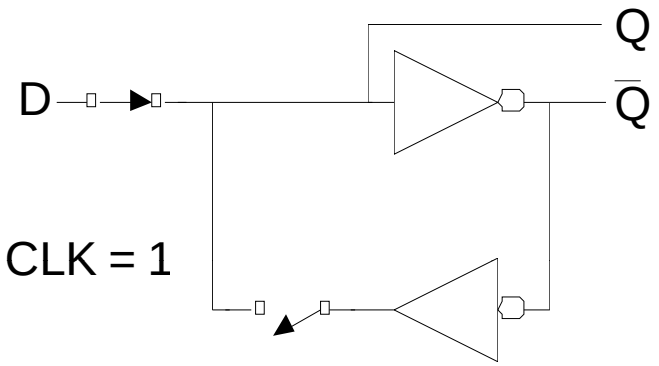
D Latch Design

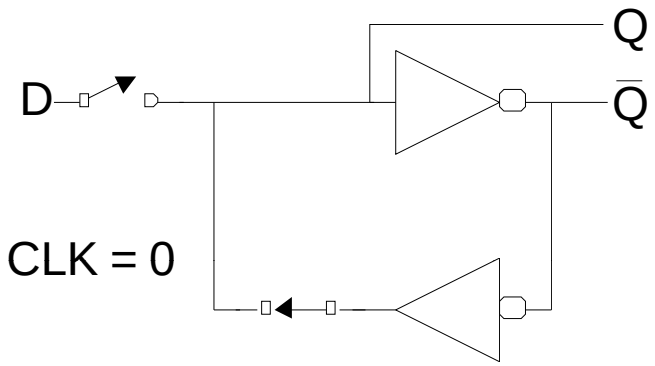
- Multiplexer chooses D or old Q

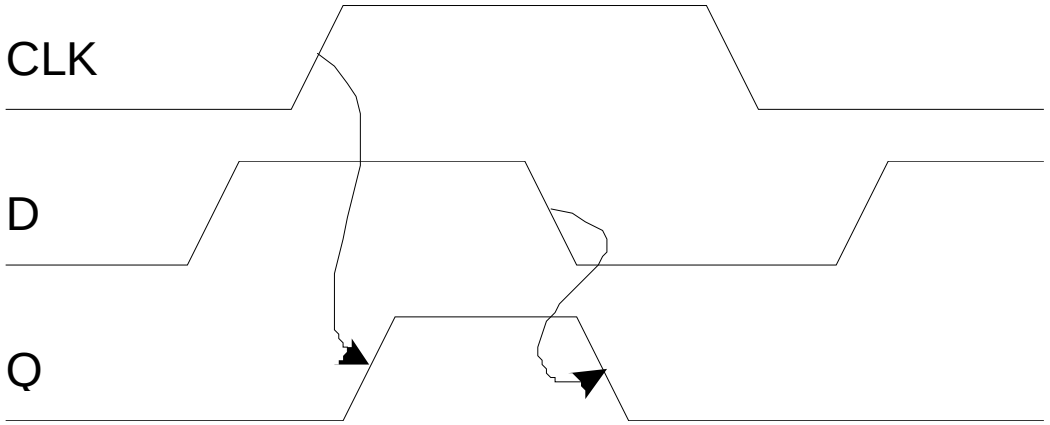




D Latch Operation

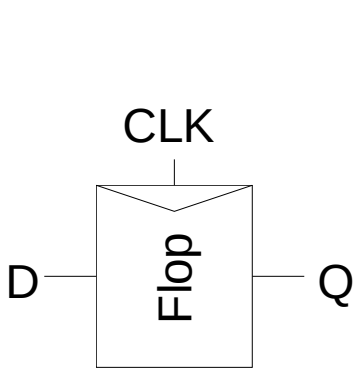


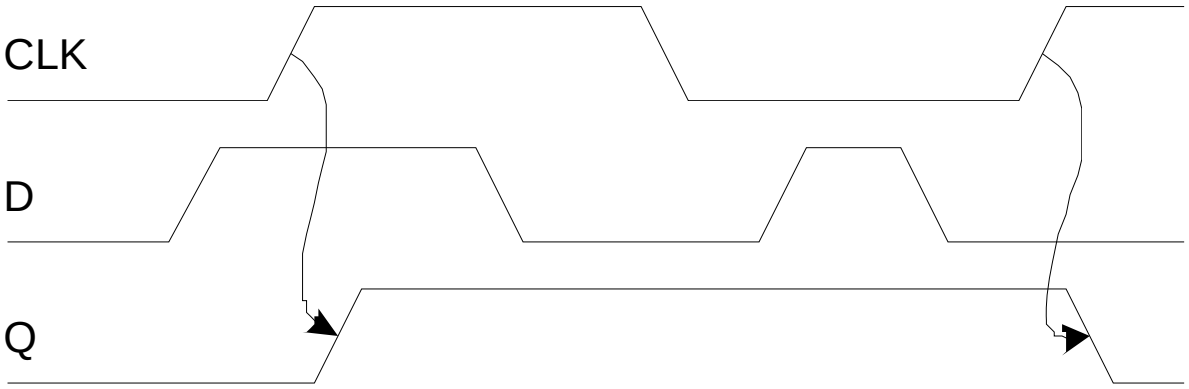




D Flip-flop

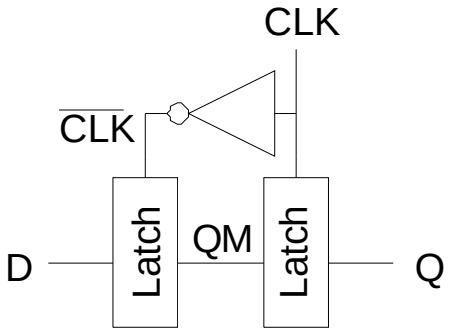
- When CLK rises, D is copied to Q
- At all other times, Q holds its value
- a.k.a. *positive edge-triggered flip-flop*, *master-slave flip-flop*

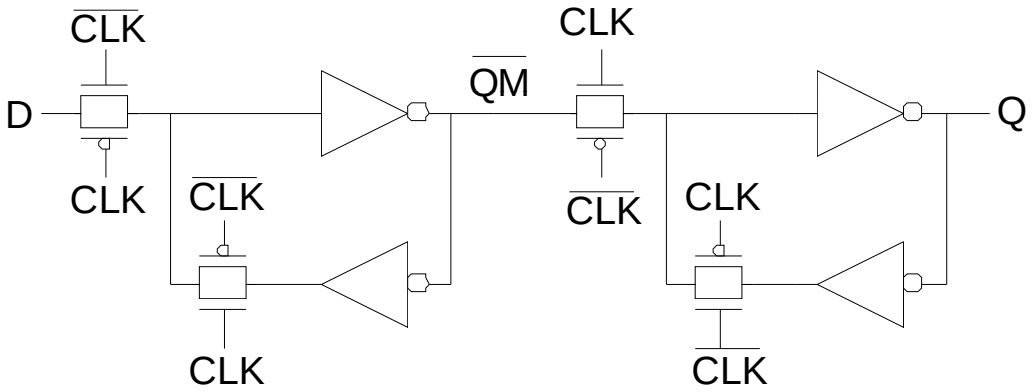




D Flip-flop Design

- Built from master and slave D latches

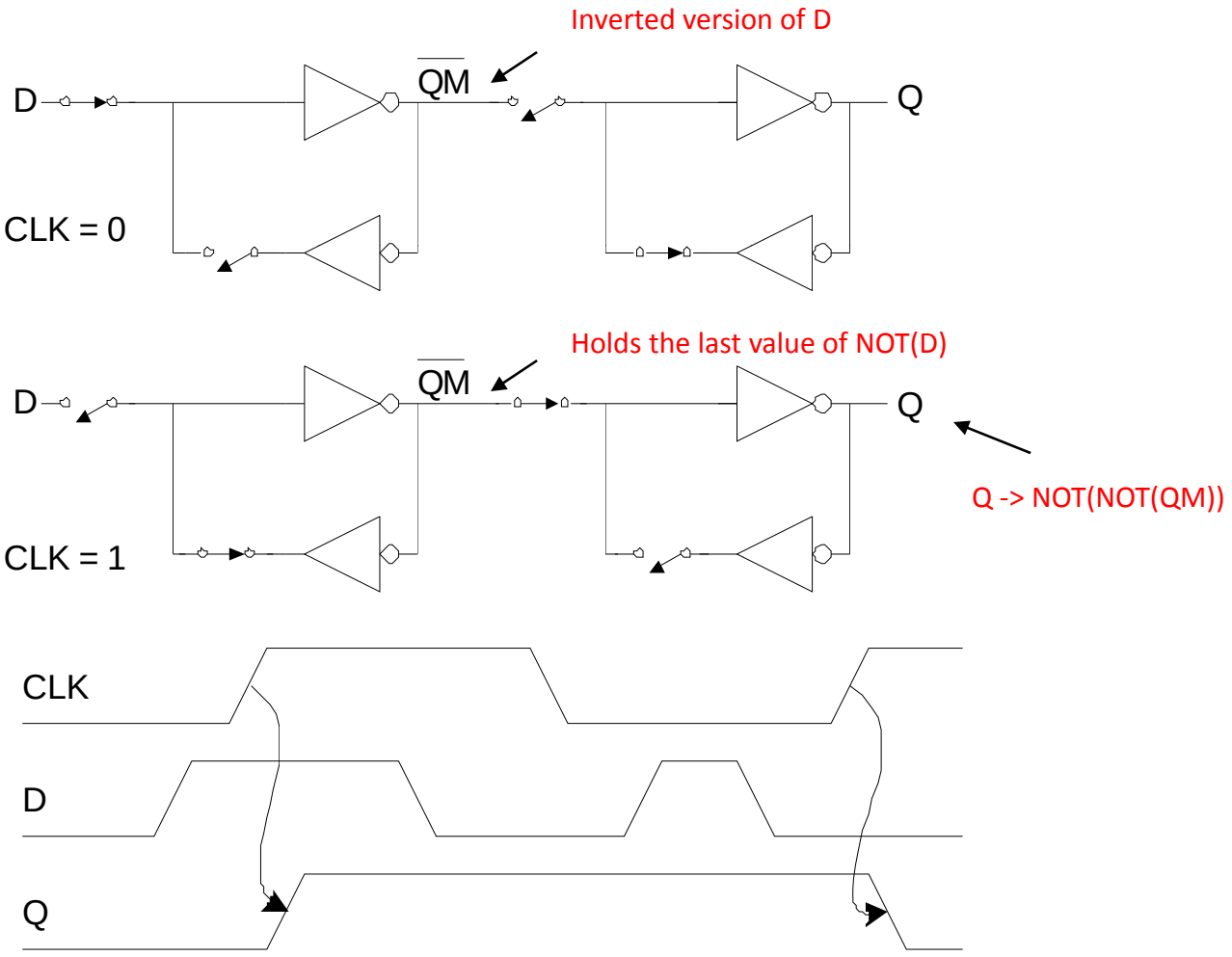




A “negative level-sensitive” latch

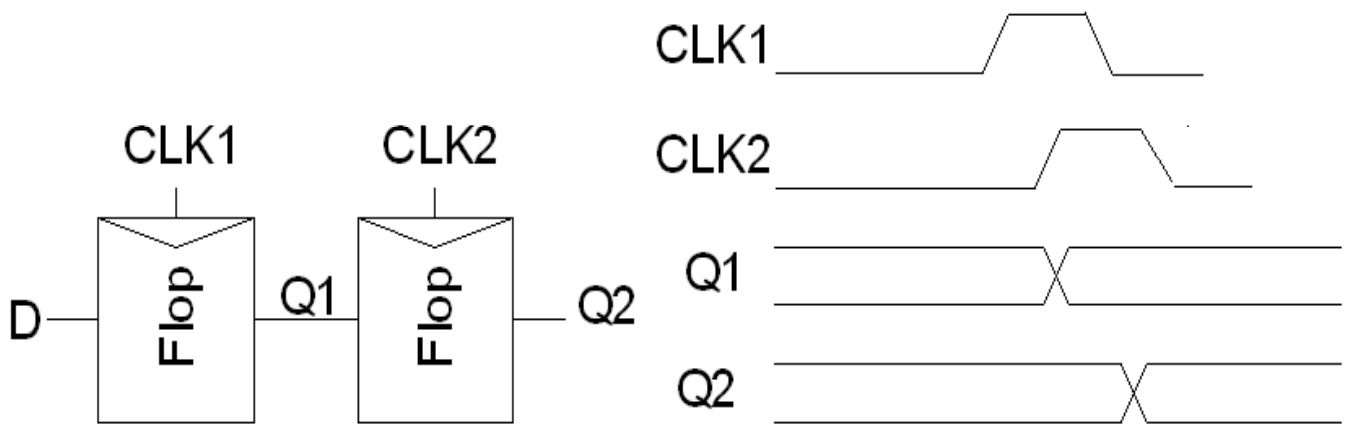
A “positive level-sensitive” latch

D Flip-flop Operation



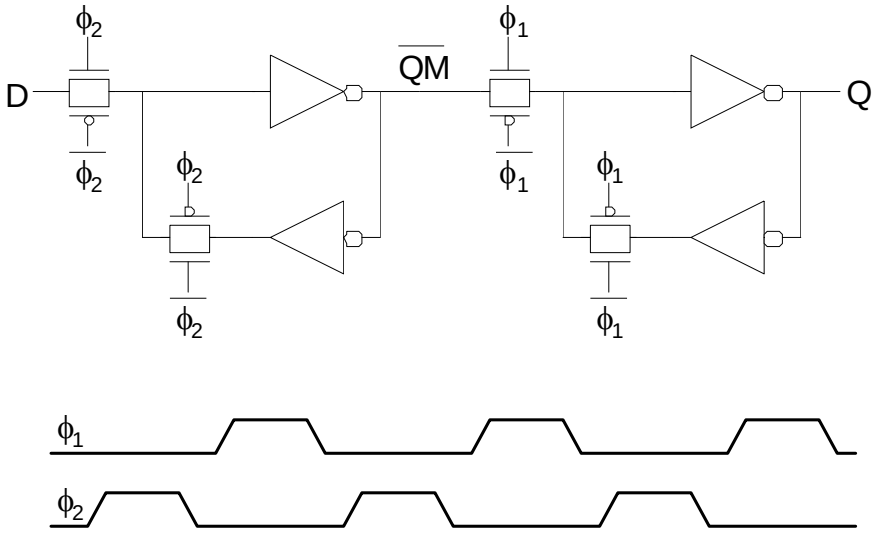
Race Condition

- Back-to-back flops can malfunction from clock skew
  - Second flip-flop fires Early
  - Sees first flip-flop change and captures its result
  - Called *hold-time failure* or *race condition*



Nonoverlapping Clocks

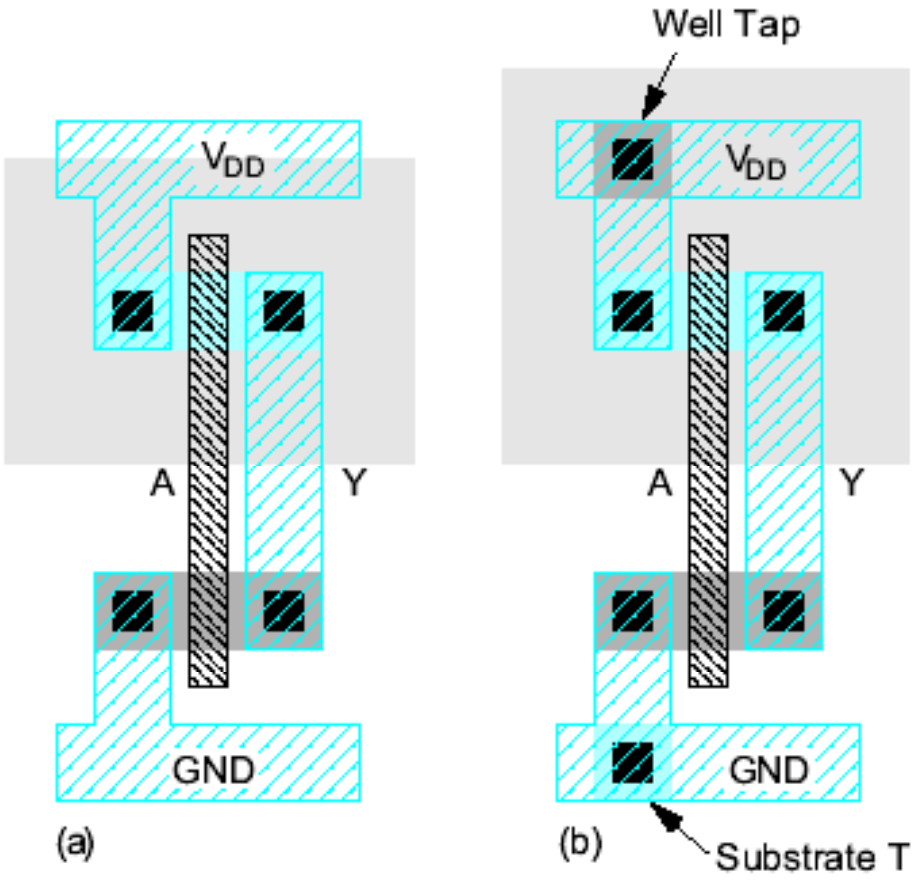
- Nonoverlapping clocks can prevent races
  - As long as nonoverlap exceeds clock skew
- Good for safe design
  - Industry manages skew more carefully instead



Gate Layout

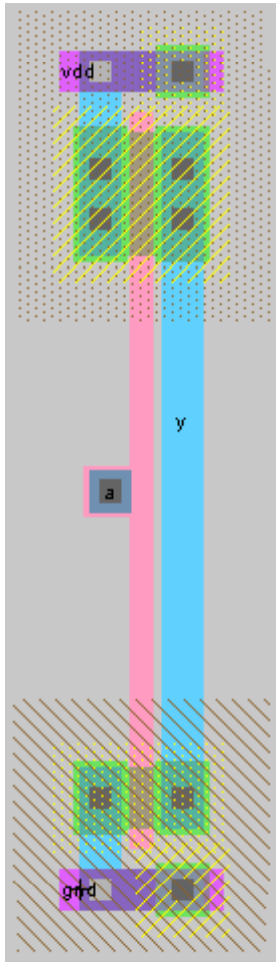
- Layout can be very time consuming
  - Design gates to fit together nicely
  - Build a library of standard cells
  - Must follow a technology rule
- Standard cell design methodology
  - V_{DD} and GND should abut (standard height)
  - Adjacent gates should satisfy design rules
  - nMOS at bottom and pMOS at top
  - All gates include well and substrate contacts

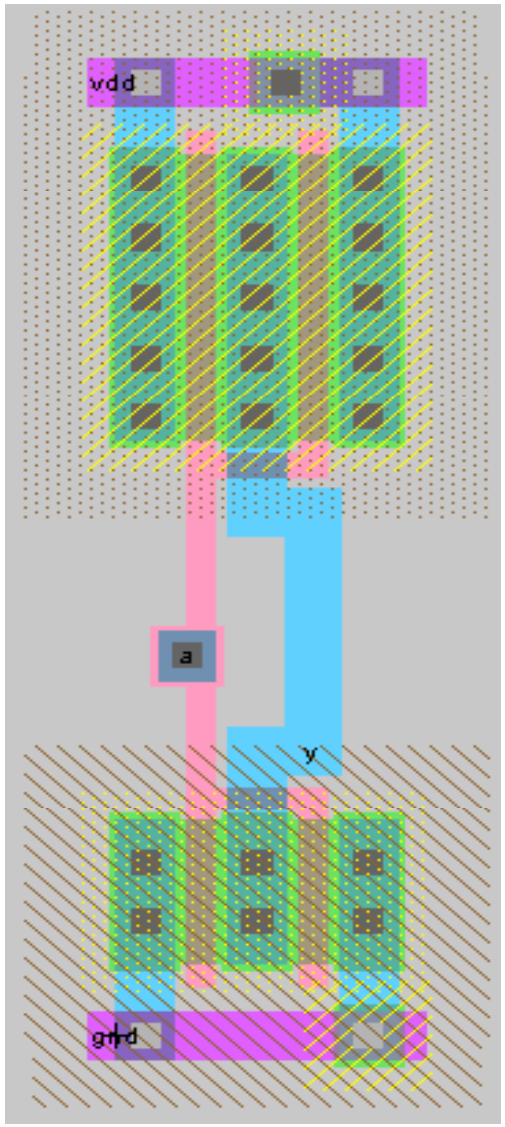
Example: Inverter



Layout using Electric

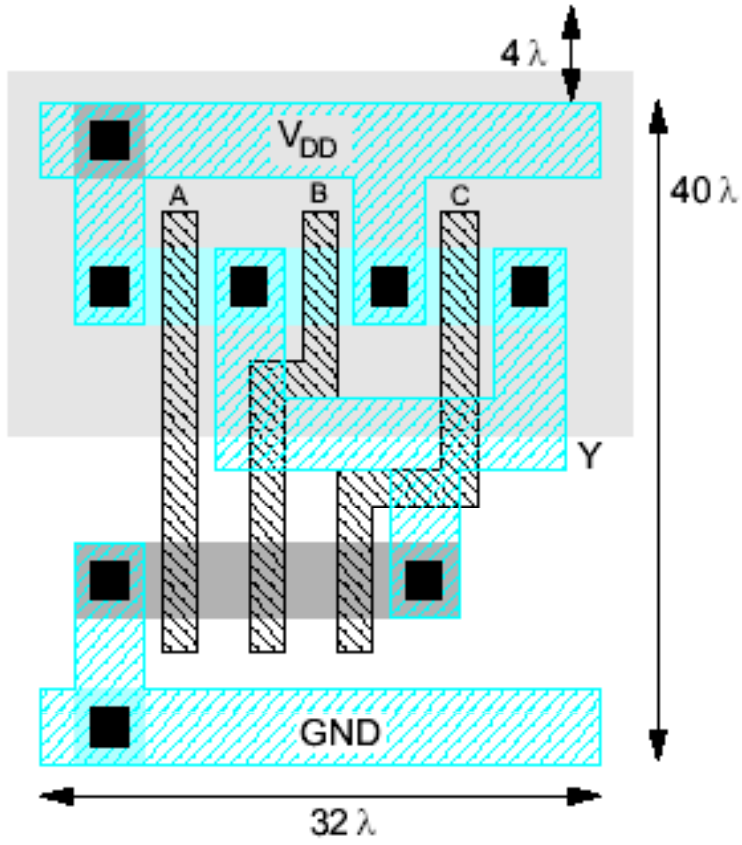
Inverter, contd..



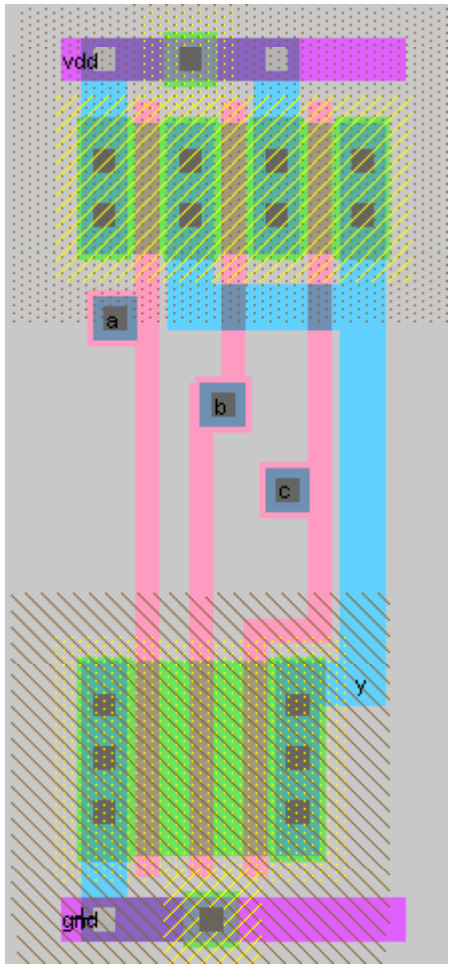


Example: NAND3

- Horizontal N-diffusion and p-diffusion strips
- Vertical polysilicon gates
- Metal1 V_{DD} rail at top
- Metal1 GND rail at bottom
- 32 λ by 40 λ

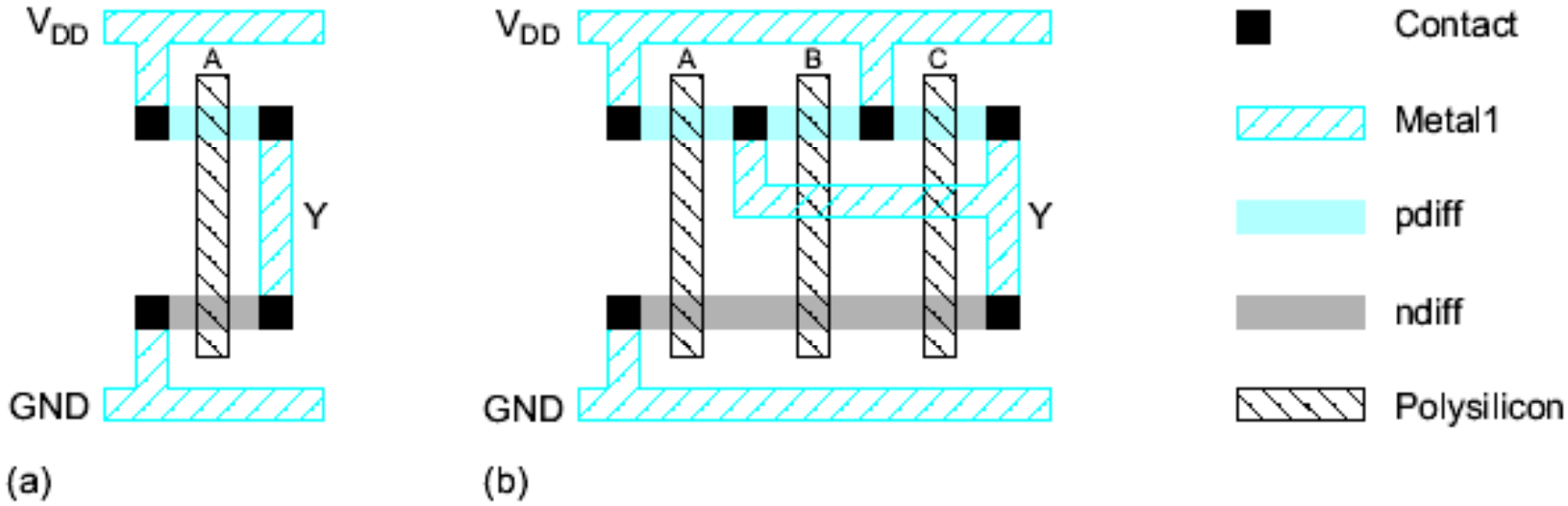


NAND3 (using Electric), contd.



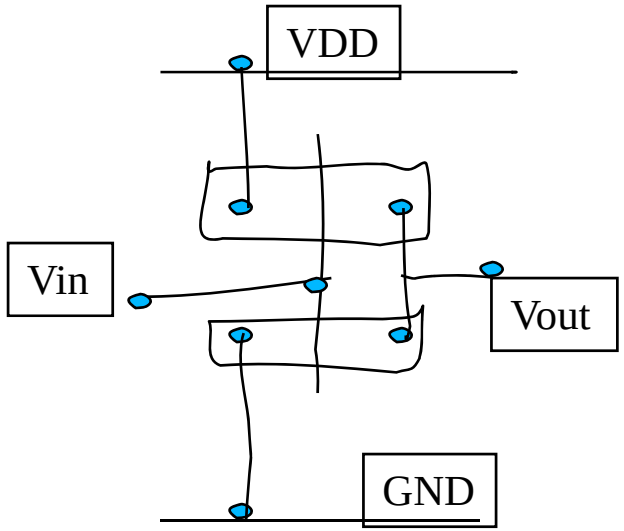
Stick Diagrams

- *Stick diagrams* help plan layout quickly
  - Need not be to scale
  - Draw with color pencils or dry-erase markers



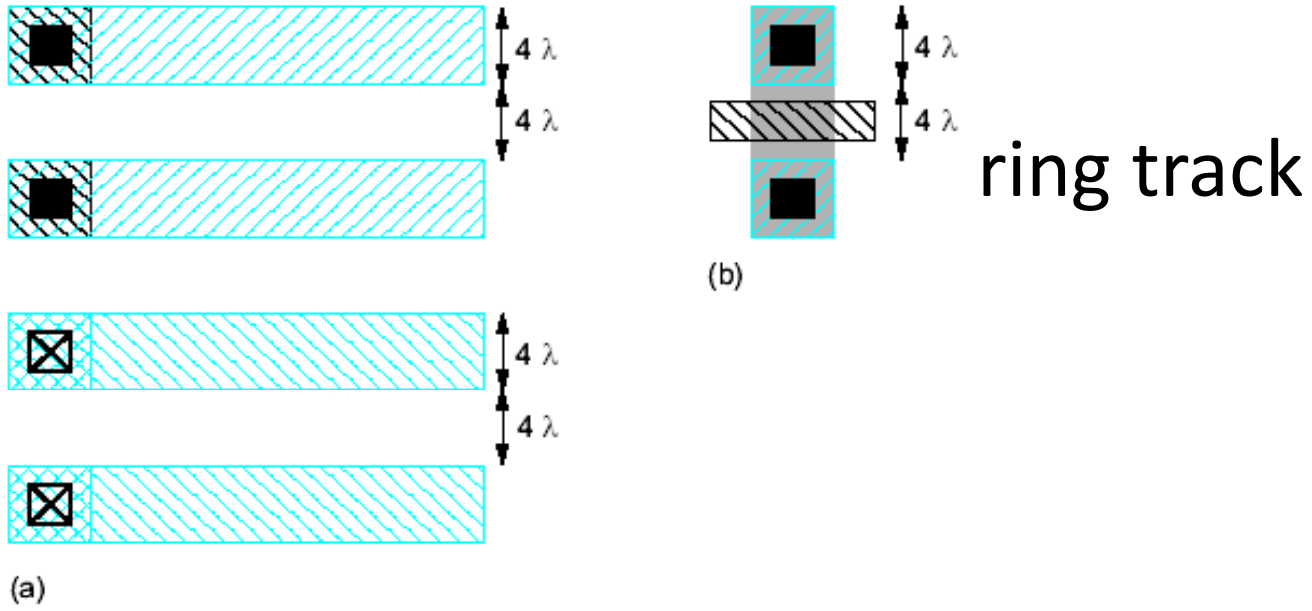
Stick Diagrams

- *Stick diagrams* help plan layout quickly
  - Need not be to scale
  - Draw with color pencils or dry-erase markers



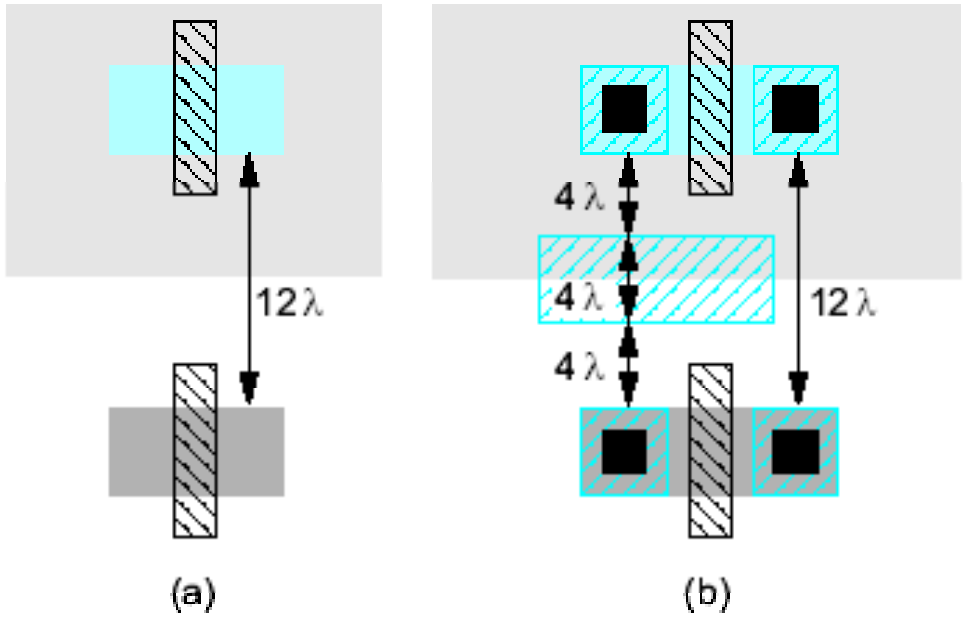
Wiring Tracks

- A *wiring track* is the space required for a wire
  - 4 λ width, 4 λ spacing from neighbor = 8 λ pi
- Trar ring track



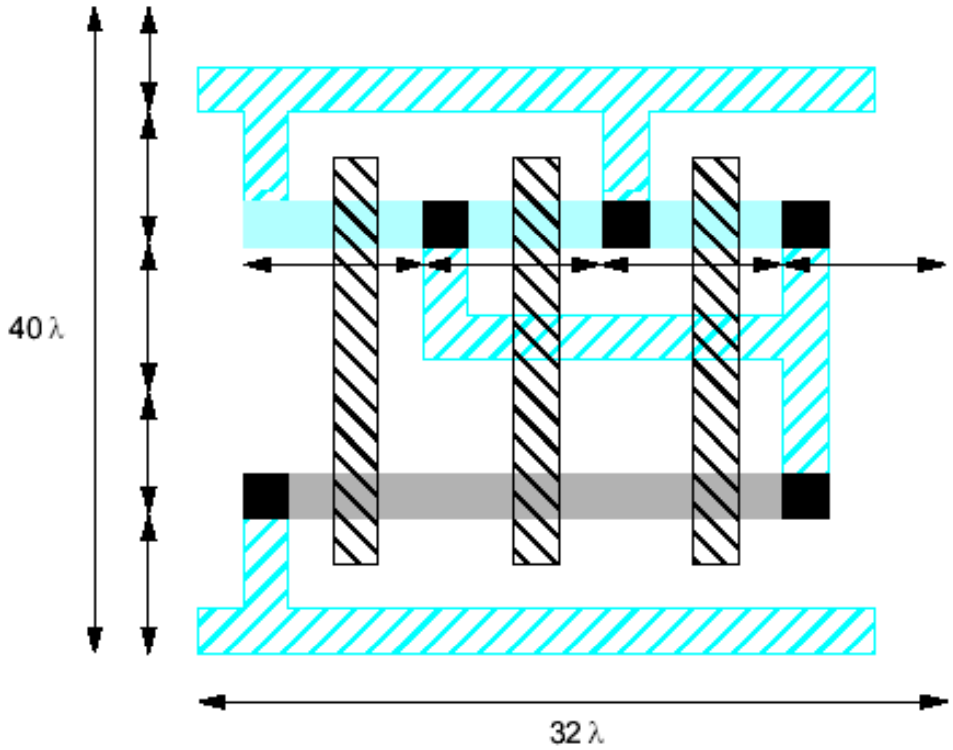
Well spacing

- Wells must surround transistors by 6 λ
  - Implies 12 λ between opposite transistor flavors
  - Leaves room for one wire track



Area Estimation

- Estimate area by counting wiring tracks
  - Multiply by 8 to express in λ

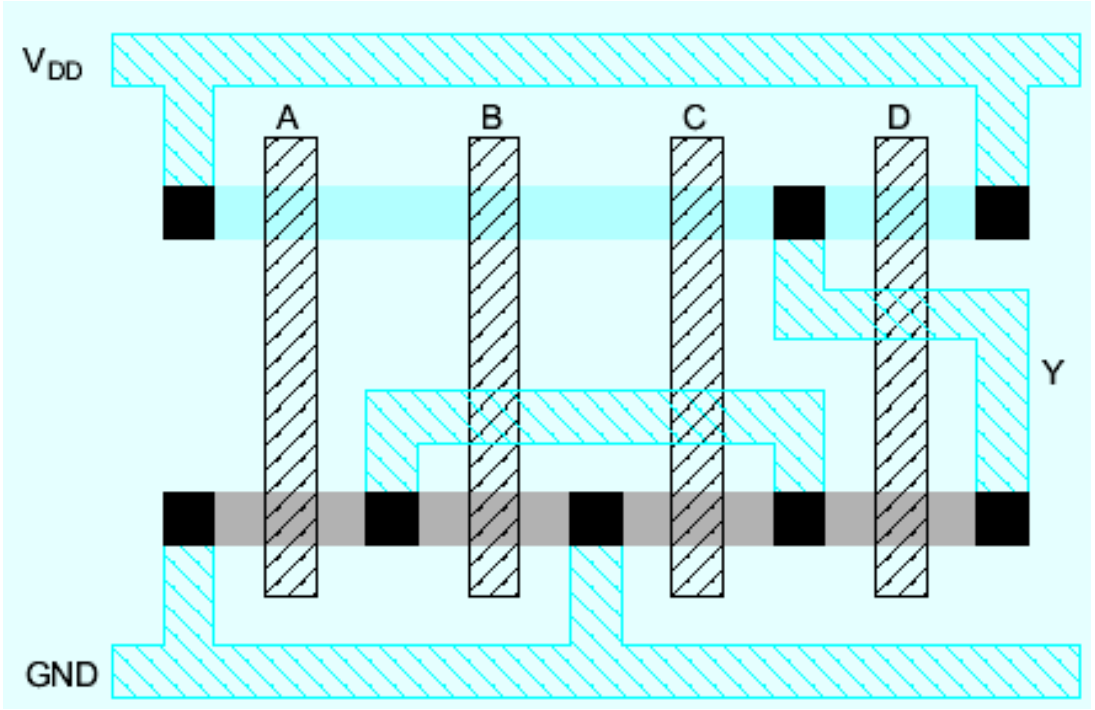


Example: O3AI

- Sketch a stick diagram for O3AI and estimate area
  - $$Y = \overline{(A + B + C) \bullet D}$$

Example: O3AI

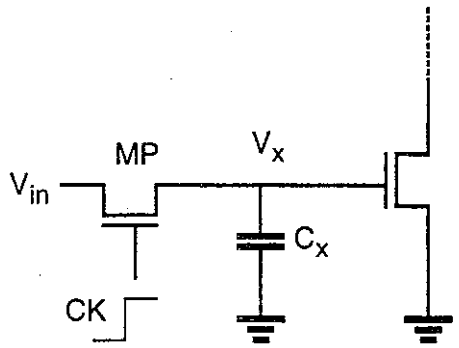
- Sketch a stick diagram for O3AI and estimate area
  - $Y = \overline{(A + B + C) \bullet D}$

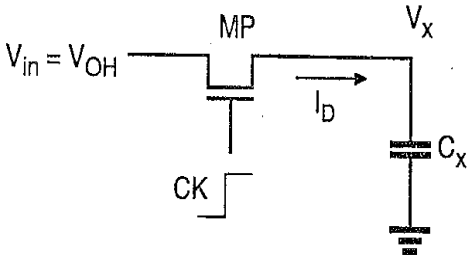


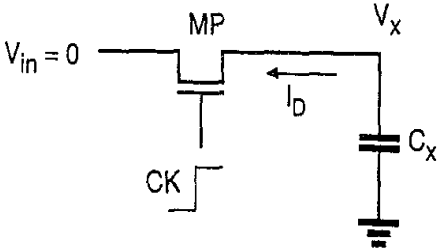
Dynamic Logic Circuits

- Dynamic logic is temporary (**transient**) in that output levels will remain valid only for a certain period of time
  - Static logic retains its output level as long as power is applied
- Dynamic logic is normally done with charging and selectively discharging capacitance (i.e. capacitive circuit nodes)
  - **Precharge** clock to charge the capacitance
  - **Evaluate** clock to discharge the capacitance depending on condition of logic inputs
- Advantages over static logic:
  - Avoids duplicating logic twice as both N-tree and P-tree, as in standard CMOS
  - Typically can be used in very high performance applications
  - Very simple sequential memory circuits; amenable to synchronous logic
  - High density achievable
  - Consumes less power (in some cases)
- Disadvantages compared to static logic:
  - Problems with clock synchronization and timing
  - Design is more difficult

NMOS Dynamic Logic Basic Circuit

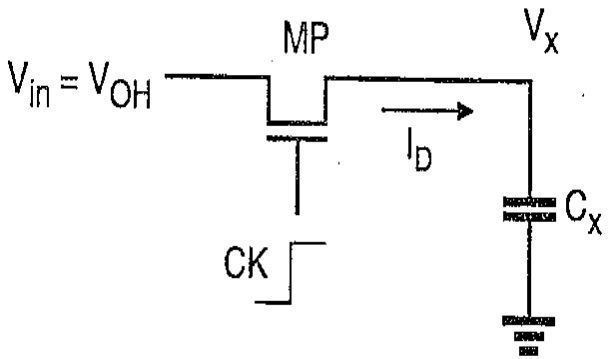






- The basic **dynamic logic gate concept** is shown at left (top)
  - the pass transistor MP is an NMOS device, but could also be implemented with a transmission gate TG
  - C_x represents the equivalent capacitance of the input gate of the second NMOS device (part of an inverter or logic gate) as well as the PN junction capacitance of MP's drain (source)
  - When clock CK goes high, MP is turned on and allows the input voltage Vin to be placed on capacitor Cx
    - Vin could be a high ("1") or a low ("0") voltage
  - When CK goes low, MP is turned off, trapping the charge on Cx
- Operation for a 1 or a 0:
  - If Vin is high (say V_{OH}), then MP will allow current to flow into Cx, charging it up to Vdd – Vtn (assume CK up level is Vdd)
  - If Vin is low (say GND), then MP will allow current to flow out of Cx, discharging it to GND
- Due to leakage from the drain (source) of MP, Cx can only retain the charge Q for a given period of time (called soft node)
  - If MP is NMOS, Cx will discharge to GND
  - If MP is PMOS, Cx will discharge to VDD
  - If MP is a TG, Cx could discharge in either direction

Dynamic NMOS Logic: Transfer "1" Event



- Charging event with NMOS operating in source-follower mode:
  - MP will be saturated during transfer "1" transient
  - Max voltage attainable at Vx will be Vdd – Vtn, assuming that the CK pulse height is Vdd
  - Solve for increasing voltage Vx versus time:

    $$\mathbf{C_x\,(dV_x/dt) = \tfrac{1}{2}\,\beta_n(Vdd - V_x - Vtn)^2}$$

  - Solution:

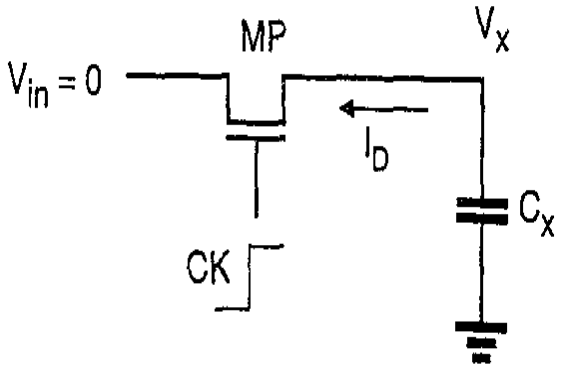
$$\mathbf{t = (2Cx/\,\beta_n)[\{1/(Vdd - Vx - Vtn) - 1/(Vdd - Vtn)\}]}$$

or, solving for $V_x(t)$

$$\mathbf{V_x(t) = (Vdd - Vtn)[1 - 1/\{1 + (Vdd - Vtn)(\beta_n/2C_x)t\}]}$$

- As t → infinity, $V_x(t)$ → Vdd – Vtn
- Solve for time needed to reach 90% (Vdd – Vtn):
  - Set $V_x(t) = 0.9\,(Vdd - Vtn)$ → $\mathbf{t_{90\%} = 18\,C_x/\,\beta_n(Vdd - Vtn)}$
    - i.e. 18 time constants

Dynamic NMOS Logic: Transfer "0" Event



- On a transfer "0" event, the NMOS transfer device is in its common source configuration, i.e. the source is at GND and the drain is discharging Cx
  - MP is operating in the linear mode for the entire transient since the starting value is Vdd – Vtn
  - Solve for decreasing Vx with time:

  $$\mathbf{C_x\,(dV_x/dt) = -\,\beta_n\,V_x\,(Vdd - Vtn - \tfrac{1}{2}\,V_x)^2}$$

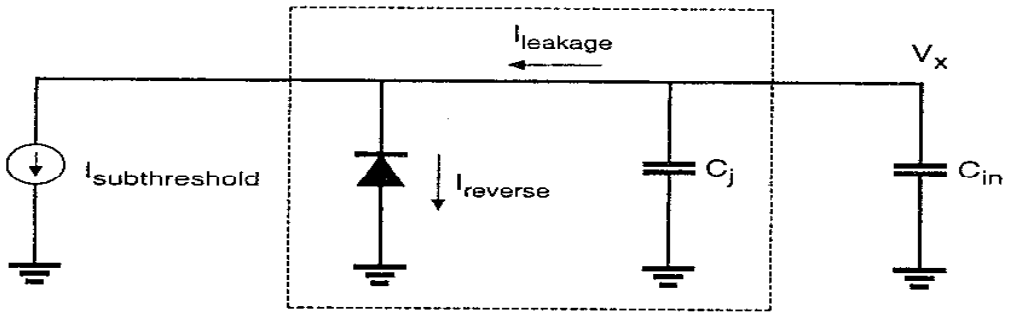
  - Solution:

  $$\mathbf{t = Cx/(\beta_n(Vdd - Vtn))\ ln\{(2(Vdd - Vtn) - Vx)/Vx\}}$$

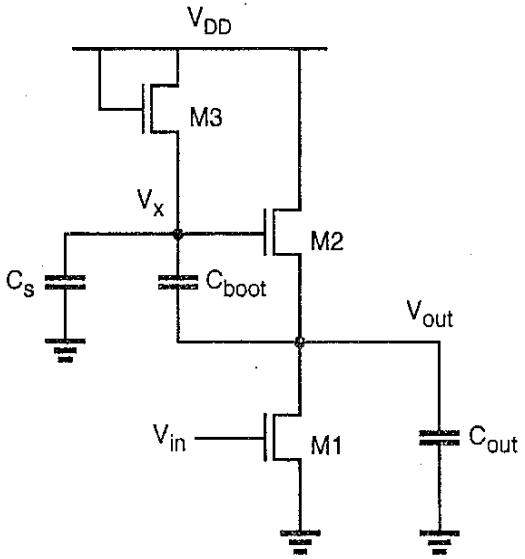
- Solve for time needed for Vx to fall to 10% (Vdd – Vtn):
  - Set $V_x(t) = 0.1\,(Vdd - Vtn)$ ➔ $\mathbf{t_{10\%} = 2.9\,C_x/\,\beta_n(Vdd - Vtn)}$
    - i.e. 2.9 time constants
- Therefore, the time to discharge Cx with an NMOS MP pass transistor is much shorter than the time to charge Cx due to the source-follower operation during charging.

Leakage and Subthreshold Current in Dynamic Pass Gate

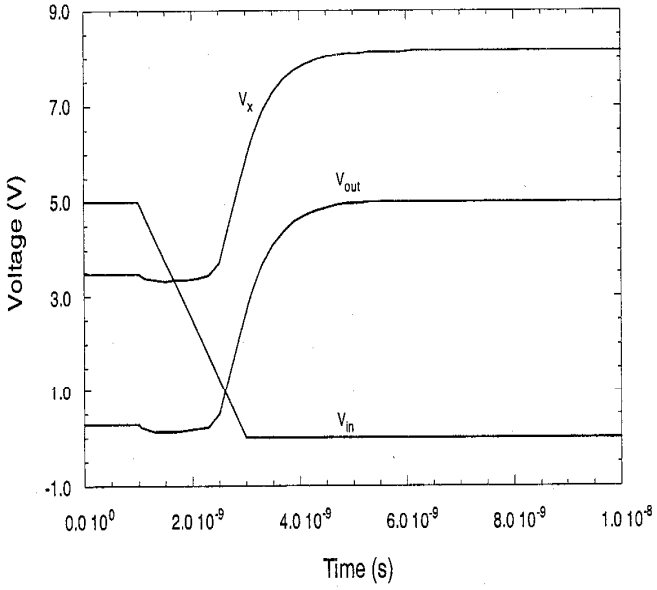
- Charge can leak off the storage capacitor Cx mainly from two sources:
  - PN junction leakage of the NMOS drain (source) junction
  - Subthreshold current (I_{OFF}) through MP when its gate is down at zero volts
- One can solve for the maximum amount of time Δt that charge can be retained on Cx using the differential equation C dv/dt = I, where
  - I is the total of the reverse PN junction leakage and the I_{OFF} current
  - C is the total load capacitance due to gate, junction, wire, and poly capacitance
  - the maximum allowable ΔV in order to preserve the logic "1" level is known
    - Typically ΔV ~ Vdd – Vtn – ½ Vdd = ½ Vdd – Vtn
- The minimum frequency of operation can be found from f ~ 1/(2 Δt)

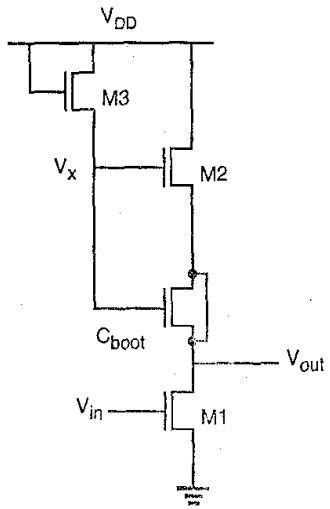


Dynamic Bootstrapping Technique



- **Bootstrapping** is a technique that is sometimes used to charge up a transistor gate to a voltage higher than Vdd when that transistor has to drive a line to the full Vdd
- At left is a NMOS bootstrap driver often used in memory circuits to drive a highly capacitive word line
- Operation:
  - When Vin = high, M1 is on holding Vout low while M3 charges Vx to Vdd – Vt. Thus, C_{boot} is charged to Vdd – Vt – V_{OL}
  - When Vin goes low, turning M1 off, M2 starts charging Vout high. If $C_{boot} > C_s$, most of the increase in Vout is "booted" to Vx, raising the voltage at Vx to well above Vdd.

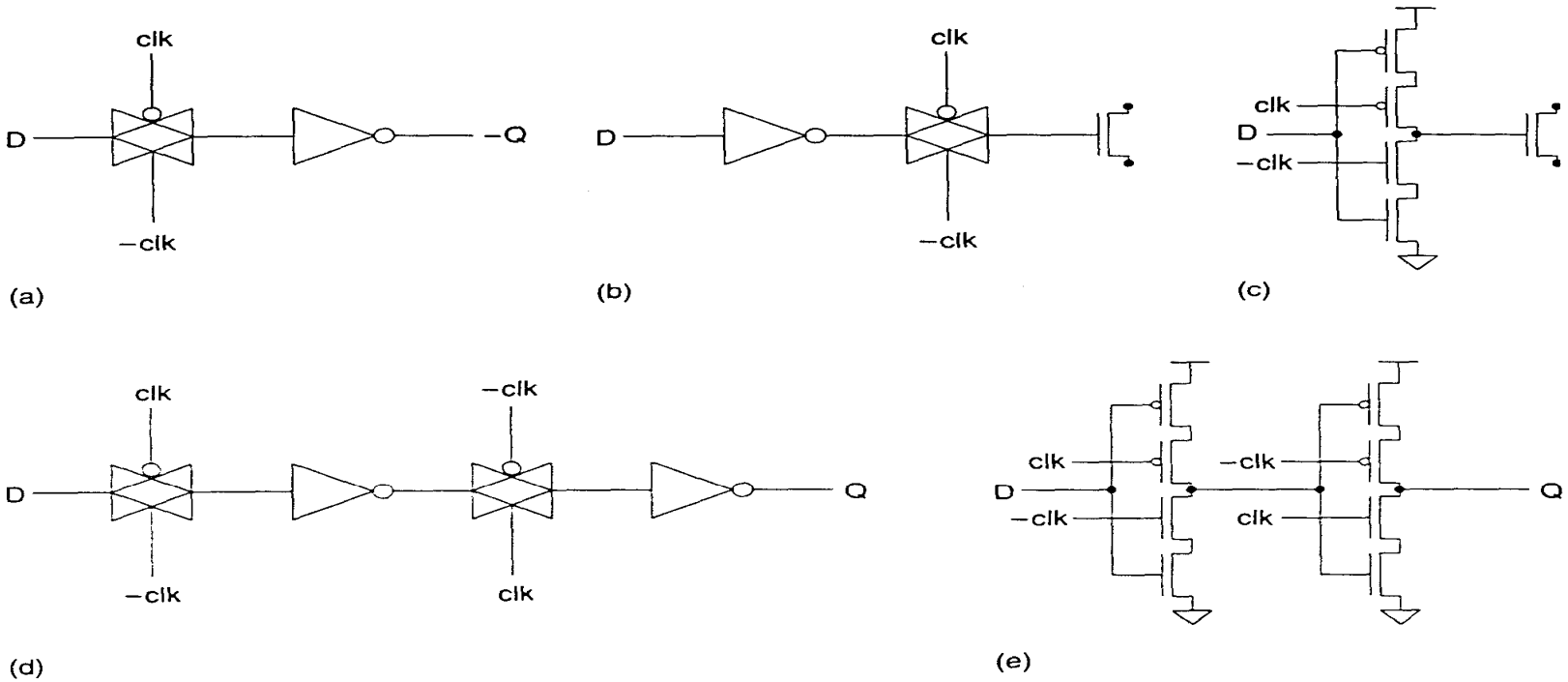




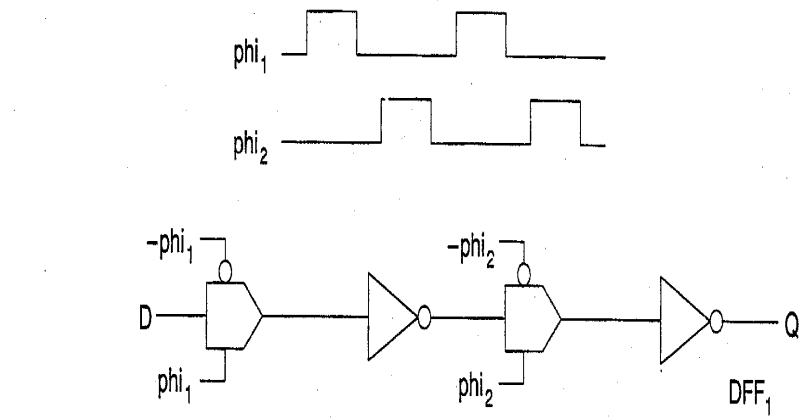
- It is desired to obtain Vx > Vdd + Vt in order to keep M2 linear, to allow Vout to be charged fully to Vdd.
- Parasitic capacitor C_s bleeds some of the charge off C_{boot}, limiting the max voltage on Vx (charging coupling eq.)
- At left C_{boot} is implemented with a transistor having source tied to drain.

Dynamic Latches with a Single Clock

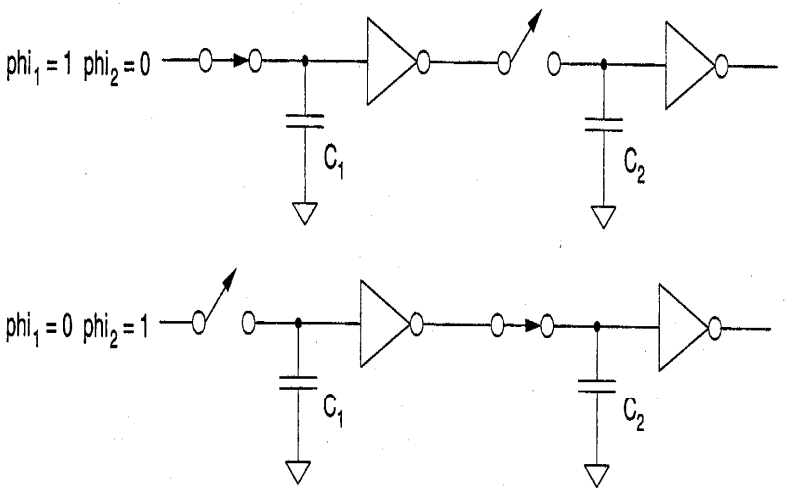
- Dynamic latches eliminate dc feedback leg by storing data on gate capacitance of inverter (or logic gate) and switching charge in or out with a transmission gate
  - Minimum frequency of operation is typically of the order of 50-100 KHz so as not to lose data due to junction or gate leakage from the node
  - Can be clocked at high frequency since very little delay in latch elements
- Examples:
  - (a) or (b) show simple transmission gate latch concept
  - (c) tri-state inverter dynamic latch holds data on gate when clk is high
  - (d) and (e) dynamic D register



Dynamic Registers with Two Phase Clocks



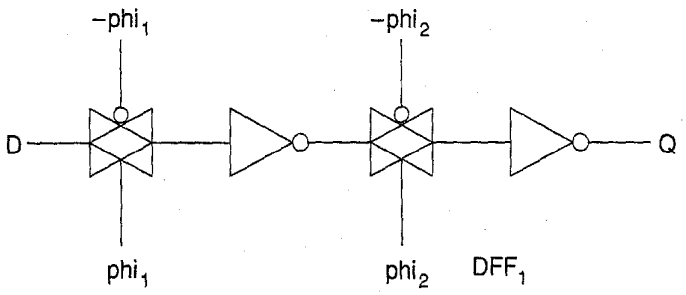
(a)

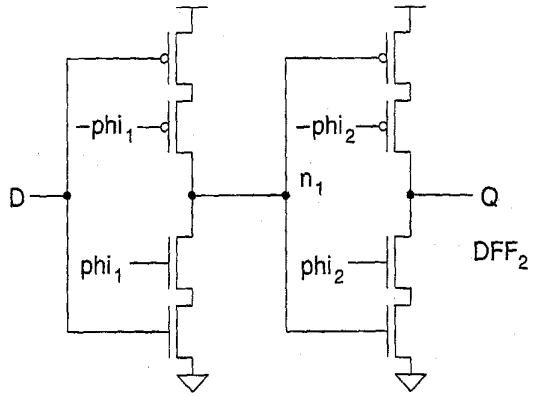


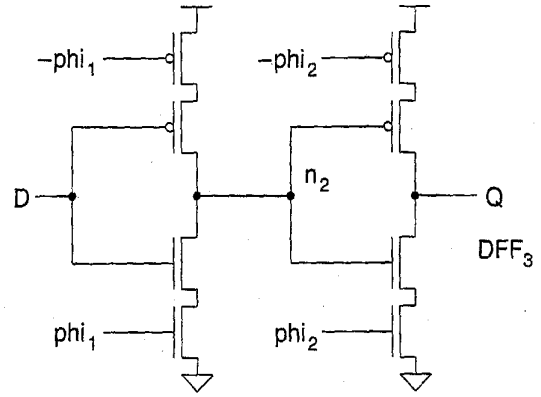
(b)

- Dynamic register with pass gates and two phase clocking is shown
  - Clocks phi1 and phi2 are non-overlapping
  - When phi1 is high & phi2 is zero,
    - 1st pass gate is closed and D data charges gate capacitance C1 of 1st inverter
    - 2nd pass gate is open trapping prior charge on C2
  - When phi1 is low and phi2 is high,
    - 1st pass gate opens trapping D data on C1
    - 2nd pass gate closes allowing C2 to charge with inverted D data
- If clock skew or sloppy rise/fall time clock buffers cause overlap of phi1 and phi2 clocks,
  - Both pass gates can be closed at the same time causing mixing of old and new data and therefore loss of data integrity!

Two Phase Dynamic Registers (Compact Form)



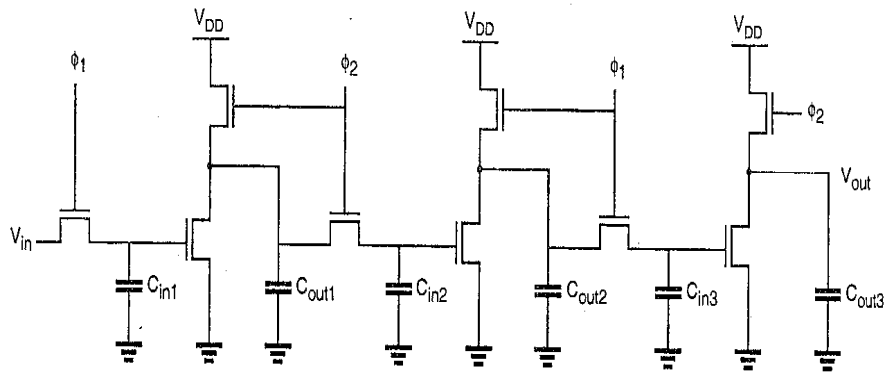


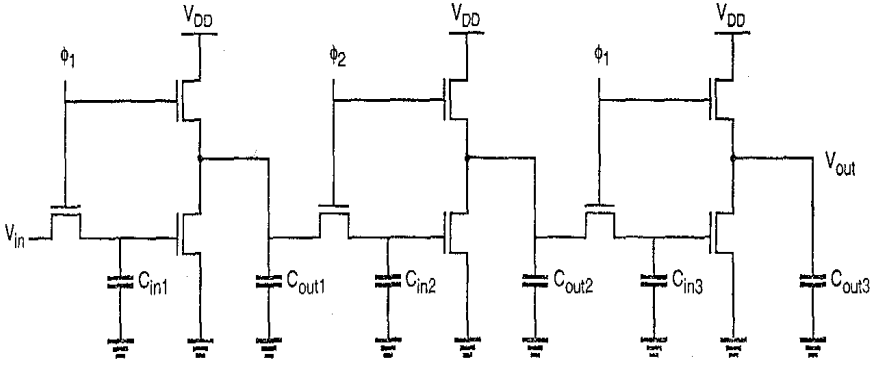


Both of these dynamic registers have to drive a local storage gate.

- Compact implementation of of two phase dynamic registers shown at left using a tri-state buffer form.
  - Transmission gate and inverter integrated into one circuit
  - Two versions:
    - Pass devices closest to output
    - Inverter devices closest to output
- Two phase dynamic registers and logic is often preferred over single phase because
  - Due to finite rise and fall times, the CLK and CLK' are not truly non-overlapping
  - Clock skew often is a problem due to the fact that CLK' is usually generated from CLK using an inverter circuit and also due to the practical problem of distributing clock lines without any skew

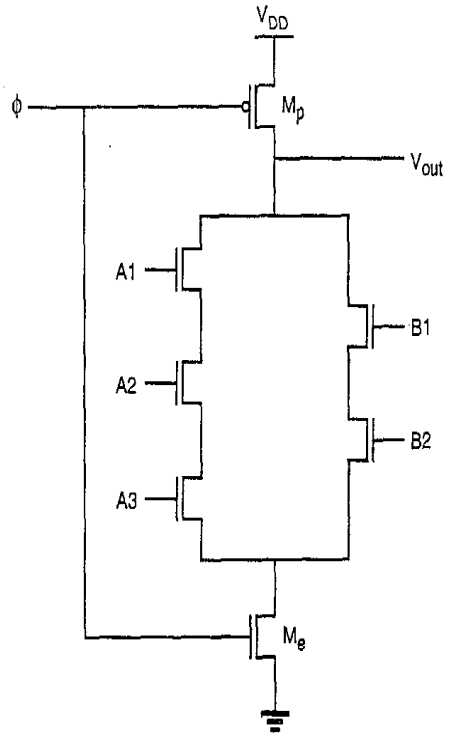
Dynamic Shift Registers with Enhancement Load

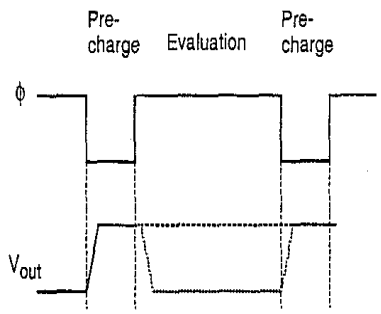




- At left (top) is a dynamic shift register implemented with a technique named "**ratioed dynamic logic**".
  - ϕ1 and ϕ2 are non-overlapping clocks
  - When ϕ1 is high, Cin1 charges to Vdd – Vt if Vin is high or to GND if Vin is low
  - When ϕ1 drops and ϕ2 comes up, the input data is trapped on Cin1 and yields a logic output on Cout1 which is transferred to Cin2
  - When ϕ2 drops and ϕ1 comes up again, the logic output on Cout1 is trapped on Cin2, which yields a logic output on Cout2, which is transferred to Cin3, etc.
  - To avoid losing too much voltage on the logic high level, $Cout_n >> Cin_{n+1}$ is desired
  - Each inverter must be ratioed to achieve a desired V_{OL} (e.g. when ϕ2 is high on 1st inv)
- The bottom left dynamic shift register is a "**ratioless dynamic logic**" circuit
  - When ϕ2 is high transferring data to stage 2, ϕ1 has already turned off the stage 1 load transistor, allowing a $V_{OL} = 0$ to be obtained without a ratio condition between load and driver transistors.

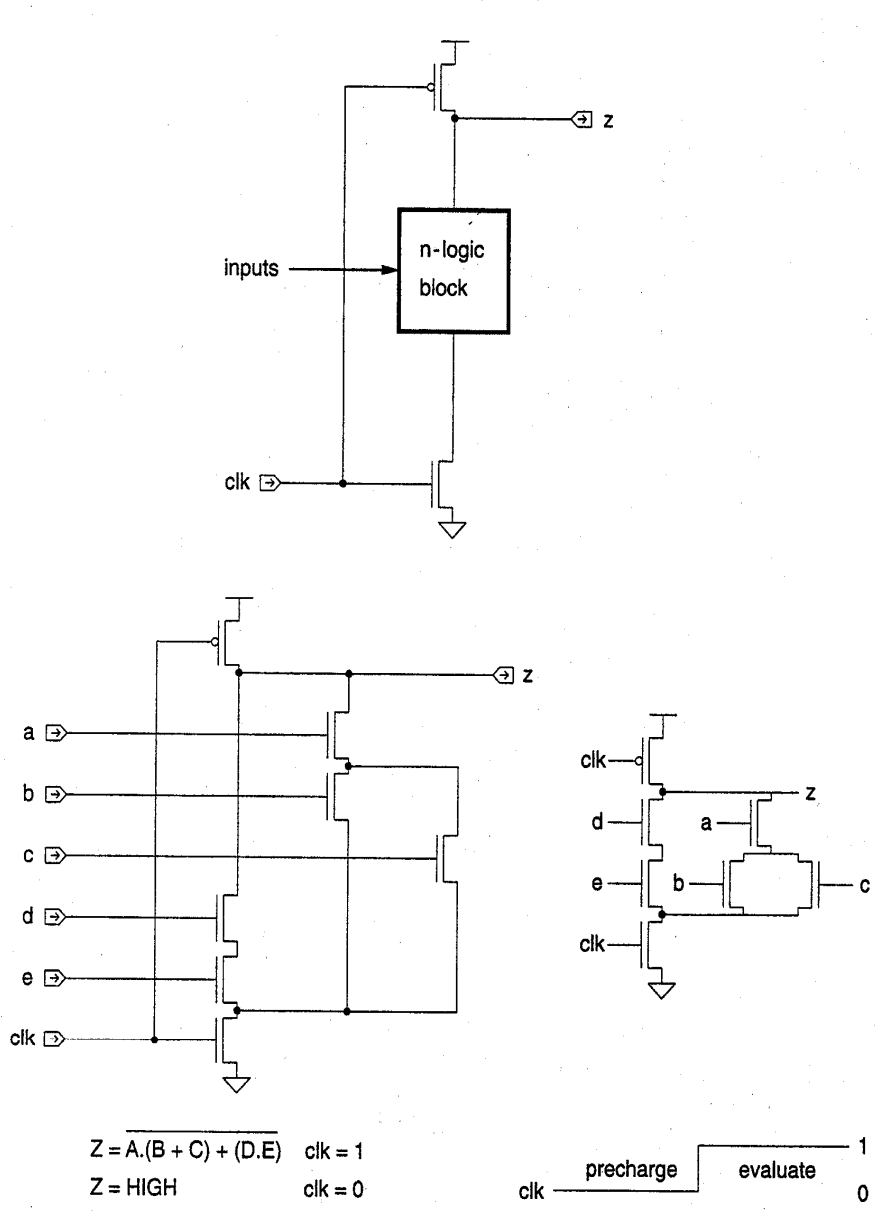
Dynamic CMOS Logic Gate





- In dynamic CMOS logic a single clock ϕ can be used to accomplish both the pre-charge and evaluation operations
  - When ϕ is low, PMOS pre-charge transistor Mp charges Vout to Vdd, since it remains in its linear region during final pre-charge
    - During this time the logic inputs A1 … B2 are active; however, since Me is off, no charge will be lost from Vout
  - When ϕ goes high again, Mp is turned off and the NMOS evaluate transistor Me is turned on, allowing for Vout to be selectively discharged to GND depending on the logic inputs
    - If A1 … B2 inputs are such that a conducting path exists between Vout and Me, then Vout will discharge to 0
    - Otherwise, Vout remains at Vdd

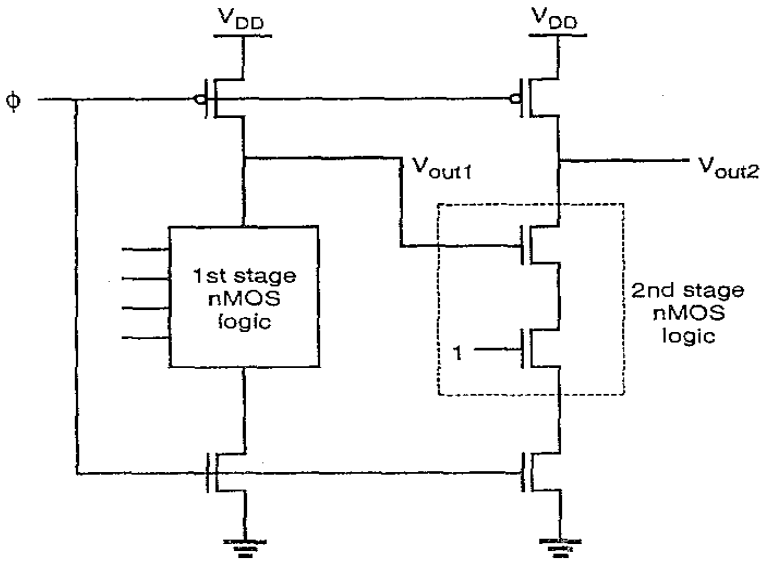
Dynamic CMOS Logic Circuits

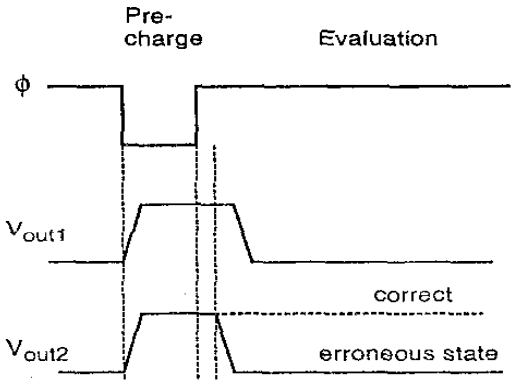


- Dynamic CMOS Logic circuits require a clock to precharge the output node and then to pull down the logic tree (assuming the logic inputs provide a path for current to flow)
  - Precharge Phase: clock is down turning on the P precharge transistor; N pull-down transistor is off. Output capacitance C_N charges to Vdd.
  - Evaluation Phase: clock goes high turning on the N pull down transistor and turning off the P precharge transistor. If logic inputs are such that neg Z is true, then output capacitance C_N discharges to ground.
  - No dc current flows during either the precharge or the evaluate phase.
  - Power is dynamic and is given by $P = C_N V_{dd}^2 f \alpha$ where C_N represents an equivalent total capacitance on the output, f = clock frequency, α =logic repetition rate

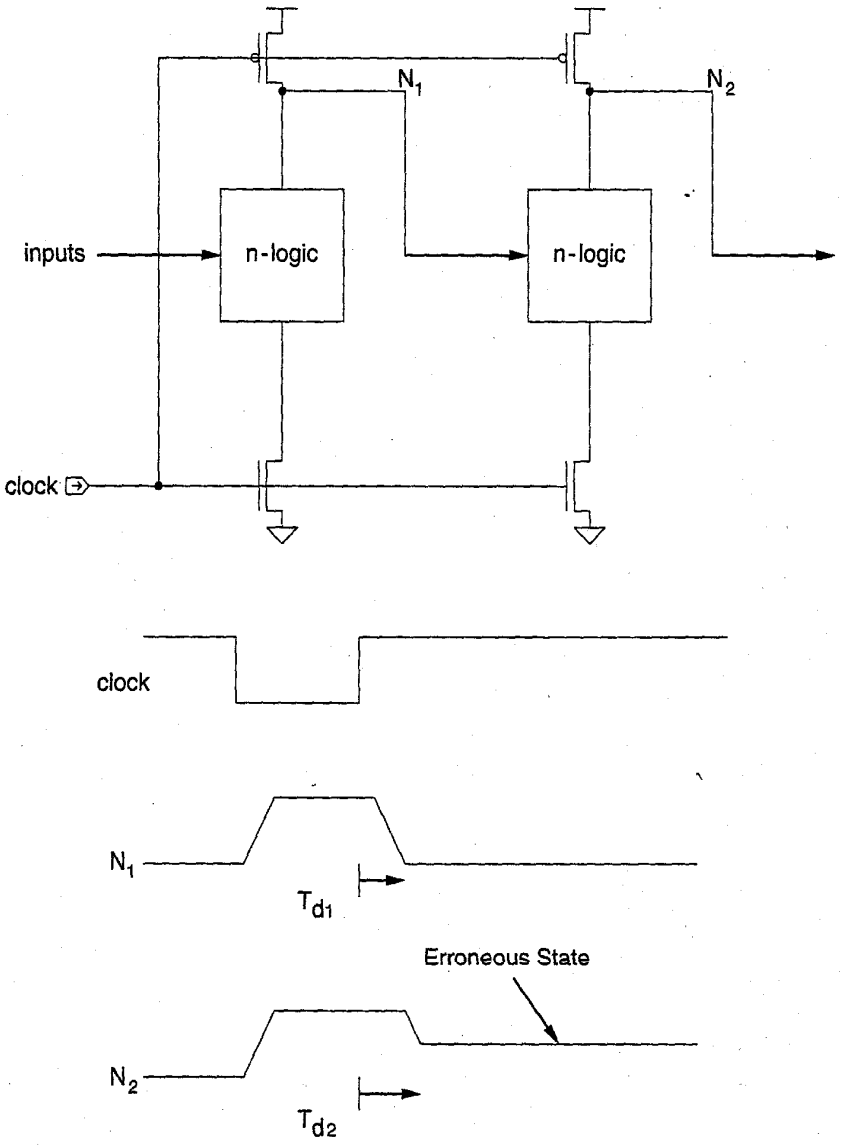
Cascading Problem in Dynamic CMOS Logic

- If several stages of the previous CMOS dynamic logic circuit are cascaded together using the same clock ϕ, a problem in evaluation involving a built-in "race condition" will exist
- Consider the two stage dynamic logic circuit below:
  - During **pre-charge**, both Vout1 and Vout2 are pre-charged to Vdd
  - When ϕ goes high to begin **evaluate**, all inputs at stage 1 require some finite time to resolve, but during this time charge may erroneously be discharged from Vout2
    - e.g. assume that eventually the 1st stage NMOS logic tree conducts and fully discharges Vout1, but since all the inputs to the N-tree all not immediately resolved, it takes some time for the N-tree to finally discharge Vout1 to GND.
    - If, during this time delay, the 2nd stage has the input condition shown with bottom NMOS transistor gate at a logic 1, then Vout2 will start to fall and discharge its load capacitance until Vout1 finally evaluates and turns off the top series NMOS transistor in stage 2
  - The result is an error in the output of the 2nd stage Vout2



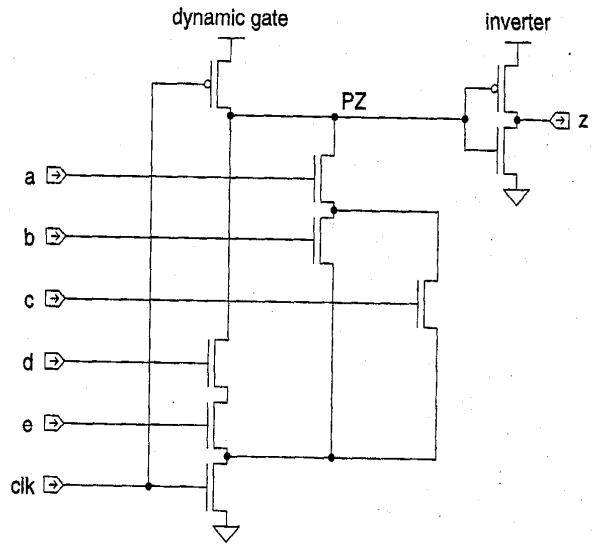


Cascaded Dynamic CMOS Logic Gates: Evaluate Problem

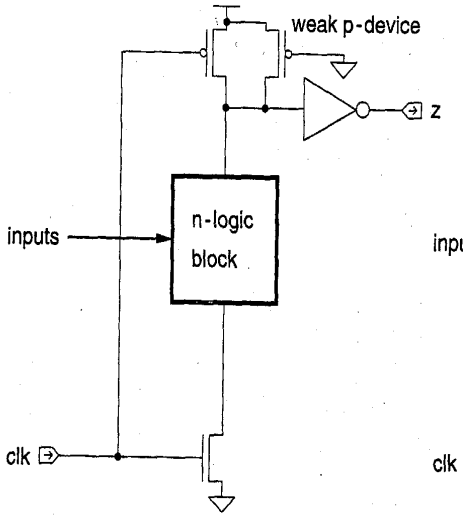


- With simple cascading of dynamic CMOS logic stages, a problem arises in the evaluate cycle:
  - The pre-charged high voltage on Node N2 in stage 2 may be inadvertently (partially) discharged by logic inputs to stage 2 which have not yet reached final correct (low) values from the stage 1 evaluation operation.
  - Can not simply cascade dynamic CMOS logic gates without preventing unwanted bleeding of charge from pre-charged nodes
- Possible Solutions:
  - two phase clocks
  - use of inverters to create Domino Logic
  - NP Domino Logic
  - Zipper/NORA logic

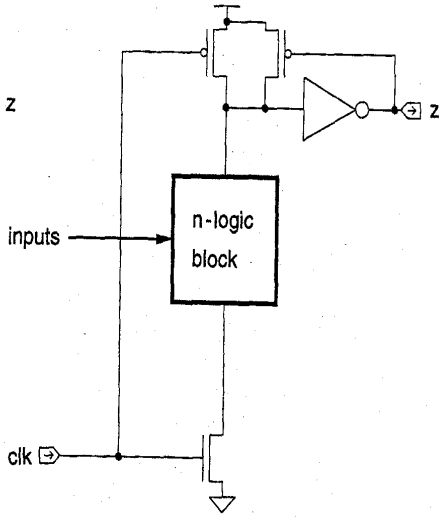
CMOS Domino Logic



(a)



(b)

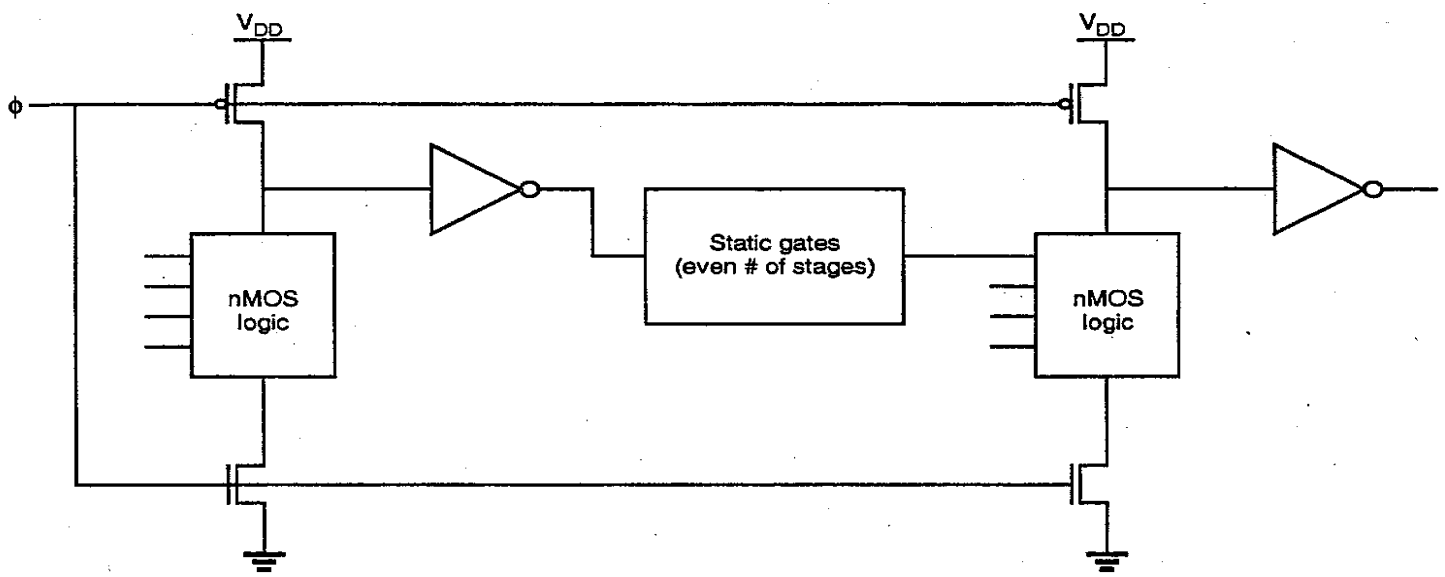


(c)

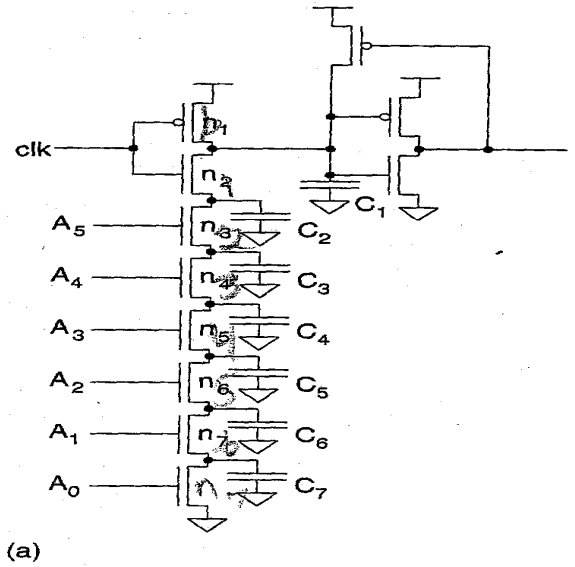
- The problem with faulty discharge of precharged nodes in CMOS dynamic logic circuits can be solved by placing an inverter in series with the output of each gate
  - All inputs to N logic blocks (which are derived from inverted outputs of previous stages) therefore will be at zero volts during precharge and will remain at zero until the evaluation stage has logic inputs to discharge the precharged node PZ.
  - This circuit approach avoids the race problem of “vanilla” cascaded dynamic CMOS
  - However, all circuits only provide non-inverted outputs
- In (b) a weak P device compensates for charge loss due to charge sharing and leakage at low frequency clock operation
- In (c) the weak P device can be used to latch the output high

Mixing Domino CMOS Logic with Static CMOS Logic

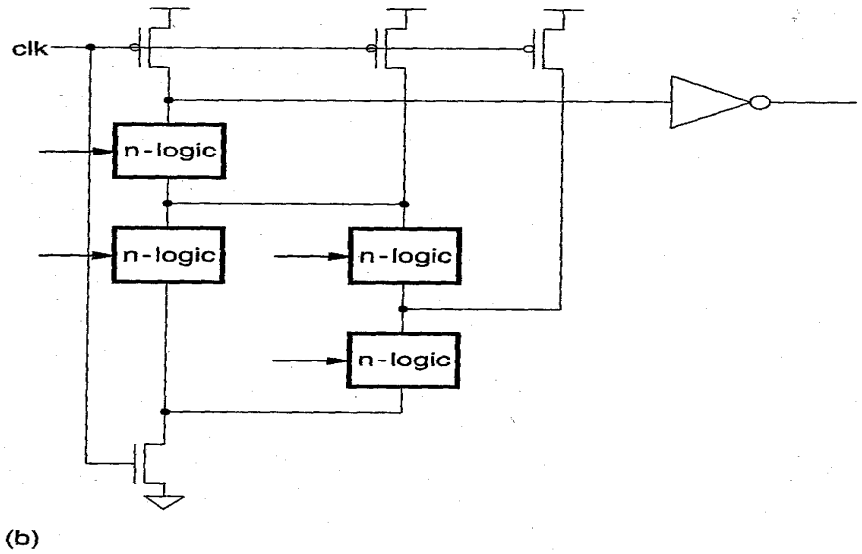
- We can add an **even** number of static CMOS inverting logic gates after a Domino logic stage prior to the next Domino logic stage
  - Even number of inverting stages guarantees that inputs to the second Domino logic stage experience only 0-to-1 transitions (since 1-to-0 transitions may cause an erroneous logic level as discussed in prior charts 5-67 and 5-68)
- In the cascaded **Domino** logic structure, the evaluation of each stage ripples through the cascaded stages similar to a chain of Dominos (from which it takes the name)
  - The evaluate cycle must be of sufficient duration to allow all cascaded logic stages (between latches) to complete their evaluation process within the **clock evaluation interval**



CMOS Domino Logic Design Hazards



(a)

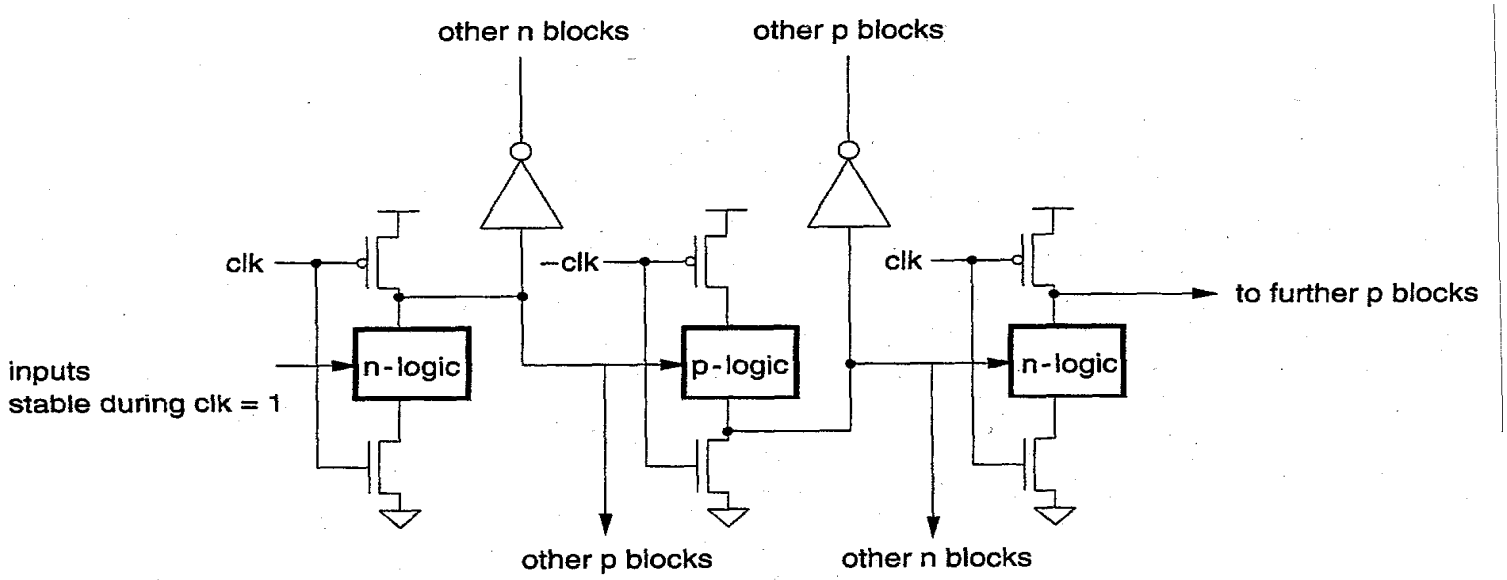


(b)

- In (a) the N evaluate transistor is placed nearest to the output C1 node (poor design)
  - During precharge C1 is charged high to Vdd, but C2-C7 do not get charged and may be sitting at ground potential.
  - When the clock goes high for the evaluate phase, some or all of capacitors C2-C7 will bleed charge from the larger node capacitor C1, thus reducing the voltage on C1.
    - VC1 may reduce to Vdd(C1/(C1 + C2 + C3 + C4 + C5 + C6 + C7)) in the worst case
  - The solution is to put the discharge transistor N1 at the bottom of the logic tree thus allowing the possibility of getting C2-C7 charged during the precharge phase
- Using additional precharge P transistors (as in b) to charge intermediate nodes in a complex logic tree will help with the charge sharing problem.

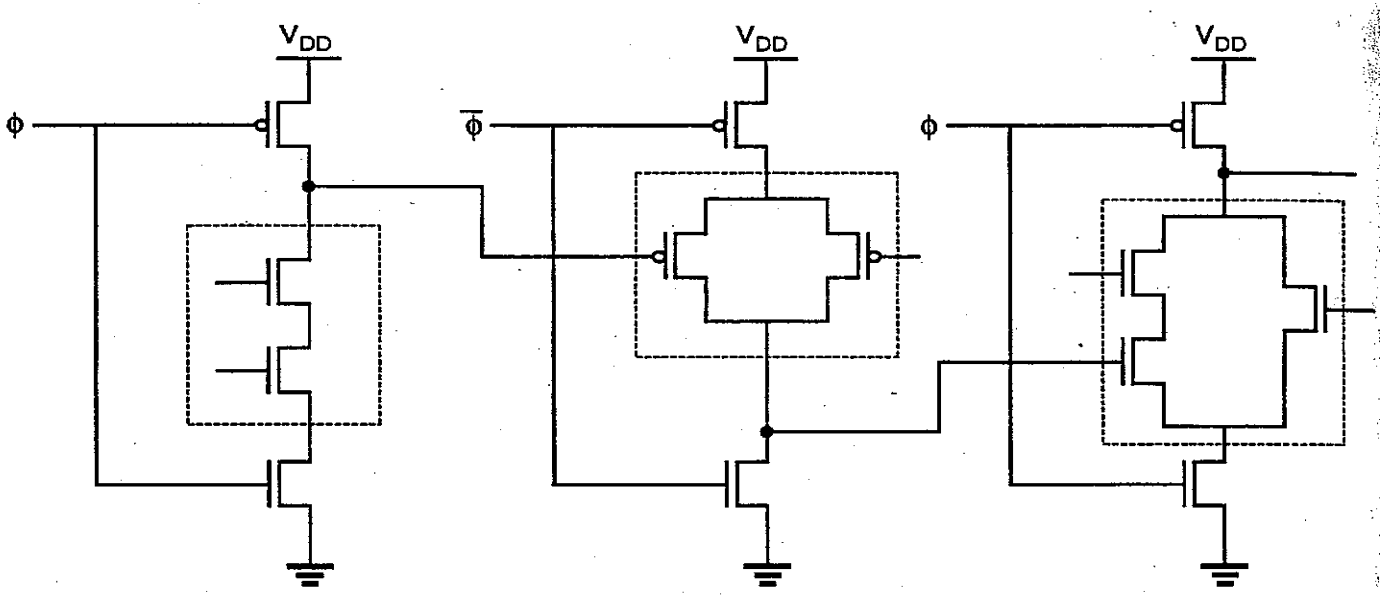
NP Domino Logic (NORA Logic)

- An elegant solution to the dynamic CMOS logic "erroneous evaluation" problem is to use NP Domino Logic (also called NORA logic) as shown below.
  - Alternate stages of N logic with stages of P logic
    - N logic stages use true clock, normal precharge and evaluation phases, with N logic tree in the pull down leg. P logic stages use a complement clock, with P logic stage tied above the output node.
    - During precharge clk is low (-clk is high) and the P-logic output precharges to ground while N-logic outputs precharge to Vdd.
    - During evaluate clk is high (-clk is low) and both type stages go through evaluation; N-logic tree logically evaluates to ground while P-logic tree logically evaluates to Vdd.
- Inverter outputs can be used to feed other N-blocks from N-blocks, or to feed other P-blocks from P-blocks.

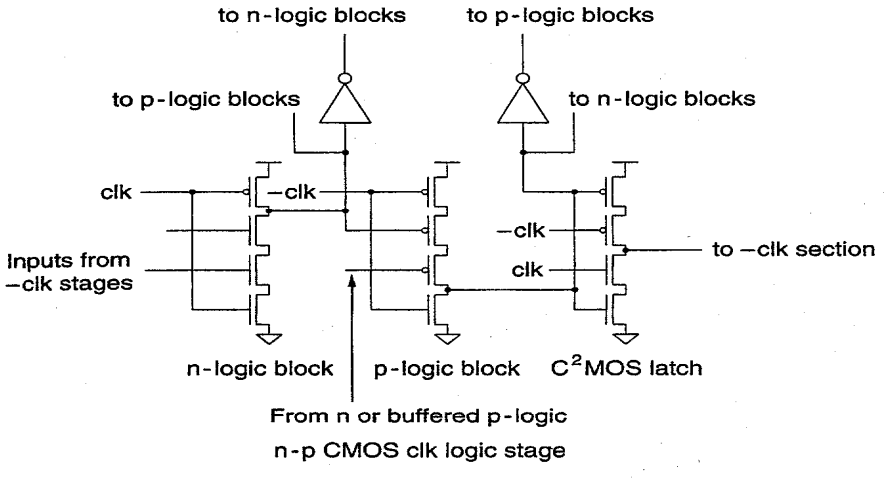


NORA CMOS Logic Circuit Example

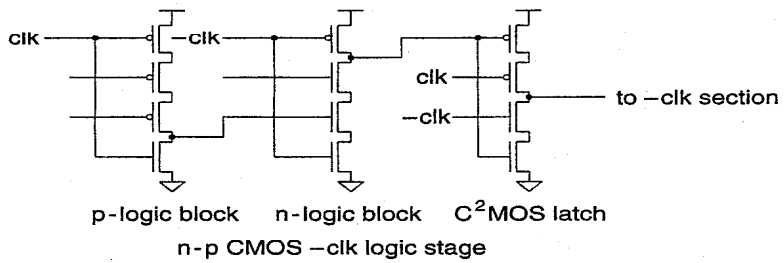
- An example of NP or NORA (No Race) logic is shown below:
- During ϕ low (ϕ' high), each stage pre-charges
  - N logic stages pre-charge to Vdd; P logic stages pre-charge to GND
- When ϕ goes high (ϕ' low), each stage enters the evaluation phase
  - N logic evaluates to GND; P logic stages evaluate to Vdd
  - All NMOS and PMOS stages evaluate one after another in succession, as in Domino logic
- Logic below:
  - Stage 1 is X = (A · B)'
  - Stage 2 is G = X' + Y'
  - Stage 3 is Z = (F · G + H)'



Single-Phase NP Dynamic Logic Structures



(a)



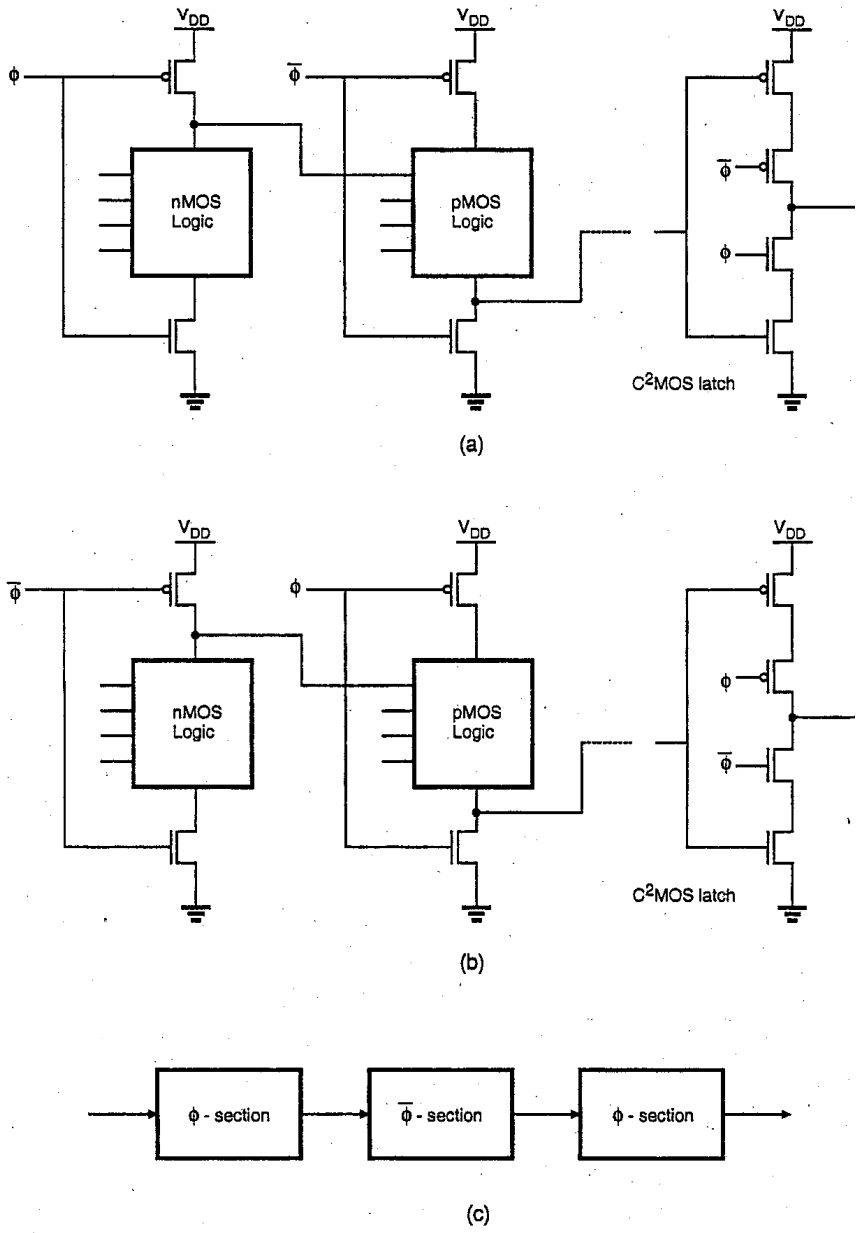
(b)

clk	–clk			
0	1	evaluation	precharge	evaluation
1	0	precharge	evaluation	precharge

(c)

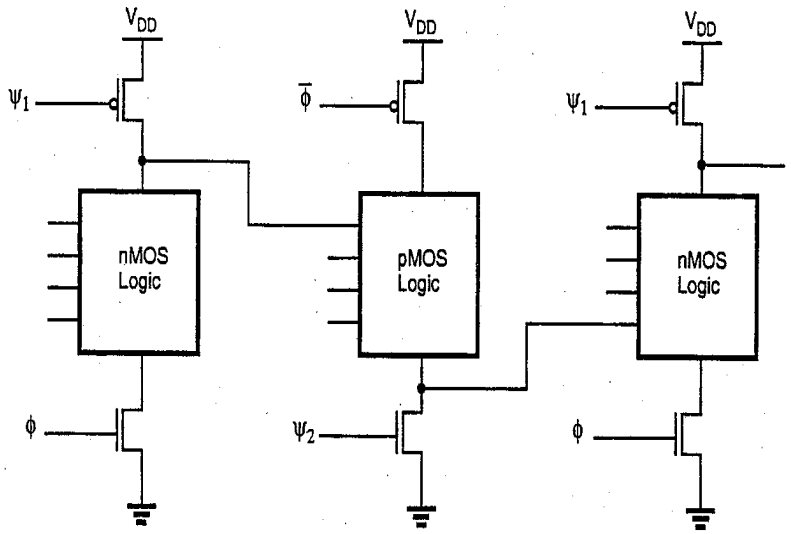
- Combines NP Domino logic sections with C^2MOS latch
  - n-logic block can drive p-logic block or another n-logic block with a static inverter
  - similarly for a p-logic block
  - Must end in a C^2MOS latch
- clk logic: (a) prechrg on clk=0, eval clk=1
- -clk logic: (b) pre on clk=1, eval on clk=0
- clk logic can feed –clk logic & vice-versa
- can mix static logic with NP domino logic
- Rules to avoid race conditions:
  - During precharge, logic blocks are OFF
  - During eval, internal inputs make only one transition
- Pipeline design:
  - Even # of inversions between C^2MOS, or
  - at least 1 dynamic stage and even # inversions prior to it

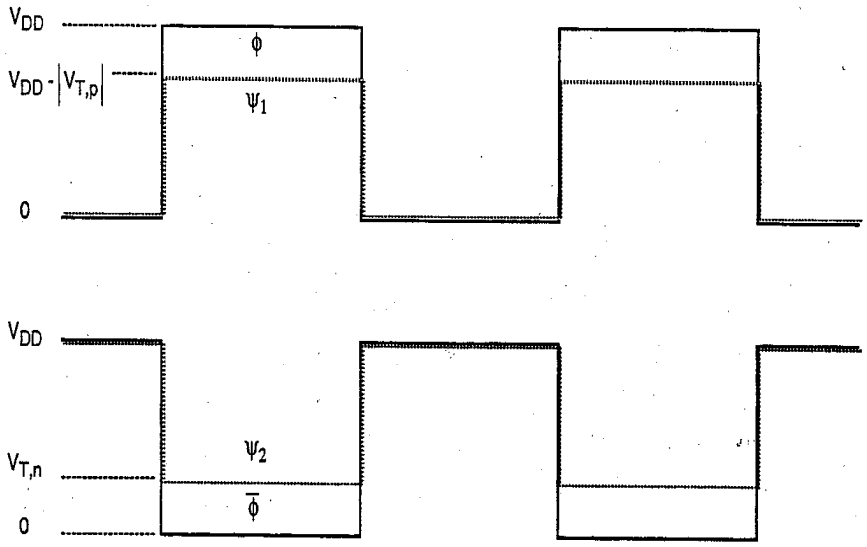
Pipelined NORA CMOS Circuit Operation



- With pipelined NORA CMOS logic design
  - one can alternate N and P stages between C^2MOS latches where ϕ high is used for evaluation as shown in (a)
  - Or, one can alternate N and P stages similarly between C^2MOS latches with ϕ' high used for evaluation as in (b)
  - ϕ sections may be alternately cascaded with ϕ' sections as shown in (c)
- During the evaluation phase, the logic ripples through each stage in succession up to the next C^2MOS latch

Zipper CMOS Dynamic Logic

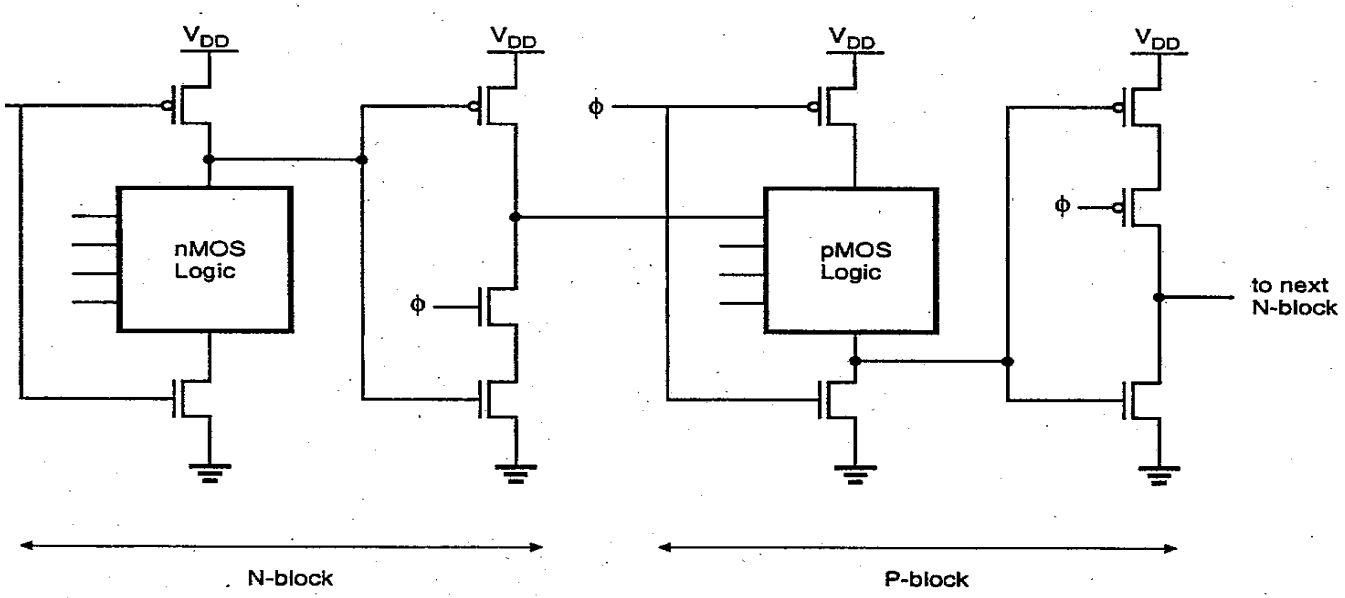




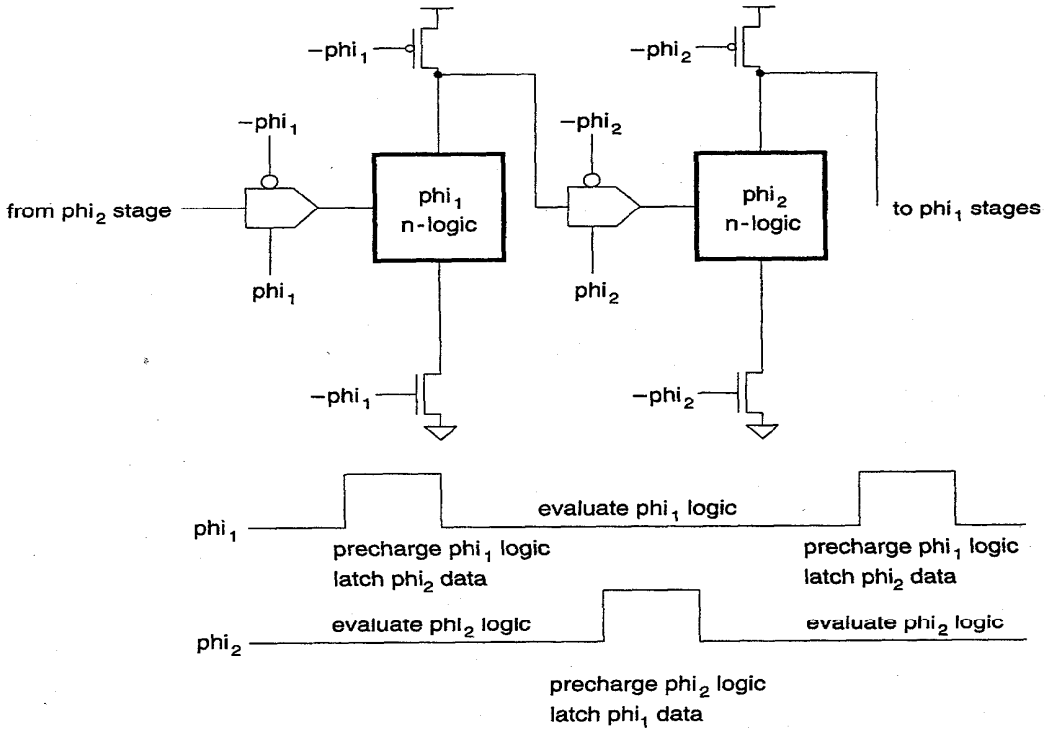
- **Zipper CMOS logic** is a scheme for improving charge leakage and charge sharing problems
- Pre-charge transistors receive a slightly modified clock where the clock pulse (during pre-charge off time) holds the pre-charge transistor at weak conduction in order to provide a trickle pre-charge current during the evaluation phase
  - PMOS pre-charge transistor gates are held at Vdd - |Vtp|
  - NMOS pre-charge transistor gates are held at Vtn above GND

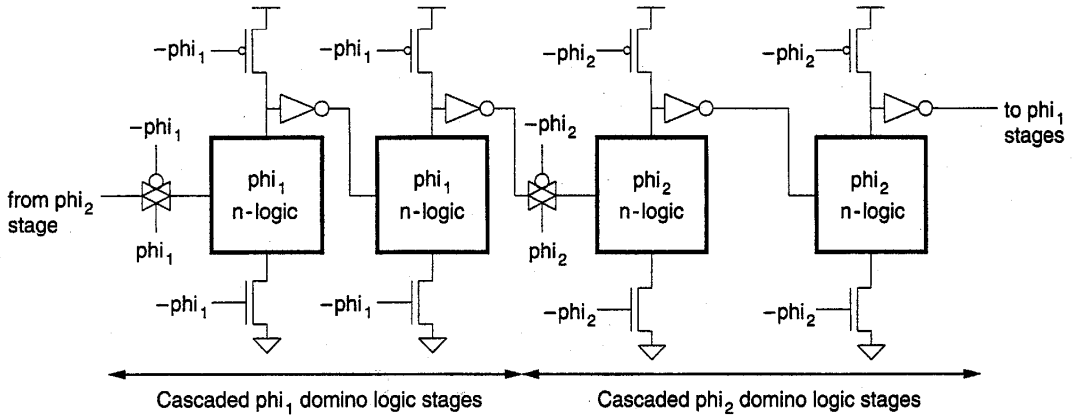
Pipelined True Single Phase Clock (TSPC) CMOS

- A true single phase clock system (without any inverted clocks required) can be built as shown below
- Each NMOS and PMOS stage is followed by a dynamic latch (inverter) built with only the single phase clock ϕ
- The single phase clock ϕ is used for both NMOS and PMOS stages
  - NMOS logic stages pre-charge when ϕ is low and evaluate when ϕ is high
  - PMOS logic stages pre-charge when ϕ is high and evaluate when ϕ is low
- With inverter latches between each stage, an erroneous evaluate condition can not exist
- Attractive circuit for use in pipelined, high performance processor logic



Two-Phase Dynamic Logic

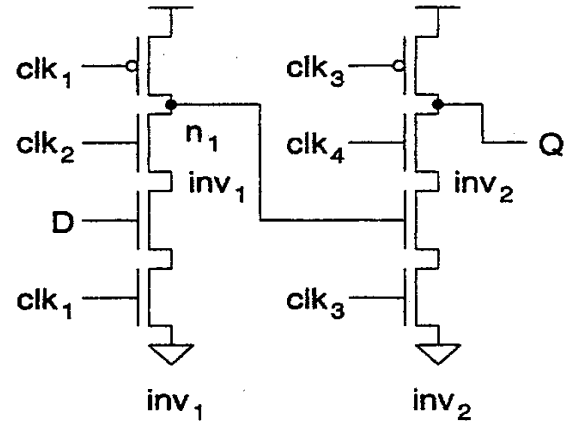




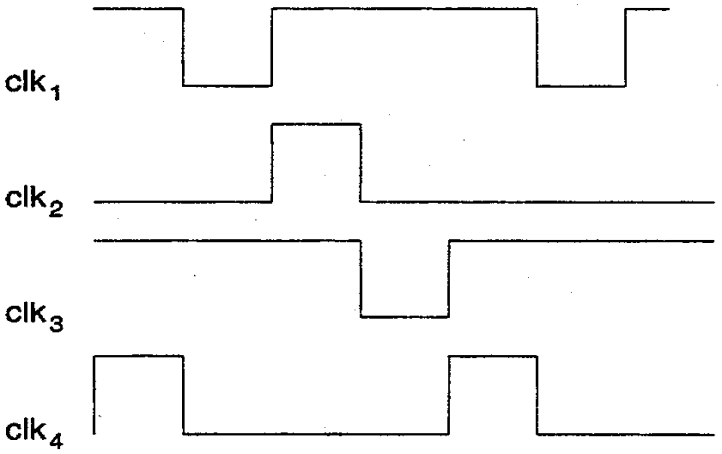
- Two phase dynamic logic similar to two phase dynamic register circuits
- Top figure shows n type logic stages with two phase non-overlapping clocks
  - phi1 high: precharge phi1 logic, evaluate phi2 logic
  - phi2 high: precharge phi2 logic, evaluate phi1 logic
- Bottom figure shows use of Domino logic having both phi1 and phi2 logic stages
  - Each block is separated from other by a clocked pass gate register/latch to store the logic result
  - Note that inverters must be used between successive stages of the same clock logic

Four Phase Clocking and Registers

- Four phase logic adds an evaluation phase to the existing precharge and evaluation phases of two-phase structures.
- Simple four-phase structure below illustrates operation:
  - during clk1 inverter1 is in **precharge** phase; node n1 charges to Vdd
  - during clk2 inverter1 **evaluates** since both NFET devices in n-tree leg are ON
  - during clk3 inverter2 precharges and inverter1 is in **hold** phase (i.e. both N and P devices are OFF isolating node n1
  - during clk4 inverter2 evaluates while inverter1 continues in **hold** phase
- Note that the hold phase is really two clock phases long
- Due to charge sharing during the clk2 phase, clk2 is sometimes replaced by clk12 (and clk4 is replaced by clk34) by keeping clk12 high during both clk1 and clk2 phases.

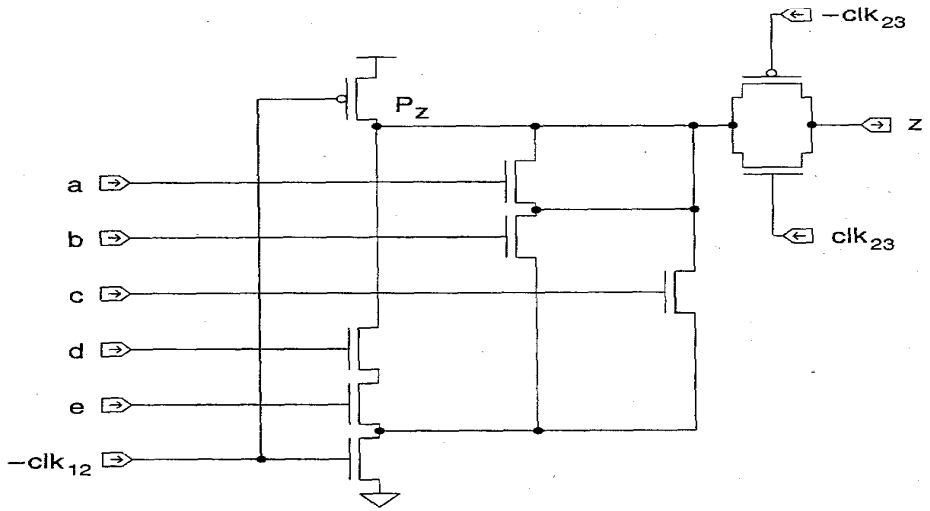


(a)

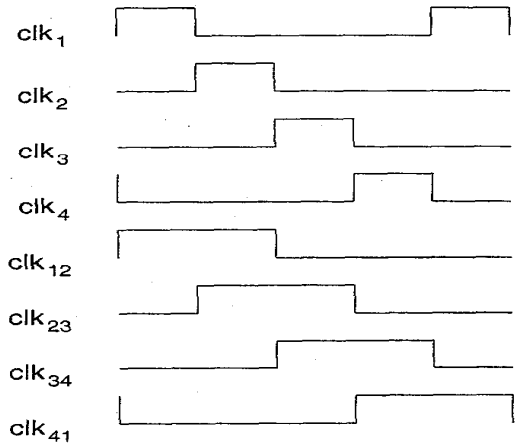


Four-Phase Logic Structures

- Four phase logic structure shown using transmission gate to isolate data on z during hold time
- Operation:
  - during clk1 time, -clk12 is down causing Pz to be precharged to Vdd
  - during clk2 time, -clk12 is still down keeping precharge active, but clk23 goes high thus precharging node z
  - during clk3 time, precharge of node Pz ends and evaluation begins with Xgate still closed
  - during clk4 time, the transmission gate opens and the correct data is isolated on node z
- For the gate shown (Type 3), z is valid during phases 4 and 1

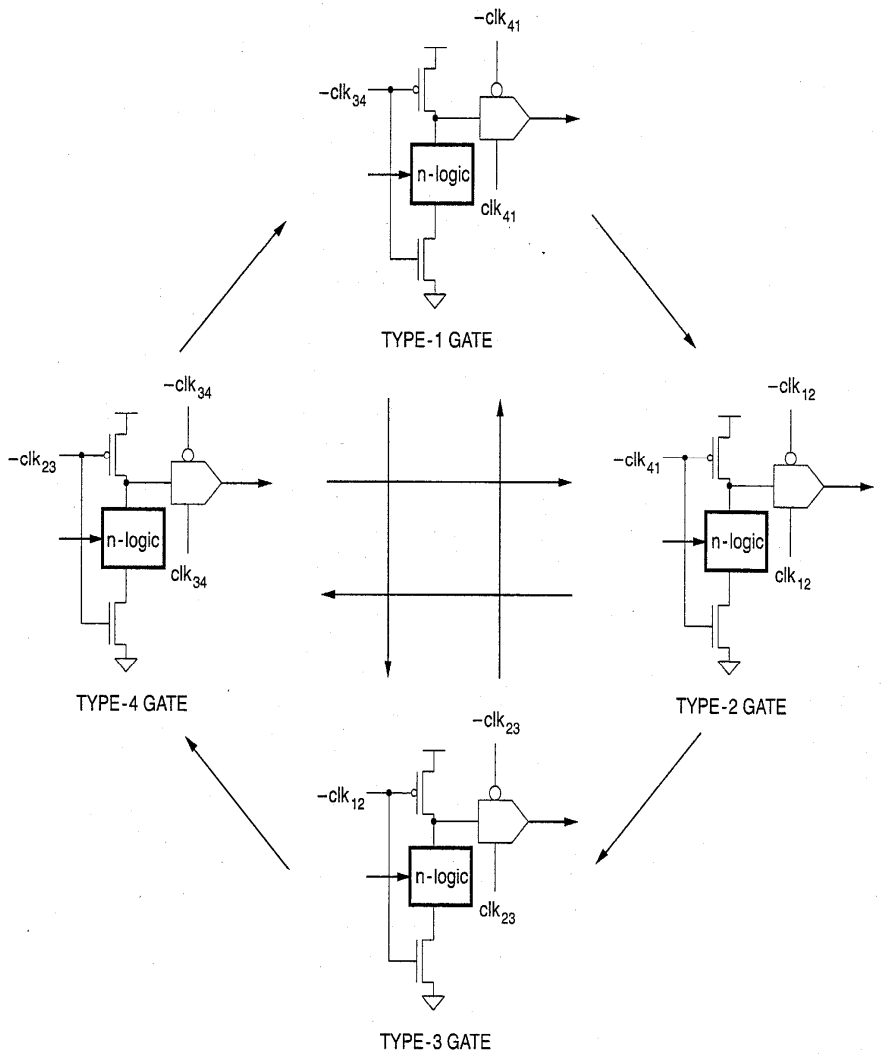


(a)



(b)

Four Phase Logic: Allowable Interconnections

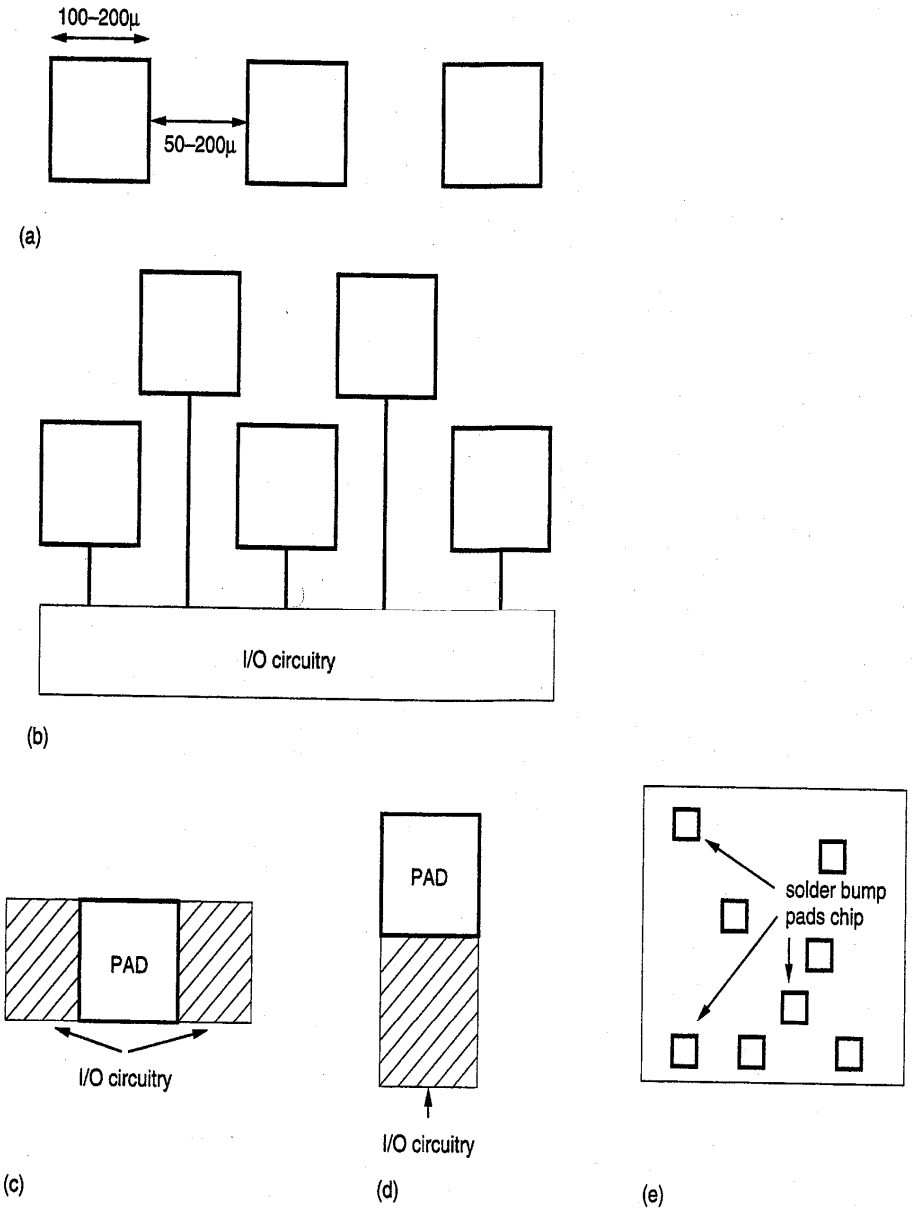


- Using four different type logic gates as shown in previous chart (where Type refers to the evaluation phase time), four phase logic can be used in pipelined logic structures where each type must be used per the allowable interconnection diagram at the left
  - a Type 1 gate can feed Type 2 or Type 3 gates
  - a Type 2 gate can feed Types 3 and 4
  - a Type 3 gate can feed Types 4 and 1
  - a type 4 gate can feed Types 1 and 2

Two-Phase Clock Generator

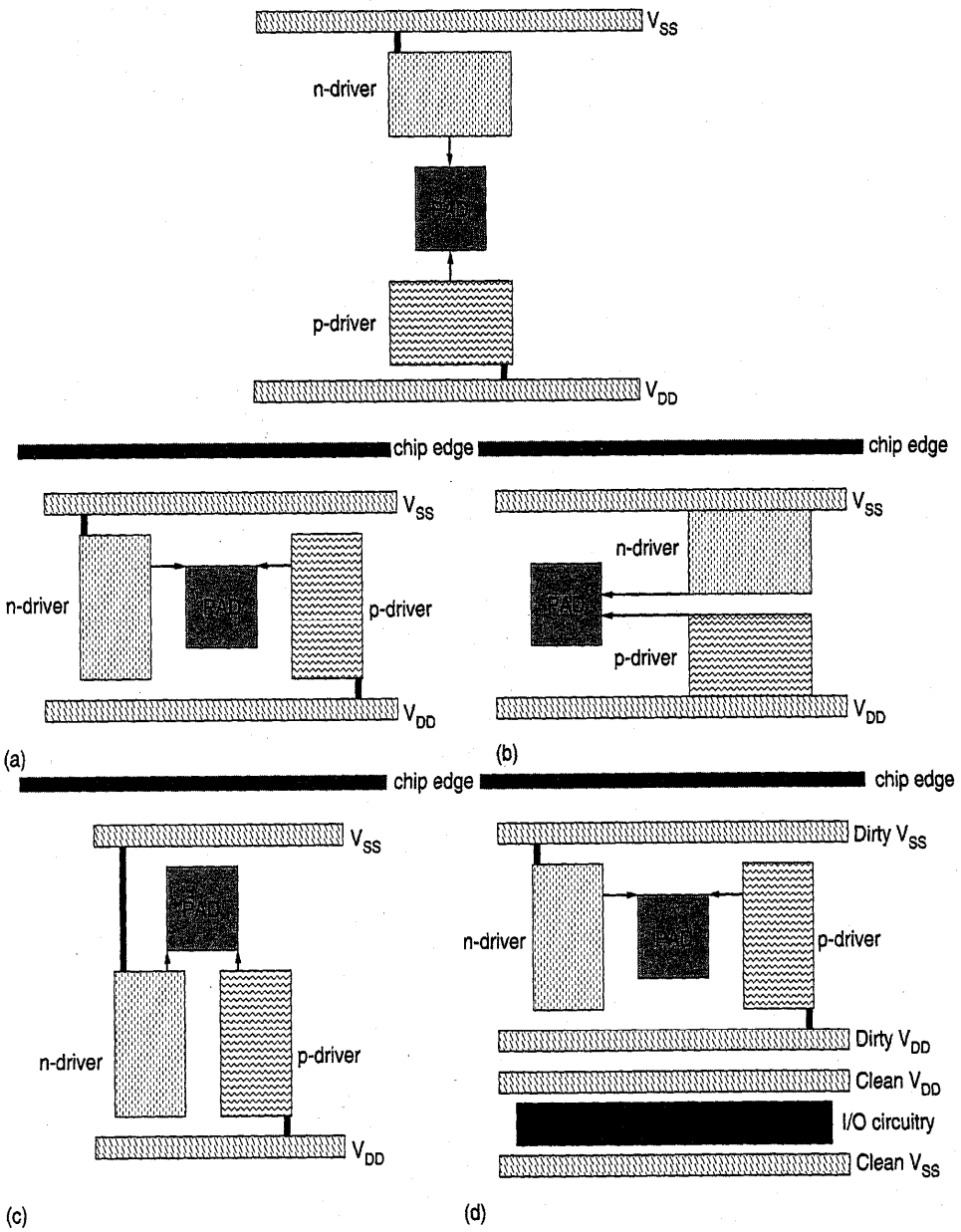
- Phi1 and Phi2 clocks may be generated from a master clock using a two-phase clock generator circuit
  - RS type cross-coupled latch with delay built into each feedback loop
  - Use an even number of inverters in each feedback loop
    - The delay built into the feedback loop sets the non-overlap period in the two out-of-phase clocks
  - NOR (or NAND) gates used to synchronize the generator with a master clock input
- Alternately, it may be desired to bring both phases on the chip as inputs and distribute both clocks globally

Options for Input/Output Pad Layout



- Pad size and pitch determined by minimum area needed to bond a wire to each pad and keep wires from shorting
- Options:
  - (a) single row of pads around chip periphery
  - (b) double row of perimeter pads used for high pad count
  - (c) chip size limited by internal circuitry: put I/O circuits on sides of pads
  - (d) pad-limited chip size: put I/O circuitry inside pads
  - (e) area pads such as are used in solder bump flip-chip packaging
    - Entire chip area can be used to distribute several hundred pads (solder balls)

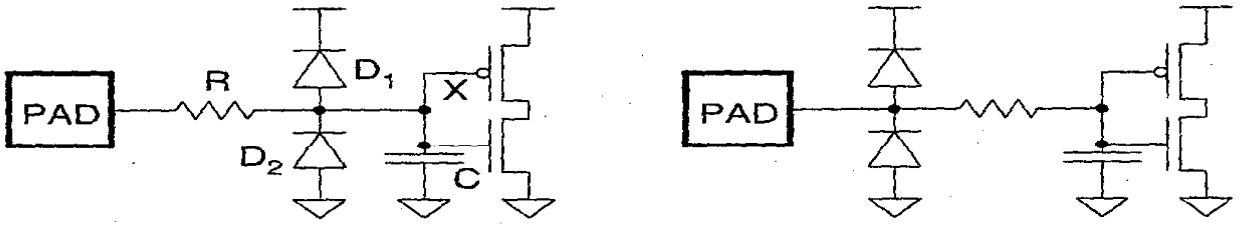
I/O Pad Layout Detail



- I/O Pad Layout Detail Options:
  - Note proximity of Vss power bus to the n-driver device and the Vdd bus to the p-driver device
  - Large output driver transistors oriented either N-S or E-W of I/O pad depending on pad ring density requirements
  - General practice is to keep Vss bus closest to chip edge
  - Some chip designs separate Vss and Vdd buses into a dirty supply bus for providing power to off-chip drivers and clean supply buses for providing “glitch-free” voltages for input receivers, latches, and internal circuitry
    - Reflections on output lines can couple severe noise glitches onto the n-driver and p-driver supply buses

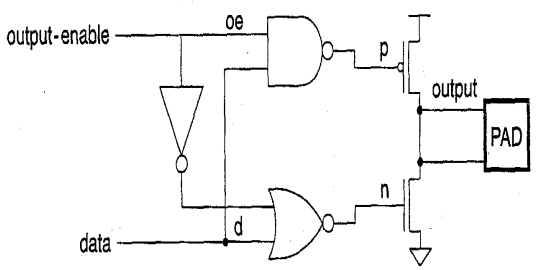
I/O Pad Electrostatic Discharge Protect Device

- ESD protect circuits are a requirement on most inputs in order to protect against gate oxide breakdown on input MOS transistors due to electrostatic charge buildup
  - ESD phenomenon can cause a voltage spike of 100's of volts under certain conditions of low humidity by a human touching a critical input pad/pin (ex. walking across a carpet in winter)
- ESD protection achieved in various ways
  - N and P diodes on the input clamp the coupled ESD voltage spike to a forward diode drop above Vdd and below Vss
    - High speed diodes are a requirement
    - Formed by FET device drain regions with the gate tied to Vdd (p-FET) and to Vss (n-FET)
    - Double guard ring structures needed to prevent latchup due to reflections causing forward biasing
  - Series resistance used to limit current & dissipate transient power during ESD event
    - 200 ohms to several K ohms may be used
  - In bi-di I/O's with bi-directional driver/receiver circuits, the diodes D1 and D2 are provided by the drain regions of the n and p driver output transistors

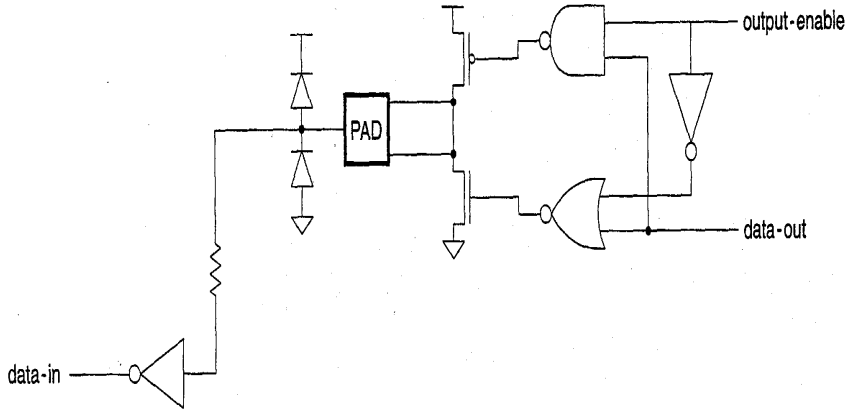


typical input-protection circuits

Tri-State and Bi-Di Driver Pad Design



(a)



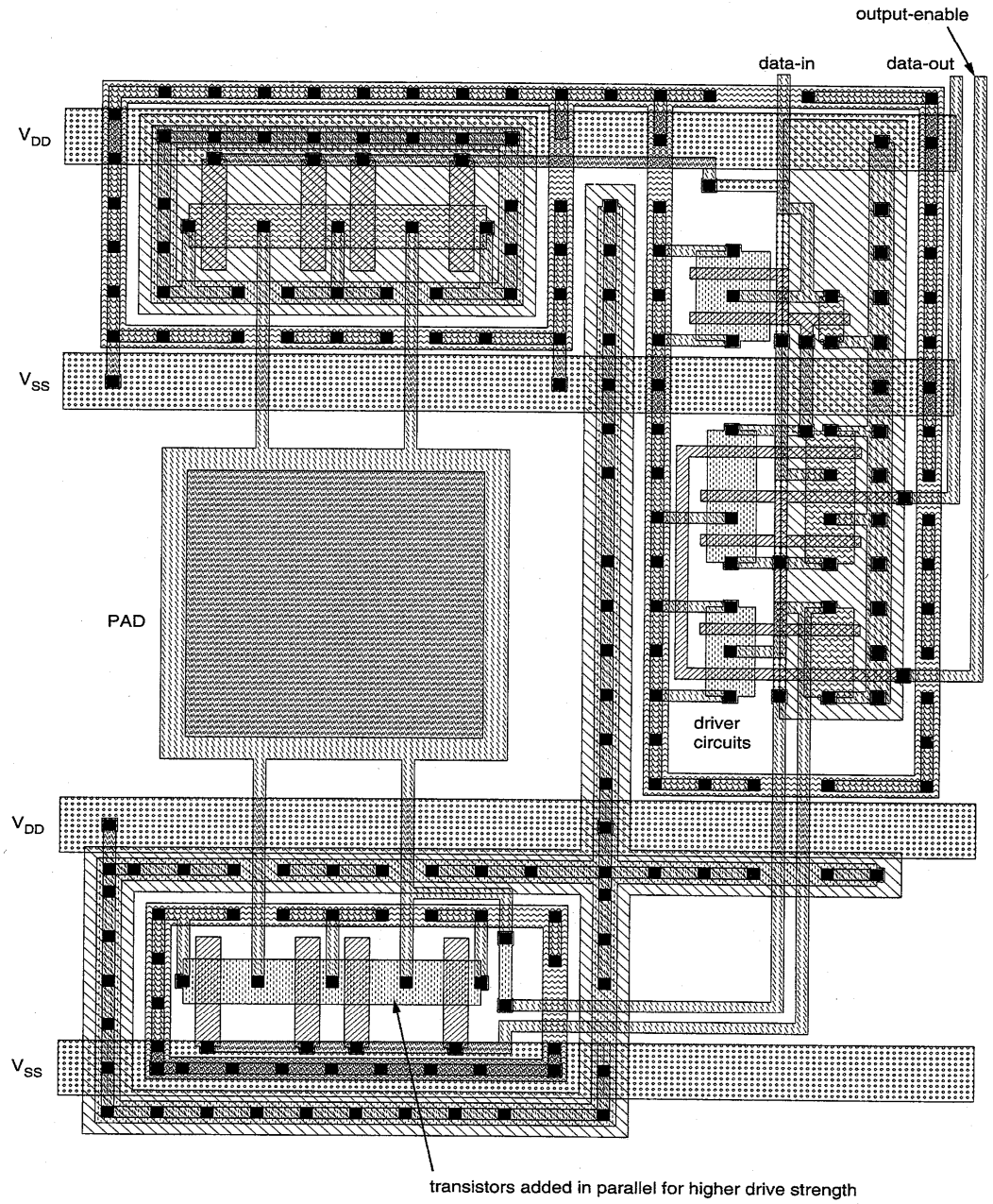
TRUTH TABLE

OE	D	N	P	OUT	
0	X	0	1	Z	(high impedance)
1	0	1	1	0	
1	1	1	0	0	1

(b)

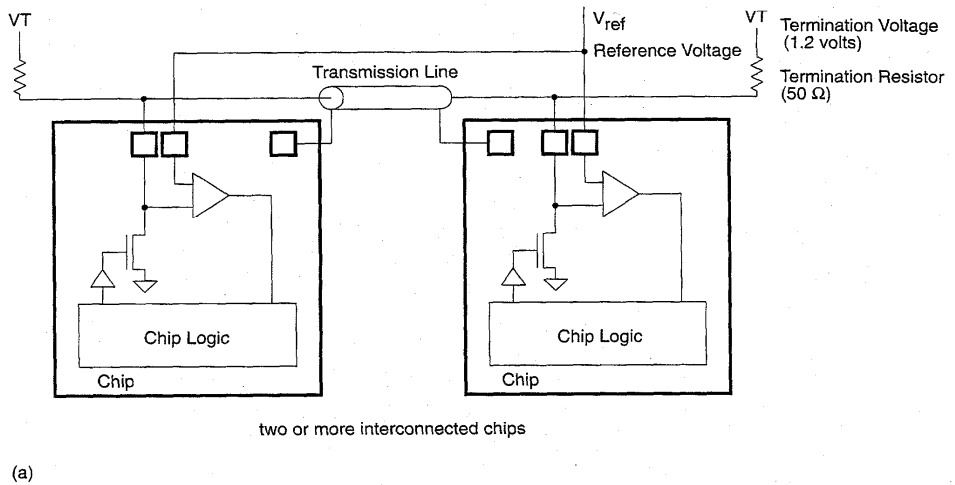
- Fig. (a) shows a tri-state driver feeding an output pad
  - Large p and n push-pull output devices are fed by output enable-controlled NAND and NOR logic
    - If oe is high, inverted data is fed to the gates of the p and n driver devices
    - If oe is down, the p-device gate is pulled to Vdd and the n-device gate is pulled to ground, causing a high Z output
- Addition of an input receiver/buffer allows construction of a bi-directional I/O pad
  - ESD diodes may be combined with the drain regions of n & p driver transistors
  - Diffused resistor provides ESD series resistor for input buffer

Bi-Directional Driver I/O Pad Symbolic Layout

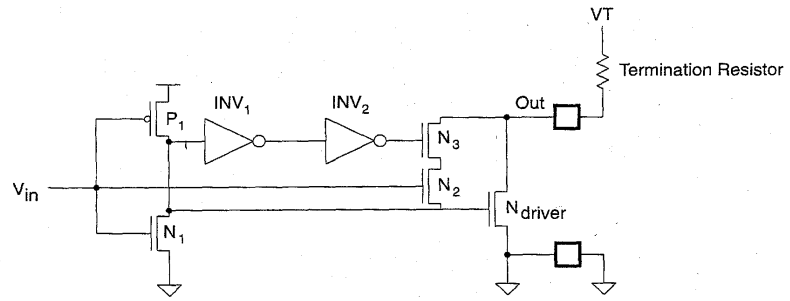


- Symbolic layout of a bi-directional I/O pad
  - Large n and p output devices formed by four parallel transistors for each
    - Driver drain regions double as ESD protect diodes
  - NAND, NOR, and inverter circuits for tri-state driver buffer are shown to the right of pad
  - Input buffer/inverter shown to right and above pad
  - Series diffused resistor shown below pad and to the right of pull-down n device
  - Both n and p driver devices are double guard-ringed to prevent latchup

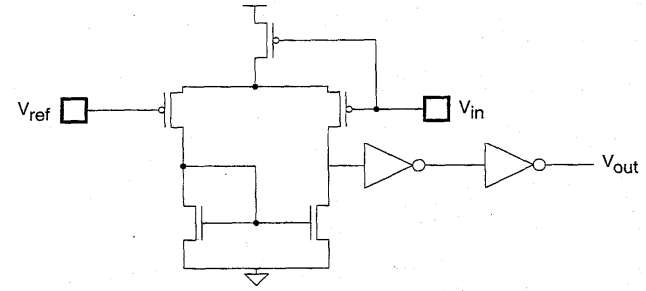
Low Voltage Swing GTL I/O Design



(a)



(b)



(c)

- Low voltage swing drivers are often needed to drive bipolar ECL inputs or to reduce dynamic power on I/O buses
- GTL driver (Gunning Transistor Logic) uses open-drain N pull-down output devices with 50 ohm terminating resistors connected to VT (1.2 volts) to drive chip-to-chip interconnections
  - Allows limited swing (0.8 volt) chip-to-chip buses to reduce ac power
  - Similar to bipolar open collector driver scheme used in ECL logic
  - Driver ckt (b) includes devices N2 & N3 plus INV1 & INV2 to reduce di/dt and limit overshoot reflections
    - Down level V_{OL} is 0.4 volt due to dc ratio with 50 ohm pull up resistor
  - Reduced swing input receiver (c) is a differential amplifier swinging +/- 0.4 volt around Vref = 0.8 volt