EECS 427
Lecture 6: Project architecture and intro logic styles
Reading: handout, 6.2

Reminders

- CAD3 is due next Wednesday
  - You have until Thursday noon to submit your design
- Looking ahead:
  - HW3 – Project initial proposal
    - Due Wednesday 10/7
    - Based on answering a series of questions. Template is posted
  - Quiz1 Wednesday 10/14, 2.5 weeks away!

Last Time – Logical Effort

Path effort $H = GFB$

Optimal stage effort $\hat{h} = H^{1/N}$

Optimal path delay $t_p = \sum p_i + NH^{1/N}$

Stage sizing $C_{out,i} = f_i C_{in,i} = \frac{\hat{h}}{g_i} C_{in,i}$

1. Compute path effort
2. Compute optimal stage effort
3. Add buffers (determine optimal number of stages)
4. Compute fan-out *f* of each stage
5. Size individual gates (working backward or forward)

Last Time – Logical Effort

- Limitations
  - Assumption of P/N = 2
  - Ignores internal capacitances
  - Simplistic view of stack effect
  - Branched path sizes up proportionally
  - Does not account for input slope, nor interconnect capacitance effect
  - Both R and C scale linearly

Lecture Overview

- Project architecture description (handout)
- Static and dynamic logic styles

Project architecture

- 2-stage pipeline, 1 word per instruction
  - 1st stage of pipe: instruction fetch (IF)
  - 2nd stage: instruction decode (ID), execute (EX)
- 16-bit words, with four 4-bit components
  - Most significant 4 bits are the operation code (opcode)
  - Tells which instruction (e.g., ADD, MOV, STOR) is to be performed
  - Next 4 bits give the register address to which the result of the instruction should be written (with a few exceptions)
  - Next 8 bits can contain several pieces of information:
    - Immediate data to be acted upon (rather than accessing this data from a register location)
    - Opcode extensions (since there are more than 2^4 or 16 ops)
    - Address of source register to draw data from

Example instructions

- Direct vs. immediate instructions
- *Add* Rsrc Rdest
  - Rdest ← Rdest *Add* Rsrc
  - Where Rdest and Rsrc are register addresses
- *Add* Imm Rdest
  - Rdest ← Rdest *Add* Imm
  - Where Imm is 8 bits of data (not an address)
- Typical instructions:
  - MOV moves data from 1 reg location to another
  - LOAD loads data from memory to the RF
  - STOR writes data to memory
  - Control flow instructions (conditional branches, jumps, jump and link)
- Look over baseline instructions and extra instructions, think about target application
- Weste 2nd edition handout is useful as overview of a processor architecture (note it does not exactly reflect our own architecture)

Building Blocks for Digital Architectures

Arithmetic unit

- Bit-sliced datapath (adder, multiplier, shifter, comparator, etc.)

Memory

- RAM, ROM, Buffers, Shift registers

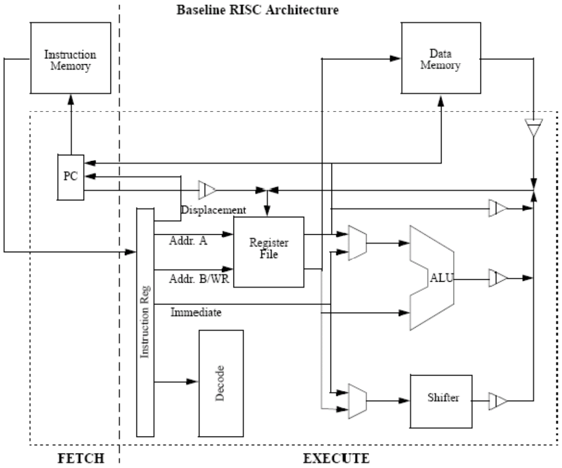
Control

- Finite state machine (PLA, random logic)
- Counters

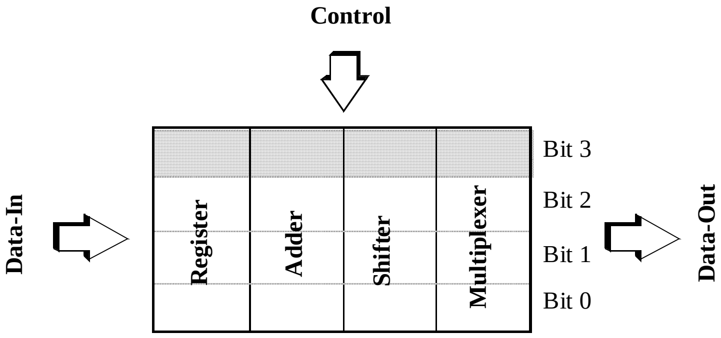
Interconnect

- Switches
- Arbiters
- Bus

A Generic Digital Processor



Bit-Sliced Design



Control

Data-In

Register

Adder

Shifter

Multiplexer

Bit 3

Bit 2

Bit 1

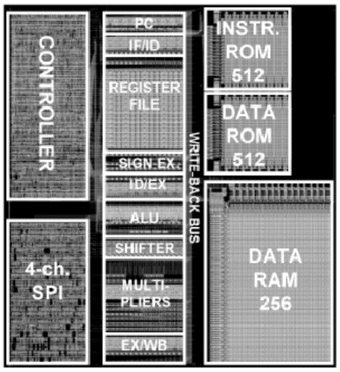
Bit 0

Data-Out

Tile identical processing elements

Project Ideas from the Past

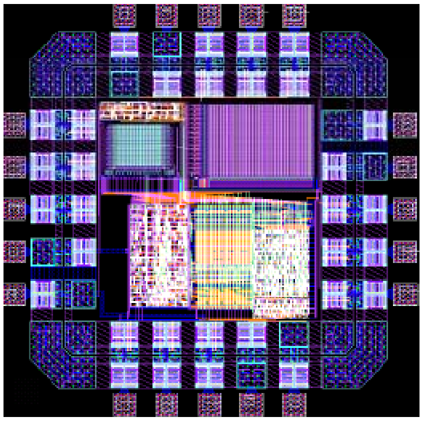
- **A Low-Power Dual-VDD Microprocessor for General Purpose Correlation Applications**
- *143 MHz*
- *reconfigurable multiplier, customized for correlation algorithms.*
- *Low-power techniques such as dual-Vdd (2.5/1.8V) and clock gating reduced power by 39% without compromising performance.*



Project Ideas from the Past

- **A 200 MHz 16-bit RISC Floating Point DSP for Electrocardiogram Systems**

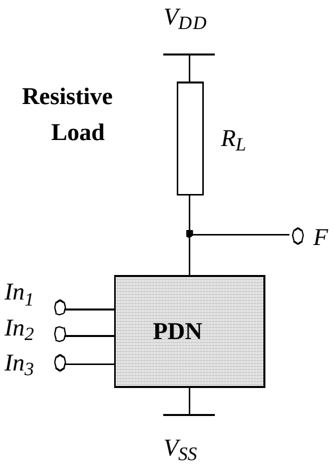
- **Floating-point DSP intended for medical instrumentation applications, such as electrocardiogram (ECG).**
- **Dedicated floating point unit (FPU)**
- **The test program performs FIR filtering on a sample electrocardiogram signal.**



Project Ideas

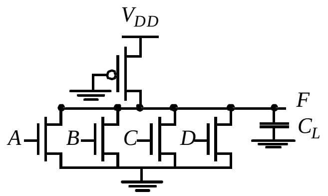
- Memory
  - SRAM design, sense amplifier, 6T variants
  - Pulse register, sense-amp-based register
- ALU
  - Carry look-ahead adder: Kogge-Stone radix 2, radix 4, Brent-Kung, Ling
  - Logic styles: PTL, domino, OPL
  - Multiplier
- Low power
  - Sleep mode, low-VDD, body biasing
- Dedicated processing
  - FFT, CORDIC, FIR

Ratioed Logic



- **N transistors + Load**
- **$V_{OH} = V_{DD}$**
- **$V_{OL} = \dfrac{R_{PN}}{R_{PN} + R_L}$**
- **Assymetrical response**
- **Static power consumption**
- **$t_{pL} = 0.69\ R_L C_L$**

Psuedo-NMOS



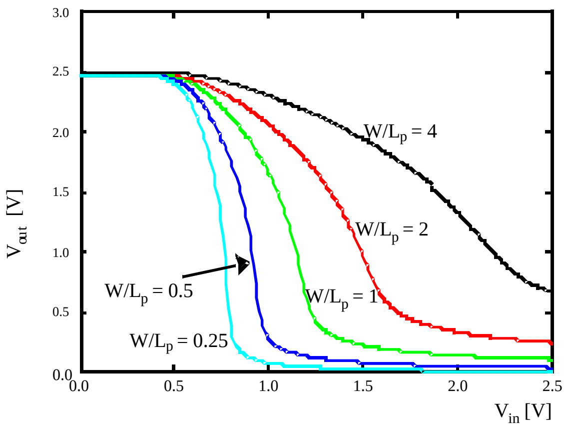
$V_{OH} = V_{DD}$ (similar to complementary CMOS)

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

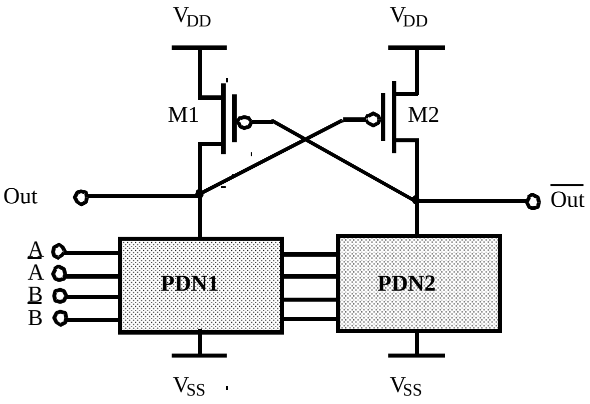
$$V_{OL} = (V_{DD}-V_T)\left[1-\sqrt{1-\frac{k_p}{k_n}}\right]$$ **(assuming that $V_T = V_{Tn} = |V_{Tp}|$)**

SMALLER AREA & LOAD BUT STATIC POWER DISSIPATION!!!

Pseudo-NMOS VTC

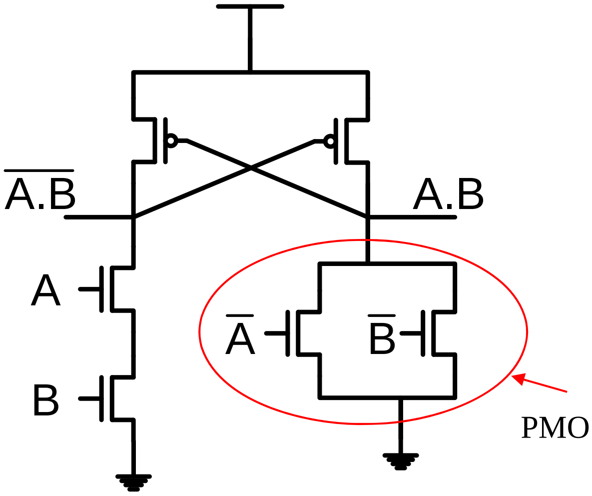


DCVSL



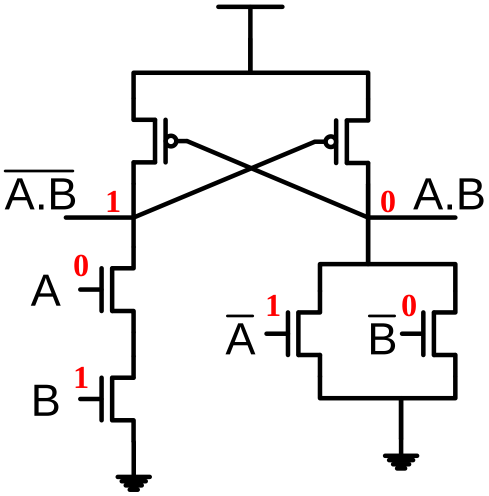
Differential Cascode Voltage Switch Logic (DCVSL)

DCVSL Example

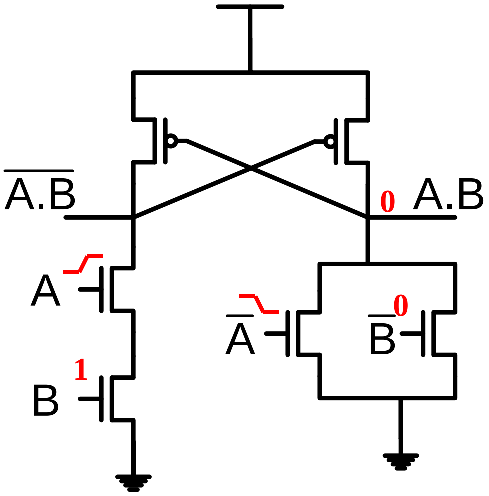


PMOS stack with NMOS

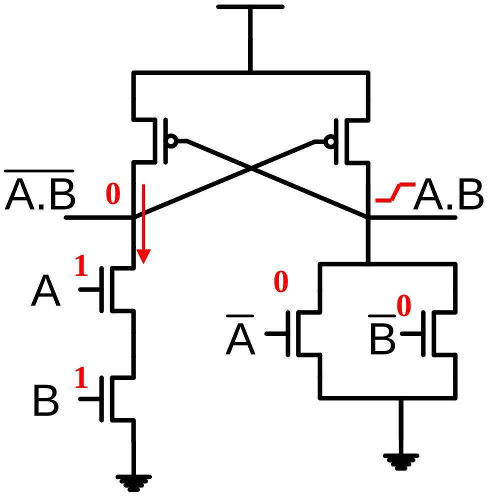
DCVSL Example



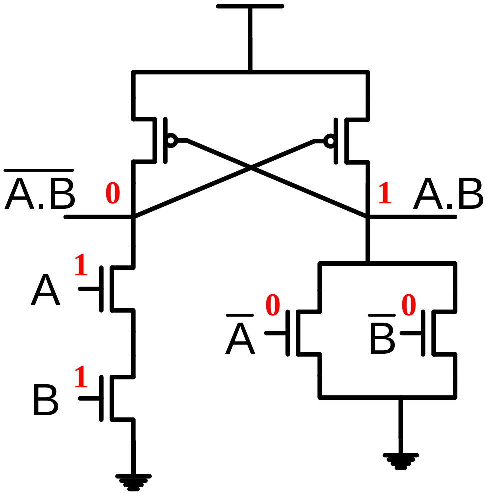
DCVSL Example



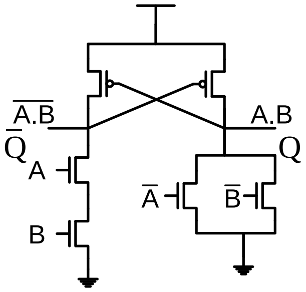
DCVSL Example

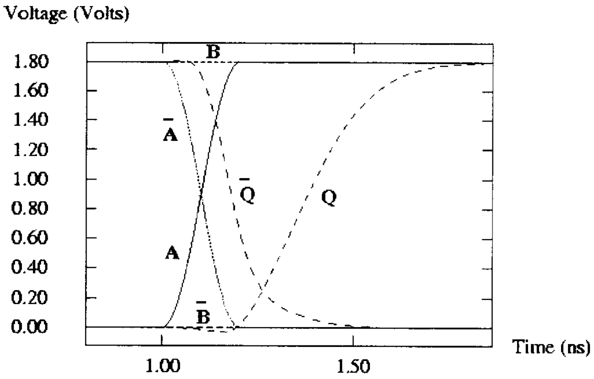


DCVSL Example

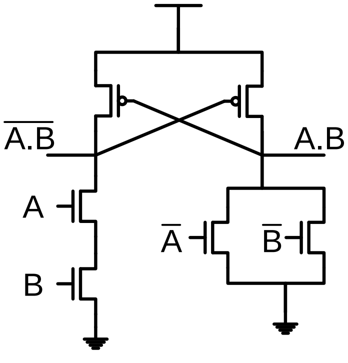


DCVSL Example



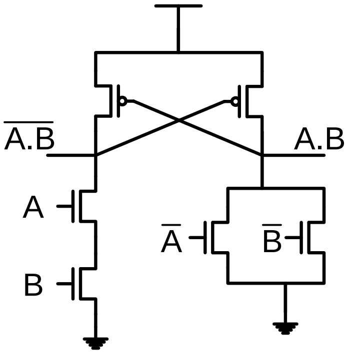


DCVSL



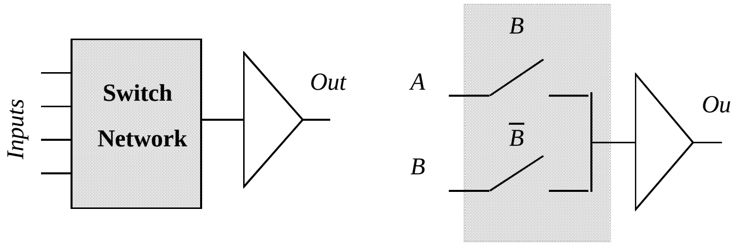
- Advantages:
  - No PMOS duality
    - Lower input cap.
    - Use only NMOS
  - Faster than CMOS
  - Can evaluate complex logic trees in 1 stage

DCVSL



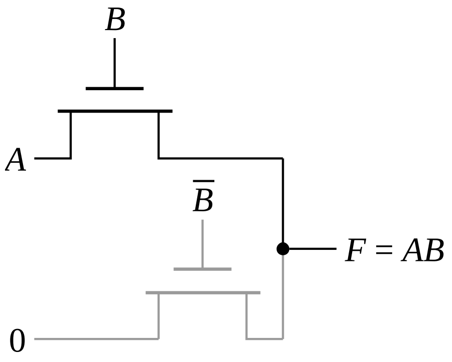
- Disadvantages:
  - – Need complementary inputs (dual rail)
  - – Cross-bar current
    - Sensitive to input timing
  - – Sizing of PMOS is hard
    - Too large → PDN does not switch the output
    - Too small → Slow rise time

Pass-Transistor Logic

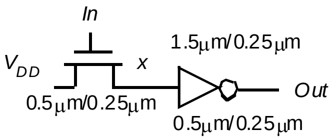


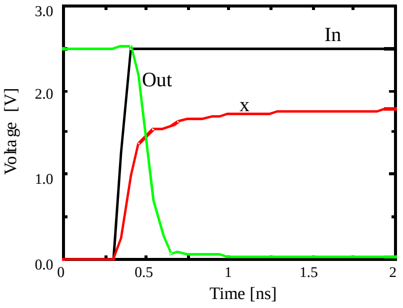
- N transistors
- No static consumption

AND Gate

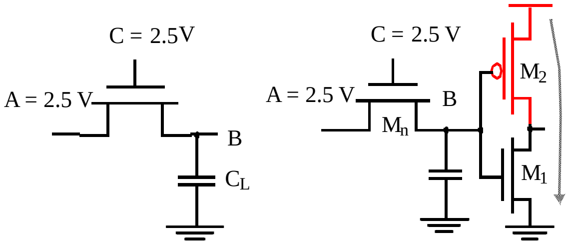


NMOS-Only Logic





NMOS-Only Switch

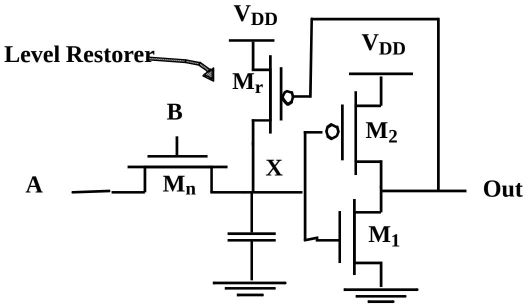


V_B does not pull up to 2.5V, but 2.5V - V_{TN}

Threshold voltage loss causes static power consumption

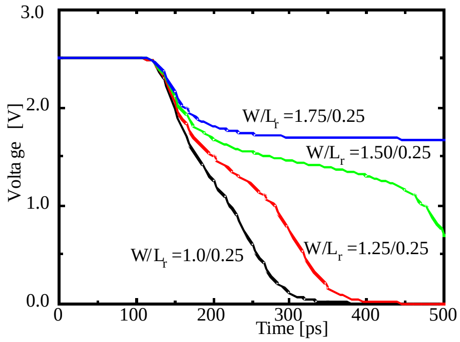
NMOS has higher threshold than PMOS (body effect)

Level Restoration



- Advantage: Full Swing
- Restorer adds capacitance, takes away pull down current at X
- Ratio problem

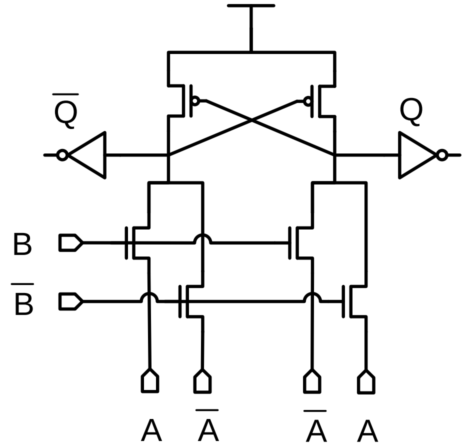
Restorer Sizing



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

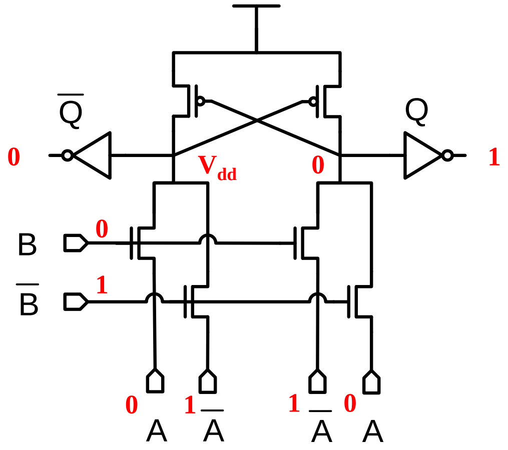
---

CPL

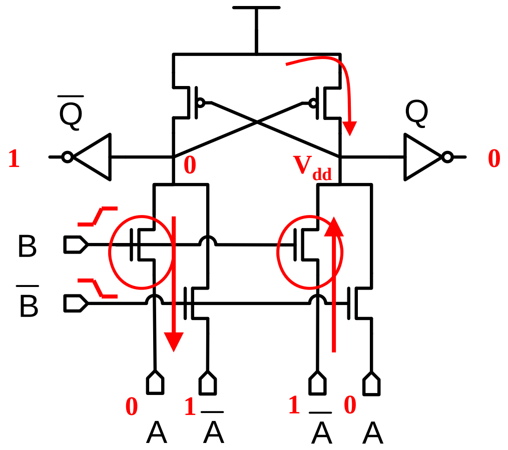


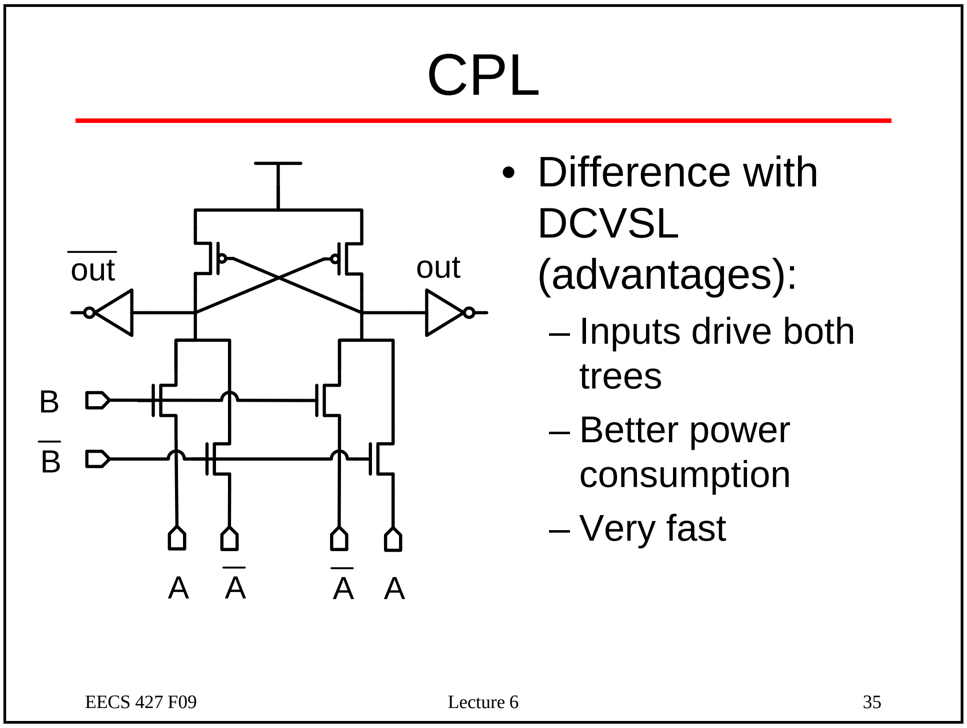
- Dual rail Pass gate logic with differential cascode voltage switch logic
  - – Combination of DCVS and PTL

CPL


$\overline{Q}$
0
V_{dd}
0
Q
1
B
0
$\overline{B}$
1
0
1
1
0
A
$\overline{A}$
$\overline{A}$
A

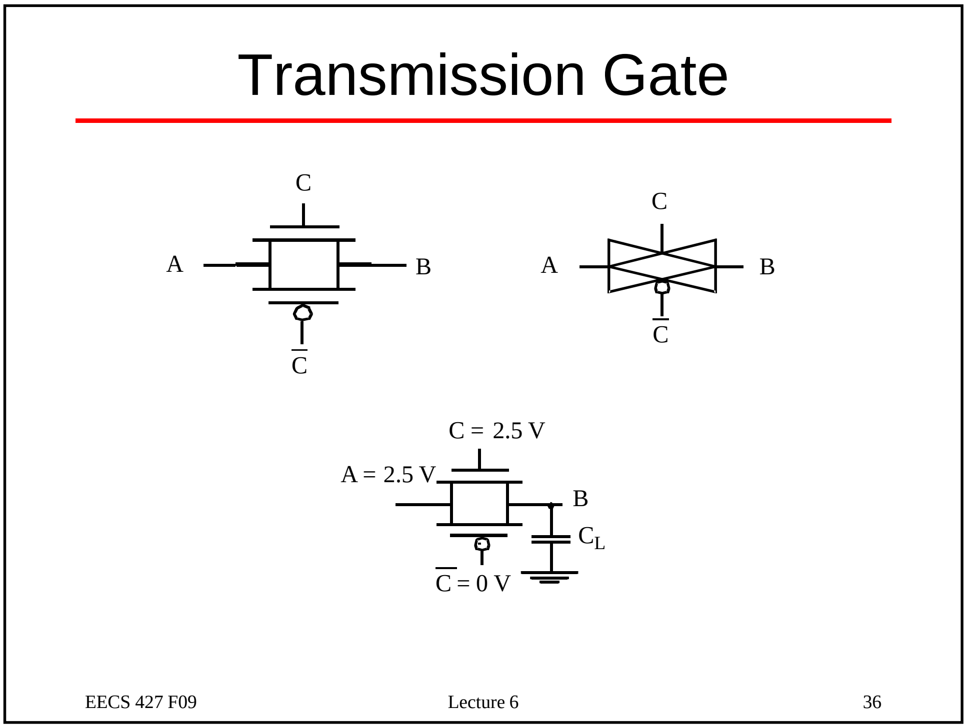
CPL


$\overline{Q}$
1
0
V_{dd}
Q
0
B
$\overline{B}$
0
1
1
0
A
$\overline{A}$
$\overline{A}$
A



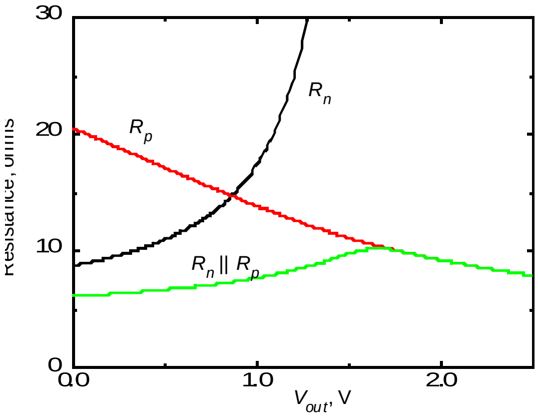
CPL

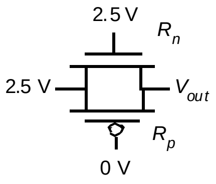
- Difference with DCVSL (advantages):
  - Inputs drive both trees
  - Better power consumption
  - Very fast



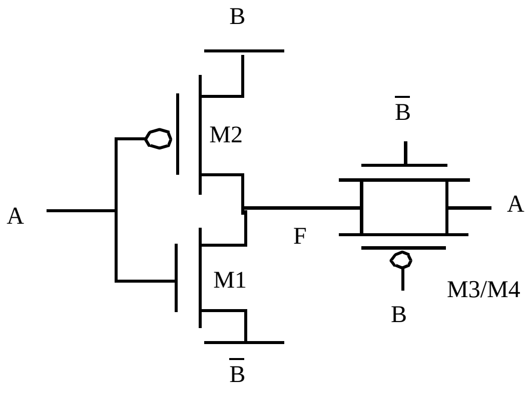
Transmission Gate

Equivalent Resistance

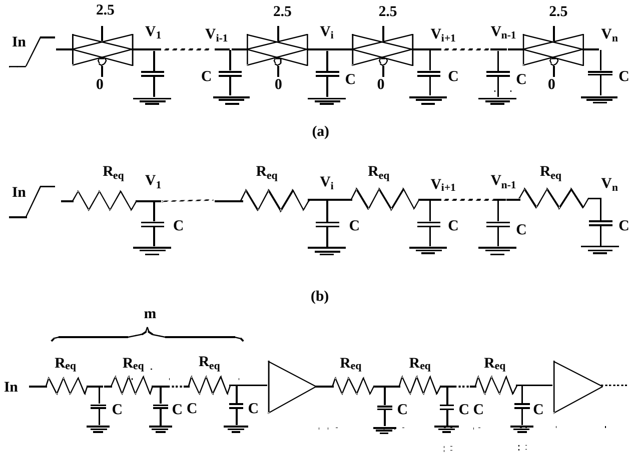




Transmission Gate XOR



Transmission Gate Network



(a)

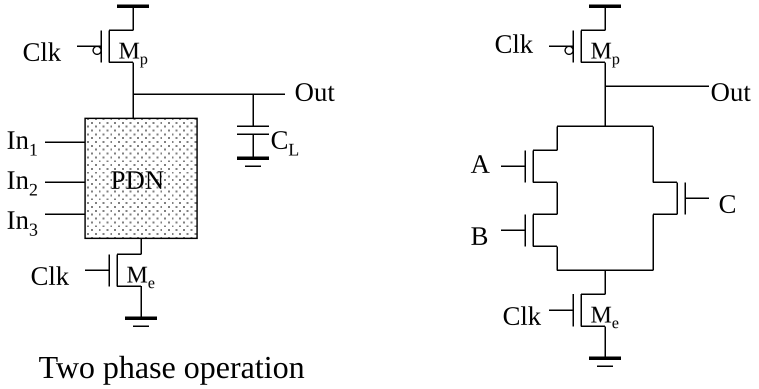
(b)

(c)

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

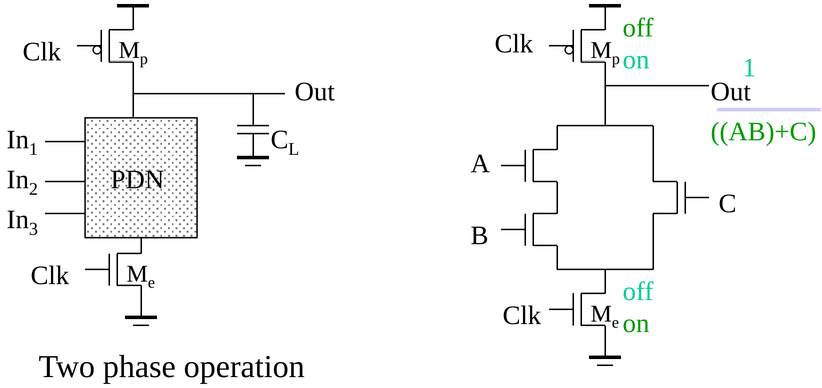


Two phase operation

Precharge (CLK = 0)

Evaluate   (CLK = 1)

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

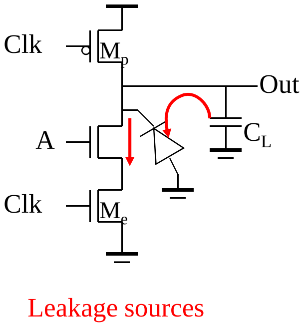
Properties of Dynamic Gates

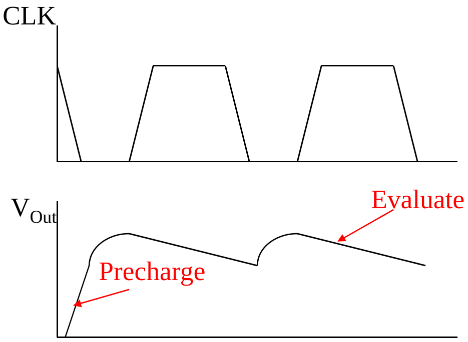
- Logic function is implemented by the PDN only
  - number of transistors is N + 2 (versus 2N for static complementary CMOS)
- Full swing outputs (V_{OL} = GND and $V_{OH} = V_{DD}$)
- Non-ratioed - sizing of the devices does not affect the logic levels
- Faster switching speeds
  - reduced load capacitance 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

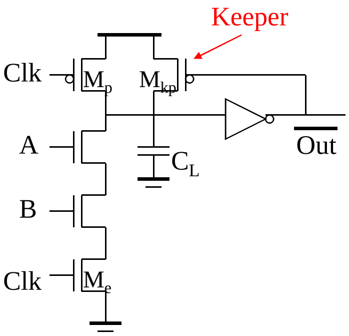
Charge Leakage





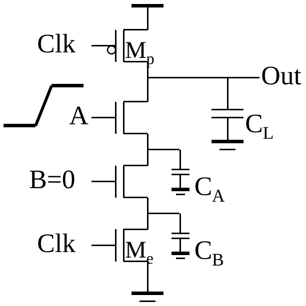
Dominant component is subthreshold current

Keeper



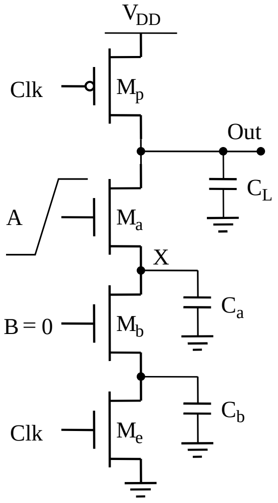
Same approach as level restorer for pass-transistor logic

Charge Sharing



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

Charge Sharing



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

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

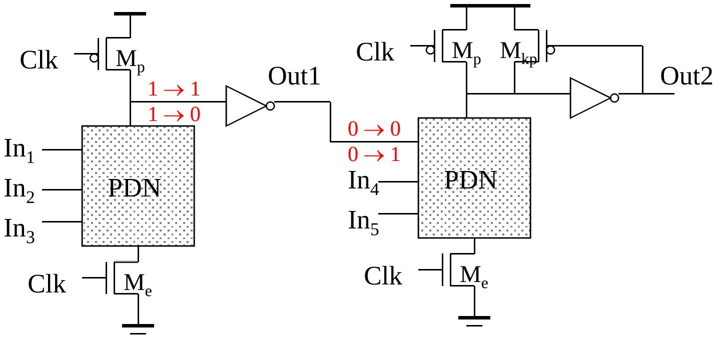
or

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

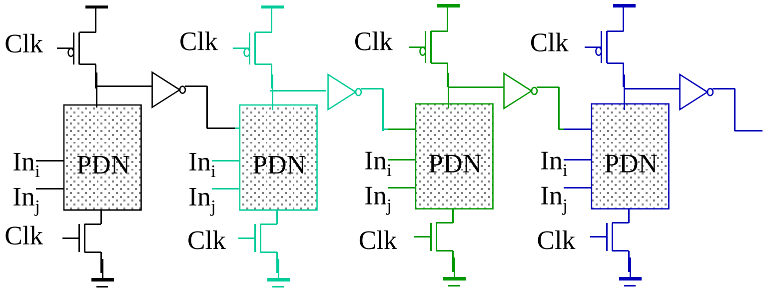
case 2) if $\Delta V_{out} > V_{Tn}$

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

Domino Logic



Cascading Dominos

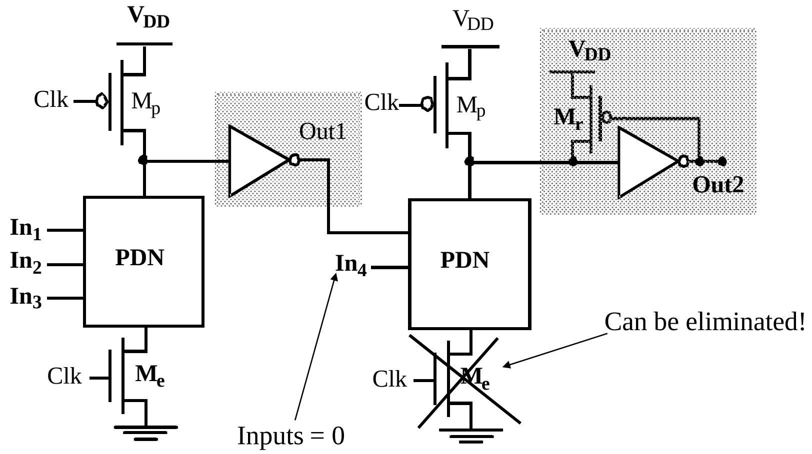


Like falling dominos!

Properties of Domino Logic

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

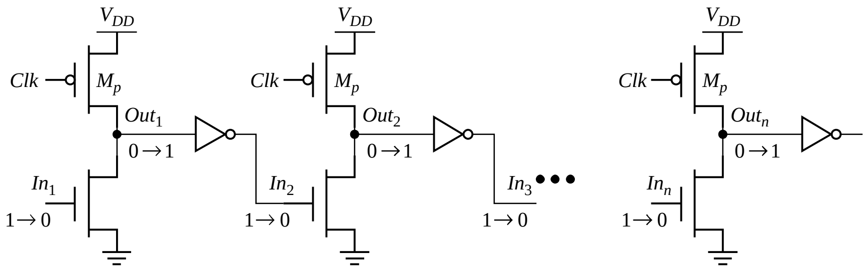
Design with Domino Logic



Can be eliminated!

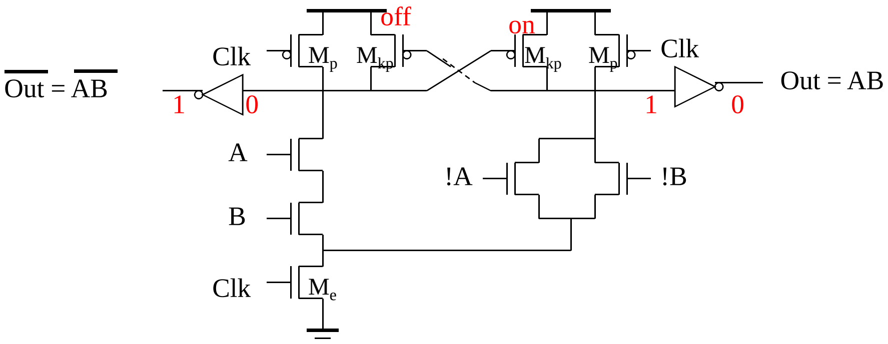
Inputs = 0
during precharge

Footless Domino



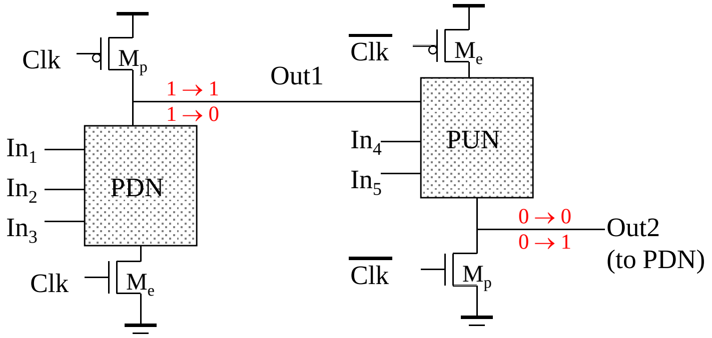
The first gate in the chain needs a foot switch
Precharge is rippling – short-circuit current
A solution is to delay the clock for each stage

Dual-Rail Domino



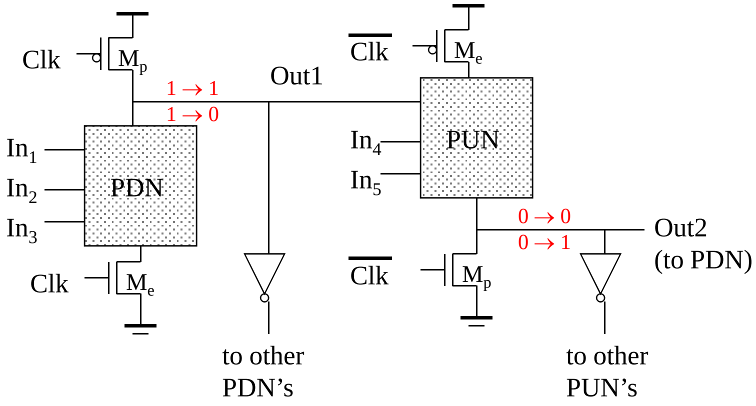
Solves the problem of non-inverting logic

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!

Summary

- Ratioed logic – improved loads
  - Psudo-NMOS – static current, low noise margin
  - DCVSL – no static current but cross-over current
- Pass-transistor circuits – simplified logic
  - PTL – threshold drop, causing static current in following gate
  - Transmission gate – no threshold drop
  - CPL – one side pulls up and the other pulls down
- Dynamic circuits – high performance
  - Dynamic logic – non-ratioed, dynamic power only, no static current, higher activity, low noise margin
  - Domino logic – can be safely cascaded, only non-inverting logic
  - Footless domino – ripple precharge, delayed clock, extra power
  - Dual-rail domino
  - NP CMOS