EE-429
Fundamentals of VLSI Design

Alternative Logic Styles

Andreas Burg

Different Logic Styles

- **Static CMOS is the most popular logic style due to its**
  - Robustness against noise (good noise margins)
  - Robust against variations in transistor parameters (un-ratioed logic)
  - Almost-zero static power consumption
  - Ability to easily cascade multiple logic stages of any type of gates
  - Ease of use and very good suitability for abstracting delay models (good for EDA)

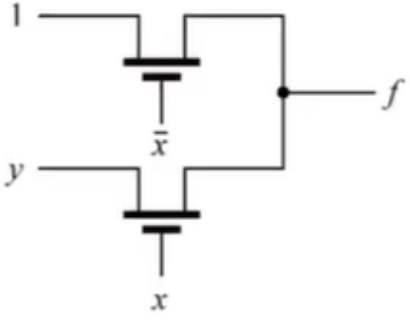
- **Other styles have been proposed to improve some short-comings of CMOS**
  - Limited fanin (number of inputs to a single-stage gate)
  - Inability to describe certain Boolean functions in a single logic stage
  - Notable area overhead due to the need for complementary PMOS and NMOS
  - Imbalance between PMOS and NMOS stage (differences in mobility)

Pass-Transistor Logic

- **CMOS always drives gate-output from the rails**
  - Good for output noise margins
  - Less obvious when outputs are trivial functions of the inputs (e.g., MUX) or for some basic non-unate gates such as XORs

- **Solution: drive output from one of the inputs** using **switches** made **from NMOS or PMOS transistors**

- **Systematic construction** from Boolean equations **using Shannon expansion**
  - Start from Boolean expression: $f(x_1, x_2, \ldots, x_n)$ and re-write as $f(x_1, x_2, \ldots, x_n) = x_1 \cdot f(1, x_2, \ldots, x_n) + \overline{x_1} \cdot f(1, x_2, \ldots, x_n)$

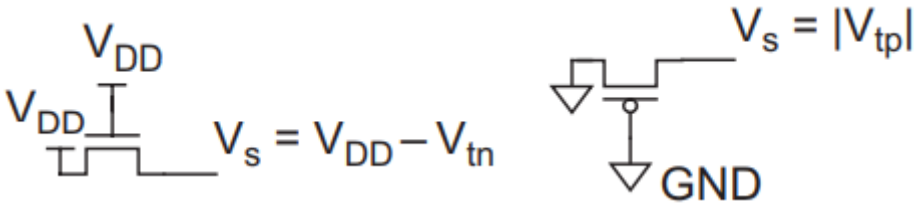


x	y	f
0	0	1
0	1	1
1	0	0
1	1	1

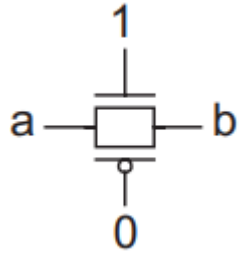
$f(x, y) = \bar{x} + x \cdot y =$
$x \cdot (0 + 1 \cdot y) + \bar{x} \cdot (1 + 0 \cdot y) =$
$x \cdot (y) + \bar{x} \cdot (1)$

Pass-Transistor Logic

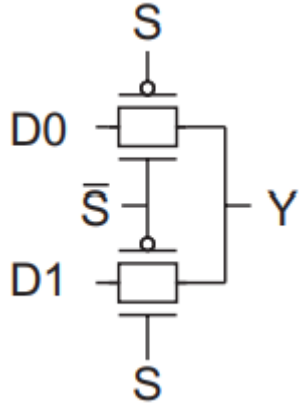
- **Problem: VT drop as NMOS and PMOS lack the ability to drive a strong ‘1’ and a strong ‘0’ respectively**



- **Solution: drive output from one of the inputs** using switches made from Transmission gates instead of **NMOS or PMOS transistors**



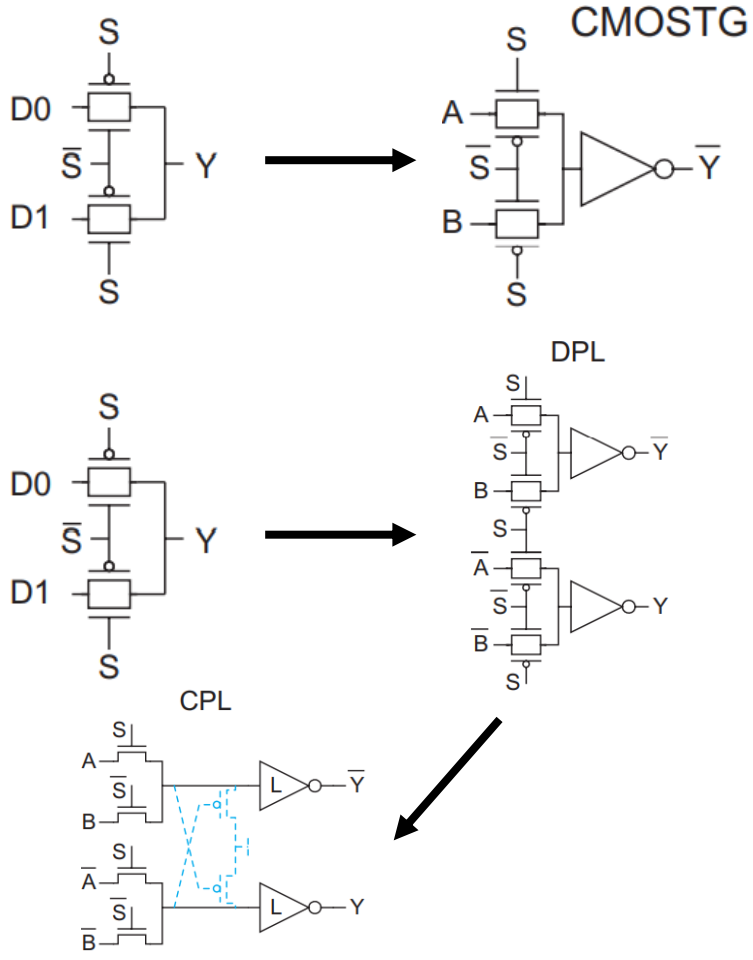
- **Example 2-input MUX**
  - Doubles the number of transistors
  - Adds PMOS to an otherwise NMOS-only circuit
  - Need for inverted inputs



Pass-Transistor Logic

- **Systematic construction** from Boolean equations **using Shannon expansion**

- **Many disadvantages:** with various solutions
  - Logic output can depend directly on the input: drive strength degrades gradually across multiple logic stage
    Solution CMOSTG: regularly buffer outputs with CMOS inverters
  - Typically requires inputs and also inverted inputs
    Solution (DPL): generate both outputs and inverted outputs
  - Large overhead from transmission gates
    Solution (CPL): accept a VT drop in one branch of the differential (complementary) output

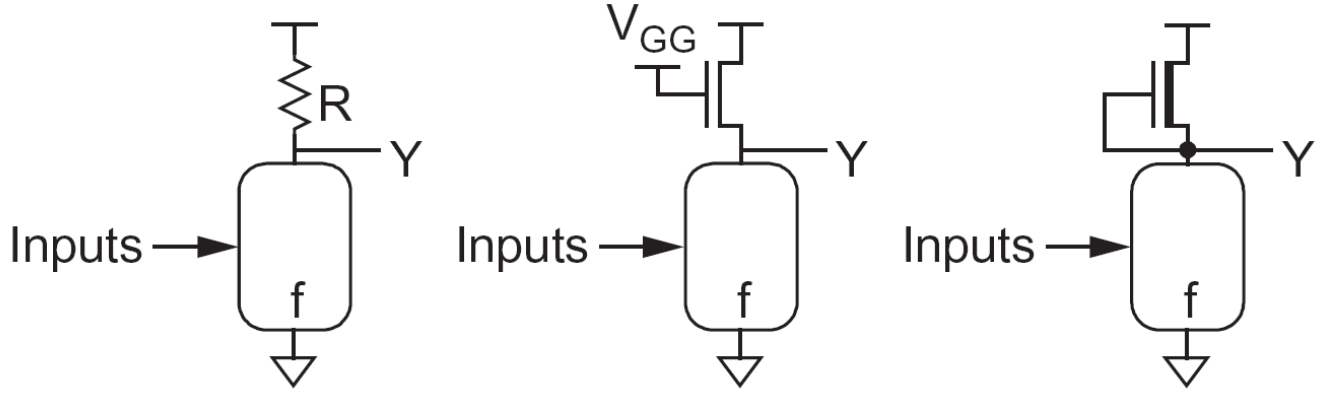


Ratioed Logic Gates

- **CMOS gates "waste"** 50% of the **transistors** to cut one of the supplies when the other supply rail is connected to the output

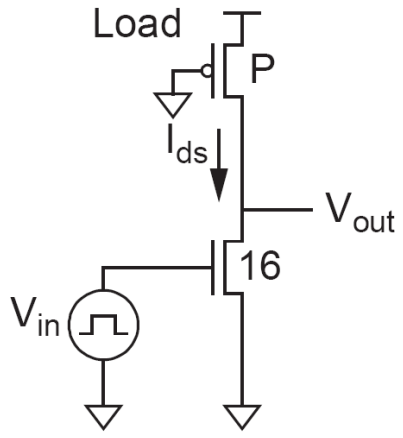
- **Solution**: **replace pull-up or pull-down switch** with constant "resistance" that is
  - Sufficiently strong to **pull** the output **when it is not pulled actively to the other side**
  - Sufficiently weak to **marginally impact** the output **when it is actively pulled to the other side**

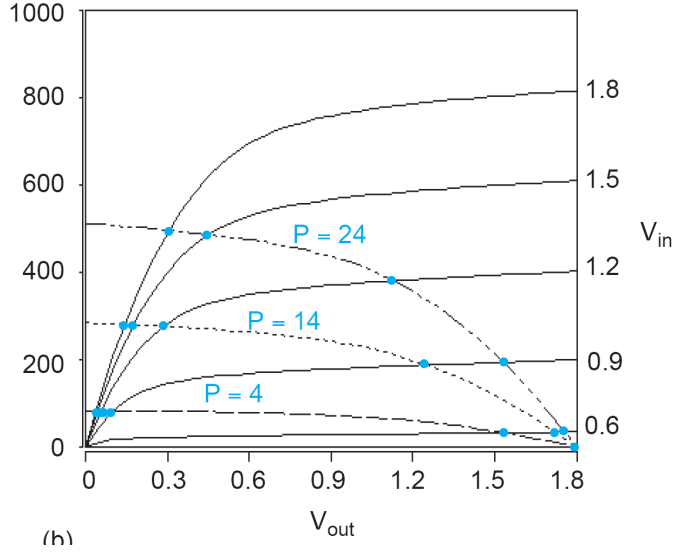
- **Especially attractive to remove the larger PMOS**

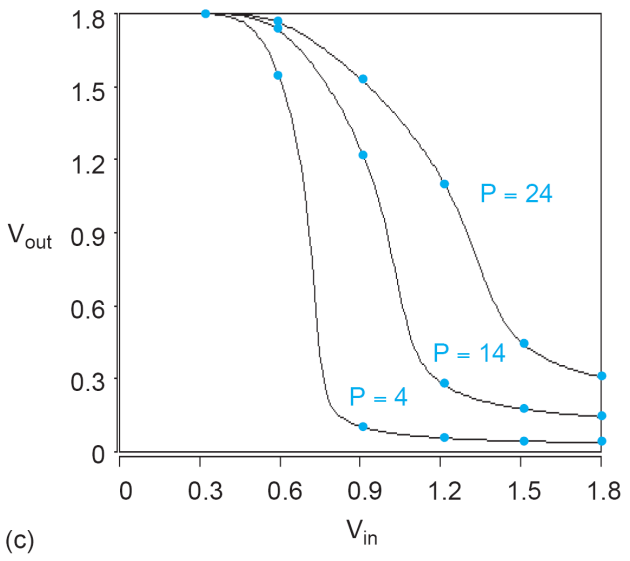


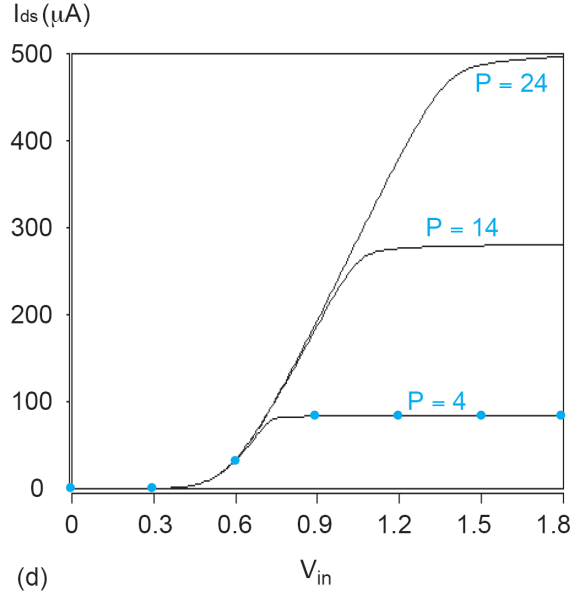
Pseudo NMOS Logic

- **Replaces PMOS network with an always weakly-on biased PMOS**
  - Constant power consumption due to short-circuit when output is low
  - Fall time determined by sizing of the NMOS network
  - Rise time determined by sizing of the always-on PMOS
    - Determines static power consumption
    - Strong PMOS (short rise time) degrades output low-level noise margin
- **Example: inverter**





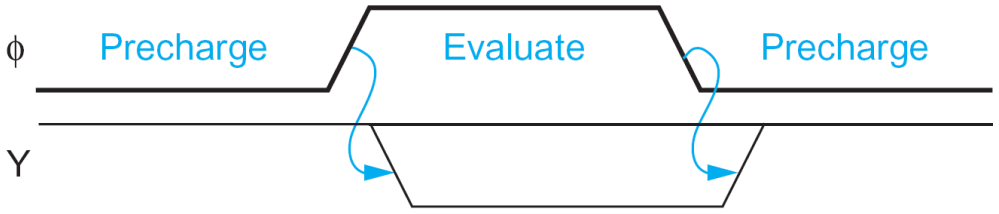


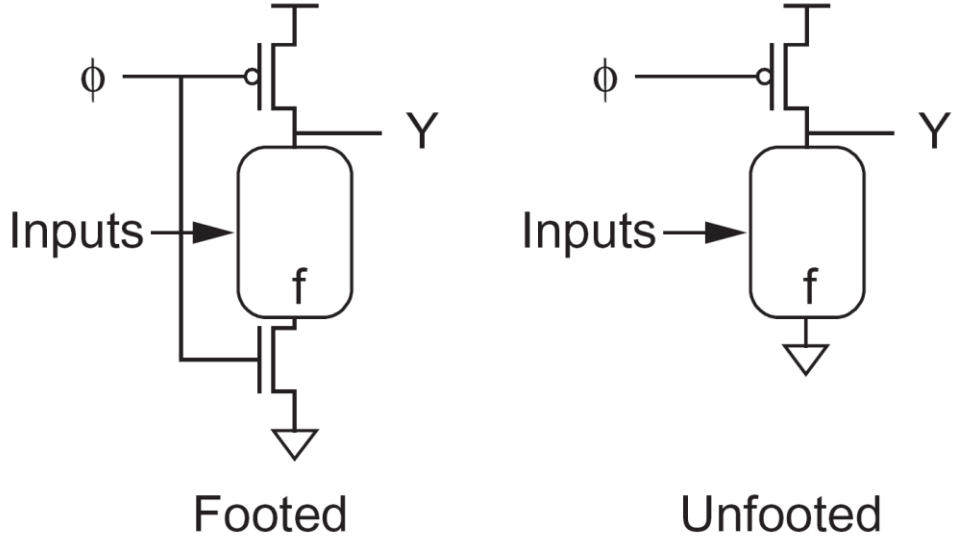


Dynamic Logic

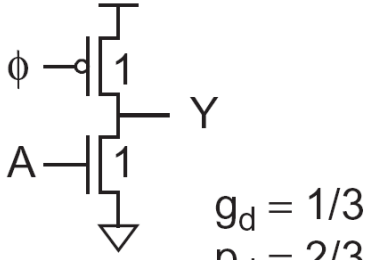
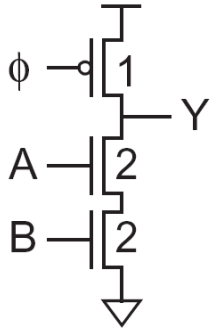
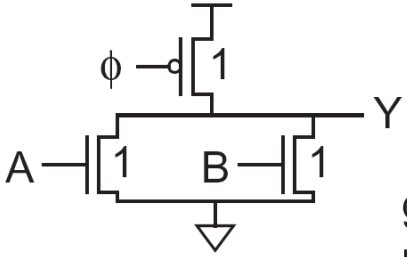
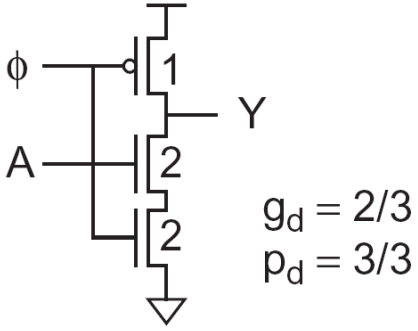
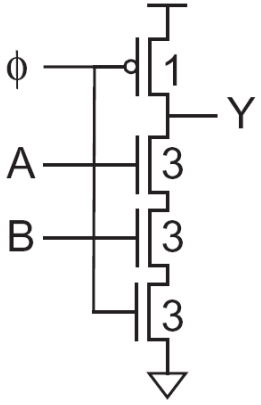
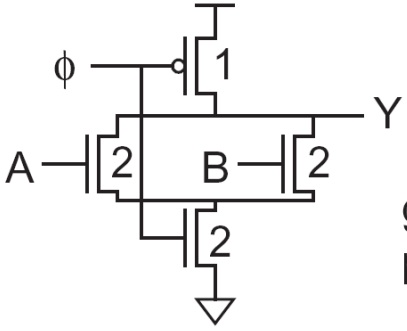
- **Drawback of Ratioed Logic: short-circuit current (for one of the outputs)**
  - Limits the drive strength of the always-on (pull-up or pull-down) device

- **Solution: operation in two phases separates pull-up from pull-down**
  - **Phase 1** pre-charge (pre-discharge): always pull-up (pull-down) (optional footer prevents short-circuit during precharge)
  - **Phase 2** evaluate output: conditionally pull-down (pull-up) depending on inputs





Dynamic Logic Examples

	Inverter	NAND2	NOR2
Unfooted	 $g_d = 1/3$ $p_d = 2/3$	 $g_d = 2/3$ $p_d = 3/3$	 $g_d = 1/3$ $p_d = 3/3$
Footed	 $g_d = 2/3$ $p_d = 3/3$	 $g_d = 3/3$ $p_d = 4/3$	 $g_d = 2/3$ $p_d = 5/3$