计算机组织与系统结构

技术工艺与延迟模型

Technology and Delay Modeling

（第五讲）

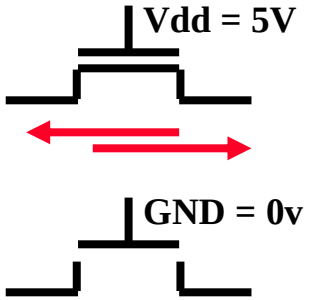
程 旭

2020.11.12

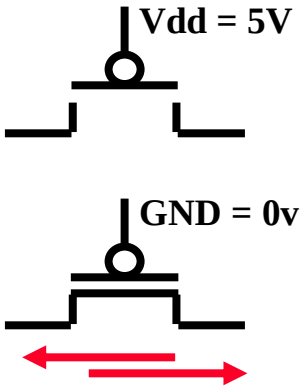
基本技术工艺: CMOS

- CMOS: Complementary Metal Oxide Semiconductor
  - NMOS (N-Type Metal Oxide Semiconductor) transistors
  - PMOS (P-Type Metal Oxide Semiconductor) transistors

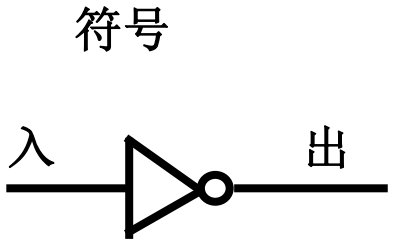
- NMOS 晶体管
  - 向逻辑门施加 高电平 （HIGH、Vdd、 5V）
    将该晶体管 变成 导体
  - 向逻辑门施加 低电平 （LOW 、GND, 0v）
    关闭导通路径

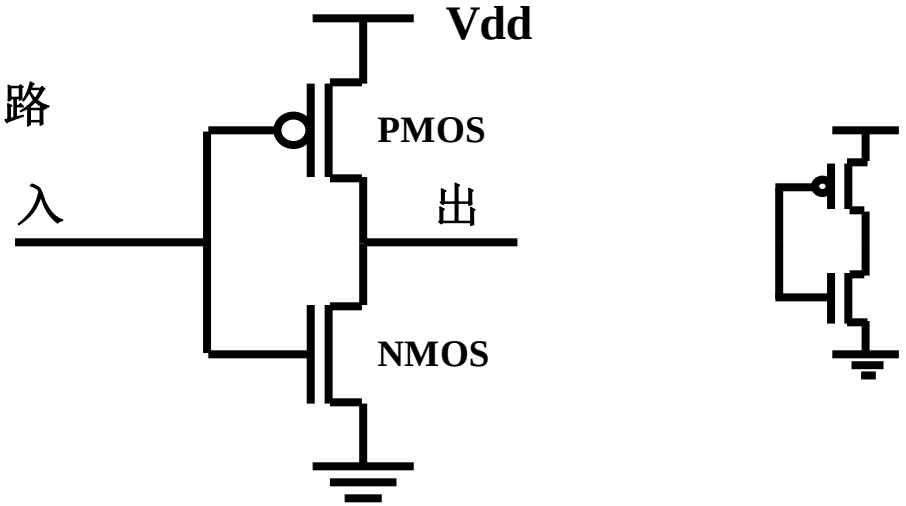


- PMOS晶体管
  - 向逻辑门施加 高电平 （HIGH、Vdd、 5V）
    关闭导通路径
  - 向逻辑门施加 低电平 （LOW 、GND, 0v）
    将该晶体管 变成 导体

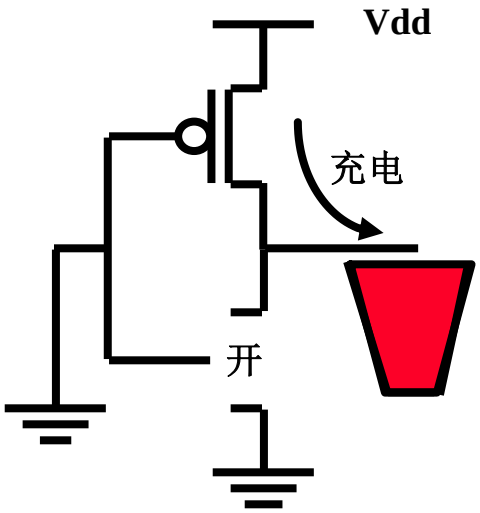


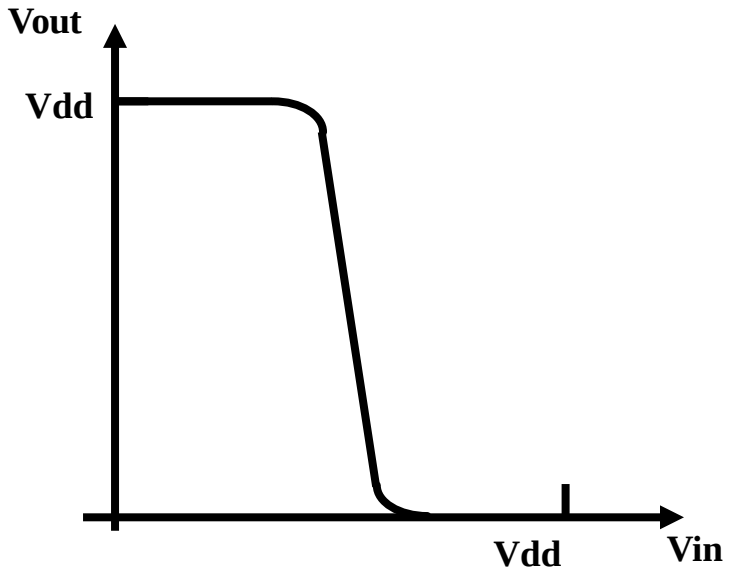
基本部件：CMOS 反向器

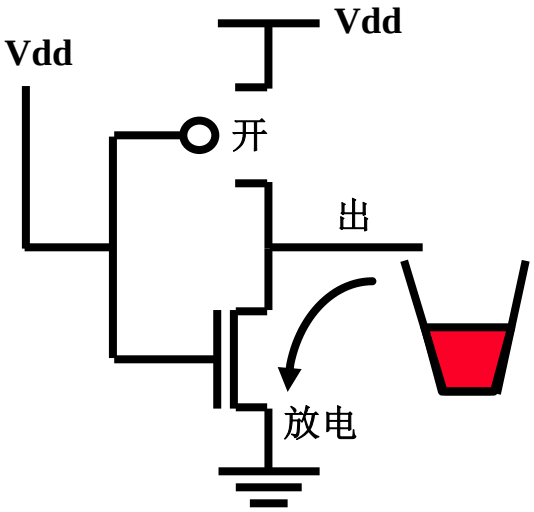




■反向器工作过程

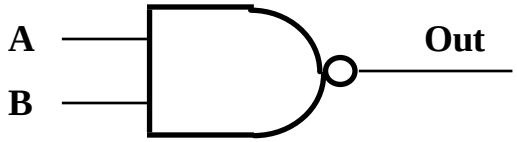




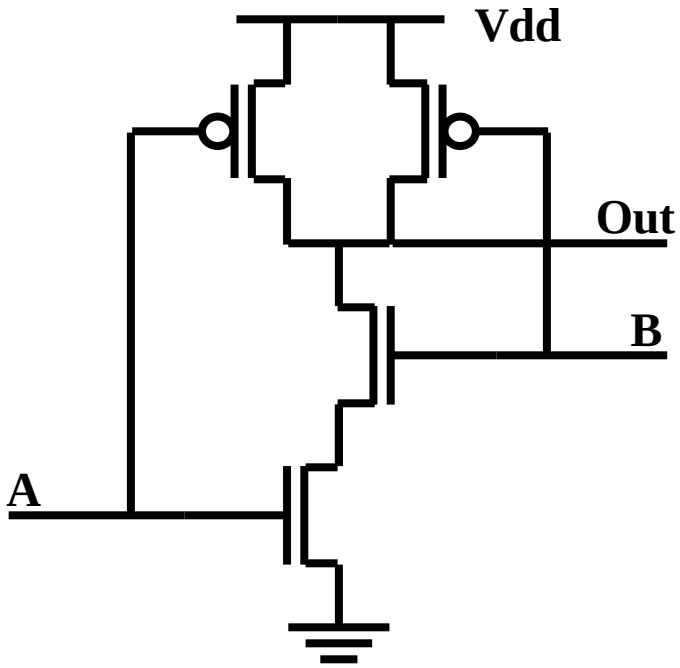


基本部件: CMOS 逻辑门

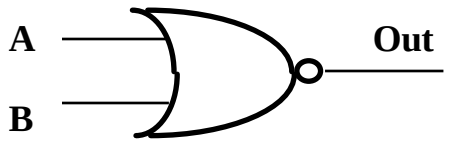
与非门（NAND Gate）



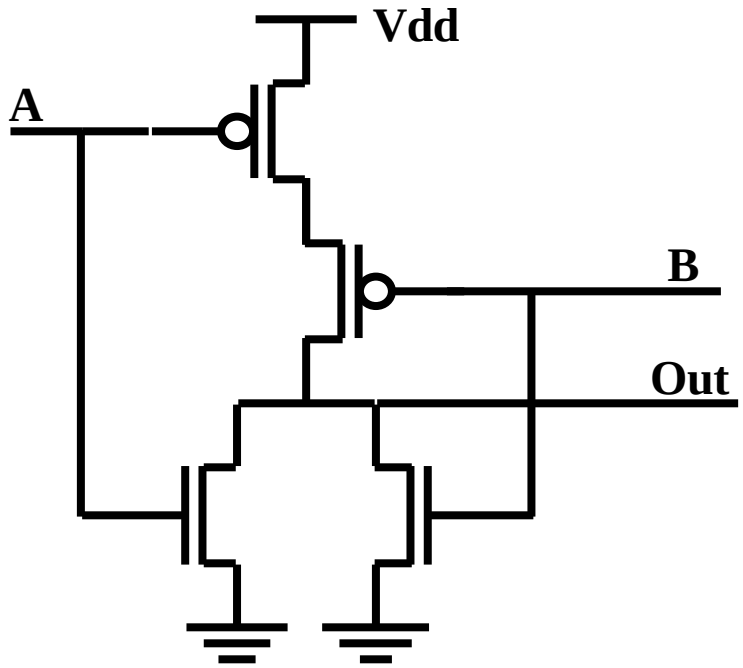
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0



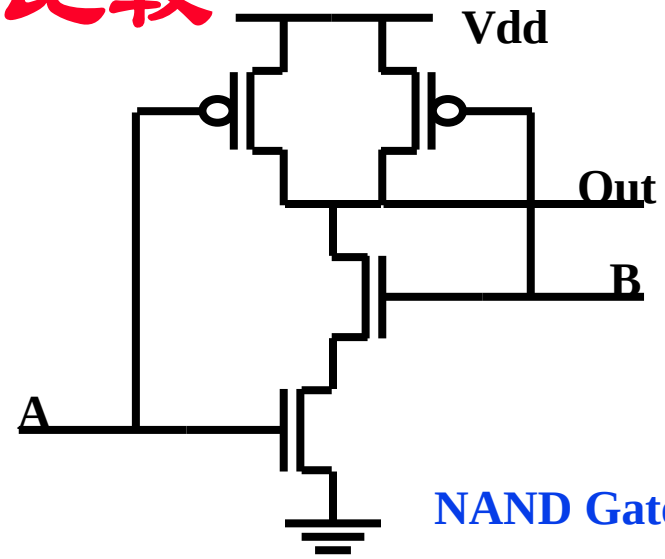
或非门（NOR Gate）



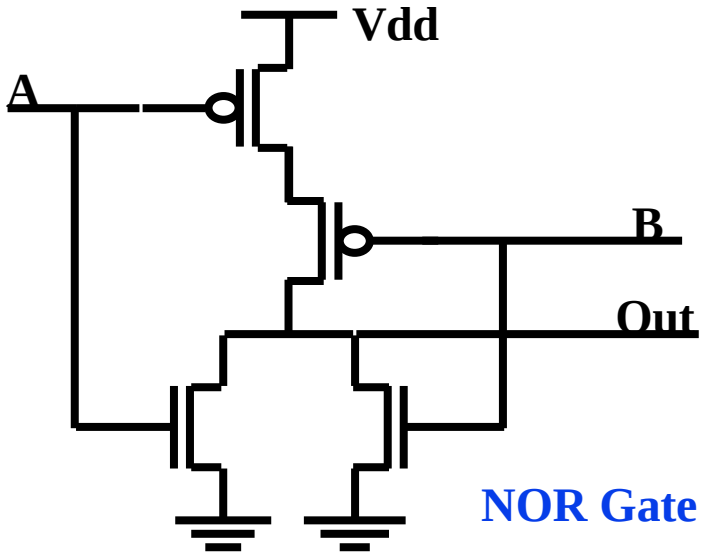
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0



门比较



NAND Gate



NOR Gate

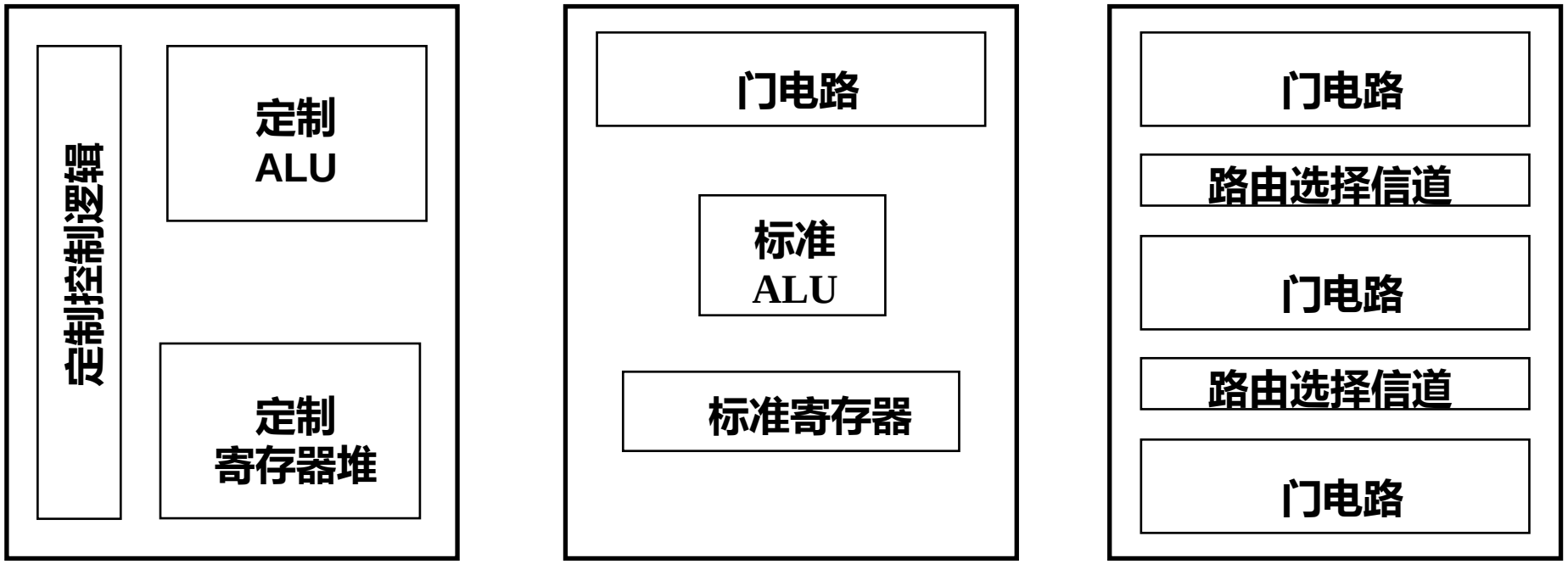
- 如果 PMOS晶体管较快，　那么
  - 就选用 PMOS 晶体管系列
  - 首选 或非门（NOR gate）
- 同样，如果 H ⇒ L比 L⇒ H更关键, 也首选 或非门（NOR gate）

设计风格的选择范围

定制设计（Custom Design）

标准单元（Standard Cell）

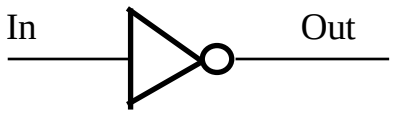
门阵列（Gate Array）/FPGA/PLD

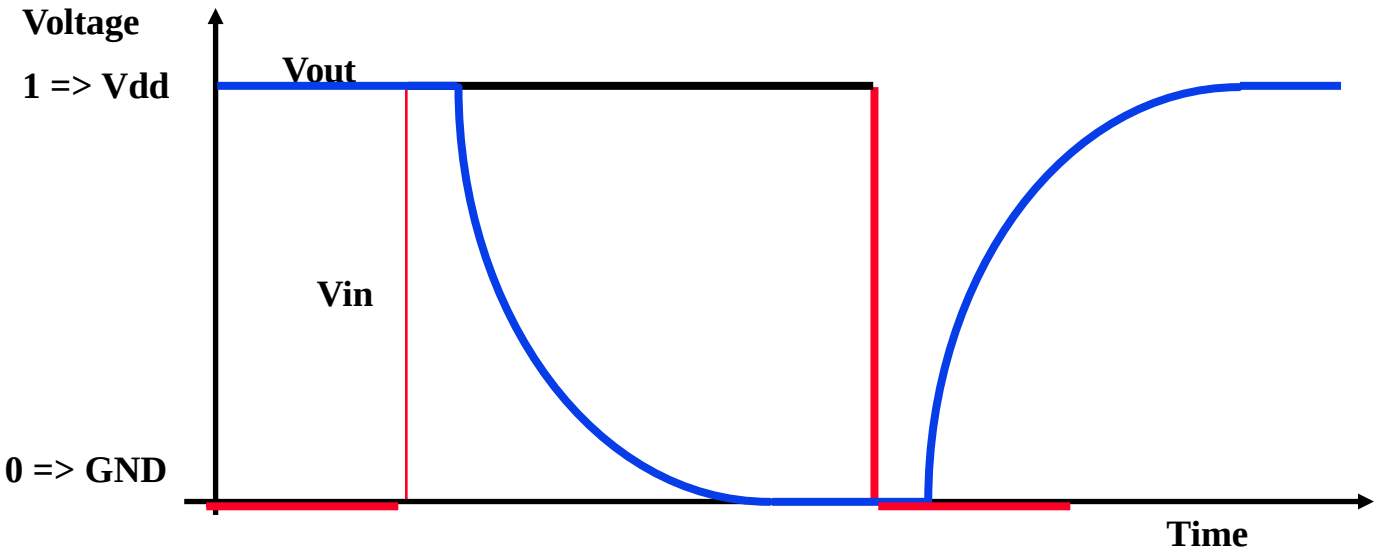


时间就是金钱！

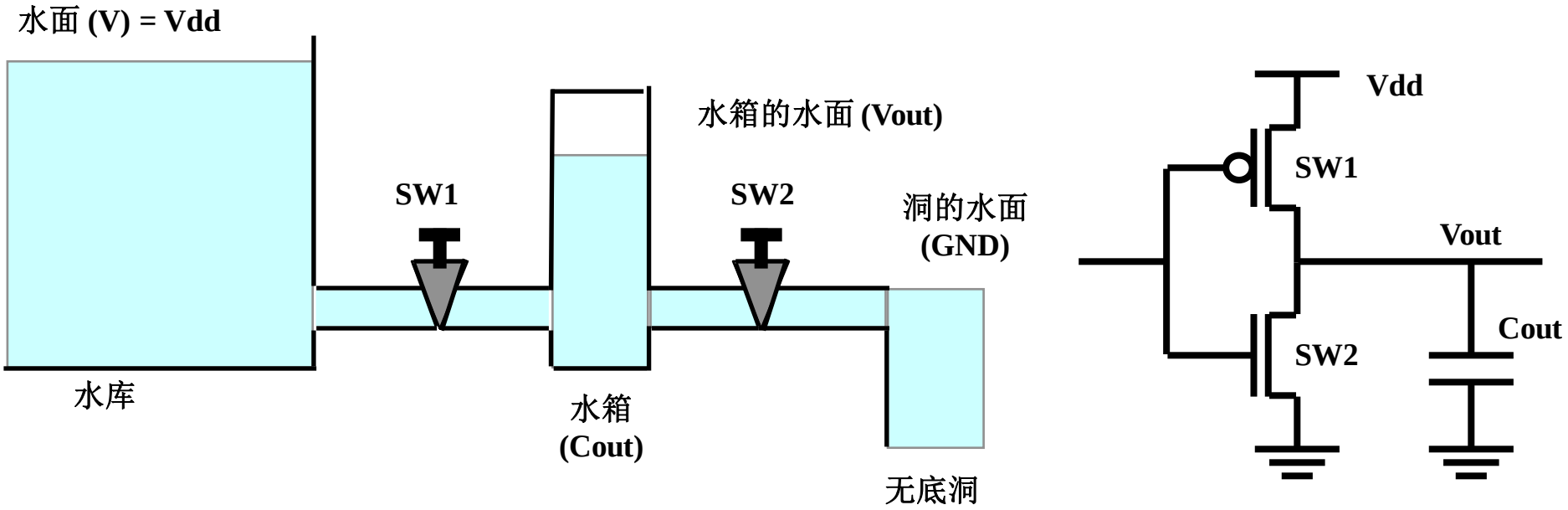
理想（计算机科学）与 现实（电子工程）

- 当输入 0 ⇒ 1时, 输出 1 ⇒ 0，注意：并非立即完成
  - 输出 1 ⇒ 0: 输出电压从高电平 (5v) 变成 0v
- 当输入 1 ⇒ 0时, 输出 0 ⇒ 1，注意：并非立即完成
  - 输出 1 ⇒ 0: 输出电压从0v变成高电平(5v)
- 电压并不能立即变化



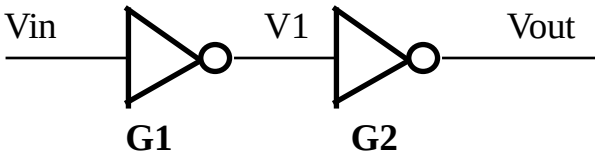


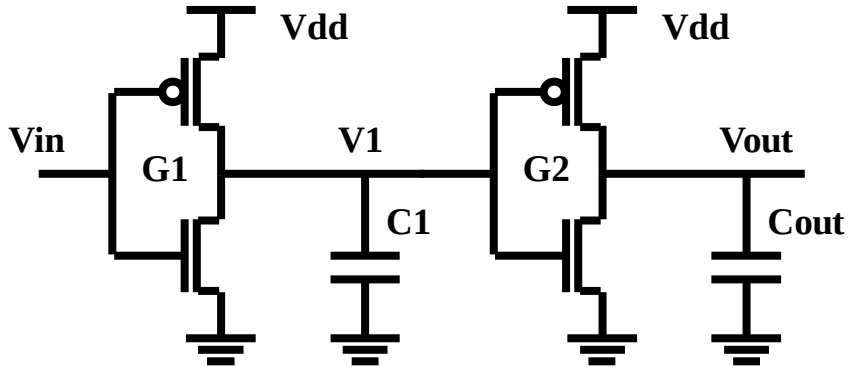
流度定时模型 (Fluid Timing Model)

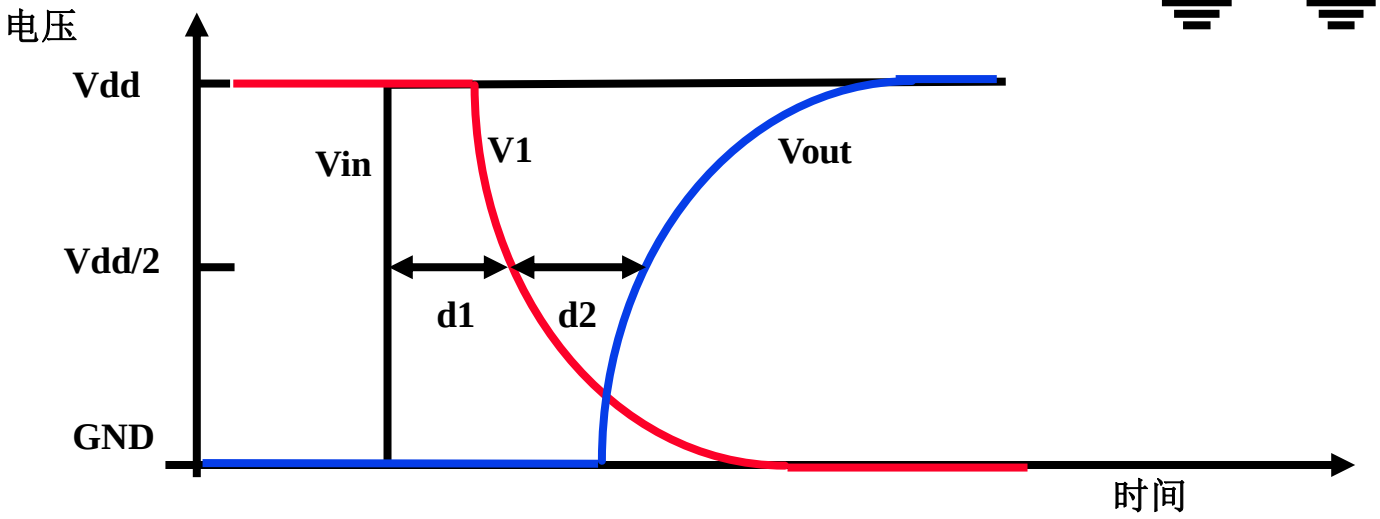


- 水 ⇔ 电荷　　水箱容量 ⇔ 电容 (C)
- 水面 ⇔ 电压　　水流 ⇔ 电荷流动（电流）
- 导管大小 ⇔ 晶体管的电导系数（Strength or conductance of Transistors） (G)
- 装满水箱的时间 ~ C / G

串联（Series Connection）

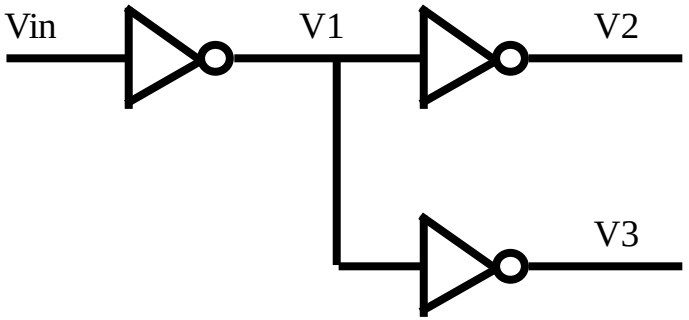


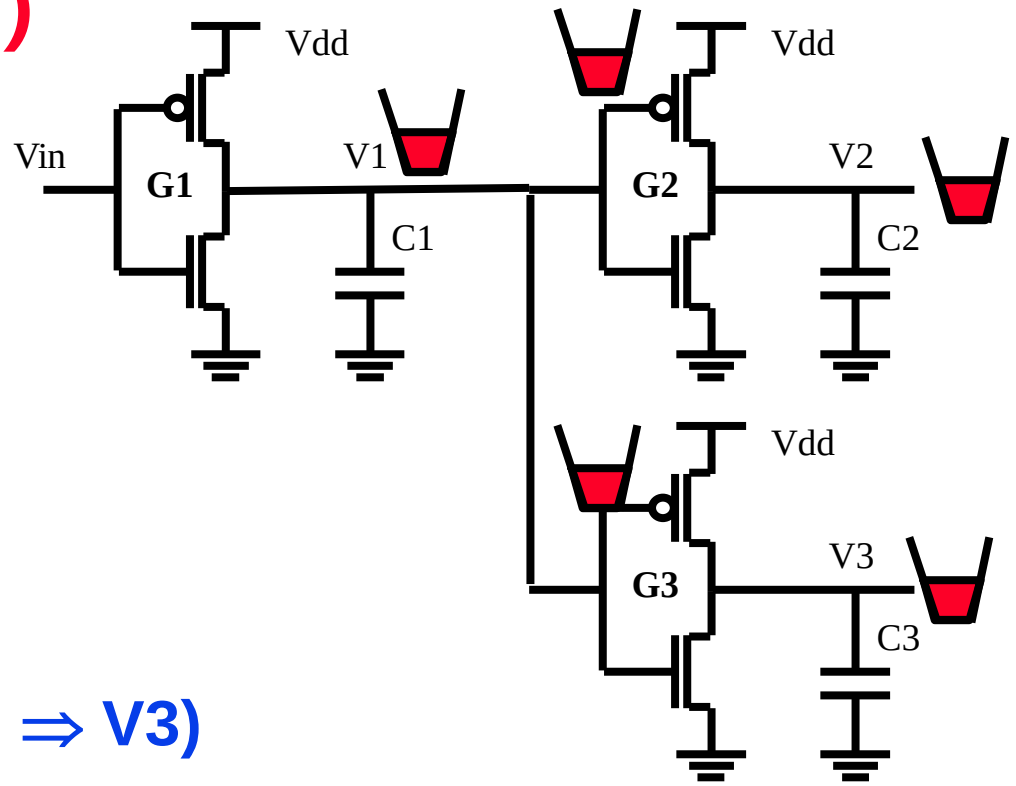




- 总传播延迟（**Total Propagation Delay**） = 所有个体延迟之和 = **d1 + d2**
- 电容 **C1** 由两部分组成:
  - 连接两个门的导线的电容
  - 第二个反向器的输入电容

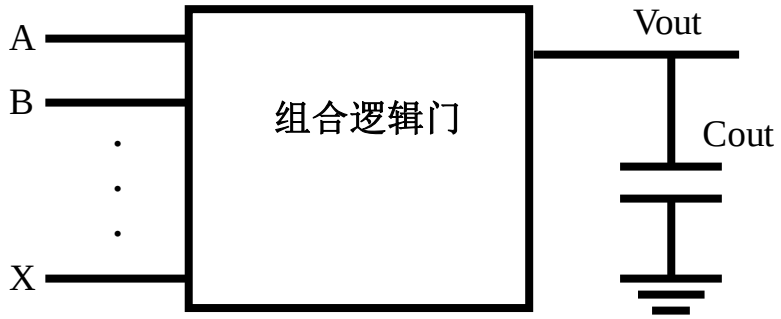
并联（Parallel Connection）

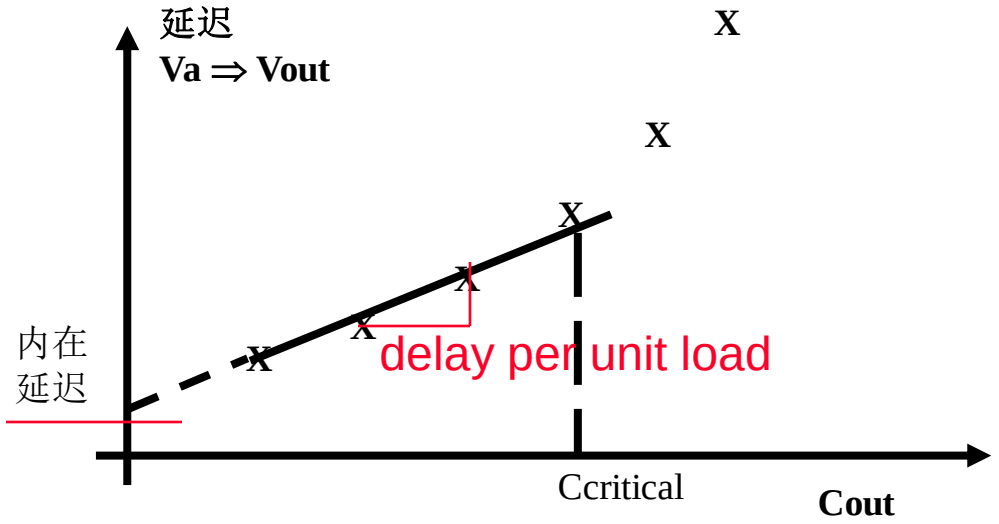




- Delay (Vin ⇒ V2) ! = Delay (Vin ⇒ V3)
  - Delay (Vin ⇒ V2) = Delay (Vin ⇒ V1) + Delay (V1 ⇒ V2)
  - Delay (Vin ⇒ V3) = Delay (Vin ⇒ V1) + Delay (V1 ⇒ V3)
- 关键路径（Critical Path） = N条并联路径中的最长路径
- C1 = 导线电容C + G2的Cin + G3的Cin

常规单元延迟模型（General Cell Delay Model）

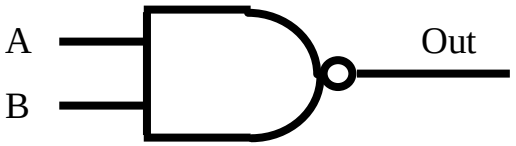




- 约束 B ... X， A的变化将导致 Vout变化
  - 例如：如果这是一个 AND 门, 约束输入 B, C, ... X为高
- 不要用这个门来驱动任何 Cout > Ccritical的负载
- 使用线性方程（linear equation），简化这一模型：
  - Delay (A ⇒ Out) = 内在延迟 + (负载相关延迟) x Cout

刻画一个门电路

- 每个输入的输入电容
- 对于每个输入-输出路径:
  - 对于每个输出变迁类型 (H⇒L, L ⇒ H, H ⇒ Z, L ⇒ Z ... 等等.)
    - 内在延迟 (ns)
    - 负载相关延迟 (ns / fF)
- 示例: 双输入 NAND门



For A and B: Input Load = 61 fF(10-`12F)

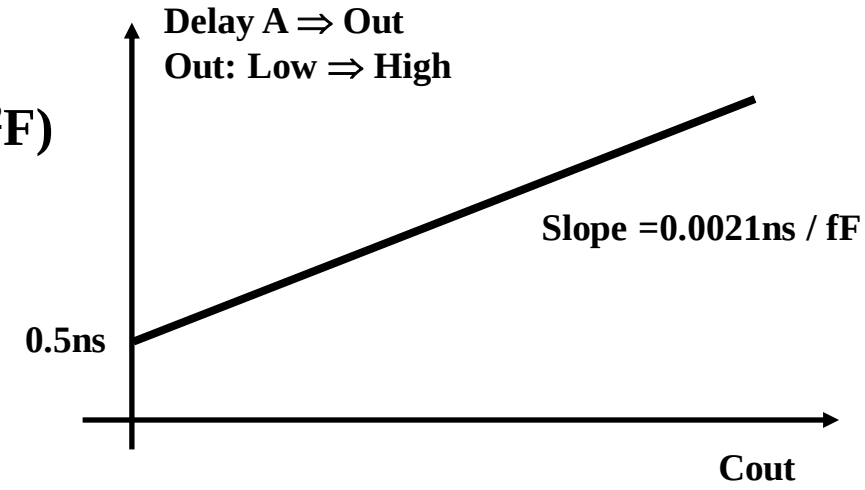
For either A⇒ Out or B ⇒ Out:

TPlh = 0.5ns Tplhf = 0.0021ns / fF

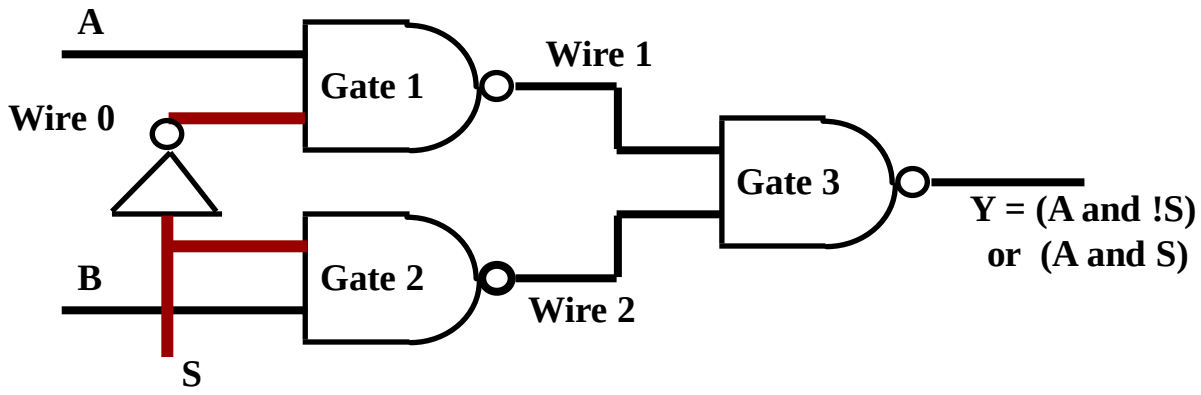
TPhl = 0.1ns TPhlf = 0.0020ns / fF

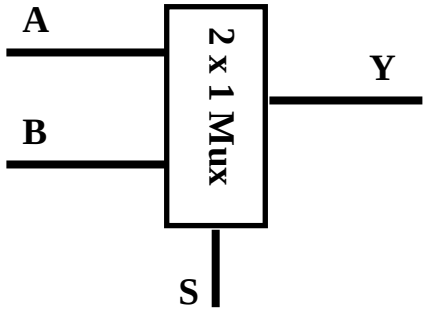
internal delay with output from low to high

slope



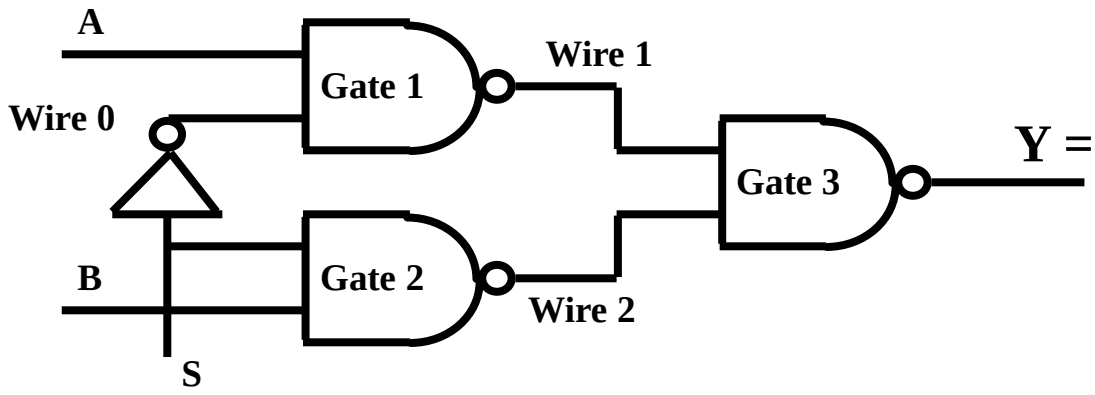
特例: 2 - 1 MUX





- 输入负载
  - A, B: I.L. (NAND) = 61 fF
  - S: I.L. (INV) + I.L. (NAND) = 50 fF + 61 fF = 111 fF
- 负载相关延迟: 与门 3 相同
  - TAYlhf = 0.0021 ns / fF　　TAYhlf = 0.0020 ns / fF
  - TBYlhf = 0.0021 ns / fF　　TBYhlf = 0.0020 ns / fF
  - TSYlhf = 0.0021 ns / fF　　TSYlhf = 0.0020 ns / fF

2 - 1 MUX: 计算内在延迟

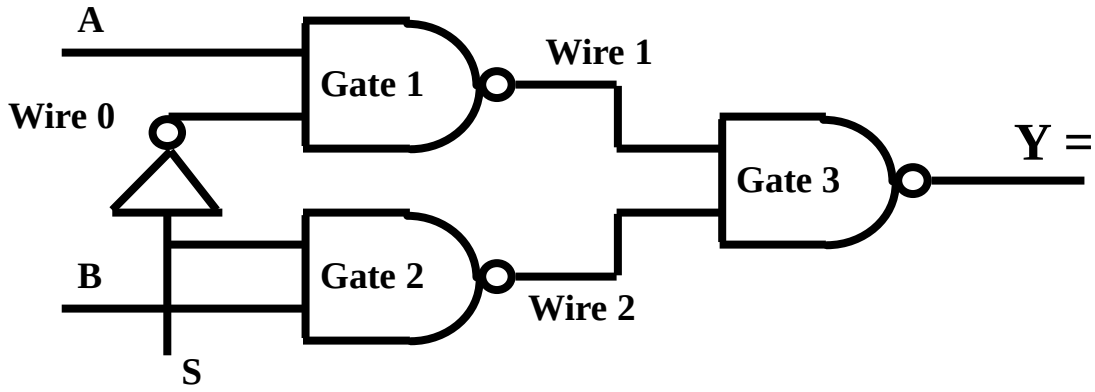


- 内在延迟:
  - A to Y: I.D. G1 + (Wire 1 C + G3 Input C) × L.D.D G1 + I.D. G3
  - B to Y: I.D. G2 + (Wire 2 C + G3 Input C) × L.D.D. G2 + I.D. G3
  - S to Y (最坏情况):

    I.D. Inv + (Wire 0 C + G1 Input C) × L.D.D. Inv + A到Y的内在 延迟

- 可以大致估算 Wire 1 C的作用，：
  - 假设 Wire 1 同 所有与它相连的门的电容相同
  - 门1需要取驱动的总 C: 2.0 x 门 3 的输入C

2 - 1 MUX: 计算内在延迟(续)

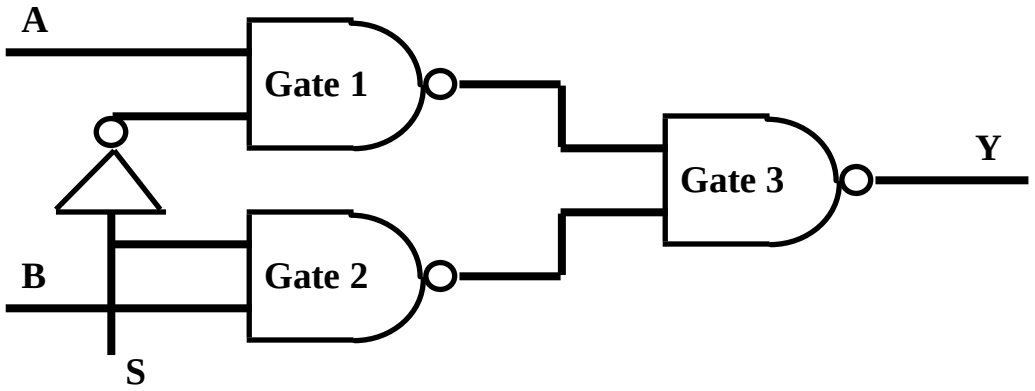


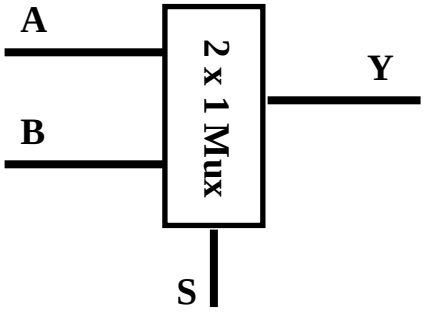
- 内在延迟:
  - A 至 Y: I.D. G1 + (Wire 1 C + G3 Input C) × L.D.D G1 + I.D. G3
  - B 至 Y: I.D. G2 + (Wire 2 C + G3 Input C) × L.D.D. G2 + I.D. G3
  - S 至 Y (最坏情况):

    I.D. Inv + (Wire 0 C + G1 Input C) × L.D.D. Inv + A至Y的内在延迟
- 特例:
  - TAYlh = TPhl G1 + (2.0 × 61 fF) × TPhl G1 + TPlh G3
    = 0.1ns + 122 fF × 0.0020ns/fF + 0.5ns = 0.844ns

internal delay from Input A to output Y with output Y doing a Low to High transition

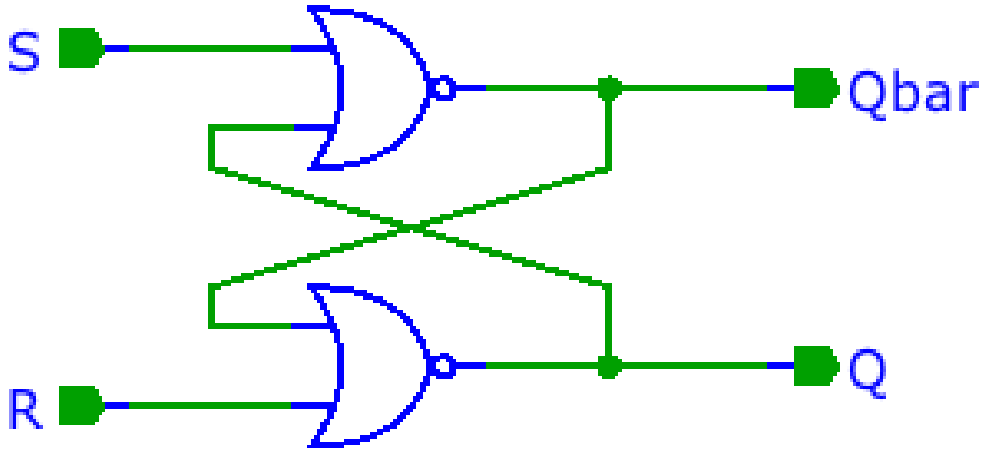
抽象: 2 - 1 MUX

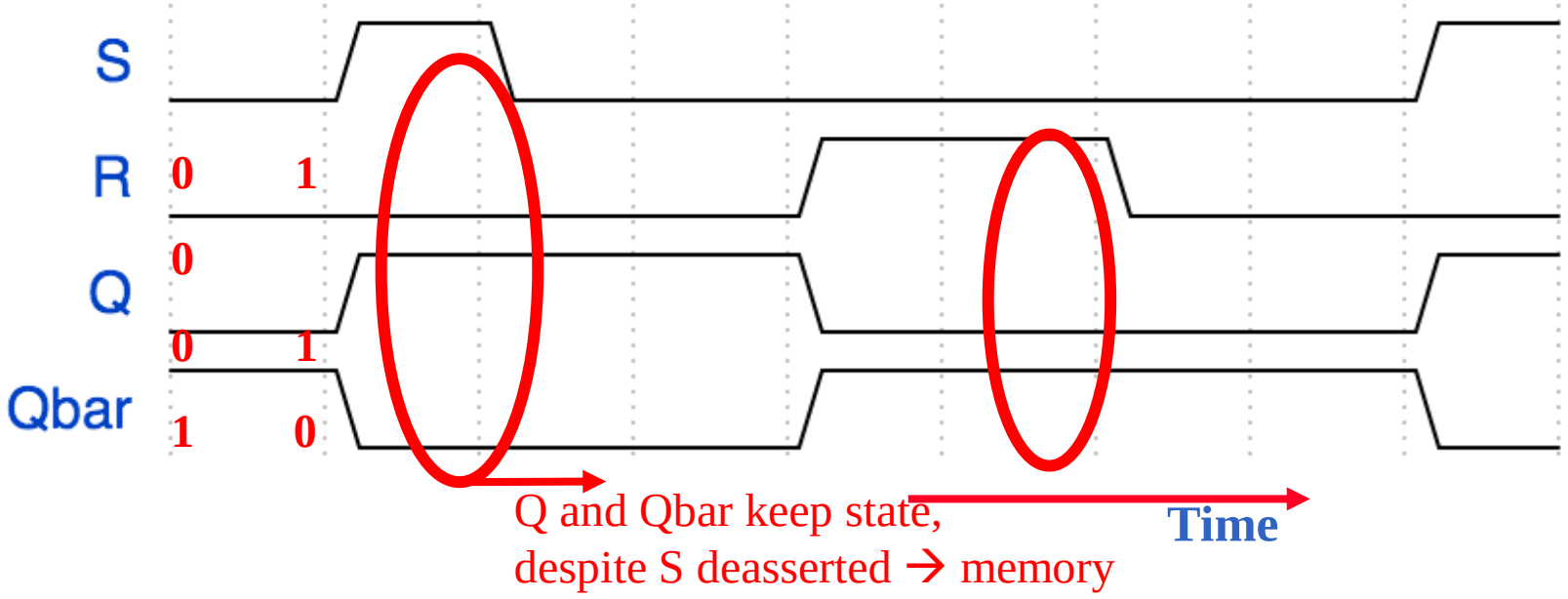




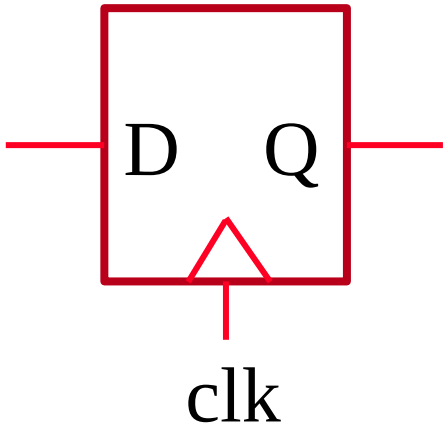
- 输入负载: A = 61 fF, B = 61 fF, S = 111 fF
- 负载相关延迟:
  - TAYlhf = 0.0021 ns / fF　　TAYhlf = 0.0020 ns / fF
  - TBYlhf = 0.0021 ns / fF　　TBYhlf = 0.0020 ns / fF
  - TSYlhf = 0.0021 ns / fF　　TSYlhf = 0.0020 ns / f F
- 内在延迟:
  - TAYlh = TPhl G1 + (2.0 × 61 fF) × TPhlf G1 + TPlh G3
    = 0.1ns + 122 fF × 0.0020ns/fF + 0.5ns = 0.844ns
  - 趣味题: TAYhl, TBYlh, TSYlh, TSYlh

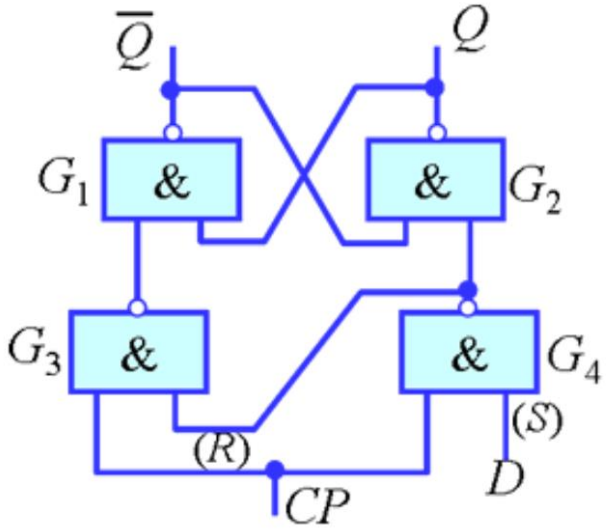
RS Latch


S
Qbar
R
Q


S
R
Q
Qbar
0
1
0
0
1
1
0
Q and Qbar keep state,
despite S deasserted → memory
Time

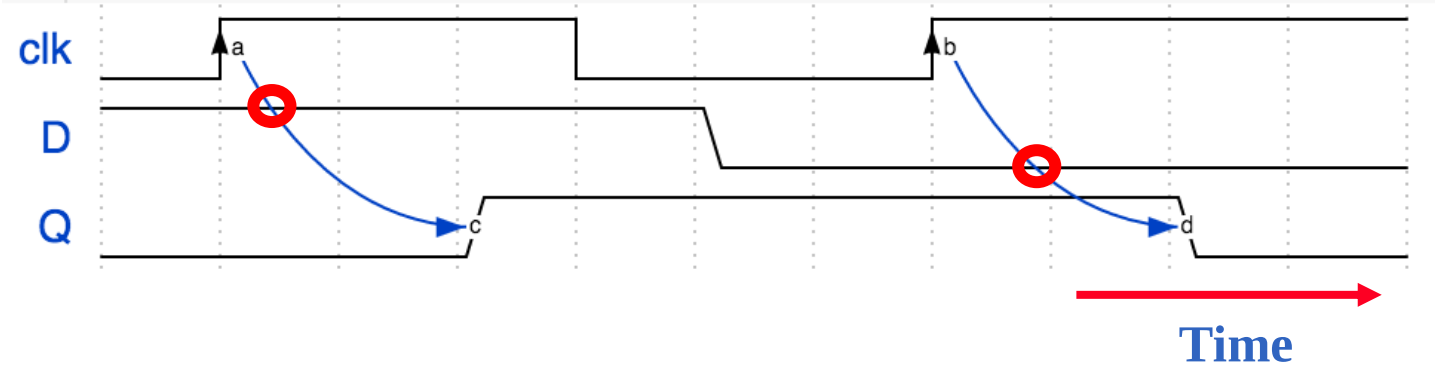
(Positive) Edge Triggered *Flip-Flop*



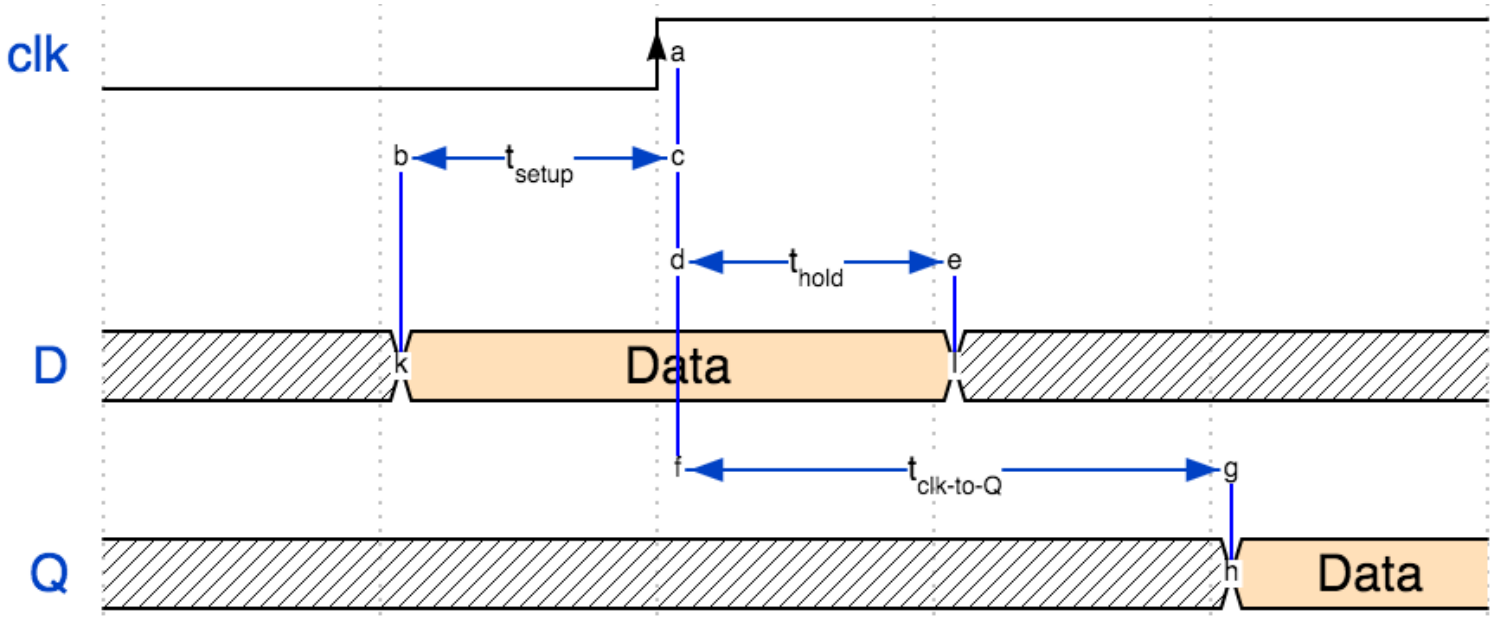


RS *latch* operates on input levels
D *FF* operates on (clock) edges

Clock "clk"

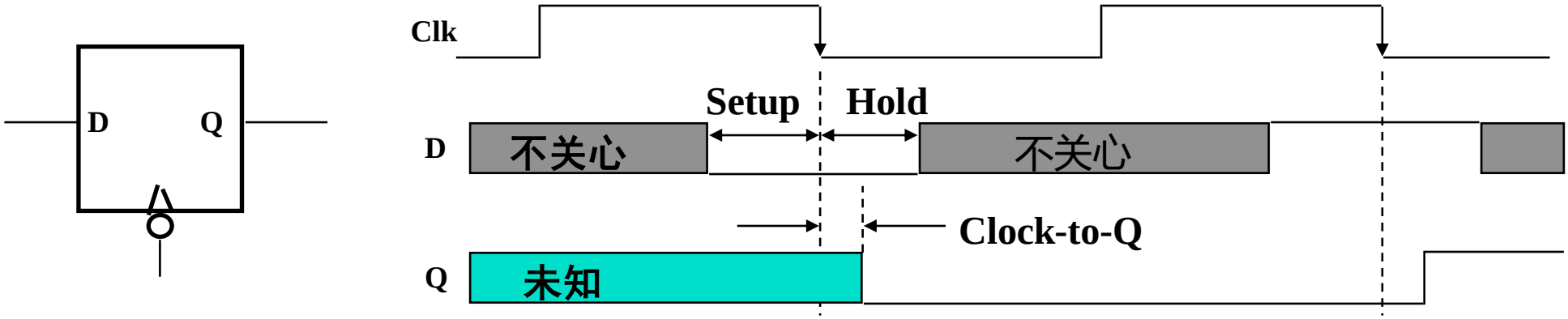


Flip-Flop Timing



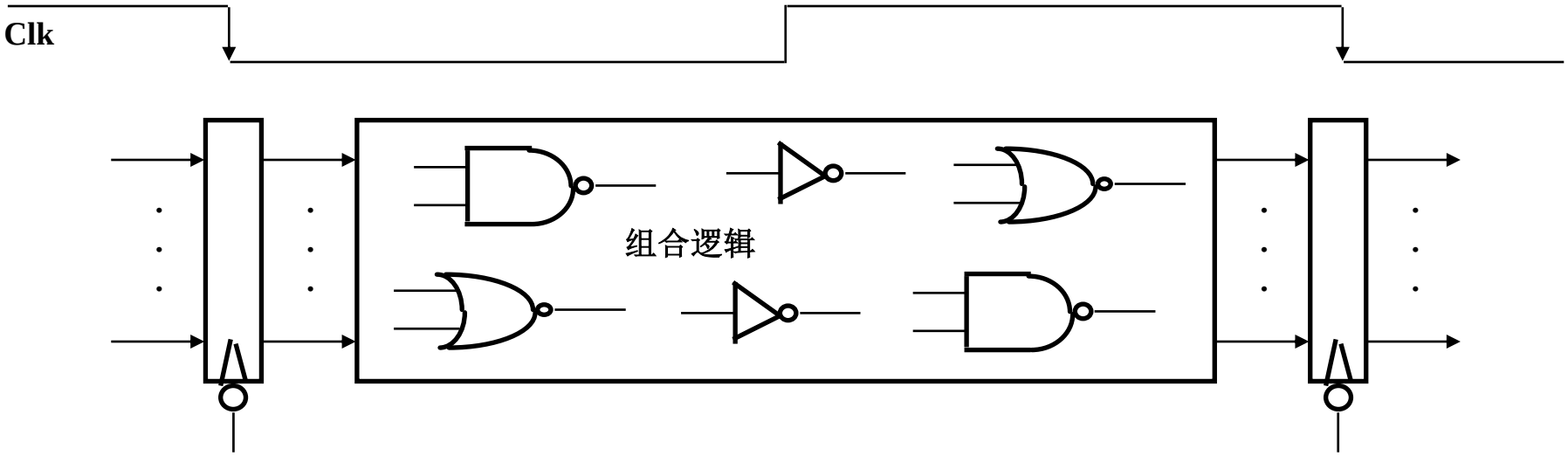
Time	Description
t_{setup}	time during which D must not change *before* positive clock edge
t_{hold}	time during which D must not change *after* positive clock edge
$t_{clk\text{-}to\text{-}Q}$	delay after positive clock edge after which D appears at Q output

存储元件的时序模型



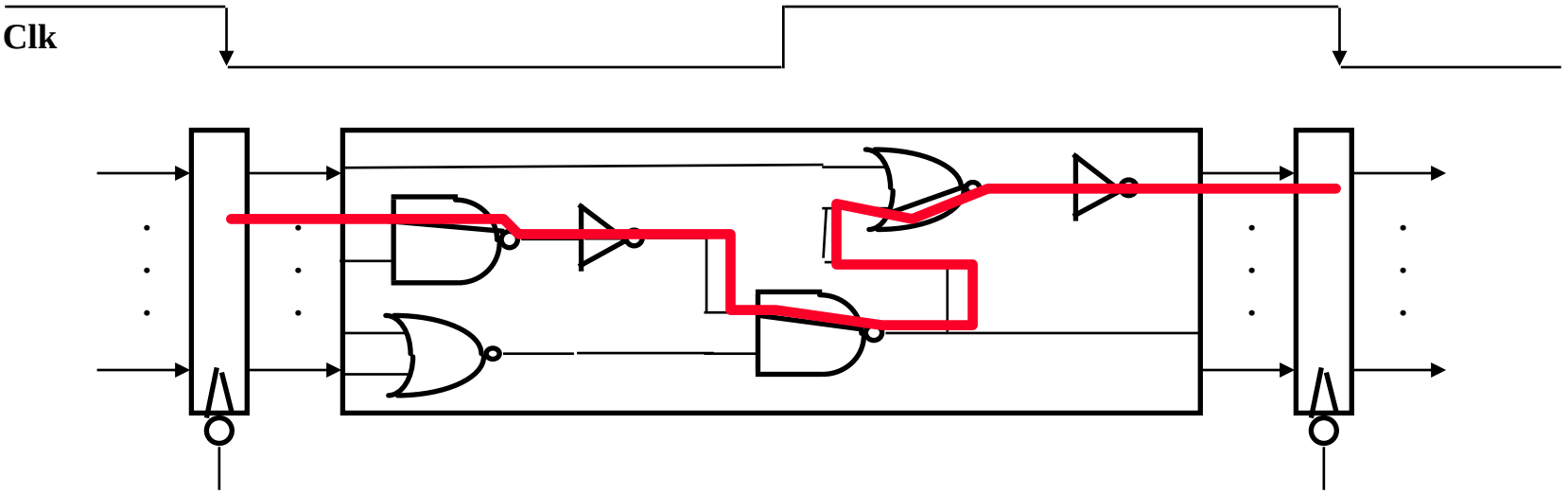
- 建立时间（Setup Time）：在触发时钟边沿 之前 输入必须稳定
- 保持时间（Hold Time）：在触发时钟边沿 之后 输入必须保持
- Clock-to-Q time:
  - 在触发时钟边沿，输出并不能立即变化
  - 与逻辑门的延迟类似，也由两部分组成:
    - 内在 Clock-to-Q时间
    - 负载相关 Clock-to-Q时间

时钟同步方法 (Clocking Methodology)



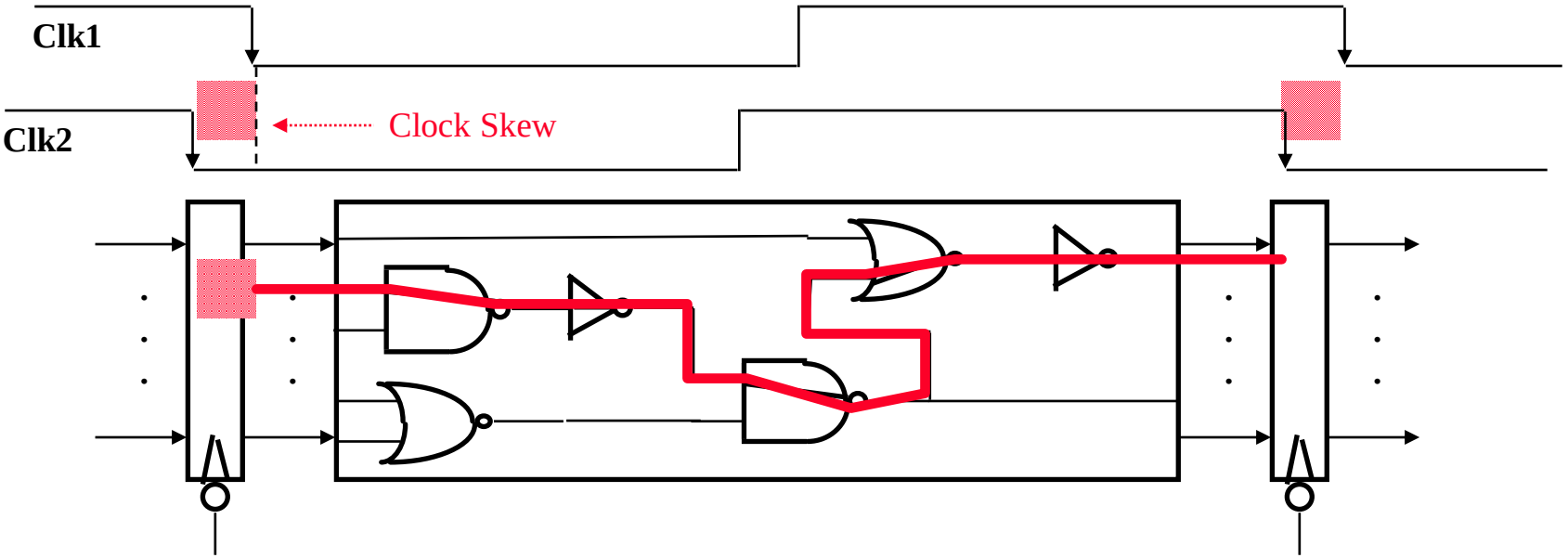
- 所有的存储单元都由同一时钟边沿到来时进行时钟同步（clocked）
- 组合逻辑电路部分:
  - 在每个时钟信号有效时，输入才改变
  - 在下一次时钟信号有效之前， 所有输出 必须 稳定

关键路径（Critical Path）和 时钟周期 （Cycle Time）



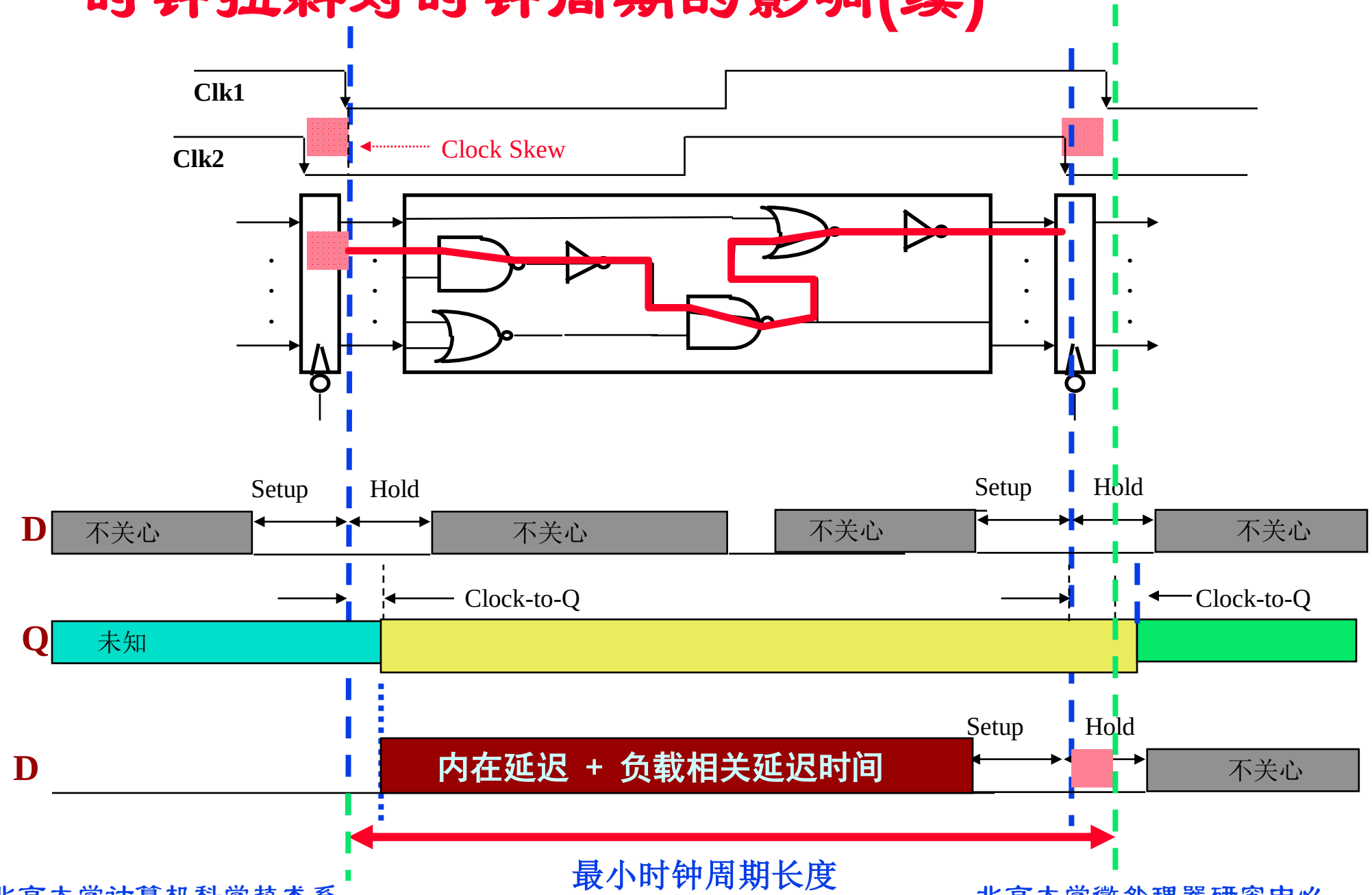
- 关键路径: 任何两个存储设备之间的最慢路径
- 时钟时间 依赖于 这条关键路径
- 具体而言, 时钟时间必须大于:
  - **Clock-to-Q + 穿越组合逻辑的最长延迟 + Setup**

时钟扭斜（Clock Skew）对时钟周期的影响



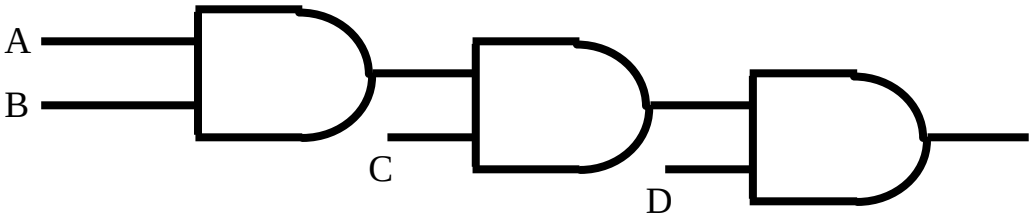
- 考虑时钟时间的最坏情况:
  - 输入寄存器 看见 CLK1
  - 输出寄存器 看见 CLK2
- 时钟时间 >= CLK-to-Q + 最长延迟路径 + 建立时间 + 时钟扭斜

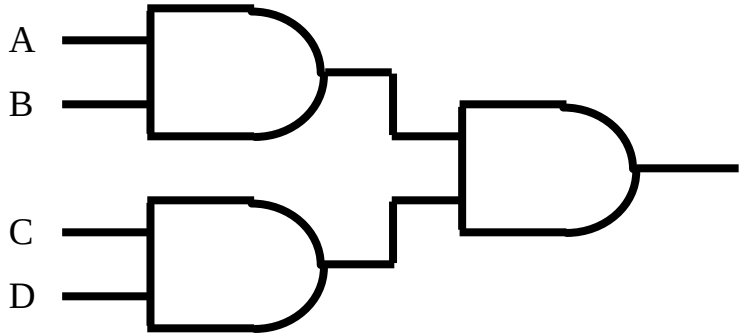
时钟扭斜对时钟周期的影响(续)



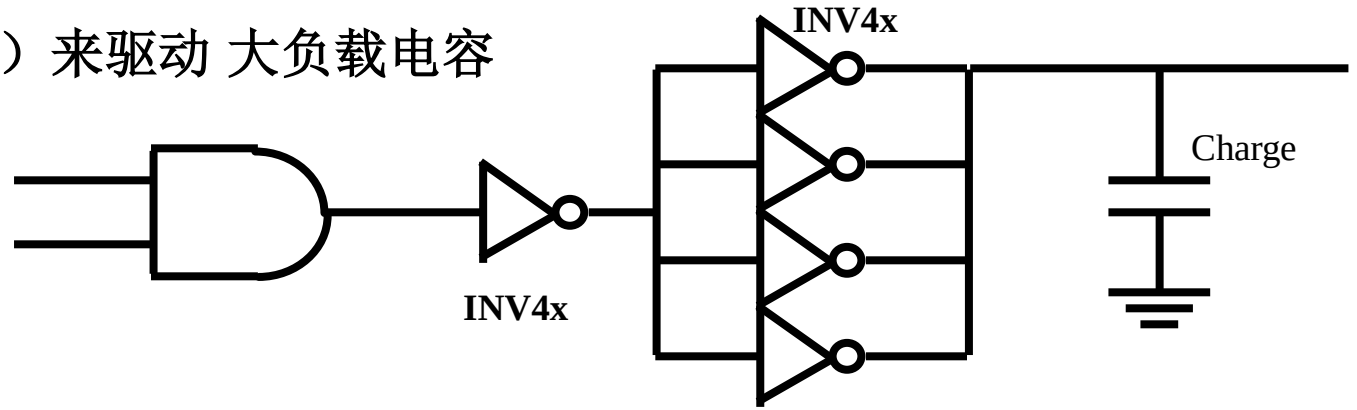
减小时钟周期的窍门

减少门的级数

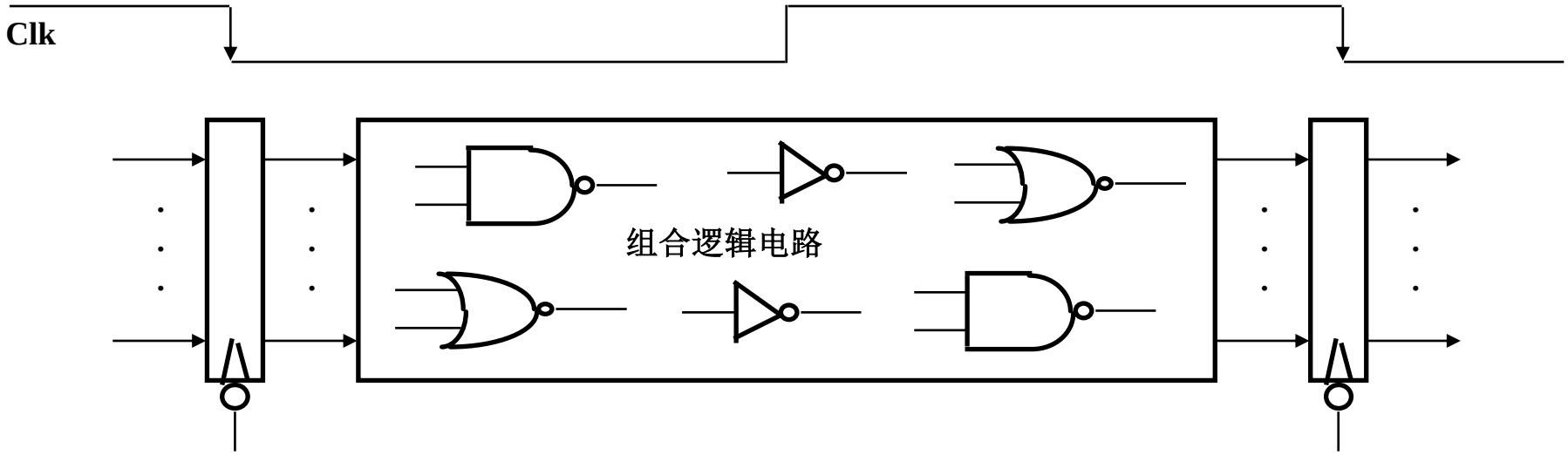




- 注意负载情况
  - 用一个门驱动很多门 是 坏方法
  - 避免一个小门来驱动长线
- 使用多段（**stages**）来驱动 大负载电容

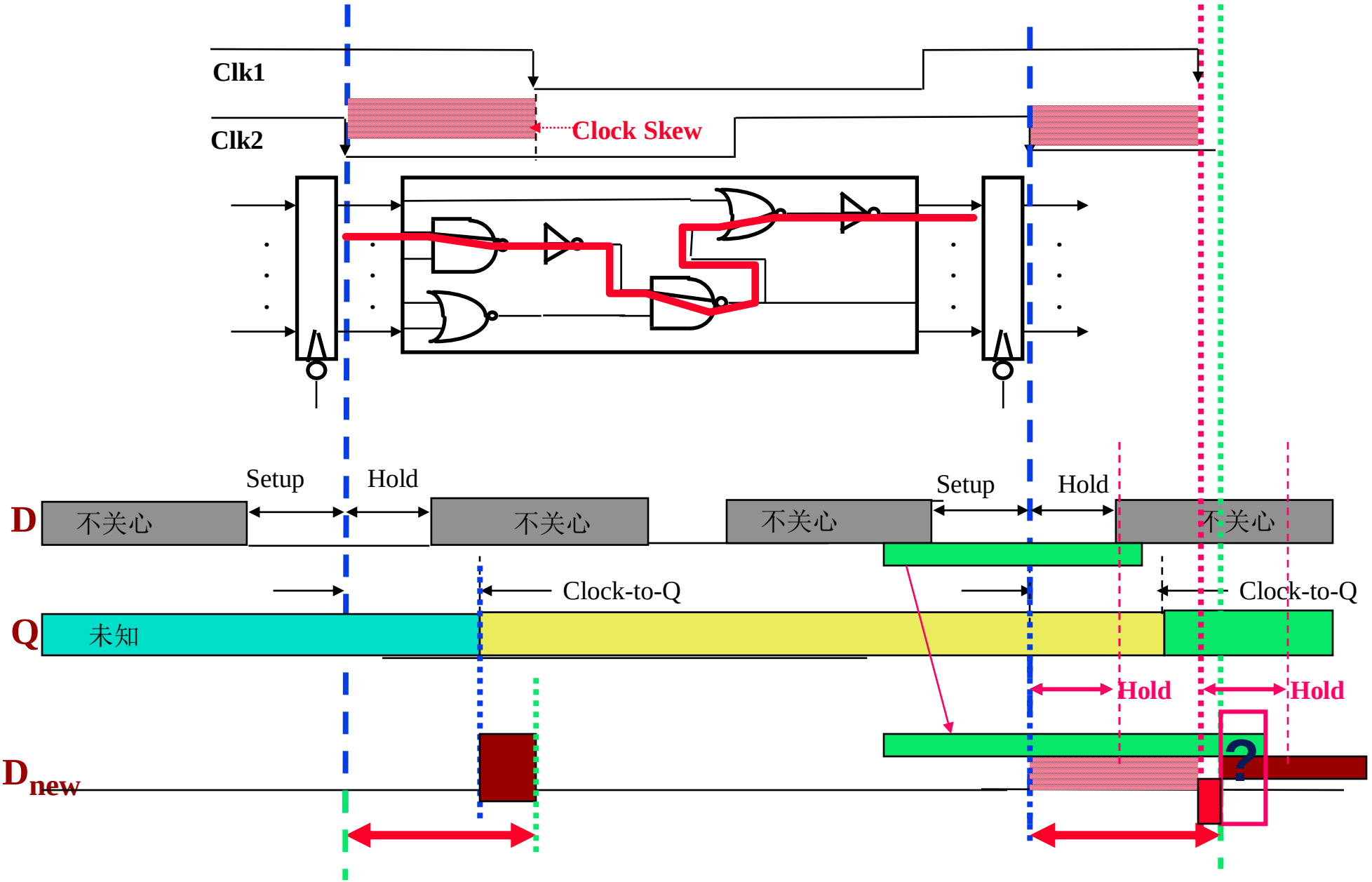


如何避免保持时间违例（Hold Time Violation）？

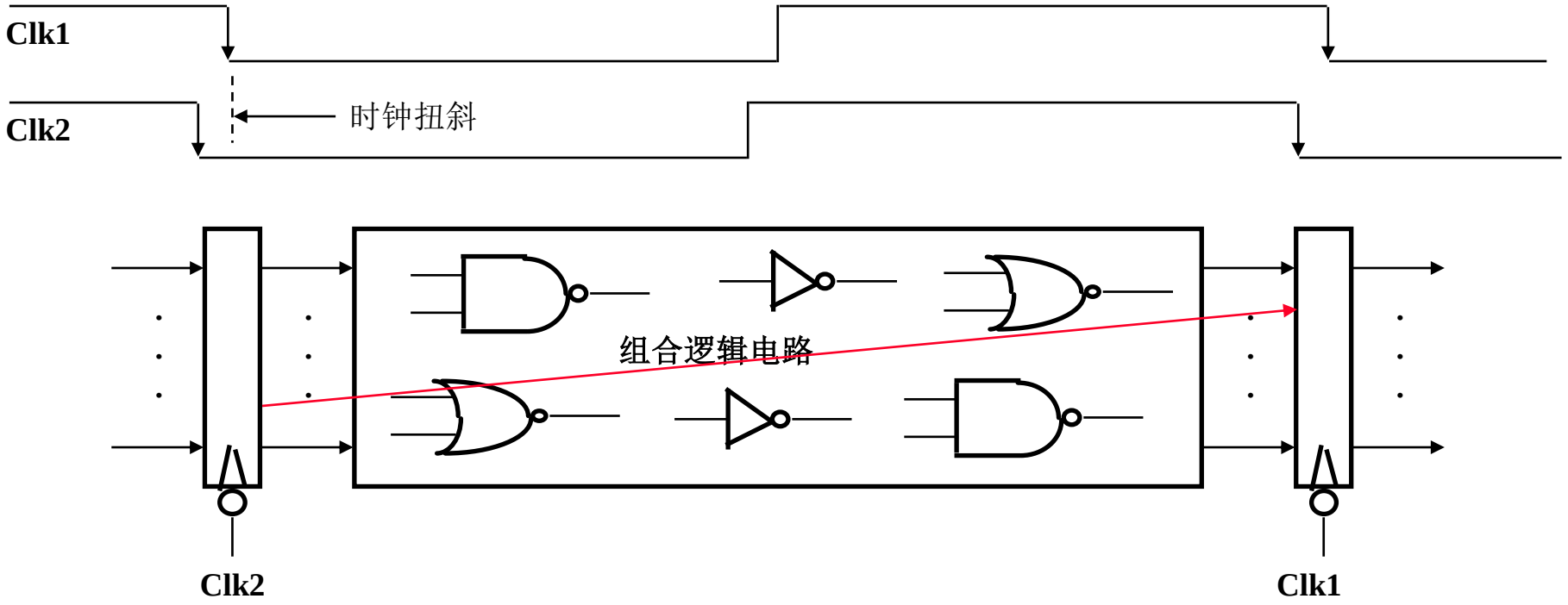


- 保持时间需求:
  - 在时钟有效之后，到寄存器的输入 一定不要 立即变化
- 在边沿触发的时钟同步方案中，经常会遇到这一情况
- CLK-to-Q + 最短延迟路径 必须大于 保持时间

时钟扭斜对保持时间的影响



时钟扭斜对保持时间的影响(续)



- 对保持时间考虑，最坏的情况:
  - 输入寄存器 看见 CLK2
  - 输出寄存器 看见 CLK1
- (CLK-to-Q + 最短延迟路径 - 时钟扭斜) > 保持时间

小结

- 性能和技术工艺发展趋势
  - 尽可能保持设计简单，以充分利用最新技术工艺的优势
  - CMOS 反向器 和 CMOS 逻辑门
- 延迟建模和门的特征描述
  - 延迟 = 内在延迟 + (负载相关延迟 x 输出负载)
- 时钟同步方法 和时序考虑
  - 最简单的同步方法
  - 所有的存储单元使用相同的时钟边沿
  - 时钟时间 >= CLK-to-Q + 最长延迟路径 + 建立 + 时钟扭斜
  - CLK-to-Q + 最短延迟路径 - 时钟扭斜 > 保持时间