
CS152
Computer Architecture and Engineering
Lecture 3: Review Technology & Delay Modeling

September 3, 1997

Dave Patterson (<http://cs.berkeley.edu/~patterson>)

lecture slides: <http://www-inst.eecs.berkeley.edu/~cs152/>

Outline of Today's Lecture

- **Review (1 minute)**
- **ISA, Performance Wrap-up (5 minutes)**
- **Performance and Technology (10 minutes)**
- **Administrative Matters and Questions (2 minutes)**
- **Delay Modeling and Gate Characterization (20 minutes)**
- **Questions and Break (5 minutes)**
- **Clocking Methodologies and Timing Considerations (25 minutes)**

Summary: Salient features of MIPS I

- **32-bit fixed format inst** (3 formats)
- **32 32-bit GPR** (R0 contains zero) and 32 FP registers (and HI LO)
 - partitioned by software convention
- **3-address, reg-reg arithmetic instr.**
- **Single address mode for load/store:** base+displacement
 - no indirection, scaled
- 16-bit immediate plus LUI**
- **Simple branch conditions**
 - compare against zero or two registers for =, ≠
 - no integer condition codes
- **Delayed branch**
 - execute instruction after the branch (or jump) even if the branch is taken (Compiler can fill a delayed branch with useful work about 50% of the time)

Summary: Instruction set design (MIPS)

- Use general purpose registers with a load-store architecture: YES
- Provide at least 16 general purpose registers plus separate floating-point registers: 31 GPR & 32 FPR
- Support basic addressing modes: displacement (with an address offset size of 12 to 16 bits), immediate (size 8 to 16 bits), and register deferred; : YES: 16 bits for immediate, displacement (disp=0 => register deferred)
- All addressing modes apply to all data transfer instructions : YES
- Use fixed instruction encoding if interested in performance and use variable instruction encoding if interested in code size : Fixed
- Support these data sizes and types: 8-bit, 16-bit, 32-bit integers and 32-bit and 64-bit IEEE 754 floating point numbers: YES
- Support these simple instructions, since they will dominate the number of instructions executed: load, store, add, subtract, move register-register, and, shift, compare equal, compare not equal, branch (with a PC-relative address at least 8-bits long), jump, call, and return: YES, 16b
- Aim for a minimalist instruction set: YES

Evaluating Instruction Sets?

Design-time metrics:

- Can it be implemented, in how long, at what cost?
- Can it be programmed? Ease of compilation?

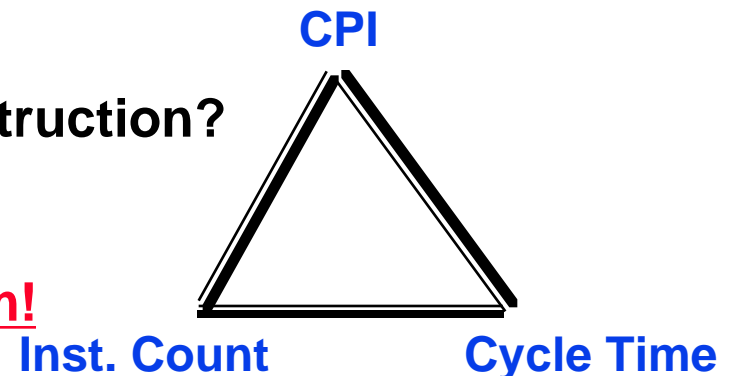
Static Metrics:

- How many bytes does the program occupy in memory?

Dynamic Metrics:

- How many instructions are executed?
- How many bytes does the processor fetch to execute the program?
- How many clocks are required per instruction?
- How "lean" a clock is practical?

Best Metric: Time to execute the program!



NOTE: this depends on instructions set, processor organization, and compilation techniques.

Review: Aspects of CPU Performance

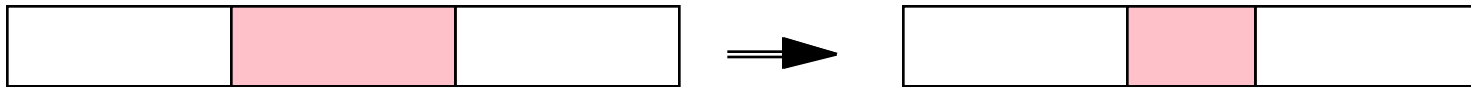
$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	instr count	CPI	clock rate
Program	X		
Compiler	X	X	
Instr. Set	X	X	
Organization		X	X
Technology			X

Amdahl's Law

Speedup due to enhancement E:

$$\text{Speedup}(E) = \frac{\text{ExTime w/o E}}{\text{ExTime w/ E}} = \frac{\text{Performance w/ E}}{\text{Performance w/o E}}$$

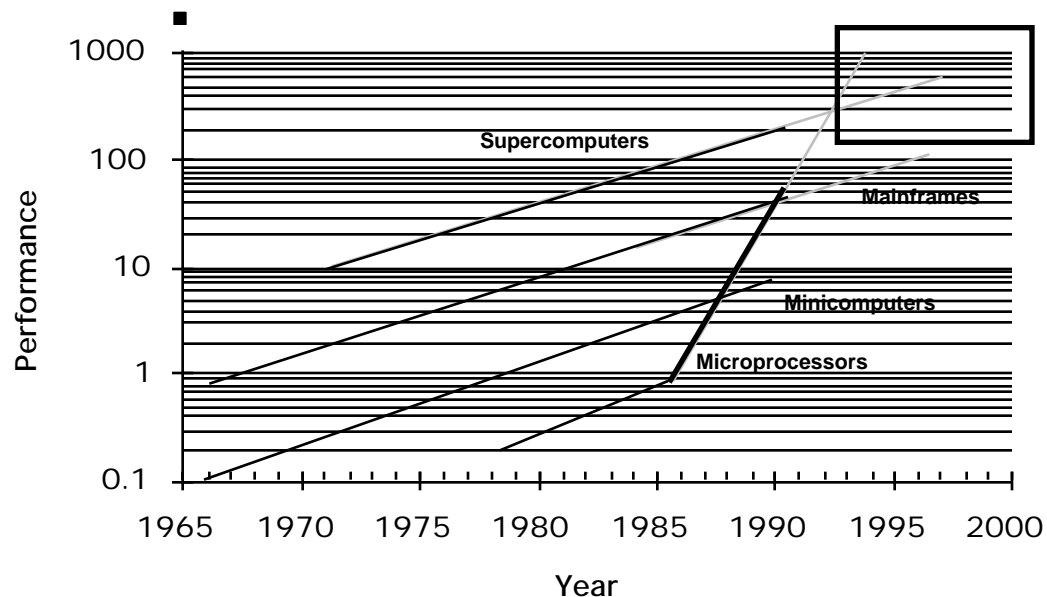


Suppose that enhancement E accelerates a fraction F of the task by a factor S and the remainder of the task is unaffected then,

$$\text{ExTime}(\text{with E}) \leq ((1-F) + F/S) \times \text{ExTime}(\text{without E})$$

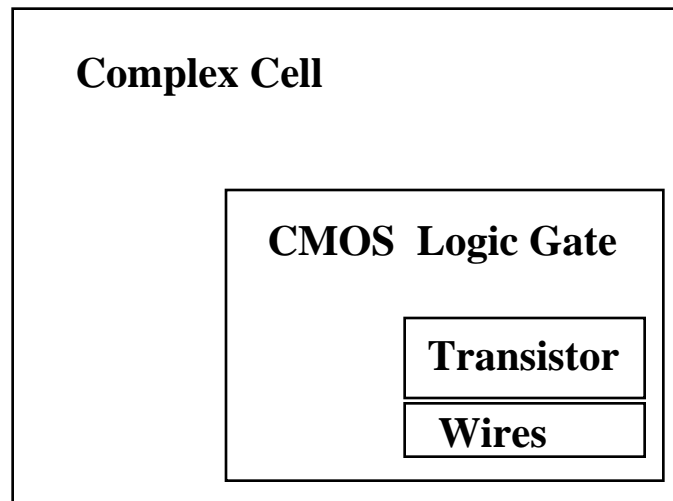
$$\text{Speedup}(\text{with E}) \leq \frac{1}{(1-F) + F/S}$$

Performance and Technology Trends



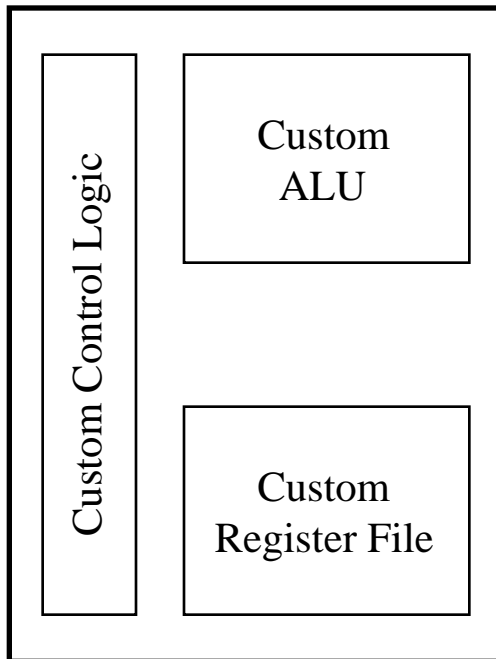
- **Technology Power: $1.2 \times 1.2 \times 1.2 = 1.7 \times / \text{year}$**
 - **Feature Size: shrinks 10% / yr. \Rightarrow Switching speed improves 1.2 / yr.**
 - **Density: improves 1.2x / yr.**
 - **Die Area: 1.2x / yr.**
- **The lesson of RISC is to keep the ISA as simple as possible:**
 - **Shorter design cycle \Rightarrow fully exploit the advancing technology (~3yr)**
 - **Advanced branch prediction and pipeline techniques**
 - **Bigger and more sophisticated on-chip caches**

Technology => Performance

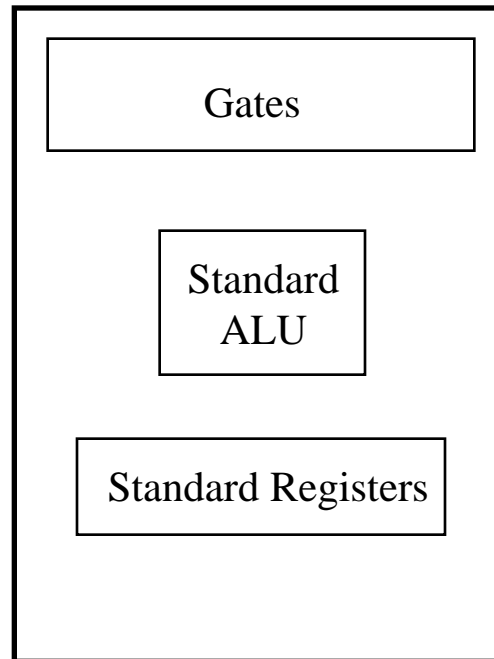


Range of Design Styles

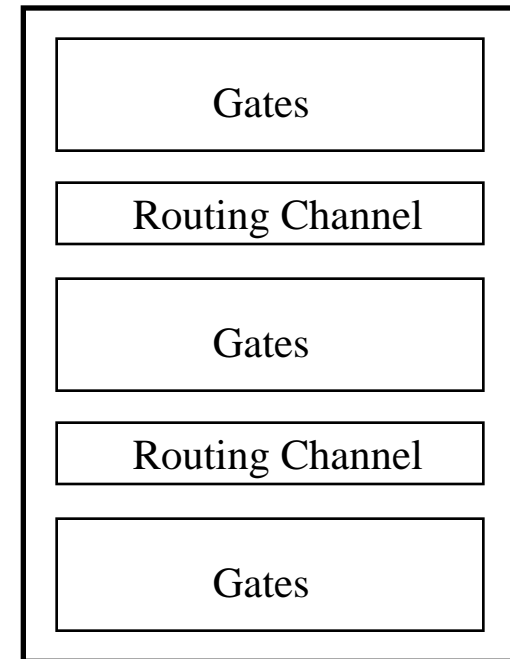
Custom Design



Standard Cell



Gate Array/FPGA/CPLD



Performance

Design Complexity (Design Time)

Compact

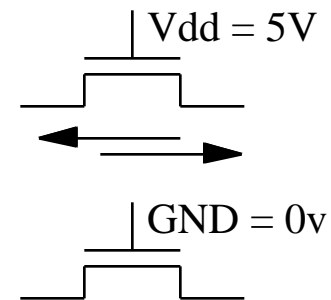
Longer wires

Basic Technology: CMOS

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

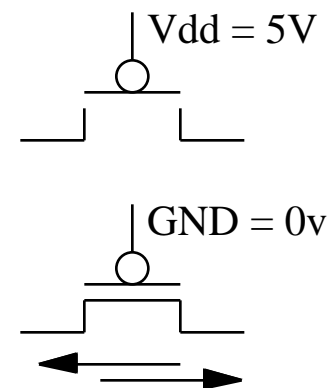
- **NMOS Transistor**

- **Apply a HIGH (V_{dd}) to its gate turns the transistor into a “conductor”**
- **Apply a LOW (GND) to its gate shuts off the conduction path**



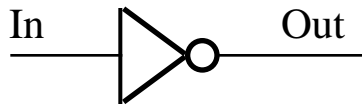
- **PMOS Transistor**

- **Apply a HIGH (V_{dd}) to its gate shuts off the conduction path**
- **Apply a LOW (GND) to its gate turns the transistor into a “conductor”**

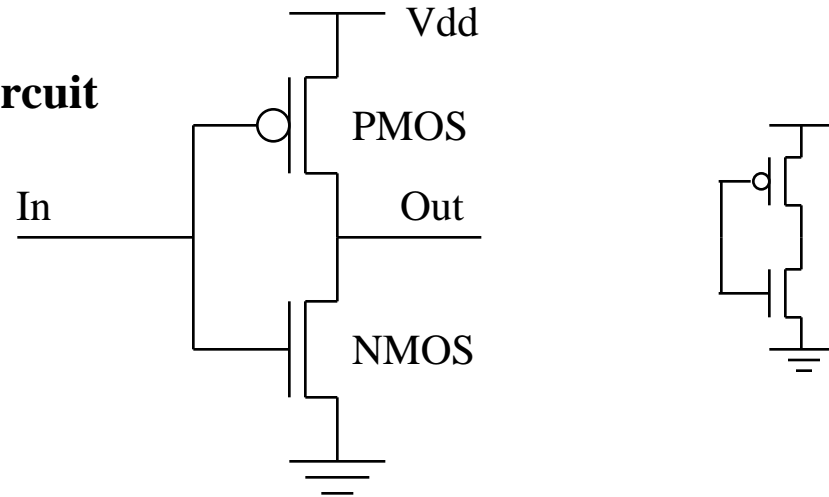


Basic Components: CMOS Inverter

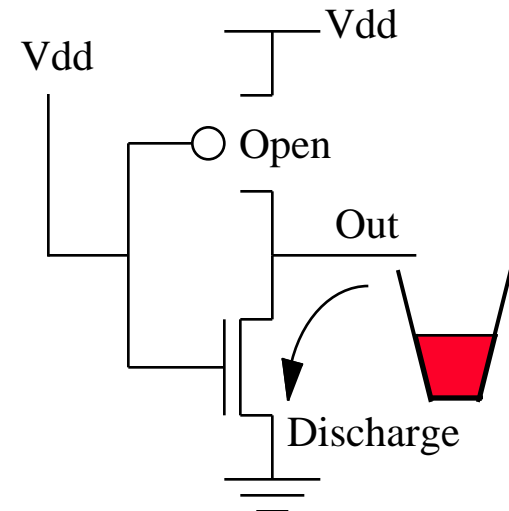
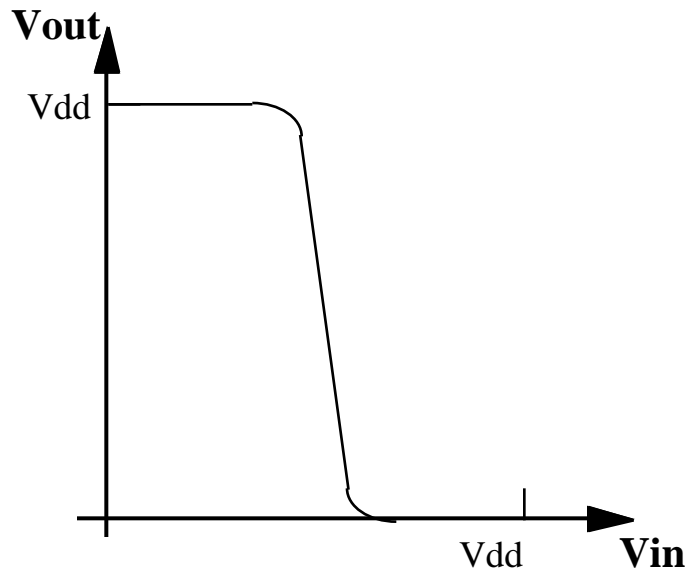
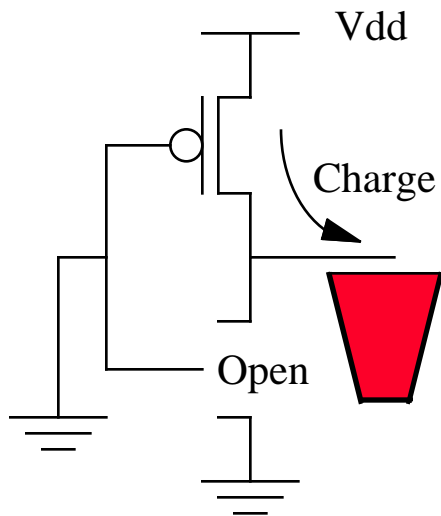
Symbol



Circuit

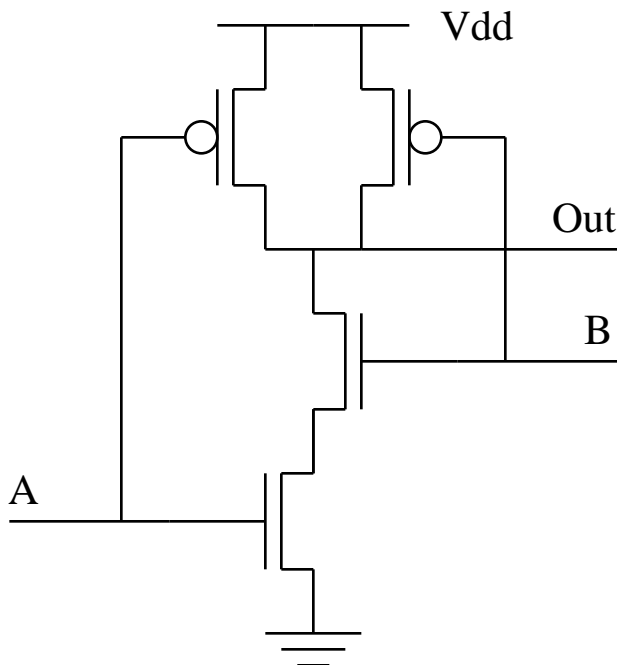
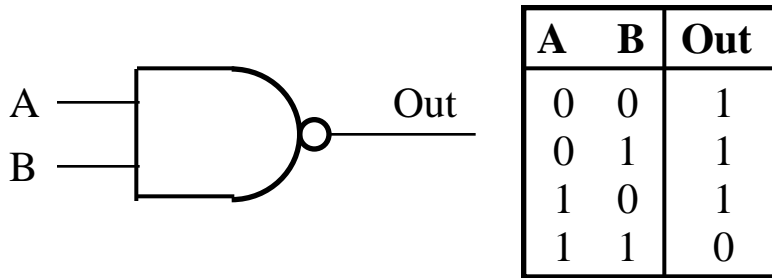


◦ Inverter Operation

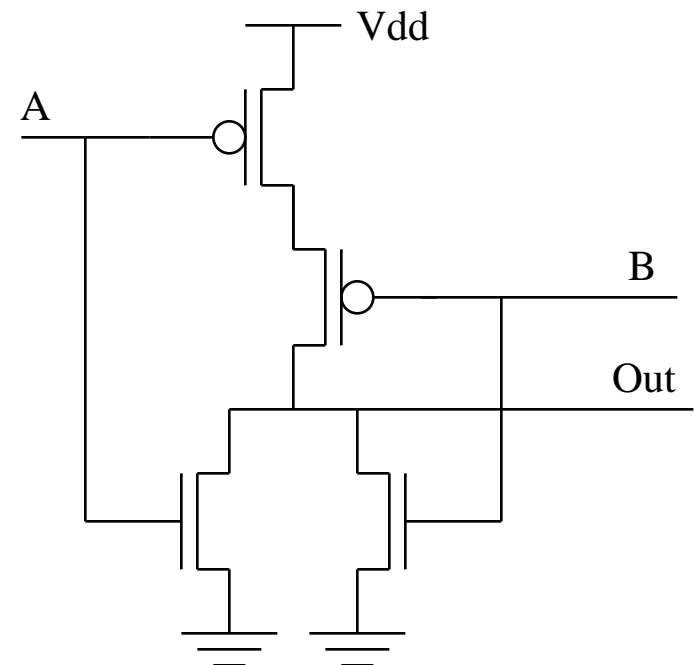
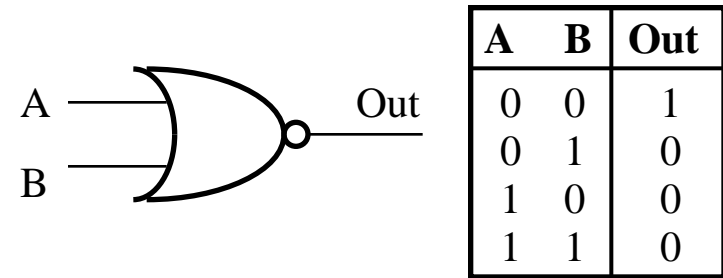


Basic Components: CMOS Logic Gates

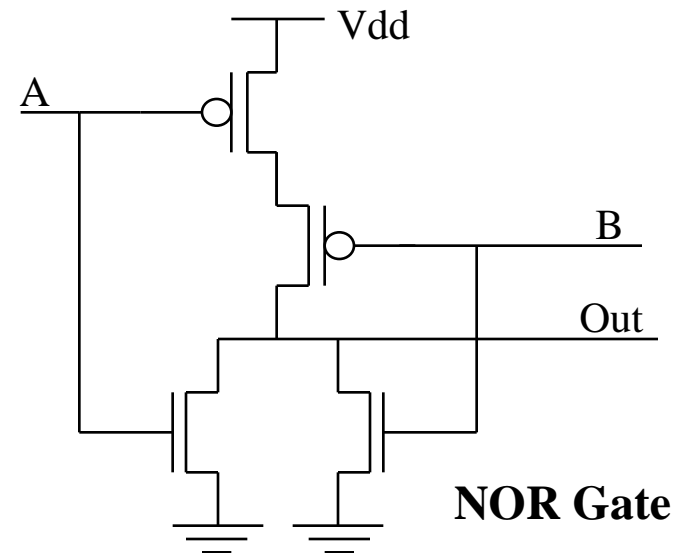
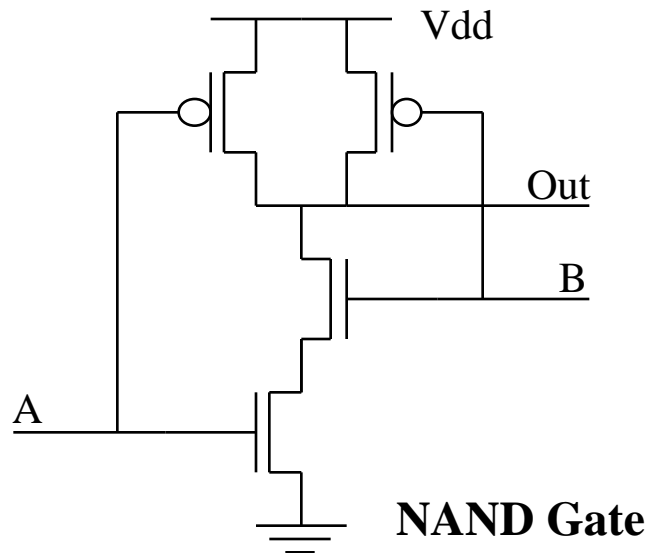
NAND Gate



NOR Gate



Gate Comparison



- If PMOS transistors is faster:
 - It is OK to have PMOS transistors in series
 - NOR gate is preferred
 - NOR gate is preferred also if H -> L is more critical than L -> H
- If NMOS transistors is faster:
 - It is OK to have NMOS transistors in series
 - NAND gate is preferred
 - NAND gate is preferred also if L -> H is more critical than H -> L

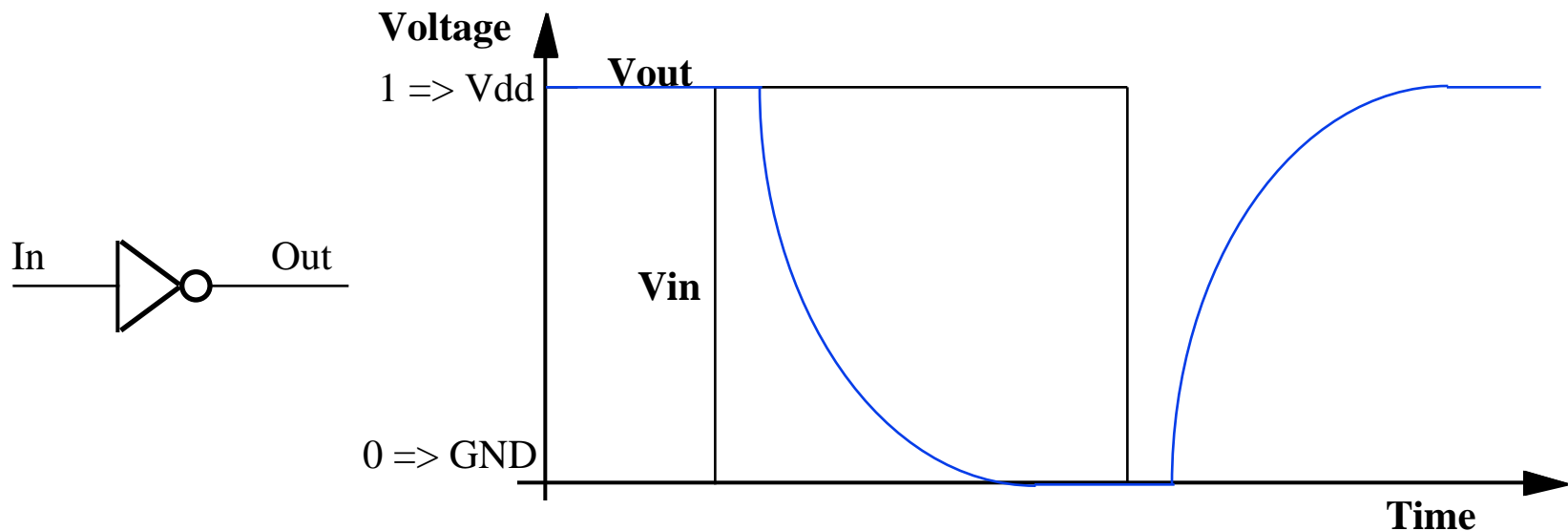
Administrative Matters

**CS152 news group: ucb.class.cs152
(email cs152@cory with specific questions)**

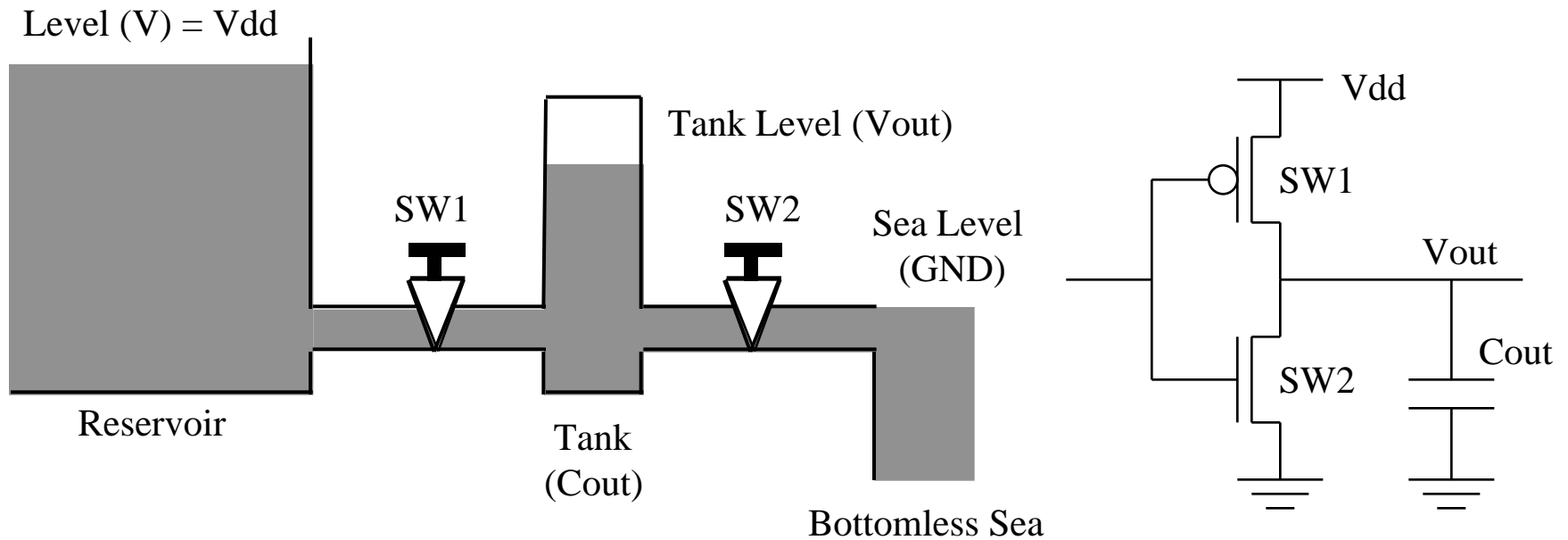
- **Slides, handouts available via WWW:
<http://www-inst.eecs.berkeley.edu/~cs152/fa97>**
- **Video tapes of lectures available for viewing
in 205 McLaughlin**
- **Prerequisite quiz Friday September 5: CS 61C, CS 150**
- **Review Chapters 1-4, 7.1-7.2 Ap, B of COD:HSI 2nd Edition**
- **Turn in survey forms with photo**

Ideal (CS) versus Reality (EE)

- When input 0 \rightarrow 1, output 1 \rightarrow 0 but NOT instantly
 - Output goes 1 \rightarrow 0: output voltage goes from Vdd (5v) to 0v
- When input 1 \rightarrow 0, output 0 \rightarrow 1 but NOT instantly
 - Output goes 0 \rightarrow 1: output voltage goes from 0v to Vdd (5v)
- Voltage does not like to change instantaneously

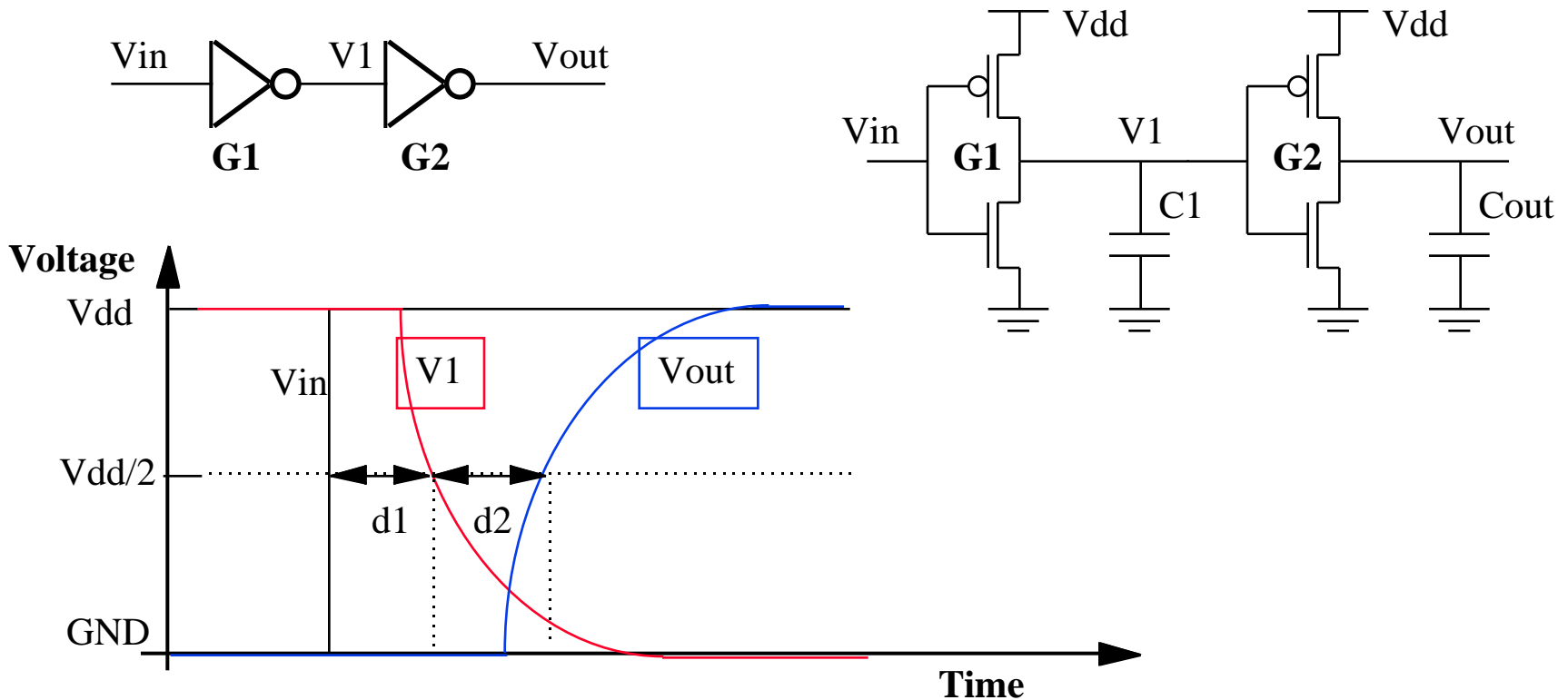


Fluid Timing Model



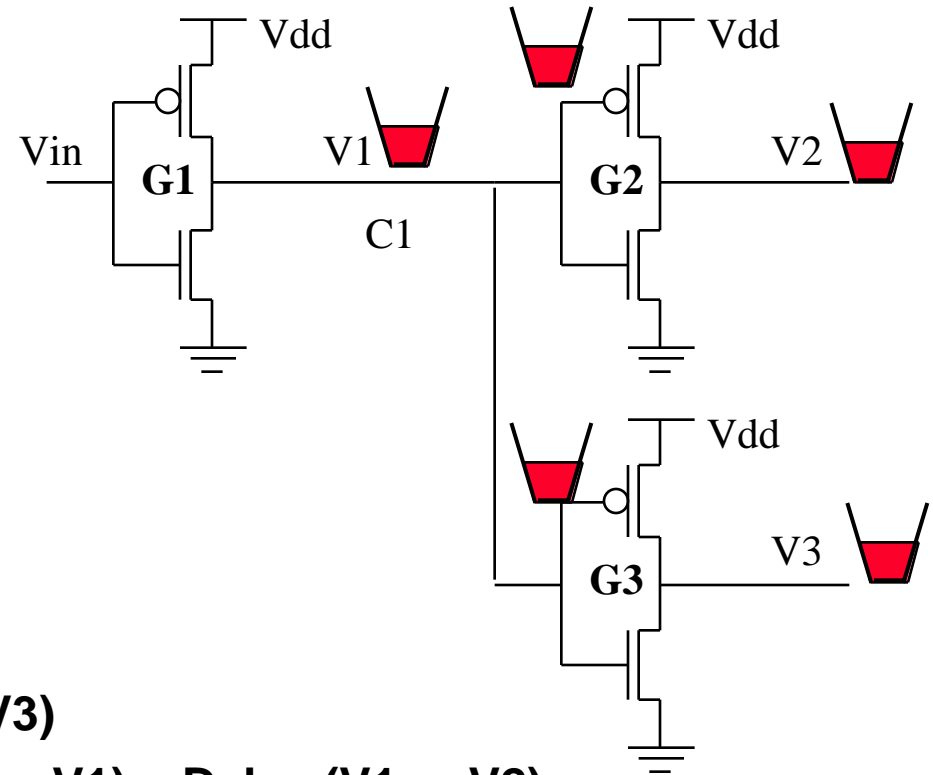
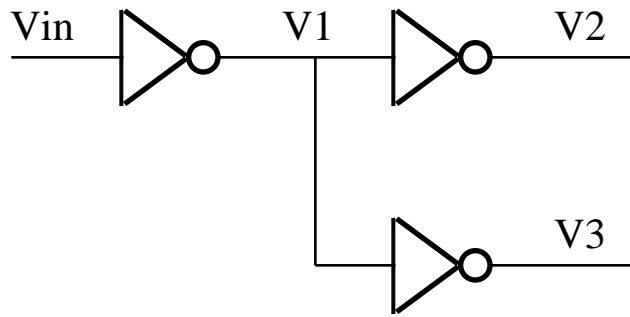
- Water \leftrightarrow Electrical Charge Tank Capacity \leftrightarrow Capacitance (C)
- Water Level \leftrightarrow Voltage Water Flow \leftrightarrow Charge Flowing (Current)
- Size of Pipes \leftrightarrow Strength of Transistors (G)
- Time to fill up the tank $\sim C / G$

Series Connection



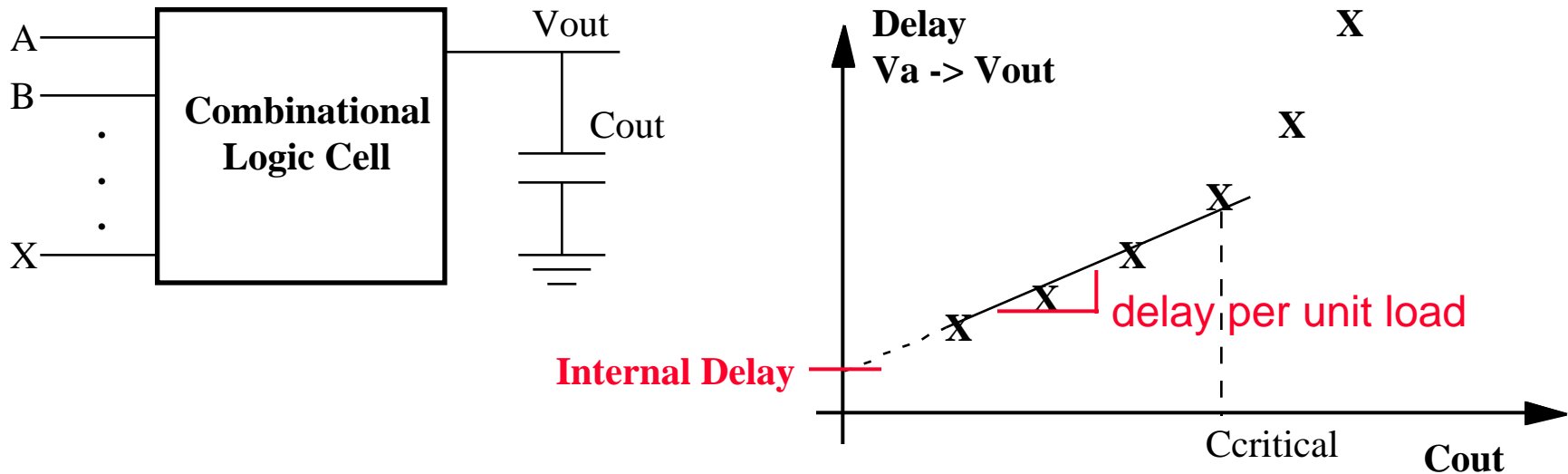
- **Total Propagation Delay = Sum of individual delays = $d1 + d2$**
- **Capacitance $C1$ has two components:**
 - **Capacitance of the wire connecting the two gates**
 - **Input capacitance of the second inverter**

Review: Calculating Delays



- Sum delays along serial paths
- Delay ($V_{in} \rightarrow V_2$) \neq Delay ($V_{in} \rightarrow V_3$)
 - Delay ($V_{in} \rightarrow V_2$) = Delay ($V_{in} \rightarrow V_1$) + Delay ($V_1 \rightarrow V_2$)
 - Delay ($V_{in} \rightarrow V_3$) = Delay ($V_{in} \rightarrow V_1$) + Delay ($V_1 \rightarrow V_3$)
- Critical Path = The longest among the N parallel paths
- C_1 = Wire C + C_{in} of Gate 2 + C_{in} of Gate 3

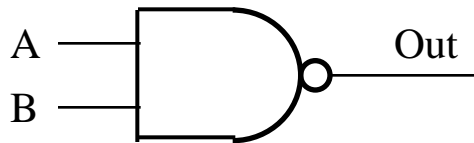
Review: General C/L Cell Delay Model



- **Combinational Cell (symbol) is fully specified by:**
 - functional (input -> output) behavior
 - truth-table, logic equation, VHDL
 - load factor of each input
 - critical propagation delay from each input to each output for each transition
 - $T_{HL}(A, o) = \text{Fixed Internal Delay} + \text{Load-dependent-delay} \times \text{load}$
- **Linear model composes**

Characterize a Gate

- Input capacitance for each input
- For each input-to-output path:
 - For each output transition type (H->L, L->H, H->Z, L->Z ... etc.)
 - Internal delay (ns)
 - Load dependent delay (ns / fF)
- Example: 2-input NAND Gate

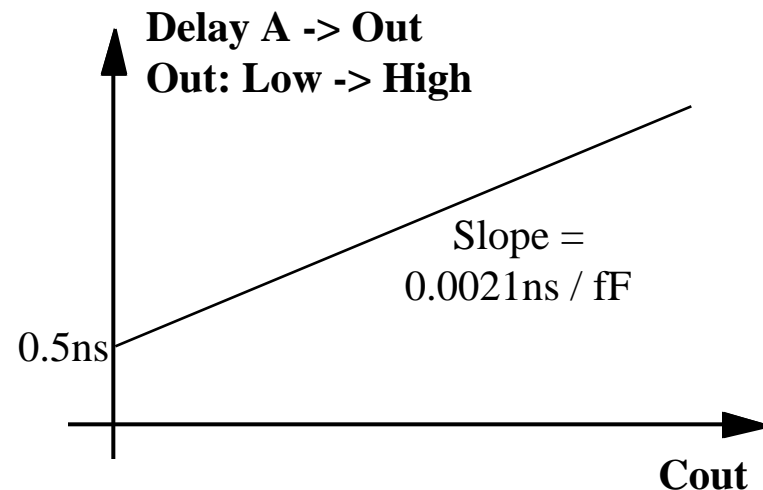


For A and B: Input Load = 61 fF

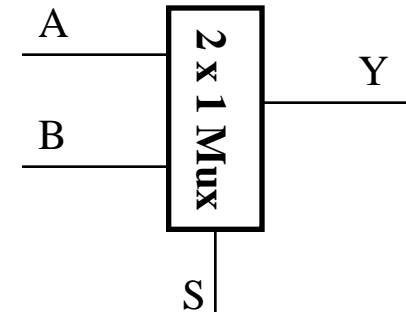
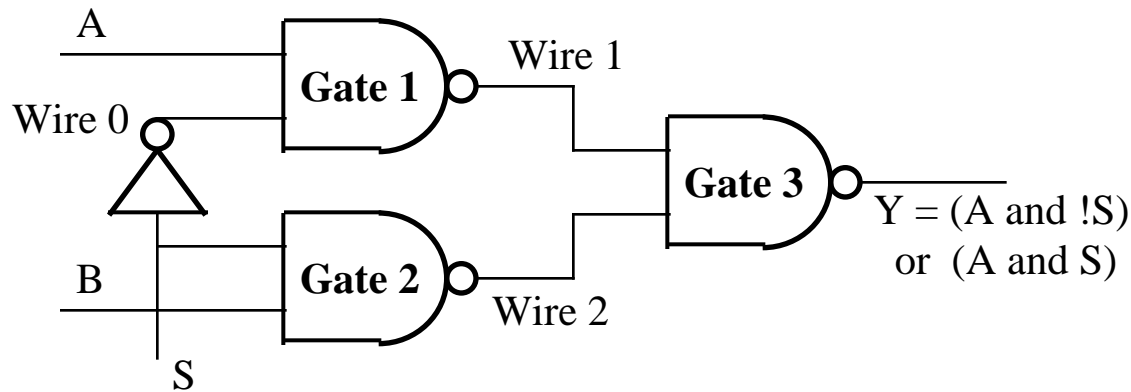
For either A -> Out or B -> Out:

$$T_{Plh} = 0.5\text{ns} \quad T_{plhf} = 0.0021\text{ns} / \text{fF}$$

$$T_{Phl} = 0.1\text{ns} \quad T_{Phlf} = 0.0020\text{ns} / \text{fF}$$



A Specific Example: 2 to 1 MUX



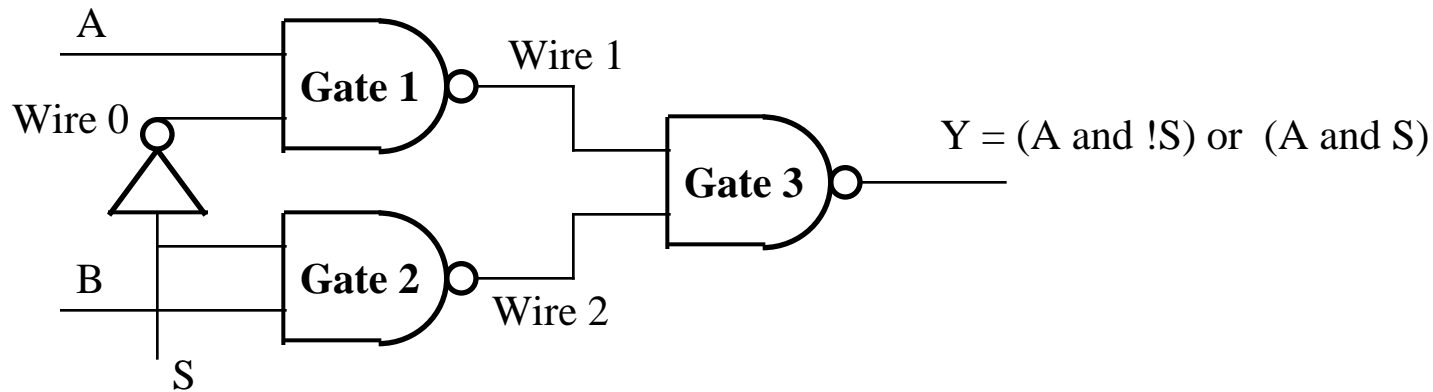
◦ Input Load (I.L.)

- A, B: I.L. (NAND) = 61 fF
- S: I.L. (INV) + I.L. (NAND) = 50 fF + 61 fF = 111 fF

◦ Load Dependent Delay (L.D.D.): Same as Gate 3

- $T_{AY|hf} = 0.021 \text{ ns} / \text{fF}$ $T_{AY|hf} = 0.020 \text{ ns} / \text{fF}$
- $T_{BY|hf} = 0.021 \text{ ns} / \text{fF}$ $T_{BY|hf} = 0.020 \text{ ns} / \text{fF}$
- $T_{SY|hf} = 0.021 \text{ ns} / \text{fF}$ $T_{SY|hf} = 0.020 \text{ ns} / \text{fF}$

2 to 1 MUX: Internal Delay Calculation



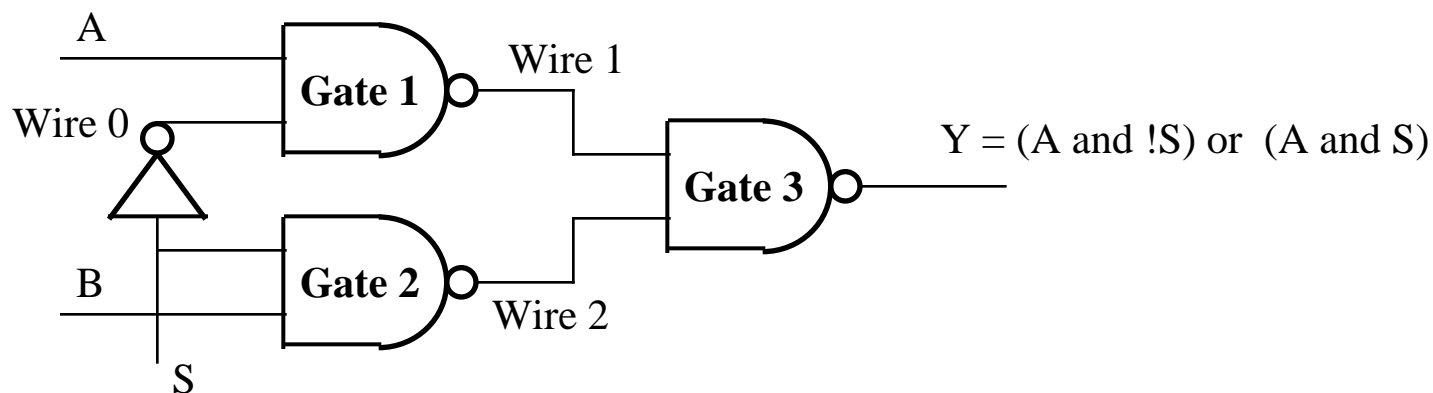
◦ Internal Delay (I.D.):

- 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 (Worst Case) : I.D. Inv + (Wire 0 C + G1 Input C) * L.D.D. Inv + Internal Delay A to Y

◦ We can approximate the effect of “Wire 1 C” by:

- Assume Wire 1 has the same C as all the gate C attache to it.
- Total C Gate 1 need to drive: 2.0 x Input C of Gate 3

2 to 1 MUX: Internal Delay Calculation (continue)



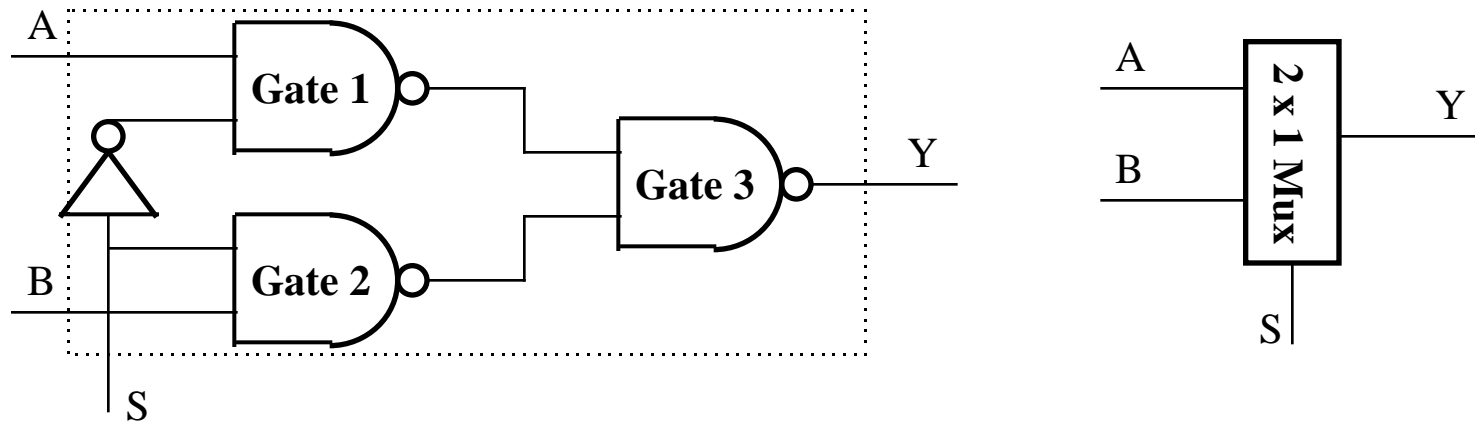
◦ Internal Delay (I.D.):

- 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 (Worst Case): I.D. Inv + (Wire 0 C + G1 Input C) * L.D.D. Inv + Internal Delay A to Y

◦ Specific Example:

- $T_{AY|H} = T_{Phl} G1 + (2.0 * 61 \text{ fF}) * T_{Phlf} G1 + T_{Plh} G3$
 $= 0.1 \text{ ns} + 122 \text{ fF} * 0.0020 \text{ ns/fF} + 0.5 \text{ ns} = 0.844 \text{ ns}$

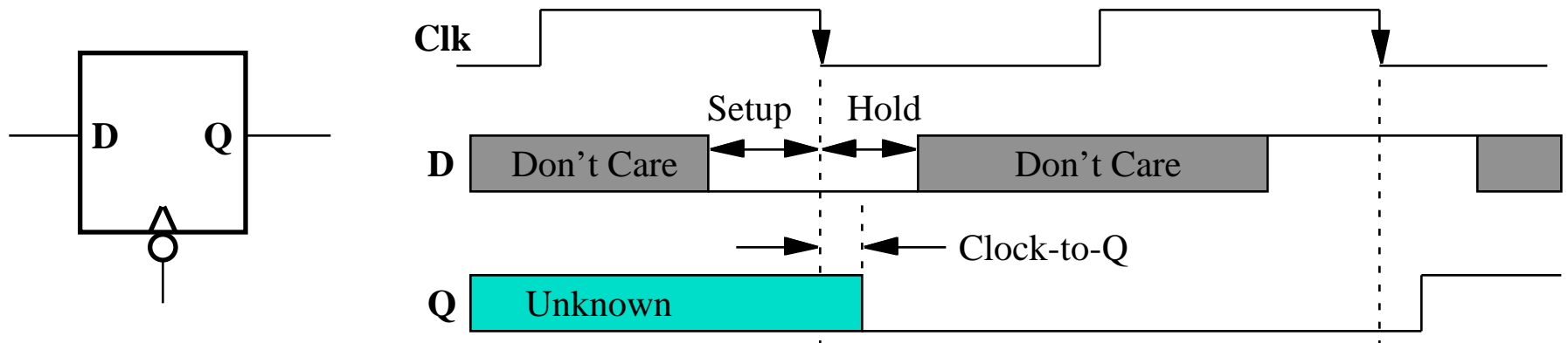
Abstraction: 2 to 1 MUX



- **Input Load: A = 61 fF, B = 61 fF, S = 111 fF**
- **Load Dependent Delay:**
 - $T_{AY|hf} = 0.021 \text{ ns / fF}$ $T_{AY|hlf} = 0.020 \text{ ns / fF}$
 - $T_{BY|hf} = 0.021 \text{ ns / fF}$ $T_{BY|hlf} = 0.020 \text{ ns / fF}$
 - $T_{SY|hf} = 0.021 \text{ ns / fF}$ $T_{SY|hlf} = 0.020 \text{ ns / fF}$
- **Internal Delay:**
 - $T_{AY|h} = T_{Phl} G1 + (2.0 * 61 \text{ fF}) * T_{Phlf} G1 + T_{Plh} G3$
 $= 0.1\text{ns} + 122 \text{ fF} * 0.0020\text{ns/fF} + 0.5\text{ns} = 0.844\text{ns}$
 - **Fun Exercises: $T_{AY|h}$, $T_{BY|h}$, $T_{SY|h}$, $T_{SY|h}$**

Break (5 Minutes)

Storage Element's Timing Model



- **Setup Time:** Input must be stable **BEFORE** the trigger clock edge
- **Hold Time:** Input must **REMAIN** stable after the trigger clock edge
- **Clock-to-Q time:**
 - Output cannot change instantaneously at the trigger clock edge
 - Similar to delay in logic gates, two components:
 - Internal Clock-to-Q
 - Load dependent Clock-to-Q

CS152 Logic Elements

- **NAND2, NAND3, NAND 4**
- **NOR2, NOR3, NOR4**
- **INV1x (normal inverter)**
- **INV4x (inverter with large output drive)**

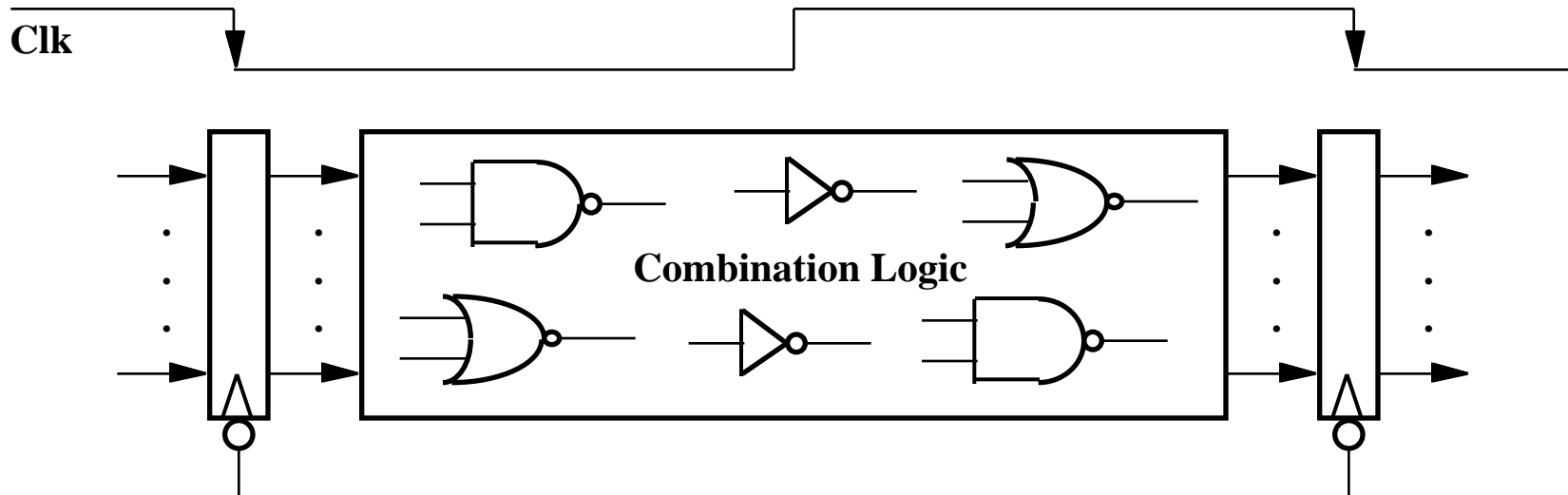
CS152 Logic Elements (Continue)

- **XOR2**
- **XNOR2**
- **PWR: Source of 1's**
- **GND: Source of 0's**
- **fast MUXes (maybe)**

CS152 Storage Element

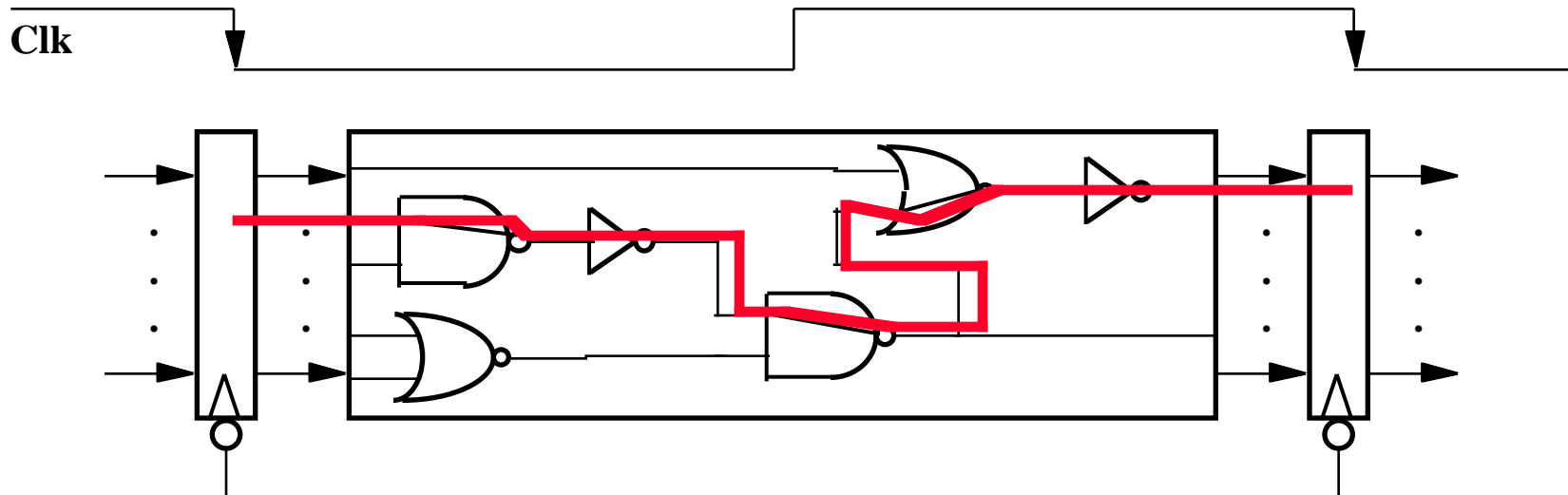
- D flip flop with negative edge triggered

Clocking Methodology



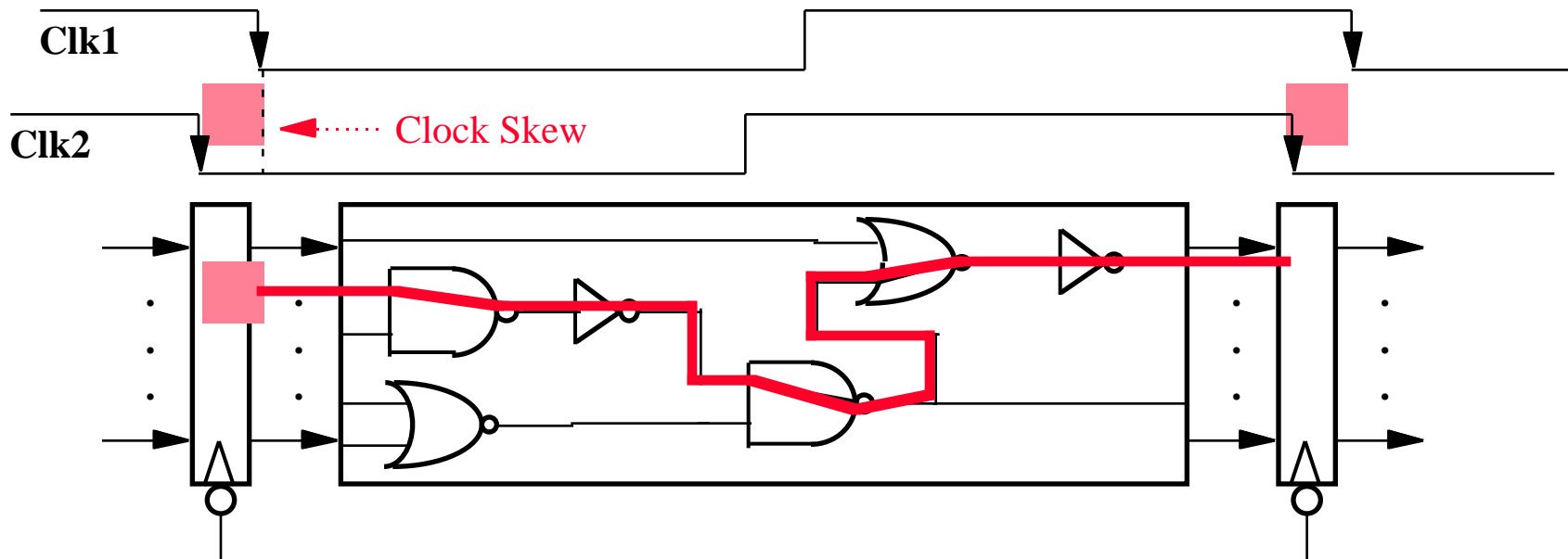
- All storage elements are clocked by the same clock edge
- The combination logic block's:
 - Inputs are updated at each clock tick
 - All outputs **MUST** be stable before the next clock tick

Critical Path & Cycle Time



- **Critical path: the slowest path between any two storage devices**
- **Cycle time is a function of the critical path**
- **must be greater than:**
 - **Clock-to-Q + Longest Path through the Combination Logic + Setup**

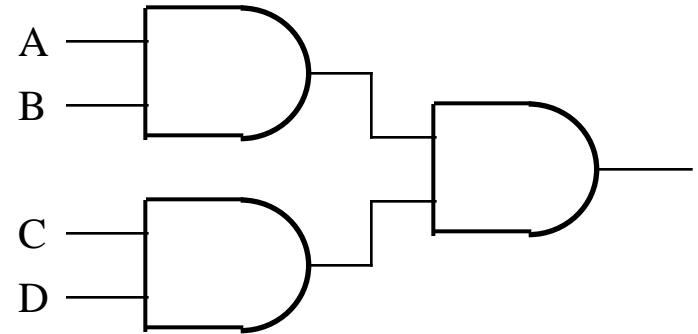
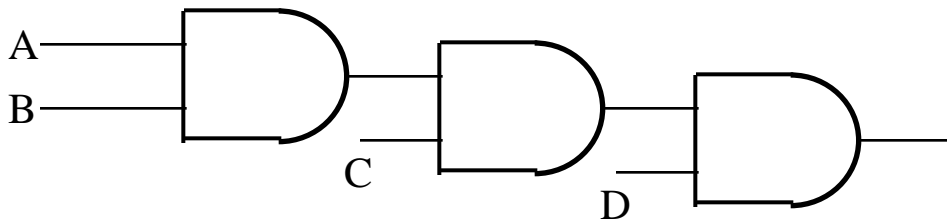
Clock Skew's Effect on Cycle Time



- The worst case scenario for cycle time consideration:
 - The input register sees CLK1
 - The output register sees CLK2
- **Cycle Time \geq CLK-to-Q + Longest Delay + Setup + Clock Skew**

Tricks to Reduce Cycle Time

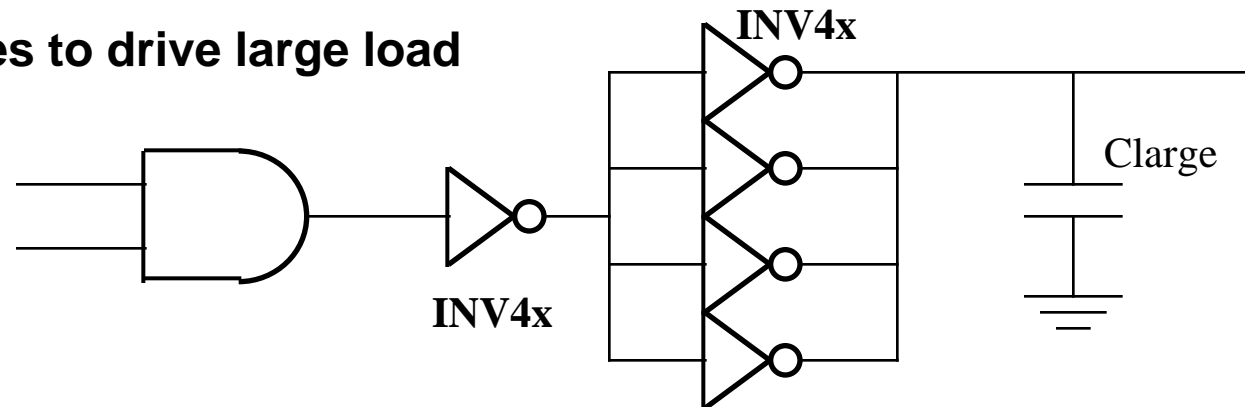
- Reduce the number of gate levels



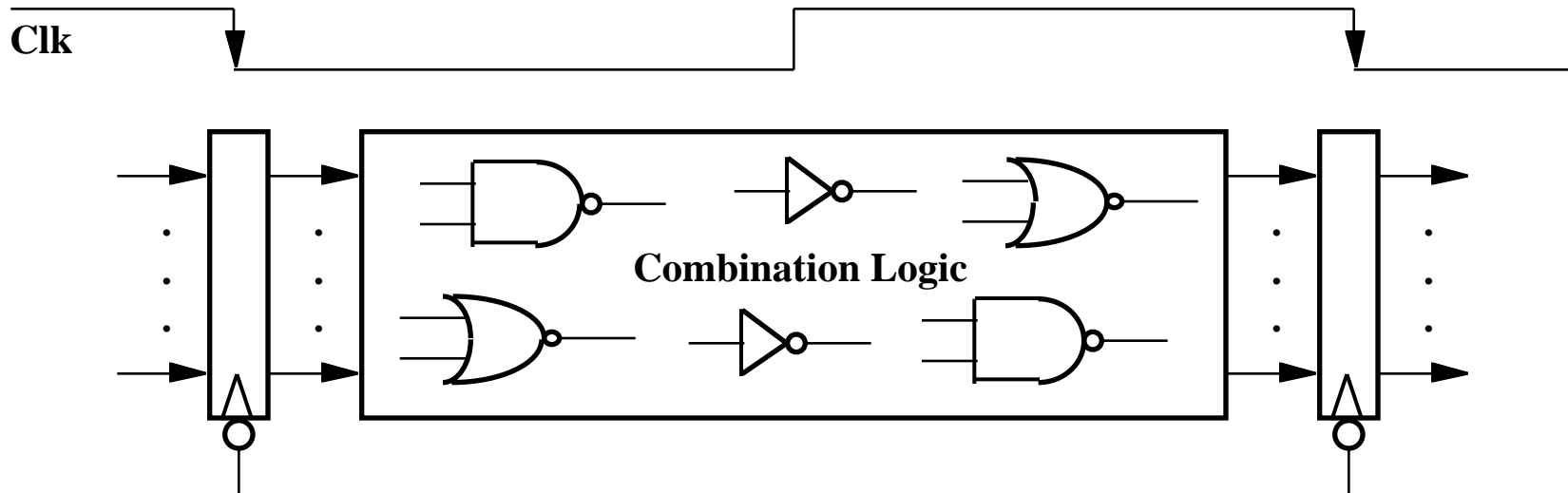
- Pay attention to loading

- One gate driving many gates is a bad idea
- Avoid using a small gate to drive a long wire

- Use multiple stages to drive large load

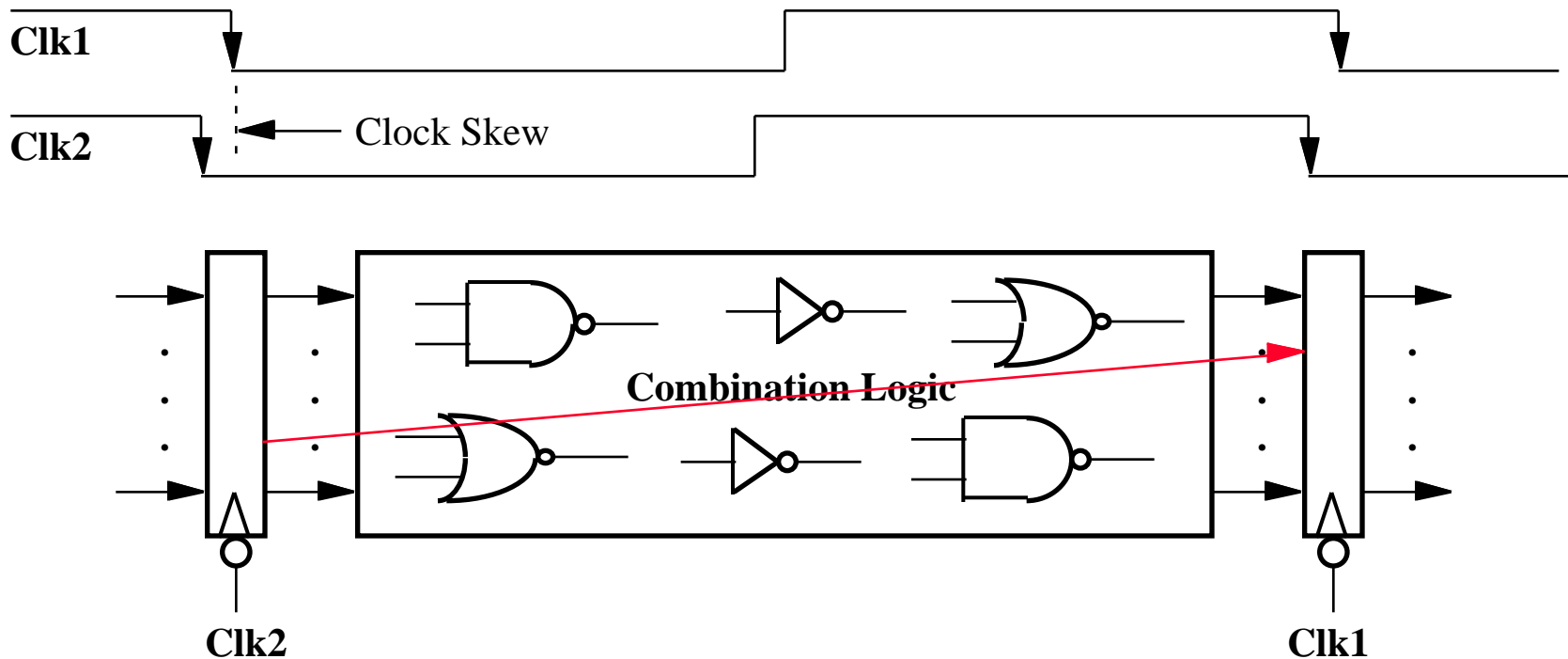


How to Avoid Hold Time Violation?



- **Hold time requirement:**
 - **Input to register must NOT change immediately after the clock tick**
- **This is usually easy to meet in the “edge trigger” clocking scheme**
- **Hold time of most FFs is ≤ 0 ns**
- **CLK-to-Q + Shortest Delay Path must be greater than Hold Time**

Clock Skew's Effect on Hold Time



- The worst case scenario for hold time consideration:
 - The input register sees CLK2
 - The output register sees CLK1
 - fast FF2 output must not change input to FF1 for same clock edge
- $(\text{CLK-to-Q} + \text{Shortest Delay Path} - \text{Clock Skew}) > \text{Hold Time}$

Summary

- **Performance and Technology Trends**
 - **Keep the design simple to take advantage of the latest technology**
 - **CMOS inverter and CMOS logic gates**
- **Delay Modeling and Gate Characterization**
 - **Delay = Internal Delay + (Load Dependent Delay x Output Load)**
- **Clocking Methodology and Timing Considerations**
 - **Simplest clocking methodology**
 - **All storage elements use the SAME clock edge**
 - **Cycle Time = CLK-to-Q + Longest Delay Path + Setup + Clock Skew**
 - **(CLK-to-Q + Shortest Delay Path - Clock Skew) > Hold Time**

To Get More Information

- **A Classic Book that Started it All:**
 - **Carver Mead and Lynn Conway, “Introduction to VLSI Systems,” Addison-Wesley Publishing Company, October 1980.**
- **A Good VLSI Circuit Design Book**
 - **Lance Glasser & Daniel Dobberpuhl, “The Design and Analysis of VLSI Circuits,” Addison-Wesley Publishing Company, 1985.**
 - **Mr. Dobberpuhl is responsible for the DEC Alpha chip design.**
- **A Book on How and Why Digital ICs Work:**
 - **David Hodges & Horace Jackson, “Analysis and Design of Digital Integrated Circuits,” McGraw-Hill Book Company, 1983.**