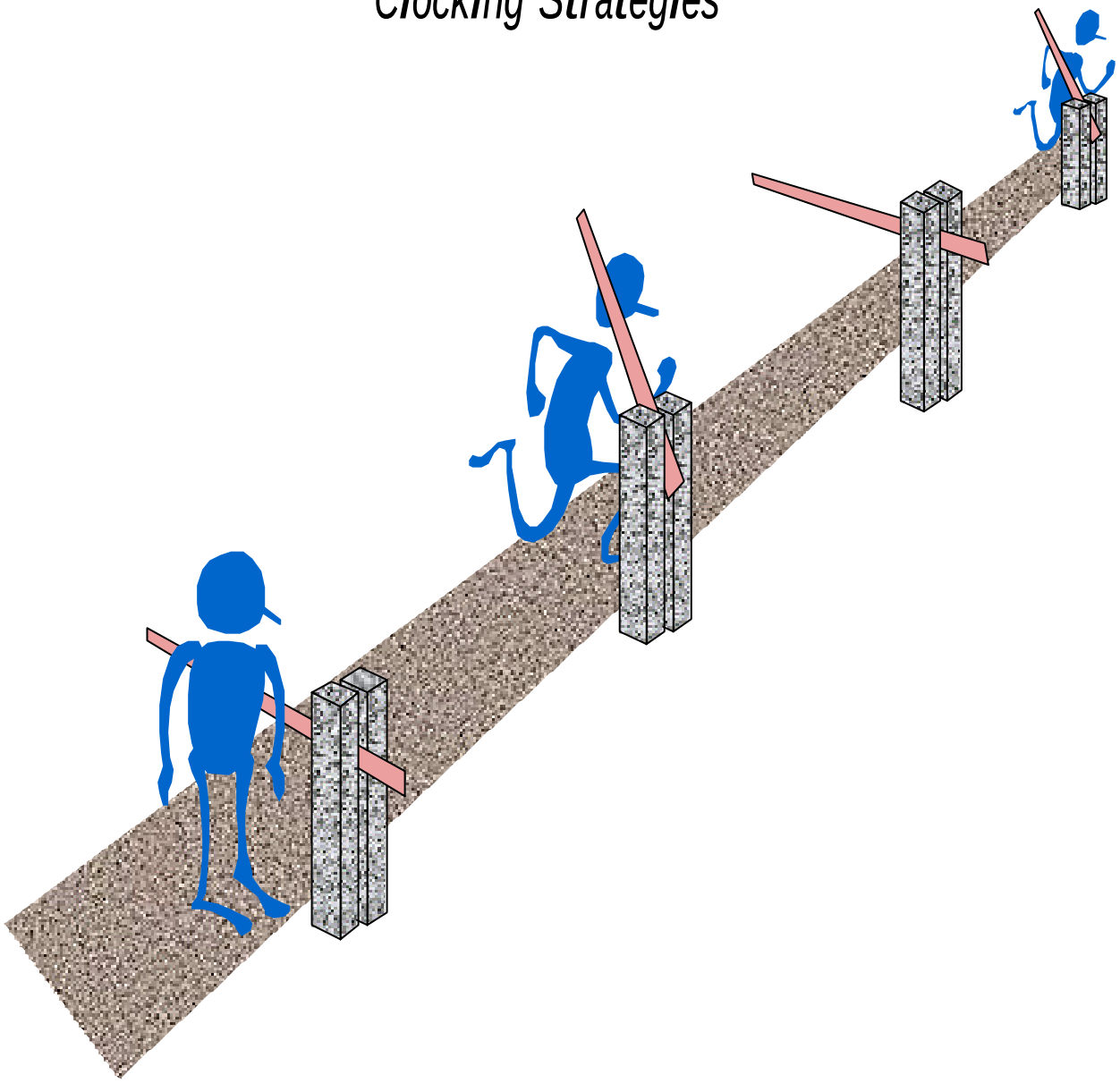
VLSI Design I

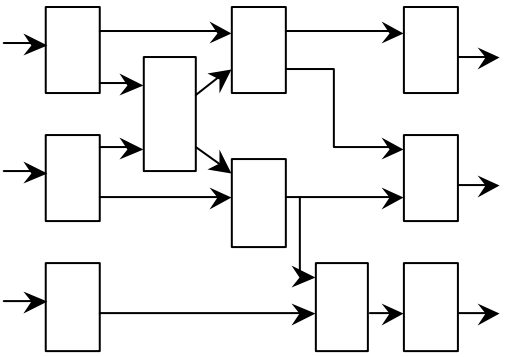
CMOS Sequential Logic
Clocking Strategies

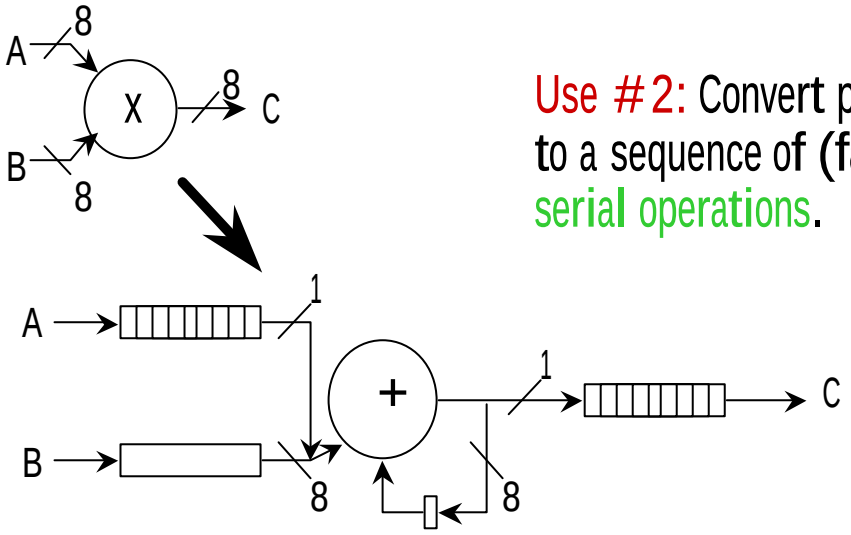


Today's handouts:
(1) Lecture Slides

Sequential Logic

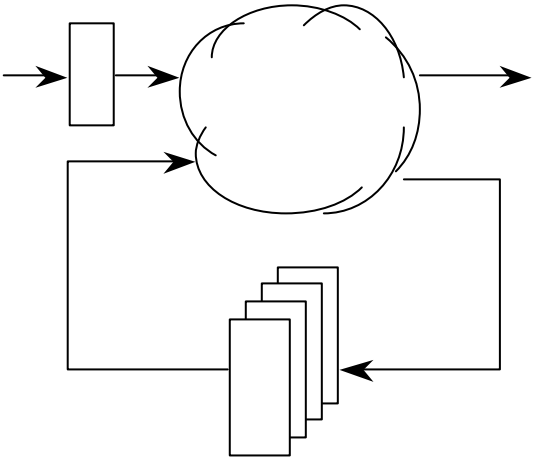
Use #1: Get better utilization from idle combinational logic blocks. *Pipeline* the system so that new computations start before the old ones complete. Add registers to keep computations separate.



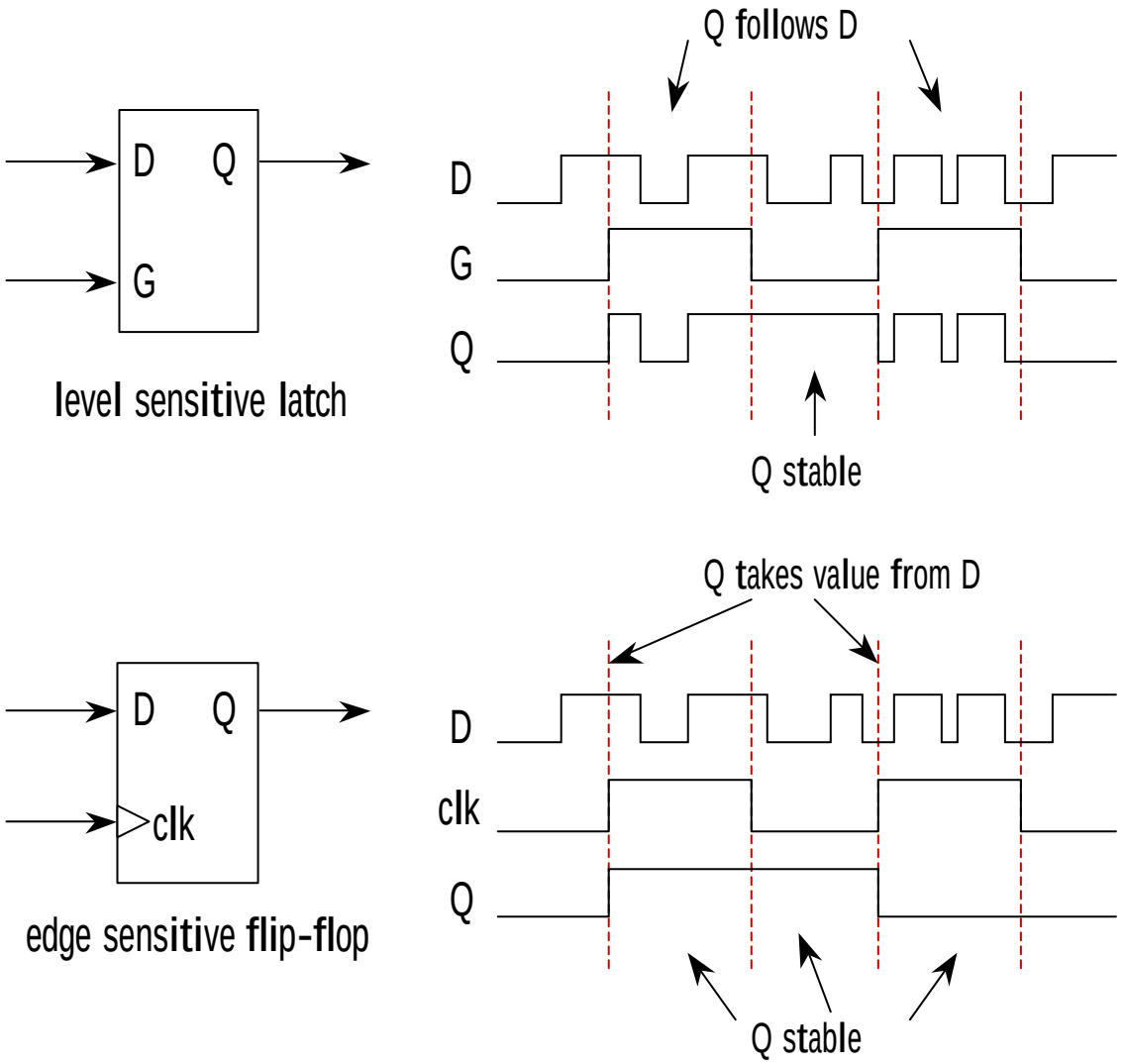


Use #2: Convert parallel operations to a sequence of (faster, smaller) serial operations.

Use #3: Need to process a sequence of inputs and want to reuse the same hardware (finit state machine).



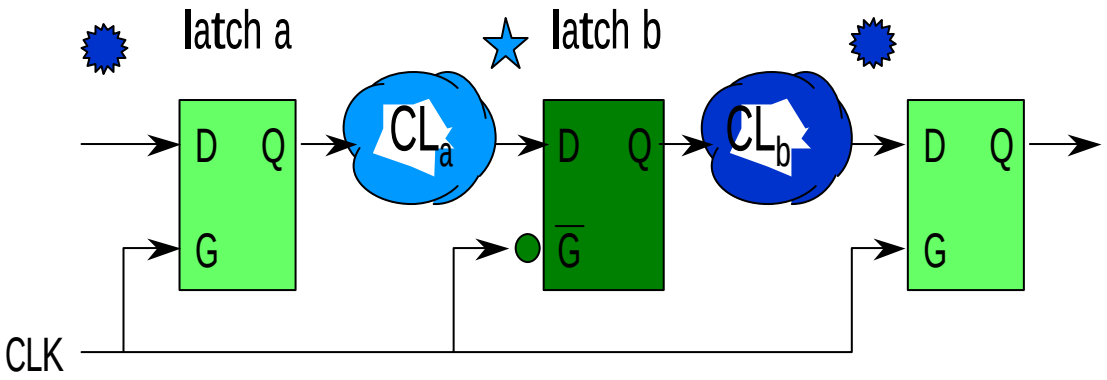
Latches and Flip-Flops

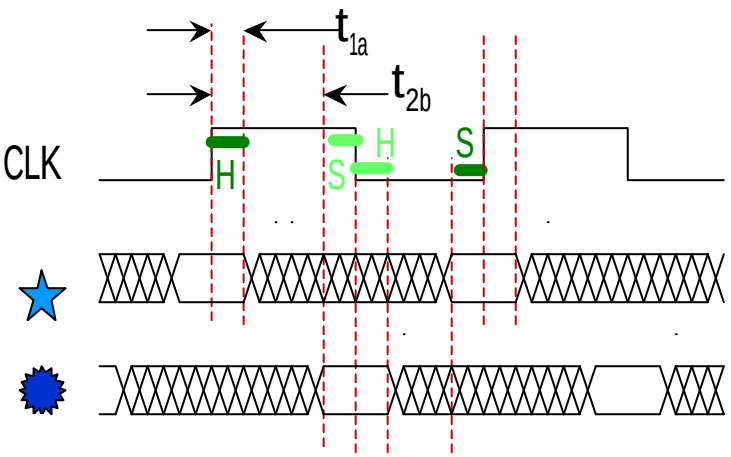


A *static* latch will hold data while G is inactive, however long that may be. A *dynamic* latch will hold data while G is inactive, but only "for a while", after which the saved value may decay.

Do static latches dissipate static power?
How long is "for a while"?
Which one should I use?

Latch Timing Constraints #1





Do I have to check ALL these constraints?

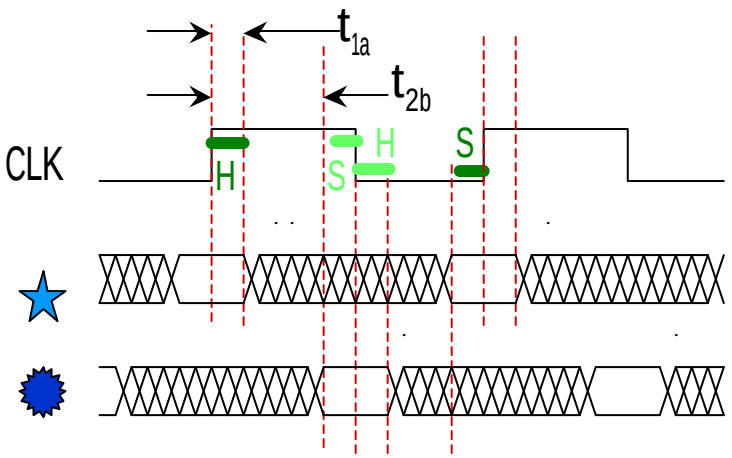
$$t_{1a} = t_{mqa} + t_{mda} > t_{hb}$$
$$t_{1b} = t_{mqb} + t_{mdb} > t_{ha}$$
$$t_{2a} = t_{qa} + t_{da} < t_{c0} - t_{sb}$$
$$t_{2b} = t_{qb} + t_{db} < t_{c1} - t_{sa}$$

t_h = hold time
t_s = setup time
t_m = min delay from invalid input to invalid output

t_d = max delay from valid input to valid output for comb. logic
t_q = max delay from G to Q

t_{c0} = low periode of clock cycle t_c

Latch Timing Constraints #2



$$t_{1a} = t_{mqa} + t_{mda} > t_{hb}$$
$$t_{1b} = t_{mqb} + t_{mdb} > t_{ha}$$
$$t_{2a} = t_{qa} + t_{da} < t_{c0} - t_{sb}$$
$$t_{2b} = t_{qb} + t_{db} < t_{c1} - t_{sa}$$

Questions for latch-based designs:

- how much time for useful work (i.e. for combinational logic delay)?

$$t_{da} + t_{db} < t_c - 2(t_s + t_q)$$

- what is the maximal clock frequency

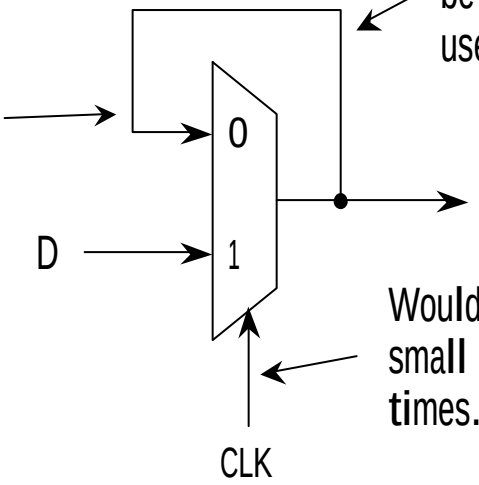
- does it help to guarantee a minimum t_m, for example, by requiring a minimum number of gates in each cloud?
- Suppose the maximum clock skew is t_{SKEW}. How does that affect the equations above? Clock skew measures the difference in arrival of CLK at two cascaded latches (not necessarily any two latches!).

Static Latches

Basic idea:

Need gain around this loop to make latch static.

Want storage node to be isolated from whatever user does to Q.

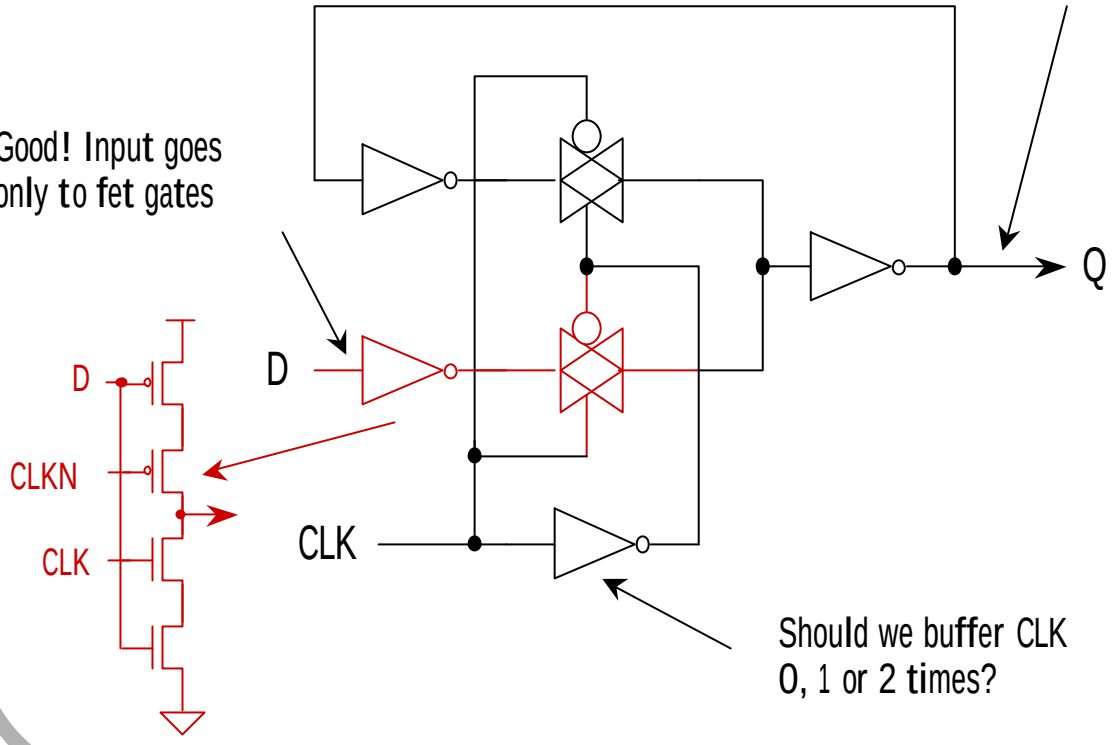


Would like fast CLK-to-Q, small setup and zero hold times.

Obvious implementation:

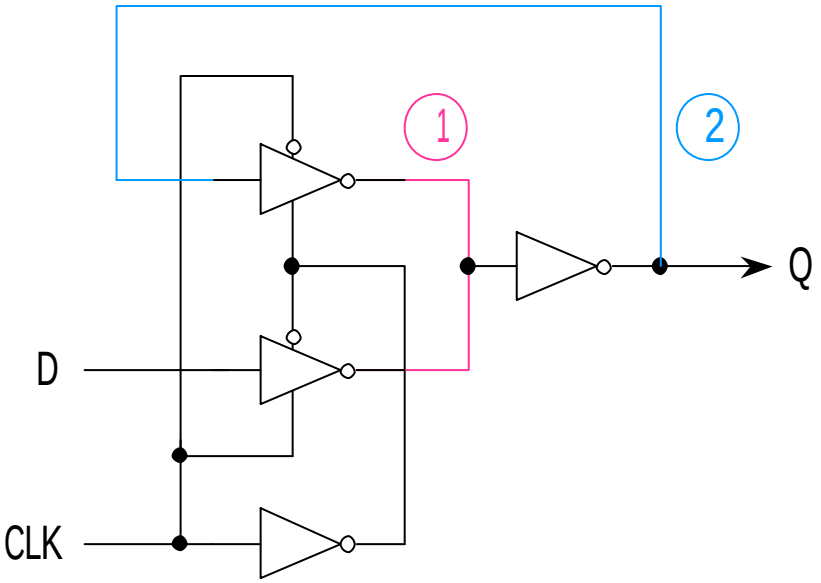
Oops… feedback not isolated from Q. Could add additional output inverters...

Good! Input goes only to fet gates



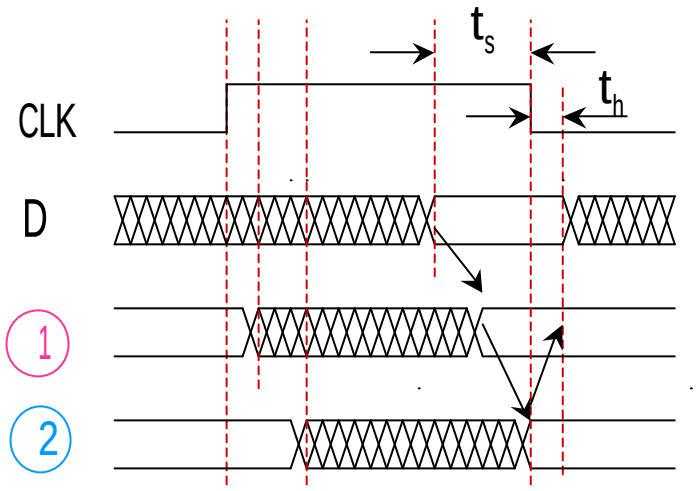
Should we buffer CLK 0, 1 or 2 times?

Latch Timing



setup time = how long D input has to be stable *before* CLK transition.

hold time = how long D input has to be stable *after* CLK transition.

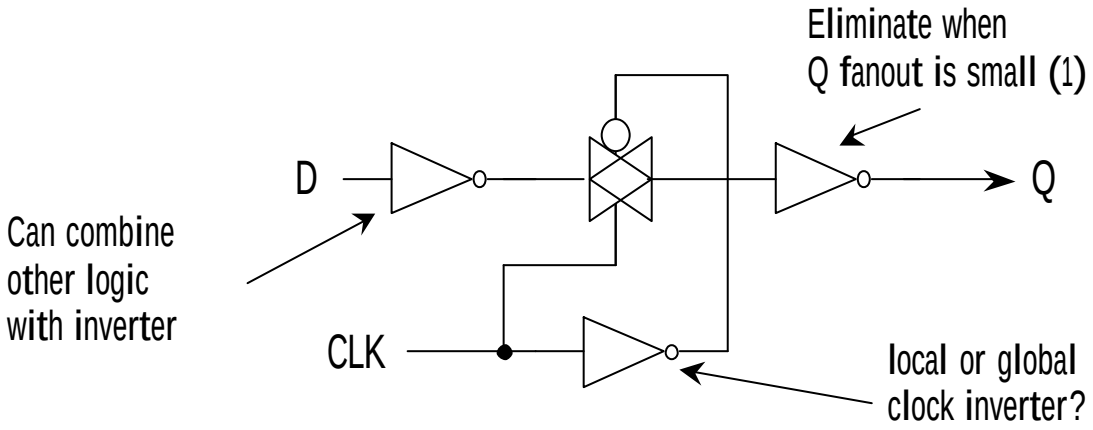


So, what node should we use to measure setup and hold times? And what should we measure?

Other time of interest: CLK-to-Q

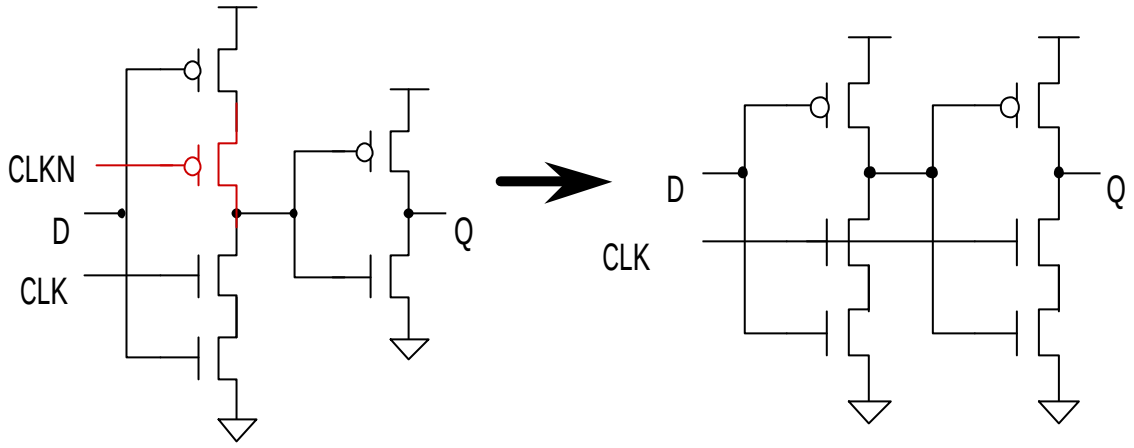
Dynamic Latches

Suppose in the interest of speed we were willing to give up the "static guarantee" and take our chances with dynamic latches, i.e., remove feedback path...



Can we do without the CLK inverter too?

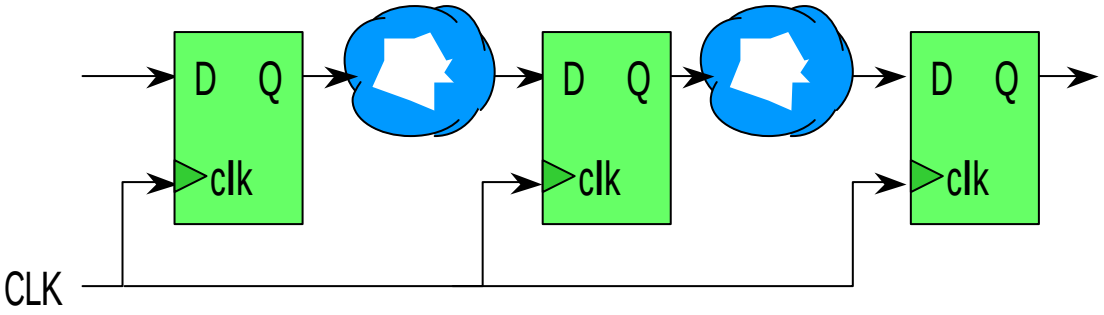
DEC did without on 21064 but put in back in for 21164



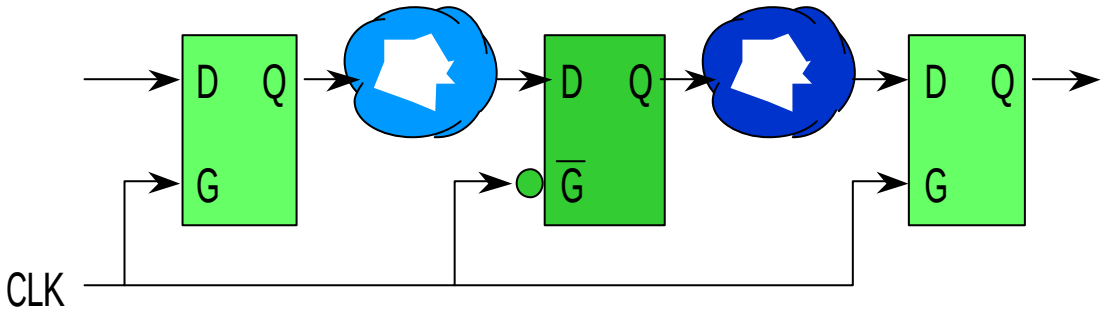
Delete the PFET driven by CLKN and then add NFET driven by CLK in Q's pulldown path to handle what happens when D goes from 1 to 0.

Single-Phase Clocked Systems

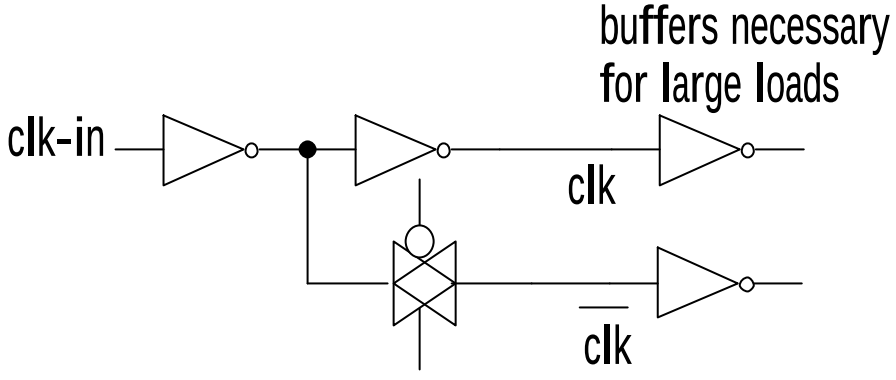
RTL #1:



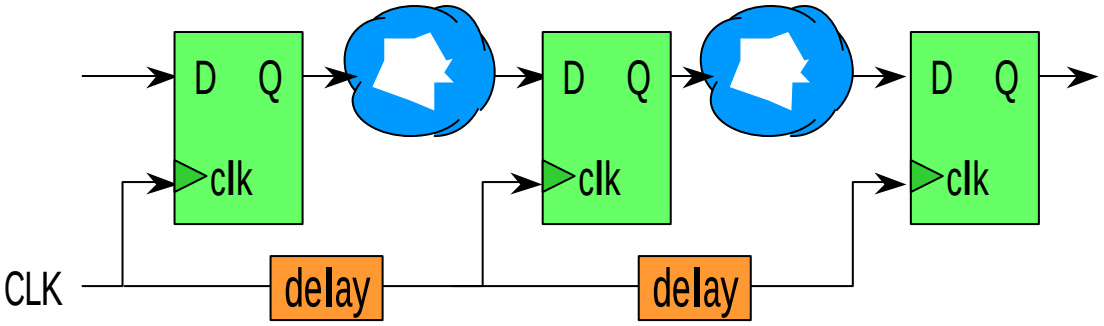
latch #2:



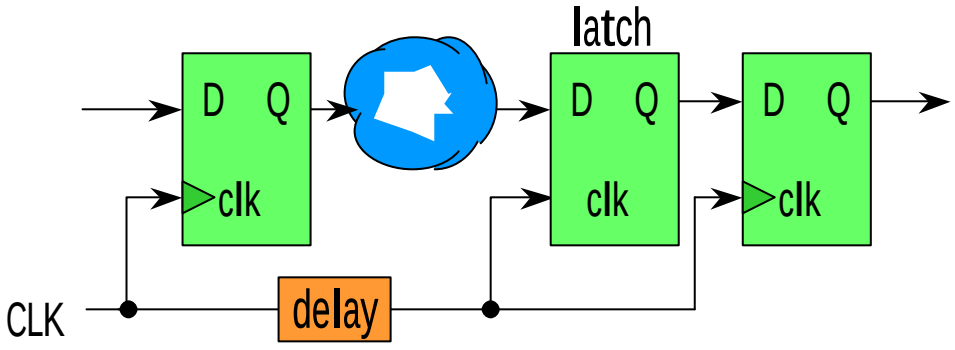
Simplest clocking methodology is to use a single clock in conjunction with a register. Clocks are generated with global clock buffers. CLK and $\overline{CLK}$ are generated locally.

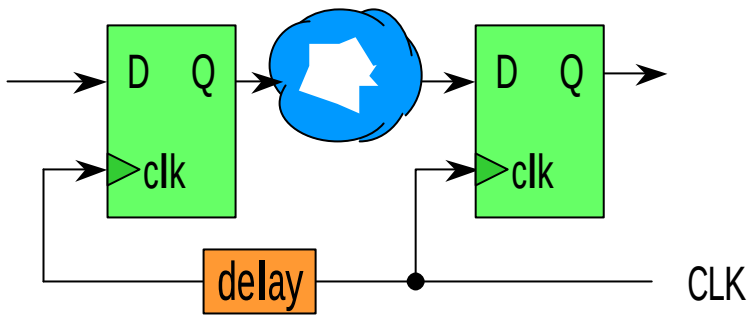


Clock Skew

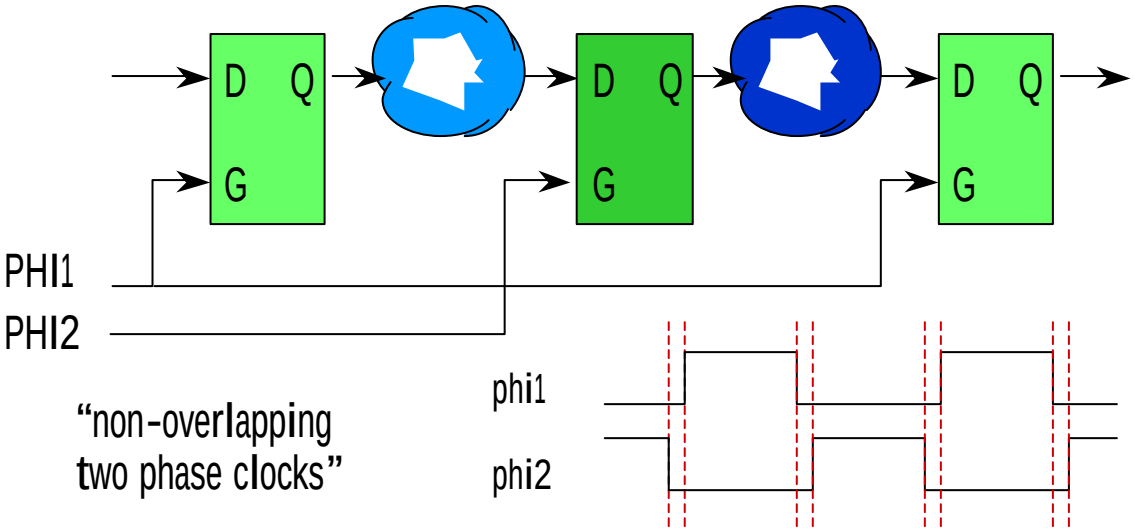


- if a clock net is heavily loaded, there might be a race between clock and data -> clock skew
- special attention has be made by designing the clock tree. CAD tools are able to design balanced clock trees.
- two methods to avoid clock skew:

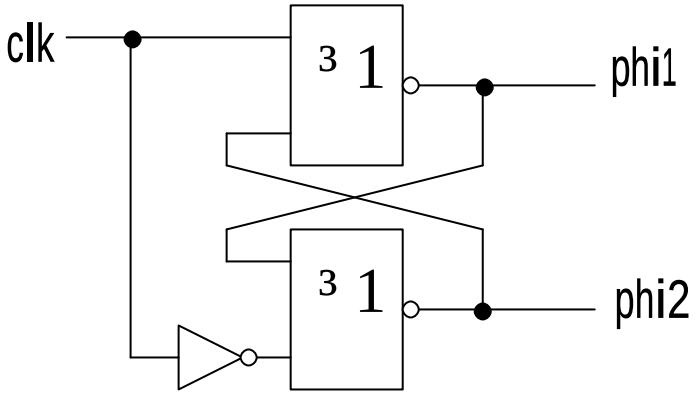




Two-Phase Clocked Systems



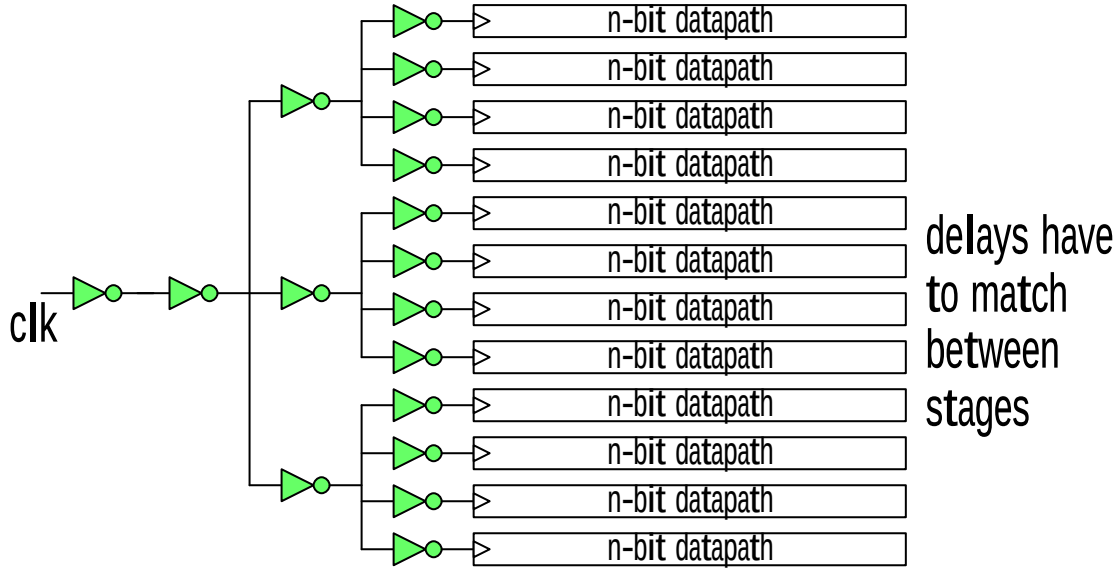
- a problem in singlem phase clocked systems is the generation ad distribution of nearly perfect overlapping clocks.
- in two-phase clocked systems this is solved by non-overlapping clocks
- non-overlapping clocks can be generated with latch structures



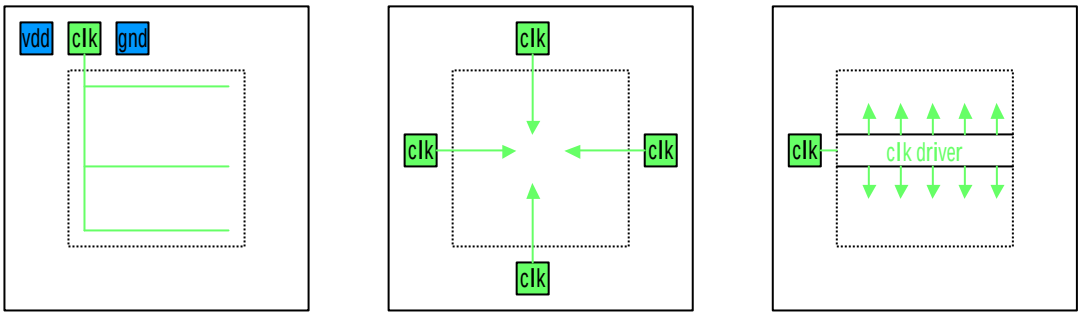
Clock Distribution

Two main techniques for clock distribution exist:

- a single large buffer (see Alpha processor)
- a distributed clock tree approach



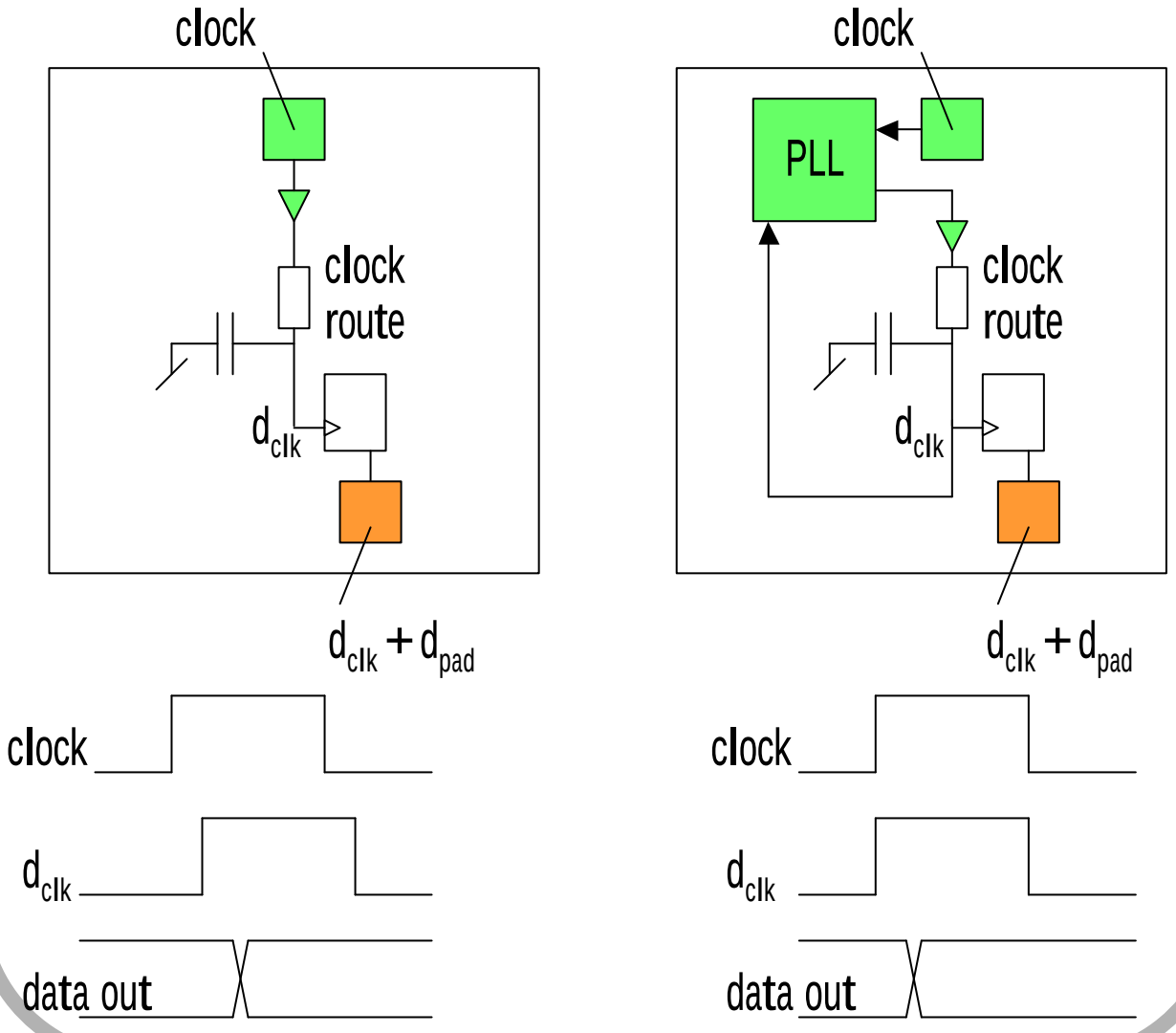
- there is no such thing as design-free clocking strategy in today's high-performance processes
- clock buffers should be surrounded by power pads due to its large power consumption



Phase Locked Loop Clock Technique

Phase locked loops (PLL) are used to generate internal clocks on chips for two main reasons:

- to synchronize the internal clock of a chip with an external clock
- to operate the internal clock at a higher rate than the external clock input

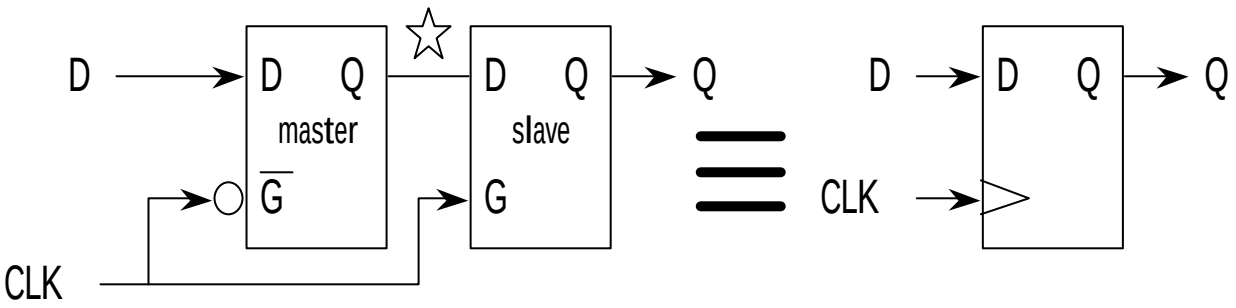


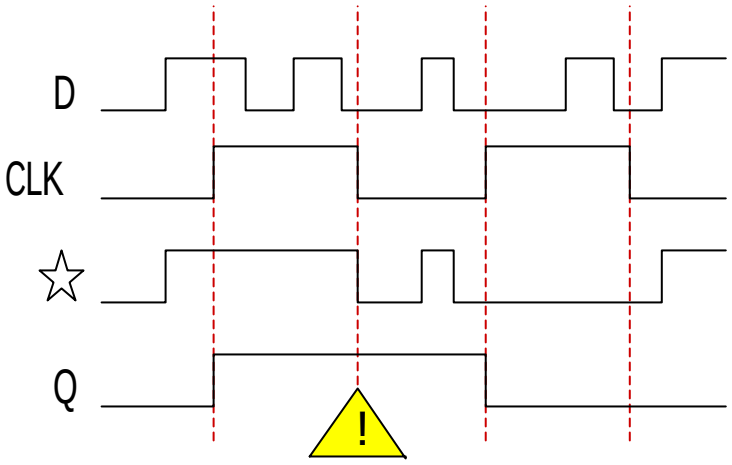
Flip-flops (registers)

Using alternating positive and negative dynamic latches with a single clock gives great speed and small area, but…

- lots of worries about clock skew
- must balance logic delays to minimize wastage
- need latch size checks (check optimizations!)

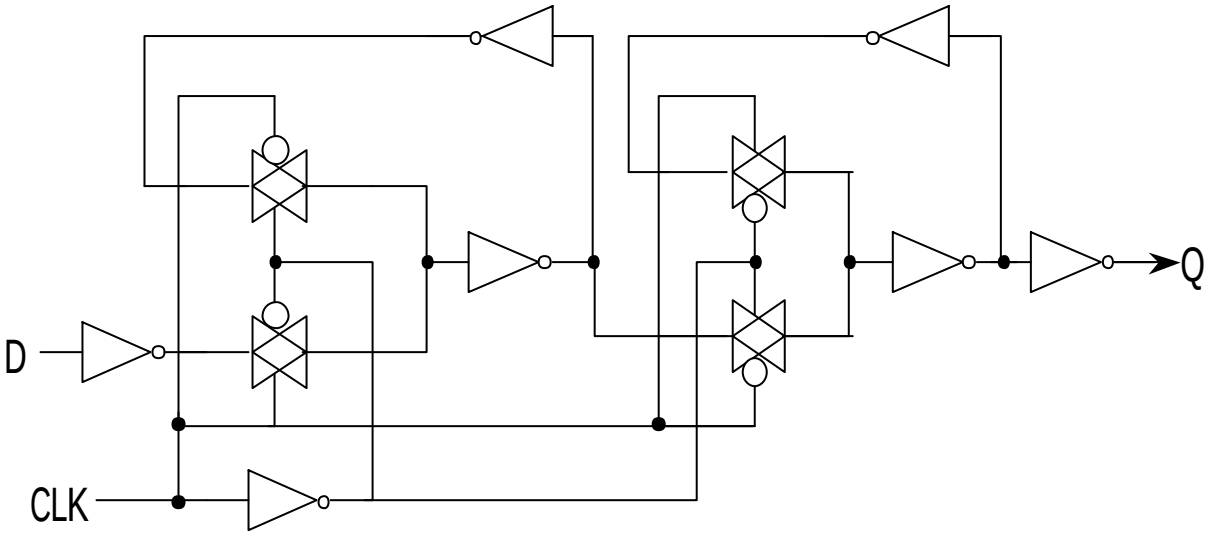
What about those of us who don't have buildings full of engineers to sweat the details? Use D-flip-flops and address all the problems once!



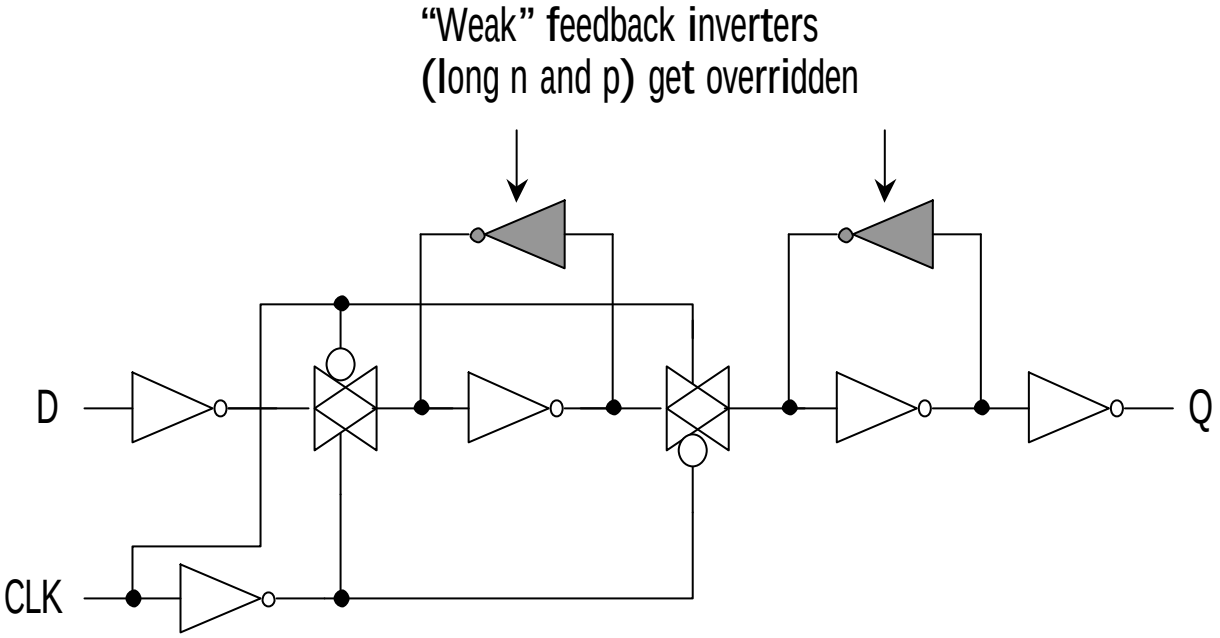


Flip-flop Implementations

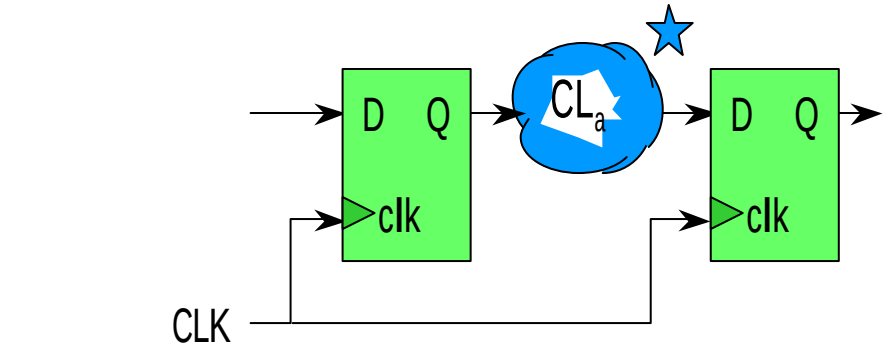
Obvious implementation:

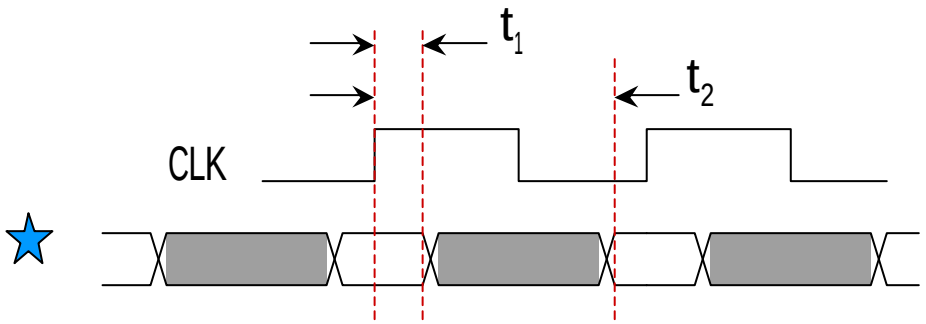


Use “jamb” latches to lighten CLK load:



Flip-Flop Timing





$$t_1 = t_{mq} + t_{ma} > t_h$$
$$t_2 = t_q + t_{da} < t_c - t_s$$

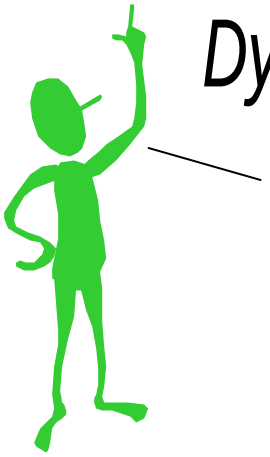
Questions for register-based designs:

- how much time for useful work (i.e. for combinational logic delay)?
- does it help to guarantee a minimum t_m? How about designing registers so that

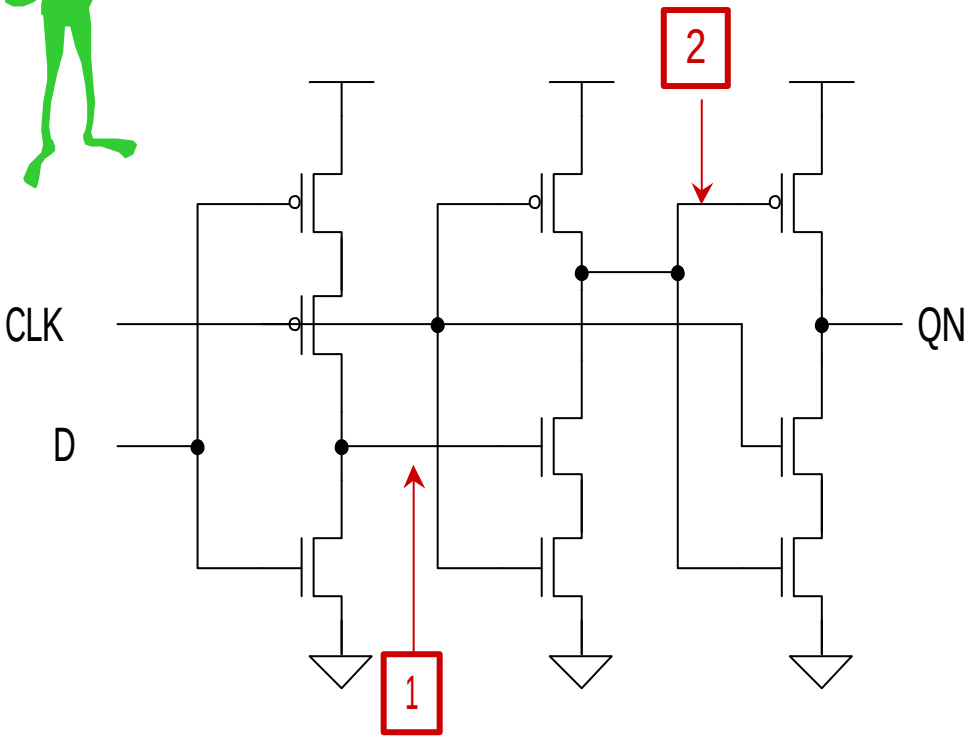
$$t_{mq} > t_h?$$

- Supose the maximum clock skew is t_{SKEW}. How does that affect the equations above?

Dynamic Flip-Flops



I'll have the Christer Svensson special please!



CLK is low:

- node 1 follows not(D)
- node 2 pulled up
- QN is "floating" with it's old value

CLK is high:

- node 2 = "0" if node 1 = "1",
  otherwise it stays "1"
  ⇨ node 2 = not(node 1) shortly after CLK↑
- QN = not(node 2) ⇨ stable soon after CLK↑
- node 1 can be pulled down if D goes to "0" (capacitive coupling), but node 2 won't change!

Static Timing Analysis

Do I have to check ALL the constraints?

Yup, for every pair of connected register/latches AND for all possible data values!

We need a CAD tool: *static timing analyzer.* Here's how it works:

Step 1: "Level-ize" all signal nodes.

Start by assigning all register outputs and top-level inputs a level of 0. For all other gates: $level_{OUTPUT} = max(level_{INPUT}) + 1$.

Step 2: Compute min/max signal delays.

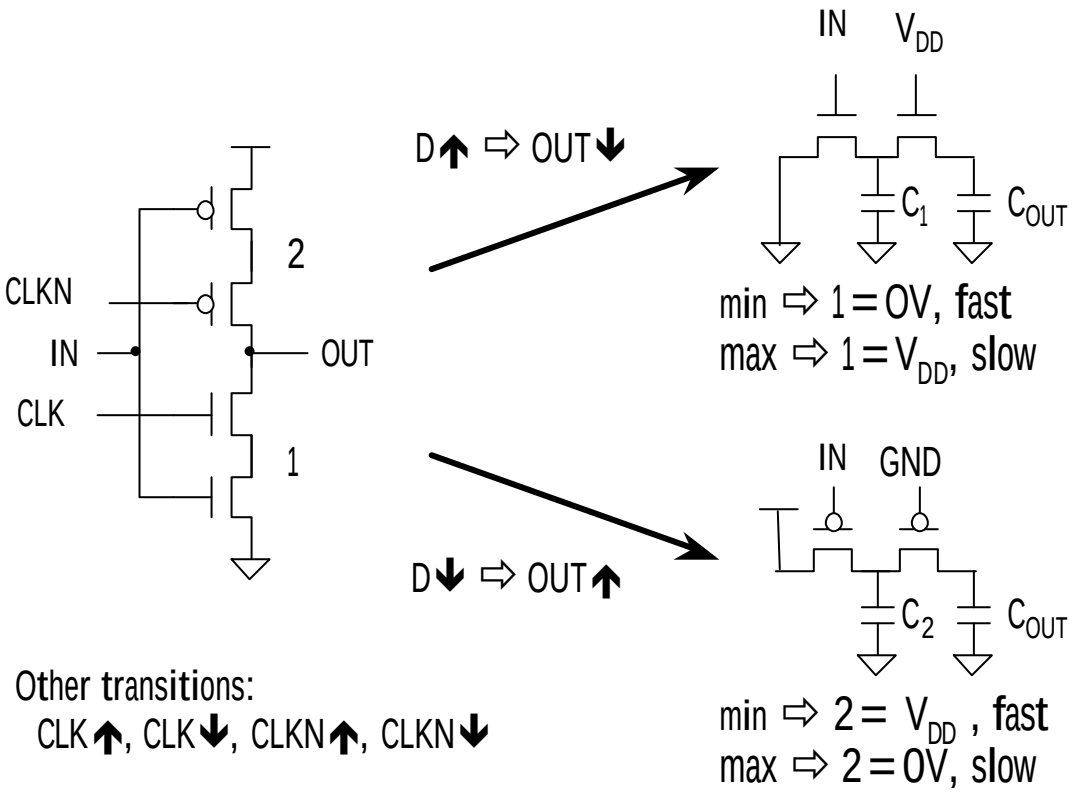
For each successive node level, compute min and max time for all nodes on that level (see next slide for details). This is a "data independent" computation. Might need case analysis to avoid false paths.

Step 3: Check setup and hold constraints

Use min times of register inputs to check hold time. Use max times and t_{CLK} to check setup time or use max time + t_{SETUP} to determine min t_{CLK}.

Stage Delay Computation

Look at each gate and use knowledge of input timing and rise/fall timing to compute earliest and latest time output could change for both rising and falling output transitions.



Use Penfield-Rubenstein model to compute $t_{d,in-out}$ = sum(R_i,C_i) over all nodes "i" in the stage, where R_i is total "effective resistance" to power rail and C_i is non-zero if node capacitor needs to be charged/discharged. Multiply by derating factor to account for rise/fall time of input.

Coming Up...

Next topic…

Finite state machines: state diagrams, state minimization, state assignment, logic and PLA implementations.

Readings for next time…

Weste:

- Sections 5.5 thru 5.5.6 (latch, FF)
- 5.5.8 thru 5.5.11 (clock strategy)
- 5.5.15 and 5.5.16 (clock strategy)

Selfstudy…

Weste:

- PLL section 9.3.5.3

Exercises: VLSI-10

Ex vlsi10.1 (difficulty: easy): calculate peak current and power cnsumption of a 100MHz clock driver with rise and fall times of 1ns driving 30k registers bits at 100fF each with Vdd = 3.3V

Result: $I_{peak} = 9.9A$, $P_d = 2.18$ Watt