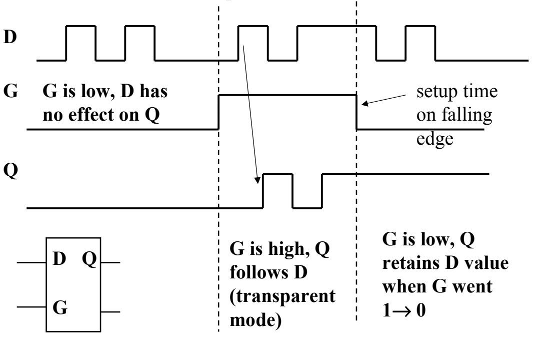
Latch Operation Revisited



G is low, D has no effect on Q

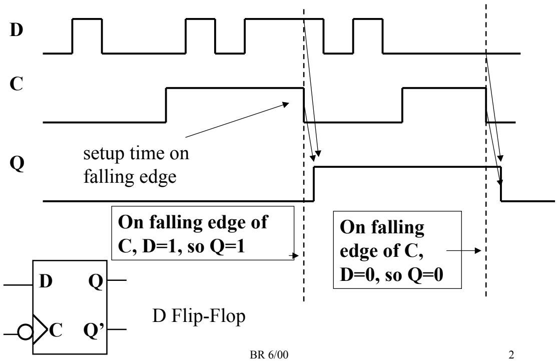
setup time on falling edge

G is high, Q follows D (transparent mode)

G is low, Q retains D value when G went 1→ 0

Transparent Latch

Flip-Flop Revisited



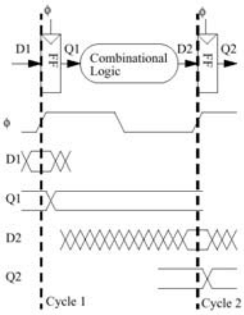
setup time on falling edge

On falling edge of C, D=1, so Q=1

On falling edge of C, D=0, so Q=0

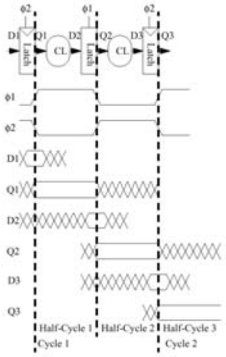
D Flip-Flop

System Design with Flip-Flops



Result propagate between FFs between active edges.

System Design with Latches



Two phase clocks.

Clock phases alternate pipeline stages.

While one set of latches is holding data, the other set of latches is capturing data.

System Timing with FFs

Clock Period = Tc2q + Tcl + Tsu + Tskew

Tc2q : Clock to Q prop delay
Tcl : max delay of combinational logic
Tsu : setup time of clock
Tskew : max clock skew

Clock skew is the difference in clock edge arrival times at different FFs in the system. If Clock arrives Tskew early, then Tskew adds to Tsu. If clock arrives Tskew late, then Tskew adds to Tc2q. Either way, additional delay is Tskew.

System Timing with Latches

At first glance, timing looks like (assume logic of FF example was broken evenly between two stages so total logic delay is same as FF example)

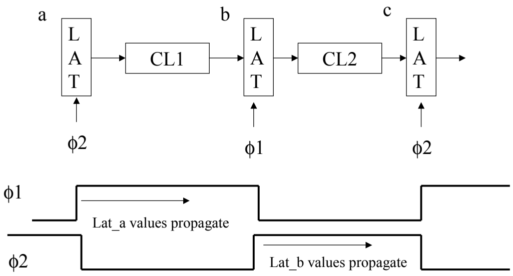
Clock Period = C2Q (first latch) + Tlogic + Tsu + Tskew +
C2Q (middle latch) + Tsu + Tskew

However, actual equation is:

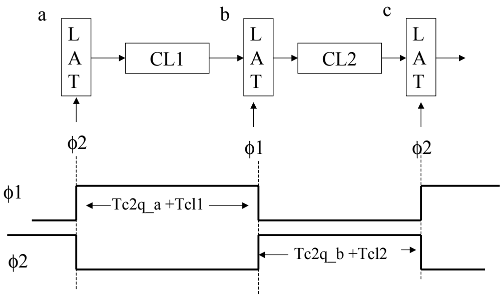
Clock Period = C2Q (first latch) + Tlogic +
C2Q (middle latch)

Tsu, Tskew does not enter into this equation because arrival of inputs from one phase can be used immediately.

Latch System Timing



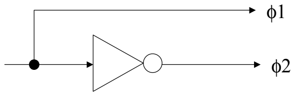
Latch System Timing



Clock period now equal to Tc2q_a + Tcl1 + Tc2q_b + Tcl2
Two latch delays = 1 DFF Tc2q(approximately)

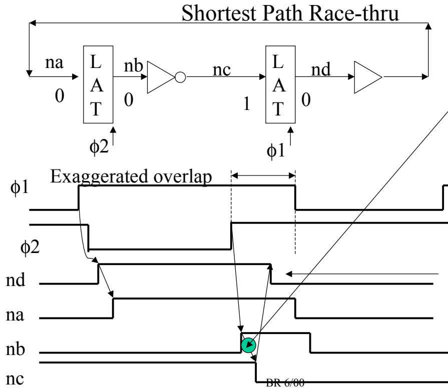
Skew, Clock buffer Delay

Skew, Clock buffer delay can cause clocks to overlap



Phases will overlap by at least inverter delay.

How much overlap can be tolerated?
Shortest Path Race-thru



Shortest path from φ2 latch output to φ1 latch input.

Race condition, wrong value in φ1 latch.

How much overlap can be tolerated?

Need to satisfy:

Tc2q + Tcl (min path) + Tsu > Toverlap

to avoid race condition in overlapped clock phases.

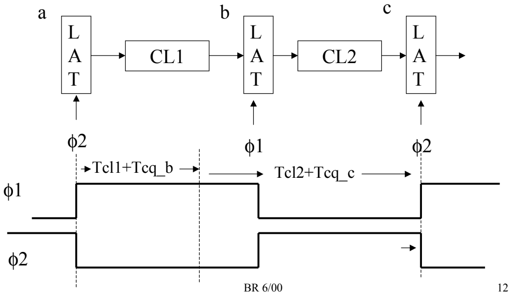
If all of overlap due to Tskew, then:

Tc2q + Tcl (min path) + Tsu > Tskew

Tcl(min path) > Tskew – Tcq – Tsu

Clock Borrowing

'Clock Borrowing' – logic delay not split evenly between stages – evaluation time spread between phases unevenly.



Clock Borrowing can be useful

Not always possible to perfectly balance logic delays in each stage.

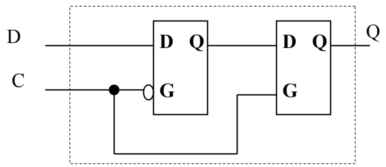
Latches allow us to do clock borrowing.

Helps alleviate the pipeline balancing problem.

Can we do clock borrowing in Flip-Flop System?

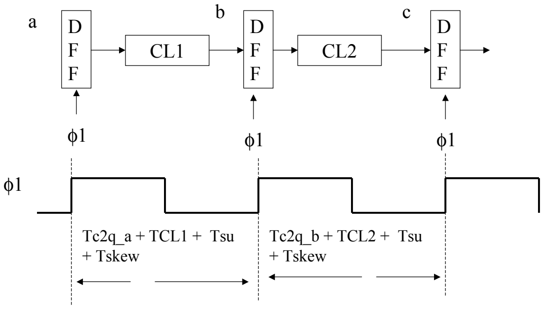
To do clock borrowing in a FF-system, we need a FF with *negative* setup time (input can change after active clock edge)

Master/Slave DFF Design

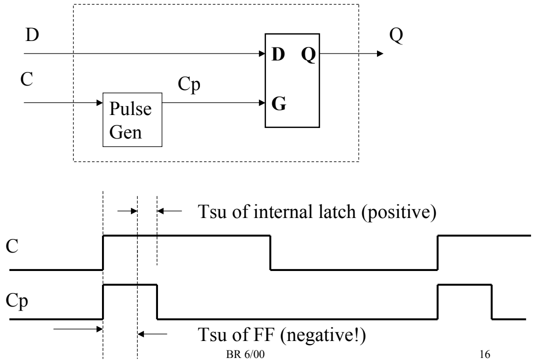


Rising edge-triggered, Tsu of FF is whatever Tsu is for the master latch (assume it is positive)

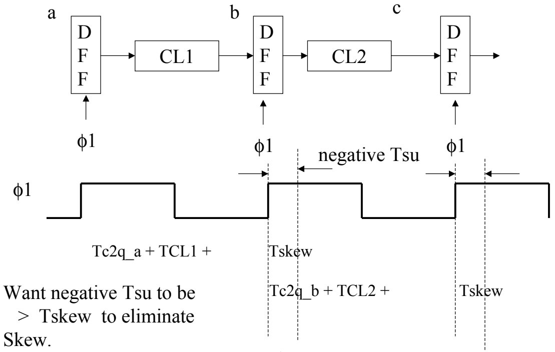
If DFF has positive Setup time, No Clock Borrowing



Rising Edge DFF using Pulsed Latch design



Clock Borrowing in DFF system (DFFs with negative Tsu)

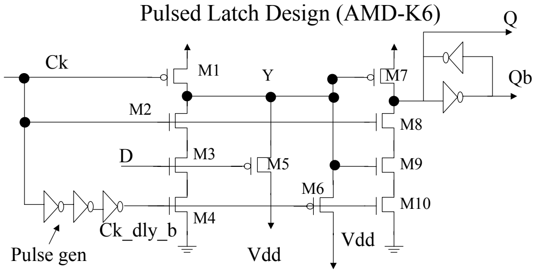


Want negative Tsu to be > Tskew to eliminate Skew.

Timing Summary

- Latch-based System
  - Allows clock borrowing
  - System timing is:
    Clock period = Tcl (phase1) + Tcl (phase2) + 2*Tc2q
  - If clock phase overlap, then:
    Tcl(min path) > Tskew – Tcq – Tsu
- DFF-based System
  - DFFs with positive Setup:
    Clock Period = Tc2q + Tcl (max) + Tsu + Tskew
  - DFFs with negative Setup:
    Clock period = Tcq + Tcl (max)
    if Tsu (negative) > Tskew

Pulsed Latch Design (AMD-K6)

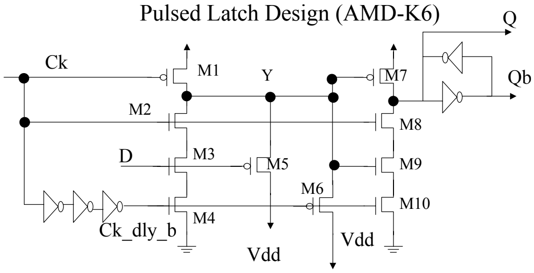


Ck =0 → M1 on, Y=Vdd → M7 off → Q = Qold

M2 off → M8 off

Ck_dly_b = 1 → M4 on → M10 on, M6 off

Pulsed Latch Design (AMD-K6)

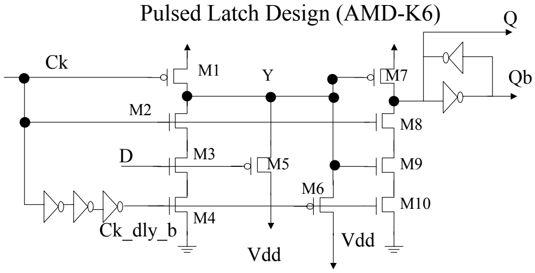


Ck 0 → 1 → M1 off, M2 on → D affects Y → Q new = Not (Y)

M4 on, M6 off

Ck_dly_b = 1 → M10 on

Pulsed Latch Design (AMD-K6)

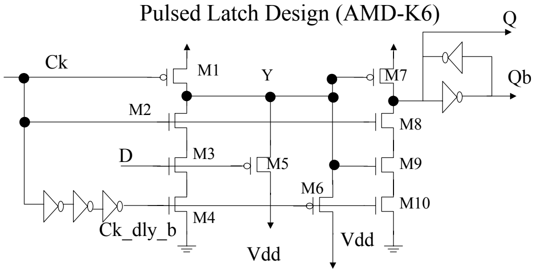


Ck 1 → M1 off, M2 on → Y = Vdd → Q = Qold

Ck_dly_b = 0 → M4 off, M10 off

M6 on

Pulsed Latch Design (AMD-K6)



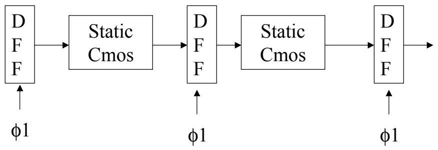
Ck 1 → 0 → M1 on → Y = Vdd → Q = Qold

M2 off,
M8 off

Ck_dly_b = 0 → M4 off, M10 off

M6 on

Pipelined DFF System (pulsed Latches) with Static CMOS

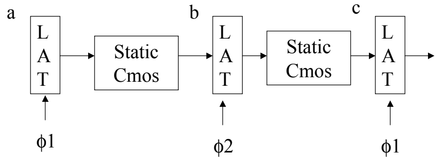


Clock period = Tcq + Tcl (max)

if Tsu (negative) > Tskew

Desireable Feature: want DFFs with low latency (small Tcq)

Pipelined Latch System with Static CMOS

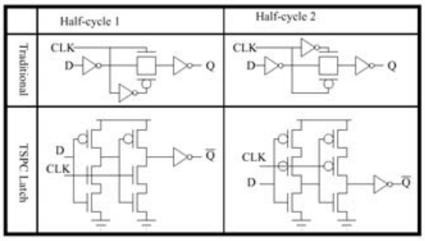


Clock Period = 2* Tc2q + Tcl (max path)

If clock phase overlap, then:

Tcl(min path) > Tskew – Tcq – Tsu (to avoid racethru)

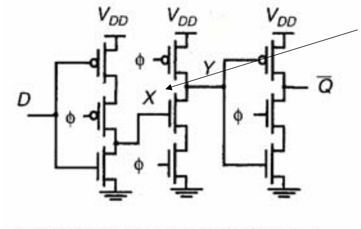
Level-Sensitive Latch Design



TSPC = True Single Phase Clocking (clock does not have to inverted), use on Alpha 21064 (first Alpha, one LARGE clock driver). TSPC latches slower than traditional latches, easy to invert clock locally.

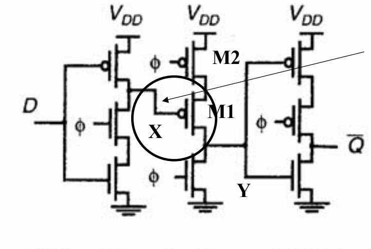
Flip-Flop Design

- Can combine TSPC Latches in several ways to make a Flip-Flop



(a) Positive edge-triggered *D* flip-flop

Node X is a dynamic node. If clock =1 and D=0, then node is floating! Lowers noise immunity, susceptible to charge sharing, charge coupling. Note that this FF has a long latency.

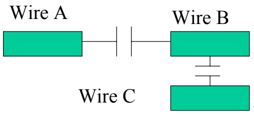


(b) Negative edge-triggered *D* flip-flop

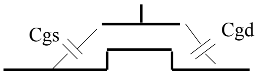
Charge coupling problem on this node. Node is truly dynamic, will be a case where it is not driven.

What is Charge Coupling?

Charge coupling is where a signal transition on one node causes a voltage change on another node via a coupling capacitor.

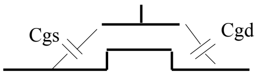


Charge coupling between wires due to sidewall capacitance or layer-to-layer capacitance



Charge coupling between gate and source/drain due to Cgs, Cgd.

Gate and Source/Drain Coupling

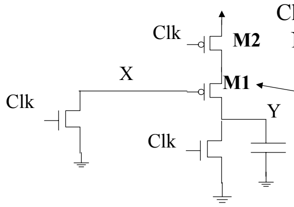


When channel is off, Cgs = Cgd = 0

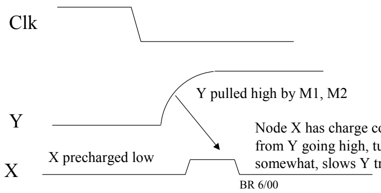
When channel is saturated (on), Cgs, Cgd is maximized.

Gate and Source/Drain coupling is a problem when the coupled node is not being actively driven (which is the case for a dynamic node).

Charge Coupling Problem in Negative Edge TPSC Dff



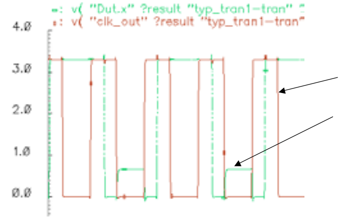
Note that X is precharged low, which turns on M1, maximizing charge coupling to node X from Y.



Y pulled high by M1, M2

Node X has charge coupled to it from Y going high, turns off M1 somewhat, slows Y transition.

X precharged low



Clk

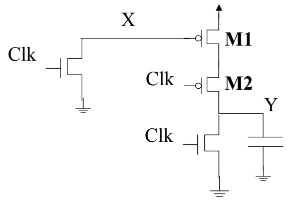
Node X, note charge coupling.

Only effect in this case is a slowing down of rise time of node Y, this will adversely affect H to L transition on output (TPHL)

For Vdd=3.3, 0.25u:

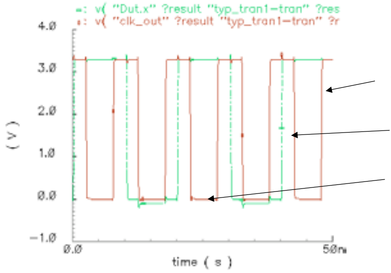
TPHL = 273 ps, TPLH = 90

How to fix this?



Swap positions of M1, M2. Note that both source and drain of M1 are now precharged high, eliminating charge coupling problem.

New Simulation



Clk

Node X

Note that charge coupling is eliminated.

For Vdd=3.3, 0.25u:

TPHL = 195 ps (improved by 30%)
TPLH = 85 ps

A Pulsed DFF

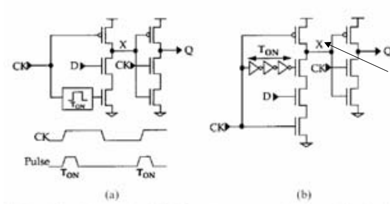


Fig. 1. Single-phase pulsed flip-flop: (a) concept and (b) possible implementation.

Dynamic node, not actively driven during most of evaluation

From: F. Klass, et.al., "A New Family of Semidynamic and Dynamic Flip-Flops with Embedded Logic for High-Performance Processors, IJSSC, Vol 34, No 5, May 1999.

Operation

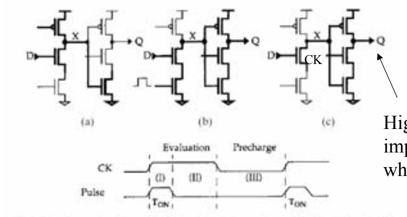


Fig. 2. Operation of a pulsed flip-flop: (a) precharge-III, (b) evaluation-I, and (c) evaluation-II.

High impedance when CK=0

Hold time = Ton time; Ton length is critical – too short, metastability, too long then hold time impacts performance.

Improved Design

Keeper gets rid of dynamic node.

Keeper ensures output always actively driven.

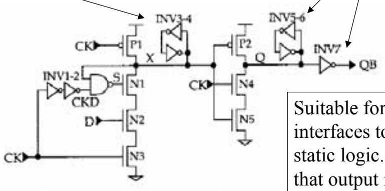


Fig. 3. Semidynamic edge-triggered flip-flop.

Suitable for interfaces to static logic. Note that output is stable through precharge.

Why NAND Gate?

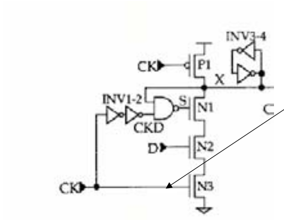


Fig. 3. Semidynamic edge-triggered fl

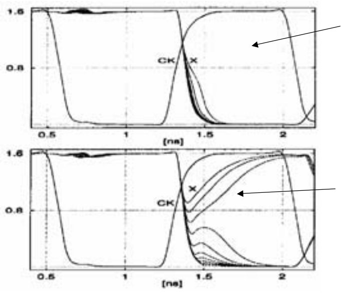
Pulldown path conditional with value of D.

If D=1, then X=0, making S=

In Precharge, CK=0, CKD=0, X=1, S=H, N1 = ON.

At Eval start, CK=1 (N3 on), CK=0, X=1. If D=1, then N2 on, so X pulled low. X going low forces S to stay high, resulting in more robust pulldown of node X. This is called “conditional shutoff” – N1 is not shutoff if D=1.

Spice Simulation Results



With conditional shutoff. Delay from CK to CKD decreased from 100ps to 10 ps in 10 ps steps (decreasing TON sampling window)

Original circuit without conditional shutoff. Note failure of pulldown of node X.

Embedded Logic

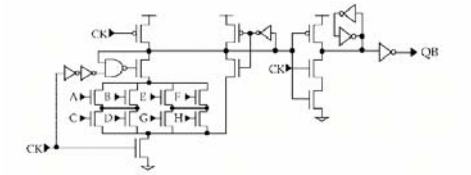


Fig. 5. SDFF with embedded $(A + B)\ (C + D) + (E + F)\ (G + H)$ function.

Embedded logic reduces number of logic stages between registers, but adds latency to DFF. Tradeoff is generally in favor of embedding logic in the DFF.

Dynamic DFF

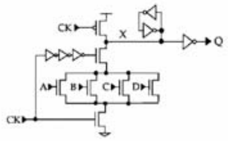


Fig. 6. Single-rail dynamic flip-flop with embedded $(A + B + C + D)$ function.

Used for interfacing to dynamic logic – output forced to a ‘0’ during precharge.

Dynamic Dual Rail DFF

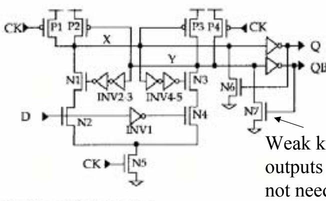
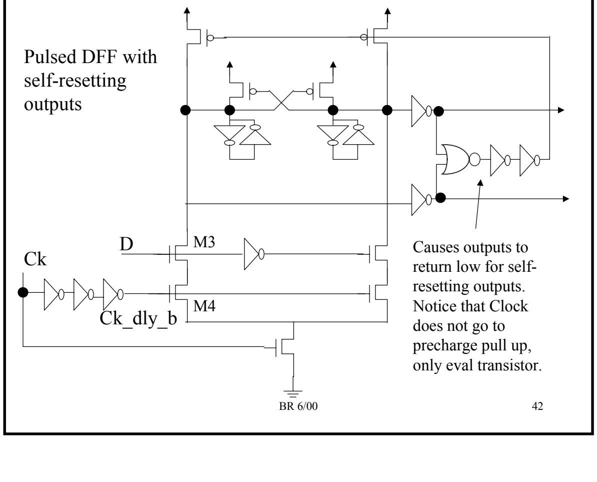


Fig. 7. Dual-rail dynamic flip-flop.

Weak keeper that keeps outputs at logic ‘1’. Do not need full keeper since outputs precharged to a ‘0’.

Useful when true and complement outputs are required for interfacing to dynamic logic because dynamic logic cannot implement inverting logic, so dual rail inputs are required.

Pulsed DFF with self-resetting outputs



Causes outputs to return low for self-resetting outputs. Notice that Clock does not go to precharge pull up, only eval transistor.

Self-Resetting CMOS (SRCMOS)

- SRCMOS differs from Domino logic in that the reset signal is generated locally
  - Less clock load
- Outputs are pulsed 0→1 → 0, inputs assumed pulsed as well
  - Because inputs are '0' at start of evaluation, then no need for evaluation transistor!
  - No evaluation transistor means one less transistor in pulldown path means a faster circuit
- Many schemes for generating reset
  - Delayed version of the clock
  - Generate reset based on dual rail outputs
  - Generate reset for an entire block of gates at one time
- Will look at this in more detail for decode circuits for RAMs.