SEQUENTIAL CIRCUIT DESIGN

* Sequential circuit o/p depends on the both past and present i/ps.

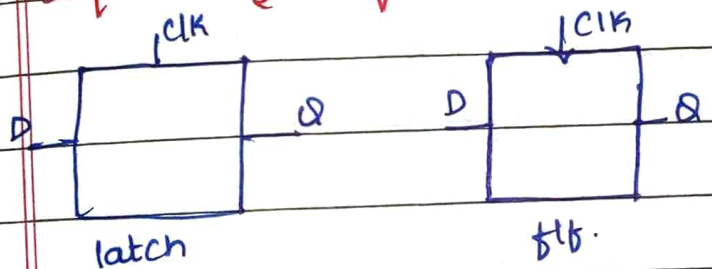
→ SC are mainly made up of flip-flops and latches. These are called as Sequencing elements.

→ The Sequencing elements inserts delay to the circuit.

① Dynamic Circuits → clock is one of the i/ps → Eg: domino, c²mos logic

② Static circuits → No clock. Eg: CMOS, pass transistors.

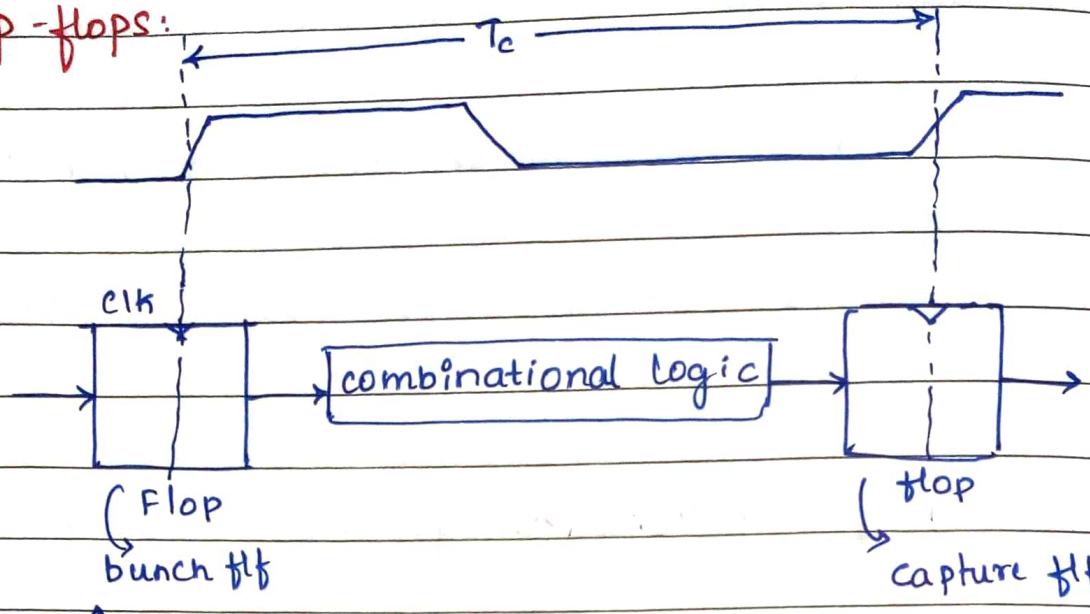
* Sequencing of Static Circuits:



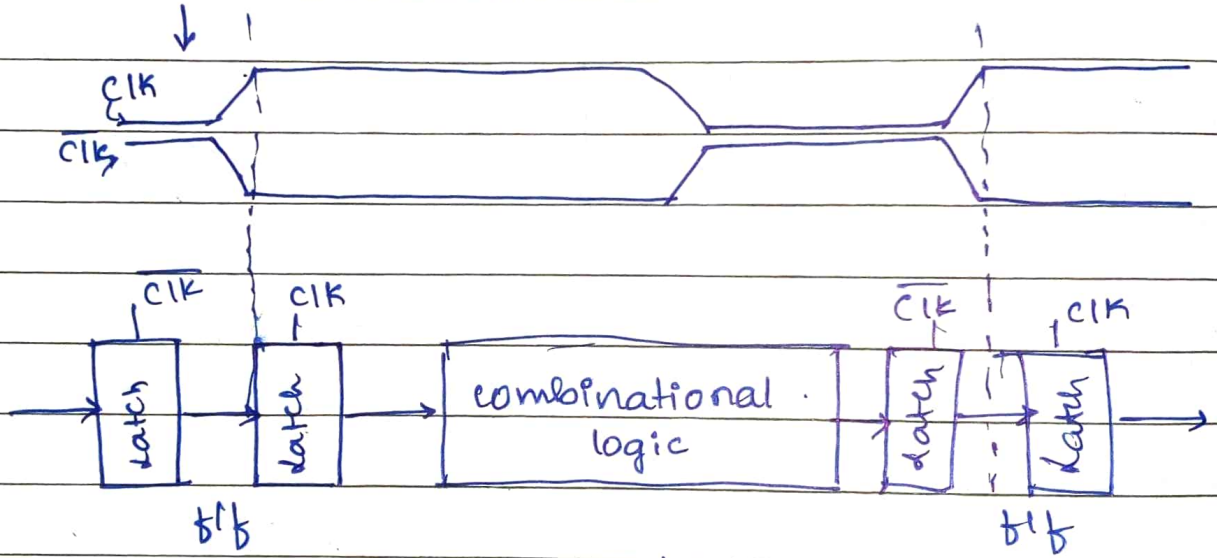
* Different types of sequencing methods:

1] flip-flops
2] 2 Phase transparent latches
3] Pulsed latches

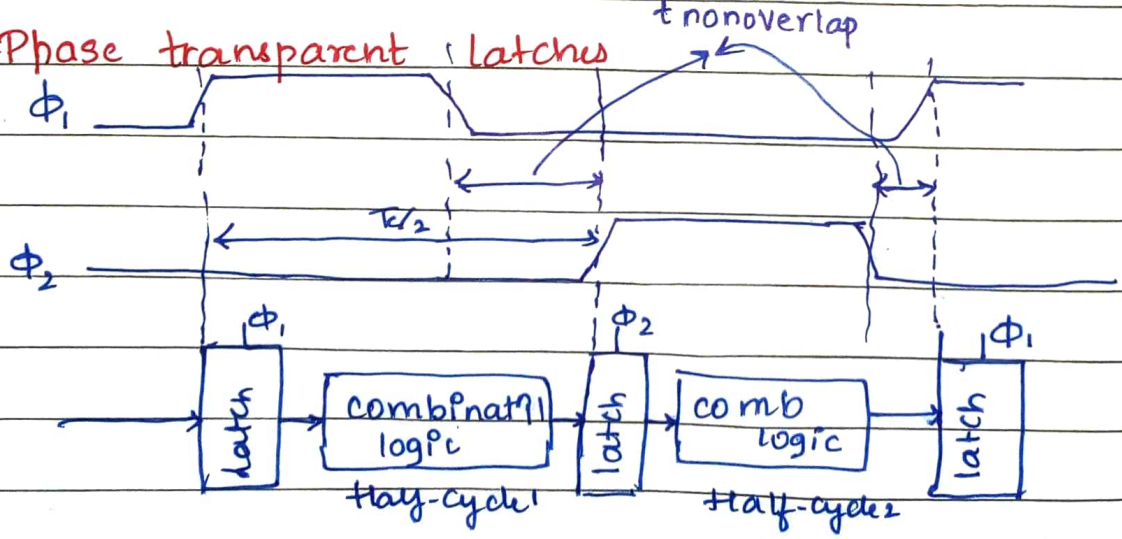
① flip-flops:



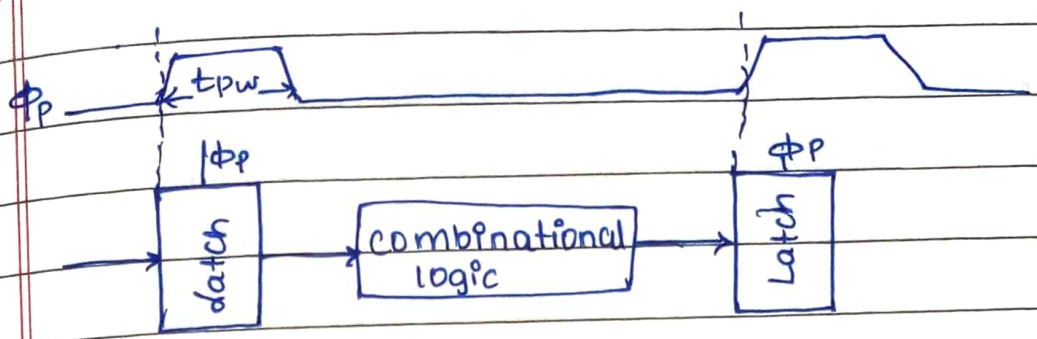
can be also connected as



② 2-Phase transparent latches

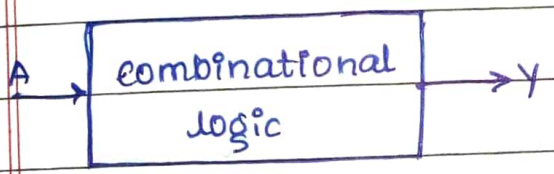


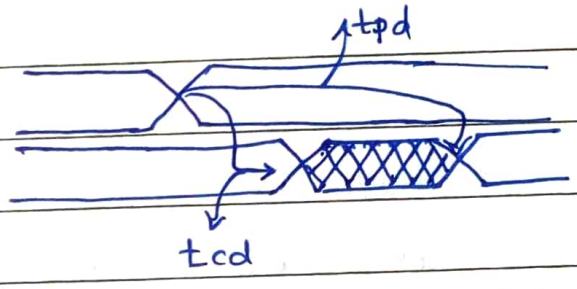
③ Pulsed latches



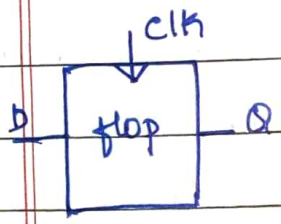
* Max-Delay Constraints:

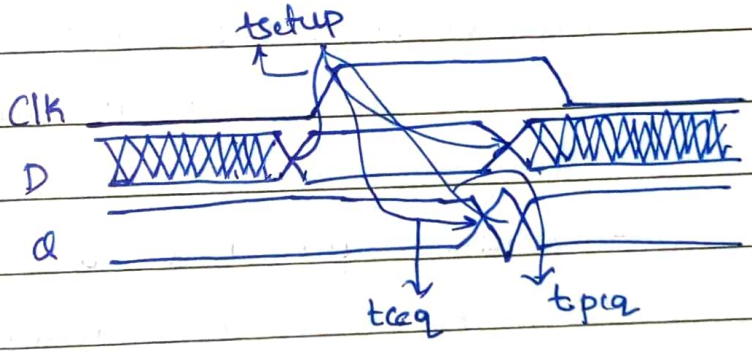
- Timing notations:





tcd → contamination delay tpd → propogation delay





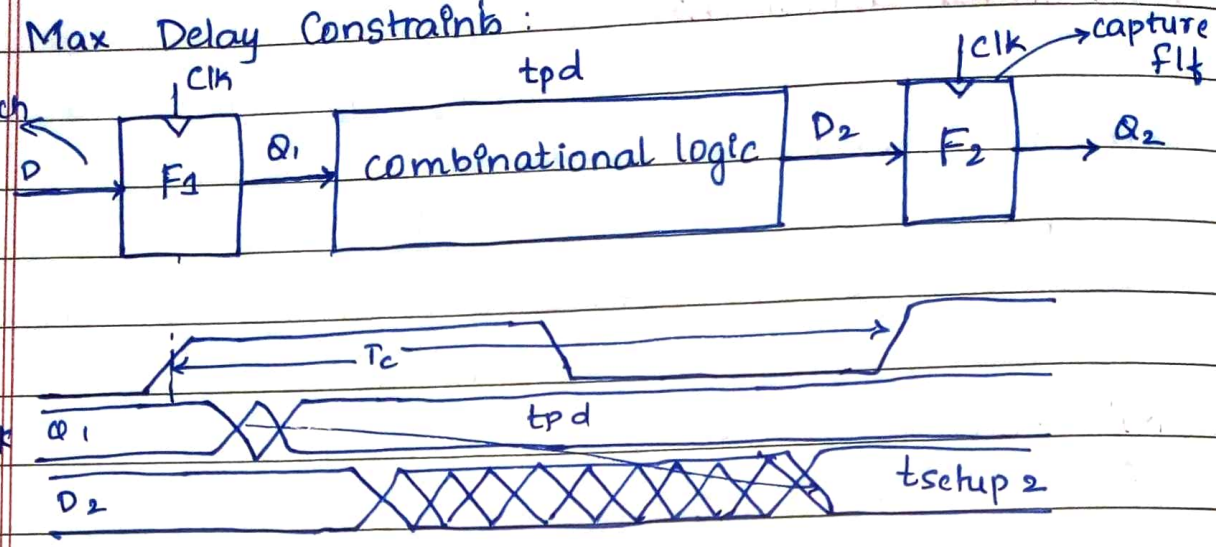
tsetup → Setuptime

thold → hold time

tcq → Contaminatn clock to Φ Q

tpcq → Propogatn clock to Φ

* Max Delay Constraints:



$$T_{cmin} \geq t_{pcq} + t_{pd} + t_{setup\,2}$$

$$T_{cmin} \longrightarrow f_{max}$$

→ Ideally clock pulse would be available for computation of combination logic

→ The sequencing overheads of flip flops consume these times.

→ If combinational logic delay is large the recieving element will miss setup time and sample wrong values resulting in setup time failure of max-delay failure.

→ Solution for this is increasing clock period.

→ The fig show max-delay timing constraints on a path from one flip-flop to next.

→ clock is assumed to be ideal

→ Path begin with rising edge of clock @ F_1 triggering F_1. The data propogates as o/p of F_1 is Q & through combinational logic to D. Setting up F2 before next rising

* ~~Min-delay constraints~~: edge.

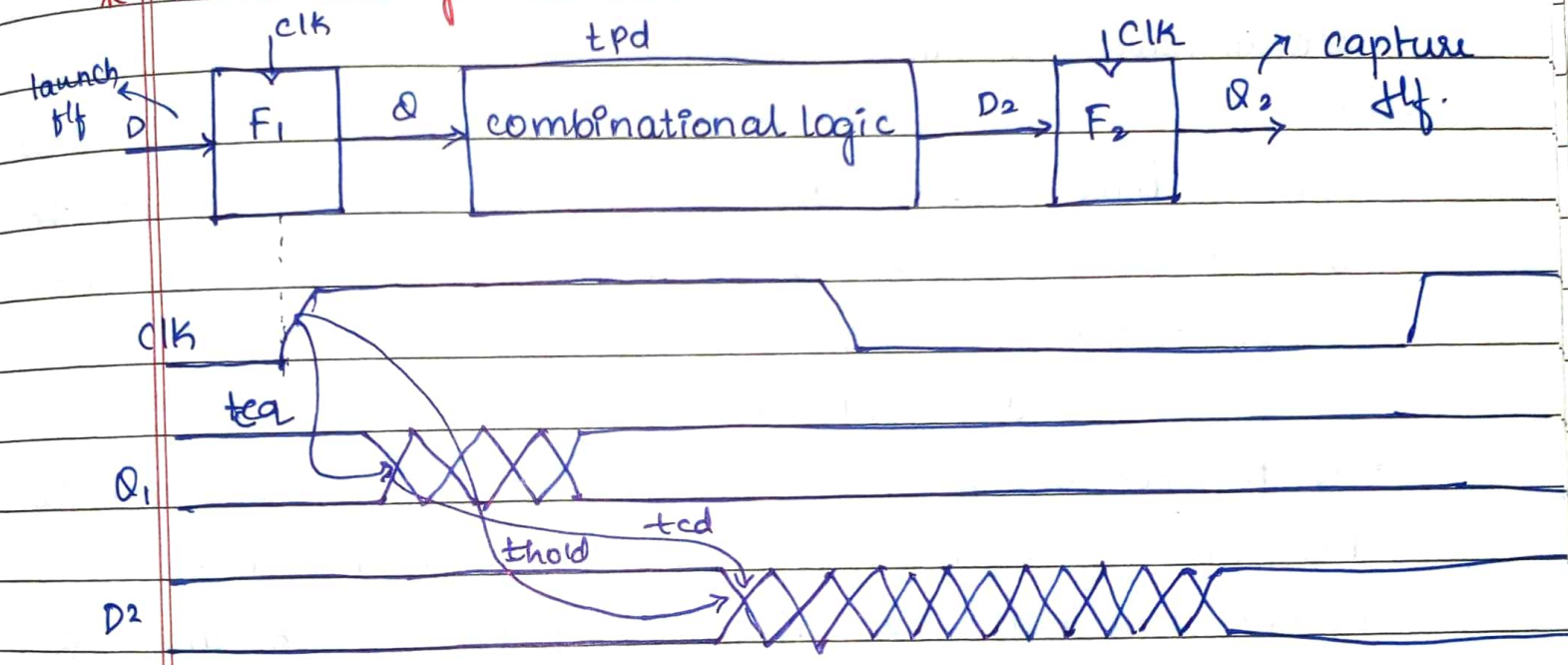
→ This implies that clock period must be large

→ Maximum delay is given by

$$t_{pd} \leq T_c - (t_{setup2} + t_{pcq})$$

↳ Sequencing overhead.

* Min-delay Constraints



→ Sequencing elements can be placed back to back without any intervening combinational circuit and still get proper functionality.

→ But if the hold time is large and the contamination delay is small then data can incorrectly propogate between the stages/system.

→ This is called race condition, hold time failure, min-delay failure

→ It can only be fixed by redesigning the logic, not by slowing the clock.

→ fig shows the min-delay timing constraints on a path from 1 flf to next assuming ideal clocks with no skew.

→ The path begins with the rising edge of clock triggering F1.

→ The data may begin to change @ Q1 after a clk-to-Q contamination delay, & @ D_2 after another logic contamination delay

→ However, it must not reach D2 until @ least the hold time thold after the clock edge, lest is corrupt the contents of F2. Hence we solve for the minimum logic contamination delay:

$$t_{cd} \geq t_{hold} - t_{ccq}$$

→ If the contaminat^n delay through the flip-flop exceeds the hold time, you can safely use back-to-back f-fs.

→ If not, you must explicitly add delay between the flfs. (e.g. with buffer) or use special slow f-fs with greater than normal contamination delay on paths that require back-to-back flops.

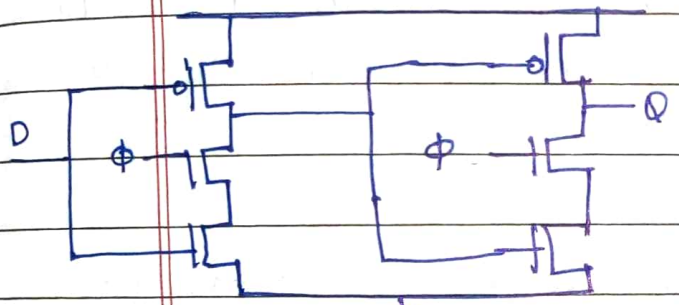
→ scan chains are common example of paths with back-to-back flops.

True Single-phase-clock (TSPC) latches and flip-flops

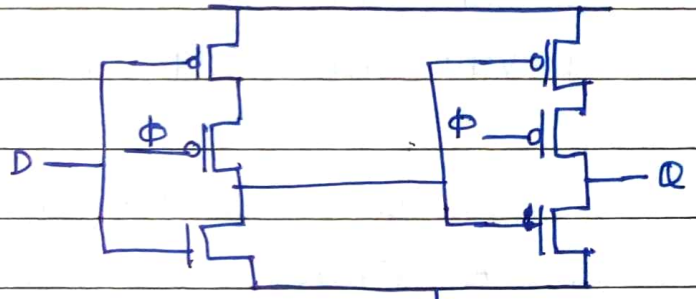
→ Conventional latches require both true and complementary clock signals.

→ In modern CMOS systems, the complement is normally generated locally with an inverter in the latch cell

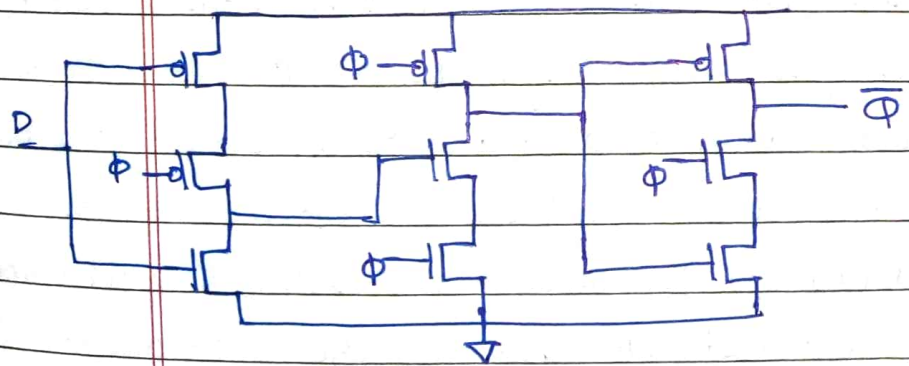
→ The TSPC latches and flip-flops replace the inverter-transmission gate or C²MOS stage with a pair of stages requiring only clk, not its complement.



Active high TSPC dynamic latch



Active low TSPC dynamic latch



TSPC dynamic f-f.

→ Note that this f-f produces a momentary glitch on $\bar{Q}$ after the rising clock edge when D is low for multiple cycles; this ↑s the activity factor of downstream clks and costs power.

→ extends TSPC principle to handle domino, RAMs, and other precharged circuits.

→ The dynamic TSPC latches were used on groundbreaking Alpha 21064 microprocessor.

→ Logic can be built into first stage of each latch.

→ The latch is not easy to staticize.

→ In any case, the clk must also be reasonably shap to prevent races with both transistors are partially ON.

GLOBAL CLOCK GENERATION.

A global clock generator receives an external clock signal and produces the clock that will be distributed across the die. Clock generator is simply a buffer to drive the large capacitance of the clock distribution network. However, the input pad, bufferings, distributn network, & gates have significant delay that leads to a large skew bt external clk & physical clks received @ clked elements.

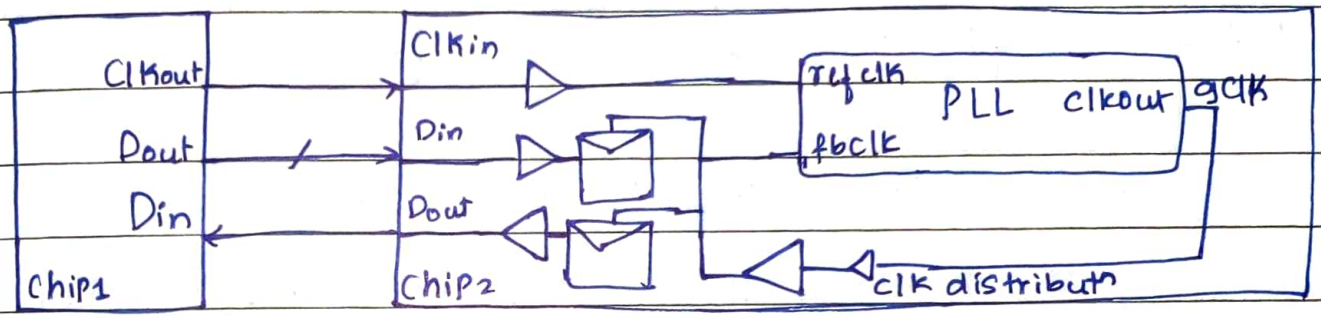
Moreover, this delay varies with processing & environment Becos of this skew, clocked elements on the chip are no longer synchronized with external I/O sigs.

Guaranteeing setup & hold times becomes problematic, particularly @ high freq where skew exceeds half of clk period.

More sophisticated clock generators use phase-locked loops (PLLs) to compensate for this delay.

PLLs can perform freq multiplicatn to provide an on-chip @ a higher frequency than the external clock.

→ synchronous chip interface with PLL

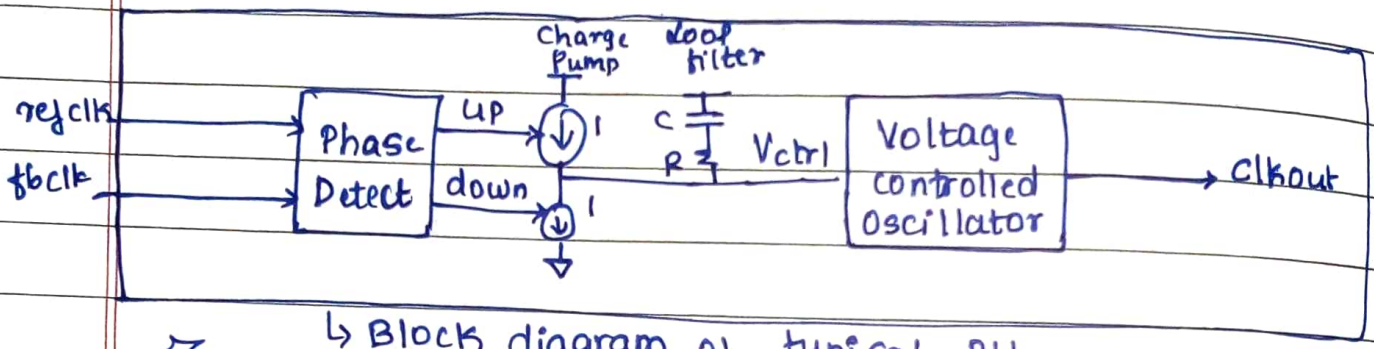


Chip1 sends a clk and Dout to chip2 and receives Din back from chip2. The data should be synchro-nized to clk so each chip can sample it on the +ve clk edge. However, the internal clk on chip2 is delayed through clk distribution network. The PLL adjusts internal clock in chip2 to correct for this delay & keep the clock synchronized with the data.

clk

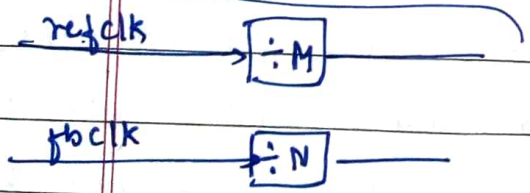
→ A PLL receives a reference clock and a feedback clock and produces an o/p clock. The phase and frequency of the output clock is automatically adjusted until the feedback clock is exactly aligned with the reference clk.

In our applicatn, ref clk comes from chip 1. The fdbk clk is tapped off one of the physical clock wires anthat also drives registers. The Op clk gclk signal sent into clock distribut network. ∴ PLL adjusts gclk until clk driving data registers is aligned with the external clock exclk. This eliminates the symmetric skew caused by the clock distribution delay



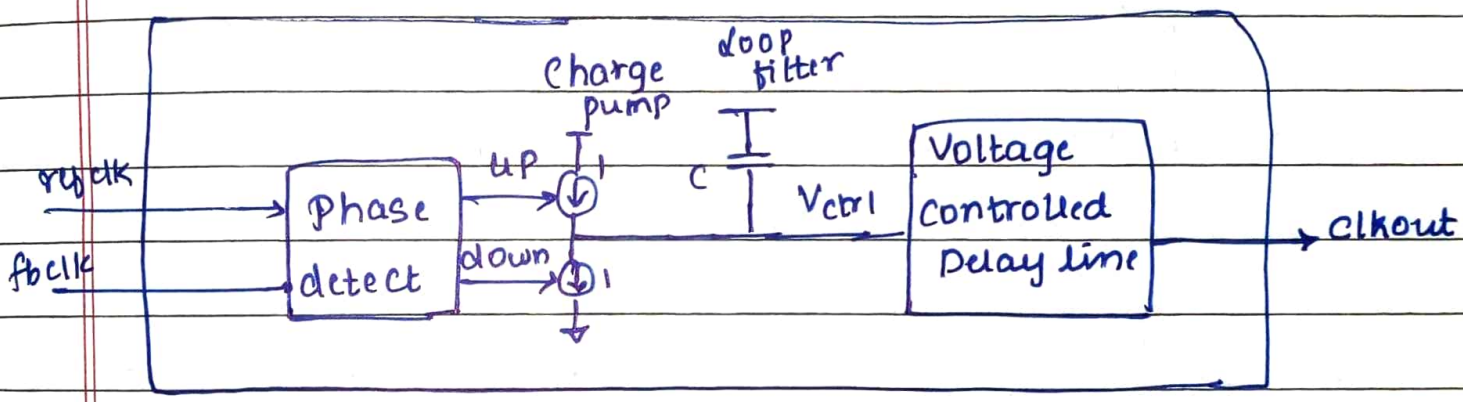
↳ Block diagram of typical PLL

The heart of the PLL is a Voltage-controlled oscillator (vco). The control voltage is adjusted until the oscillator produces an Olp clk of proper phase @ same freq as ref clk. This control is performed with charge pump & RC filter. A phase detector determines whether fdback clk leads or lags ref clk. The charge pump consists of a pair of current sources enabled by the up and down signals to adjust the voltage on Vctrl until the feedback clk becomes aligned with the reference.



The above diagram is block diagram of PLL the performs freq multiplicatn. It uses a divide-by-N counter on the fbclk to generate a clk aligned in phase but @ N times freq of the ref clk. Ex: the Pentium 4 uses a 100 MHz external sys clk that can be multiplied up to a 3GHz core clock. With another divide-by-M counter on refclk terminal, PLL can produce any rational N/M multiple of the i/p freq.

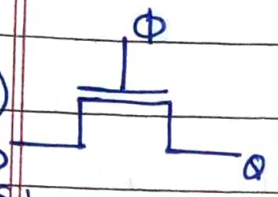
★ DLL (Delay-locked loop)



Conventional CMOS latches

→ Pass Transistor latch
→ transparent latch buffer

a)



→ has four limitations

1] Olp doesn't swing from rail to rail (from gnd- vdd).

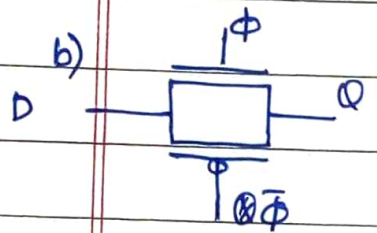
2] It never rises above $V_{DD}-V_t$

3] dynamic output

4] potential noise issue.

limitations:
→ olp floats, diffusion i/p
→ olp noise sensitivity

b)

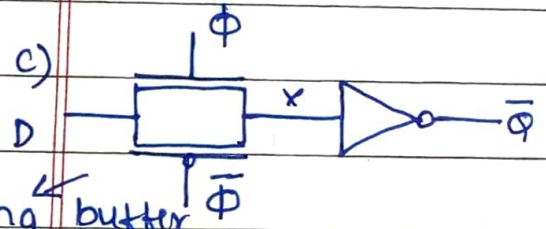


→ uses CMOS TG (~~transmission gate~~) instead of single CMOS pass transistor to offer rail to rail olp swing

No V_t drop

→ It require additional complementary clock.

c)

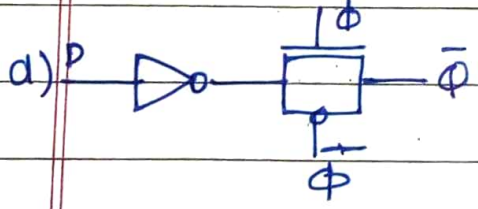


→ Adds an output inverter. so that the state node is isolated from the output noise

→ This creates an inverting latch.

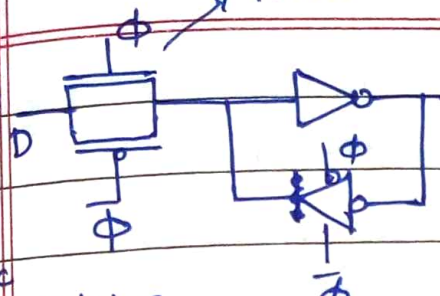
Inverting buffer
+ Restoring
- inverted olp

d)



→ behaves as inverting latch with buffered input and non buffered output.

e) Tristate feedback

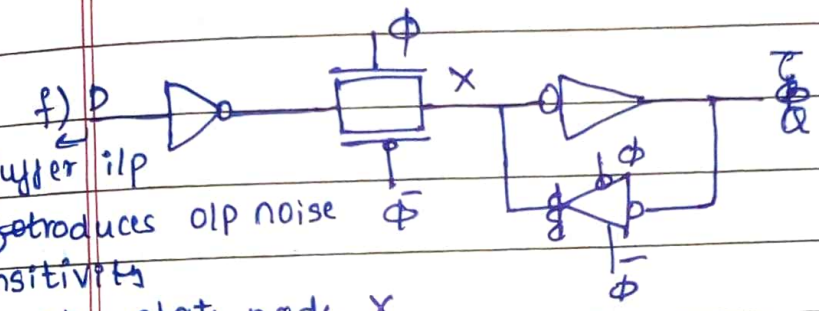


→ When clock = 1 input transmission gate is ON and feedback tristate is OFF

→ When clock = 0 Input transmission gate is OFF however feedback tristate is ON.

+ static
+ Static latches are now essential

f)

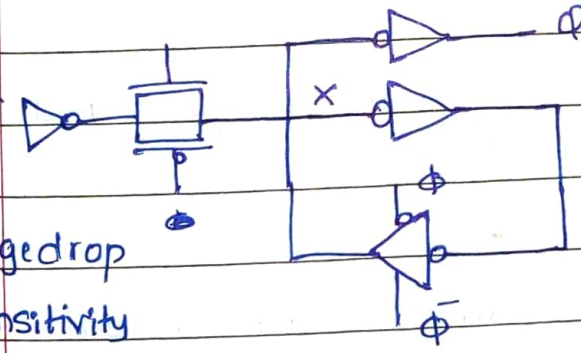


Buffer i/p
- Reintroduces o/p noise sensitivity
- corrupts state node X

→ adds an input to the buffer so that there is input @ the transistor gate rather than unbuffered diffusion

but both (e) & (f) introduces large noise glitches @ the output that may enter @ x through the feedback circuit.

g)

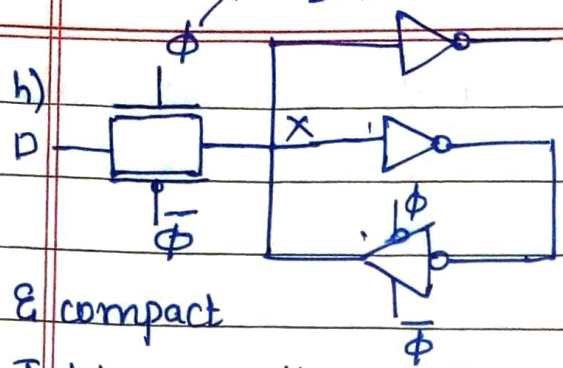


Buffered o/p
+ Static
+ No voltage drop
+ Noise sensitivity
+ diffusion i/p

→ Robust transparent latch that address all the deficiencies mentioned so far.
→ The latch is static
→ all nodes swing rail to rails
→ Input drivers transistor gate rather than diffusion
→ State noise is isolated.

→ Datapath latch → used data path applications

h)



→ Input noise is preferably controlled

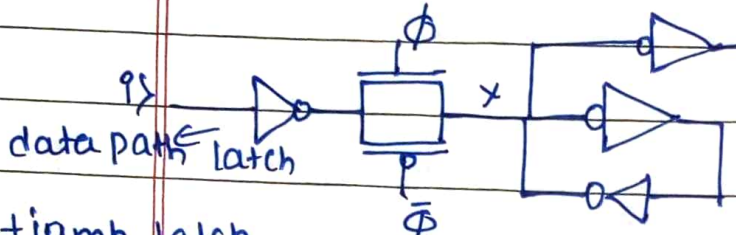
→ much faster

→ more compact.

↑faster & compact

Ex: Intel uses this as datapath latch

i)



data path latch

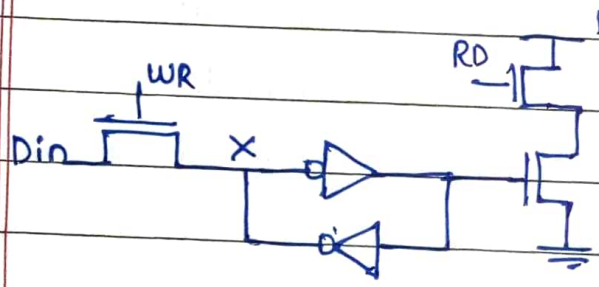
↑jamb latch

→ jamb latch, a variation of g

→ Reduces clock load

→ Saves two transistor by using a weak feedback inverter inplace of a tristate.

j)



→ jamb latch commonly used in register files, FPGA cells

→ They read out on to a sing data output line.

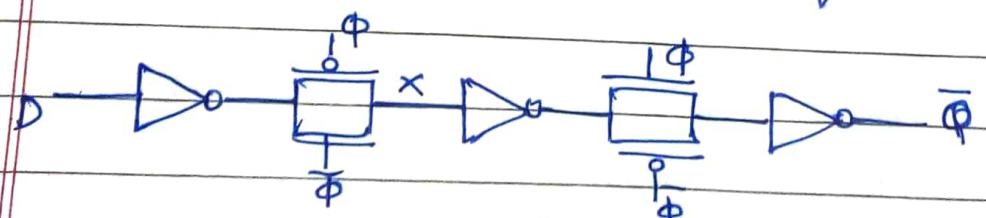
→ and any one is enabled (Read/write) @ any given time with its RD signal.

→ commonly used in register files and fpga cells

Conventional flip-flop design

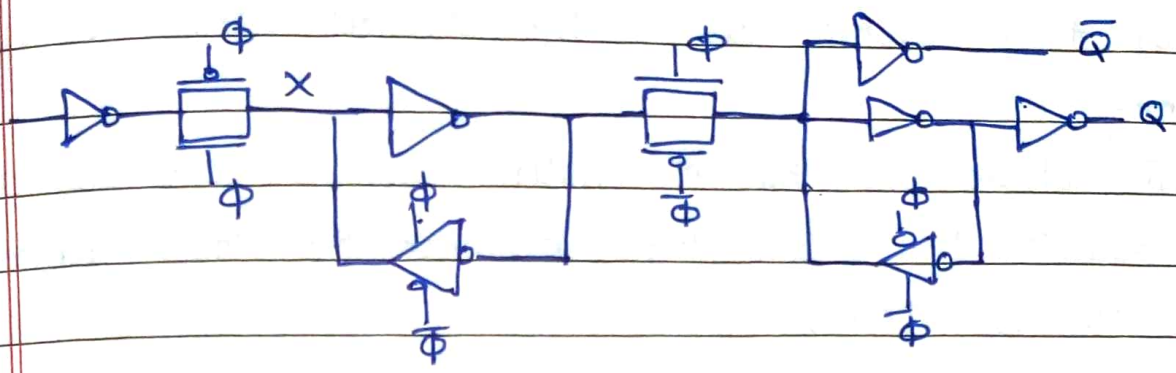
- flip-flop is built as pair of back-to-back latches

a) →



a) → dynamic inverter flflop

b) → with feedback , non-inverting static flipflop



Global clock distribution.

→ The global clock must be distributed across the chip in a way that reaches all of the clocked elements @ nearly same time.

The clock distribution system must be carefully designed to equalize the flight time between the clock generator and the clocked receivers

→ can be classified as grids, H-trees, spines, adhoc, or hybrid.

★ **Grids:** A clock grid is a mesh of horizontal and vertical wires driven from the middle or edges.

The mesh is fine enough to deliver the clock to points nearby every clocked element. The resistance is low bt any two nearby points in the mesh so the skew is also low bt nearby clocked elements.

This reduces the chance of hold-time problems because such problems tend to occur bt nearby elements where propogatn delay bt elements is also small.

Grids also compensate for much of random skew because shorting the clock together makes variaths in delay irrelevant.

They can be routed early in the design without detailed knowledge of latch placement. However, grids do have significant systematic skew bt pts closest to drivers and the

points furthest away. They also consume a large amnt of metal resources & hence have a high switching cap and power consumptn.

★ H-Trees An H-tree is a fractal structure built by drawing an H shape, then recursively drawing H shapes on each of the vertices.

With enough recursions, the H-tree can distribute a clock from the center to within an arbitrarily short distance of every point on the chip while maintaining exactly equal wire lengths. Buffers are added as necessary to serve as repeaters. If the clock loads were uniformly distributed around the chip, the H-tree would have zero systematic skew.

Moreover, the trees tend to use less wire and thus have lower capacitance than grids.

In practice, H-tree still shows some skew because the clock loads are not uniform, loading some leaves of the tree more than others. Moreover, the tree often must be routed around obstructions such as memory arrays. The leaves of H do not reach every point on the chip, so some short physical clock wires are required after the local clock gater.

★ **Spines:** As with the grid, the clock buffers are located in a few rows across the chip. However, instead of driving a single clock grid across entire die, the spines drive length-matched serpentine wire to each small group of clocked elements. If loads are uniform, the spine avoids the systematic skew of the grid by matching the length of the clock wires. Each serpentine is driven individually so gaters can be used to save power by not switching certain wires.

The serpentine is also easy to design and each load can be tuned individually. However, a system with many clocked elements may require a large number of serpentine routes, leading to high area and capacitance for the clock n/w. Like trees, spines also may have large local skews between nearby elements driven by different serpentines.

★ **Ad-hoc:** Many ASICs running @ relatively low frequencies (100's of MHz) still get away with an ad-hoc clock distribution network in which the clock is routed haphazardly with some attempt to equalise wire lengths or add buffers to equalize delay. Such adhoc NIWs can have reasonably low systematic skews because the buffers sizes can be adjusted until the nominal delays

are nearly equal.

However, they are subject to serve random skew when process variations affect wire and gate delays differently. This is the level that most commonly available tools support. Most design teams using ad-hoc clock networks also lack the resources to do a careful analysis of random skew, jitter, & drift.

∴ They should be conservative in defining a skew budget & must be careful about hold time violations.

* hybrid: A hybrid combination of the H-tree and grid offers lower skew than either an H-tree or grid alone. In the hybrid approach, an H-tree is used to distribute the clock to a large number of points across the die. A grid shorts these points together. Compared to a simple grid, the hybrid approach has lower systematic skew because the grid is driven from many points instead of just the middle or edge.

Compared to H-tree, the hybrid approach is less susceptible to skew from nonuniform load distribution. The grid also reduces local skew & brings the clock near every location where it is needed.