Lecture 15:

Clock Recovery

Computer Systems Laboratory
Stanford University
horowitz@stanford.edu

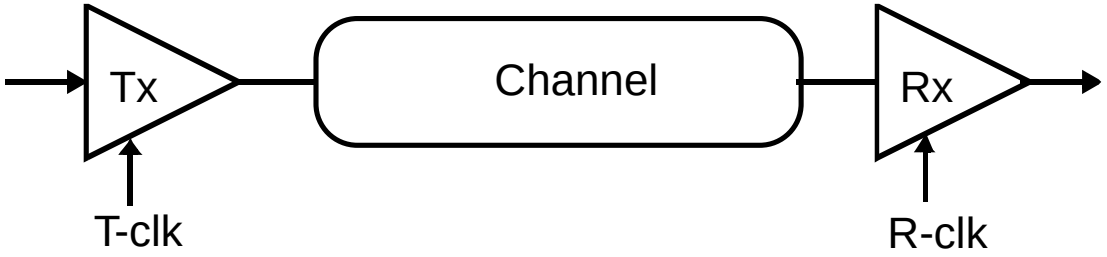
Overview

- Reading
  - Chapter 19 - High Speed Link Design, by Ken Yang, Stefanos Sidiropoulos

- Introduction
  - One of the critical tasks in building high-speed IO is getting the receive clock to be properly aligned to the incoming data. This means you need to control the phase (and sometimes the frequency) of the receive clock. Clock alignment is usually done using a feedback system that controls the phase, and is called a phase-locked loop or PLL. There are two ways to build this kind of system, one using a voltage controlled oscillator and the other using a delay line.

Timing

- The timing (clocking) discipline dictates the transmission and sampling of the signals on the channel:

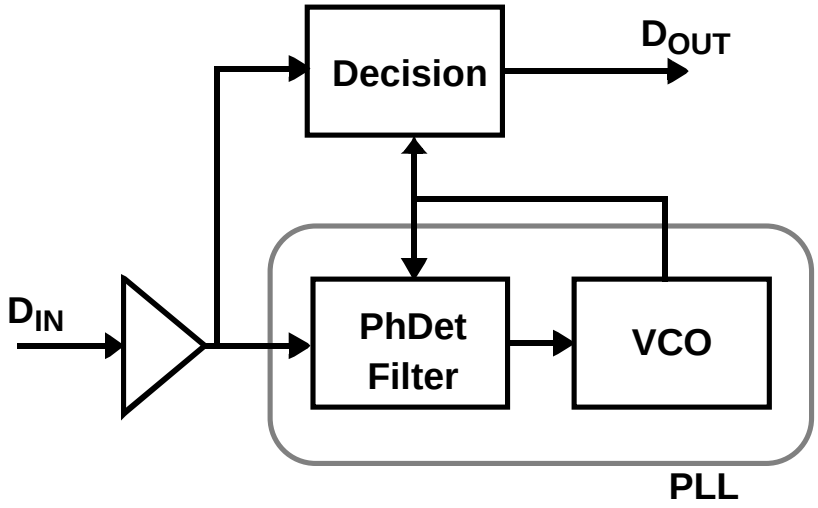


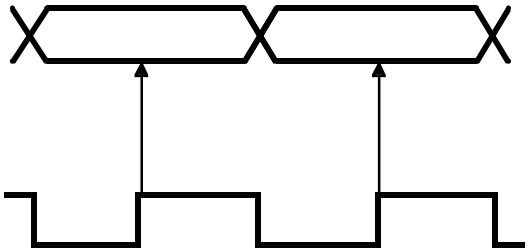
- i.e. determines how we generate the clocks that drive the transmitter and receiver ends of the link
- Clocking circuit design is tightly coupled with signal encoding for timing recovery:
  - High-bandwidth serial links recover timing based on the transitions of the data signals (need encoded data to guarantee spectral characteristics)
  - Low latency/parallel systems use a source synchronous discipline (transmitter clock is sent along with the data)
    - The basic circuit block is a Phase Locked Loop

Outline

- Clock-recovery/phase-alignment approaches
  - Traditional CDRs
  - Oversampled CDRs
  - Source synchronous links
- Timing Loop Design
  - Delay Locked Loops
  - Phase Locked Loops
- Circuit Components
  - Variable delay/frequency generation
  - Phase Detectors
  - Filters

Classic Clock/Data Recovery

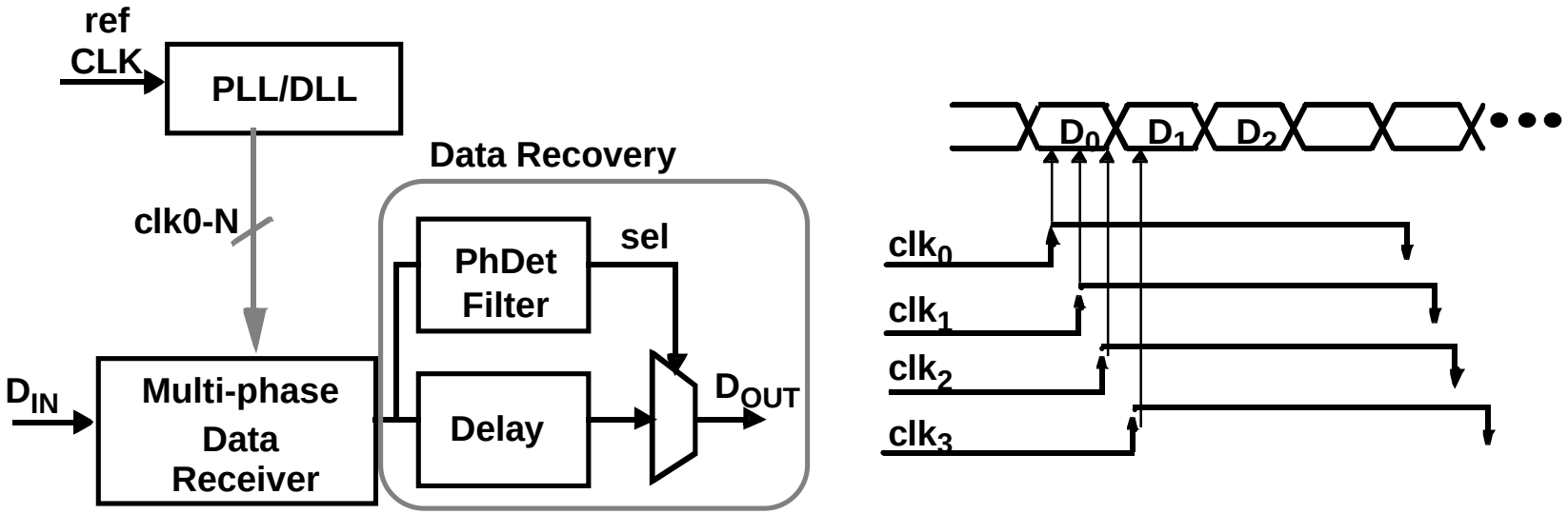




- Many different implementations ([1]-[5])
- Data stream must guarantee transitions (i.e. PSD content)
- State of system is stored in analog filter

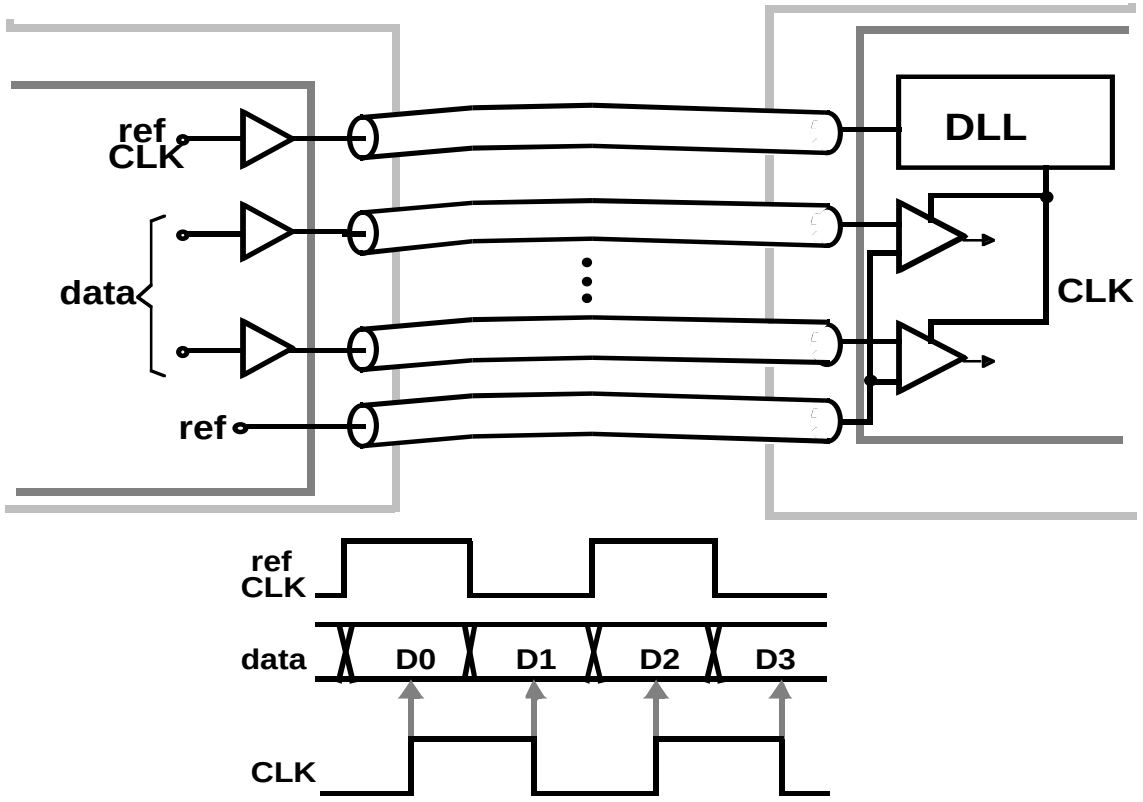
Oversampled Clock/Data Recovery

- Oversample the data and perform phase alignment digitally



- Alternatives range from closed digital loop systems to feed-forward systems ([6]-[9])
- De-couples the clock generator from the tracking of the data
- Still data must guarantee transitions to ensure proper tracking

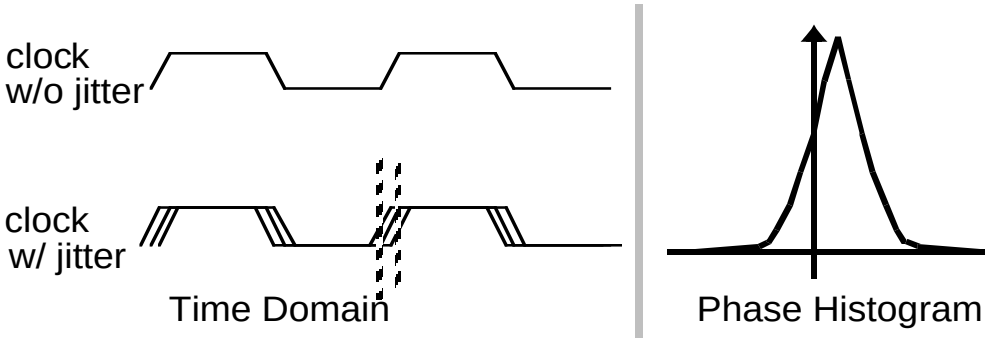
Phase Alignment in Source Synchronous Systems



- Timing information is carried by an explicit clock signal ([10]-[13])
- State can be stored either in analog filter or digital logic

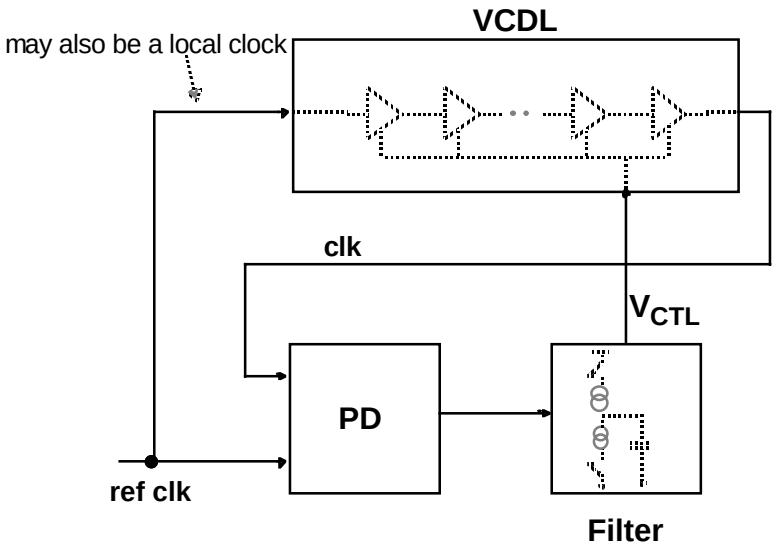
Timing Loop Performance Parameters

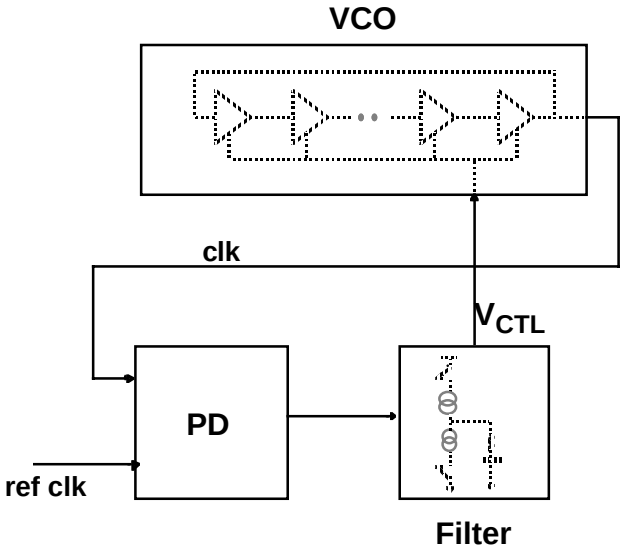
- Phase Error:



  - AC - jitter: The uncertainty of the output phase
  - DC - phase offset: Undesired difference of the average output phase relative to the input phase.
- Bandwidth: Rate at which the output phase tracks the reference phase
- Lock time, Frequency Range
- Duty cycle (in classic CDRs and most source synchronous systems)
  - Spacing uniformity of multiple edges (in oversampled CDRs)

Loop Architectures: DLL vs PLL

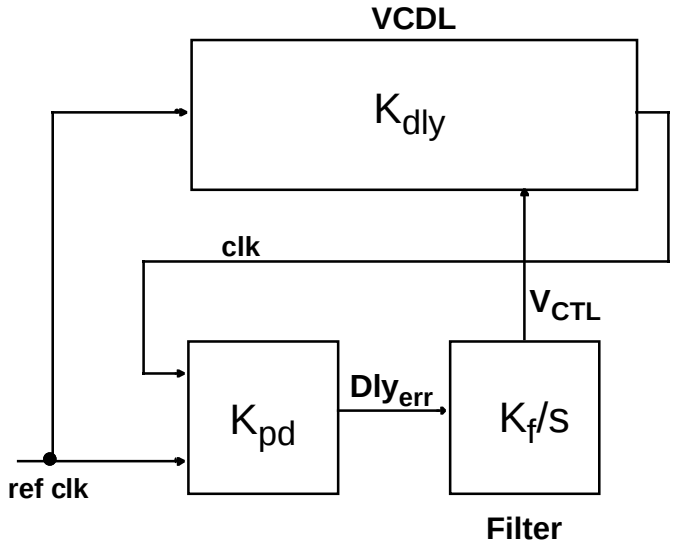




- First order loop:
  - easily stabilizable
  - frequency synthesis a problem
  - ref clk jitter passes through
  - no phase error accumulation
- Second/Third order loop:
  - stability is an issue
  - frequency synthesis easy
  - filtering of ref clk jitter
  - phase error accumulation

Delay Locked Loop

- Controlled variable is *delay* through the VCDL



K_{dly} (sec/V)

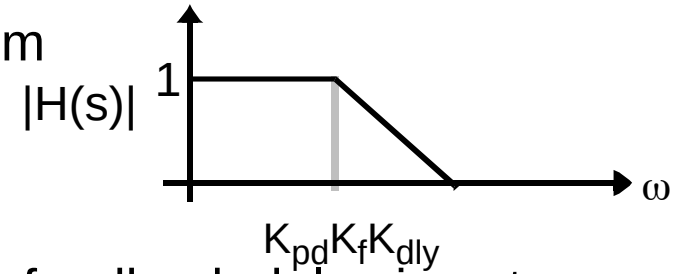
$K_{pd}K_f$ (V/sec^2)

- Open Loop TF: $T(s) = \dfrac{K_{pd}K_fK_{dly}}{s}$

- Closed Loop TF: $H(s) = \dfrac{K_{pd}K_fK_{dly}}{s + K_{pd}K_fK_{dly}}$

DLL Dynamics

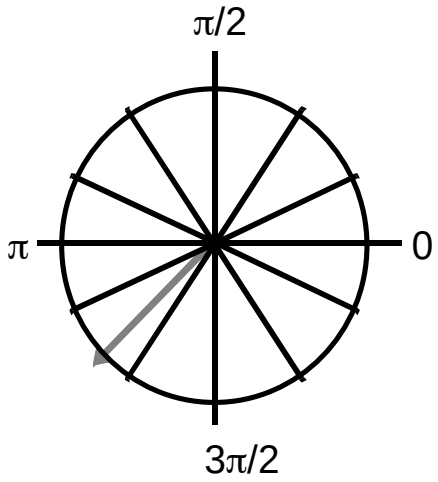
- Single pole system

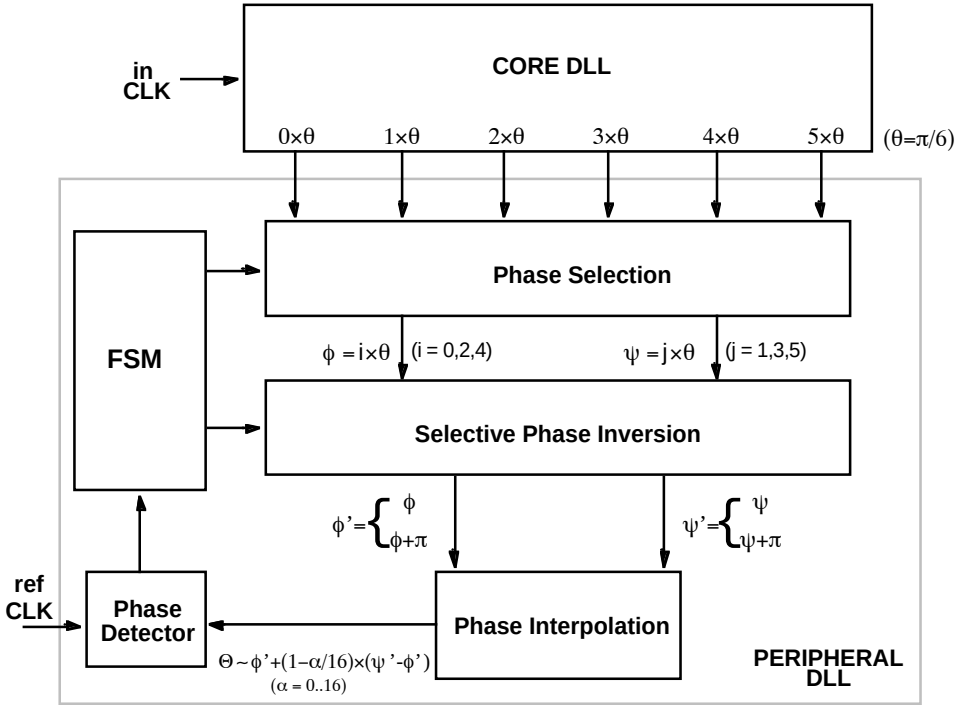


- Stable as long as feedback delay is not excessive
- Jitter sources:
  - Device noise: usually negligible
  - Noise sensitivity of the delay line
  - Noise sensitivity of the subsequent clock buffer
- System issues:
  - Phase noise of the input signal -> systems with DLL's require low jitter differential clocks
  - Limited locking range -> need to ensure adequate VCDL range and employ special reset

Interpolating DLL's

- Pick two successive coarse edges and then interpolate between them to generate the desired output phase [13], [22], [23]:

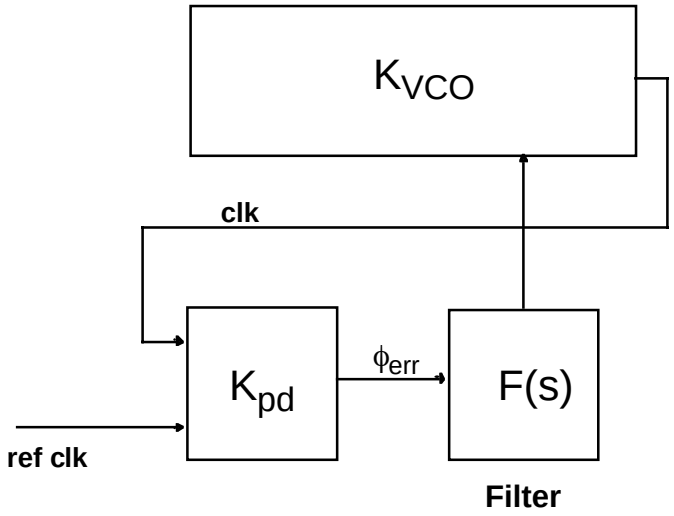




  - No range boundaries on the generated delay
  - Can use digital control

VCO-based Phase Locked Loop

- Controlled variable is *phase* of the output clock



K_{VCO} (Hz/V)

$K_{pd}F(s)$ (V/rad)

- Main difference from DLL is the VCO transfer function: $H_{VCO}(s) = \frac{K_{VCO}}{s}$

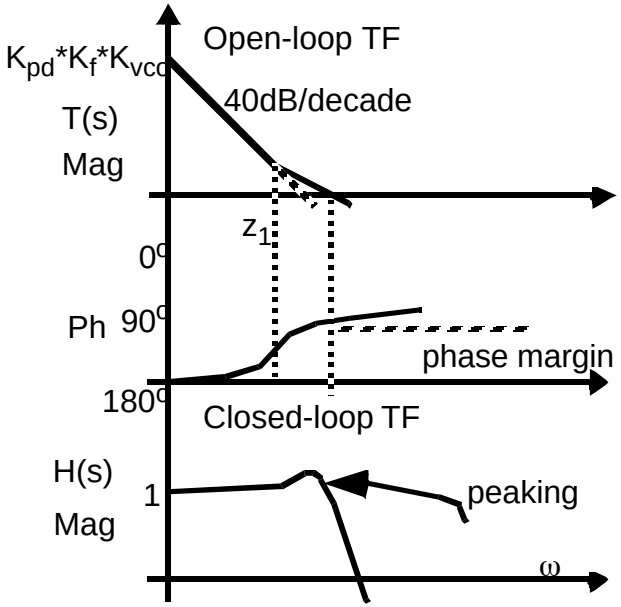
- The extra VCO pole needs to be compensated by a zero in the loop filter:

$$F(s) = \frac{K_f(1 + s/z_1)}{s}$$

PLL Dynamics

- Open Loop TF: $T(s) = \dfrac{K_{pd}K_f(1 + s/z_1)K_{vco}}{s^2}$

- Closed loop TF: $H(s) = \dfrac{K_{pd}K_fK_{vco}(1 + s/z_1)}{s^2 + K_{pd}K_fK_{vco}(1 + s/z_1)}$

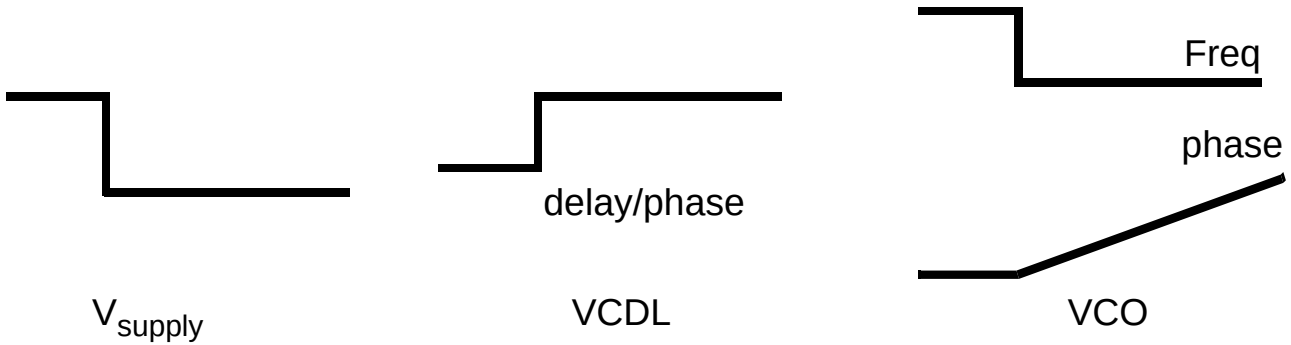


i.e: we are adding proportional control (z_1) to adjust the output phase while the filter integrator (pole at 1/s) holds the frequency information

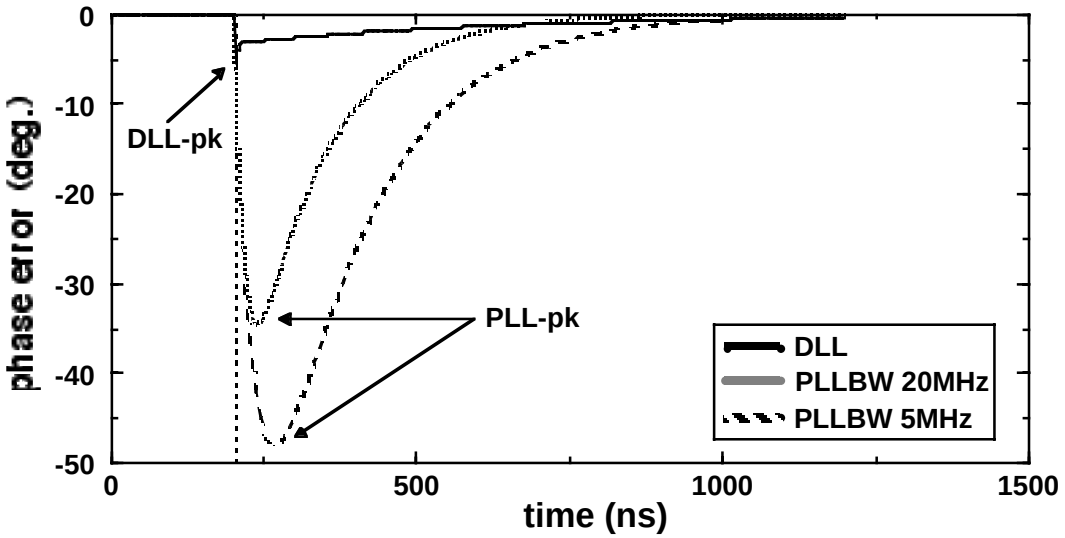
PLL Dynamics (cont'd)

- Other effects that reduce PLL stability/performance [14]:
  - Higher order poles:
    - Suppress ripple but may compromise phase margin
  - Sampled nature of the feedback system
    - Keep $\omega_{bw} < \omega_{ref}/10$
- Ultimately limits the lock range of the loop

- Phase error accumulation (VCO is an integrator i.e.: $\theta = \int \omega dt$):

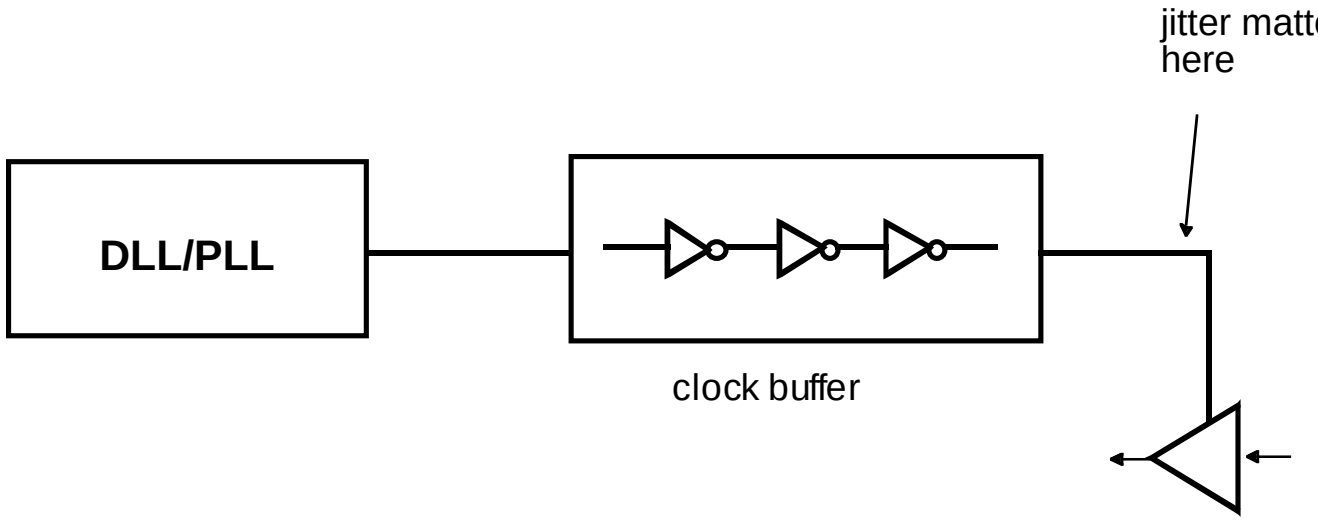


PLL vs DLL: Phase Error Accumulation



- Simulated data for 6-stage PLL vs 6-stage DLL @250MHz:
  - supply sens: 20ps/element/Volt, supply-step: 300-mV @200-ns
- This would suggest that if no clock multiplication is needed and the input clock is quiet, the obvious choice is a DLL. However:
  - Multiplication is often necessary from a system stand-point (EMI, clock generator chips not fast enough)
  - Jitter really matters on the pins...

System Jitter



- A lot of energy is usually spent optimizing half of the problem:
  - A “state of the art” inverter has a supply sensitivity of ~ 1%-delay/%-supply
  - An average PLL/DLL has a supply sensitivity of < 1%-delay/%-supply
- -> If the clock buffer delay approaches a cycle, more than half of the system jitter comes from that buffer....

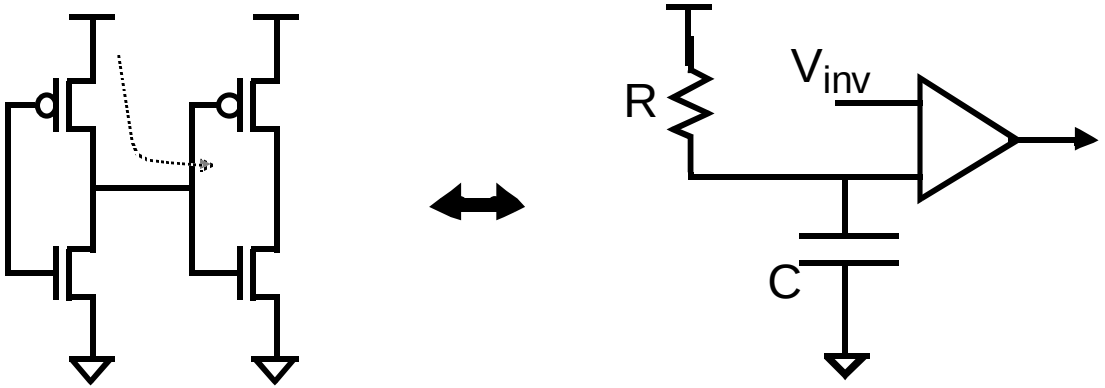
Loop Components

- Variable delay/frequency generators
  - Mainly built as voltage controlled delay elements
    - Main issue is supply/substrate voltage sensitivity
- Phase detectors
  - Linear and non-linear designs depending on the system
    - Main goal is to achieve low offset
- Loop filters
  - Almost always constructed around a charge pump
    - Main issue is to minimize offset and ripple

- Other: Signal amplifiers, Supply de-coupling

Variable delay elements

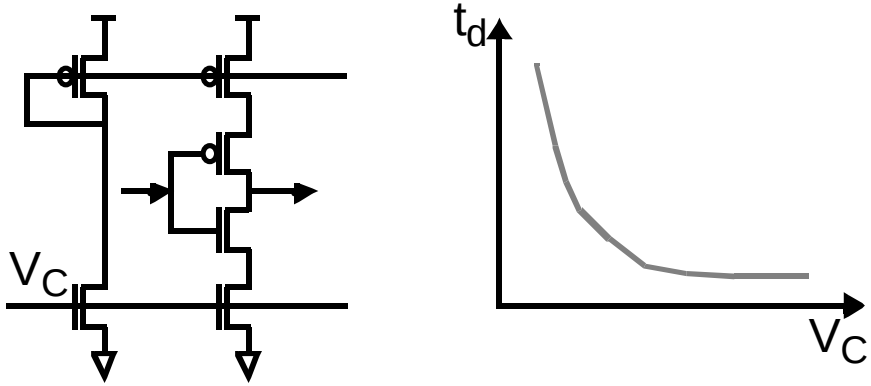
- Delays in CMOS are usually generated by RC elements. e.g.:



- Delay can be controlled by varying R (or I), C, or Vinv.
- All of the above can be changed easily, but the problem is that they also change with varying Process, Temperature, and Supply voltage:
  - Process - Usually not a problem if the total variation is reasonable
  - Temperature - Slowly varying -> well below the loop bandwidth
  - Voltage - Both supply and substrate change rapidly
- **Design Goals:** high supply & substrate noise rejection; adequate range

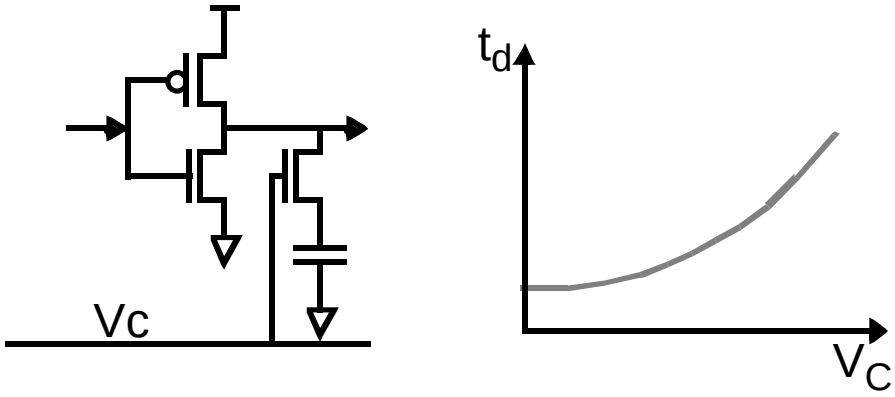
Simple delay elements

- "Current starved" inverter [15]



controls delay by limiting maximum current through a standard inverter

- "Shunt capacitor" inverter [16]

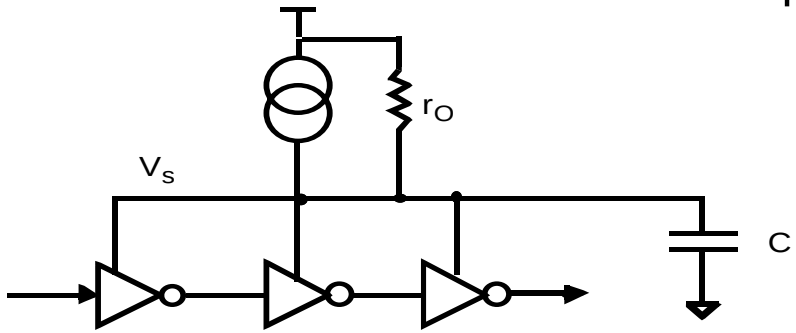


controls delay by changing effective capacitance at the output node

- Both have poor supply rejection (>= 1%delay/1%supply)

Improving simple delay elements

- Make a high impedance current source to isolate completely the inverter supply

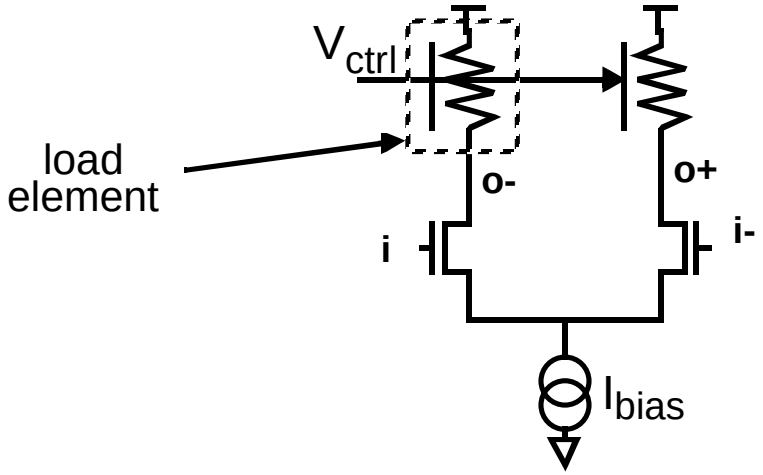


- Vs tracks Vss for frequencies higher than $1/r_OC$
- Current source can be implemented as cascode or “active-cascode” [18]
- Or you can use a source follower to achieve a similar effect [17]
- What about substrate noise?
  - DEC makes ground and substrate the same node by supplying current to the inverters through the p-epi!! [18]
    - OK, as long as max drop through epi resistor is small and constant

Differential delay elements

1. Isolate supply with a high impedance current source
2. Increase CMRR by making signals self-referenced

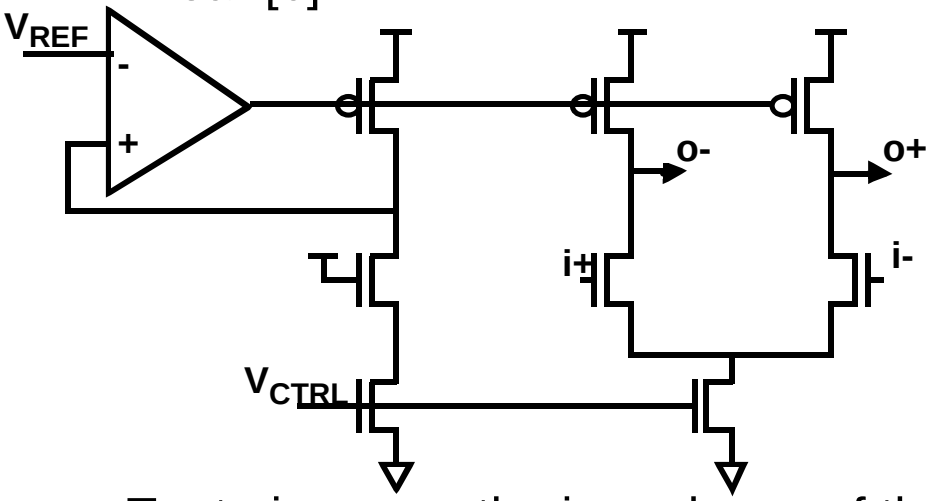
- Ideally we need load elements that look like perfect resistors whose value is adjusted by the control voltage/current:



- Main problem: Create a “resistor” that is both variable AND linear

Differential delay elements with linear loads

- Change the current through a linear-FET-loaded diff pair and adjust the gate voltage of the loads to change the resistance.
  - Replica-feedback circuit keeps swing constant and the loads linear [6]



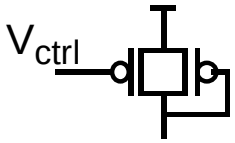
Swing must be quite small to get a real "resistive" behavior

Even then transients might slightly saturate the loads and decrease CMRR

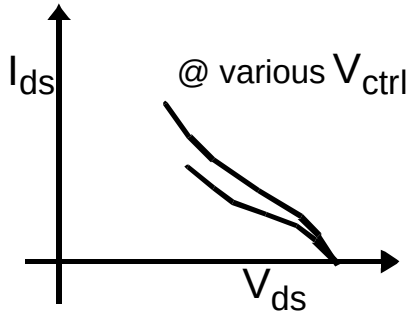
  - Try to increase the impedance of the current source by cascoding
    - -> small head-room
  - Use a load with larger dynamic range [20]

A variable load with high CMRR

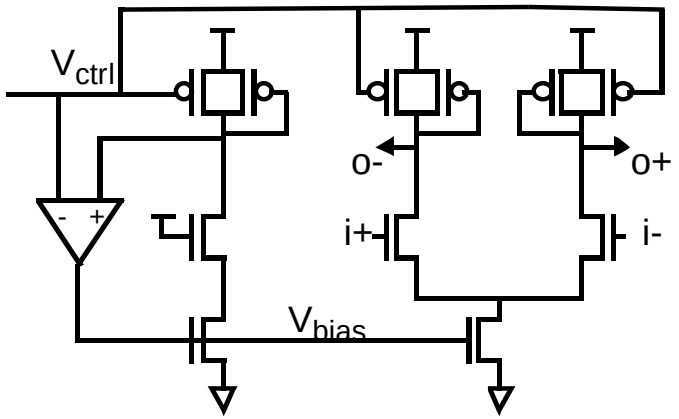
- FET's are non-linear but what we really need is to clamp the swing. Also if load transfer function is symmetric CMRR is improved [19]



Sum of transistor I-V
and diode-connected I-V curve



- Use replica feedback biasing to cancel substrate and supply noise



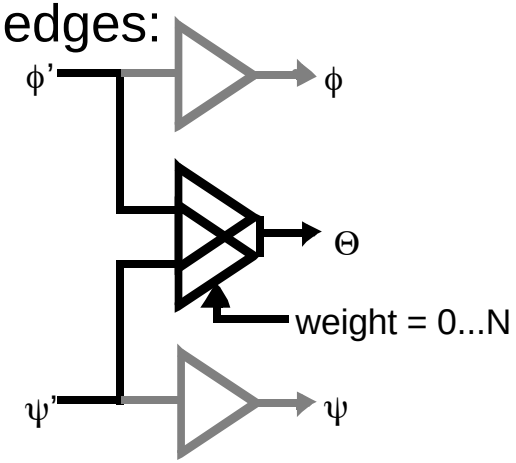
Replica loop keeps the swing of the buffers equal to Vctrl

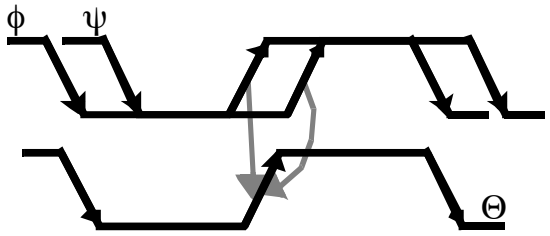
In VCO's the replica loop bandwidth should be high enough to stop the VCO from accumulating phase error for many cycles

Watch out for loop stability

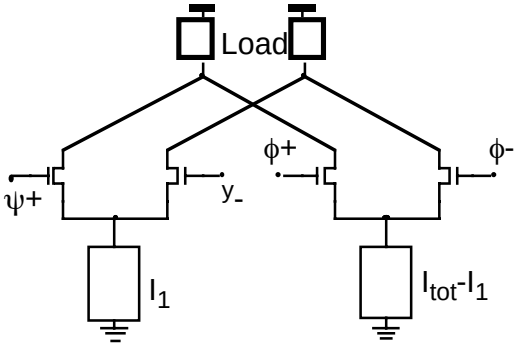
Interpolative Delay Generation

- Generate an edge based on the weighted sum of two other edges:





$\Theta = [(N\text{-weight}) \times \phi + \text{weight} \times \psi]/N$

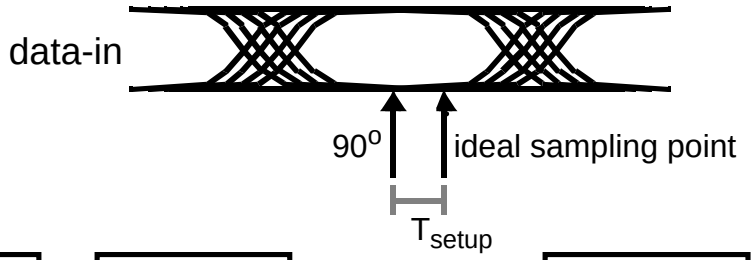


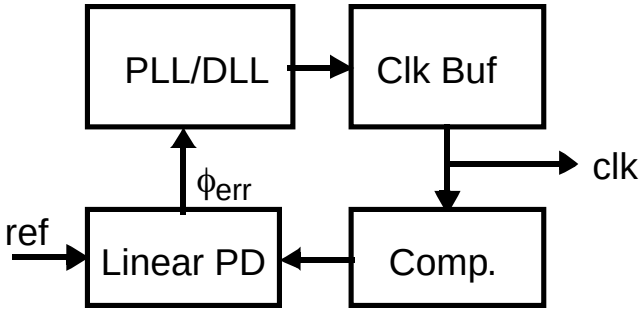
- Useful for:
  - DLL's with unlimited phase shift
  - Fine edge placement for oversampled CRCs
  - Some designs use this technique even for their main VCO [4]
- Non-linearity might be large if input edges are spaced far apart relative to the time constant at the output node

Courtesy D. Foty

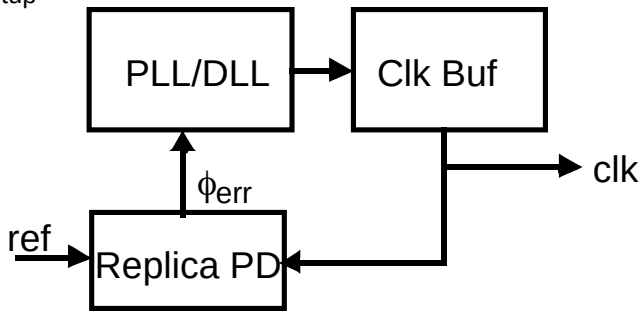
Phase Detectors

- Goal: Align the clock to the right place (e.g. the center of the data eye)





Use "perfect" PD and compensate set-up time

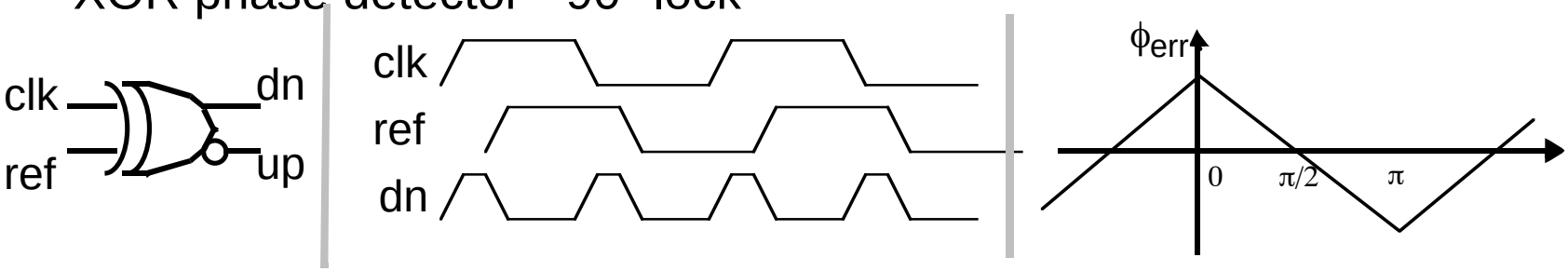


Use "replica" PD to auto-compensate [13]

- Other requirements:
  - Fast acquisition
  - Minimum "dead-band" (i.e. area in which the PD is "blind" to its inputs)

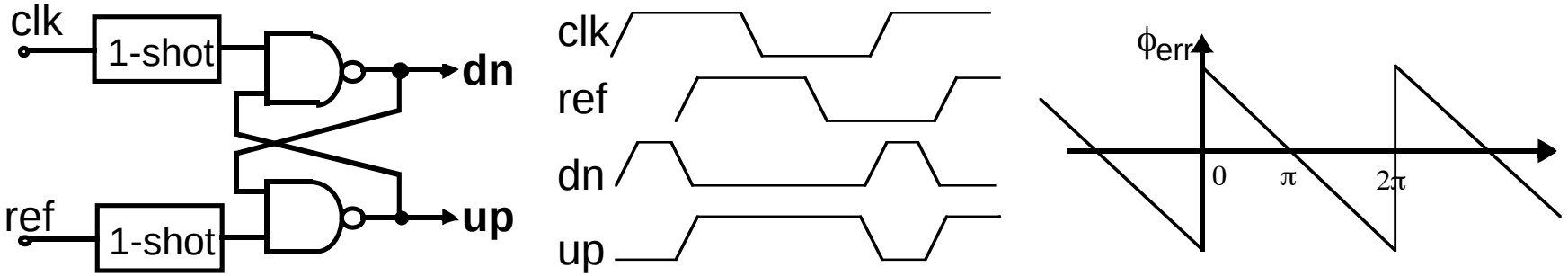
Linear Phase Detectors

- XOR phase detector - 90^o lock



  - sensitive to input duty cycle

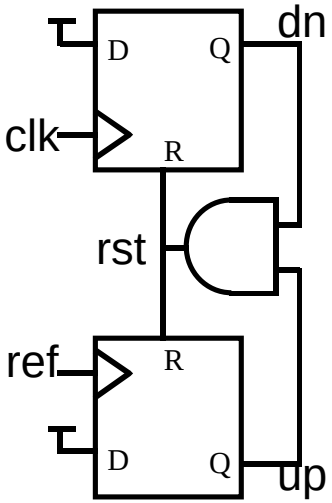
- SR phase detector - 180^o lock

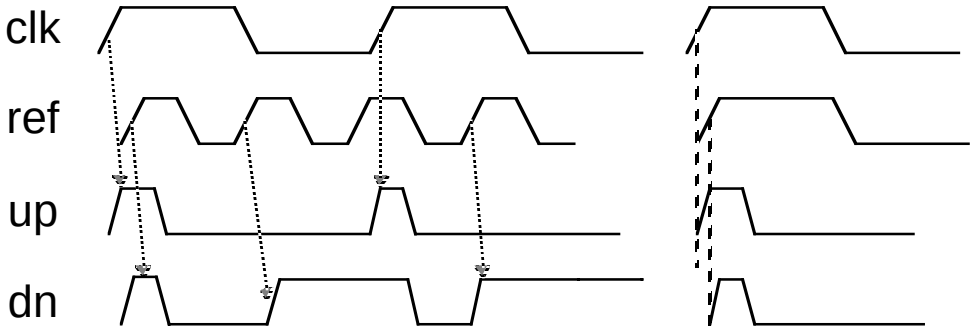


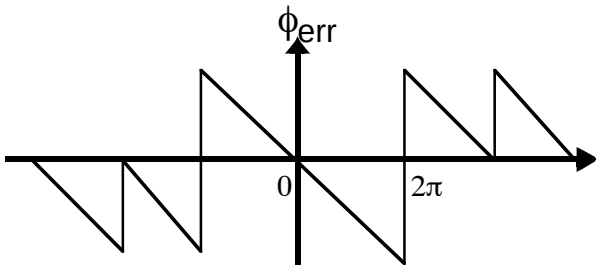
  - 1-shots remove duty cycle sensitivity

Phase/Frequency Detector

- Aids in frequency acquisition



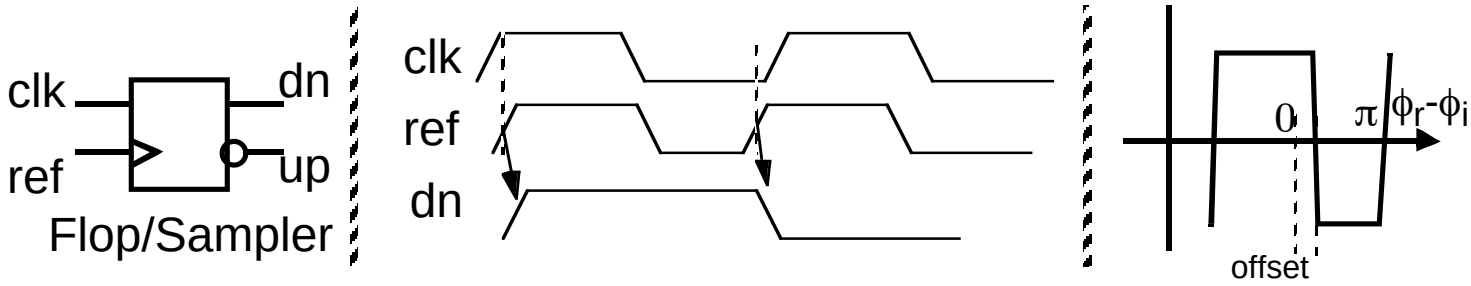




- Overlap up/dn pulses to eliminate dead-band
- Can implement flip/flops in various ways to maximize speed/operating frequency [18]-[20]

Non-linear Phase Detector

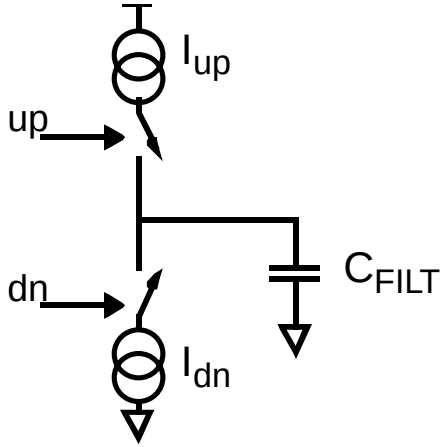
- An ideal flip/flop should force a loop to lock at 0^o



- The set-up time of the flip-flop will introduce phase offset
  - Symmetric structures can eliminate this problem [16]
  - Can be used to cancel the set-up time of an input sampler [13]
- The loop dynamics change:
  - The loop is now a “bang-bang” system which dithers around a locking point:
    - Risky for a PLL, routinely done for DLL’s.
    - The dither magnitude depends on the delay through the loop and the “loop-gain”

Typical DLL Loop Filter

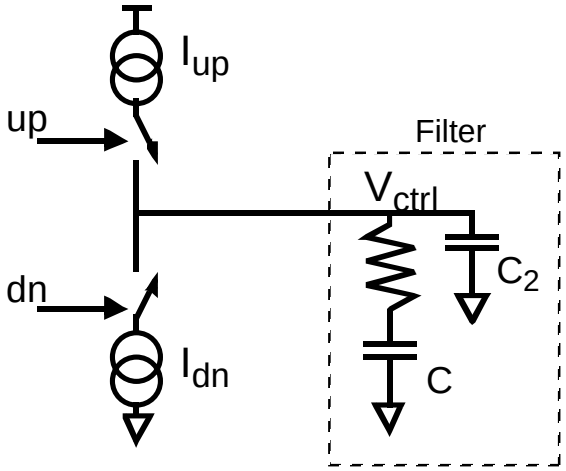
- A “charge-pump” acting a perfect integrator i.e. $K_f = I_{pmp}/C_{filt}/s$.



- We could have static phase offset if I_{up} is not equal to I_{dn}
  - Does not matter much for a “bang-bang” loop
  - Can be an important issue in linear PD-based loops
    - Current sources with high output impedance (cascoding)
    - Differential charge pumps [23]
    - Replica-feedback biased charge pumps [19]

Typical 2nd/3rd order PLL filter

- For a VCO based PLL, insert a resistor in series with integrating C.
  - Explicit C_2 is often used to suppress ripple.

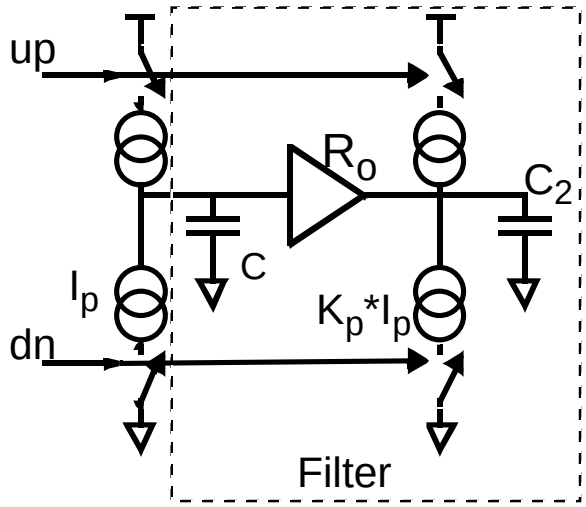


- Implement resistor in high resistivity layer (poly, diffusion, well)
- Difficult: layer might not exist, or ρ might not be well controlled, or have high TC

PLL filter without “real” resistors

- Sum a *proportional current* to an integrated current [21]

  i.e: $I_p/(sC) + K_p I_p R_o$



- If Ro and Ip are scaled appropriately, we can achieve scaling of the loop bandwidth with operating frequency [19]

Conclusion

- Timing/Phase-alignment circuits are crucial in system interconnect performance
- **The good news**: No black magic required!!
  - With many architectures you only have to use basic control theory
- **The bad news**: A lot of opportunities to make a mistake
  - Noise is your worst enemy but lots of techniques exist to alleviate it
- **The challenge**: Make the right trade-offs early and optimize what matters for your system

Clocking References

1. S. Y. Sun, “An Analog PLL-Based Clock and Data Recover Circuit with High Input Jitter Tolerance”, JSSC, April 1989
2. T. Lee and J Bulzacchelli, “A 155-MHZ Clock Recovery Delay and Phase Locked Loop”, JSSC, December 1992
3. L. DeVito, “A Versatile Clock Recovery Architecture and its Monolithic Implementation”, in “Monolithic PLL and CRC Circuits” Behzad Razavi Ed.
4. B. Lai, and R. Walker, “A Monolithic 622Mb/s Clock Extraction Data Retiming Circuit”, ISSCC, February 1991
5. M. Banu and A. Dunlop, “A 660Mb/s CMOS Clock CRC with Instantaneous Locking for NRZ Data and Burst-Mode Transmission”, ISSCC February 1993.
6. B. Kim et. al. “A 30-MHZ Hybrid Analog/Digital RCR in 2-um CMOS,”, IEEE JSSC, December 1990
7. T. Hu, “A Monolithic 480 Mb/s Parallel AGC/Decision/Clock-Recovery Circuits in 1.2um CMOS”, JSSC, Dec. 1993.
8. C.K. Yang et. al., “A 0.5-um CMOS 4Gb/s transceiver with data recovery using oversampling”, JSSC May 1998
9. K. Lee, “A CMOS serial link for fully duplexed data communication”, JSSC, April 1995
10. R. Mooney et. al., “A 900 Mb/s bidirectional signalling interface”, JSSC, Dec 1995
11. T. Takahashi, “A CMOS gate array with 600 Mb/s simultaneous bidirectional I/O circuits”, JSSC, Dec 1995a

Clocking References

12. S. Sidiropoulos and M. Horowitz. “A 700 Mbps/pin CMOS signalling interface using current integrating receivers,” JSSC, May 1997
13. M. Horowitz et. al., “PLL Design for a 500 MB/s Interface,” ISSCC-93
14. F. Gardner, “Phaselock techniques,” 2nd ed. John Wiley, 1979
15. D.K. Jeong, “Design of PLL-based clock generation circuits,” JSSC, April 1987
16. M. Johnson and E. Hudson, “A Variable Delay Line PLL for CPU-Coprocessor Synchronization”, JSSC, October 1988
17. D. Draper, “Circuit techniques in a 266-MHz MMX-enabled processor”, JSSC, Nov. 1997
18. V. vonKaenel, “A 320 MHz, 1.5 mW@1.35 V CMOS PLL for microprocessor clock generation”, JSSC, Nov 1996.
19. J. Maneatis, “Low-jitter process-independent DLL and PLL based on self-biased techniques”, JSSC, Nov. 1996
20. I. Young, “A PLL Clock Generator with 5 to 110 MHZ locking range for Microprocessors”, JSSC, November 1992
21. I. Novoff, “Fully integrated CMOS PLL with 15-200 MHZ range and +/- 50-ps jitter,”, JSSC, Nov. 1995.
22. S. Sidiropoulos, “A semi-digital Delay Locked Loop,” JSSC, December 1997
23. T. Lee, et. al. “A 2.5V CMOS Delay-Locked Loop for an 18Mbit, 500 MByte/s DRAM”, JSSC, Dec 1994