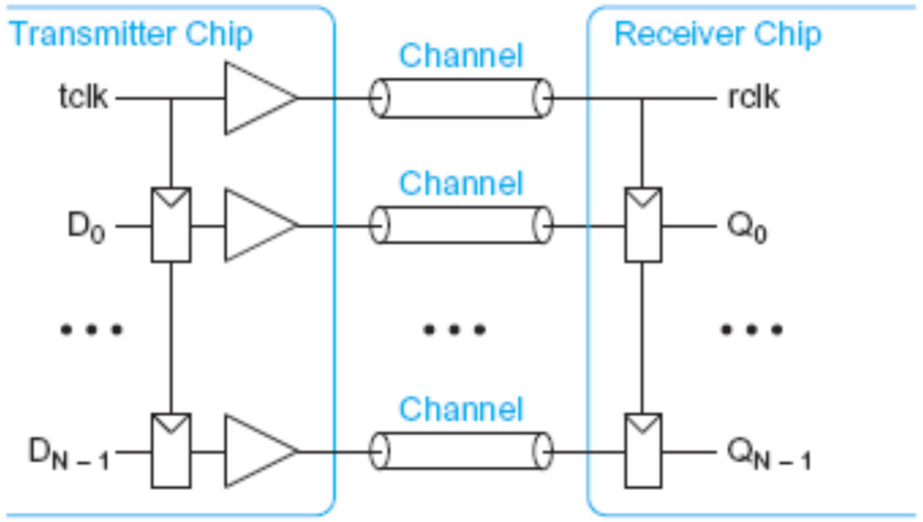
Chap. 12: Clocking, PLLs and DLLs

Outline

- Clock Recovery
- Clock System Architecture
- Phase-Locked Loops
- Delay-Locked Loops
- Clock Distribution
- High Speed Links
- Synchronizer

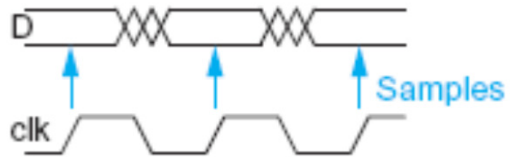
Source-Synchronous Clocking

- Send clock with the data
- Flight times roughly match each other
  - Transmit on falling edge of tclk
  - Receive on rising edge of rclk

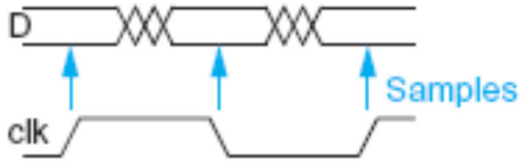


Single vs. Double Data Rate

- In ordinary single data rate (SDR) system, clock switches twice as often as the data

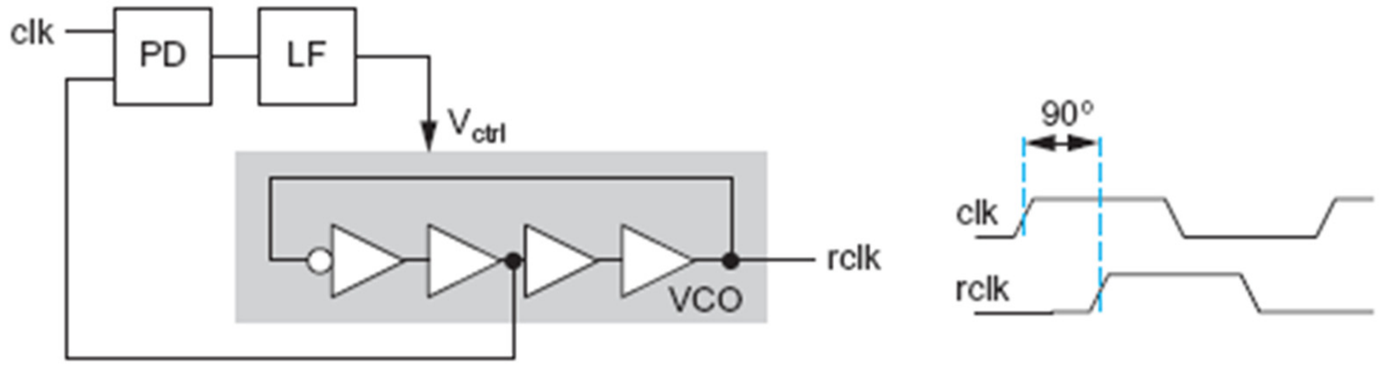


- If the system can handle this speed clock, the data is running at half the available bandwidth
- In double-data-rate (DDR) transmit and receive on both edges of the clock



Phase Alignment

- If the DDR clock is aligned to the transmitted clock, it must be shifted by 90º before sampling
- Use PLL

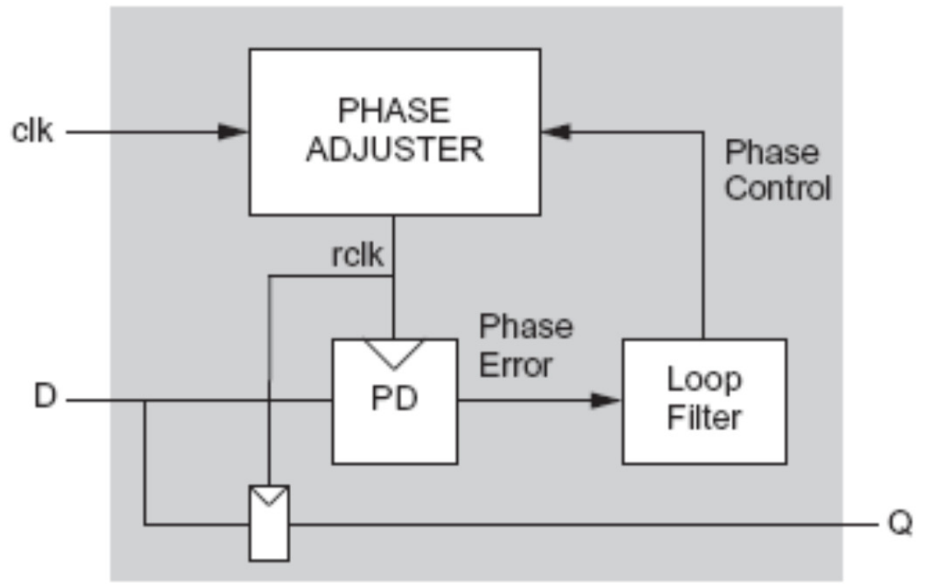


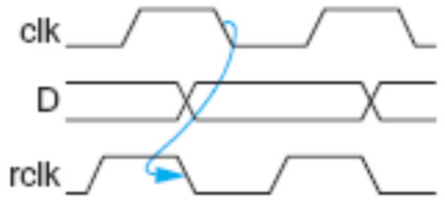
Mesochronous Clocking

- As speeds increase, it is difficult to keep clock and data aligned
  - Mismatches in trace lengths
  - Mismatches in propagation speeds
  - Different in clock vs. data drivers
- Mesochronous: clock and data have same frequency but unknown phase
  - Use PLL/DLL to realign clock to each data channel

Phase Calibration Loop

- Special phase detector compares clock & data phase





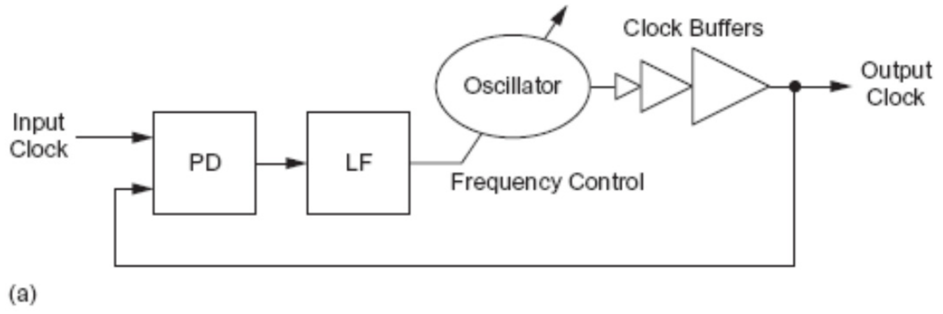
Clock Generation

- ❑ Low frequency:
  - – Buffer input clock and drive to all registers
- ❑ High frequency
  - – Buffer delay introduces large skew relative to input clocks
    - • Makes it difficult to sample input data
  - – Distributing a very fast clock on a PCB is hard

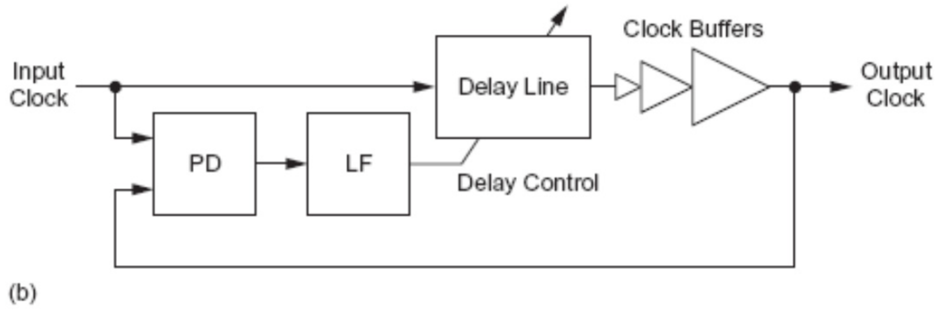
Zero-Delay Buffer

- If the periodic clock is delayed by T_c, it is indistinguishable from the original clock
- Build feedback system to guarantee this delay

Phase-Locked Loop (PLL)

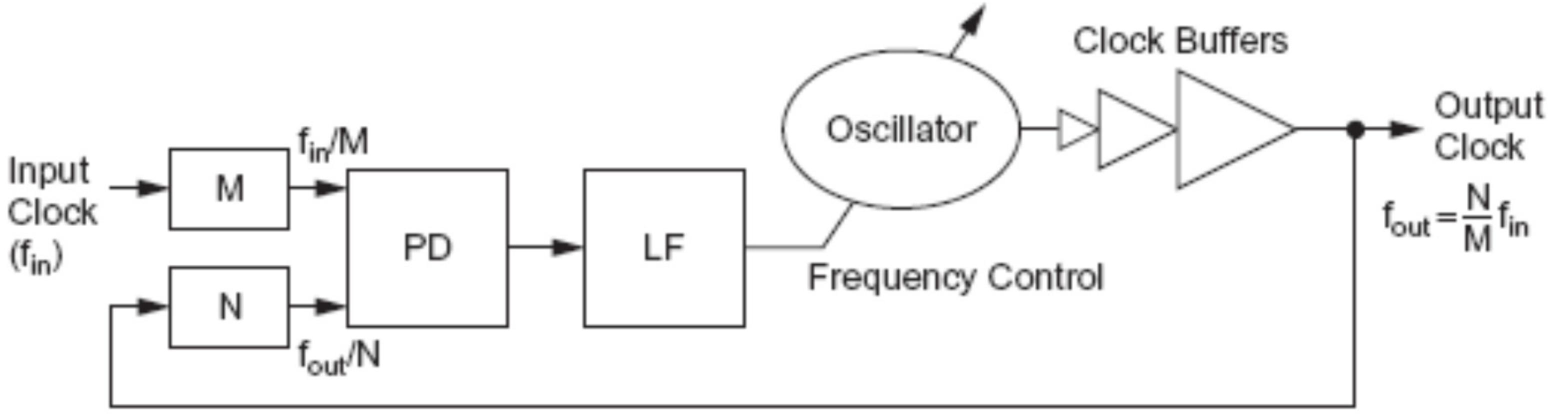


Delay-Locked Loop (PLL)



Frequency Multiplication

- PLLs can multiply the clock frequency

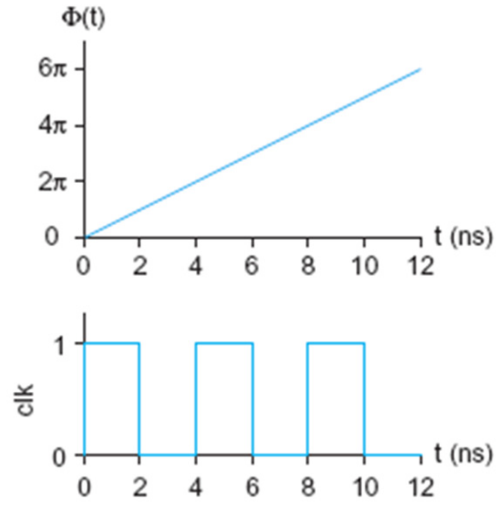


Phase and Frequency

- Analyze PLLs and DLLs in term of phase Φ(t) rather than voltage v(t)

$$\text{clk} = \begin{cases} 1 & \Phi(t) \bmod 2\pi < \pi \\ 0 & \Phi(t) \bmod 2\pi \geq \pi \end{cases}$$

$$\Phi(t) = 2\pi \int_0^t f(t)dt$$



- Input and output clocks may deviate from locked phase
  - Small signal analysis

$$\Phi_{\text{in}}(t) = \Phi(t) + \Delta\Phi_{\text{in}}(t)$$

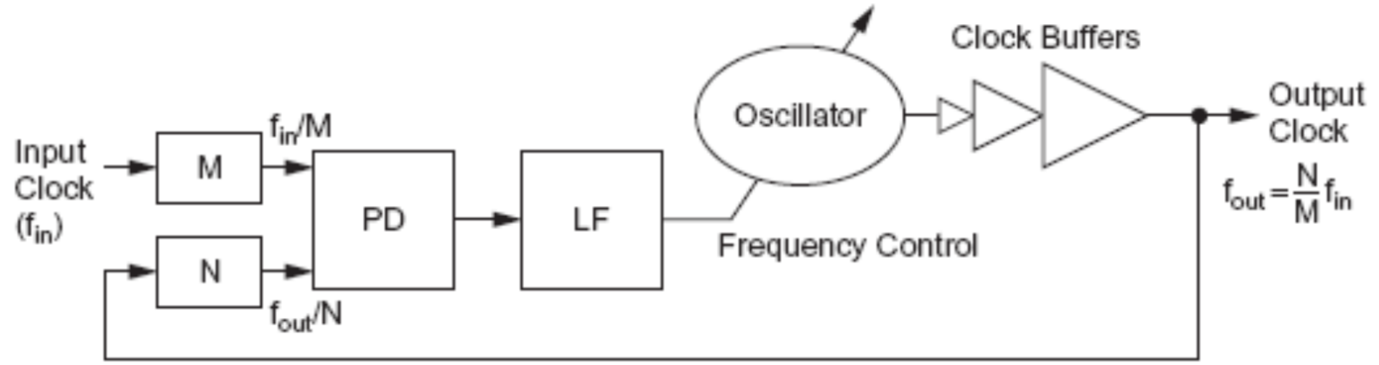
$$\Phi_{\text{out}}(t) = N\Phi(t) + \Delta\Phi_{\text{out}}(t)$$

Linear System Model

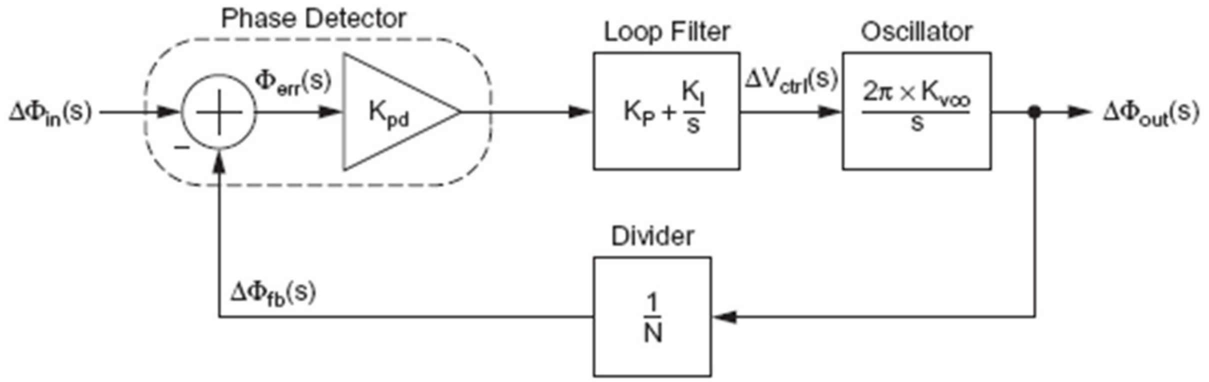
- Treat PLL/DLL as a linear system
  - Compute deviation of desired freq. from locked position
  - Assume small deviations from locked
  - Treat system as linear for these small changes
- Analysis is not valid far from lock
  - e.g. during acquisition at startup
- Continuous time assumption
  - PLL/DLL is really a discrete time system
    - Updates once per cycle
  - If the bandwidth << 1/10 clock freq, treat as continuous
- Use Laplace transforms and standard analysis of linear continuous-time feedback control systems

Phase-Locked Loop (PLL)

- System

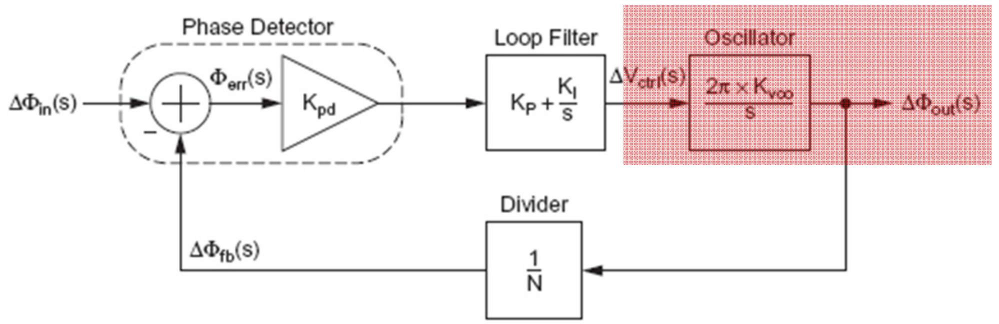


- Linear Model



Voltage-Controlled Oscillator

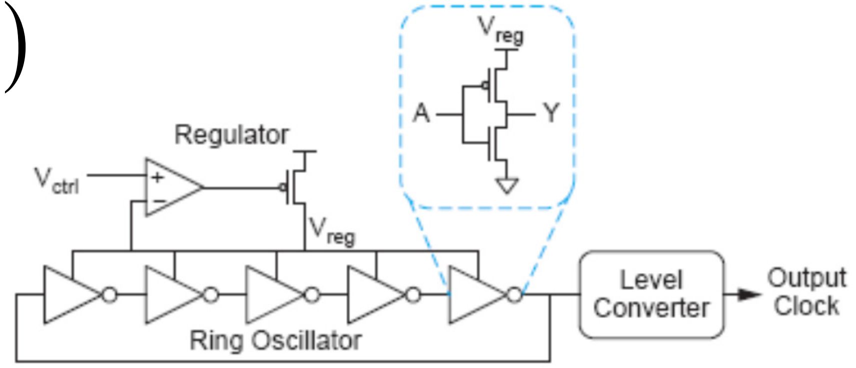
- VCO

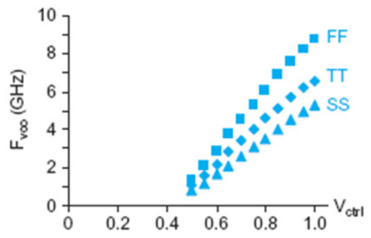


$$V_{\text{ctrl}}(t) = V_{\text{ctrl0}} + \Delta V_{\text{ctrl}}(t)$$

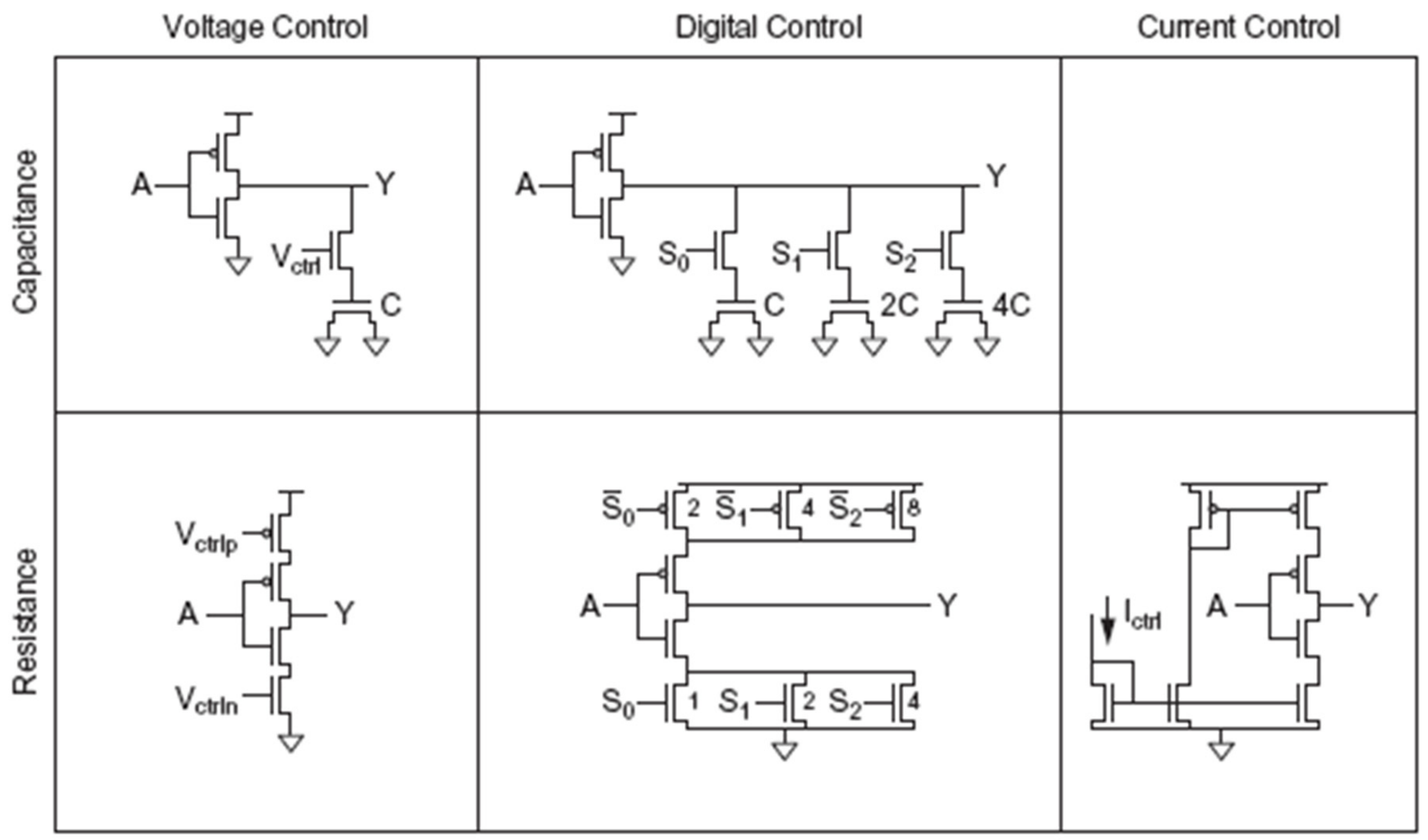
$$\frac{\Delta f_{out}}{\Delta V_{\text{ctrl}}} = K_{vco}$$

$$\frac{\Delta\Phi_{out}(s)}{\Delta V_{ctrl}(s)} = \frac{2\pi K_{vco}}{s}$$



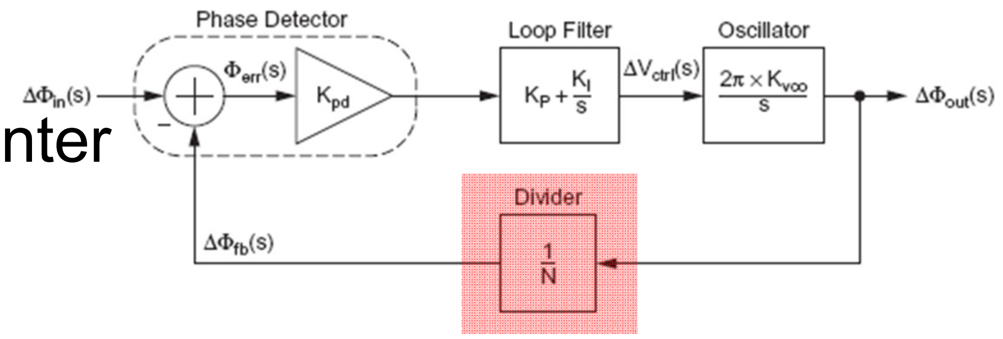


Alternative Delay Elements



Frequency Divider

- Divide clock by N
  - Use mod-N counter

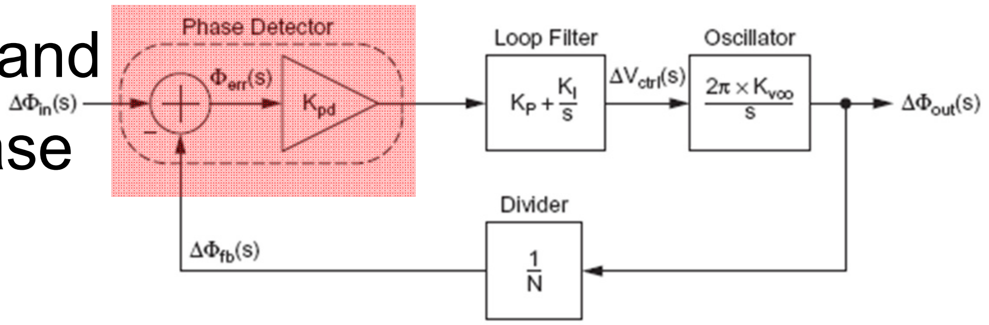


$$\Delta f_{\text{fb}} = \frac{\Delta f_{\text{out}}}{N}$$

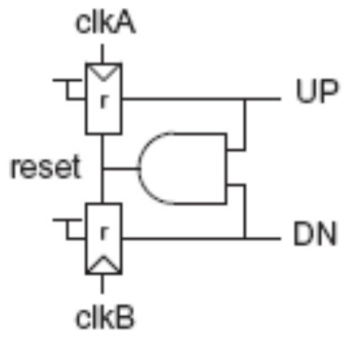
$$\Delta\Phi_{\text{fb}} = \frac{\Delta\Phi_{\text{out}}}{N}$$

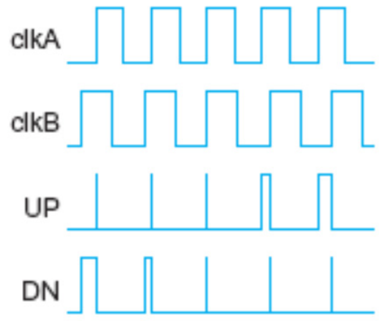
Phase Detector

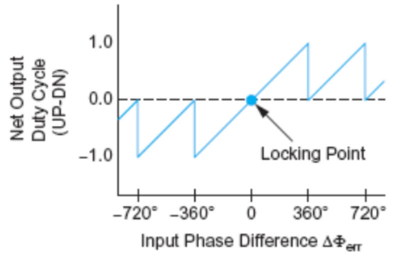
- Difference of input and feedback clock phase



- Often built from phase-frequency detector (PFD)



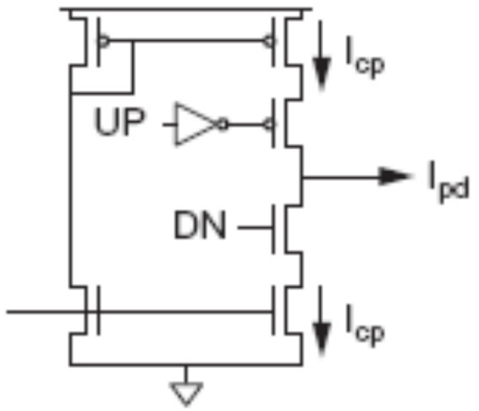




Phase Detector

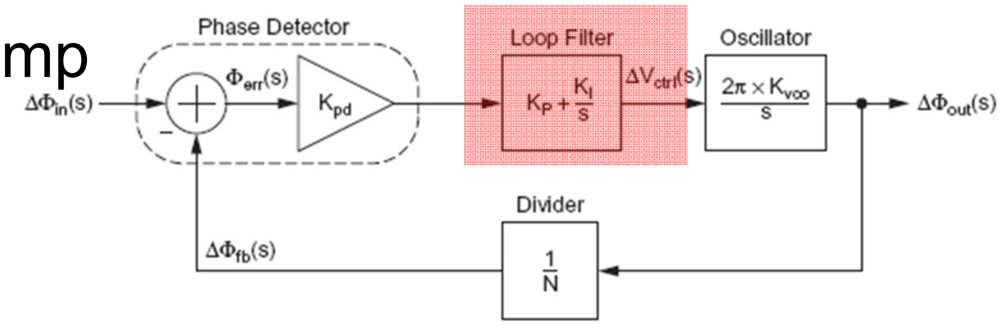
- Convert up and down pulses into current proportional to phase error using a charge pump

$$\frac{I_{pd}(s)}{\Phi_{\mathrm{err}}(s)} = \frac{I_{\mathrm{cp}}}{2\pi} = K_{pd}$$



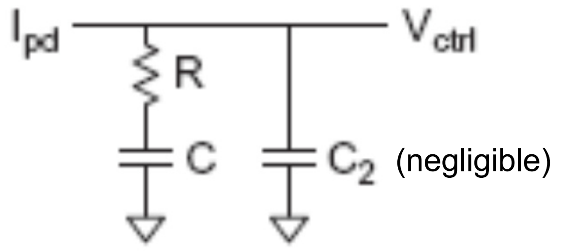
Loop Filter

- Convert charge pump current into V_{ctrl}



- Use proportional-integral control (PI) to generate a control signal dependent on the error and its integral
  - Drives error to 0

$$\frac{V_{\text{ctrl}}(s)}{I_{\text{pd}}(s)} = \frac{1}{sC} + R$$



PLL Loop Dynamics

- Closed loop transfer function of PLL

$$H(s) = \frac{\Delta\Phi_{\text{out}}(s)}{\Delta\Phi_{\text{in}}(s)} = \frac{K_{pd}\left(R + \frac{1}{sC}\right)\frac{2\pi K_{\text{vco}}}{s}}{1 + \frac{1}{N}K_{pd}\left(R + \frac{1}{sC}\right)\frac{2\pi K_{\text{vco}}}{s}}$$

- This is a second order system

$$H(s) = N\frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

$$\omega_n = \sqrt{\frac{I_{cp}K_{\text{vco}}}{NC}}$$

$$\zeta = \frac{\omega_n}{2}RC$$

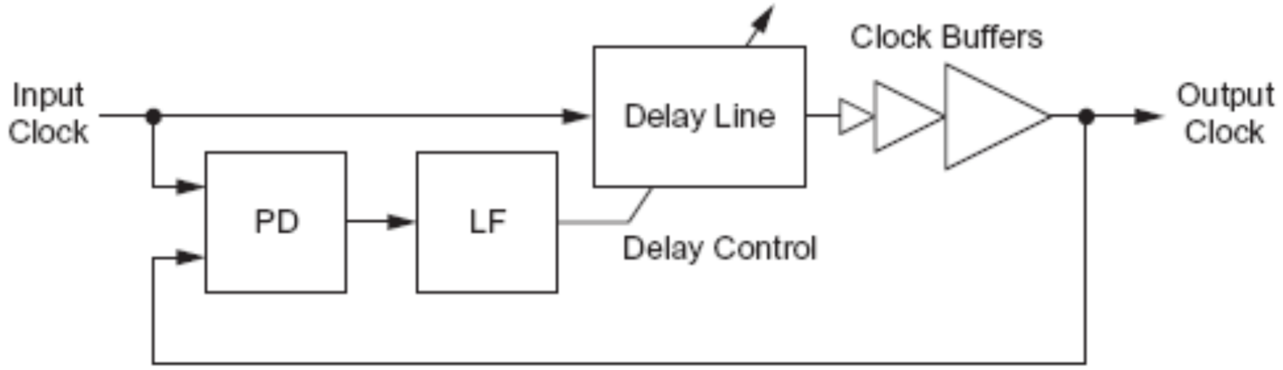
- ω_n indicates loop bandwidth
- ζ indicates damping; choose 0.7 – 1 to avoid ringing

Delay Locked Loop

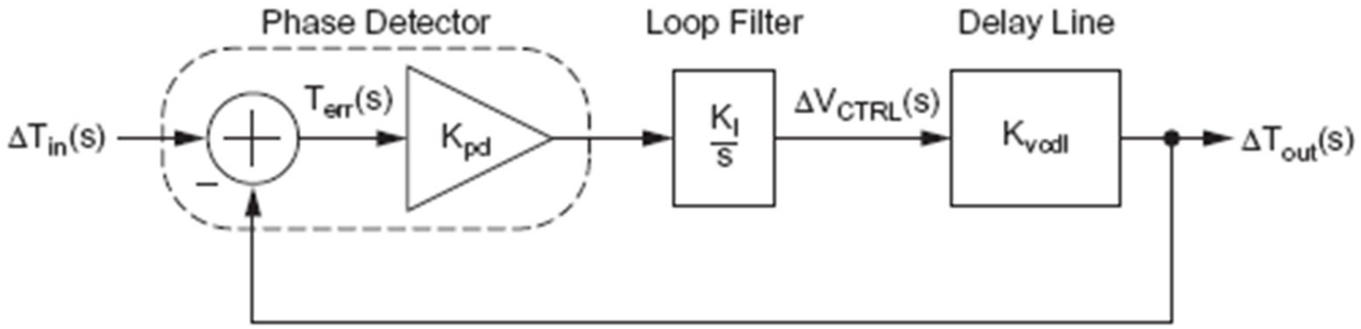
- Delays input clock rather than creating a new clock with an oscillator
- Cannot perform frequency multiplication
- More stable and easier to design
  - 1st order rather than 2nd
- State variable is now time (T)
  - Locks when loop delay is exactly T_c
  - Deviations of ΔT from locked value

Delay-Locked Loop (DLL)

- System

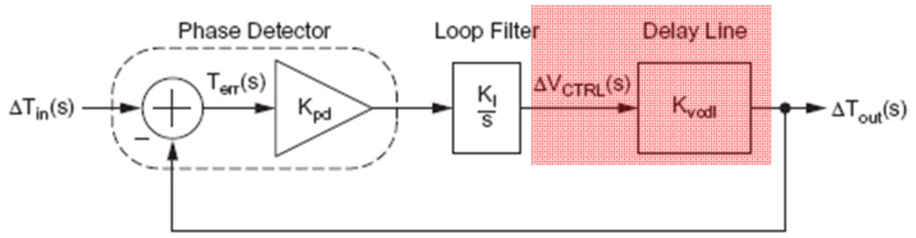


- Linear Model



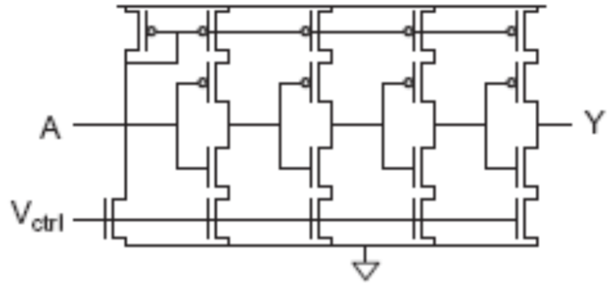
Delay Line

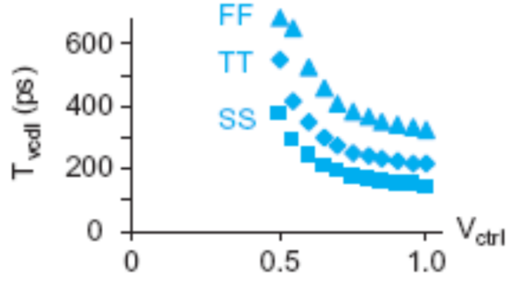
- Delay input clock



- Typically use voltage-controlled delay line

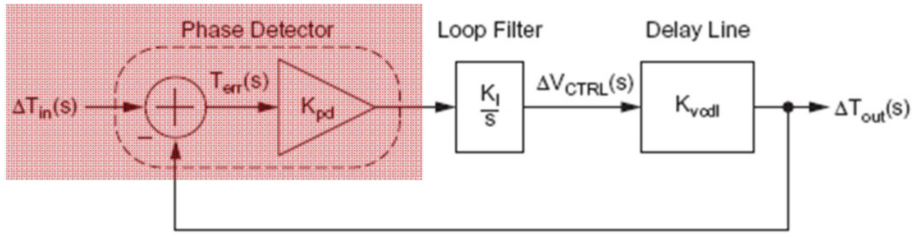
$$\frac{\Delta T_{\text{out}}(s)}{\Delta V_{\text{ctrl}}(s)} = K_{vcdl}$$





Phase Detector

- Detect phase error

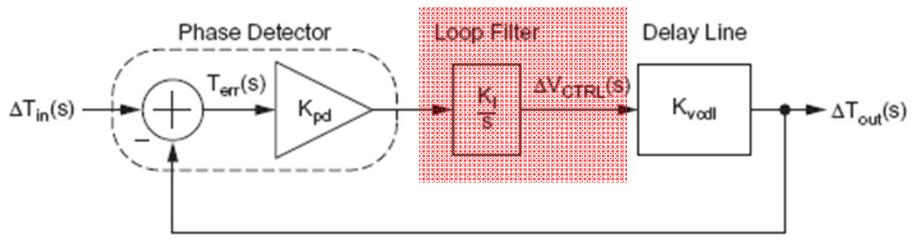


- Typically use PFD and charge pump, as in PLL

$$\frac{I_{pd}(s)}{T_{\text{err}}(s)} = \frac{I_{cp}}{T_c}$$

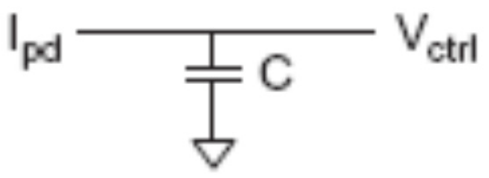
Loop Filter

- Convert error current into control voltage



- Integral control is sufficient
- Typically use a capacitor as the loop filter

$$\frac{\Delta V_{\text{ctrl}}(s)}{I_{pd}(s)} = \frac{K_I}{s} = \frac{1}{sC}$$



DLL Loop Dynamics

- Closed loop transfer function of DLL

$$H(s) = \frac{\Delta T_{\text{out}}(s)}{\Delta T_{\text{in}}(s)} = \frac{1}{s\tau + 1}$$

- This is a first order system

$$\tau = \frac{1}{K_{pd} K_I K_{vcdl}} = \frac{C T_c}{I_{cp} K_{vcdl}}$$

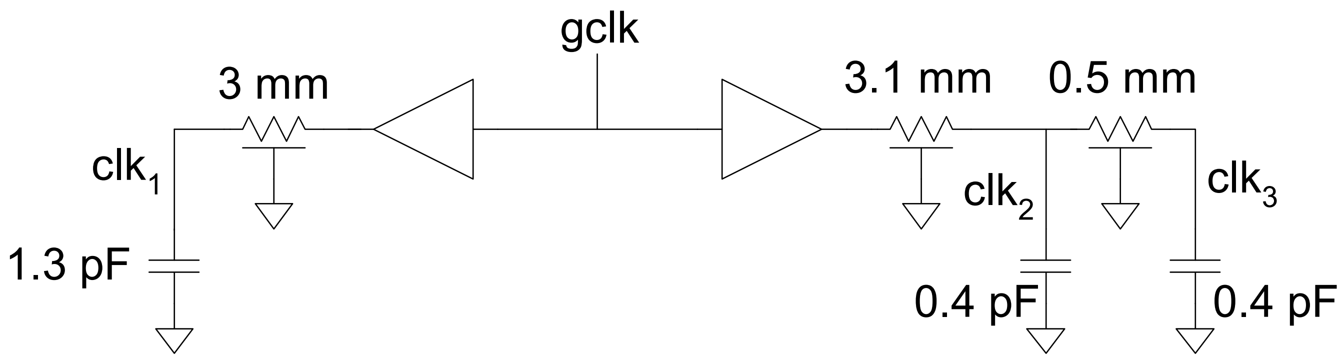
- τ indicates time constant (inverse of bandwidth)
  - Choose at least 10T_c for continuous time approx.

Clock Distribution

- On a small chip, the clock distribution network is just a wire
  - And possibly an inverter for clkb
- On practical chips, the RC delay of the wire resistance and gate load is very long
  - Variations in this delay cause clock to get to different elements at different times
  - This is called *clock skew*
- Most chips use repeaters to buffer the clock and equalize the delay
  - Reduces but doesn't eliminate skew

Example

- Skew comes from differences in gate and wire delay
  - With right buffer sizing, clk_1 and clk_2 could ideally arrive at the same time.
  - But power supply noise changes buffer delays
  - clk_2 and clk_3 will always see RC skew

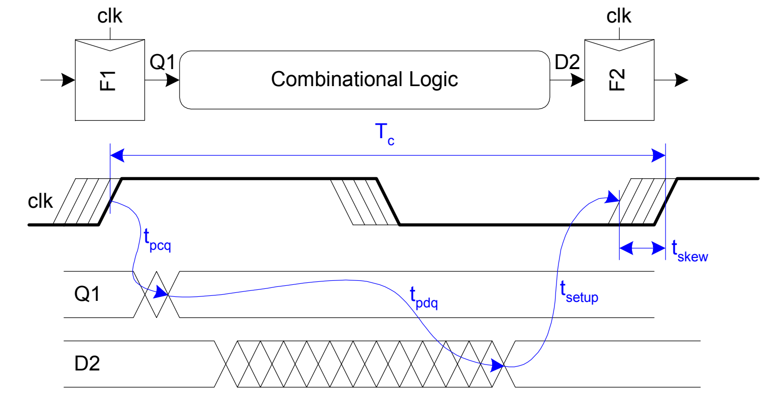


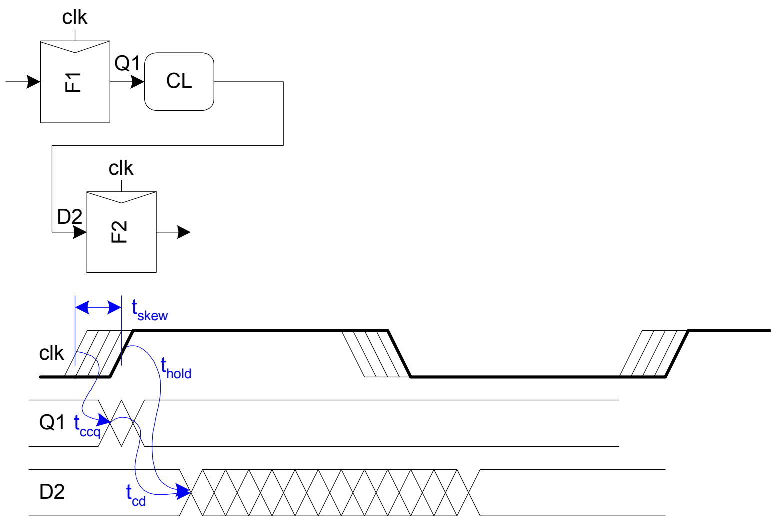
Review: Skew Impact

- Ideally full cycle is available for work
- Skew adds sequencing overhead
- Increases hold time too

$$t_{pd} \le T_c - \underbrace{\left(t_{pcq} + t_{\text{setup}} + t_{\text{skew}}\right)}_{\text{sequencing overhead}}$$

$$t_{cd} \ge t_{\text{hold}} - t_{ccq} + t_{\text{skew}}$$





Solutions

- ❑ Reduce clock skew
  - – Careful clock distribution network design
  - – Plenty of metal wiring resources
- ❑ Analyze clock skew
  - – Only budget actual, not worst case skews
  - – Local vs. global skew budgets
- ❑ Tolerate clock skew
  - – Choose circuit structures insensitive to skew

Clock Dist. Networks

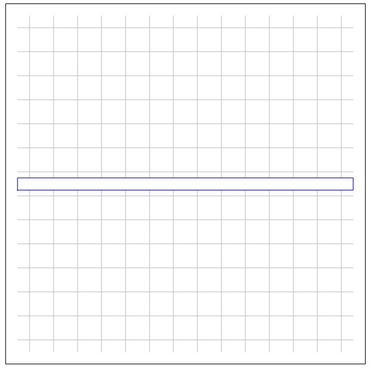
- *Ad hoc*
- Grids
- H-tree
- Hybrid

Clock Grids

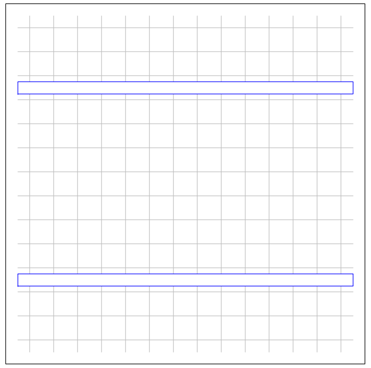
- Use grid on two or more levels to carry clock
- Make wires wide to reduce RC delay
- Ensures low skew between nearby points
- But possibly large skew across die

Alpha Clock Grids

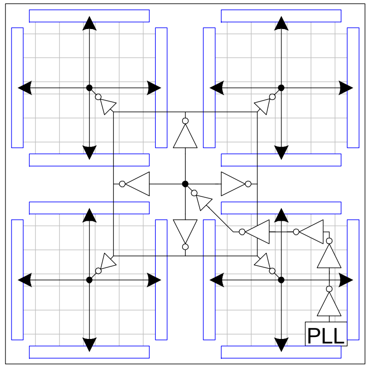
Alpha 21064

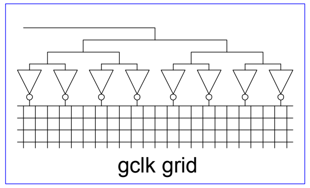


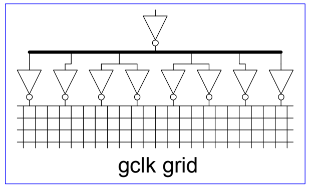
Alpha 21164

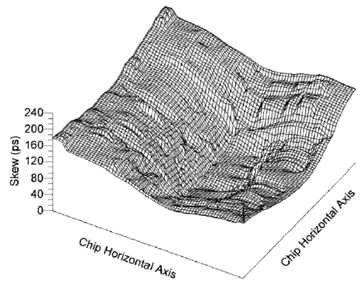


Alpha 21264

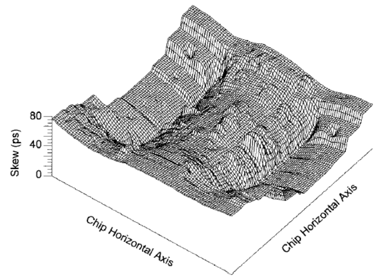




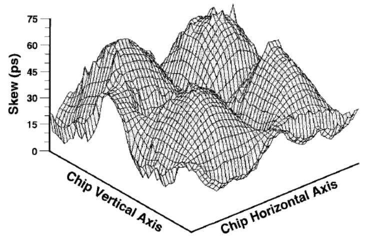




Alpha 21064



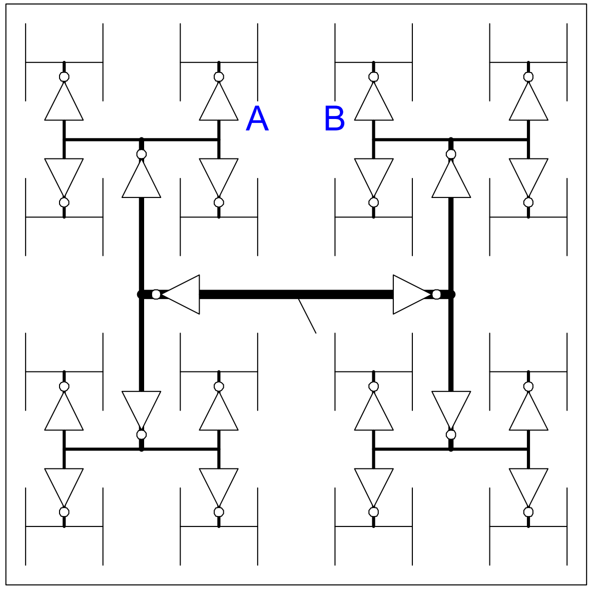
Alpha 21164



Alpha 21264

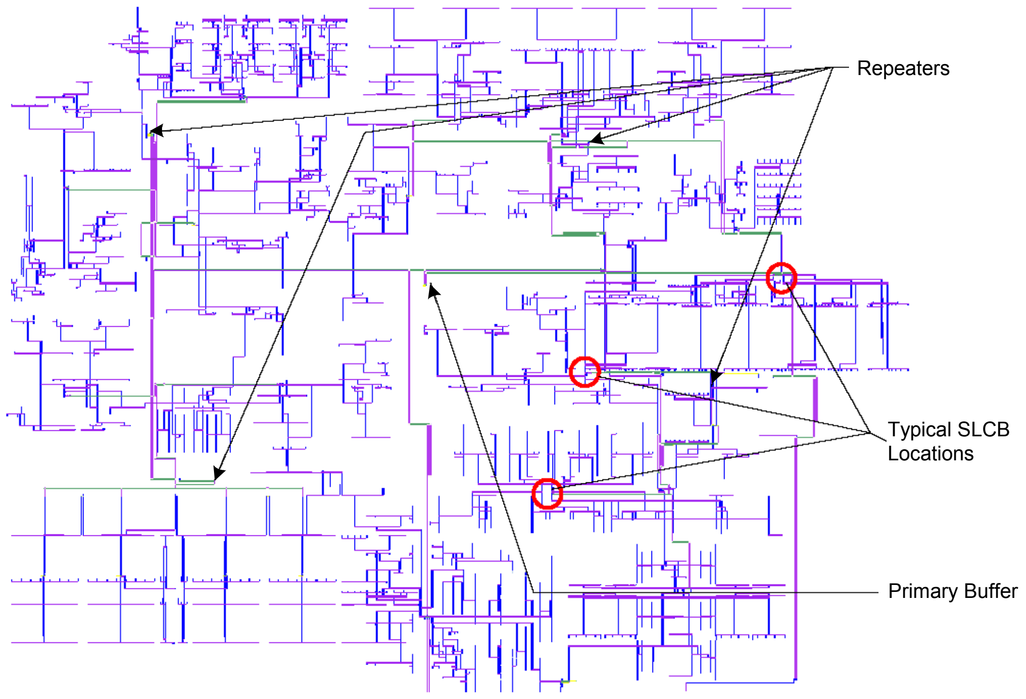
H-Trees

- Fractal structure
  - Gets clock arbitrarily close to any point
  - Matched delay along all paths
- Delay variations cause skew
- A and B might see big skew



Itanium 2 H-Tree

- Four levels of buffering:
  - Primary driver
  - Repeater
  - Second-level clock buffer
  - Gater
- Route around obstructions



Hybrid Networks

- Use H-tree to distribute clock to many points
- Tie these points together with a grid

- Ex: IBM Power4, PowerPC
  - H-tree drives 16-64 sector buffers
  - Buffers drive total of 1024 points
  - All points shorted together with grid

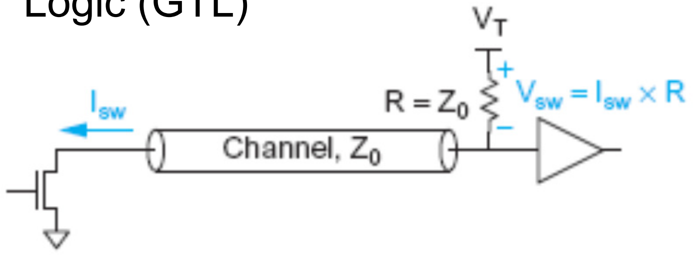
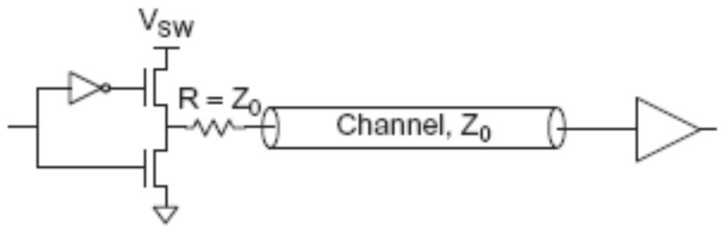
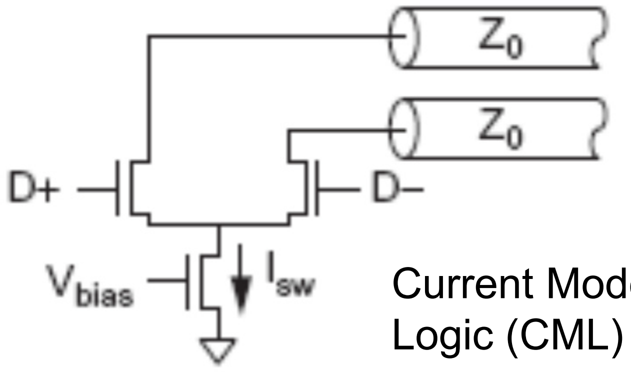
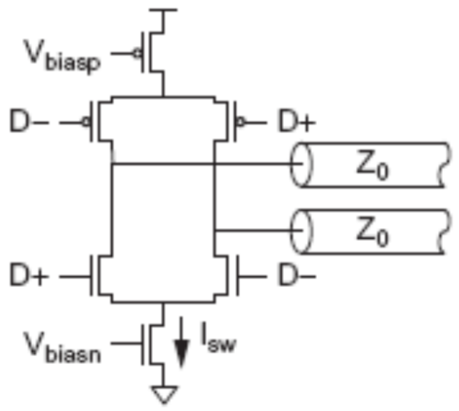
High-Speed I/O

- Transmit data faster than the flight time along the line
- Transmitters must generate very short pulses
- Receivers must be accurately synchronized to detect the pulses

High Speed Transmitters

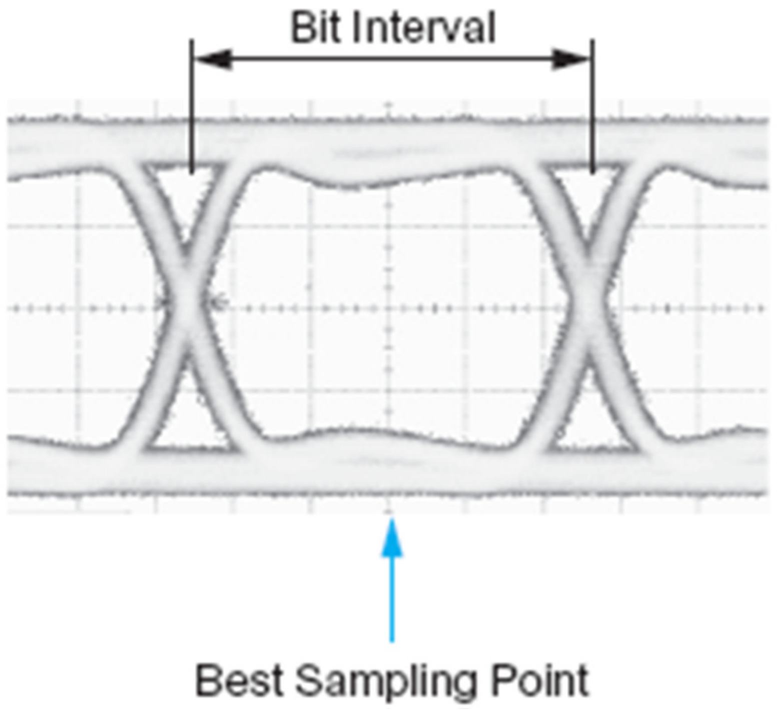
- How to handle termination?
  - High impedance current-mode driver + load term?
  - Or low-impedance driver + source termination
- Single-ended vs. differential
  - Single-ended uses half the wires
  - Differential is Immune to common mode noise
- Pull-only vs. Push-Pull
  - Pull-only has half the transistors
  - Push-pull uses less power for the same swing

High-Speed Transmitters

	Pull-Only	Push-Pull
Single-Ended	Gunning Transceiver Logic (GTL)  V_T + $R = Z_0$ $V_{sw} = I_{sw} \times R$ − I_{sw} Channel, Z_0	
Differential	 Current Mode Logic (CML)	 V_{biasp} D− D+ Z_0 Z_0 D+ D− V_{biasn} I_{sw} Low-Voltage Differential Signalling (LVDS)

High-Speed Receivers

- Sample data in the middle of the bit interval
- How do we know when?


Bit Interval
Best Sampling Point