**Bandwidth Constraints for Integer-N Synthesizers**

- PFD output has a periodicity of $1/T$
  - $1/T = \text{reference frequency}$
- Loop filter must have a bandwidth $\ll 1/T$
  - PFD output pulses must be filtered out and average value extracted

**Closed loop PLL bandwidth often chosen to be a factor of ten lower than 1/T**
**Bandwidth Versus Frequency Resolution**

- **Frequency resolution set by reference frequency** \((1/T)\)
  - Higher resolution achieved by lowering \(1/T\)

\[
\text{frequency resolution} = \frac{1}{T}
\]
Use a reference divider to achieve lower $1/T$
- Leads to a low PLL bandwidth ( < 20 kHz here )
The Issue of Noise

- Lower 1/T leads to higher divide value
  - Increases PFD noise at synthesizer output
Modeling PFD Noise Multiplication

- Influence of PFD noise seen in model from Lecture 16
  - PFD spectral density multiplied by $N^2$ before influencing PLL output phase noise

High divide values $\rightarrow$ high phase noise at low frequencies
Outline Of Talk

- Dual-loop Synthesizers
- Direct Digital Synthesizers
- Fractional-N Synthesizers
  - Traditional Approach
  - Sigma-Delta Concepts
  - Synthesizer Noise Analysis
**Dual-Loop Frequency Synthesizer**

- **Overall synthesizer output**

  \[ \text{out}(t) = \cos(w_1 t) \cos(w_2 t) + \sin(w_1 t) \sin(w_2 t) \]

- **From trigonometry: \( \cos(A-B) = \cos A \cos B + \sin A \sin B \)**

  \[ \Rightarrow \text{out}(t) = \cos((w_1 - w_2)t) \]
Advantage #1: Avoids Large Divide Values

- Choose top synthesizer to provide coarse tuning and bottom synthesizer to provide fine tuning
  - Choose $w_1$ to be high in frequency
    - Set $\text{ref}_1$ to be high to avoid large $N$ → low resolution
  - Choose $w_2$ to be low in frequency
    - Allows $\text{ref}_2$ to be low without large $M$ → high resolution
Advantage #2: Provides Suppression of VCO Noise

- Top VCO has much more phase noise than bottom VCO due to its much higher operating frequency
  - Suppress top VCO noise by choosing a high PLL bandwidth for top synthesizer
    - High PLL bandwidth possible since ref₁ is high
Alternate Dual-Loop Architecture

- Calculation of output frequency

\[ y(t) = \cos((w_1 - w_2)t) \]

\[ \Rightarrow N w_{\text{ref}_1} = w_1 - w_2 \]

\[ \Rightarrow \text{out}(t) = \cos((N w_{\text{ref}_1} + w_2)t) \]
**Advantage of Alternate Dual-Loop Architecture**

- **Issue**: a practical single-sideband mixer implementation will produce a spur at frequency $w_1 + w_2$
- **PLL bandwidth of top synthesizer can be chosen low enough to suppress the single-sideband spur**
  - **Negative**: lower suppression of top VCO noise
Direct Digital Synthesis (DDS)

- Encode sine-wave values in a ROM
- Create sine-wave output by indexing through ROM and feeding its output to a DAC and lowpass filter
  - Speed at which you index through ROM sets frequency of output sine-wave
    - Speed of indexing is set by increment value on counter (which is easily adjustable in a digital manner)
Pros and Cons of Direct Digital Synthesis

- **Advantages**
  - Very fast adjustment of frequency
  - Very high resolution can be achieved
  - Highly digital approach

- **Disadvantages**
  - Difficult to achieve high frequencies
  - Difficult to achieve low noise
  - Power hungry and complex
Hybrid Approach

- Use DDS to create a finely adjustable reference frequency
- Use integer-N synthesizer to multiply the DDS output frequency to much higher values
- Issues
  - Noise of DDS is multiplied by $N^2$
  - Complex and power hungry
**Fractional-N Frequency Synthesizers**

- Break constraint that divide value be integer
  - Dither divide value dynamically to achieve fractional values
  - Frequency resolution is now arbitrary regardless of $1/T$
- Want high $1/T$ to allow a high PLL bandwidth
Use an accumulator to perform dithering operation
- Fractional input value fed into accumulator
- Carry out bit of accumulator fed into divider
Carry out bit is asserted when accumulator residue reaches or surpasses its full scale value
- Accumulator residue increments by input fractional value each clock cycle
Fractional-N Synthesizer Signals with $N = 4.25$

- Divide value set at $N = 4$ most of the time
  - Resulting frequency offset causes phase error to accumulate
  - Reset phase error by “swallowing” a VCO cycle
    - Achieved by dividing by 5 every 4 reference cycles
The Issue of Spurious Tones

- **PFD error is periodic**
  - Note that actual PFD waveform is series of pulses – the sawtooth waveform represents pulse width values over time
- **Periodic error signal creates spurious tones in synthesizer output**
  - Ruins noise performance of synthesizer

\[
N_{sd}[k] = N + \text{frac}[k]
\]
The Phase Interpolation Technique

- Phase error due to fractional technique is predicted by the instantaneous residue of the accumulator
  - Cancel out phase error based on accumulator residue
The Problem With Phase Interpolation

- Gain matching between PFD error and scaled D/A output must be extremely precise
  - Any mismatch will lead to spurious tones at PLL output
Is There a Better Way?
Sigma-Delta dithers in a manner such that resulting quantization noise is “shaped” to high frequencies.
**Linearized Model of Sigma-Delta Modulator**

- Composed of two transfer functions relating input and noise to output
  - **Signal transfer function (STF)**
    - Filters input (generally undesirable)
  - **Noise transfer function (NTF)**
    - Filters (i.e., shapes) noise that is assumed to be white
Example: Cutler Sigma-Delta Topology

- Output is quantized in a multi-level fashion
- Error signal, $e[k]$, represents the quantization error
- Filtered version of quantization error is fed back to input
  - $H(z)$ is typically a highpass filter whose first tap value is 1
    - i.e., $H(z) = 1 + a_1 z^{-1} + a_2 z^{-2} \ldots$
  - $H(z) - 1$ therefore has a first tap value of 0
    - Feedback needs to have delay to be realizable
Represent quantizer block as a summing junction in which $r[k]$ represents quantization error

- Note:

$$e[k] = y[k] - u[k] = (u[k] + r[k]) - u[k] = r[k]$$

- It is assumed that $r[k]$ has statistics similar to white noise
  - This is a key assumption for modeling – often not true!
Calculation of Signal and Noise Transfer Functions

- Calculate using Z-transform of signals in linearized model

\[
Y(z) = U(z) + R(z) \\
= X(z) + (H(z) - 1)E(z) + R(z) \\
= X(z) + (H(z) - 1)R(z) + R(z) \\
= X(z) + H(z)R(z)
\]

- NTF: \( H_n(z) = H(z) \)
- STF: \( H_s(z) = 1 \)
A Common Choice for $H(z)$

$$H(z) = (1 - z^{-1})^m$$

$$\Rightarrow |H(e^{j2\pi fT})| = |(1 - e^{-j2\pi fT})^m|$$
Example: First Order Sigma-Delta Modulator

- Choose NTF to be

\[ H_n(z) = H(z) = 1 - z^{-1} \]

- Plot of output in time and frequency domains with input of

\[ x[k] = 0.5 + 0.25 \sin \left( \frac{2\pi k}{100} \right) \]
Example: Second Order Sigma-Delta Modulator

- Choose NTF to be

\[ H_n(z) = H(z) = (1 - z^{-1})^2 \]

- Plot of output in time and frequency domains with input of

\[ x[k] = 0.5 + 0.25 \sin \left( \frac{2\pi}{100} k \right) \]
Example: Third Order Sigma-Delta Modulator

- Choose NTF to be

\[ H_n(z) = H(z) = (1 - z^{-1})^3 \]

- Plot of output in time and frequency domains with input of

\[ x[k] = 0.5 + 0.25 \sin \left( \frac{2\pi}{100} k \right) \]
Observations

- Low order Sigma-Delta modulators do not appear to produce “shaped” noise very well
  - Reason: low order feedback does not properly “scramble” relationship between input and quantization noise
    - Quantization noise, r[k], fails to be white
- Higher order Sigma-Delta modulators provide much better noise shaping with fewer spurs
  - Reason: higher order feedback filter provides a much more complex interaction between input and quantization noise
**Warning: Higher Order Modulators May Still Have Tones**

- Quantization noise, $r[k]$, is best whitened when a “sufficiently exciting” input is applied to the modulator
  - Varying input and high order helps to “scramble” interaction between input and quantization noise
- Worst input for tone generation are DC signals that are rational with a low valued denominator
  - Examples (third order modulator):

\[
\begin{align*}
x[k] &= 0.1 \\
x[k] &= 0.1 + 1/1024
\end{align*}
\]
Fractional Spurs Can Be Theoretically Eliminated


Cascaded Sigma-Delta Modulator Topologies

- Achieve higher order shaping by cascading low order sections and properly combining their outputs
- Advantage over single loop approach
  - Allows pipelining to be applied to implementation
    - High speed or low power applications benefit
- Disadvantages
  - Relies on precise matching requirements when combining outputs (not a problem for digital implementations)
  - Requires multi-bit quantizer (single loop does not)
MASH topology

- Cascade first order sections
- Combine their outputs after they have passed through digital differentiators
Calculation of STF and NTF for MASH topology (Step 1)

- Individual output signals of each first order modulator

\[
\begin{align*}
    y_1(z) &= x(z) - (1 - z^{-1})r_1(z) \\
    y_2(z) &= r_1(z) - (1 - z^{-1})r_2(z) \\
    y_3(z) &= r_2(z) - (1 - z^{-1})r_3(z)
\end{align*}
\]

- Addition of filtered outputs

\[
\frac{y_1(z)}{1} + (1 - z^{-1})y_2(z) + (1 - z^{-1})^2y_2(z) = x(z) - (1 - z^{-1})^3r_3(z)
\]
Calculation of STF and NTF for MASH topology (Step 1)

- Overall modulator behavior

\[ y(z) = x(z) - (1 - z^{-1})^3 r_3(z) \]

- STF: \( H_s(z) = 1 \)
- NTF: \( H_n(z) = (1 - z^{-1})^3 \)
Sigma-Delta Frequency Synthesizers

- Use Sigma-Delta modulator rather than accumulator to perform dithering operation
  - Achieves much better spurious performance than classical fractional-N approach

Riley et. al., JSSC, May 1993
Background: The Need for A Better PLL Model

- Classical PLL model
  - Predicts impact of PFD and VCO referred noise sources
  - Does not allow straightforward modeling of impact due to divide value variations
    - This is a problem when using fractional-N approach
A PLL Model Accommodating Divide Value Variations

**Parameterized Version of New Model**

\[
\Phi_{vn}(t) = \Phi_{npfd}(t) + \Phi_{n}(t) + \Phi_{nvco}(t)
\]

Noise:
- \( \Phi_{jit}[k] \)
- \( e_{spur}(t) \)
- \( I_{cpn}(t) \)

Alternate Representation:
- \( n[k] \)
- \( G(f) \)
- \( F_c(t) \)

Divide value variation:
- \( \Phi_{d}(t) \)
- \( \Phi_{c}(t) \)

PFD-referred Noise:
- \( S_{E_n}(f) \)

VCO-referred Noise:
- \( S_{\Phi_{vn}}(f) \)
- \(-20 \text{ dB/dec}\)
Spectral Density Calculations

- **Case (a):**
  \[ S_y(f) = |H(f)|^2 S_x(f) \]

- **Case (b):**
  \[ S_y(e^{j2\pi fT}) = |H(e^{j2\pi fT})|^2 S_x(e^{j2\pi fT}) \]

- **Case (c):**
  \[ S_y(f) = \frac{1}{T} |H(f)|^2 S_x(e^{j2\pi fT}) \]
Example: Calculate Impact of Ref/Divider Jitter (Step 1)

- Assume jitter is white
  - i.e., each jitter value independent of values at other time instants
- Calculate spectra for discrete-time input and output
  - Apply case (b) calculation

\[
S_{\Delta t_{jit}}(e^{j2\pi fT}) = \beta^2 \quad \Rightarrow \quad S_{\Phi_{jit}}(e^{j2\pi fT}) = \left| \frac{2\pi}{T} \right|^2 \beta^2
\]
Example: Calculate Impact of Ref/Divider Jitter (Step 2)

- Compute impact on output phase noise of synthesizer
  - We now apply case (c) calculation

\[
S_{\Phi_n}(f) = \frac{1}{T} |TN_{nom}G(f)|^2 S_{\Phi_{jit}}(e^{j2\pi fT})
\]

\[
= \frac{1}{T} |TN_{nom}G(f)|^2 \left(\frac{2\pi}{T} \right)^2 \beta^2
\]

- Note that G(f) = 1 at DC
Now Consider Impact of Divide Value Variations

### Noise

- $\Phi_{j\text{it}[k]}$
- $\frac{T}{2\pi}$
- $\Phi_{out}(t)$
- $\Phi_{npfd}(t)$
- $\Phi_{nvco}(t)$
- $\Phi_{\text{vco-ref}}$
- $\Phi_{\text{pfd-ref}}$
- $\Phi_n(t)$
- $\Phi_{\text{out}(t)}$

### Divide Value Variation

- $n[k]$
- $\frac{2\pi z^{-1}}{1 - z^{-1}}$
- $\Phi_d(t)$
- $\Phi_c(t)$
- $\Phi_{\text{out}(t)}$

### Alternate Representation

- $\frac{1}{jf}$
- $F_c(t)$
- $\Phi_c(t)$

### Frequency Domain

- $S_{\Phi_n}(f)$
- $S_{\Phi_{\text{vco-ref}}}(f)$
- $-20 \text{ dB/dec}$
Divider Impact For Classical Vs Fractional-N Approaches

Classical Synthesizer

\[
G(f) \rightarrow n[k] \rightarrow \frac{1}{T} \rightarrow F_{out}(t)
\]

D/A and Filter

\[ n(t) \]

\[ 1 \]

\[ 1/T \]

Fractional-N Synthesizer

\[
\frac{1}{T} \rightarrow n_{sd}[k] \rightarrow \text{Dithering Modulator} \rightarrow \frac{1}{T} \rightarrow G(f) \rightarrow F_{out}(t)
\]

\[ n_{sd}(t) \]

\[ 1 \]

\[ 1/T \]

\[ 1/T \] block represents sampler (to go from CT to DT)

M.H. Perrott
Focus on Sigma-Delta Frequency Synthesizer

- Divide value can take on fractional values
  - Virtually arbitrary resolution is possible
- PLL dynamics act like lowpass filter to remove much of the quantization noise
Quantifying the Quantization Noise Impact

- Calculate by simply attaching Sigma-Delta model
  - We see that quantization noise is integrated and then lowpass filtered before impacting PLL output
**A Well Designed Sigma-Delta Synthesizer**

- **Order of G(f) is set to equal to the Sigma-Delta order**
  - Sigma-Delta noise falls at -20 dB/dec above G(f) bandwidth
- **Bandwidth of G(f) is set low enough such that synthesizer noise is dominated by intrinsic PFD and VCO noise**

---

*M.H. Perrott*
Impact of Increased PLL Bandwidth

- Allows more PFD noise to pass through
- Allows more Sigma-Delta noise to pass through
- Increases suppression of VCO noise
Impact of Increased Sigma-Delta Order

- PFD and VCO noise unaffected
- Sigma-Delta noise no longer attenuated by G(f) such that a -20 dB/dec slope is achieved above its bandwidth

M.H. Perrott
Summary of Advanced Analog Synthesizers

- Integer-N synthesizers have limited resolution for a given PLL bandwidth
- Advanced synthesizers improve the achievable resolution for a given bandwidth
  - Dual-loop synthesizers leverage two synthesizers and an I/Q mixer
  - Direct digital synthesizers use a lookup table and digital logic
  - Fractional-N synthesizers leverage Sigma-Delta modulation
    - Simpler structure than the other approaches
    - Primary issue is introduction of Sigma-Delta quantization noise
Clock And Data Recovery
A challenging component is the clock and data recovery circuit (CDR)

- Two primary functions
  - Extract the clock corresponding to the input data signal
  - Resample the input data
Outline of Talk

- Clock and Data Recovery circuits
  - Jitter specifications
  - Phase detection
  - Modeling
  - Data dependent jitter
  - Bang-bang systems

- Delay locked loops
  - Implementation
  - The issue of infinite delay
PLL Based Clock and Data Recovery

- Use a phase locked loop to tune the frequency and phase of a VCO to match that of the input data

- Performance issues
  - Jitter
  - Acquisition time
  - Bit error rate (at given input levels)

- Let’s focus on specifications for OC-192
  - i.e., 10 Gbit/s SONET
Definition
- The amount of jitter at the output of the CDR when no jitter (i.e., negligible jitter) is present on the data input

SONET requires
- < 10 mUI rms jitter
- < 100 mUI peak-to-peak jitter

Note: UI is unit interval, and is defined as the period of the clk signal (i.e., 100 ps for 10 Gbit/s data rates)
**Jitter Tolerance**

- **Definition**
  - The maximum amount of jitter allowed on the input while still achieving low bit error rates (< 10^{-12})

- **SONET specifies jitter tolerance according to the frequency of the jitter**
  - Low frequency jitter can be large since it is tracked by PLL
  - High frequency jitter (above the PLL bandwidth) cannot be as high (PLL can’t track it out)
    - Limited by setup and hold times of PD retiming register
Example Jitter Tolerance Mask

- CDR tested for tolerance compliance by adding sine wave jitter at various frequencies (with amplitude greater than mask) to the data input and observing bit error rate.
### Jitter Transfer

- **Definition**
  - The amount of jitter attenuation that the CDR provides from input to output

- **SONET specifies jitter transfer by placing limits on its transfer function behavior from input to output**
  - **Peaking behavior:** low frequency portion of CDR transfer function must be less than 0.1 dB
  - **Attenuation behavior:** high frequency portion of CDR transfer function must not exceed a mask limit
Example Jitter Transfer Mask

CDR tested for compliance by adding sine wave jitter at various frequencies and observing the resulting jitter at the CDR output.
Summary of CDR Performance Specifications

- **Jitter**
  - Jitter generation
  - Jitter tolerance
  - Jitter transfer (and peaking)

- **Acquisition time**
  - Must be less than 10 ms for many SONET systems

- **Bit error rates**
  - Must be less than 1e-12 for many SONET systems
Phase Detectors in Clock and Data Recovery Circuits

- Key issue
  - Must accommodate “missing” transition edges in input data sequence

- Two styles of detection
  - Linear – PLL can analyzed in a similar manner as frequency synthesizers
  - Nonlinear – PLL operates as a bang-bang control system (hard to rigorously analyze in many cases)
**Popular CDR Phase Detectors**

- **Linear**
  - Hogge detector produces an error signal that is proportional to the instantaneous phase error

- **Nonlinear**
  - Alexander (Bang-bang) detector produces an error signal that corresponds to the sign of the instantaneous phase error
A Closer Look at the Hogge Detector

- **Error output, e(t), consists of two pulses with opposite polarity**
  - Positive polarity pulse has an area that is proportional to the phase error between the data and clk
  - Negative polarity pulse has a fixed area corresponding to half of the clk period
  - Overall area is zero when data edge is aligned to falling clk edge
**Example CDR Settling Characteristic with Hogge PD**

- CDR tracks out phase error with an exponential transition response
- Jitter occurring at steady state is due to VCO and non-idealities of phase detector
Modeling of CDR with Hogge Detector

- Similar to frequency synthesizer model except
  - No divider
  - Phase detector gain depends on the transition density of the input data

- The issue of transition density
  - Phase error information of the input data signal is only seen when it transitions
    - VCO can wander in the absence of transitions
  - Open loop gain (and therefore the closed loop bandwidth) is decreased at low transition densities
Use a lead/lag filter to implement a type II loop

- Integrator in $H(s)$ forces the steady-state phase error to zero (important to minimize jitter)
Key issue: an undesired pole/zero pair occurs due to stabilizing zero in the lead/lag filter
**Corresponding Closed Loop Frequency Response**

- Undesired pole/zero pair causes peaking in the closed loop frequency response.
- SONET demands that peaking must be less than 0.1 dB.
  - For classical lead/lag filter approach, this must be achieved by having a very low-valued zero.
    - Requires a large loop filter capacitor.
An Interesting Observation

- Calculation of closed loop transfer function

\[
\frac{Y(s)}{X(s)} = \frac{N_A(s)/D_A(s)}{1 + N_B(s)/D_B(s) \cdot N_A(s)/D_A(s)} = \frac{N_A(s)D_B(s)}{D_A(s)D_B(s) + N_B(s)N_A(s)}
\]

- Key observation
  - Zeros in feedback loop do not appear as zeros in the overall closed loop transfer function!
Method of Achieving Zero Peaking

- We can implement a stabilizing zero in the PLL feedback path by using a variable delay element
  - Loop filter can now be implemented as a simple integrator
- Issue: delay must support a large range
Model of CDR with Delay Element

- Delay “gain”, $K_d$, is set by delay implementation
- Note that $H(s)$ can be implemented as a simple capacitor
  - $H(s) = 1/(sC)$
Derivation of Zero Produced by Delay Element

- Zero set by ratio of delay gain to VCO gain
Alternate Implementation

- Can delay data rather than clk
  - Same analysis as before
The Issue of Data Dependent Jitter

- For classical or Bulzacchelli CDR
  - Type II PLL dynamics are employed so that steady state phase detector error is zero
- Issue: phase detector output influences VCO phase through a double integrator operation
  - The classical Hogge detector ends up creating data dependent jitter at the VCO output
The double integral of the e(t) pulse sequence is nonzero (i.e., has DC content)

- Since the data transition activity is random, a low frequency noise source is created
  - Low frequency noise not attenuated by PLL dynamics
One Possible Fix

- **Modify Hogge so that the double integral of the e(t) pulse sequence is zero**
  - Low frequency noise is now removed

- **See L. Devito et. al., “A 52 MHz and 155 MHz Clock-recovery PLL”, ISSCC, Feb, 1991**
Error output consists of pulses of fixed area that are either positive or negative depending on phase error.

Pulses occur at data edges:
- Data edges detected when sampled data sequence is different than its previous value.

Above example illustrates the impact of having the data edge lagging the clock edge.
Above example illustrates the impact of having the data edge *leading* the clk edge
- Error pulses have opposite sign from lagging edge case
Bang-bang CDR response is slew rate limited
- Much faster than linear CDR, in general
- Steady-state jitter often dominated by bang-bang behavior (jitter set by error step size and limit cycles)
The Issue of Limit Cycles

- **Bang-bang loops exhibit limit cycles during steady-state operation**
  - Above diagram shows resulting waveforms when data transitions on every cycle
  - Signal patterns more complicated for data that randomly transitions
- **For lowest jitter: want to minimize period of limit cycles**
The Impact of Delays in a Bang-Bang Loop

- Delays increase the period of limit cycles, thereby increasing jitter
Practical Implementation Issues for Bang-Bang Loops

- Minimize limit cycle periods
  - Use phase detector with minimal delay to error output
  - Implement a high bandwidth feedforward path in loop filter
    - One possibility is to realize feedforward path in VCO

- Avoid dead zones in phase detector
  - Cause VCO phase to wonder within the dead zone, thereby increasing jitter

- Use simulation to examine system behavior
  - Nonlinear dynamics can be non-intuitive
  - For first order analysis, see R.C. Walter et. al., “A Two-Chip 1.5-GBd Serial Link Interface”, JSSC, Dec 1992
Recall the CDR Model (Hogge Det.) From Lecture 21

- Similar to frequency synthesizer model except
  - No divider
  - Phase detector gain depends on the transition density of the input data

\[ \Phi_{data}(t) \]
Key Observation: Must Use a Type II Implementation

- Integrator in $H(s)$ forces the steady-state phase error to zero
  - Important to achieve aligned clock and to minimize jitter
Issue: *Type II System Harder to Design than Type I*

- **A stabilizing zero is required**
- **Undesired closed loop pole/zero doublet causes peaking**

**Evaluation of Phase Margin**

- Open loop gain increased
- $20 \log |A(f)|$
- $0 \text{ dB}$
- $f_z$, $f_p$
- $\angle(A(f))$
- $120^\circ$, $140^\circ$, $160^\circ$
- $-180^\circ$

**Closed Loop Pole Locations of G(f)**

- Dominant pole pair
- Non-dominant pole
- PM = 54° for B
  - PM = 53° for A
  - PM = 55° for C

---

M.H. Perrott
Delay Locked Loops

- Delay element used in place of a VCO
  - No integration from voltage input to phase output
  - System is Type 1
System Design Is Easier Than For CDR

Evaluation of Phase Margin

Open loop gain increased

Closed Loop Pole Locations of $G(f)$

- No stabilizing zero required
  - No peaking in closed loop frequency response
Example Delay-Locked Loop Implementation

- Assume an input clock is provided that is perfectly matched in frequency to data sequence
  - However, phase must be adjusted to compensate for propagation delays between clock and data on the PC board
- A variable delay element is used to lock phase to appropriate value
  - Phase detector can be similar to that used in a CDR
    - Hogge, Bang-Bang, or other structures possible
Delay needs to support an infinite range if system to be operated continuously
  - Can otherwise end up at the end of range of delay element
    - Won’t be able to accommodate temperature variations

Methods have been developed to achieve infinite range delay elements
  - Efficient implementation of such delay elements is often the key issue for high performance designs
Delay locked loop designers always point to jitter accumulation problem of phase locked loops
- Implication is that delay locked loops can achieve much lower jitter than clock and data recovery circuits

The reality: phase locked loops can actually achieve lower jitter than delay locked loops
- PLL’s can clean up high frequency jitter of input clock
- Whether a PLL or DLL is better depends on application (and achievable VCO performance)
One Method of Achieving Infinite Delay

\[
\cos(2\pi f_{\text{in}} t + \phi) = \cos(2\pi f_{\text{in}} t) \cos(\phi) - \sin(2\pi f_{\text{in}} t) \sin(\phi)
\]

- Phase shift of a sine wave can be implemented with I/Q modulation

\[
I = \cos(\Phi), \quad Q = \sin(\Phi)
\]

- Note: infinite delay range allows DLL to be used to adjust frequency as well as phase
  - Phase adjustment now must vary continuously
  - Hard to get low jitter in practical implementations
Conceptual Implementation of Infinite Delay Range

\[
\cos(2\pi f_{\text{int}}t + \phi) = \cos(2\pi f_{\text{int}}t)\cos(\phi) - \sin(2\pi f_{\text{int}}t)\sin(\phi)
\]

- Practical designs often implement \(\cos(\Phi)\) and \(\sin(\Phi)\) signals as phase shifted triangle waves.
Some References on CDR’s and Delay-Locked Loops

- Tom Lee et. al. were pioneers of the previous infinite range DLL approach
  - See T. Lee et. al., “A 2.5 V CMOS Delay-Locked Loop for an 18 Mbit, 500 Megabyte/s DRAM”, JSSC, Dec 1994

- Check out papers from Mark Horowitz’s group at Stanford
  - Oversampling data recovery approach
    - See C-K K. Yang et. al., “A 0.5-um CMOS 4.0-Gbit/s Serial Link Transceiver with Data Recovery using Oversampling”, JSSC, May 1998
  - Multi-level signaling
    - See Ramin Farjad-Rad et. al., “A 0.3-um CMOS 8-Gb/s 4-PAM Serial Link Transceiver”, JSSC, May 2000
  - Bi-directional signaling
    - See E. Yeung, “A 2.4 Gb/s/pin simultaneous bidirectional parallel link …”, JSSC, Nov 2000
Summary

- Clock and data recovery circuits generate a clock that is phase and frequency aligned to an incoming data signal
  - Jitter characteristics are one of the key performance metrics
  - Implementations are either linear or bang-bang
    - Linear is needed for well-defined jitter transfer function
    - Bang-bang has simpler implementation
- Delay locked loops phase align an existing clock to an incoming data stream
  - Potentially simpler implementation than CDR