Samples to Digital Symbols: Symbol Clock Recovery and Improved Symbol Synchronization Blocks

Andy Walls
12 Sept 2017
Outline

• Advertisement
  – About SilverBlock

• Theoretical
  – Problem Statement
  – Symbol Synchronization Overview
  – PLL Symbol Synchronizer
  – Clock Tracking PLL Model
  – Timing Error Detector
  – Interpolating Resampler
Outline (continued)

• Practical
  – GNURadio Sync Blocks
  – Existing Blocks’ Deficiencies
  – New Symbol Sync Blocks Overview
  – New Symbol Sync Blocks Architecture
  – Adding a New Timing Error Detector
  – Adding a New Resampler
  – Using a Different Slicer
  – Existing Blocks to New Blocks
  – Usage Hints and Gotchas
  – Experimental Tuning Example
  – TED S-Curve Simulation
SilverBlock Systems

• Small company founded in 2009
  – Leonardtown, MD (0.3 miles from the waterfront)
    https://www.visitstmarysmd.com/see-do/towns-communities/leonardtown-area/
  – Rochester, NY

• What we build
  – Distributed Mission Computing and Display Software
  – RADAR Processing Algorithms and GPU Implementations
  – Experimental Communications Systems
  – Experimental Airborne Sensors and Systems

• Looking to hire 1 or 2 experienced software engineers
  – C++ required. US Citizenship required.
  – contact@silverblocksystems.net
Problem Statement

- Output a sample stream synchronized with the center of data symbols
Symbol Synch Overview

• Two broad categories of algorithms

  – Feedfoward (Open Loop)
    • Block oriented
    • Operate on samples for a number of symbols at a time
    • Non-tracking, but can be computationally complex
    • Burst mode communications, or initial acquisition of synchronization

  – Feedback (Closed Loop)
    • Stream oriented
    • Operate on immediate incoming sample or symbol
    • Tracking, and not computationally complex for any 1 input
    • Continual stream of symbols, or tracking after initial acquisition
Symbol Synch Overview

- Algorithm and Implementation Requirements
  - Resampling with interpolation
    - Imposes signal bandwidth requirements
  - Symbol clock timing estimation or timing error estimation
    - Imposes signal conditioning requirements
  - Offline modeling, simulation, and analysis by the designer!

- Correlation-based Feedforward Algorithm and Implementation
  - GNURadio has ~1 implementation

- PLL-based Feedback Algorithm and Implementation
  - GNURadio now has 4 implementations
PLL Symbol Synchronizer

- Timing Error Detector: analogous to PLL phase detector
  - Estimates the symbol clock timing error, a hidden quantity
- Resampler Control: analogous to PLL phase accumulator/NCO
- $T_s$: input sample clock period
- $T_c$: symbol clock period
- $\tau_0$: optimal symbol sampling offset

$s(\tau_1 + mT_s) \rightarrow x(\tau_2 + mT_s) \rightarrow$ Interpolating Resampler

$\rightarrow$ Resampler Control

$\rightarrow$ PI Loop Filter

$\rightarrow$ Timing Error Detector

$x(\tau_0 + nT_c)$

samples at symbol centers
Clock Tracking PLL Model

- Timing Error Detector (TED) modeled by phase detector (PD) and $K_{pd}$ gain
  - Outputs symbol clock timing error estimate, $e[n]$, once per symbol
- Proportional-Integral (PI) Loop Filter
  - Proportional arm gain, $\alpha$
  - Integral arm gain, $\beta$
  - Integral arm output: estimate of average period of symbol clock, $T_{avg}$
  - Filter output: estimate of instantaneous period of symbol clock, $T_{inst}$
- Resampler & Control maps to phase accumulator and phase signals
Clock Tracking PLL Model

- **Loop Phase Transfer Function (PI gains)**

\[
H(z) = \frac{\Theta_o(z)}{\Theta_i(z)} = K_{pd}(\alpha + \beta)z^{-1} \cdot \frac{1 - \frac{\alpha}{\alpha + \beta}z^{-1}}{1 - 2\left(1 - K_{pd}\frac{\alpha + \beta}{2}\right)z^{-1} + (1 - K_{pd}\alpha)z^{-2}}
\]

- **Zeros, Poles, and Critical Damping (PI gains)**

\[
z_1 = \frac{\alpha}{\alpha + \beta}
\]

\[
z_2 \to \infty
\]

\[
p_{1,2} = \left(1 - K_{pd}\frac{\alpha + \beta}{2}\right) \pm \sqrt{\left(1 - K_{pd}\frac{\alpha + \beta}{2}\right)^2 - (1 - K_{pd}\alpha)}
\]

\[
\alpha = -\beta \pm \frac{2}{|K_{pd}|}\sqrt{K_{pd}\beta} \quad \text{for critical damping}
\]
Clock Tracking PLL Model

- Mapping Poles of a 2\textsuperscript{nd} Order Analog Control System to z-plane

\[ s_{1,2} = -\zeta \omega_n \pm j \omega_d = -\zeta \omega_n \pm j \omega_n \sqrt{1 - \zeta^2} \]

\( \zeta \): damping factor (1.0 is critically damped)

\( \omega_n \): natural frequency (radius from origin, approx 1-sided BW for underdamped system)

\( \omega_d \): damped frequency of oscillation (ringing of underdamped impulse response)

\[ z = e^{sT} \]

\[ p_{1,2} = e^{-\zeta \omega_n T \pm j \omega_d T} \]

- Loop Phase Transfer Function (2\textsuperscript{nd} Order Control System)

\[ H(z) = \begin{cases} 
\frac{[2 - 2 \cos(\omega_d T)e^{-\zeta \omega_n T}] z - 2 \sinh(\zeta \omega_n T)e^{-\zeta \omega_n T}}{z^2 - 2 \cos(\omega_d T)e^{-\zeta \omega_n T} z + e^{-2\zeta \omega_n T}} & \text{for } \zeta < 1 \quad \text{with } \omega_d T = \omega_n T \sqrt{1 - \zeta^2} \\
\frac{[2 - 2(1)e^{-\zeta \omega_n T}] z - 2 \sinh(\zeta \omega_n T)e^{-\zeta \omega_n T}}{z^2 - 2(1)e^{-\zeta \omega_n T} z + e^{-2\zeta \omega_n T}} & \text{for } \zeta = 1 \quad \text{with } \omega_d T = 0 \\
\frac{[2 - 2 \cosh(\omega_d T)e^{-\zeta \omega_n T}] z - 2 \sinh(\zeta \omega_n T)e^{-\zeta \omega_n T}}{z^2 - 2 \cosh(\omega_d T)e^{-\zeta \omega_n T} z + e^{-2\zeta \omega_n T}} & \text{for } \zeta > 1 \quad \text{with } \omega_d T = \omega_n T \sqrt{\zeta^2 - 1} 
\end{cases} \]
Clock Tracking PLL Model

- PI Gains from 2\textsuperscript{nd} Order Digital Control Loop parameters:

\[ \alpha = \frac{2}{K_{pd}} e^{-\zeta \omega_n T} \sinh(\zeta \omega_n T) \]

\[ \beta = \begin{cases} 
\frac{2}{K_{pd}} \left( 1 - e^{-\zeta \omega_n T} \left[ \sinh(\zeta \omega_n T) + \cos(\omega_d T) \right] \right) & \text{for } \zeta < 1 \quad \text{(under damped)} \\
\frac{2}{K_{pd}} \left( 1 - e^{-\zeta \omega_n T} \left[ \sinh(\zeta \omega_n T) + 1 \right] \right) & \text{for } \zeta = 1 \quad \text{(critically damped)} \\
\frac{2}{K_{pd}} \left( 1 - e^{-\zeta \omega_n T} \left[ \sinh(\zeta \omega_n T) + \cosh(\omega_d T) \right] \right) & \text{for } \zeta > 1 \quad \text{(over damped)} 
\end{cases} \]
Timing Error Detector

- A TED emits an error value proportional to the time difference between
  - Optimal current symbol sampling time (blue)
  - Actual current symbol sampling time (red)
Timing Error Detector

- A TED is just an expression for the error value, $e[n]$
  - Formally derived - *Pay attention to the assumptions!*
  - Can include factors such as
    - Current symbol estimate
    - Nearby samples or symbol estimates
    - Current symbol decision (Decision Directed TEDs)
    - Nearby symbol decisions (Decision Directed TEDs)
    - Estimate of signal slope at symbol sampling time
    - Estimate of $E_s/N_0$
  - Usually a simple expression in the end
- Examples
  - Mueller and Müller $e[n] = \text{slice}(x[n-1])x[n] - \text{slice}(x[n])x[n-1]$
  - Small signal ML approximation $e[n] = x[n]x'[n]$
Timing Error Detector

- A TED is characterized by its S-curve: \( S(\tau_e) = E\{e[n]|\tau_e\} \)
  - y-axis: expected value of timing error output, given the normalized symbol clock timing offset
  - \( \tau \)-axis: normalized symbol clock timing offset
  - Slope at timing offset \( \tau = 0 \) is TED gain, \( K_{\text{ted}} (= K_{pd}) \)
    - Units of \( K_{\text{ted}} \) might not match the units required by the loop – scaling needed

- S-Curve shape and central slope at \( \tau = 0 \) depend on
  - TED’s error estimator expression
  - Input signal amplitude
  - Pulse shaping filter / pulse shape
  - Input \( E_s/N_0 \)
  - Other factors

- Simulation required to find TED gain
  - Octave, MatLab, R, Python, or whatever
Interpolating Resampler

- Symbol synchronization process reduces sample rate
- Input samples (red) not at optimal symbol sampling (blue)
- Need to resample input between samples

\[
T_s \cdot (n + f) = T_{\text{inst}} = T_{\text{avg}} + \alpha e[n]
\]

\[
0.333 T_s = \mu T_s
\]
Interpolating Resampler

- Interpolation implemented with FIR filters
  - Fractional Delay (FD)
    - MMSE interpolator polyphase filterbank
    - Matched filter/interpolator polyphase filterbank
  - Polynomial Interpolation using Farrow structure
    - Lagrange
    - B-spline

- Practical interpolation filters impose bandwidth constraint
  - Input signal bandwidth must be some fraction of $F_s$
  - Bounds the error in the interpolated output samples
Interpolating Resampler

- Ideal FD interpolation filter frequency responses
  - Interpolator
    \[ H(\omega) = 1 \cdot e^{j\omega\mu} \quad \text{for} \quad -\pi \leq \omega \leq \pi \]
  - Differentiating interpolator
    \[ H(\omega) = j\omega \cdot e^{j\omega\mu} \quad \text{for} \quad -\pi \leq \omega \leq \pi \]
  - \( \mu \) is intersample interpolation fraction in \([0.0, 1.0]\)

- Ideal FD interpolation filter impulse responses
  - Interpolator
    \[ h[n] = \text{sinc}(n + \mu) \]
  - Differentiating interpolator
    \[ h[n] = \frac{1}{n + \mu} \left[ \cos(\pi[n + \mu]) - \text{sinc}(n + \mu) \right] \]
  - For a particular \( \mu \) in \([0.0, 1.0]\)
  - These ideal responses from IDTFT are infinite length
Interpolating Resampler

- GR Minimum Mean Squared Error interpolators
  - Truncated, MMSE version of ideal filters
  - Bank of 129, 8 tap FIR filters
  - For \( \mu \) in \{0/128, 1/128, 2/128, ..., 127/128, 128/128\}
  - Each filter has a MMSE \( H(\omega) \) only in \([-F_s/4, F_s/4]\)

- One sided freq response and error\(^2\) of the filters
Interpolating Resampler

- GR’s MMSE interpolator is a polyphase filter bank
- Equivalent PFB prototype filter vs truncated ideal
GNURadio Sync Blocks

**Existing**

- Clock Recovery MM
  - Omega:
  - Gain Omega:
  - Mu:
  - Gain Mu:
  - Omega Relative Limit:

- MSK Timing Recovery
  - Gain:
  - Samples per symbol:
  - Error limit:
  - Output samples per symbol:

- Polyphase Clock Sync
  - Samples/Symbol:
  - Loop Bandwidth:
  - Taps:
  - Filter Size:
  - Initial Phase:
  - Maximum Rate Deviation:
  - Output SPS:

**New**

- Symbol Sync
  - Timing Error Detector: Gardner
  - Samples per Symbol:
  - Expected TED Gain:
  - Loop Bandwidth:
  - Damping Factor:
  - Maximum Deviation:
  - Output Samples/Symbol:
  - Interpolating Resampler: MMSE, 8 tap FIR

- Symbol Sync
  - Timing Error Detector: Early-Late
  - Samples per Symbol:
  - Expected TED Gain:
  - Loop Bandwidth:
  - Damping Factor:
  - Maximum Deviation:
  - Output Samples/Symbol:
  - Interpolating Resampler: Polyphase Filterbank, MF Filterbank Arms:
  - PPB MF Taps:
Existing Blocks’ Deficiencies

- Incorrect tag propagation
- Conflates symbol clock phase and interpolator phase
  - Self noise & Unable to stay locked on a clock pattern
- Incorrect decision slicer constellation
- Drops some input, when > 8 samples/symbol
- No reset on receipt of time_est tag
- No way to change TED, slicer, or resampler
  - Whole new blocks needed - bringing new bugs
- Initializes to very overdamped loop filter
- PI filter gain computations ignored TED gain
- Restricted to 1 or 2 samples/symbol on output
New Symbol Sync Blocks

- Fixed, replacement superset of existing blocks
  - Except can’t change MF taps on the fly (yet)
- Selectable TED, slicer, and resampler
New Symbol Sync Blocks

$S(t, +mT_s)$  $X(t, +mT_s)$

**MATCHED FILTER**

$T_s$: input sample clock period  
$t_1, t_2$: input sample phase/delay

$T_c$: symbol clock period  
$k$: FEO inputs required per symbol  
$l$: output samples per symbol

$X_0$: input  
$Y_0$: resampled output  
$D_0$: symbol decision

**SYMBOL SYNCHRONIZER**

**INTERPOLATING FILTER BANK**

**INTERPOLATING FILTER BANK**

**INTERPOLATING DIFFERENTIATOR FILTER BANK**

**P-I LOOP FILTER**

**TIMING ERROR DETECTOR (TED)**

**SLICER CONSTELLATION**

@awalls-cx18 for not using an actual napkin for that diagram.
New Symbol Sync Blocks

- **External clocks**
  - Input sample
- **Internal clocks**
  - Interp output
  - Block output
  - TED input
  - Symbol
    (TED Output)
Adding a New TED

• Modify the following files
  – gr-digital/include/.../timing_error_detector_type.h
  – gr-digital/grc/digital_symbol_sync_xx.xml
  – gr-digital/lib/timing_error_detector.*

• Your new derived TED class only needs
  – A simple constructor
  – Two methods to compute the error output term
    • Complex input
    • Float input

• Leave the symbol sync blocks’ code alone
  – tag handling, slicer, resampler, & loop filter – all done!
Adding a New Resampler

- Modify the following files
  - `gr-digital/include/.../interpolating_resampler_type.h`
  - `gr-digital/grc/digital_symbol_sync_xx.xml`
  - `gr-digital/lib/interpolating_resampler.*`

- Your two (1 float, 1 complex) new derived resampler classes each need
  - A constructor
  - A simple `ntaps()` method
  - An `interpolate()` method
  - A `differentiate()` method (interpolating differentiator)

- Leave the symbol sync blocks’ code alone
  - tag handling, slicer, TED, & loop filter – all done!
Using a Different Slicer

- Instantiate a custom Constellation Object
  - Slicer only needed for decision directed TEDs though
    - M&M, Modified M&M, Zero Crossing

- Pass in the constellation object as the TED slicer

- Leave the symbol sync blocks’ code alone
  - tag handling, resampler, TED, & loop filter – all done!
Existing Block to New Block

• Polyphase Clock Sync
Existing Block to New Block

- Polyphase Clock Sync comparison (MPSK tutorial)
## Existing Block to New Block

- **Clock Recovery MM, Complex I/O**

### Properties: Clock Recovery MM

<table>
<thead>
<tr>
<th>ID</th>
<th>digital_clock_recovery_mm_xx_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Complex</td>
</tr>
<tr>
<td>Omega</td>
<td>sps</td>
</tr>
<tr>
<td>Gain Omega</td>
<td>beta</td>
</tr>
<tr>
<td>Mu</td>
<td>0.5</td>
</tr>
<tr>
<td>Gain Mu</td>
<td>alpha</td>
</tr>
<tr>
<td>Omega Relative Limit</td>
<td>((sps + \text{deviation}_\text{sps})/\text{sps} - 1.0)</td>
</tr>
</tbody>
</table>

### Properties: Symbol Sync

<table>
<thead>
<tr>
<th>ID</th>
<th>digital_symbol_sync_xx_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O Type</td>
<td>Complex</td>
</tr>
<tr>
<td>Timing Error Detector</td>
<td>Modified Mueller and Müller</td>
</tr>
<tr>
<td>TED Slicer Constellation</td>
<td>digital.constellation_qpsk().base()</td>
</tr>
<tr>
<td>Samples per Symbol</td>
<td>sps</td>
</tr>
<tr>
<td>Expected TED Gain</td>
<td>1.0</td>
</tr>
<tr>
<td>Loop Bandwidth</td>
<td>omega_n_T</td>
</tr>
<tr>
<td>Damping Factor</td>
<td>zeta</td>
</tr>
<tr>
<td>Maximum Deviation</td>
<td>deviation_sps</td>
</tr>
<tr>
<td>Output Samples/Symbol</td>
<td>1</td>
</tr>
<tr>
<td>Interpolating Resampler</td>
<td>MMSE, 8 tap FIR</td>
</tr>
</tbody>
</table>

\(\text{MATH!}\)
Existing Block to New Block

- Clock Recovery MM, Float I/O
Existing Block to New Block

- **MSK Timing Recovery**

<table>
<thead>
<tr>
<th>Properties: MSK Timing Recovery</th>
<th>Properties: Symbol Sync</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td><strong>General</strong></td>
</tr>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>digital_msk_timing_recovery_cc_0</td>
<td>digital_symbol_sync_xx_0</td>
</tr>
<tr>
<td><strong>Gain</strong></td>
<td>I/O Type</td>
</tr>
<tr>
<td>alpha</td>
<td>Complex</td>
</tr>
<tr>
<td><strong>Samples per symbol</strong></td>
<td>Timing Error Detector</td>
</tr>
<tr>
<td>sps</td>
<td>D'Andrea and Mengali Gen MSK</td>
</tr>
<tr>
<td><strong>Error limit</strong></td>
<td>Samples per Symbol</td>
</tr>
<tr>
<td>(sps+deviation_sps)/sps - 1.0</td>
<td>sps</td>
</tr>
<tr>
<td>Output samples per sym</td>
<td>Expected TED Gain</td>
</tr>
<tr>
<td>osps</td>
<td>1.0</td>
</tr>
<tr>
<td><strong>Loop Bandwidth</strong></td>
<td>Damping Factor</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td><strong>Maximum Deviation</strong></td>
<td>Output Samples/Symbol</td>
</tr>
<tr>
<td>deviation_sps</td>
<td>osps</td>
</tr>
<tr>
<td><strong>Interpolating Resampler</strong></td>
<td>Interpolating Resampler</td>
</tr>
<tr>
<td>MMSE, 8 tap FIR</td>
<td></td>
</tr>
</tbody>
</table>

**MATH!**
Usage Hints and Gotchas

- **Easy stuff**
  - Output samples/symbol can be in \([1, 2, 3, 4, 5, 6, \ldots]\)
    - Normally set to 1; or to 2, if upstream from an equalizer block
  - Maximum deviation is in units of samples/symbol
    - Smaller is better for acquiring lock at start of burst
    - Too small misses data when symbol clock is far from nominal

- Tracking resets on a “time_est” or “clock_est” tag
  - time_est tag value is a PMT double
    - Sample offset estimate, in \([-1.0, 1.0]\) samples, relative to tagged sample
  - clock_est tag value is a PMT 2-tuple of doubles
    - Sample offset estimate, in \([-1.0, 1.0]\) samples, relative to tagged sample
    - Symbol clock period estimate, in samples/symbol
  - clock_est tag has priority over time_est tag
Usage Hints and Gotchas

- Input signal conditioning and filtering
  - MSK signals and MSK TEDs don’t use matched filters
    • But a narrow IF filter can be beneficial
  - Input signal should be at a consistent amplitude (e.g. +/- 1.0)
    • AGC
    • TEDs have specific assumptions about input amplitudes !!!
  - Input signal amplitude should match constellation
    • Only for decision directed TEDs: M&M, Modified M&M, Zero Crossing
    • GNURadio’s Constellation Object silently scales your constellation !!!
  - Input signal should normally be NRZ (no DC offset)
  - Input signal should be peaked at symbol centers
    • Except for MSK signals and MSK TEDs
    • Normally accomplished with a matched filter
  - Sync block’s “PFB, MF” resampler can do the matched filtering
    • Except for rectangular pulse filter and a TED that needs a derivative !!!
Usage Hints and Gotchs

- Loop parameters and tuning
  - Use simulation to determine TED gain, $K_{\text{ted}}$
    - Cannot know damping regime without it !!!
    - Ensure TED gain is scaled to the proper units for the loop !!!
  - Start with a critically damped, or over damped, loop
    - Damping factor, $\zeta$, of 1.0, or greater than 1.0
    - An under damped loop usually isn’t desirable for timing recovery
  - Use a Loop BW, $\omega_n T$, in $[0.0, \pi(?)]$, usually closer to 0.0
  - Use simulation to determine optimal $\zeta$ & $\omega_n T$ for best BER vs. Es/No
  - If you just want to play around and can accept suboptimal results
    - Start with $K_{\text{ted}} = 1.0$, $\zeta = 1.0$, $\omega_n T$ = a number close to 0.0
    - Use GUI sliders to control all 3 of those values
    - Send all 4 outputs of the block to a single Time Sink/Scope
    - Adjust $\omega_n T$ slider first, observing the primary, $T_{\text{inst}}$, and $T_{\text{avg}}$ traces
    - Adjust $\zeta$ to 1.3 or 1.5 or 2.0 (or 0.707 or 0.5), and try adjusting $\omega_n T$ again
    - See gr-digital/examples/demod/symbol_sync_test_float.grc
Experimental Tuning Example

- Intentionally terrible loop BW example
TED S-Curve Simulation

- `gr-digital/examples/demod/*_ted_gain.m`
  - M&M TED gain: $K_{\text{ted}} = 0.28271$ sample$^{-1}$
  - Gardner TED gain: $K_{\text{ted}} = 0.11810$ sample$^{-1}$