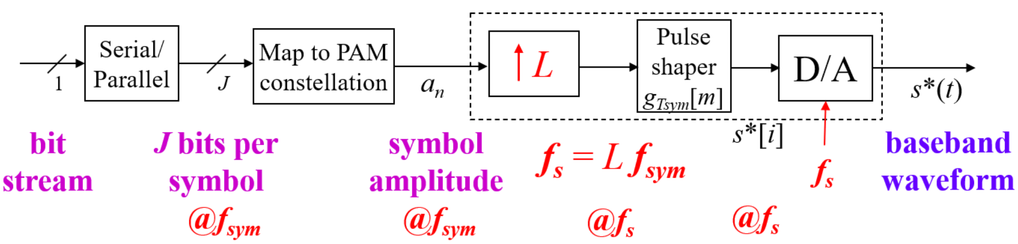
[10:30 – 10:45am] Digital Pulse Amplitude Modulation (Slides 13-3, 13-7,13-8)

Review of the Part 1 for Lecture 13 of Digital PAM from Oct. 28, 2020

- Goal: move bits from transmitter to receiver (over wire, space, underwater, etc)
- We will encode the bits in the amplitude of a pulse
- Each symbol of bits encoded into a unique amplitude by a constellation map
- Amplitudes undergo discrete-to-continuous time conversion
- For PAM transmission, modulate the amplitude of a pulse shape
- The pulse shape guides the interpolation of the D→C conversion
  - Pulse shape is lowpass (sinc, rectangular, raised cosine, etc)
  - Filter should be lowpass with cutoff frequency π/L
  - Impulse response should have value of zero at multiples of symbol time (L)



[10:50am – 11:10am] Polyphase Filter Bank for Pulse Shaping Slide 13-9

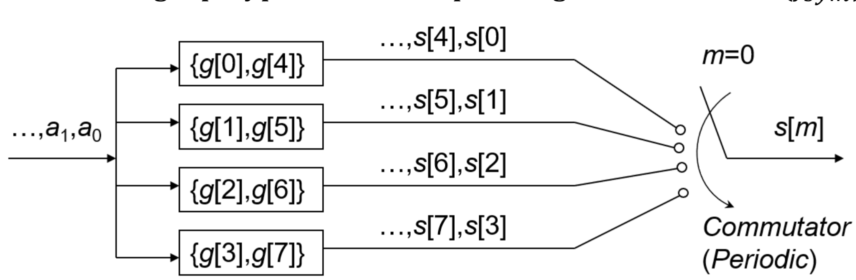
- Applying filter to unsampled signal would result in many multiplies by zeros
  - Results in unnecessary computation and storage

$$s[m] = g[0]x[m] + g[1]x[m-1] + g[m]x[m-2] + \cdots$$

I.C. are zero:

$$s[0] = g[0]x[0] = g[0]a_0$$
$$s[1] = g[0]\underbrace{x[1]}_{0} = g[1]\underbrace{x[0]}_{a_0} + \cdots = g[1]a_0$$
$$s[2] = g[2]a_0$$
$$s[3] = g[3]a_0$$
$$s[4] = g[0]a_1 + g[4]a_0$$

- Each sample requires only two multiplications and one addition
- Can implement using L polyphase filters operating at a lower rate (f_{sym})



- Results in a factor of L saving in multiplies per second
- Also yields factor of L savings in storage
- Even more efficiency can be gained by parallelization

[11:10am – 11:20am] Filter bank example #2 Slide 13-10

- Pulse shaping filter has $N = 24$ samples
- $L = 4$ samples per symbol, resulting in 4 different polyphase filters
- Each filter has $N_g = N/L = 6$ symbol periods per pulse
- Filter #0 consists of first sample of pulse shape and every fourth after that
- Filter #1 consists of second sample of pulse shape and every fourth after that, etc

  Using the more efficient filter bank implementation results in *exactly* the same filtered output (no reduction in signal quality) but reduces the runtime complexity

[11:30am – 12:00pm] Steepest Descent (In Lecture Assignment #4)

- Adaptive methods can be used to correct for channel impairments
- Steepest Descent (JSK Fig 6.15 on page 116)
  - Define an objective function $J(x)$
  - Specify if the goal is to minimize or maximize

Squared error example: $J(x) = \underbrace{(x-7)}_{\text{error}}{}^2$

- x is what we have and 7 is what we want (root of the polynomial in this case)
- If we drive the error^2 to zero, we drive the error to zero.
- Slope at point $x_p = \frac{d}{dx} J(x)\Big|_{x=x_p}$
- To minimize the function, we want to go in the opposite direction of derivative

Steepest Descent Algorithm:

- Start with an initial guess $x[0]$
- Create a new guess $x[k+1]$ by moving in the opposite direction of derivative:

$$x[k+1] = x[k] - \mu \frac{dJ(x)}{dx}\Bigg|_{x=x[k]}$$

- To keep the process stable, we require the stepsize μ to be a small positive value.
- For $J(x) = (x - 7)^2$, the first derivative is $J'(x) = 2(x - 7)$, and the update becomes

$$x[k+1] = x[k] - \mu(2\,x[k] - 14) = (1 - 2\mu)x[k] + 14\mu$$

  - This is a first-order IIR filter with output x[k+1] and input 14μ u[k+1].
  - Real-valued pole location at $1-2\mu$. For BIBO stability, $0 < \mu < 1$.