MILESTONE 2:

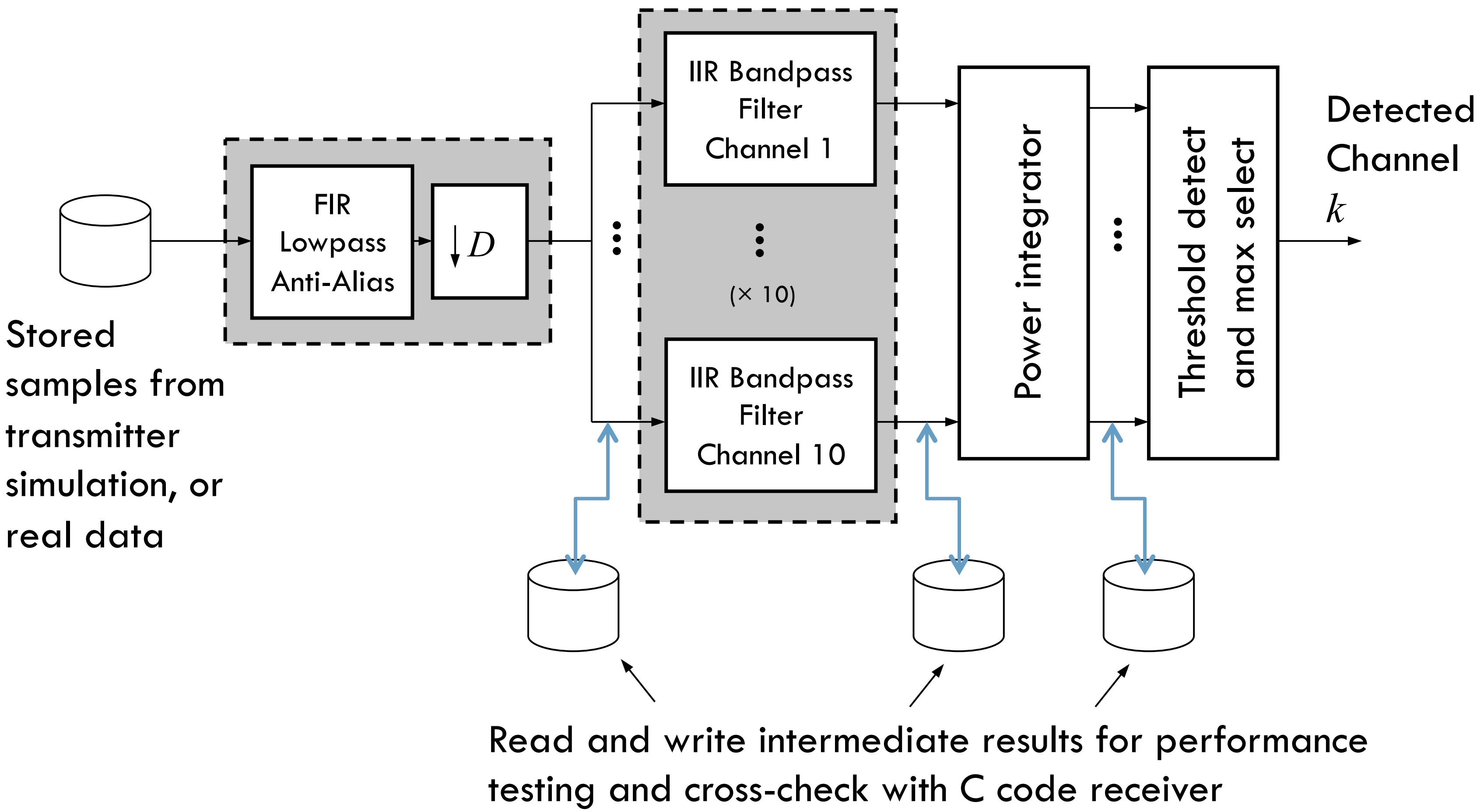
FULL MATLAB SIMULATION OF THE LASER TAG SYSTEM

Why a full MATLAB Simulation?

- First develop your DSP design for Laser Tag transmitter and receiver/detector in MATLAB
- When design is checked out, then translate to C code on the Zybo board environment.
- MATLAB is a more benign development environment:
  - Great debugging tools.
  - Powerful plotting, visualization, & DSP design tools.
- Test you MATLAB design with simulated and real data inputs.
- Generate simulated test data at all stages of C processing
  - Inject simulated test data into your C implementation
  - Compare results sample-by-sample for validation of C code.

Signal Flow Diagram: Receiver/Det.

MATLAB SIMULATION:



Design Specifications

- Transmitter:
  - Transmit 200 ms rect-windowed pulse (one window per trigger pull, 1 second lock out).
  - User selectable pulse carrier frequency (1 of 10): 1.11, 1.39, 1.72, 2.0, 2.27, 2.63, 2.91, 3.33, 3.57, 3.84 kHz.
  - Minimum "chip" increment length: 10 μs (time quantization for waveform).
  - Square wave 50% duty cycle periodic signal.
  - Unipolar, offset "ground" reference (e.g. 0.3V or 0.5V).
  - User controllable signal and random noise power levels to realistically model the communications channel.
  - Store transmit samples to disk file.

Design Specifications

- Receiver / Detector:
  - ADC is 12 bit, unipolar, 0.0-1.0V, 100 ksamp/s.
  - Process a single contiguous block of 1 sec. of data.
  - Convert to double-precision floating point for calculations and remove offset bias.
  - Downsample to 10 ksamp/s with a decimating FIR anti-alias.
  - Bank of 10 IIR bandpass filters with channels centered on the 10 transmit carrier frequencies.
  - Compute total power per channel over 200 ms, updated at least every 100 ms.
  - Threshold power to detect, and find max channel.
  - Store all intermediate outputs to disk. Allow disk input.

Filter Implementation detail

1. Anti-aliasing and decimation with an arbitrary-length decimating FIR filter:

$$y[n] = \sum_{k=0}^{M} h[k]x[Dn-k]$$

Here the length of the filter (M + 1) is larger than the decimation factor D. This allows you to build a better performing filter, but the number of multiplies goes up. M = 20 to 70 should be good. D is just the pointer increment step value you use in the input data array (x[.]) each time you increment the output sample index n by 1 in $y[n]$. A buffer for $y[n]$ would be 1/10 the size of buffer for samples $x[m]$ which spans the same time interval.

2. Bank of Nth order IIR bandpass filters:

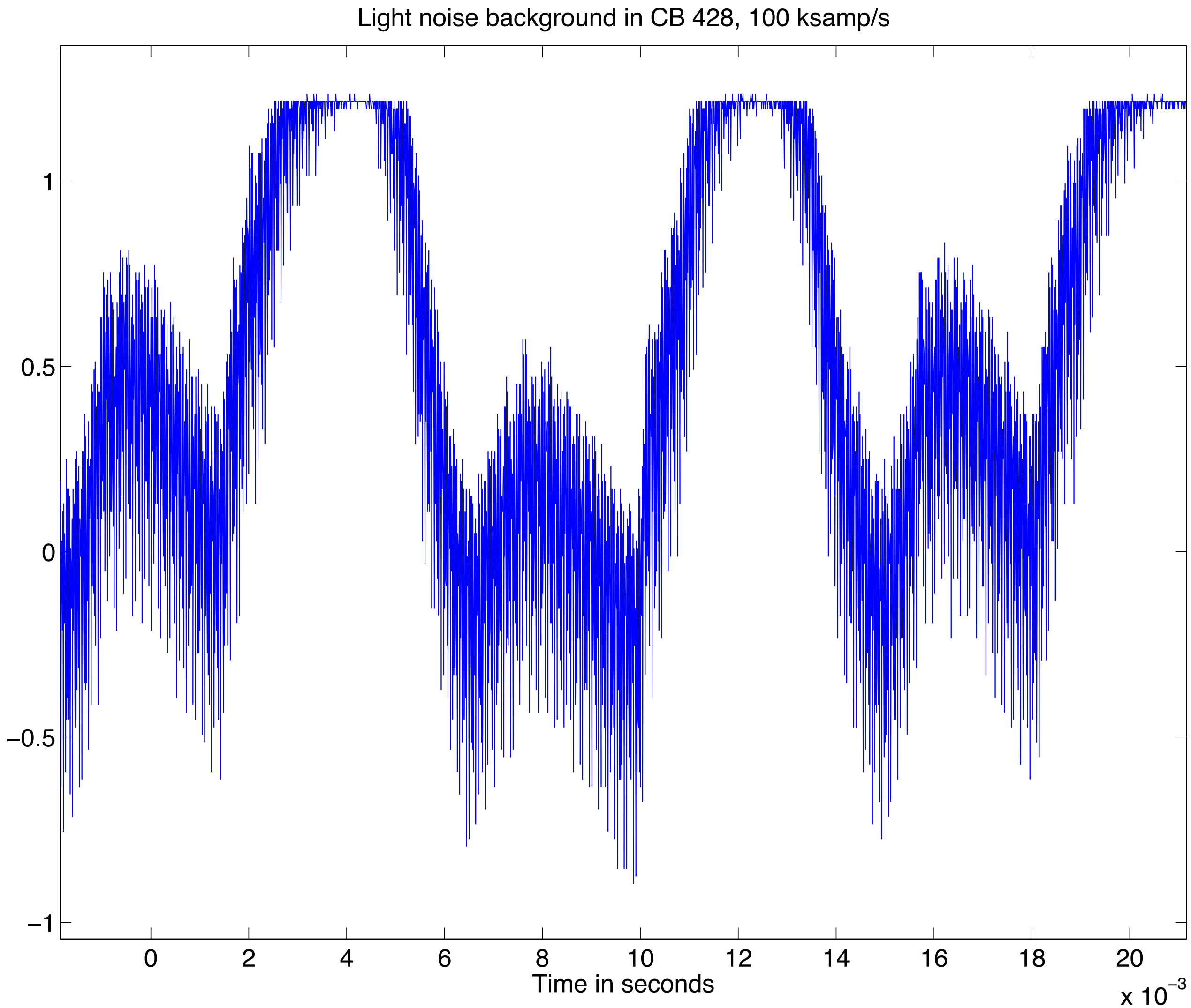
$$z_i[n] = \sum_{k=0}^{N} b_{i,k} y[n-k] - \sum_{k=1}^{N} a_{i,k} z[n-k]$$

for $1 \le i \le 10$ indicating which filter in the bank (i.e. which player frequency band). Use Butterworth design method from ECEN 380, or other IIF BPF design.

“Illegal” MATLAB commands

- Make your MATLAB “real-time” simulation code as C-like as possible.
- For the “real-time” simulation part, you may NOT use:
  - filter.m, conv.m, or similar functions (use simple for-loops instead).
  - Matrix arithmetic, index range manipulation (e.g. A(n:3:N))
  - Anything that does not have a direct C library counterpart
- For signal plotting and analysis you may use any command.
- Filter design tools, file read and write, etc. that will NOT have a zybo C code counterpart, use any MATLAB command.
- Remember: MATLAB uses “1”-based array indexing, C is “0”-based.

The Signal Environment



The Signal Environment