SBND Sensitivity to Supernova Neutrinos

Yufan Qie

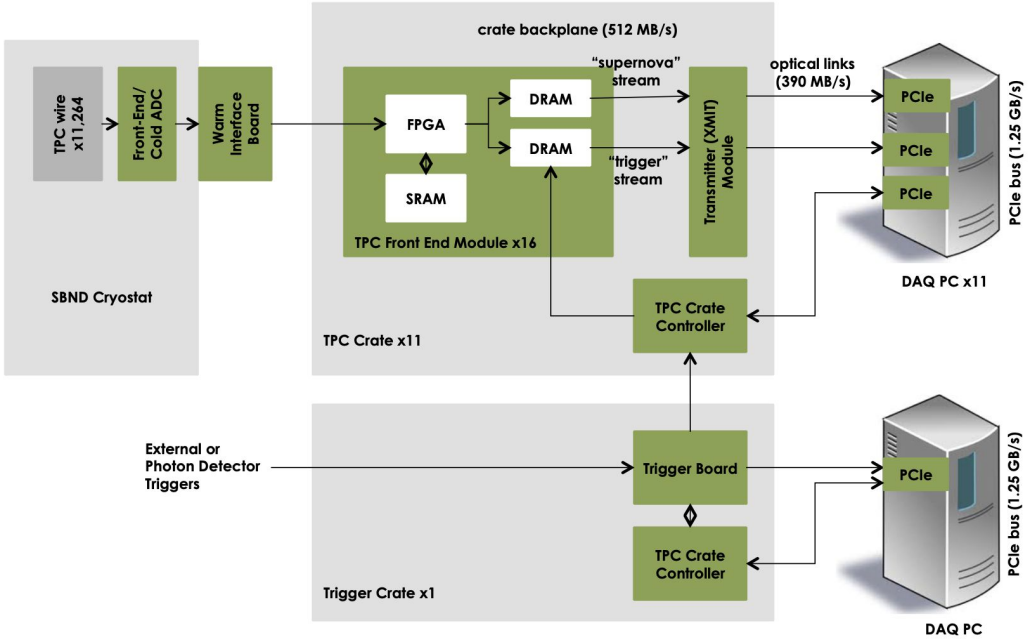
Columbia University

August 6, 2026

How SBND records supernova neutrino data

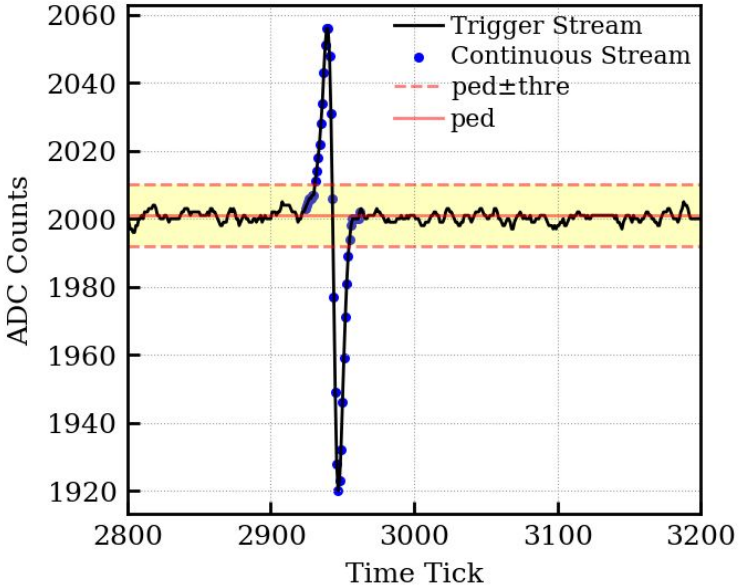
Trigger stream: triggered neutrino events, with a fixed readout window.

Supernova stream: continuously records zero-suppressed TPC data without requiring a trigger.



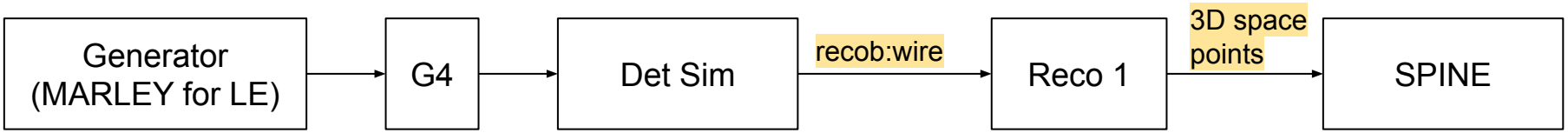
Goal

- Explore **SBND's sensitivity to supernova neutrinos** (SN), with a focus on the impact of **firmware zero-suppression (ZS)** in the continuous SN stream to SN Sensitivity.
  - Firmware ZS retains only **1D regions of Interest (ROI)** on individual channels, defined as threshold-crossing samples plus pre- and post-samples.
  - Typical ROI length: **~70 samples** for induction planes and **~30 samples** for collection plane.
  - Nearby time samples and samples on adjacent wire may be lost.
  - Reconstruction therefore need to be **adapted to work with these ZS ROIs** instead of Raw Digits.
  - The broader goal is to **understand and optimize low-energy reconstruction**.



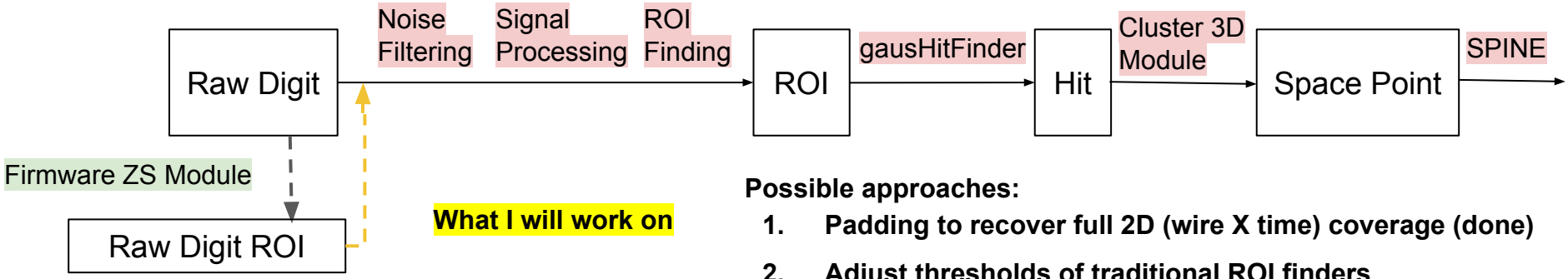
- Start with nue from muon decaying-at-rest (uDAR) simulation.

Pipeline of Simulation & Analysis



- uDAR for first try
- SN

- 2D WireCell signal processing, with both traditional and DNN-based ROI finders. (Gauss for collection plane and DNN for induction planes)
- Can save raw digit before noise filtering and truth information of deposition.



What I will work on

Possible approaches:

1. **Padding to recover full 2D (wire X time) coverage (done)**
2. **Adjust thresholds of traditional ROI finders**
3. **Retune DNN ROI model**

- Simulate what obtained from SN stream
- Currently in LArSoft, can implement in WireCell

Firmware ZS Module: Baseline & Threshold

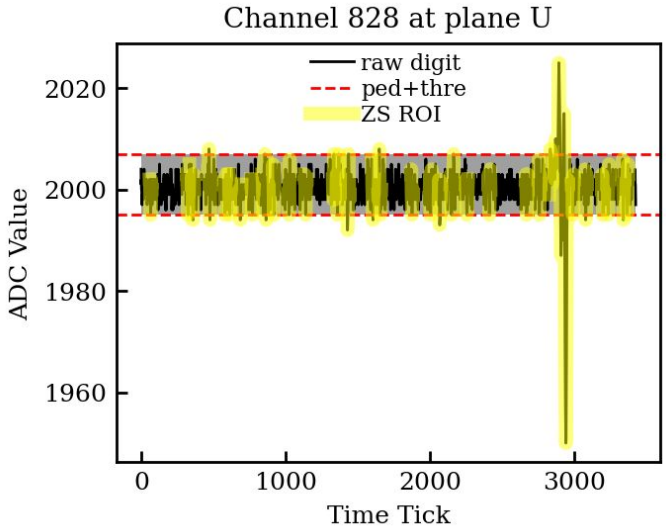
- For SN stream, static channel-dependent baseline and threshold are used, defined in Git.
- In WireCell, constant baseline is set. Not varied channel by channel.
  - Baseline = 2001.0 ADC count for induction planes. ADC > baseline + threshold
  - Baseline = 650.0 ADC count for collection plane. ADC <= baseline - threshold or ADC >= baseline + threshold

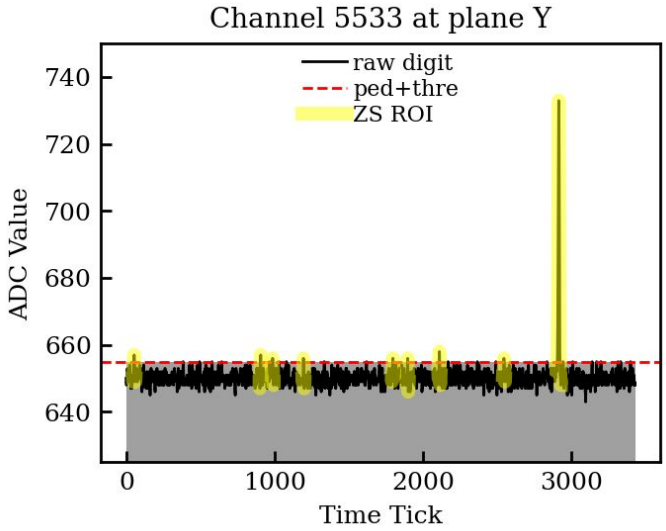
```
cfg/pgrapher/experiment/sbnd/chndb-base.jsonnet:        nominal_baseline: 2001.0,  // adc count [879.5 mV]
cfg/pgrapher/experiment/sbnd/chndb-base.jsonnet:        nominal_baseline: 650,
```

- Therefore, when applying the firmware ZS module to simulated events, using the baseline defined in WireCell and threshold from the SN stream.

Implement Firmware ZS Module

- Revised the uB firmware ZS module [Git] to sbnd configuration and implemented it in sbndcode.
- Developed two analyzer modules to dump raw digits from detsim output and zero-suppressed ROIs from the ZS output to directly validate the module.





Validate Firmware ZS Module

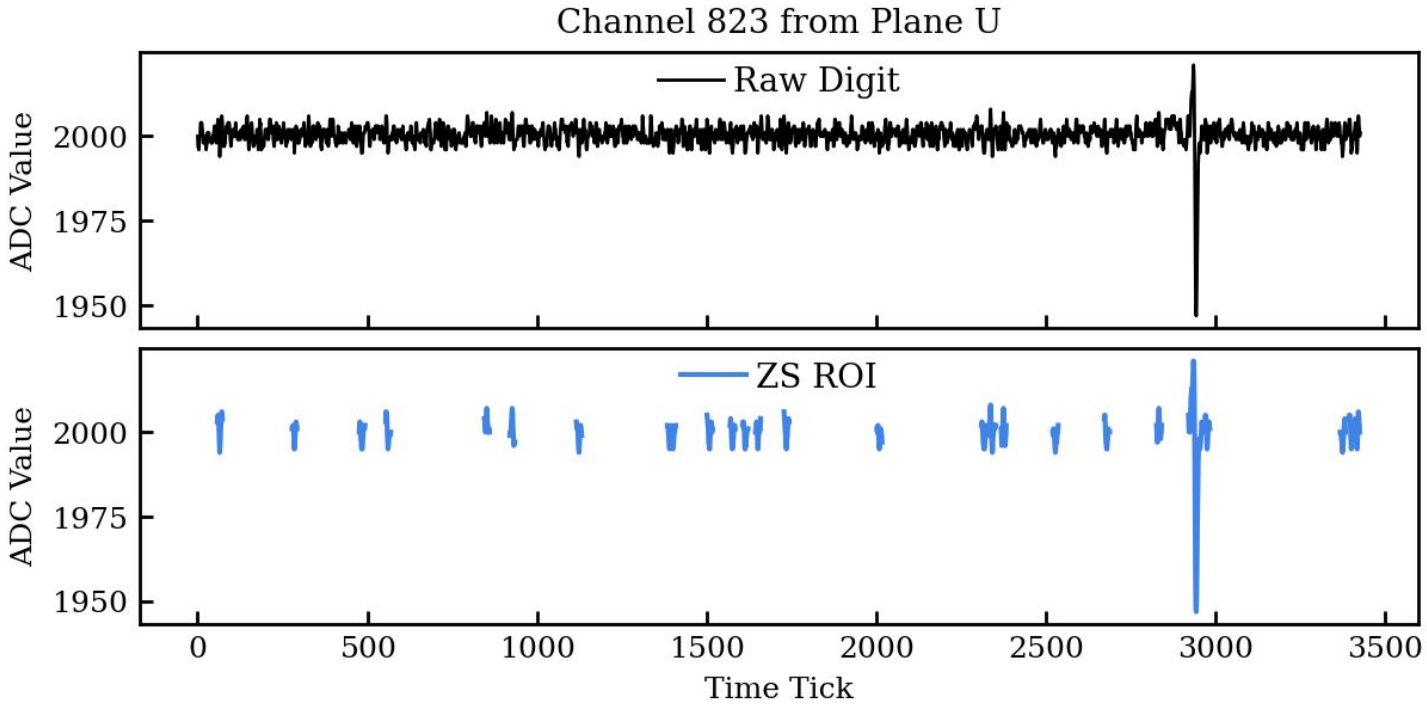
- With raw digits, apply threshold based on what set in SN stream [Git] and baseline set in Wirecell.
- For bipolar (U and V plane), check threshold crossings on both sides: baseline + threshold and baseline - threshold. For unipolar (Y plane), check crossings above baseline + threshold.
-

```
---------Summary---------
Mismatch in plane: 12 --> 0.11 %
No samples above ped+thre and no ZS ROI: 271 --> 2.4 %
Samples above ped+thre and corresponding ZS ROI: 10929 --> 96.92 %
No samples above ped+thre but ZS ROI: 0 --> 0.0 %
Samples above ped+thre but no corresponding ZS ROI: 0 --> 0.0 %
```

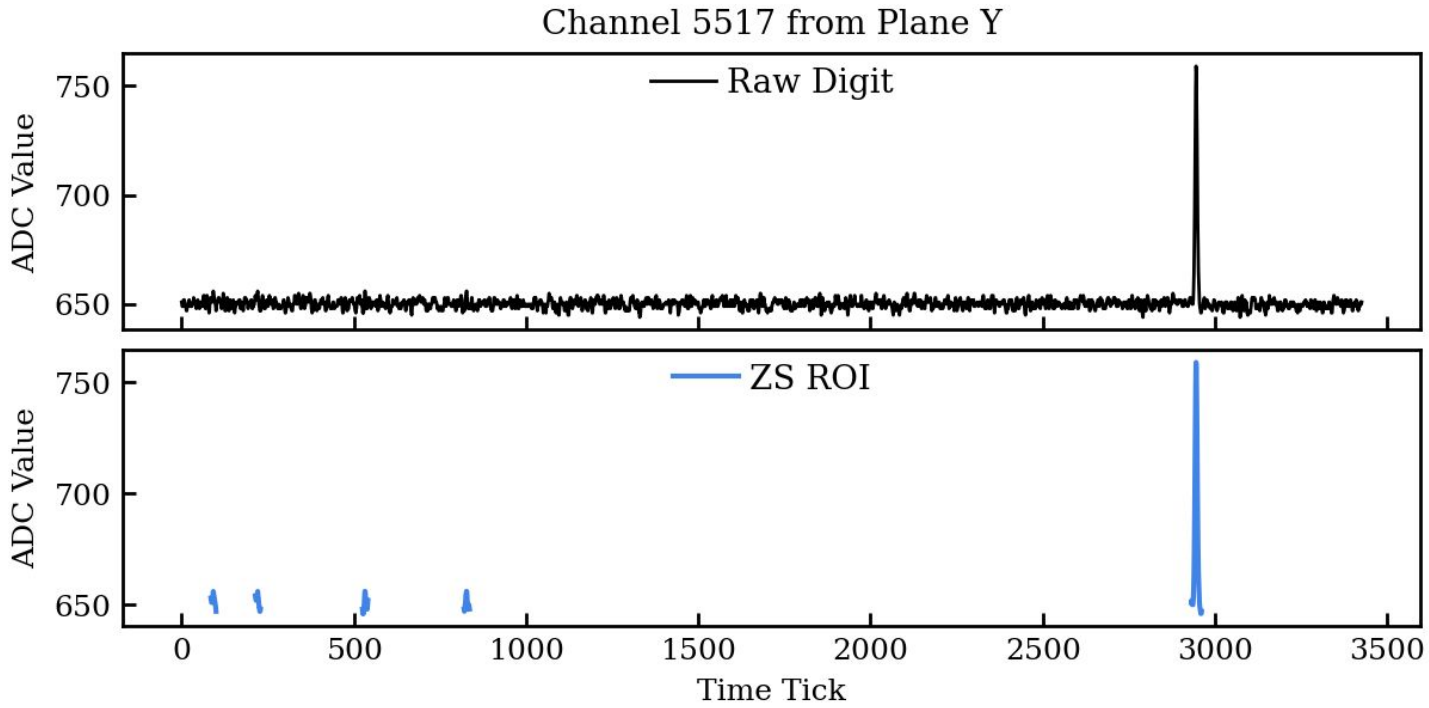
Good Consistency!

- ZS crossing rate is high: 96.92% -> This is expected as the threshold is set such that 97.56% of the ADC values are included.
- Channel mismatch issue:
  - Channels not in use but still in detsim output (Channels in Crate 8 FEM #16).
  - Channels do not exist in the channel map passed to wirecell but still in detsim output.

Firmware ZS Module: Example Output



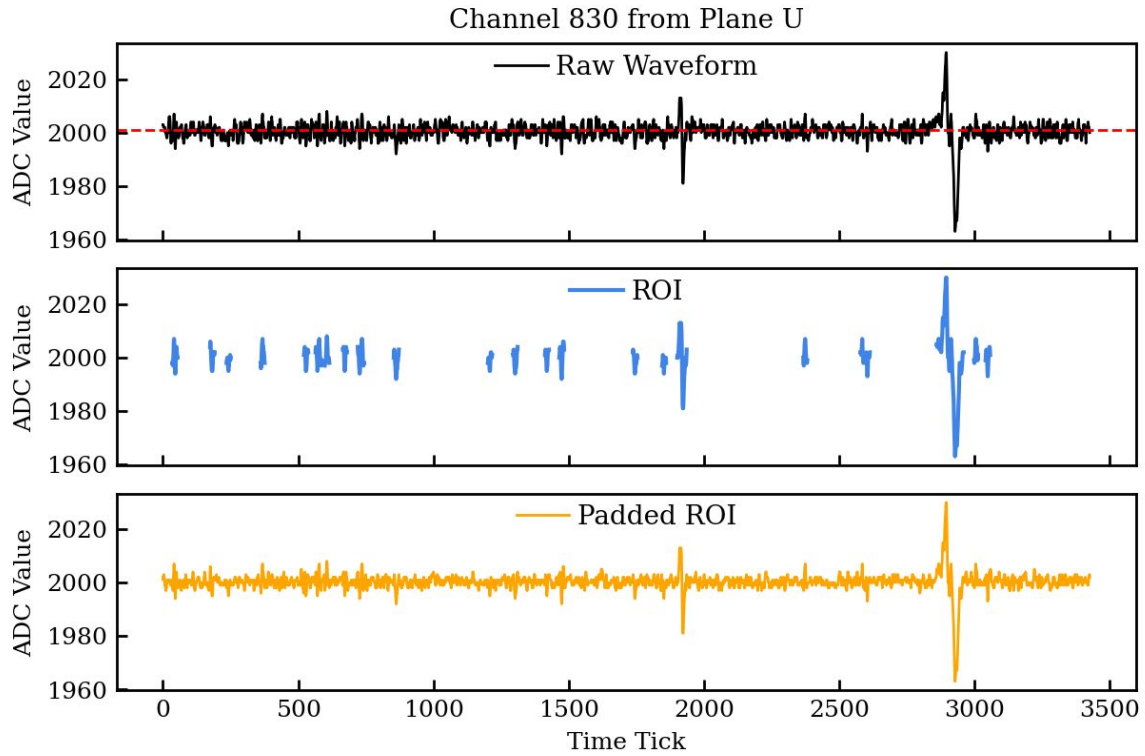
Firmware ZS Module: Example Output



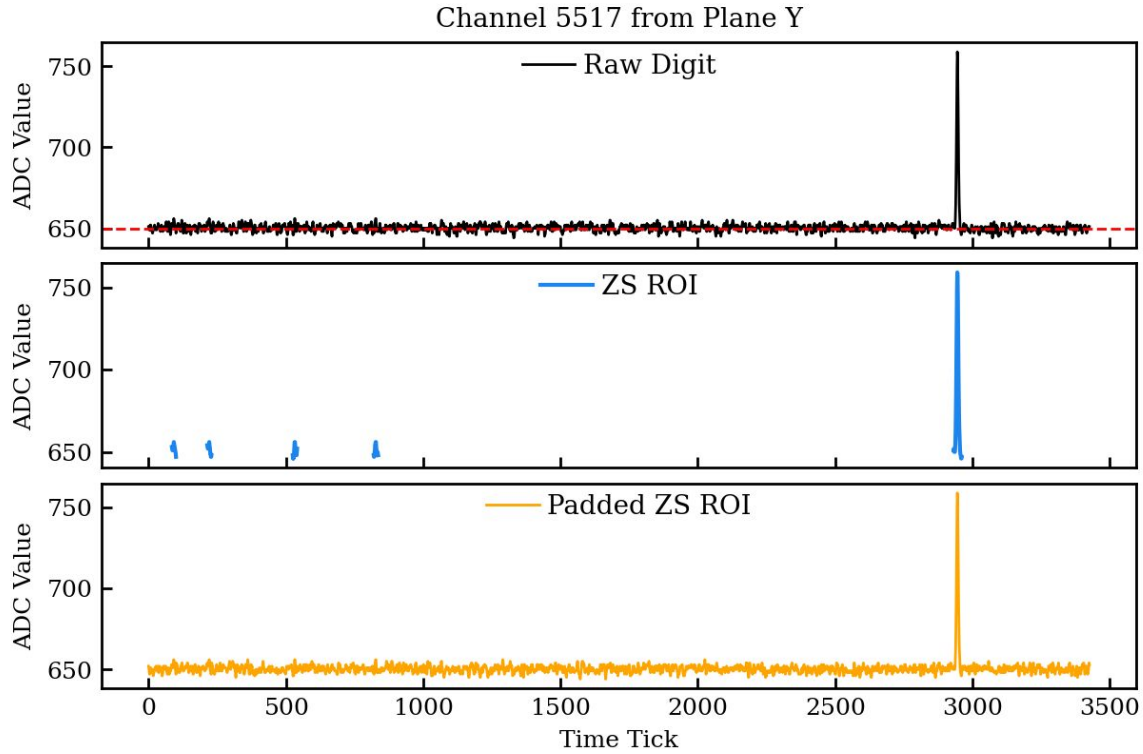
Noise Padding for ZS ROIs

- To process ZS ROIs with WireCell, noise padding restored **full 2D coverage in the wire X time plane**.
  - Noise samples are extracted from simulated raw digits, randomly shuffled, and prepared separately for the induction and collection planes.
  - For each channel, sample outside the ZS ROIs are filled with noise across the full readout window.
  - This is an initial implementation. Future studies may use noise generated from noise model.
- Zero (baseline) padding was also tested, but neither the traditional nor DNN finder identified any signals. It is therefore not considered further.

Noise Padding: Example



Noise Padding: Example



Raw Digit ⟶ WireCell ROIs

Main steps:

1. **Noise Filtering**: remove individual and coherent noise, RC effects, dead channel, etc to prepare the waveforms for signal processing.
2. **Signal Processing**: devonvolves the detector and electronics responses, then identifies ROIs using Gauss and Wiener filters.
3. **DNN-ROI**: uses intermediate signal processing outputs to improve ROI identification.
    - The induction planes use the DNN ROI finder, and the collection plane uses the traditional Gauss ROI finder.

WireCell ROI Outputs: Raw Digit vs Noise Padded

Samples: 10 events, 112,760 channel waveforms.

- 79,360 induction plane waveforms
- 33,400 collection plane waveforms
- ZS retains 109,349 waveforms (97%)

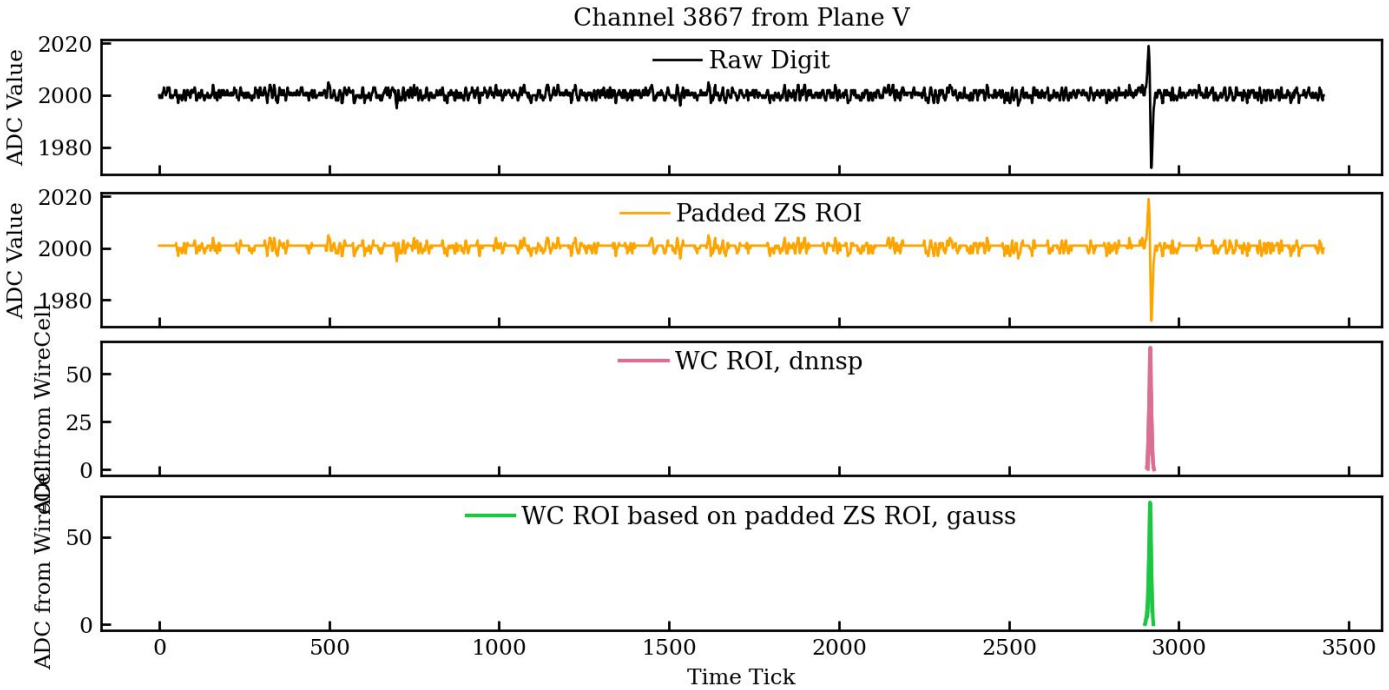
After WireCell,

Input to WireCell	Induction Plane ROIs (DNN)	Collection Plane ROIs (Gauss)
Raw digits	11,165	54,294
Noise padded ZS ROIs	0 (13,970 from Gauss)	59,245

- Noise padding preserves the collection & induction plane ROI response with traditional Gauss ROI finder.
- The DNN ROI finder returns no induction-plane ROIs for noise-padded ZS input.

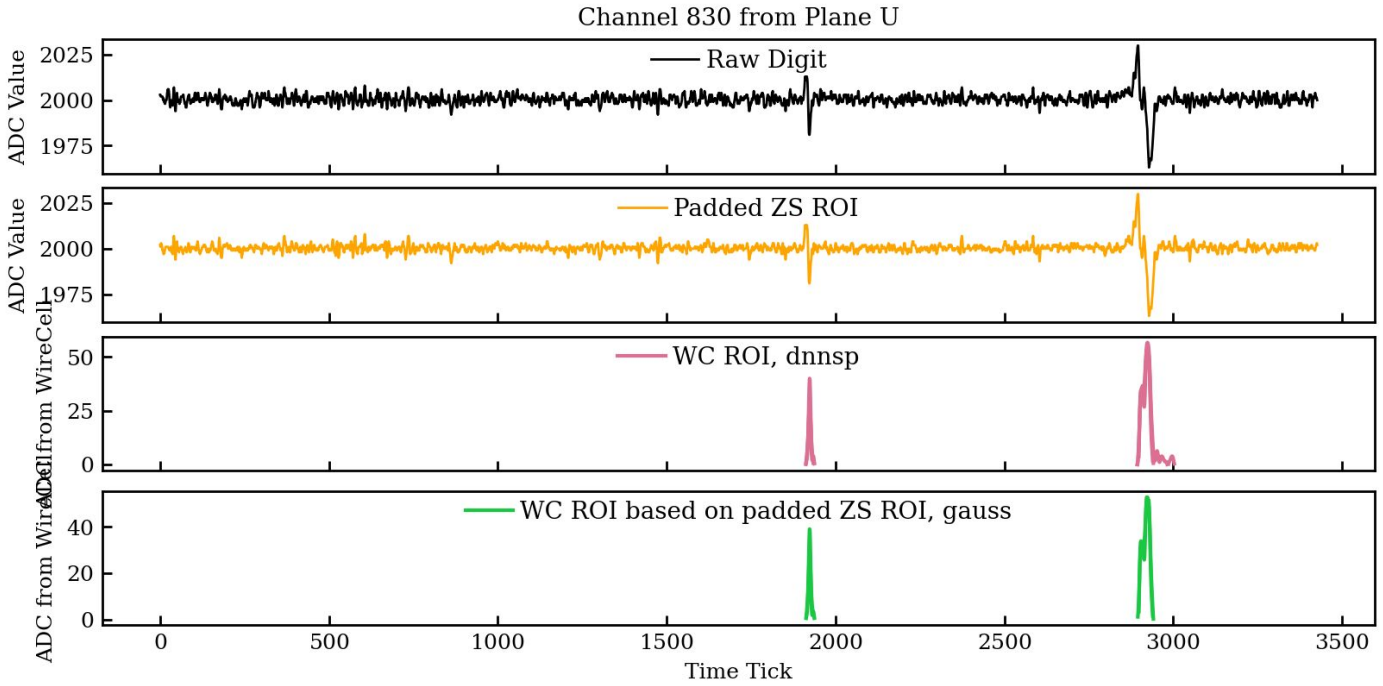
Example of WireCell ROI Output

- Without any change to Wirecell, using noise-padded ZS ROIs as input.



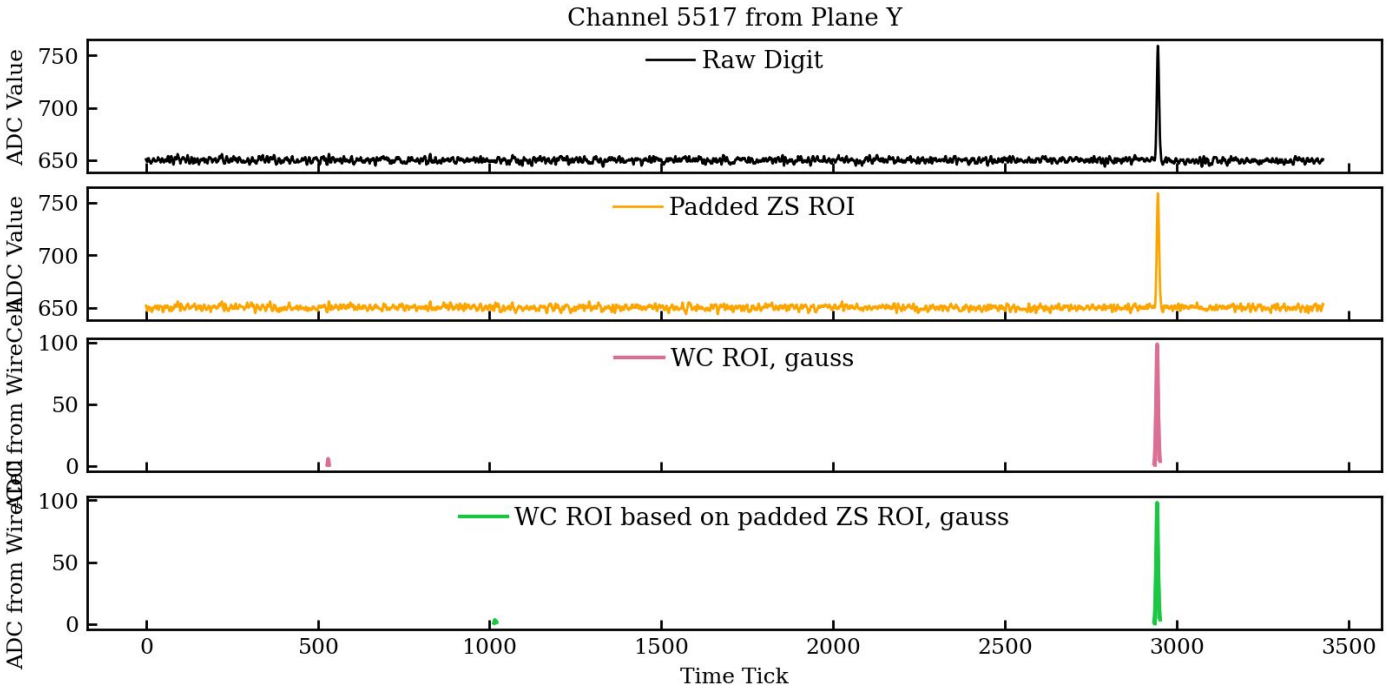
Example of WireCell ROI Output

- Without any change to Wirecell, using noise-padded ZS ROIs as input.



Example of WireCell ROI Output

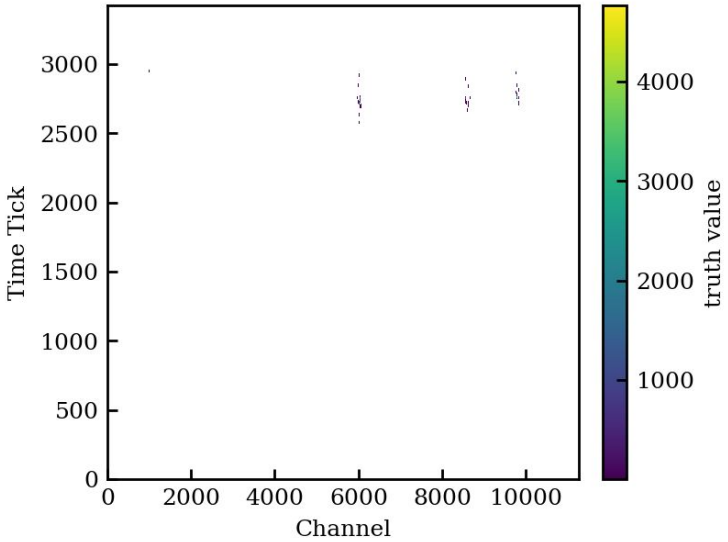
- Without any change to Wirecell, using noise-padded ZS ROIs as input.



WireCell Truth Information

- Truth information
  - Generated with [wcls-sim-drift-depoflux-nf-sp-samples_tru.jsonnet]
  - Saved by HDF5FrameTap as **ionization charge in channel x time space**.
  - Pipeline: simulated depositions -> drift -> wcls_deposetsimchannel_sink -> bi_manifold2 -> truth output.

```
local hio_truth = [g.pnode({
      type: 'HDF5FrameTap',
      name: 'hio_truth%d' % n,
      data: {
        anode: wc.tn(tools.anodes[n]),
        trace_tags: ['deposplat%d'%n],
        filename: "g4-tru-%d.h5" % n,
```



Definition of Efficiency and Purity

- Truth info is provided in **channel X time space**. For initial study, deposition magnitude is ignored.
- A match requires a ROI and a truth deposition to be on the **same channel** and **overlap in time**.

$$N_{\text{matched}} = N\,(\text{channels with truth deposition and overlapping ROI})$$

$$\text{Efficiency} = \frac{N_{\text{matched}}}{N(\text{truth channels})}$$

$$\text{Purity} = \frac{N_{\text{matched}}}{N(\text{channels containing ROIs})}$$

- Efficiency measures how many truth channels are recovered, while purity measures how many selected channels contain true signal.

Efficiency & Purity

- Comparison of WireCell ROI finding performance using raw digits and noise padded ZS ROIs as inputs.

Input	Plane	ROI Finder	Efficiency [%]	Purity [%]
Raw Digit	Induction	dnnsp	28	12
	Collection	gauss	57	4
Noise-padded ZS ROI	Induction	dnnsp	0	0
	Induction	gauss	30	10
	Collection	gauss	62	4

- The low efficiency and purity observed even with nominal raw digits input indicate that the ROI finders need to be **retuned for low-energy signals**, as well as **adapted for noise padded ZS ROIs input**.

Progress & Next Steps

- In the simulation, turned off reco1 & reco2 to focus on **detsim output**.
- Updated the detsim configuration to save **raw digits** and **truth deposition info**.
- Integrated the uB **firmware ZS module** into LArSoft and obtained ZS ROIs.
- Reconstructed full wire X time inputs using **noise-padded ZS ROIs**.
- Compared WireCell outputs for **raw digits** and **noise padded ZS ROIs** inputs and evaluated performance using **efficiency and purity**.
- **Next step: tune the Gauss ROI finder for low-energy signals & noise padded ZS ROIs input.**
- **Questions**:
  - What truth information is normally used for WireCell ROI tuning?
  - Which performance metrics are used?
  - Which traditional Gauss ROI finder parameters should be tuned?

Backup

Noise Filtering

Based on nf.jsonnet, three noise filters:

- One channel noise (Git) -> The only one enabled in WireCell now.
  - Baseline and gain corrections
  - Determine if chirping
  - FFT to frequency domain
  - Check partial noise feature
  - Frequency domain filter
  - Remove zero-frequency component
  - Inverse FFT back to waveform
- Coherent noise
- Sticky Code Miting (ADC tends to get stuck or overpopulate due to electronics)

Signal Processing

Based on sp.jsonnet, signal processing (Git) include:

- Deconvolution
  - Build wire x tick image
  - 2D deconvolution using field/electronics response
- Traditional ROI finding
  - Tight ROI
  - Loose ROI
- Wiener (better S/N separation) and Gauss (better charge reconstruction)

DNN ROI Finder

Based on dnnroi.jsonnet,:

- Applies to U and V planes.
- Takes outputs from sp (intermediate steps from traditional ROI finding) as inputs
  - Three two-dimensional images (ROI filter output, two-plane (MP2) coincidence, and three-plane (MP3) coincidence)
- Performance is evaluated using Efficiency x Purity
  - pi is the predicted ROI probability for pixel i
  - gi is the binary target truth for pixel i

$$\text{Pixel Efficiency} = \frac{\sum_{i=1}^{N} p_i g_i}{\sum_{i=1}^{N} g_i}$$

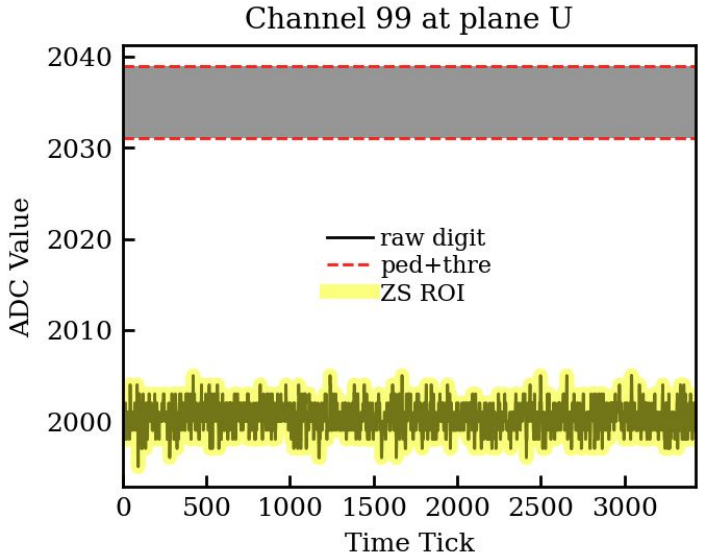
$$\text{Pixel Purity} = \frac{\sum_{i=1}^{N} p_i g_i}{\sum_{i=1}^{N} p_i}$$

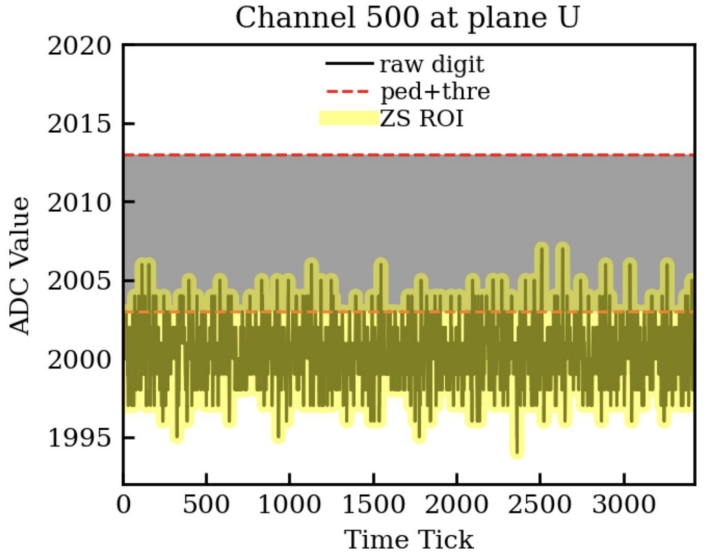
$$\text{ROI Efficiency} = \frac{1}{R} \sum_{r=1}^{R} \min\left(1, \sum_{i \in \mathcal{R}_r} p_i\right)$$

$$\text{ROI Purity} = \frac{1}{\hat{R}} \sum_{r=1}^{\hat{R}} \min\left(1, \sum_{i \in \hat{\mathcal{R}}_r} g_i\right),$$

Validate firmware ZS module

- For more than half of the waveforms, the baseline and threshold values are shifted relative to the waveform baseline, mostly in the U and V planes.





Signal Processing

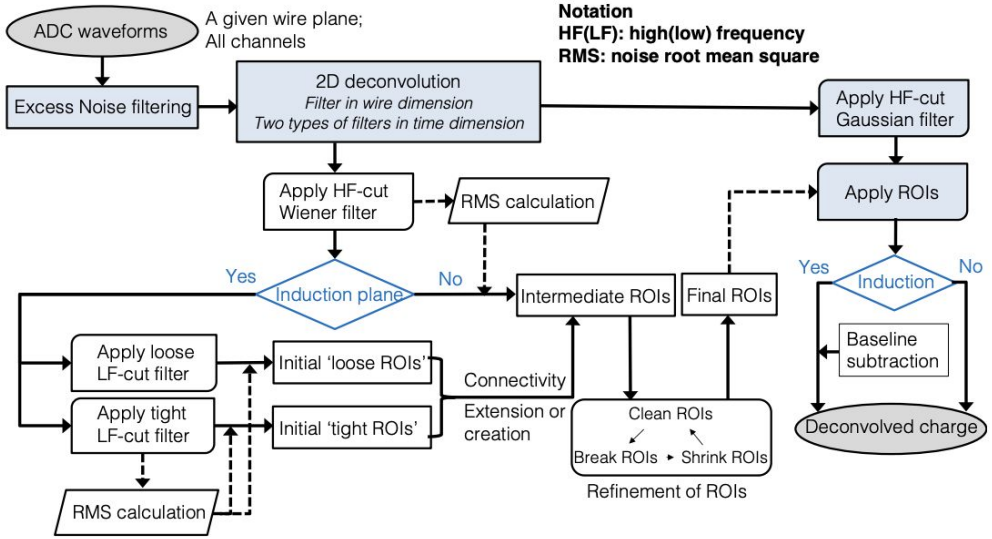


Figure 13: A flow chart of the full chain of signal processing. See text for explanation.

(JINST 13 P07006 (2018))

At SBND,

- This "Apply ROI" part is replaced by DNN ROI finding.
- DNN ROI finding uses some outputs of 2D deconvolution as inputs.