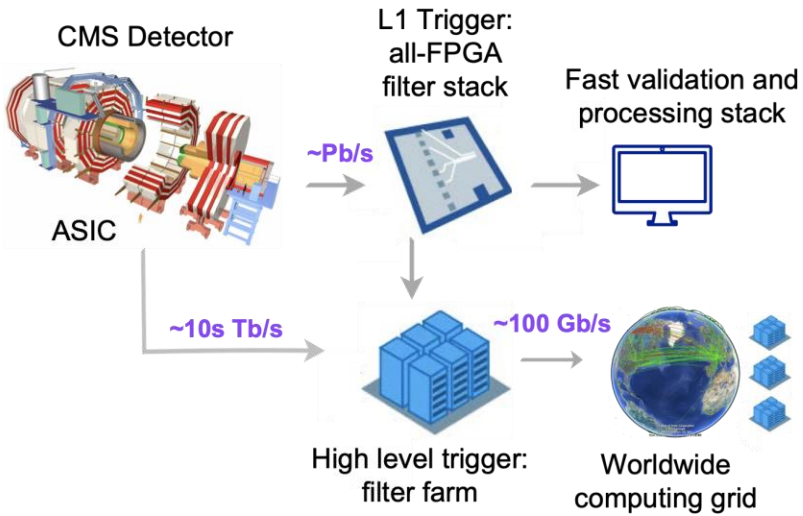
Frontend implementation of AI-machine learning neural networks for in-pixel radiation-hard edge compute

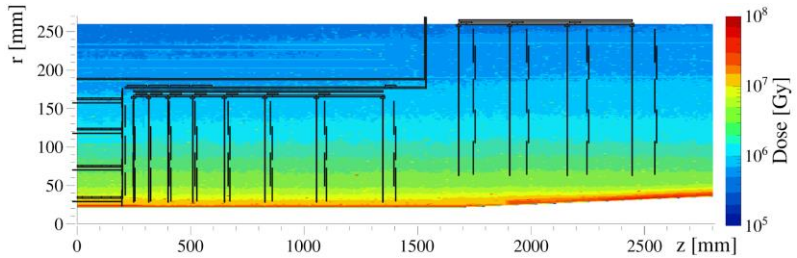
Farah Fahim on behalf of Smart Pixels collaboration

April 2023

Case Study I: 28 nm implementation for CMS Phase III R&D – reconfigurable NN for data filtering in-pixel

LHC Grand challenge





- Extreme Environment

  Total ionizing dose : ~ 1 Grad

  Single event effects : Flux of > 20 MeV hadrons = 10 MHz/cm^2. Target 1 error every ~10s

- Data Volume

  1 billion sensor channels and 1 billion collisions/sec producing raw data at >Pb/s

  More than 99.9% of data generated does not provide useful physics information

- Occupancy ~ 3 GHz/cm^2 in vertex detectors

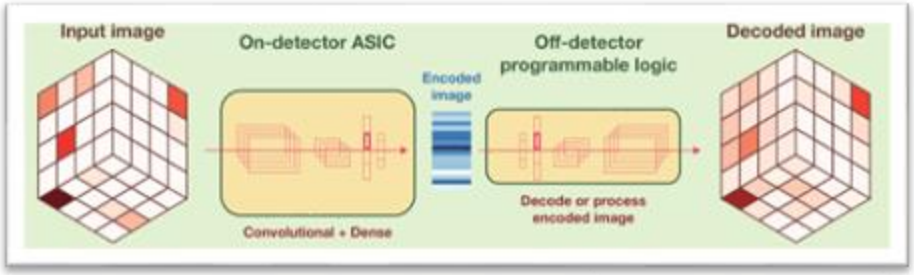
  Requires more granular detectors → increasing data volume, power consumption, more complex on-detector electronics. **Per chip ~ 1Mpixels; > 1 Tbps of zero-suppressed data.**

Fermilab

Edge computing

What is the best approach?

- Data compression
- Data filtering
- Data featurization



Where should the algorithm be implemented?

- **Near sensor: AI/ML mixed-foundry chiplet:**

  Standalone general purpose

  Does not overcome IO bottleneck (1 Tbps off-sensor)

- **Edge of sensor/ Readout integrated circuit:**

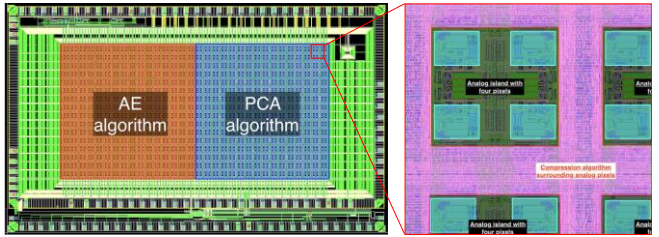
  Easier implementation (larger area and power), optimally utilizes chip bandwidth

  On-chip transfer

- **In-pixel: At the data source:**

  Minimizes data movement

  Power and area constraint



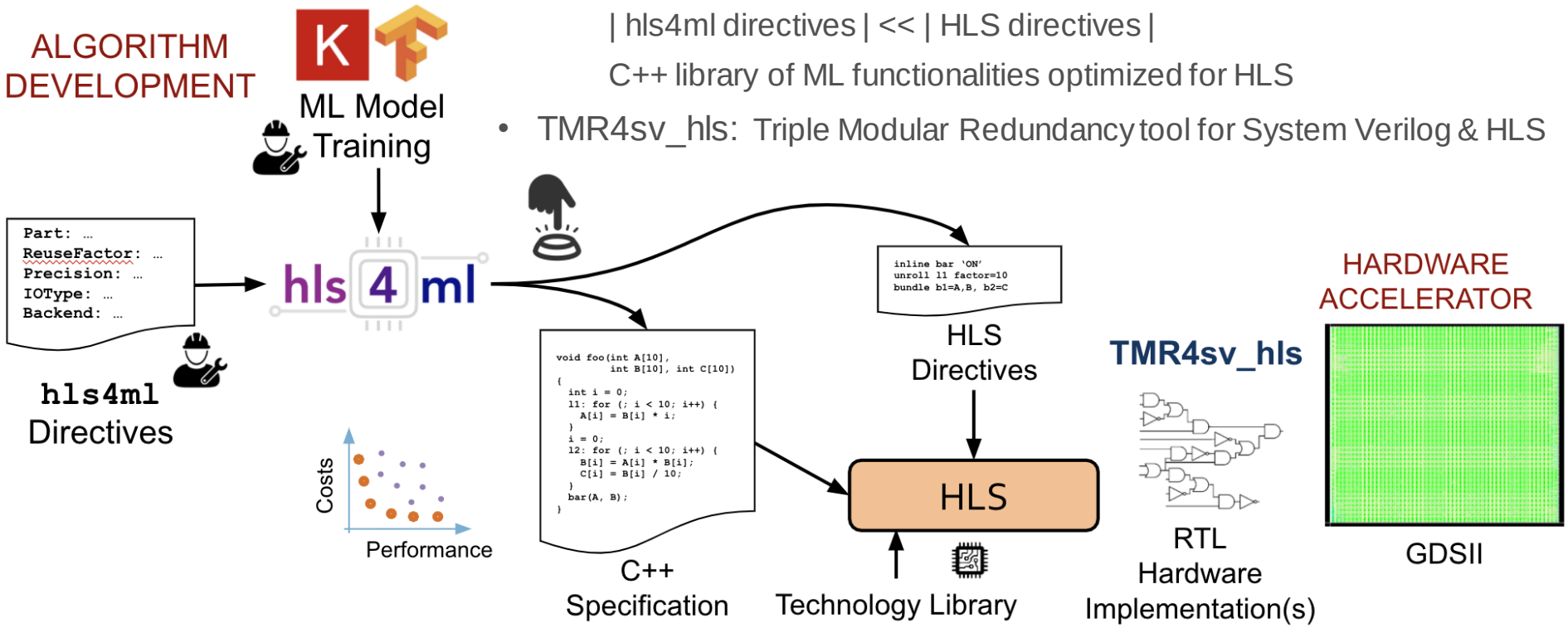
Fermilab

Design Methodology: Physics driven hardware co-design

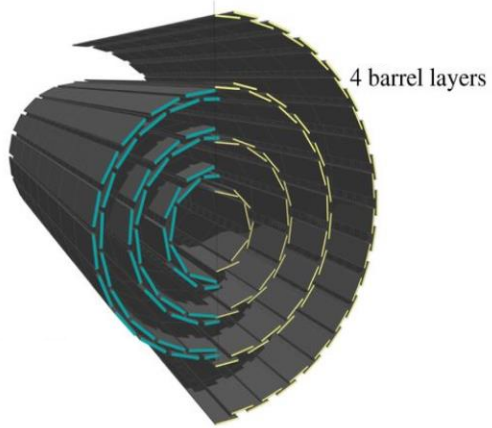
- Algorithm development based on Physics data
- **hls4ml** simplifies the design of on-chip ML accelerators

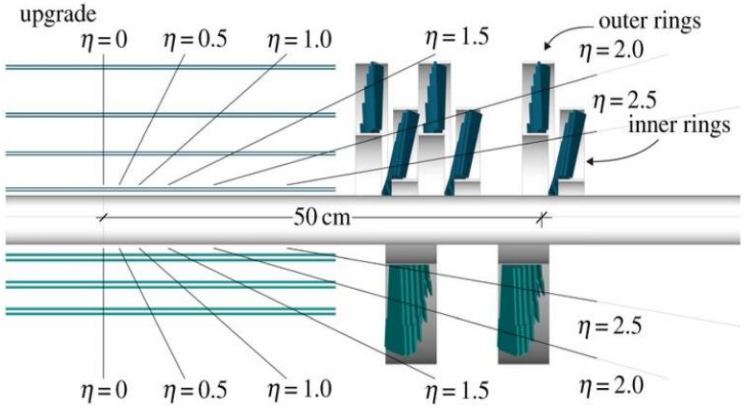
  | hls4ml directives | << | HLS directives |

  C++ library of ML functionalities optimized for HLS
- TMR4sv_hls: Triple Modular Redundancy tool for System Verilog & HLS

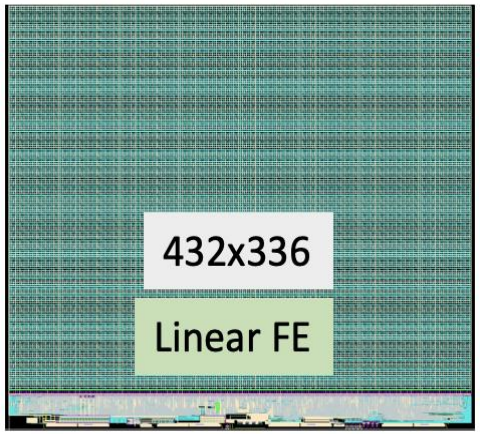


LHC Pixel detector





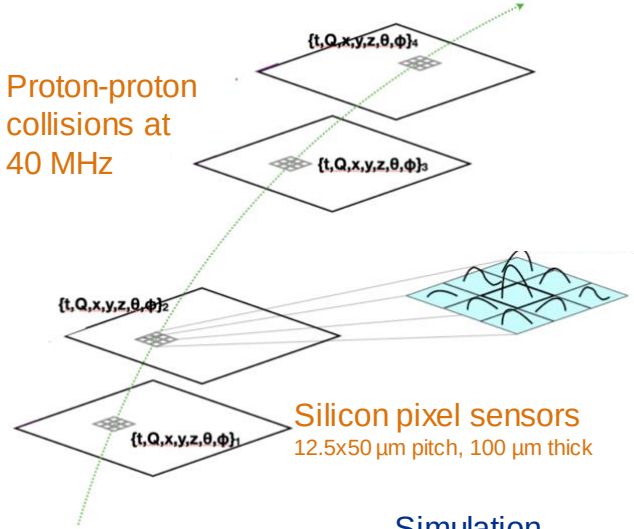
- 4 layers closest to the beam pipe
- Vertex position for high momentum tracks (smaller pixels => higher resolution)
- ~5B pixels across ~30,000 chips
- R&D for Phase II started in 2012; chips in production ~ 2022
- 20B pixels; starting R&D 2022 (expected upgrade in 2034)



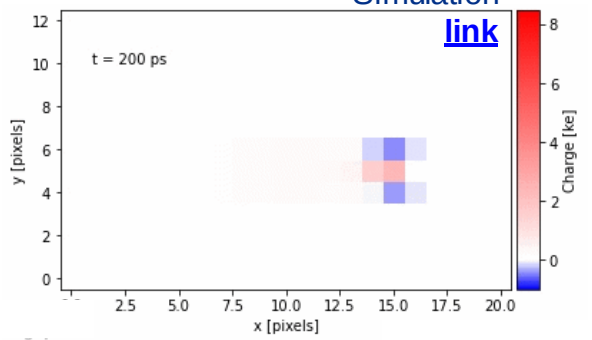
Technology	65nm CMOS	28 nm CMOS
Pixel size	50x50 µm²	25x25 µm²
Pixels	394x400 = 157.6k	788x800 = 0.63M
Detection threshold	~1000e-	~500e-
Hit rate	< 3GHz/cm²	< 3GHz/cm²
Trigger rate	1MHz	40MHz (?)
Digital buffer	12.5 µs	(?)
Readout data rate	1-4 links @ 1.28Gbps	1 photonic link @ 30 Gbps
Radiation tolerance	500Mrad at -15°C	1Grad at -15°C
Power	1 W /cm²	1 W /cm²

Fermilab

Concept: Real-time tracking



Simulation link



Cluster shapes and Pulse information for filtering out low p_T particles

- NN classifier identifies and saves clusters from tracks with $p_T > 2$ GeV
- ≥ 95% data reduction by saving only high p_T
- Low power implementation

Compact algorithms for data reduction through featurization

- **Predict** physics information (x,y,θ,ϕ) and **meaningful error** (UQ) on particle position, angle
- Potential for reduction of track seeds ⟶ saves time & computing resources down the line

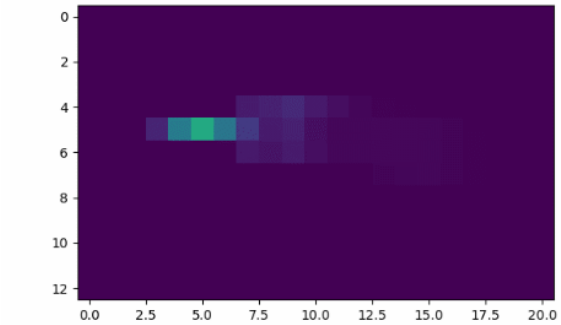
Technology development to enable on-sensor computing

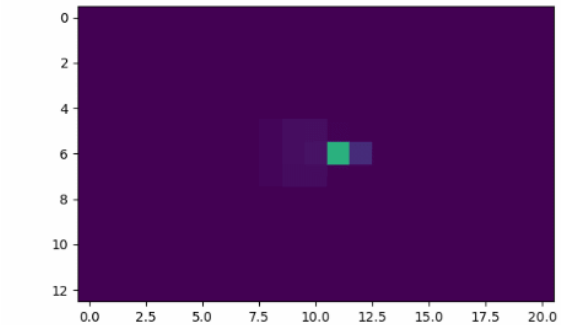
- Ultra low power in-memory compute chips
- 3D integration for optimized data processing
- Leverage emerging technologies such as novel CMOS compatible memristors

Fermilab

Algorithm development for momentum classification

Smart pixel dataset

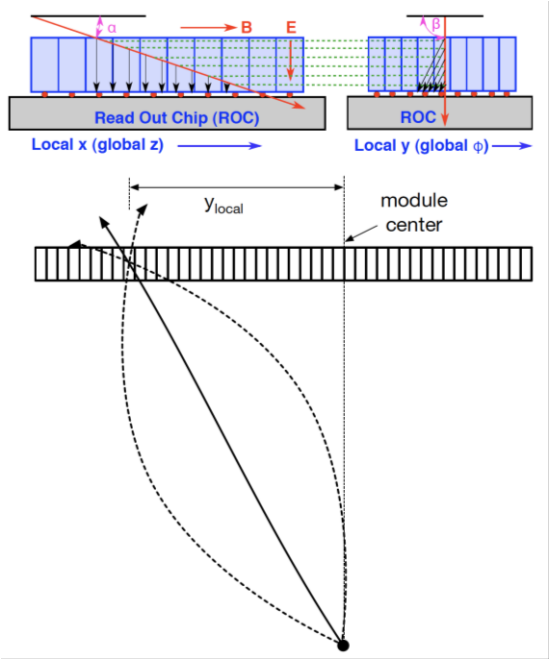


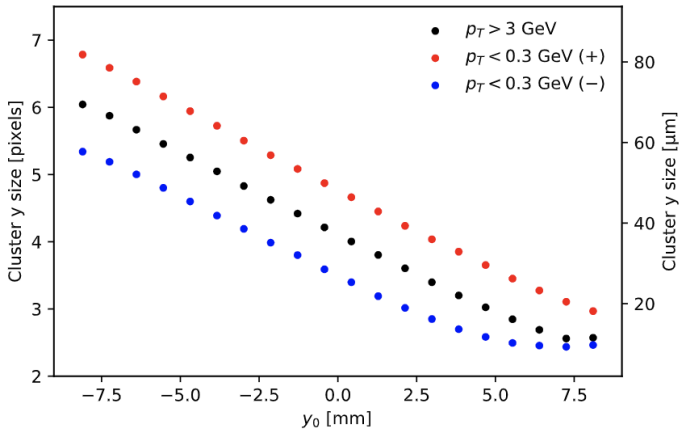


Dataset available on zenodo

- State-of-the-art dataset for developing algorithms for implementation on-ASIC
- Simulated MIP interactions in a **futue pixel detector**
  - Located at radius of 30 mm
  - 3.8 T magnetic field
  - 50 x 12.5 µm pixels
  - 100 µm thick sensor
  - Time steps of 250 picoseconds
- Use **track parameters taken from CMS data** in Run 2
  - Simulation with time-sliced PixelAV, including E field and weighting field

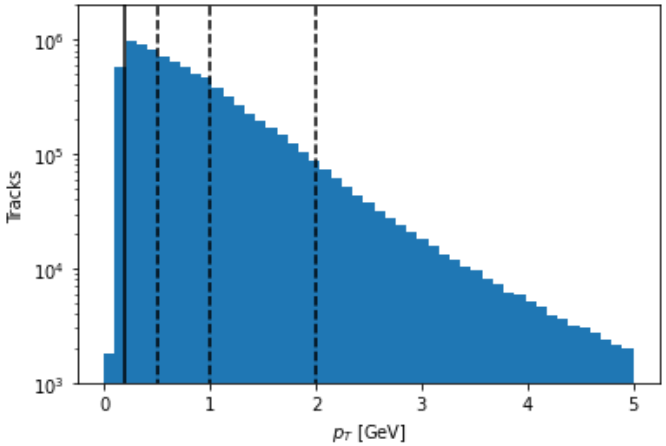
Co-relation between cluster shape and momentum





- 1/3rd of all clusters are associated with tracks
- Untracked clusters are mainly due to loopers, broken clusters & other backgrounds
- Low p_T positively charged tracks increase cluster size
- Low p_T negatively charged tracks decrease cluster size
- **Strong correlation between y-size and y-local on the module**
- **Multi-level classifier for selecting high momentum tracks**

Data filtering to reject 95% of data



Conservatively reject:

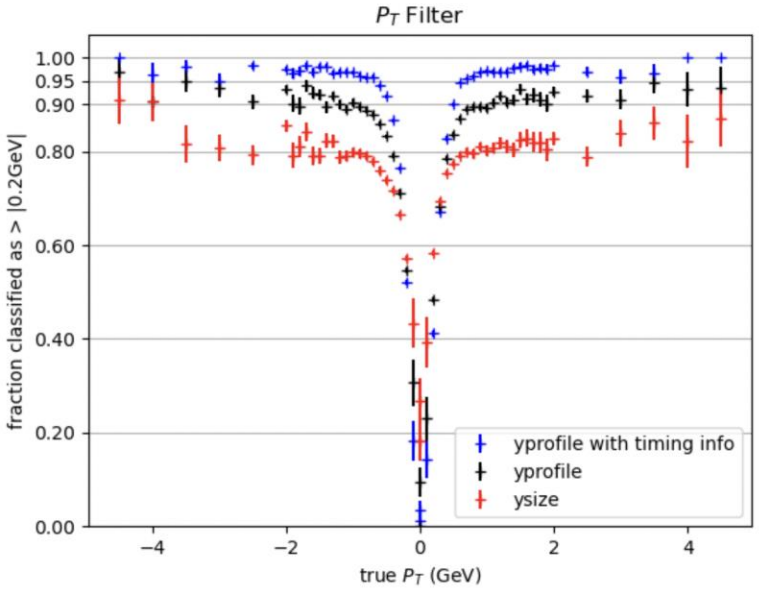
< 0.2 GeV	≥ 6%
< 0.5 GeV	≥ 36%
< 1 GeV	≥ 70%
< 2 GeV	≥ 94%

- Only hits that are reconstructed into tracks are simulated
- Rejecting lower p_T hits (often single pixel) will increase the fraction rejected
- Generating data sets associated with untracked clusters, noise hits and other background

Fermilab

Morris Swartz, Jennet Dickinson

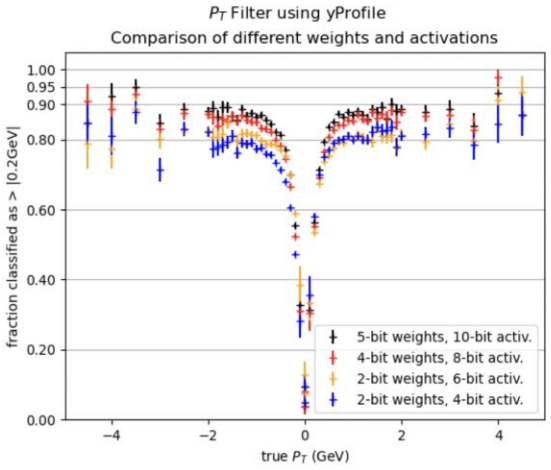
Algorithm performance

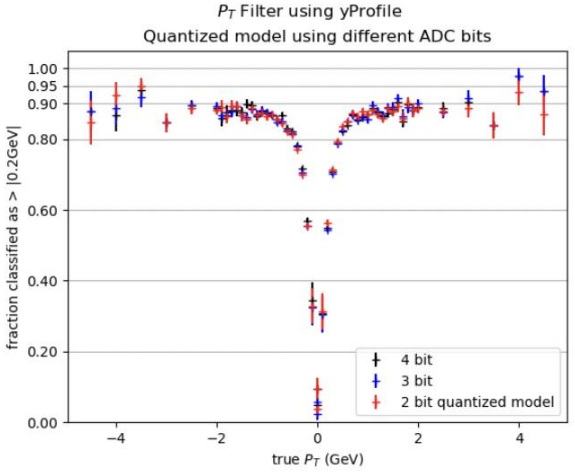


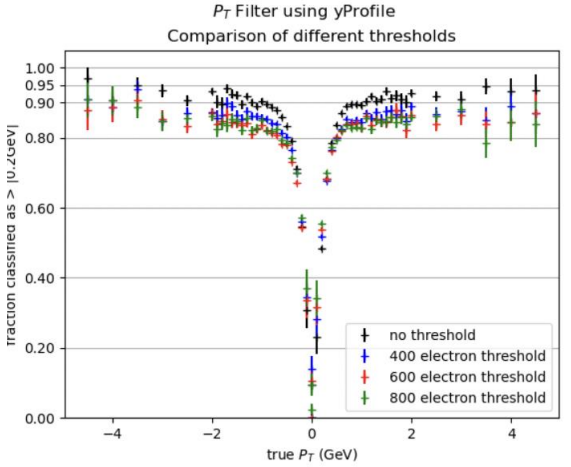
- Model 1: y-size: count no. of pixels with hit
- Model 2: y-profile: add total charge accumulated across all pixels in the y-direction
- Model 3: y-profile with timing: add total charge accumulated across all pixels in the y-direction every 250ps – 8 data points over 4ns of charge integration

Model	Data Reduction	False Positives	False Negatives
Model 1	77.9 %	13.4%	20.7 %
Model 2	65.8 %	10.9%	13.4 %
Model 3	57.4 %	8.9 %	8.1%

Algorithm co-design with frontend architecture



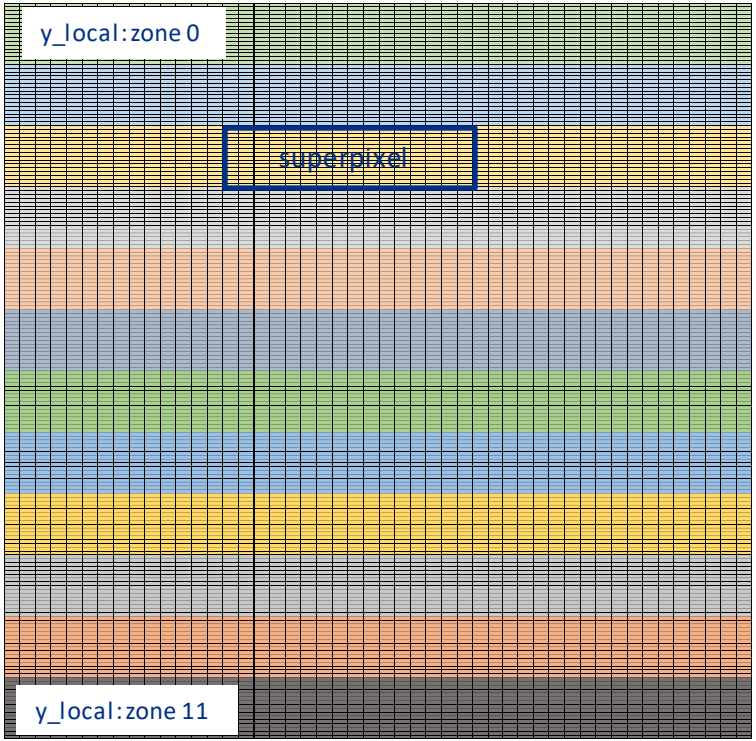




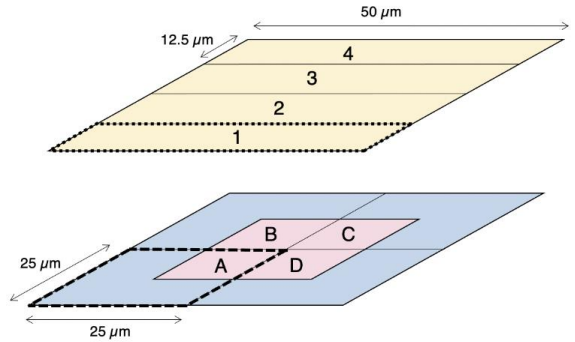
- Lowering noise threshold increases slightly improves efficiency
- Increasing no. of bits does not improve efficiency

Threshold $[e^-]$	0.2 GeV	0.5 GeV	1 GeV	2 GeV
0	78.2 %	88.2 %	91.0 %	92.6 %
400	77.5 %	85.6 %	87.5 %	88.3 %
600	74.4 %	82.2 %	84.3 %	85.9 %
800	75.3 %	82.2 %	84.3 %	86.0 %

Retraining local algorithms



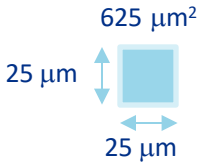
- Sensor pixels can be divided into various zones to exploit the correlation with y_local
- The algorithm is reprogrammable and weights are trained based on position dependent data
- 2×2 cm sensor module: 3200 rows & 100 cols
- Every region needs minimum 16 rows and maximum ~ 64 rows
- 16×16 sensor pixel = 8×32 ROIC pixels



Fermilab

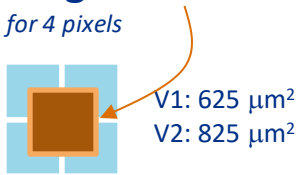
SuperPixel concept: Algorithm co-design

Single pixel area



Analog area

for 4 pixels

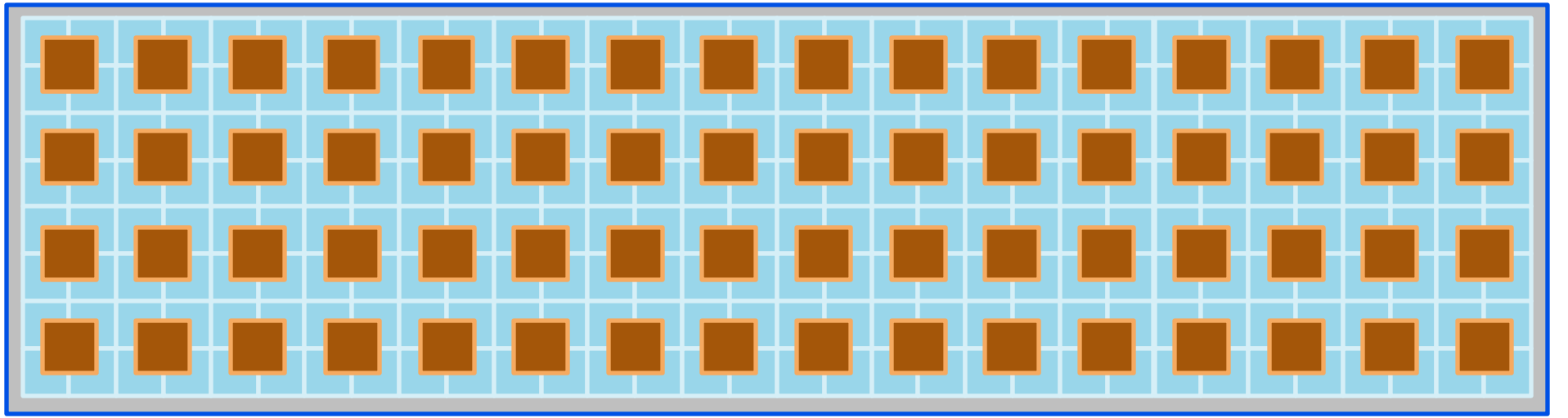


V1: 625 μm²
V2: 825 μm²

Total Digital area

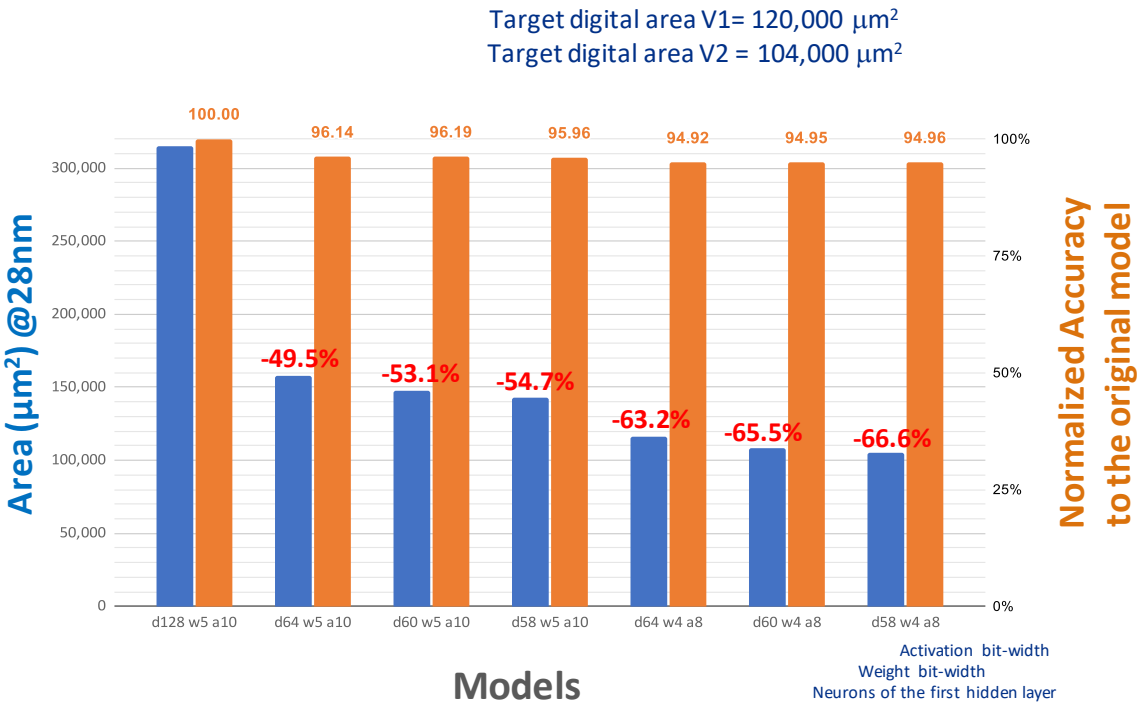
for 256 pixels

V1: 64(625×4 – 625) = 120,000 μm²
V2: 64(625×4 – 825) = 104,000 μm²



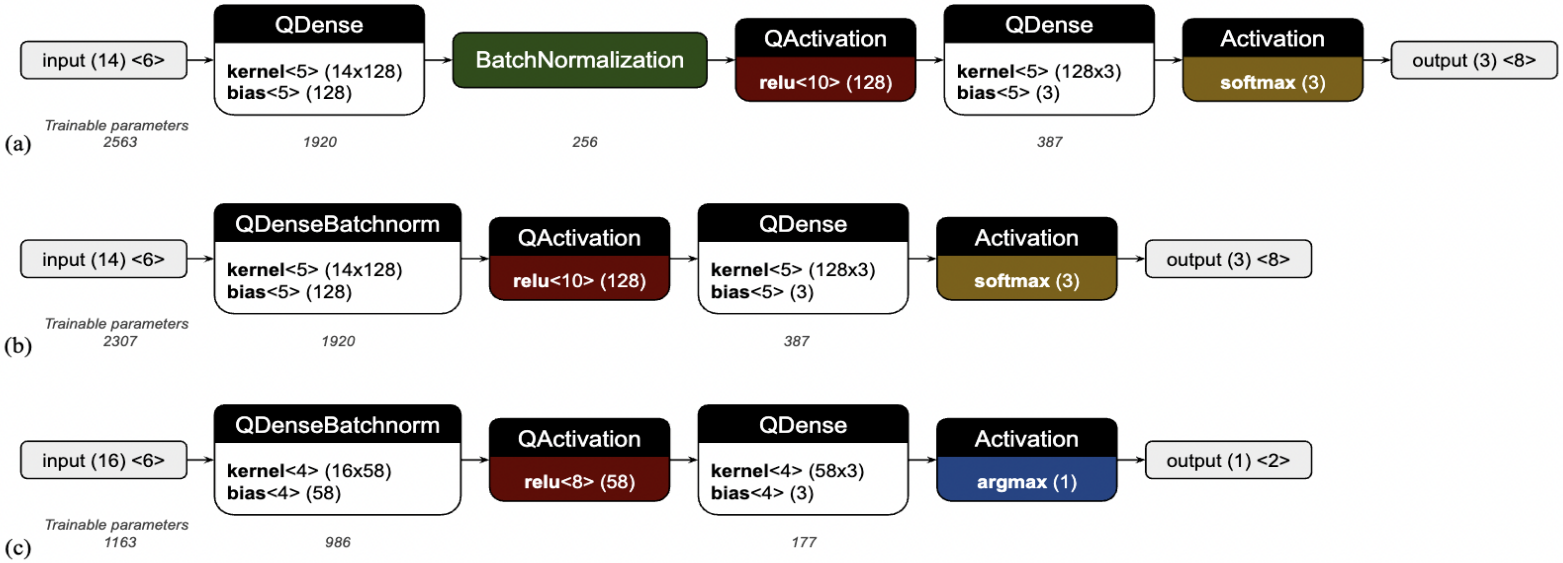
Fermilab

Design space exploration for hardware co-design



- Original model is too large for the given area
- Exploring area reduction without significantly decreasing the model accuracy
- Reducing the number of neurons in the first hidden layer and the bit-width of the weights and activation function

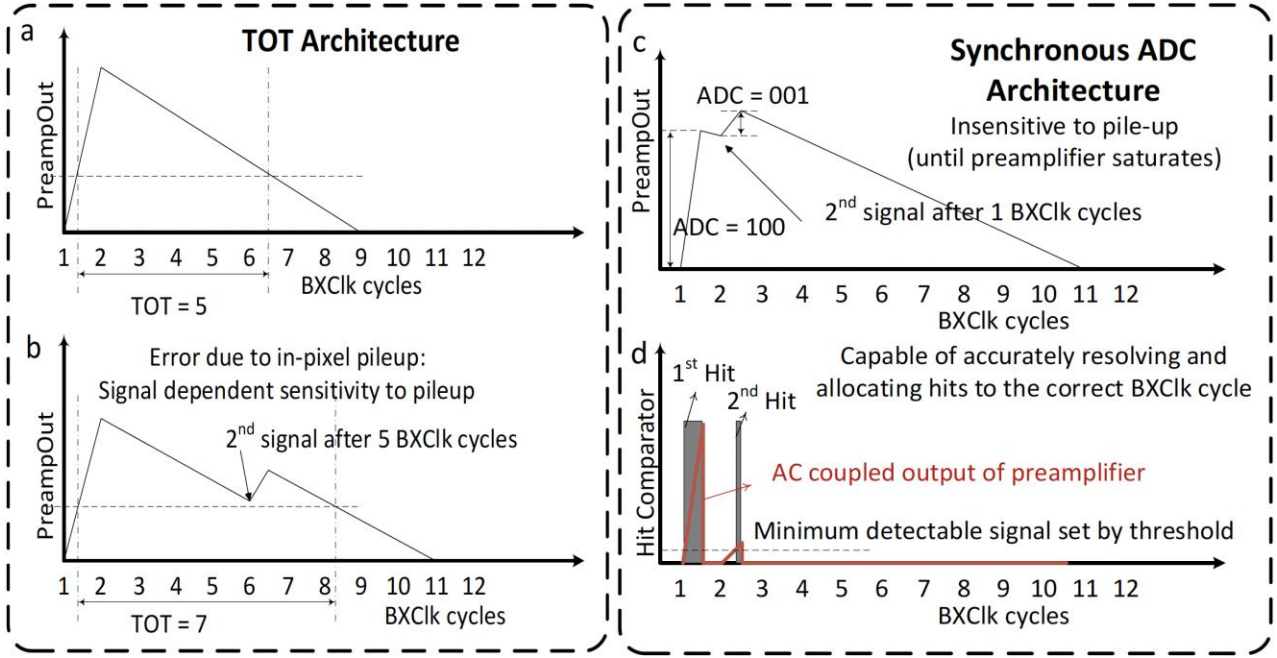
Design space exploration for hardware co-design

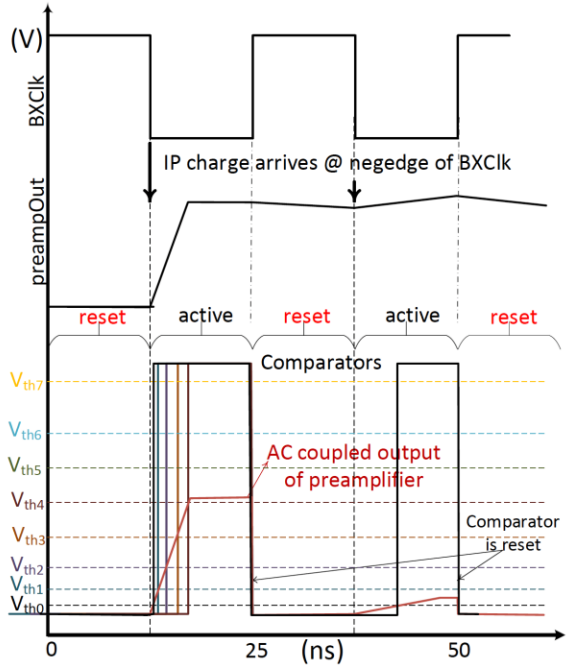


- Compact: Reduce size by reducing the number of parameters
- Energy Efficient: Eliminate clocking and make the design fully combinatorial
- **No. of operations: 75 GOPS (6b INT) per super pixel: 0.5 TOPS/mm^2 : 200 TOPS/chip**

Readout Integrated Circuit: Traditional Implementation

Front-end pixel architecture: Synchronous ADC

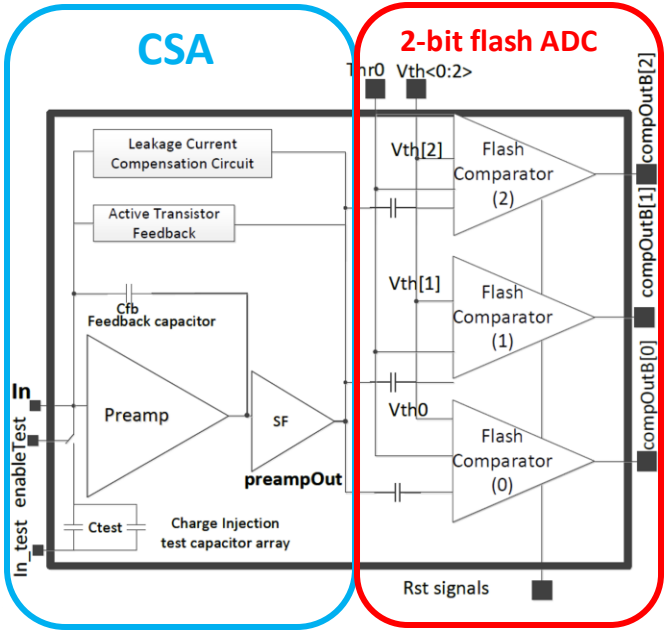


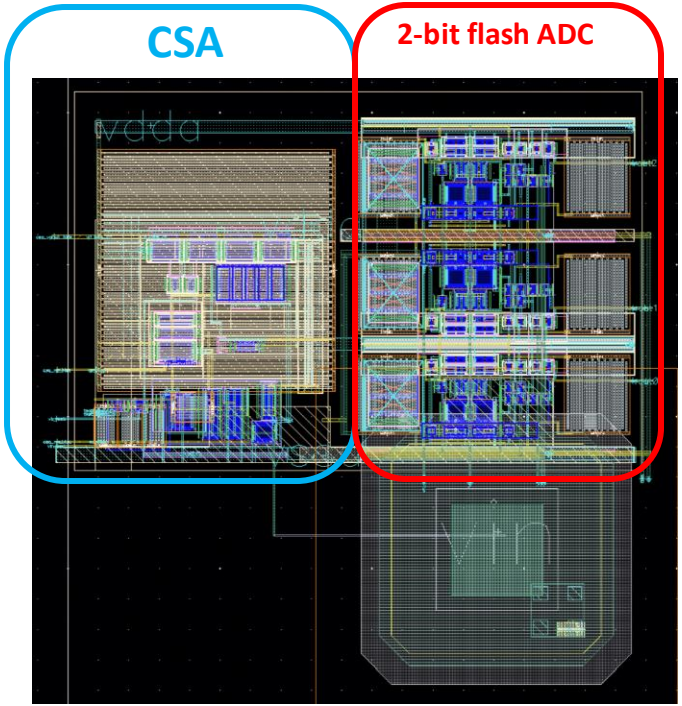


- Signal processing within 1 BXClk cycle (25 ns)
- Insensitive to pileup

Fermilab

Front-end pixel architecture





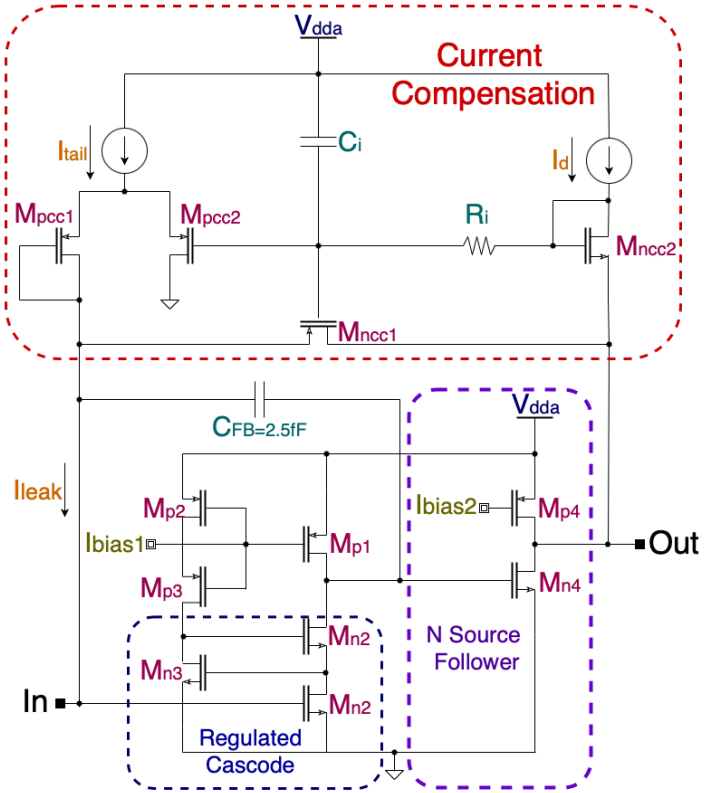
- 25µm × 25µm pixel AC coupled ADC
- 2-bit ADC:

  → version 1: Single ended

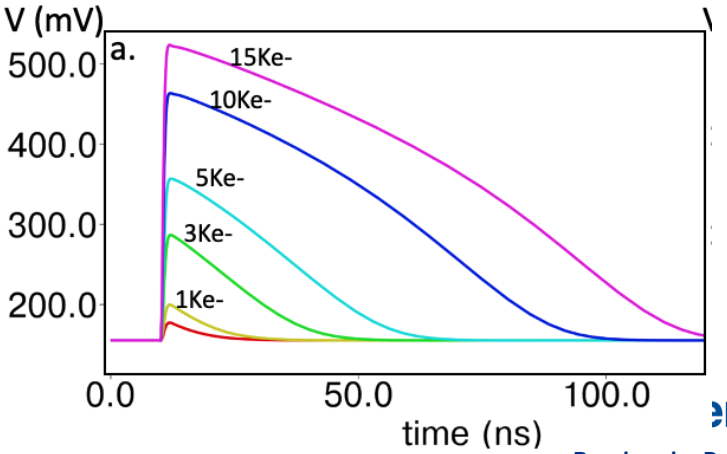
  → version 2: Differential
- Auto zero in every pixel for threshold correction

Charge Sensitive Preamplifier

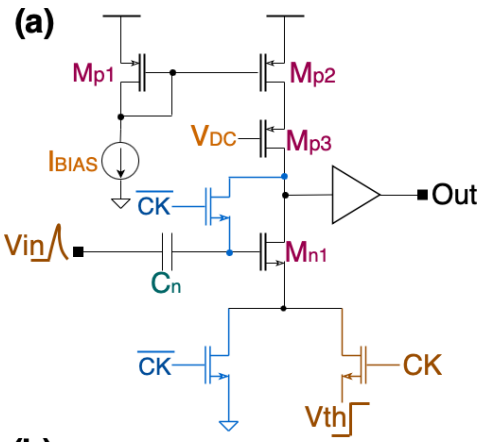
- Regulated cascode for gain stage
- Detector Capacitance 30fF
- Leakage current compensation (up to 50 nA)

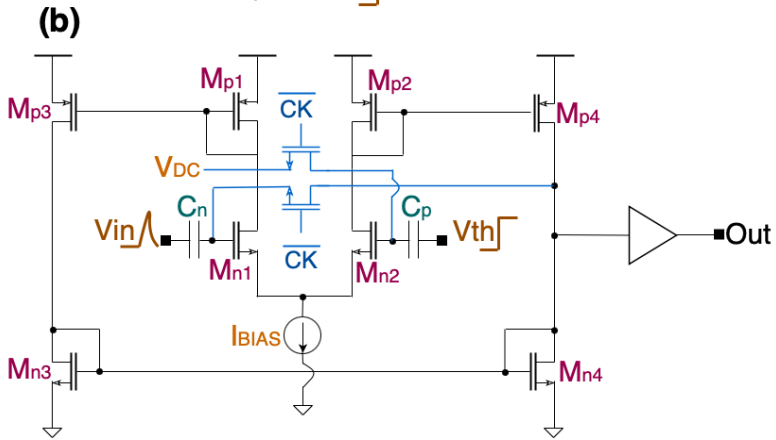


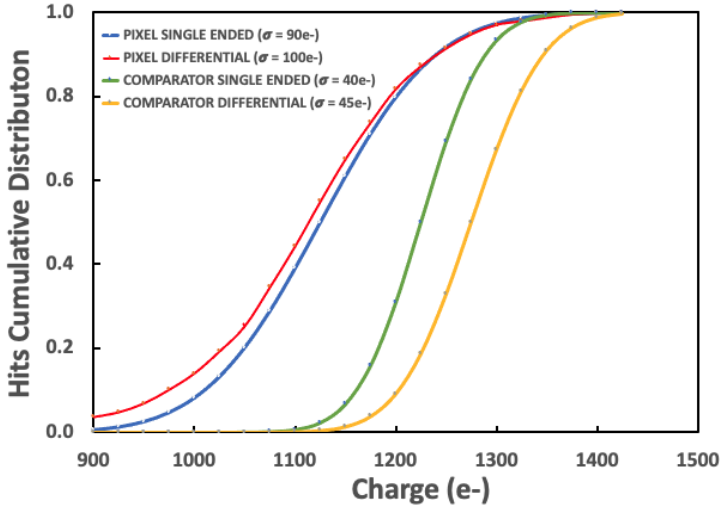
Result	Mode 1	Mode 2	unit
Power	3	4.2	μW
ENC	61	52	e-
Charge sensitivity	40.85	42.3	μV/e-
Phase Margin	65	78	degree
Dynamic Range	13	13	ke-



Comparator



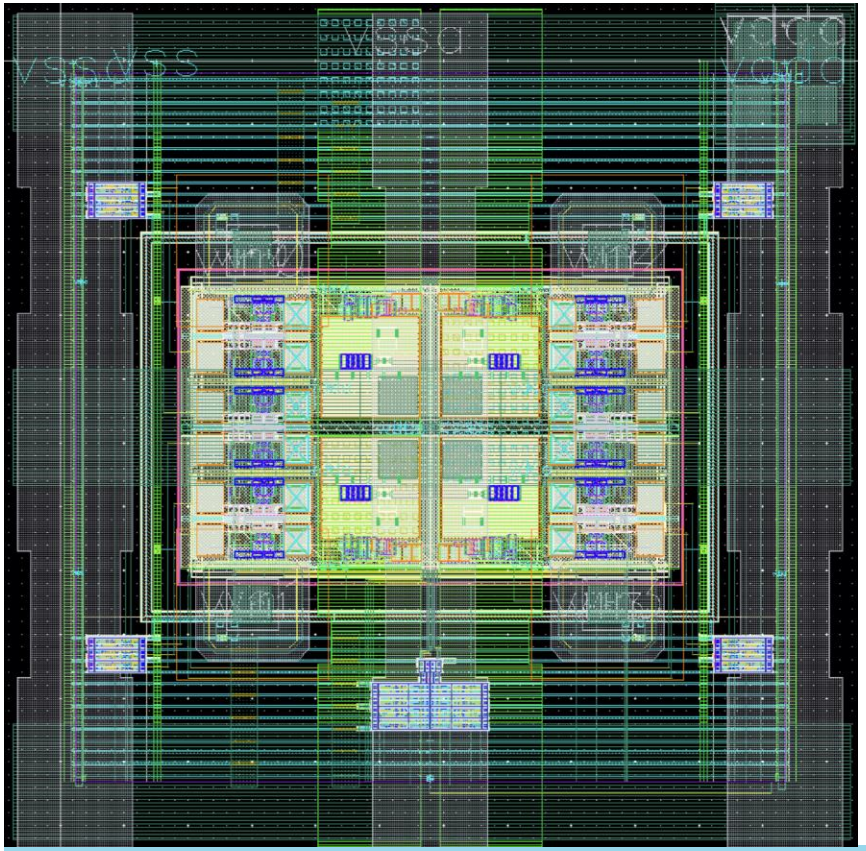


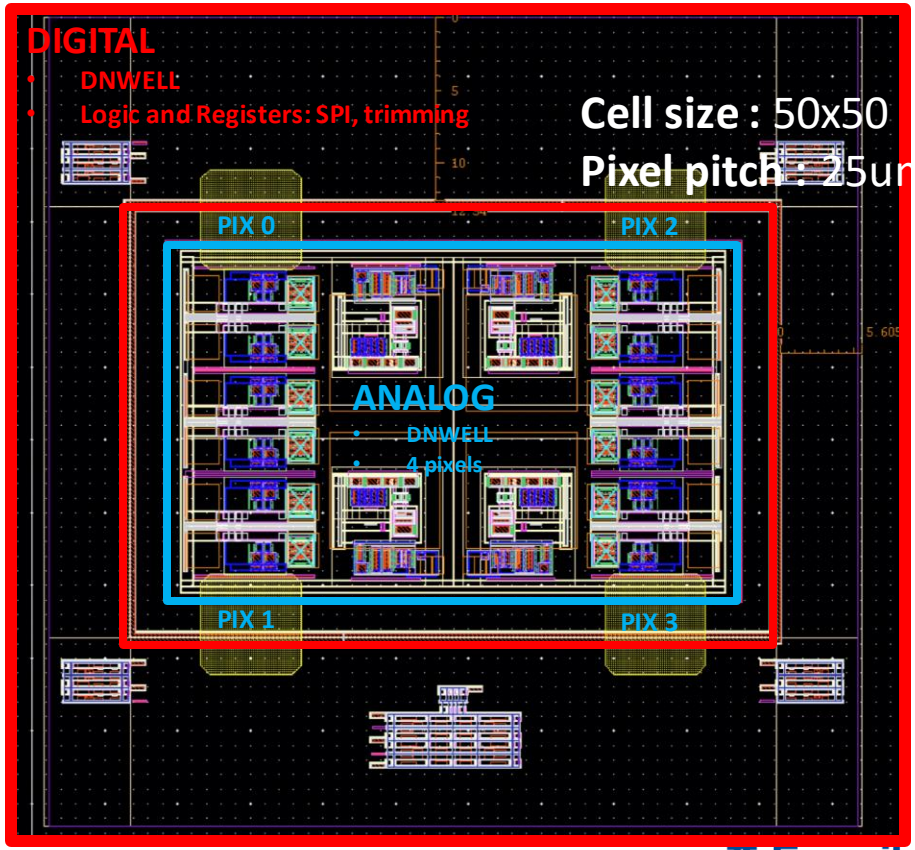


- AC coupled
- Auto-zero for offset cancellation
- Exploring an alternate clockless architecture

Result	v1	v2	unit	comments
Power	3.7	5.2	μW	Per pixel
Total Equivalent Dispersion Charge	90	100	e-	Includes: • ENC • Qth • Baseline fluctuation • kickback
Min Threshold	430	475	e-	4.75 σ
Analog Area	678	845	μm^2	

Grouping 4 pixels to create an analog island

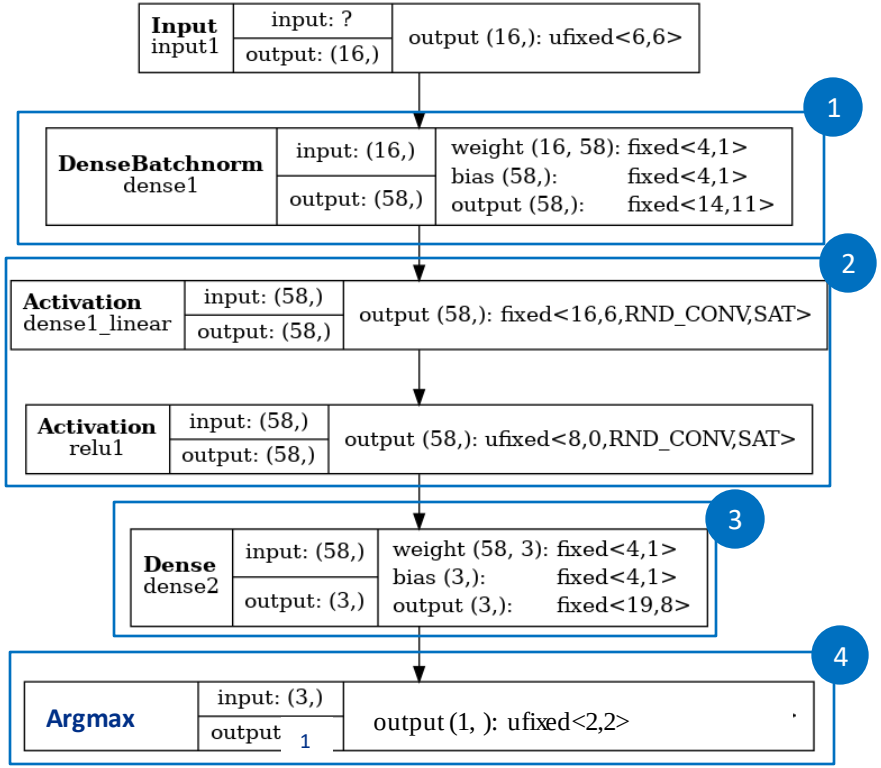


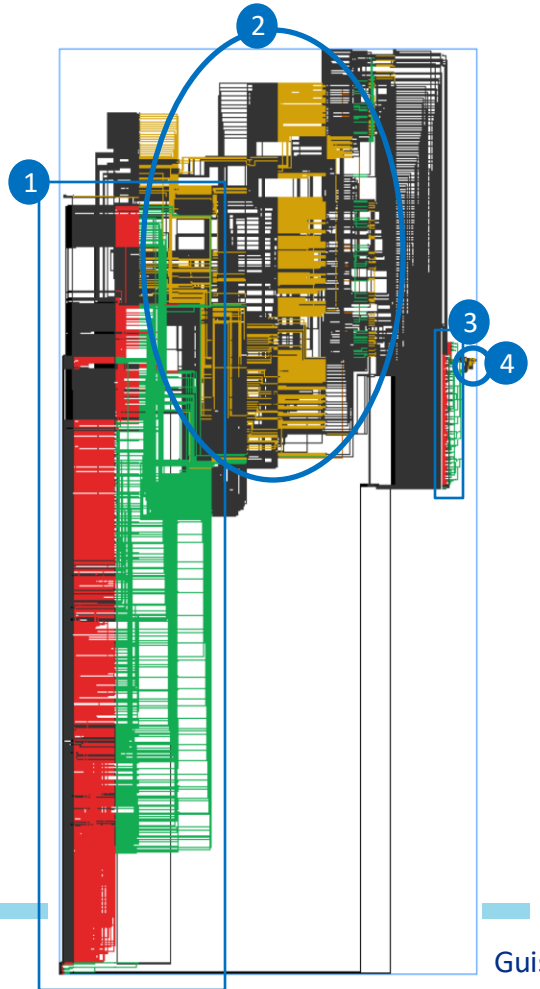


Fermilab

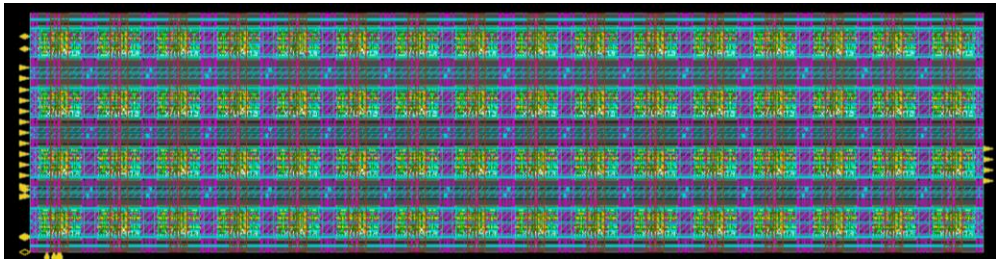
Benjamin Parpillon

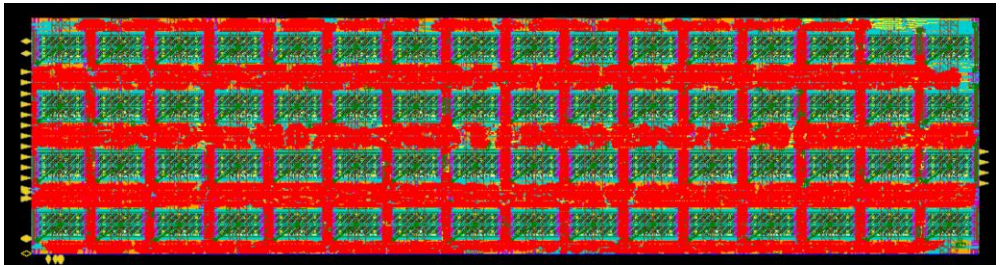
HLS implementation – multi-classifier algorithm

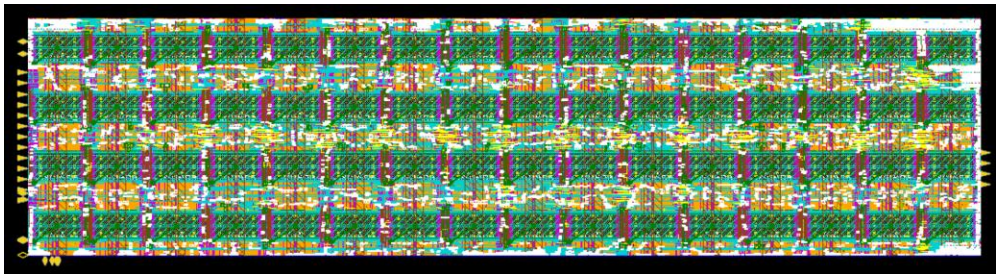




SuperPixel implementation



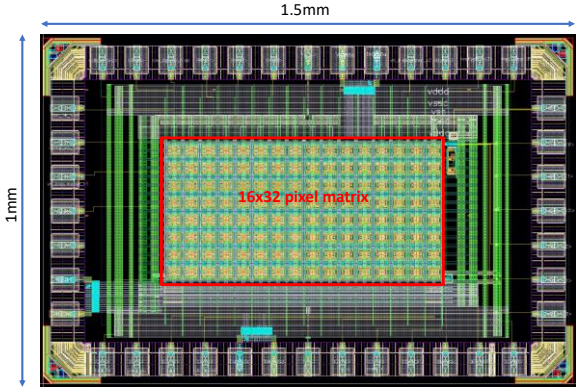


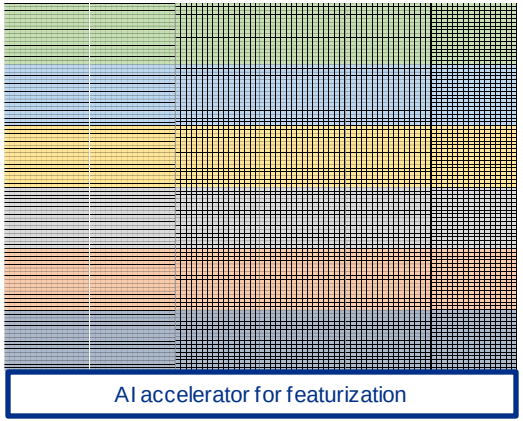


- Floorplan with analog pixels with power and bias grid
- Red highlight corresponds to the data classifier algorithm
- White highlight corresponds to the registers for programming the neural network

- Full triple modular redundancy for SEE mitigation

Current status and next steps





- Analog prototype ROIC with 16 x 32 pixels just been manufactured. In the process of starting tests
- Prototype with embedded algorithm with two super pixels expected to be submitted May 2023
- Next version of chip to include

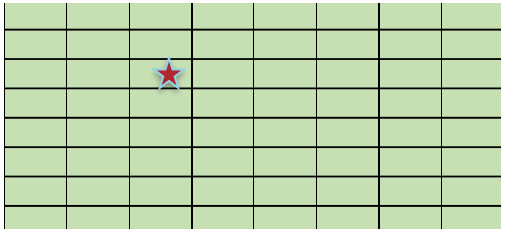
  Multiple super-pixels and peripheral data transfer of selected clusters

  Explore further data reduction through featurization in the periphery

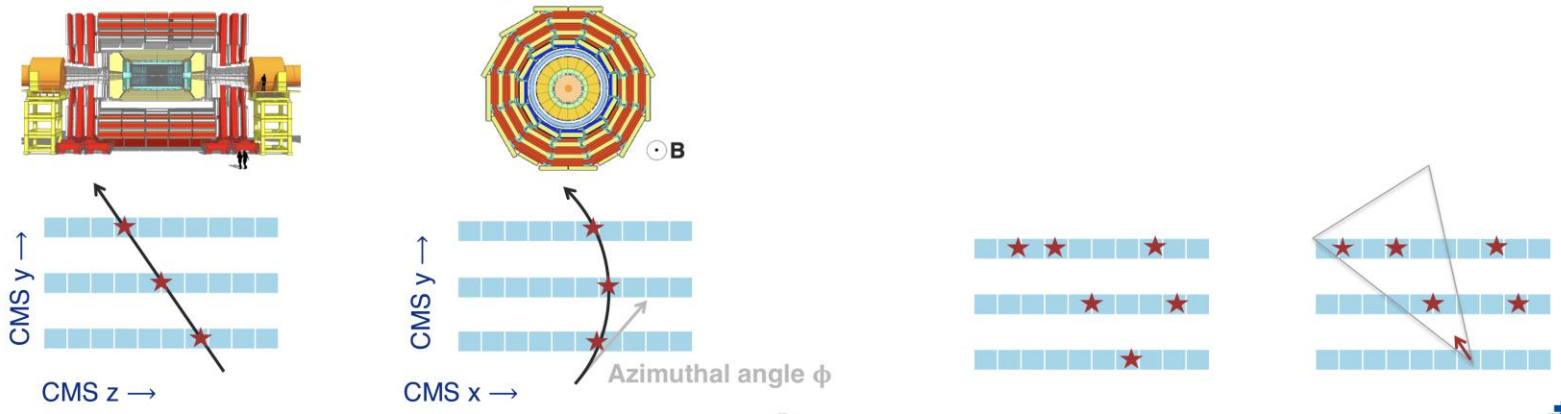
Fermilab

Algorithm development for featurization

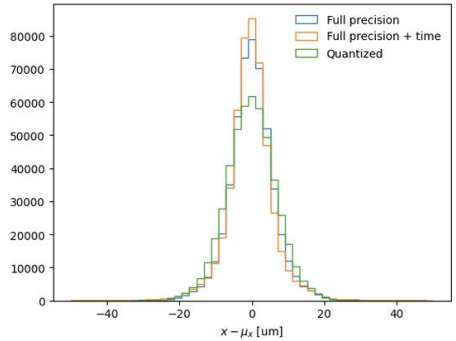
Predicting x,y, angle of incidence to sensor (α, β)

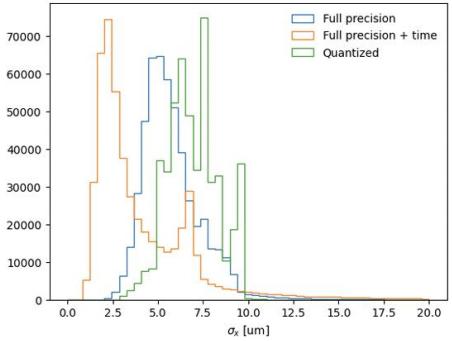


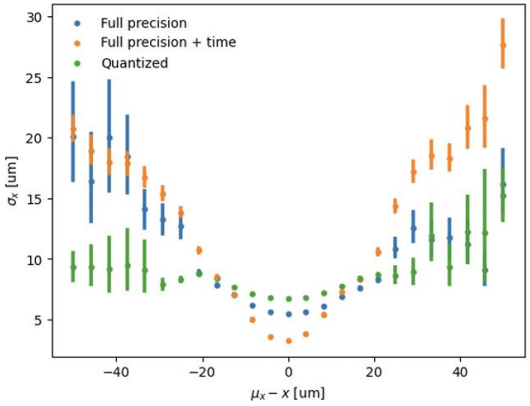
- Where does the particle cross the sensor mid-plane
- Angle of incidence to sensor (perpendicular and in- bending plane of the magnet)
- 3D tracking algorithms are computationally expensive! Grows with number of possible combinations of hits
- With angle and reliable uncertainty, area to search for next hit is defined. Save resources by not looking at bad combinations

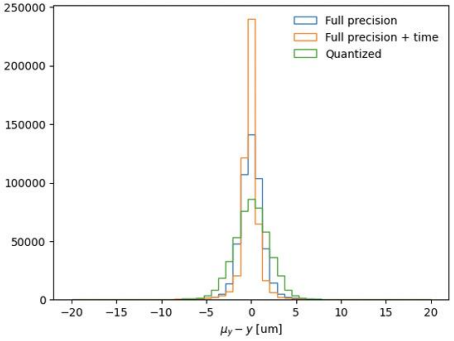


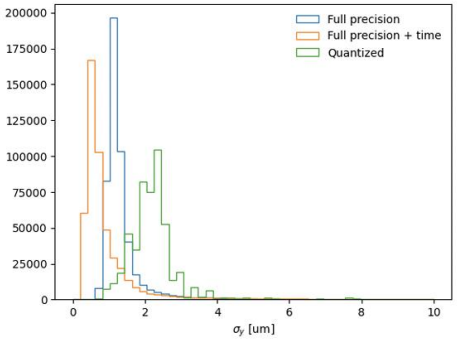
Prediction with uncertainty (x,y)

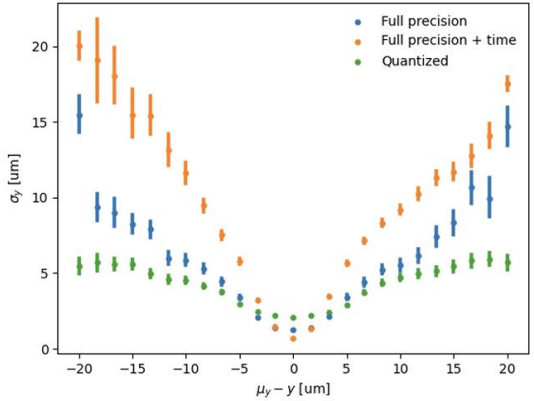










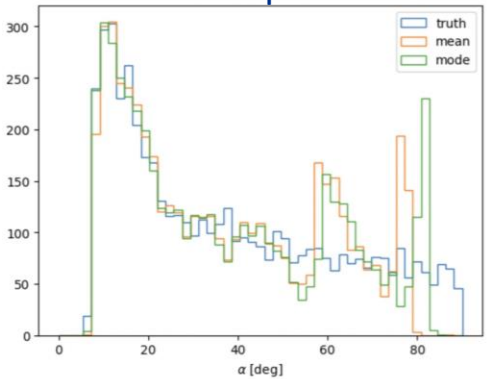


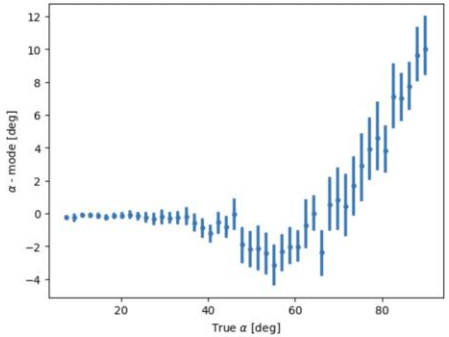
Fermilab

Jennet Dickinson, Gauri Pradhan, Lindsey Gray

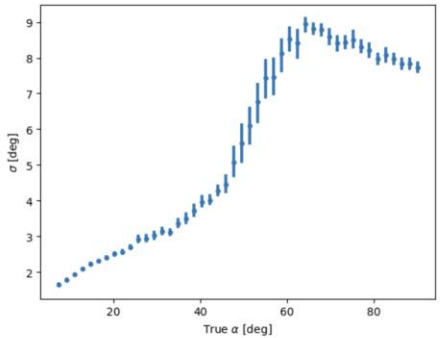
Prediction with uncertainty (α)

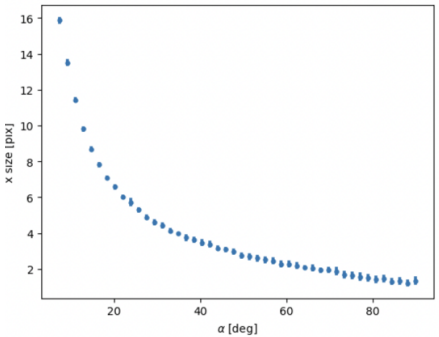
α from x-profile



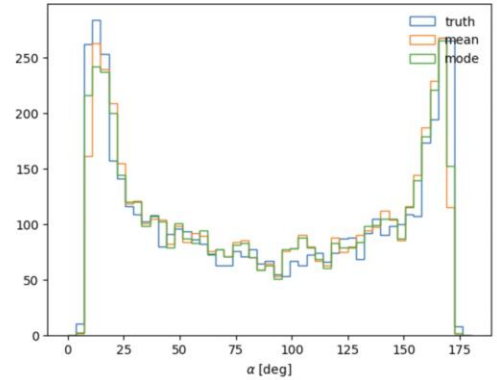


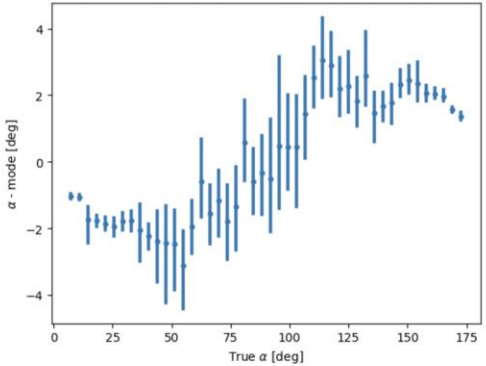
1σ uncertainty < 10° !



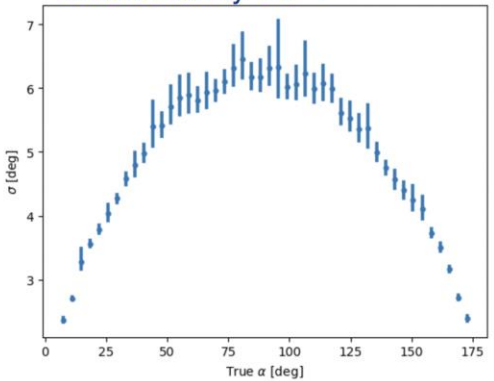


α from x-profile with time





1σ uncertainty < 7° !

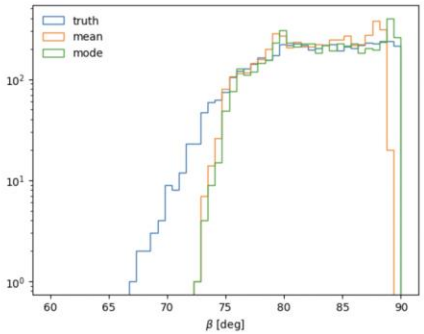


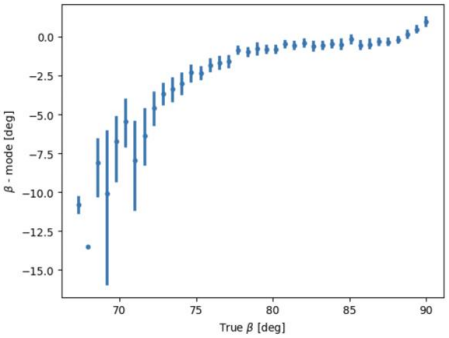
Fermilab

Jennet Dickinson, Gauri Pradhan, Lindsey Gray

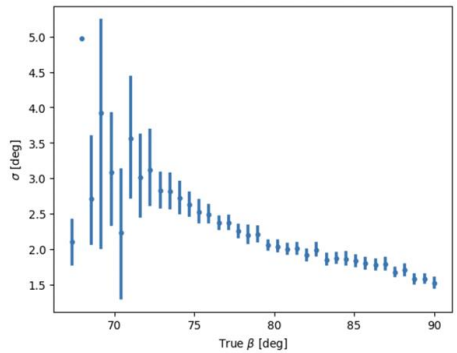
Prediction with uncertainty (β)

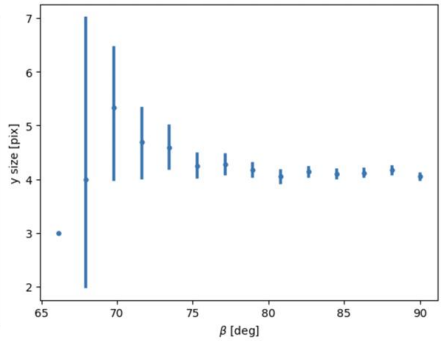
β from y-profile



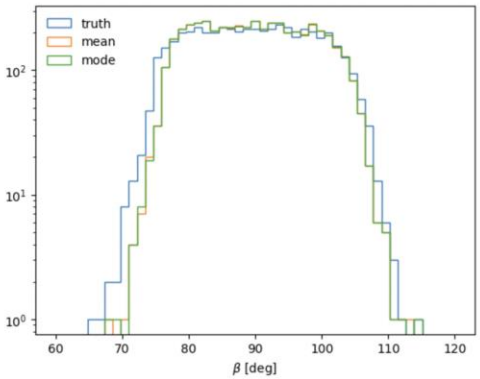


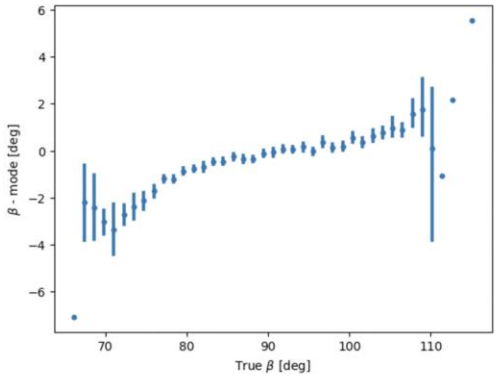
1σ uncertainty < 5° !

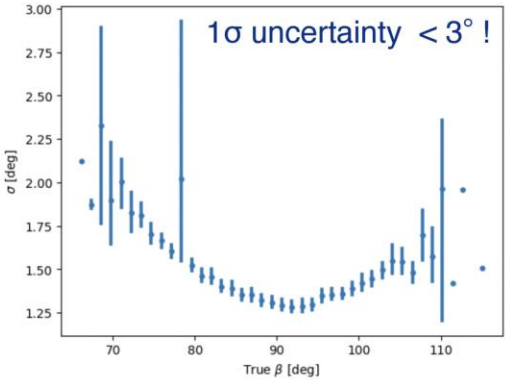




β from y-profile with time







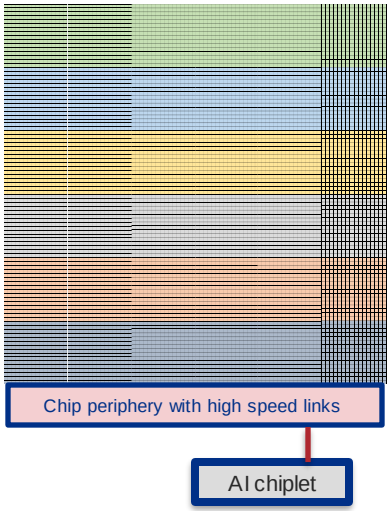
Fermilab

Jennet Dickinson, Gauri Pradhan, Lindsey Gray

Implementation ideas

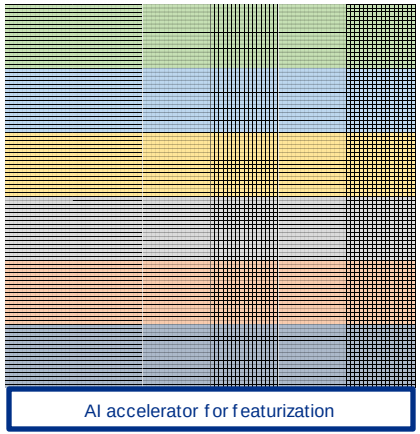
Multi-foundry chiplet

- Serialize and transfer selected clusters off-chip on a high-speed link
- AI Chiplet: 7/5/3nm node
- Highest overhead and power consumption



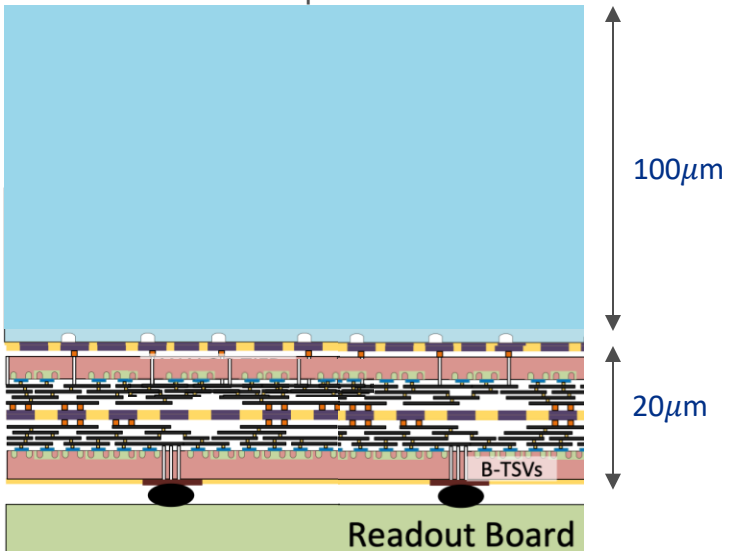
ROIC edge

- Transfer selected clusters to chip periphery
- Data transfer across chip ~ 2cm is about ~1pF
- High speed inference engine



3D-IC

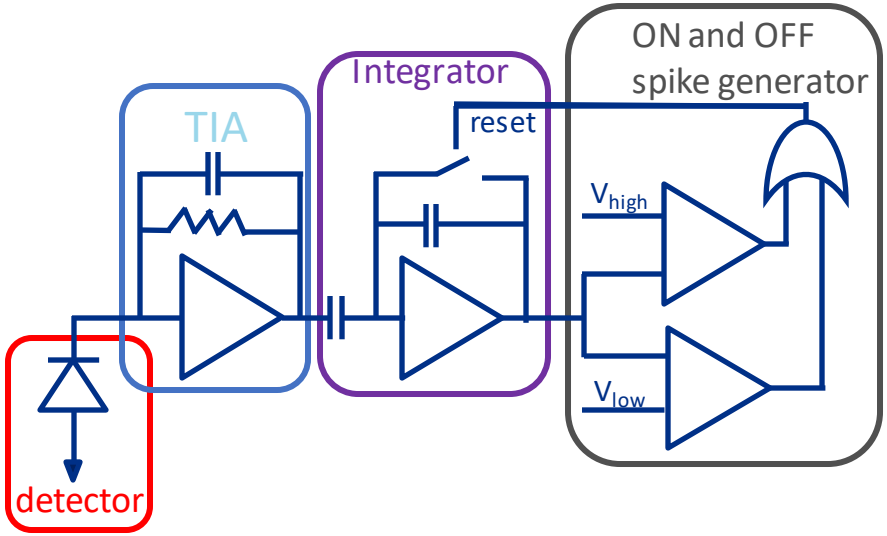
- Highly parallel processing enabled by vertical integration
- Low power data transfer
- Reduced clock speed

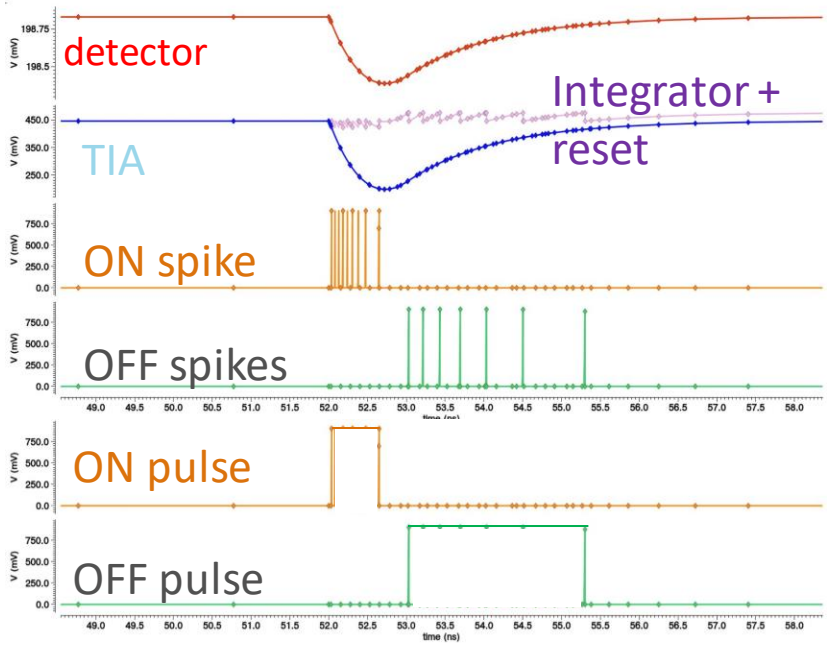


Fermilab

Neuromorphic Implementation: How to exploit the time information?

Neuromorphic frontend concept based on Dynamic Vision Sensors





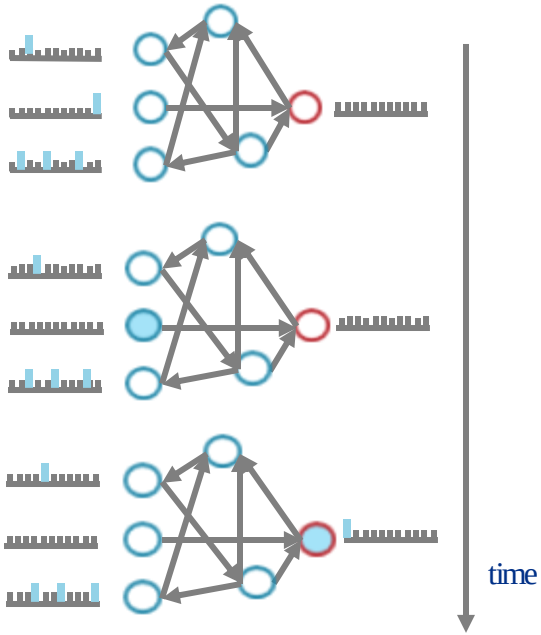
- Pulse shape encoded in spike train
- Ultra-fast signal processing ~ 100ps
- Digital processing – power constraints
- Analog processing – device constraints (speed)

Fermilab

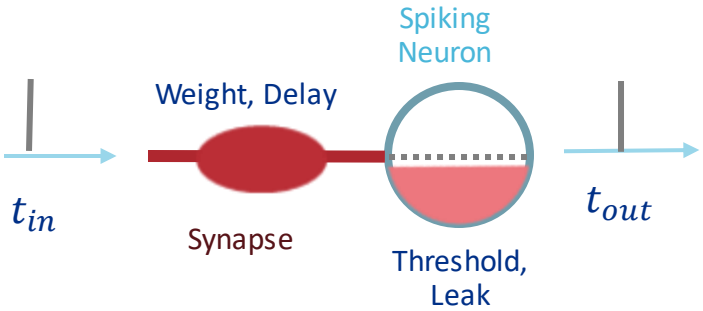
Neuromorphic Algorithm development

Neuromorphic Processing – Basic Model of Operation

Spiking Neural Networks



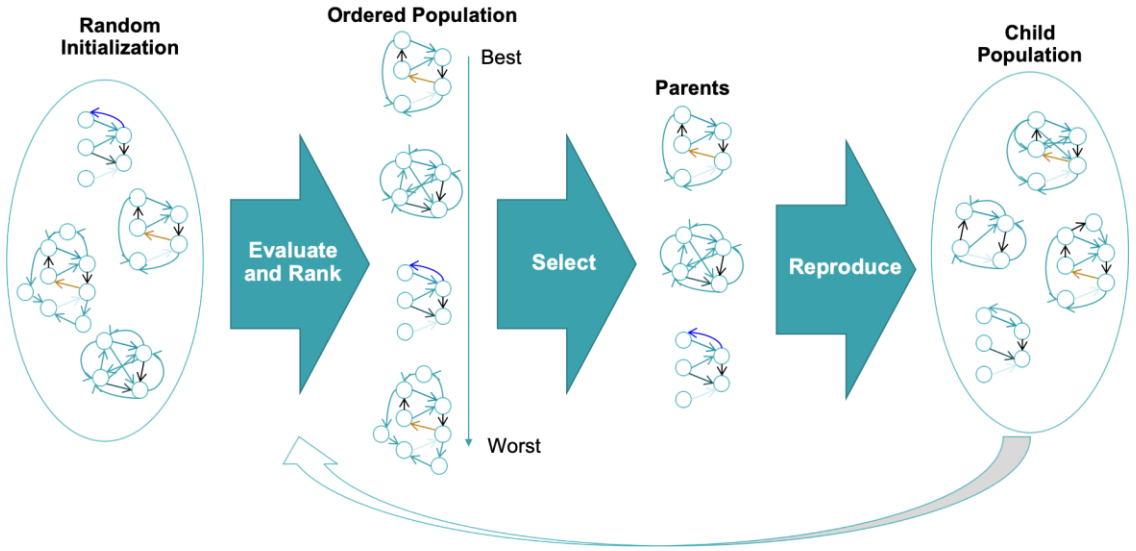
- Computation with sparse events
- Binary valued events



- Inherent temporal dynamics
- Neuron fires with charge exceeding threshold
- Multiple tunable parameters
  - Synaptic weights, delays
  - Neuronal thresholds, leaks
  - Network architecture
- **Promising solutions in designing energy-efficient and area-constrained systems**

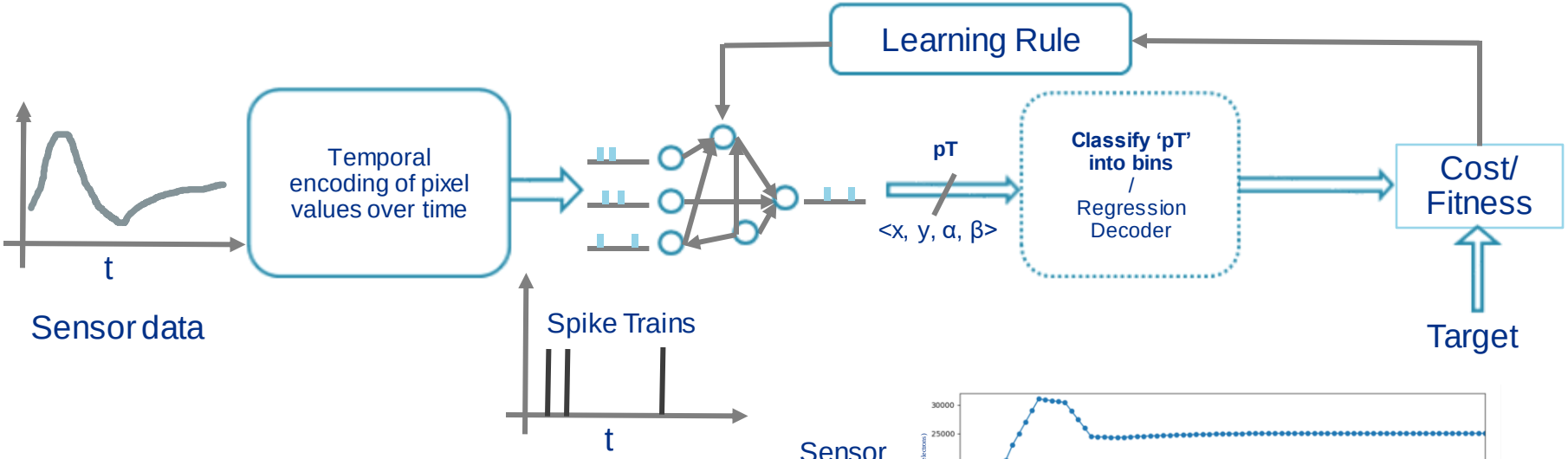
Evolutionary Optimization for Neuromorphic Systems (EONS)

- SNN Parameters:
  - Synaptic Weight and Delays
  - Neuronal Thresholds and leak
  - Network topology
- Often produces networks with **very sparse connectivity**
- **Useful for co-design**
- Can be slow to train



Schuman, Catherine D., et al. "An evolutionary optimization framework for neural networks and neuromorphic architectures." *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016.

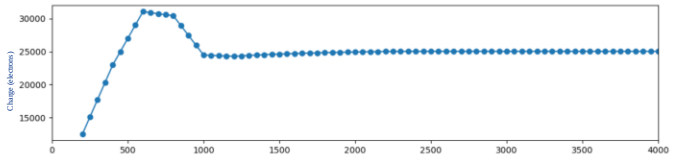
Translating dataset to spikes



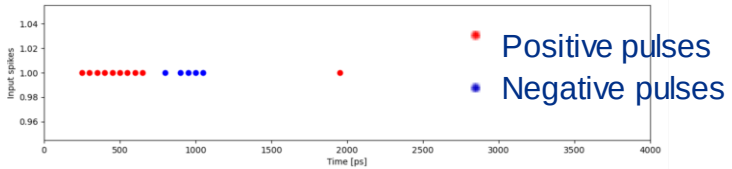
Simulated Spike Data:

- Generate spike every 400e change in the signal
- Signal measure every sampling time

Sensor Waveform

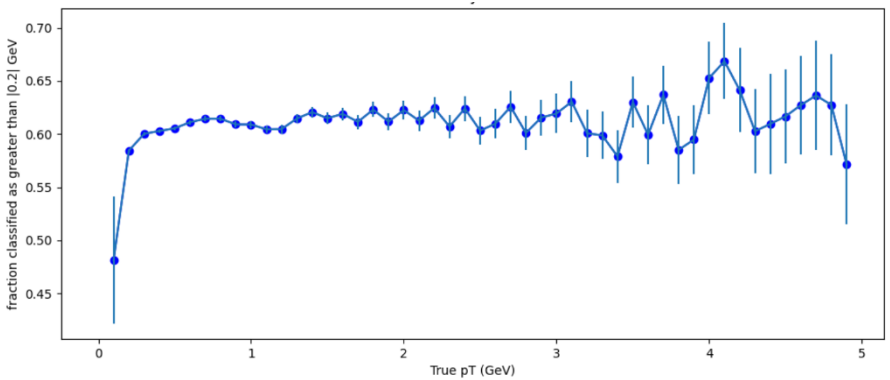


Simulated spikes

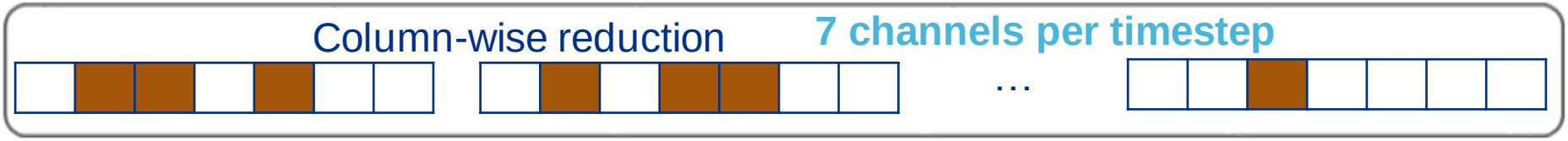


Initial training results

- **Dataset:**
  - 95% high pT samples, 5% low pT samples
- **Training**:
  - With 70k samples
  - Balanced dataset
  - SNN size: 688 neurons, 746 synapses
- **Test results**:
  - With ~650k samples (unbalanced)
  - F1 score: 0.7
  - Accuracy: 0.60



Y-profile equivalent: “OR”



Unbalanced dataset

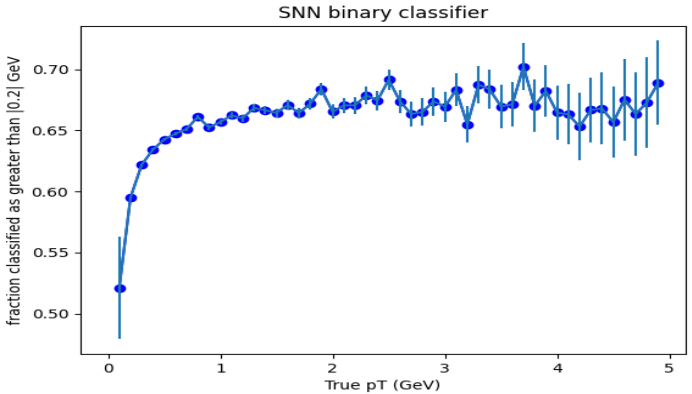
- Training samples: 2.6 million; Test samples: ~1.3 million
- Test accuracy: 0.62; Test f1 score: 0.73

Training set

True labels/ Predicted	Class 0	Class 1
True low pT (0)	**70,229**	68,480
True High pT(1)	929,830	**1,613,667**

Test set

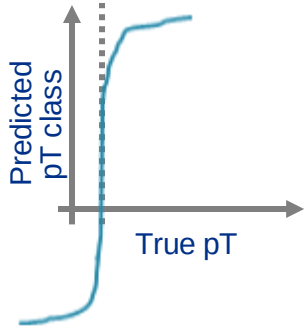
True labels/ Predicted	Class 0	Class 1
True low pT (0)	**35,997**	39,051
True High pT(1)	463,962	**814,966**



Fermilab

Fitness function with penalty term

Introduce penalty term that is high for misclassification of very high pT samples, and small for low pT (class 1) samples

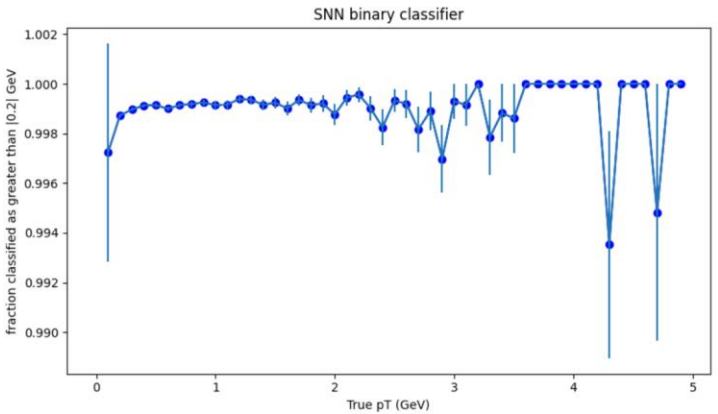


- Prediction error:

$$E = |pred - target|$$

- EONS fitness:

$$f = \sum_i -E_i \times |\tanh(k \times (pT_i - pT_t))|$$



	Predicted 0	Predicted 1
True 0	182	74,866
True 1	966	1,277,962

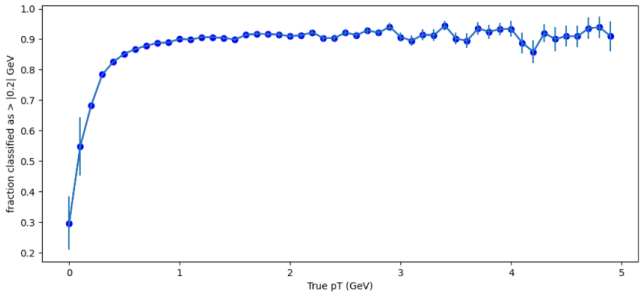
- Trained for 1000 epochs
- Network: 548 nodes, 202 edges
- Total Test samples: 1,353,976
- Testing f1 score: 0.92
- Testing Accuracy: 0.94

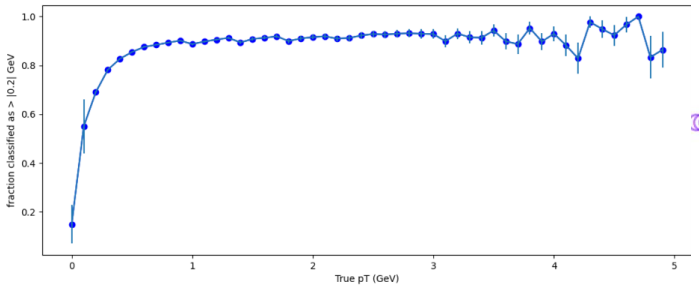
Classification is heavily biased towards high pT samples

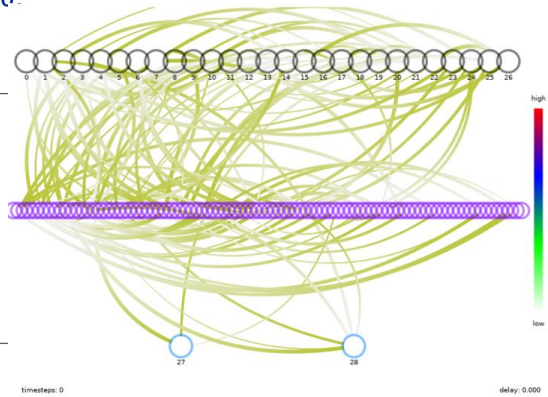
Y-profile with spatial encoding

- SNN trained on samples with:
  - pT threshold at 0.5
  - y-local $\in [-1.0, 1.0]$
- # inputs: 27 (26 + bias), #o/ps: 2
- **Training Samples**: 345,383
- Data Classification (>2GeV): 91.43%
- Data Rejection: 19.05%

- Trained network:
  - # Neurons: 46, # synapses: 89
- Performance evaluated with pT threshold at 2GeV
- **Test Samples**: 173510
- Data Classification (>2GeV): 91.91%
- Data Rejection: 19.02%



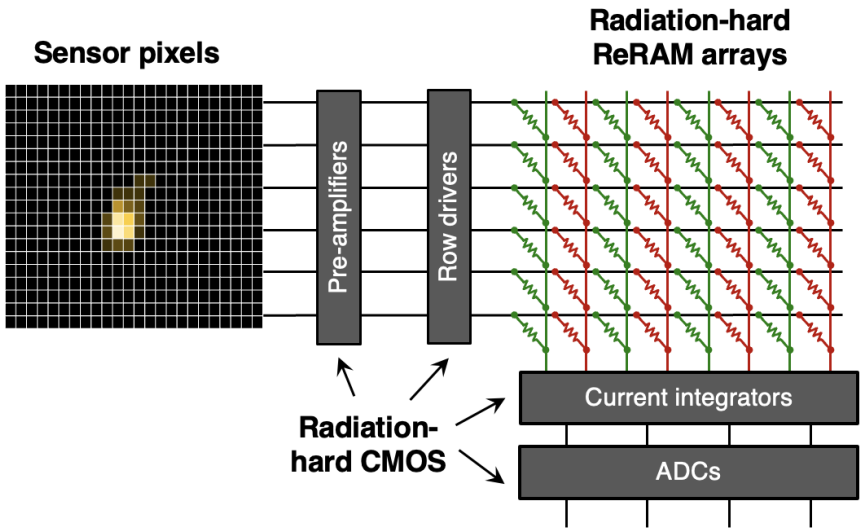


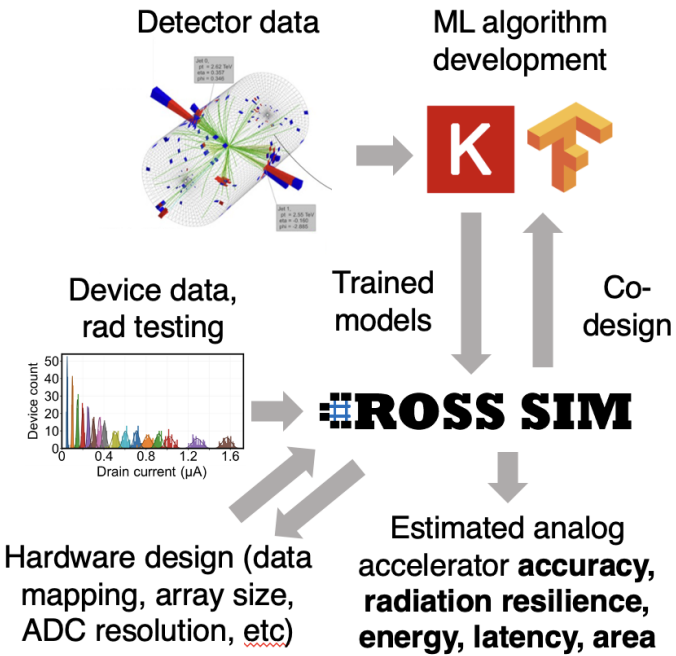


Fermilab

Neuromorphic Materials for edge computation

Analog processing potentially enables lower energy, faster, and/or more complex on-pixel processing – or enables larger, more sophisticated neural networks





Patrick Xiao, Alec Talin, Sapan Agarwal

Thank you

- Fermilab

  Nhan Tran, Doug Berry, Lindsey Gray, Jennet Dickinson, Gauri Pradhan, Benjamin Parpillon, Guiseppe Di Guglielmo

- Northwestern University

  Manuel Blanco Valentin, Seda Memik

- UIC

  Corrinne Mills, Jieun Yoo, Amit Trivedi

- Johns Hopkins

  Morris Swartz, Petar Maksimovic

- ORNL

  Shruti Kulkarni, Aaron Young

- Sandia National Lab

  Patrick Xiao, Alec Talin, Sapan Agarwal

Case Study II - 65 nm implementation

– to understand challenges and strategies to overcome them

– non-reconfigurable, non-programmable NN for data compression

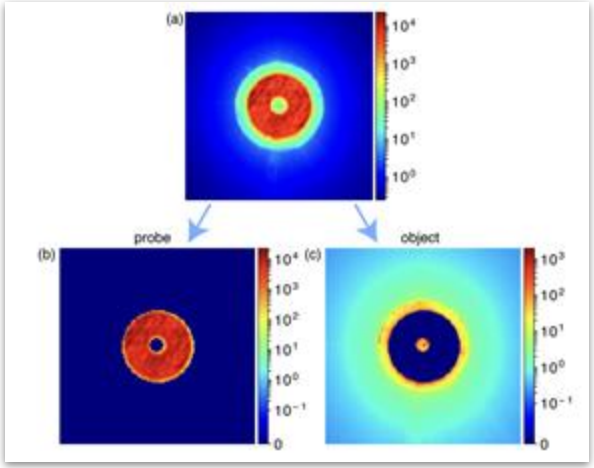
In-pixel AI/ML to enable Mfps camera operation

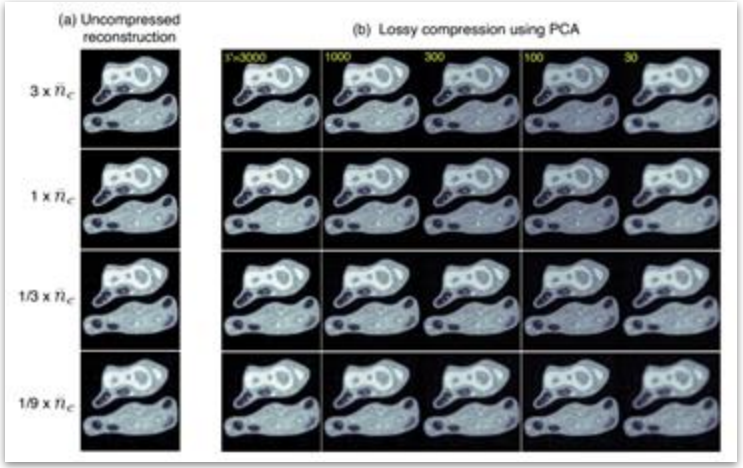
- **Charge integration → Digital Conversion → Data Compression → Off-Chip Data transfer**
- **For deadtime-less operation: Charge integration + Digitization should take the same time as Off-Chip Data Transfer**
  - While ADC is converting the signal for this cycle, readout IC is transferring data from previous cycle
- **Single chip: (400 x 400 pixels x 10b x 1Mfps) 1.6Tbps data → 32Gbps enabled by 50x lossy data compression**
- **1x 32 Gbps photonic link or 3 x 10.24 Gbps link per chip (~5cm²)**

	Charge Integration	Analog to Digital Conversion	Data Compression	Off-chip Data Transfer
Subchip = 1024 pixels for 100 Kfps operation	~1µs	~9µs	Full-readout	1024*10b @ 1GSPS = 10µs
Subchip = 1024 pixels for **100kfps operation**	~1µs	~9µs	Data compression **PCA**: 50x 1 cycle: 160n **AE**: 70x 30 cycles: 4.8µs	 220b@ 1GSPS = 220n 150b @ 1GSPS = 150n

Fermilab

Physics Background: Ptychography





Huang, P., Du, M., Hammer, M., Miceli, A., & Jacobsen, C. (2021). Fast digital lossy compression for X-ray ptychographic data. *Journal of synchrotron radiation*, 28(Pt 1), 292–300.

Ptychography: Imaging using coherent interference patterns in diffraction from object

- Large quantities of data with inherent redundancy

Solution: Image Compression for X-Ray Imaging

Detector goal: 1 Mframe/s

This prototype: Frontend at 100 Kframes/s and 50 - 70x data compression in the pixel

Fermilab

Why data compression instead of zero-suppression?

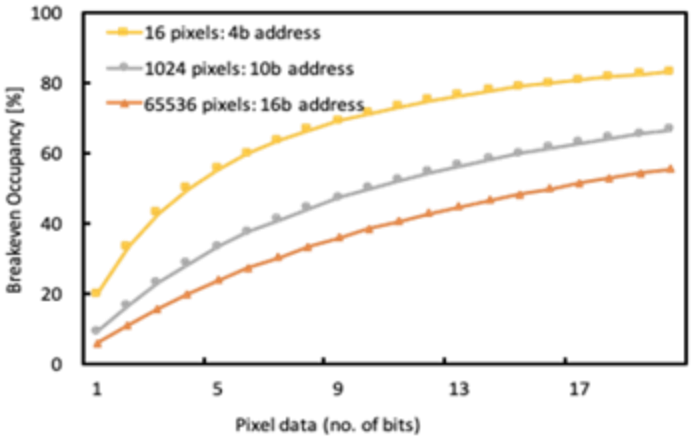
Data rate:

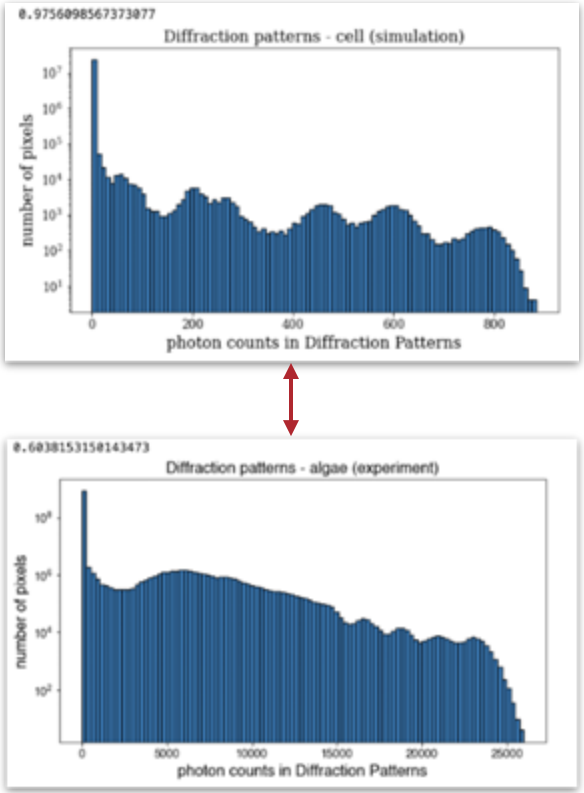
400 × 400 (pixels per chip) × 10b (ADC) × 1 MHz

~ 1.6 Tbps

Data Sparsification:

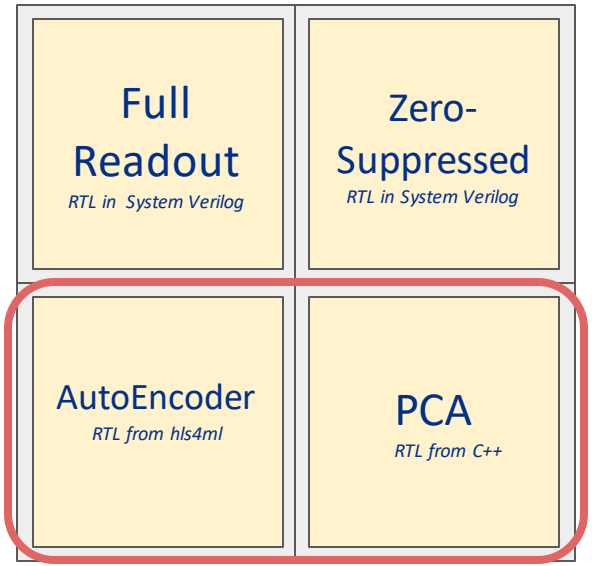
- Zero suppression (overhead address ~ 18-20b per 12b data)
- Simulated data (~97% zeros) vs. Noisy data (~60% zeros)



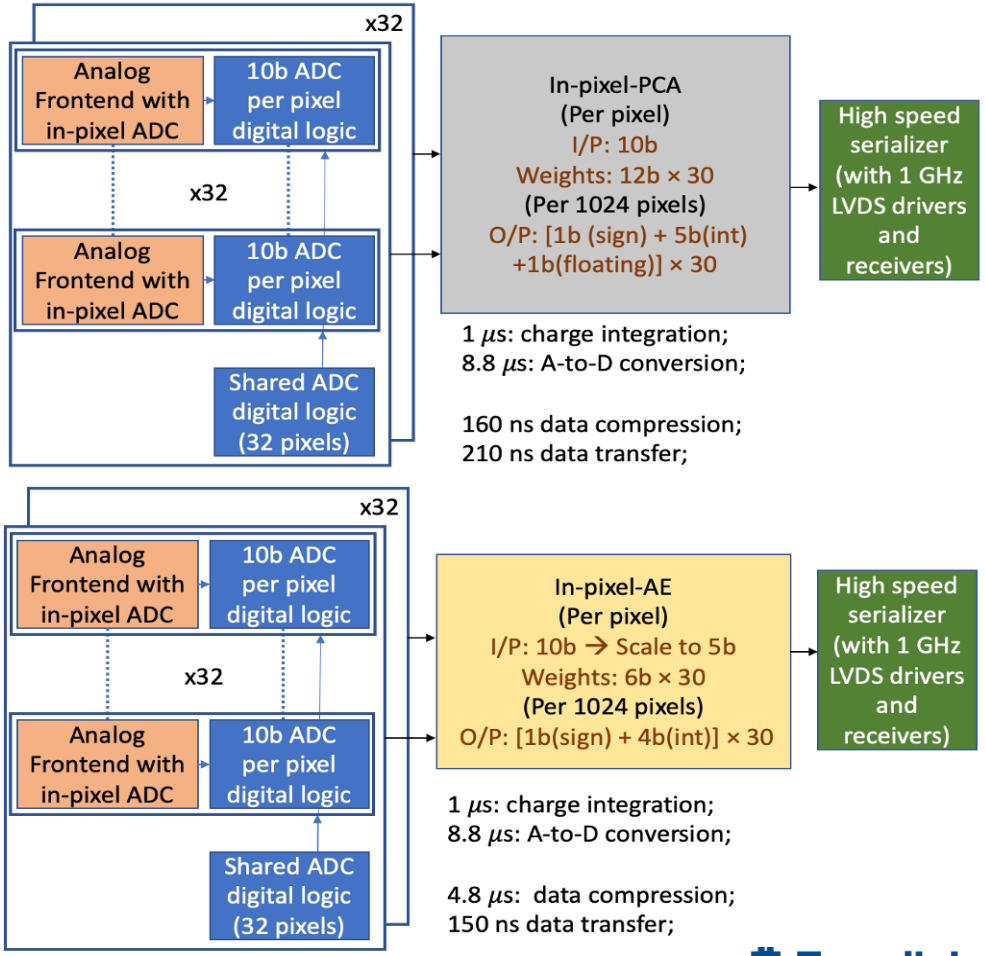


Fermilab

XROCKET: Proof of concept chip

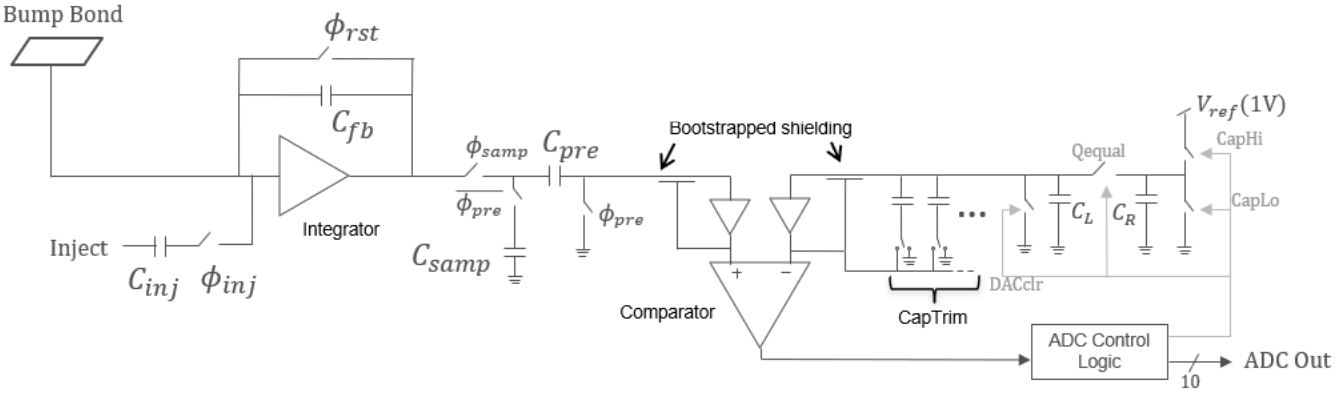


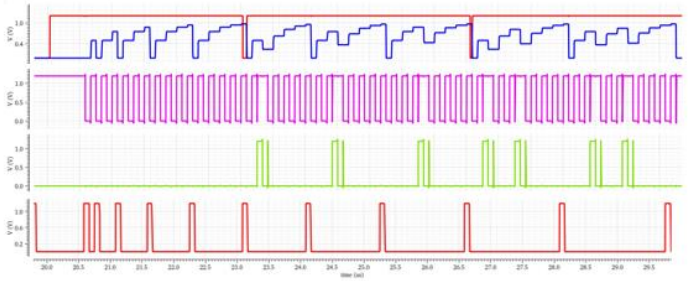
- **Area per pixel: 50 µm x 50 µm without data compression**
- **Area per pixel: 55 µm x 55 µm with data compression - Explored two approaches**

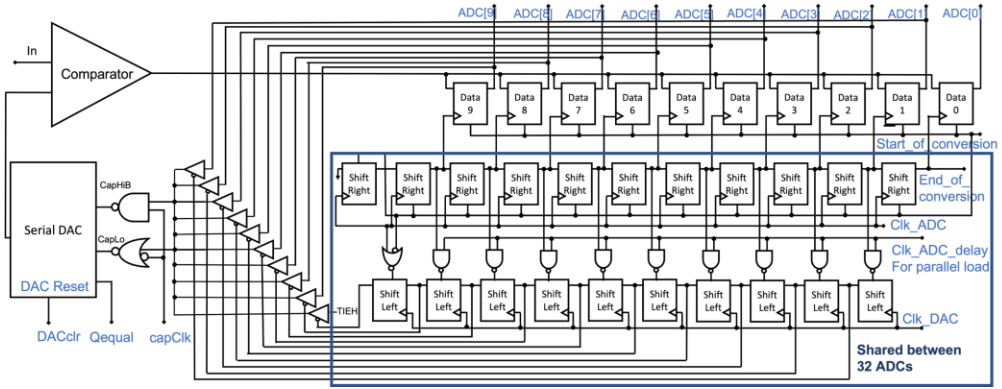


XROCKET Pixel frontend

- Analog frontend consists of preamplifier + CDS + serial SAR
- Digital pixel backend consists of serial SAR logic
- Synchronous binary tree priority encoder





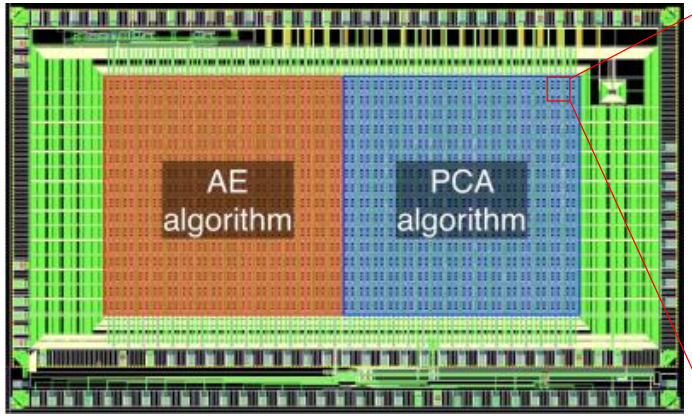


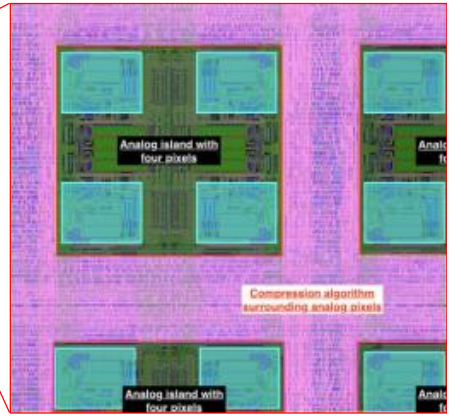
Fermilab

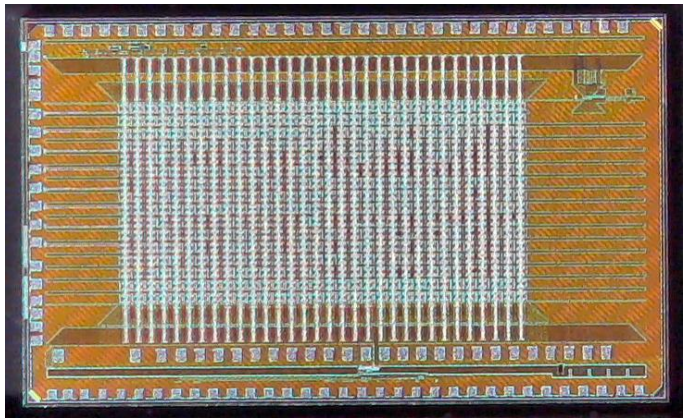
XROCKET

Data rates and design choices

- Have to compress data on chip
- Spend ~ 1pJ/b for data transfer off-chip (takes lower power to process the data on-chip)
- Should this be done in-pixel or at the periphery?





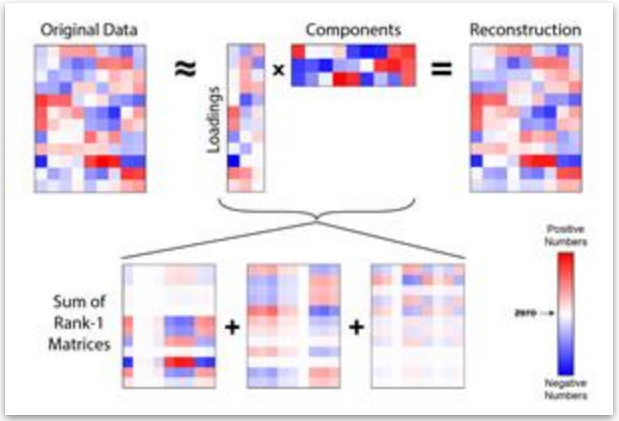


	1 pixel	32 x 32 pixels	400 x 400 pixels	Camera with 4 x 4 chips
Size	55 × 55 μm^2	1.76 × 1.76 mm^2	2.2 × 2.5 cm^2	
Power	50μW	51.2mW	8W	128W
Raw data	10b/10μs = 1Mbps	1Gbps	1.6Tbps	25.6Tbps
With 70x compression		14Mbps	23 Gbps	360Gbps

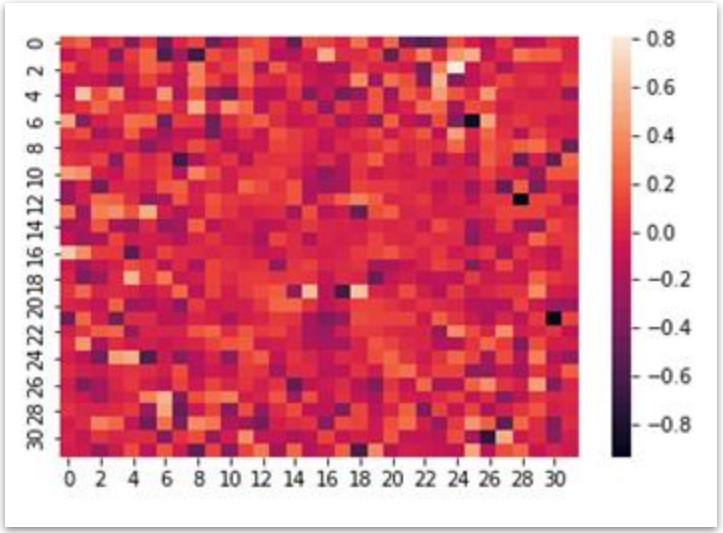
Fermilab

Algorithm Co-Design

Algorithm 1: Principal Component Analysis (PCA)



Alex Williams, *It's Neuronal Blog*



Calculated Weight Distribution, PCA

$P_n = D_j \times R_{j \times n}^{-1}$

j=1024
n=30

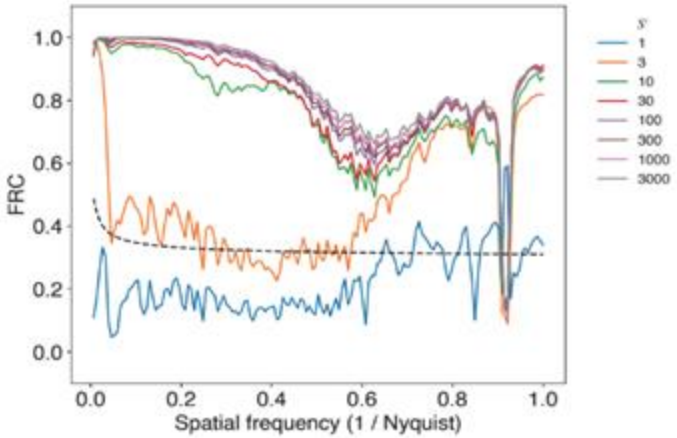
Goal: Reasonable Reconstruction of original diffraction image

Parameter Choice: 30 outputs ← tradeoff compression vs. quality loss

- Analogous to principal components
- Weights are thus mathematically calculated: for inverse of eigenimages of diffraction input

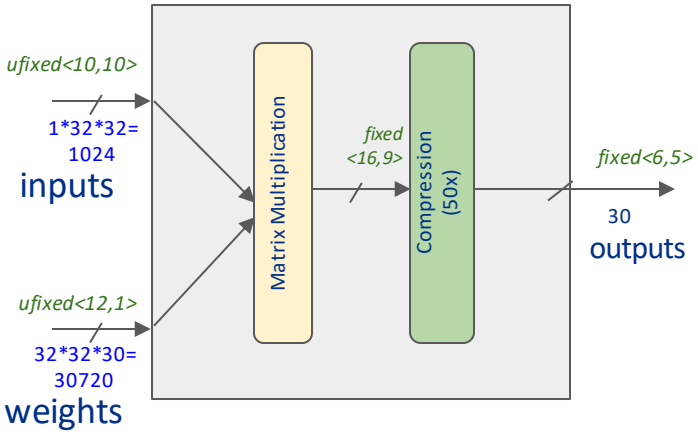
Fermilab

PCA Algorithm: C++ Realization



Huang, P., Deng, J., Noonan, D., Tran, N., Fahim, F., & Jacobsen, C. (2022). Supplementary Material: Matrix Factorization approaches to lost compression of ptychographic data. *Journal of synchrotron radiation*, Preprint, 43-50

- Computing the Fourier ring correlations (FRC) between the reconstructed image and the initial image used to simulate the diffraction patterns.
- FRC quantifies the similarity of two images across varying spatial resolutions.

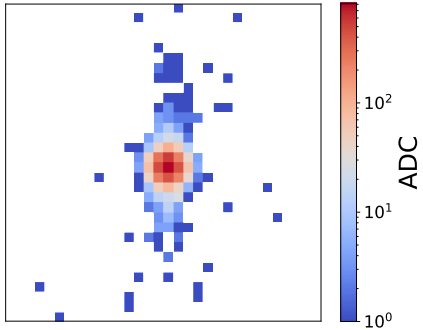


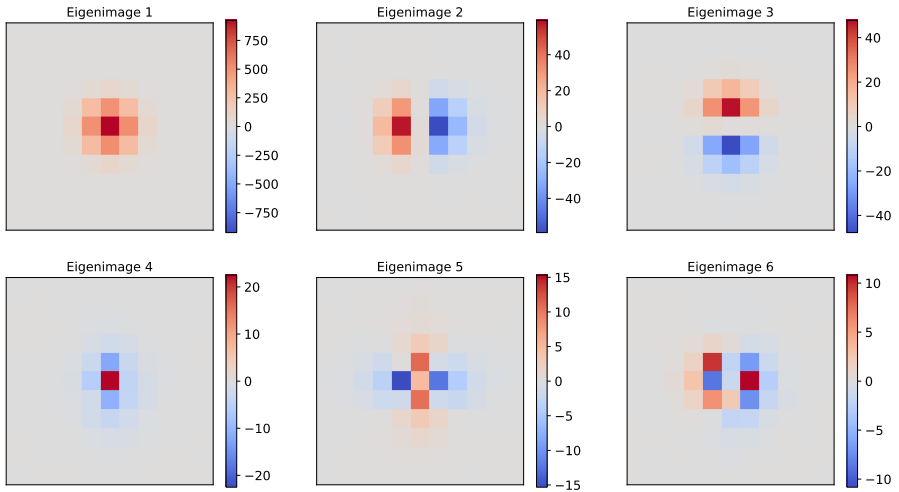
- **Output Size: 30 → based on factor of compression, here 50x**
  - Going from 1024 × 10-bit to 30 × 7-bit
- **PCA Loop Limits**
  - Number of rows : 32
  - Number of columns : 32
  - } Eigenimage dimensions
  - Output Size : 30 ↔ Number of Eigenvalues
- **Intermediate Data**
  - 16-bit: PCA sensitive to loss of precision during accumulation
    - Inverse Eigenimage Matrix

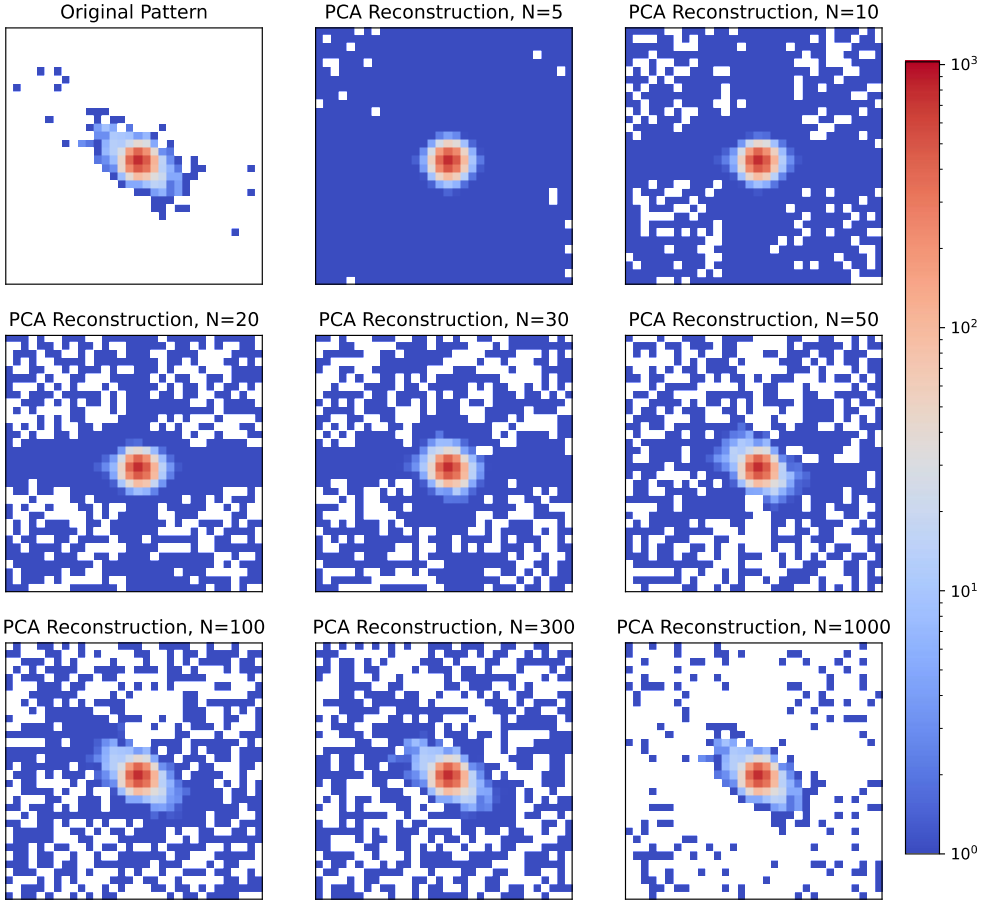
	32 x 32 pixels	400 x 400 pixels	Camera with 4 x 4 chips
TOPS-INT12 (data rate ~ 100 Kfps)	~0.06	~0.9	~14
TOPS-INT12 (data rate ~ 1 Mfps)	~0.6	~9	~140

Google edge TPU: 4TOPS (Int 8)

Fermilab

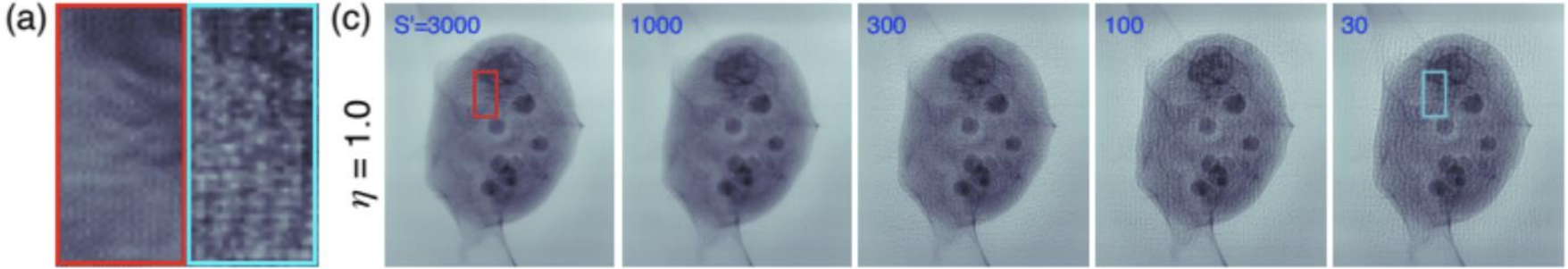

10^2
10^1
10^0
ADC


Eigenimage 1
750
500
250
0
−250
−500
−750
Eigenimage 2
40
20
0
−20
−40
Eigenimage 3
40
20
0
−20
−40
Eigenimage 4
20
10
0
−10
−20
Eigenimage 5
15
10
5
0
−5
−10
−15
Eigenimage 6
10
5
0
−5
−10


Original Pattern
PCA Reconstruction, N=5
PCA Reconstruction, N=10
PCA Reconstruction, N=20
PCA Reconstruction, N=30
PCA Reconstruction, N=50
PCA Reconstruction, N=100
PCA Reconstruction, N=300
PCA Reconstruction, N=1000
10^3
10^2
10^1
10^0

Fermilab

Can you use eigen images from one training set for another data set

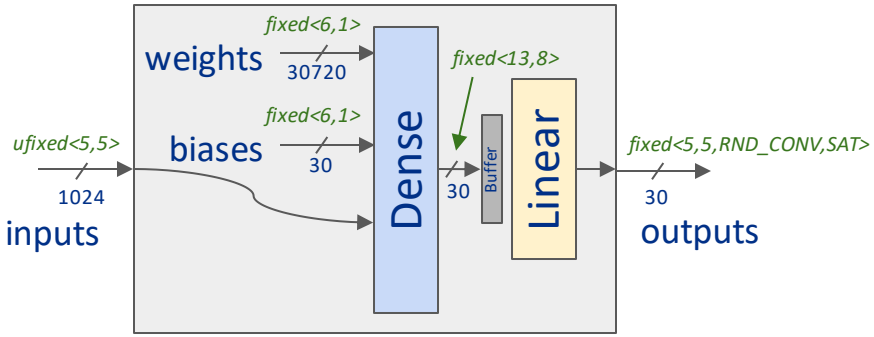


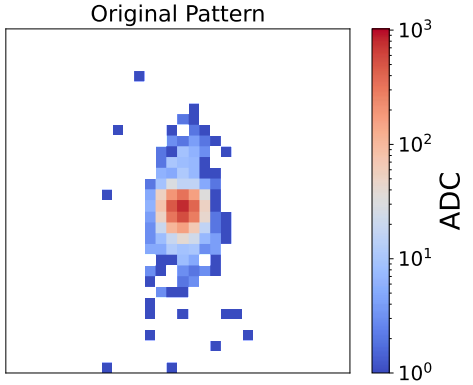
- The eigenimages are the basis of the diffraction patterns, not the initial images, and the characteristics of the diffraction pattern depend more on the experimental setup than the actual object.
- For the same experimental setup, the eigenimages from two objects are roughly the same and can be interchangeable
- **Eigen values generated from the mandrill image as the training data**
- **PCA performed on Algae data set**
- **Acceptable image reconstruction performance**
- **OK to hardcode weights !!**

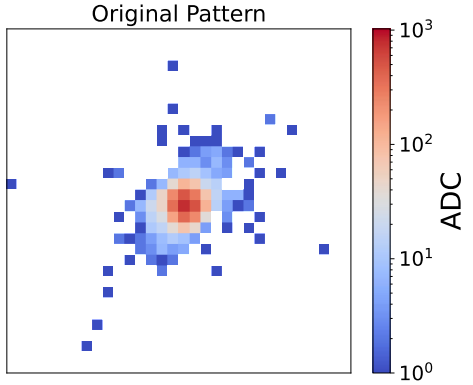
mandrill: training data

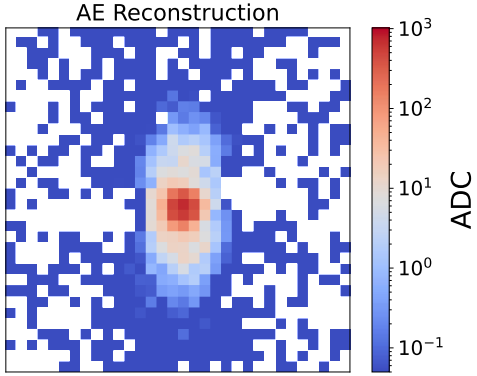
Fermilab

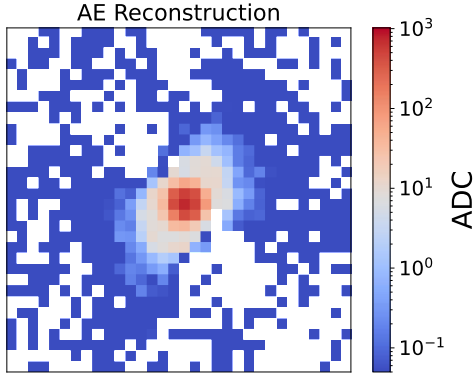
Algorithm 2: AutoEncoder (AE)











Fully-Connected Dense Layer "Encoder" Matrix

- mimic PCA implementation structure
- Outputs remain as a 30-value latent space

Weight Computation

- Determined from Quantization-Aware model training

Output Size: 30 → based on factor of compression, here 70x

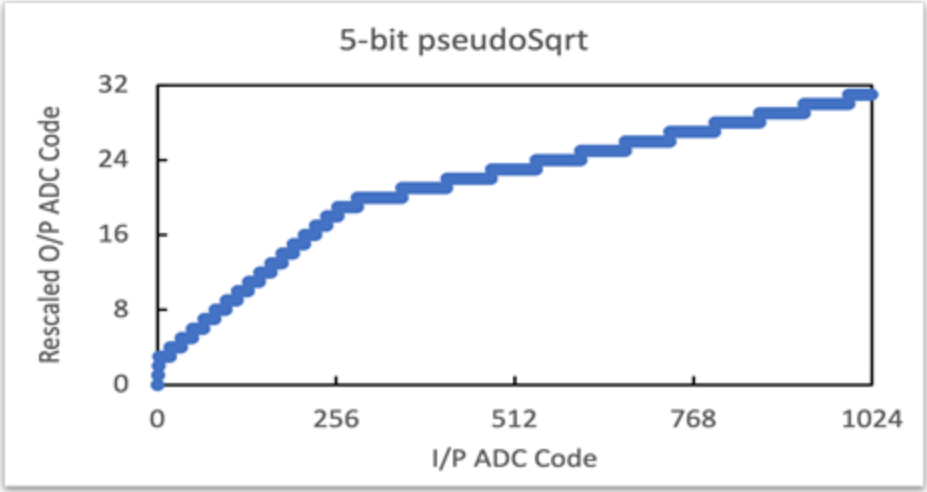
- Going from 1024 x 10-bit to 30 x 5-bit

Fermilab

AutoEncoder (AE) Algorithm

Pre-Processing Inputs

- reduce input precision (10'b ADC output) to 5'b: re-scaling
- Goal: Improve training of weights
  - Via reducing full-scale and thus increasing occupancy
- 5'b "Pseudo-square root" achieved by creating 3 gain regions: x32; x2; ÷2



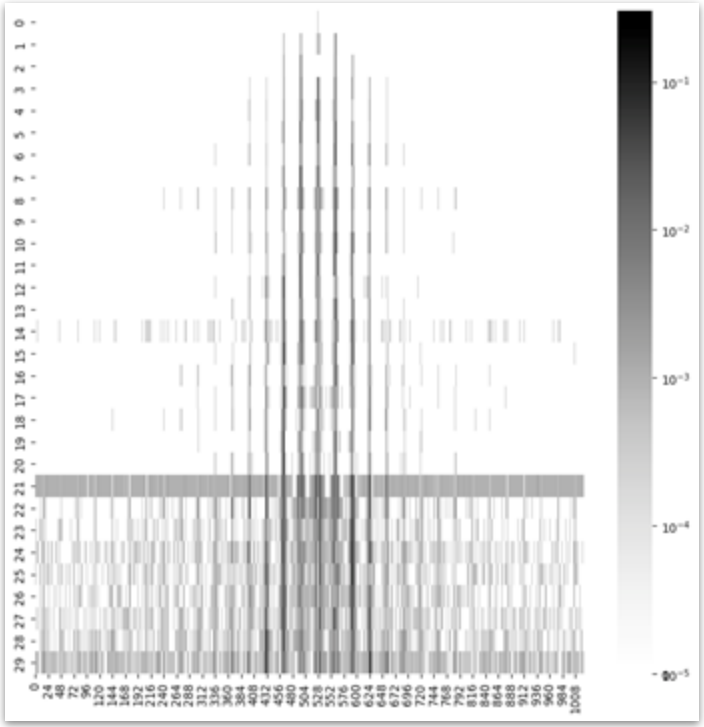
	32 x 32 pixels	400 x 400 pixels	Camera with 4 x 4 chips
TOPS-INT6 (data rate ~ 100 Kfps)	~0.06	~0.9	~14
TOPS-INT6 (data rate ~ 1 Mfps)	~0.6	~9	~140

Note INT6 vs. INT 12 for PCA – area improvement

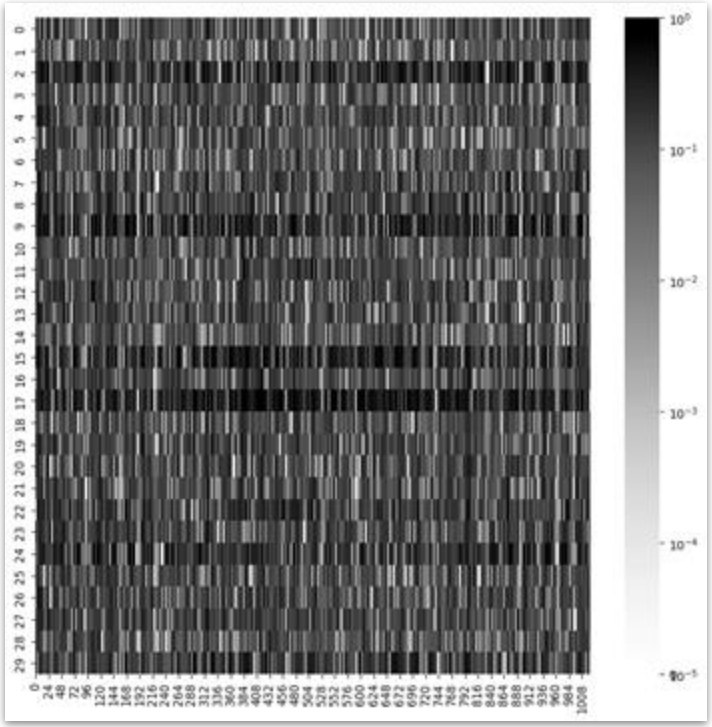
Fermilab

Weight Comparison: AE vs. PCA

- Percentage of Zero-Value Weights: PCA is 77.98 % vs. AE is 8.69%
- Weight Value Patterns per Output: none for PCA, regionalized for AE



PCA Calculated Weight Distribution



AE Post-Training Weight Distribution

Fermilab

Key takeaways

- Lossy data compression is acceptable for Ptychography due to the repetitive nature of the data
- Weights can be hardcoded since these don’t need to be recomputed for new images
- Heat maps of the weights from PCA and AE indicate the different implementation methods might be needed
- We wanted to avoid dead areas in a large area sensor and avoid moving data across the large ROIC and chose to investigate in-pixel implementation instead of chip periphery

High-Level Synthesis Implementation

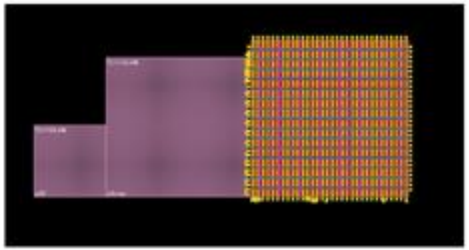
Overcoming Congestion

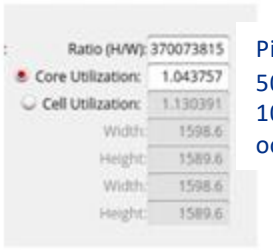
- **Fully-Dense Architecture for for both AE and PCA**
- **Challenge: Physical Interconnectivity of inputs and weights**

Post synthesis results for DNN.syn_opt12.v3 (beast1)

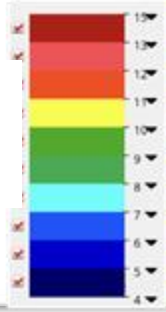
Instance	Cell count	Cell area	Net Area	Comb.	Buffer	Inverter	Flop	Total area
DNN	165,211	553,451.040	299,452.299	485,828.640	1,871.280	24,536.520	41,214.600	852,903.339

Latch, Clock-Gate, Mem and Macro areas are omitted as they are all 0

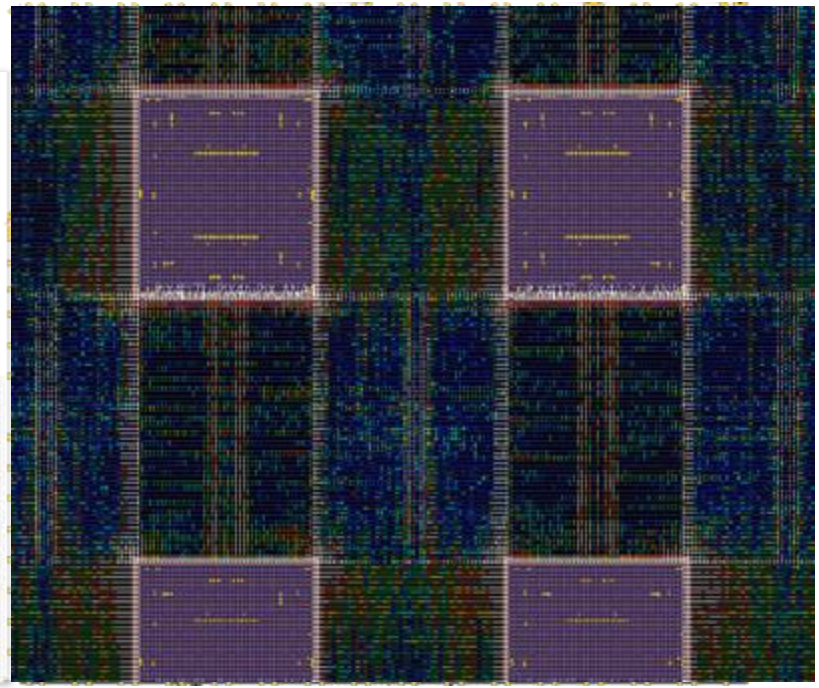




Pixel size to 50μm x 50 μm 104% occupancy



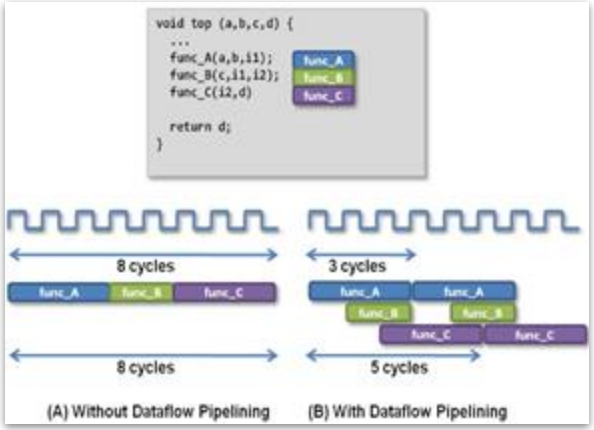
Increased pixel size to 75μm x 75 μm



Closeup of Congestion Map: zoomed around the chip's central regions

Fermilab

HLS Directives - Experiments



Xilinx: **Pipelining**

Niansong Zhang: **Unrolling**

Trade latency for resources

Inherent Data Patterns

- Diffraction “rings”: results from regional resources may cancel
- Reduce amount of latching/storage

Fermilab

Overcoming Congestion

RTL+Logic Synthesis Level

Physical Design Level

HLS Experiments: Design Conclusions

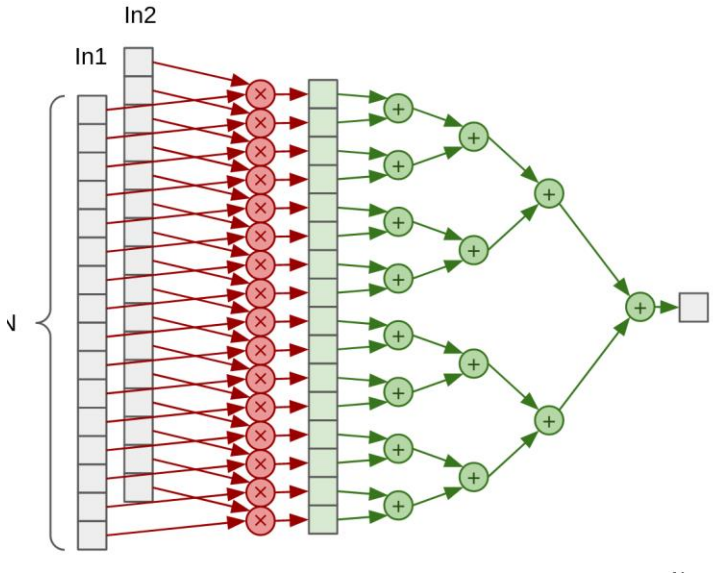
HLS Solution	AE		PCA	
	Latency	Area (mm^2)	Latency	Area (mm^2)
modular	30	0.549	30	1.516
in-line	1	1.700	1	0.652

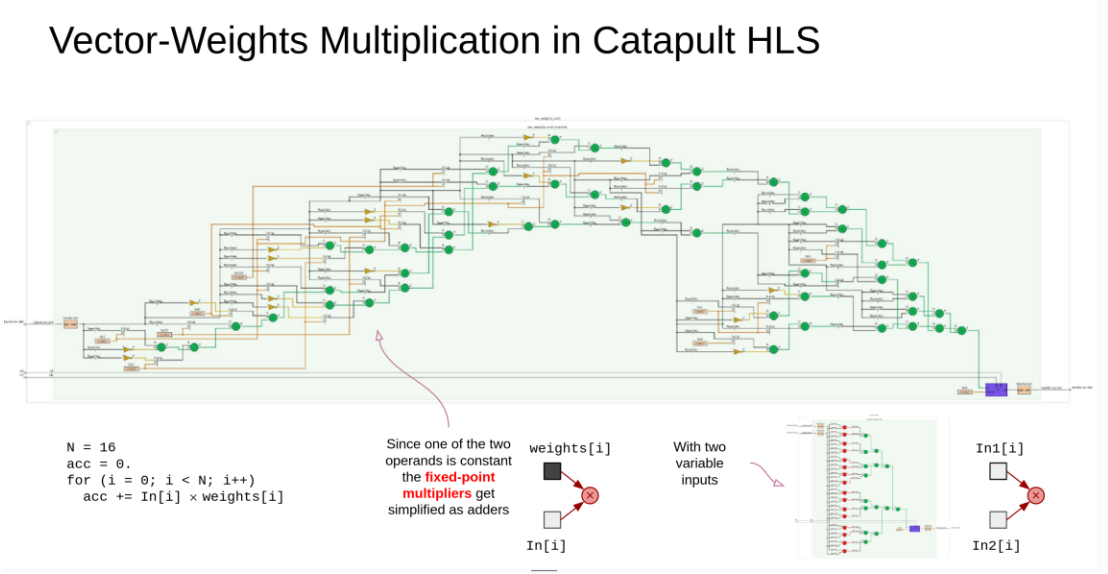
AE Restructured the weight buffers to perform 1,024 multiplication in parallel and efficiently pipeline them across each of the 30 output values

vs.

PCA Inlined all of the HLS-code functions and unrolled all the loops to take advantage of the sparsity of weights

Different approaches for AE & PCA





AE one multiplier per pixel
pipeline over 30 clock cycles

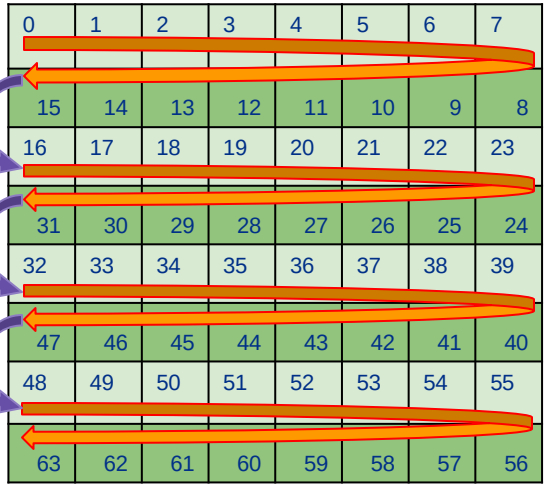
PCA: Constant propagation.
Multipliers are replaced by fixed point adders

Fermilab

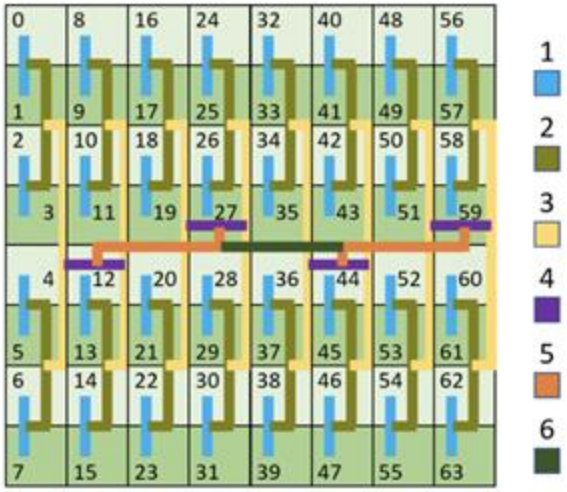
Physical Design

Three Branches:

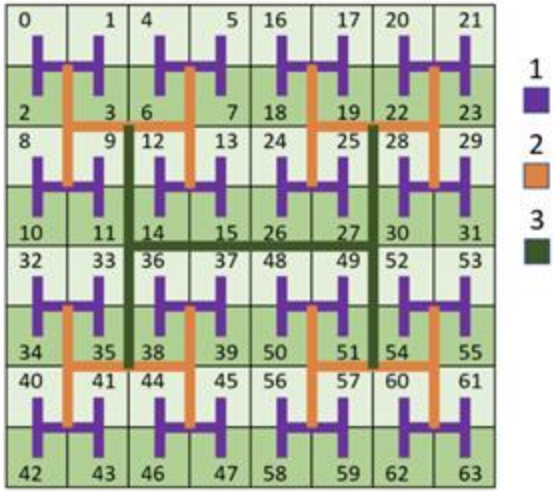
- Optimizing Pitch between pixels
- Logic Flow: Accumulation
  - Initial RTL - S tree (System Verilog) → 1D tree → changed to H tree
- Pixel Arrangement to decrease routing lengths



Initial S-Tree Logic Flow



1D hierarchy



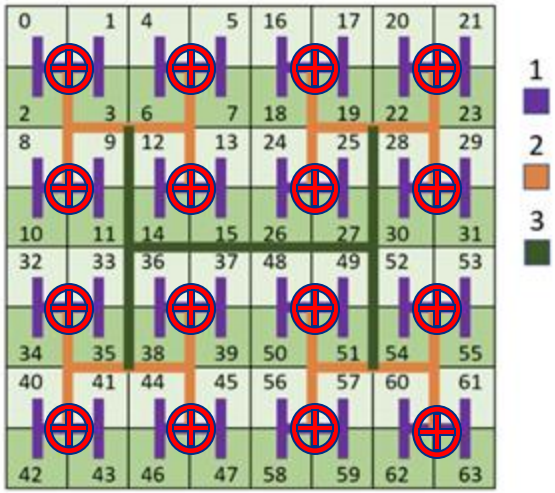
2D hierarchy

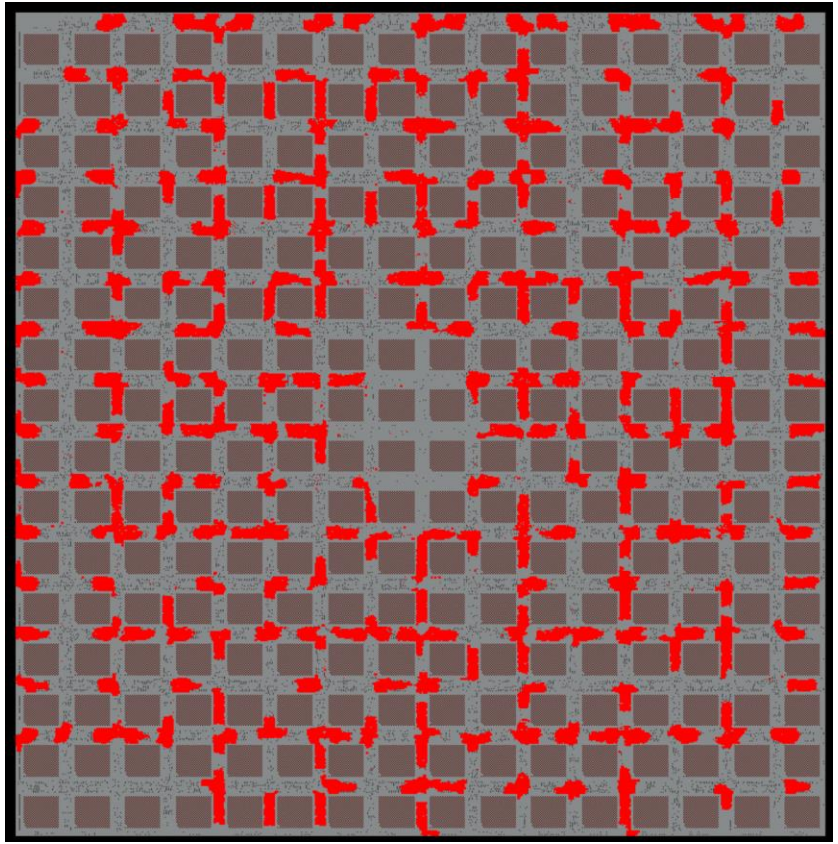
Fermilab

Physical Design

Three Branches:

- Optimizing Pitch between pixels
- Logic Flow: Accumulation
  - Initial RTL - S tree → changed to H tree
- Pixel Arrangement to decrease routing lengths





Fermilab

Changes to HLS for optimization and better layout routing

```
#define H 32 // height
#define W 32 // width
#define C 30 // channels

void ae_top(
  input_t inputs[H*W],
  weights_t weights[H*W][C], // Channel last
  biases_t biases[C],
  outputs_t outputs[C]) {

  accum_t accum[C];

  for (u32 j = 0; j < C; j++) {
    accum[j] = biases[j];
  }

  for (u32 i = 0; i < H*W; i++) {
    for (u32 j = 0; j < C; j++) {
      accum[j] += inputs[i] * weights[i][j];
    }
  }

  for (u32 j = 0; j < C; j++) {
    outputs[j] = accum[j];
  }
}
```

Multiplication last

Creates a 1D hierarchy

```
#define H 32 // height
#define W 32 // width
#define C 30 // channels

#define B 4 // block size

void ae_top(
  input_t inputs[H*W],
  weights_t weights[C][H*W], // Channel first
  biases_t biases[C],
  outputs_t outputs[C]) {

  for (u32 j = 0; j < C; j++) { // Pipeline
    accum_t accum = biases[j];
    for (u32 i = 0; i < (H*W)/B; i++) { // Unroll
        accum_t sub_accum = 0.0;
        for (k = 0; k < B; k++) { // Submodule, unroll
            sub_accum += inputs[i*B+k] *
                         weights[j][i*B+k];
        }
        accum += sub_accum;
    }
    outputs[j] = acc;
  }
}
```

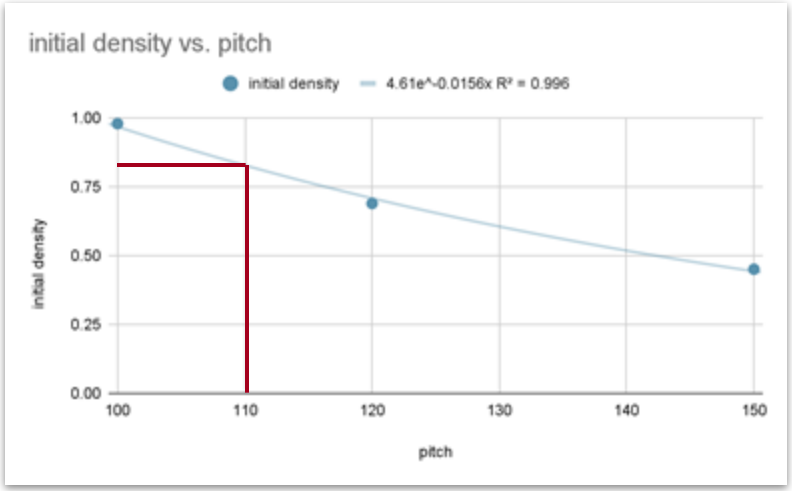
Multiplication first

Create a 2D hierarchy

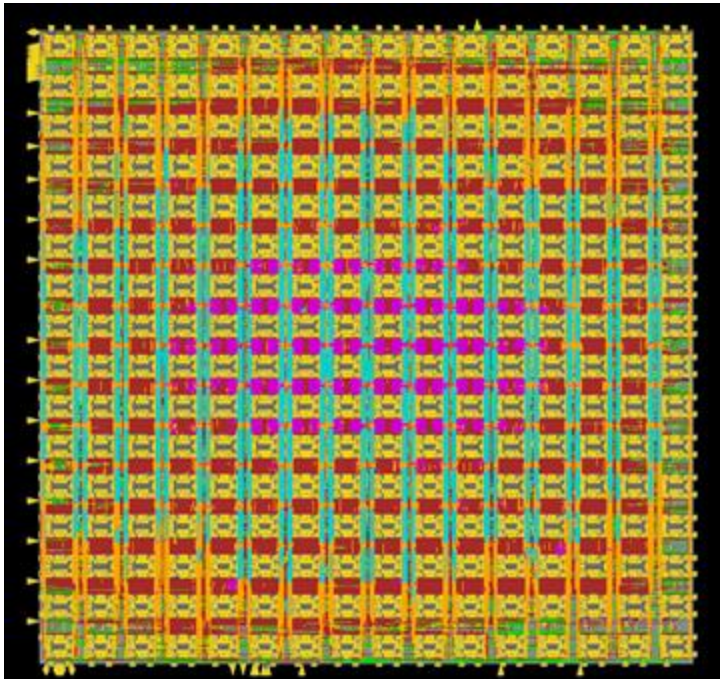
Physical Design

Three Branches:

- Optimizing Pitch between pixels
- Logic Flow: Accumulation
  - Initial RTL - S tree → changed to H tree
- Pixel Arrangement to decrease routing lengths



80% occupancy cutoff @pitch=112 → proceed with 110



With pitch=110, post-Place&Route:

- No congestion issues
- Logic distributed *throughout* quadrant
- DRC clean

Fermilab

What about a reprogrammable version

PCA –

- Massive multiplier size (12b multipliers much larger than 6b multiplier) – *Maybe Not the best path forward*

Autoencoder -

- only: 1024 multipliers (6b) already on chip
- Pixel area for encoding weights: 30x6b weights – adding registers in the pixel area; (maybe SRAMs would be smaller) if pixel size grows to 75μm x 75μm