



In-Pixel AI: From Algorithm to Accelerator

Priyanka Dilip^{1,3}

on behalf of:

Manuel Valentin², Danny Noonan¹, Giuseppe di Guglielmo¹, Panpan Huang²,
Chris Jacobsen², Seda Memik², Nhan Tran^{1,2}, Farah Fahim^{1,2}

¹Fermilab, ²Northwestern University, ³Stanford University

Coordinating Panel for Advanced Developers (CPAD) Workshop

Solid State Detectors and ASICs

November 2022

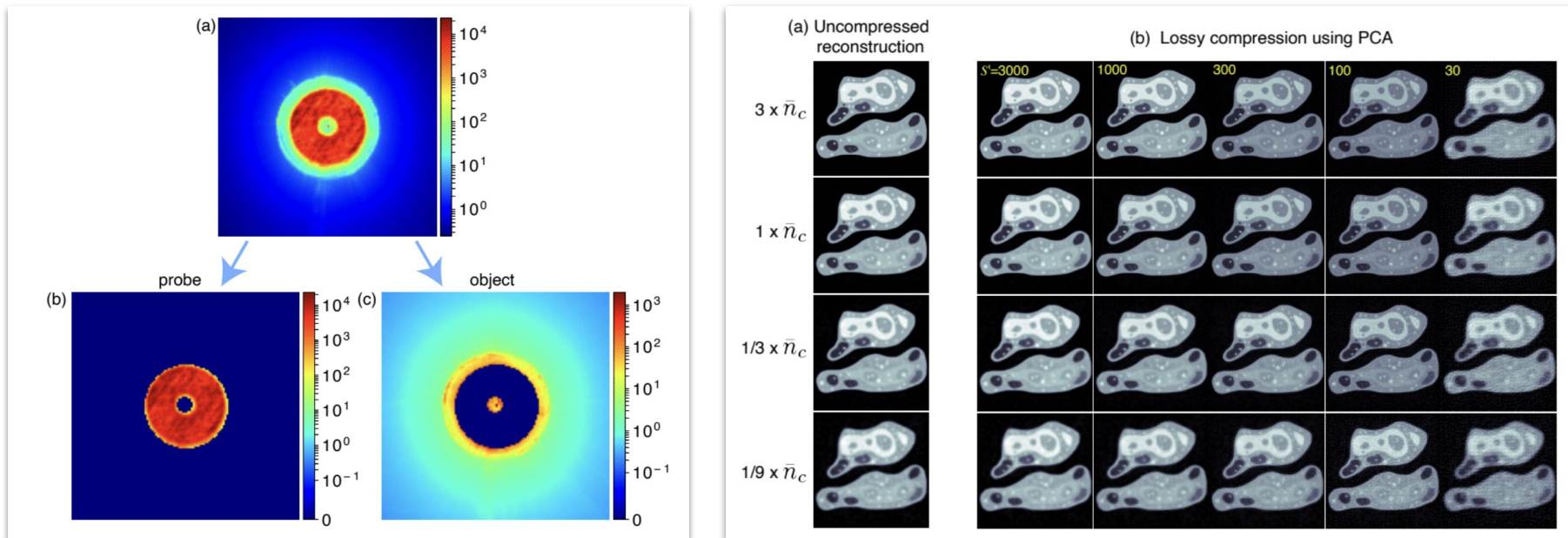
Outline

- Physics Motivation
- Algorithm Co-Design
- High-Level Synthesis Flow
- Overcoming Congestion
 - Logic Synthesis level → Hierarchical Design
 - Physical Design level → Logic Flow, Pixel Arrangement
- Future Work *and* Design Insights



Motivation and Approach

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 for backend

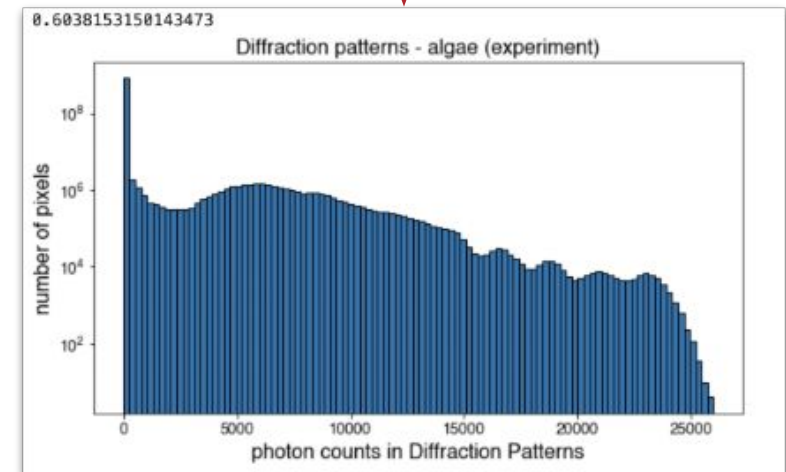
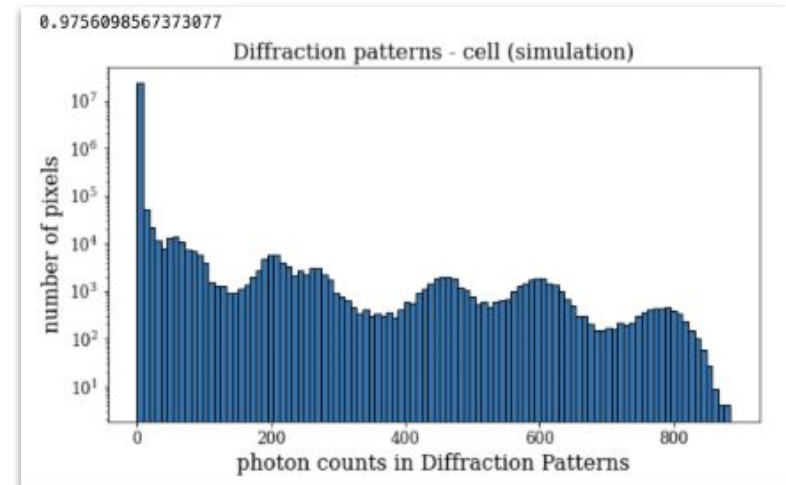
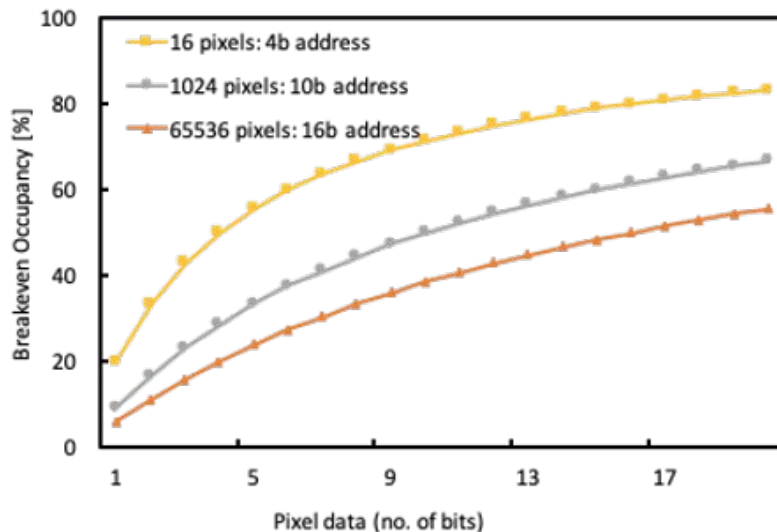
Why data compression instead of zero-suppression?

Data rate:

400×400 (pixels per chip) \times 10b (ADC) \times 1 MHz
 ~ 1.6 Tbps

Data Sparsification:

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

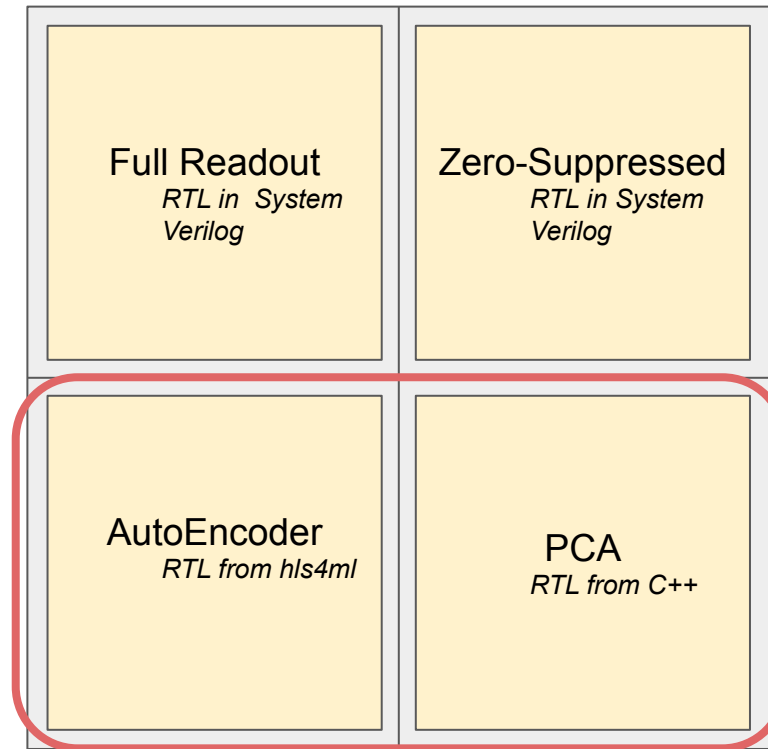


In-pixel AI/ML to enable Mfps camera operation

- ❑ 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
- ❑ 1.6Tbps data → 32Gbps enabled by **50x** lossy data compression (1x 32 Gbps photonic link or 3 x 10.24 Gbps link) per chip (~4cm²)

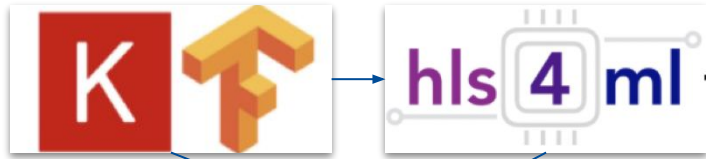
	Charge Integration	Analog to Digital Conversion	Data Compression	Off-chip Data Transfer
Subchip = 1024 pixels for 100 Kfps operation	~1μs	~9μs	Full-readout	1024*10b @ 1GSPS = 10μs
Subchip = 1024 pixels for 1Mfps operation	~0.1μs	~0.9μs	Data compression PCA: 50x 1 cycle: 25n AE: 70x 30 cycles: 750n	220b@ 1GSPS = 220n 150b @ 1GSPS = 150n

Design Parameters



- **Target technology:**
 - TSMC 65nm
- **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**

High-Level Synthesis: Catapult + flow

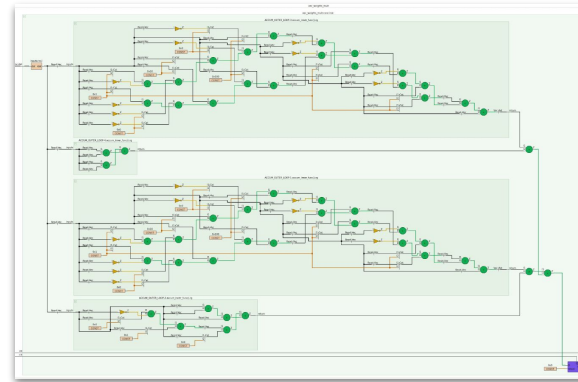


```
[pdilip@ml4asic01 src]$ cat PCA_HLS.cpp
#include "PCA_HLS.h"

#pragma hls_design top
void CCS_BLOCK(mmult) (
    matrix_t in1[MSIZE1],           // MSIZE1 (1 * 32 * 32)
    matrix2_t in2[NumPixels][n_limit], // MSIZE2 (32 * 32 * 30)
    matrix3_t Out[MSIZE3]           // MSIZE3 (1 * 30)
)
{
    static inter_t local_sum[n_limit];
    static inter_t blkRes[BlkNum][n_limit];

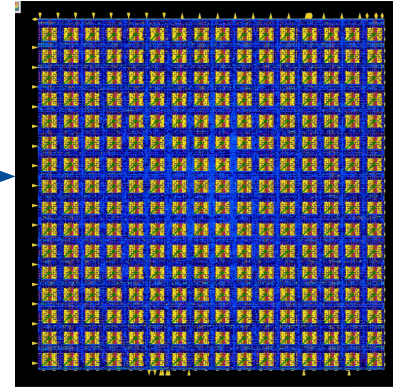
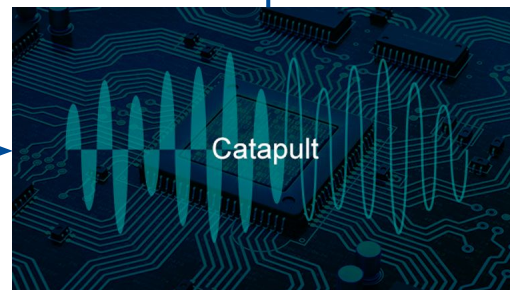
    blk_mmult; for (int i = 0; i < BlkNum; i++) {
        /* #pragma unroll yes */
        Second_dimension; for (int col = 0; col < BlkSize; col++) {
            /* #pragma unroll no */
            Third_dimension; for (int k = 0; k < n_limit; k++) {
                inter_t result = (col == 0) ? (inter_t)0 : local_sum[k];
                result += in1[i * BlkSize + col]*in2[i * BlkSize + col][k];
                local_sum[k] = result;
                if(col == BlkSize - 1) blkRes[i][k] = result;
            }
        }
    }

    static inter_t Res[n_limit];
    /* #pragma unroll yes */
    result_out; for (int j = 0; j < n_limit; j++) {
        /* #pragma unroll no */
        accum; for (int i = 0; i < BlkNum; i++) {
            Res[j] += blkRes[i][j];
        }
        Out[j] = Res[j];
    }
}
```



RTL --> Design Analyzer

pragmas +



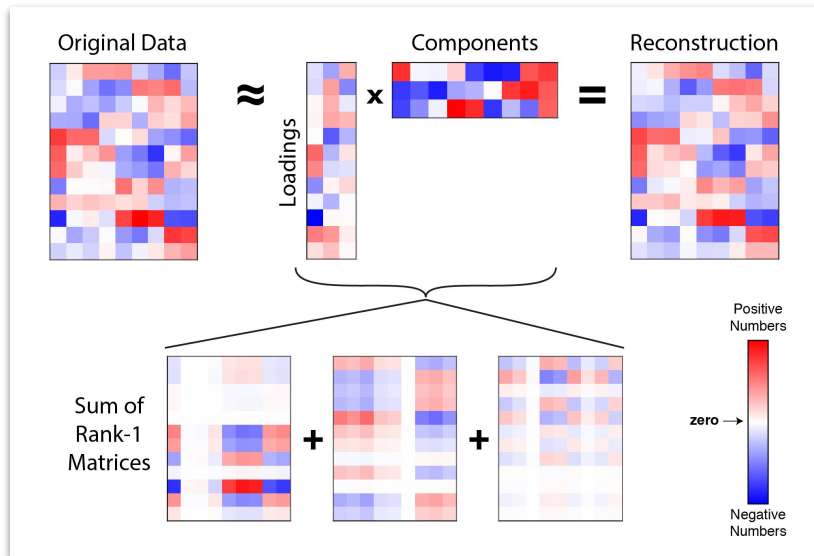
Physical Design
(digital AI logic
“mesh” around
analog “islands”)

Tapeout

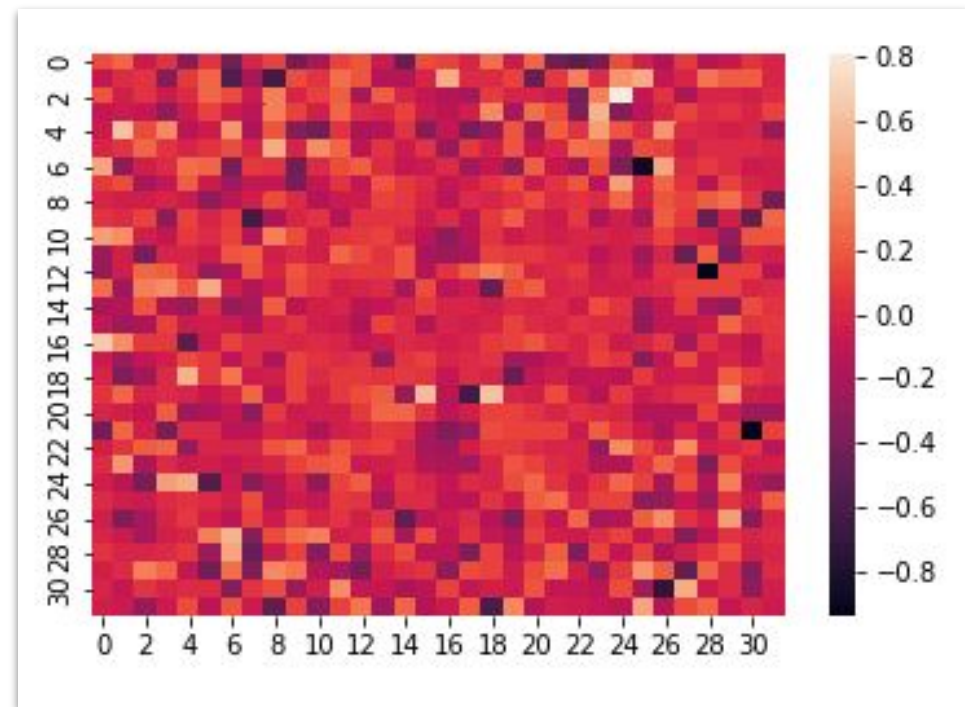


Algorithm Co-Design

Algorithm 1: Principal Component Analysis (PCA)



Alex Williams, *It's Neuronal Blog*



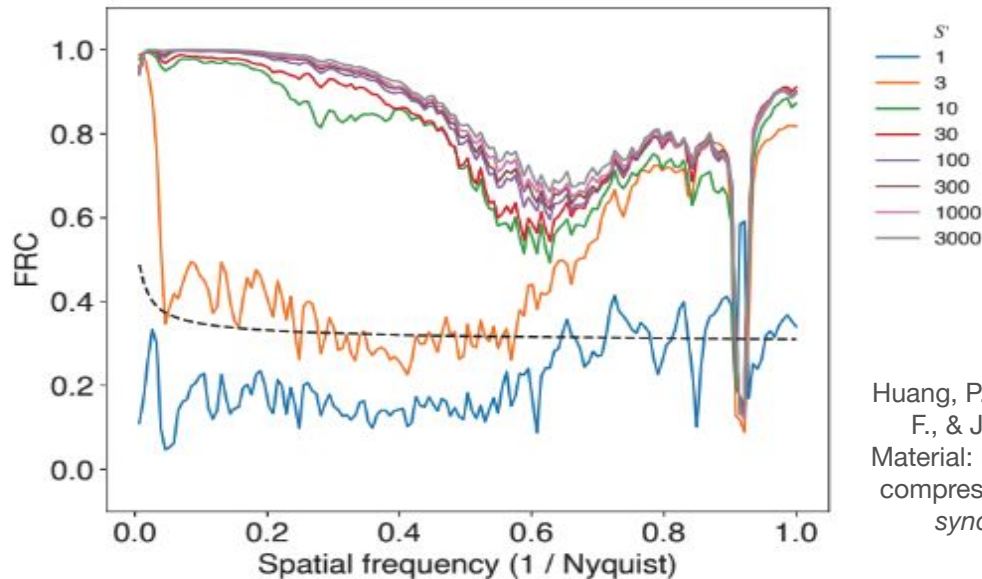
Calculated Weight Distribution, PCA

Goal: Reasonable Reconstruction of original diffraction image

Parameter Choice: 30 outputs \leftarrow tradeoff compression vs. quality loss

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

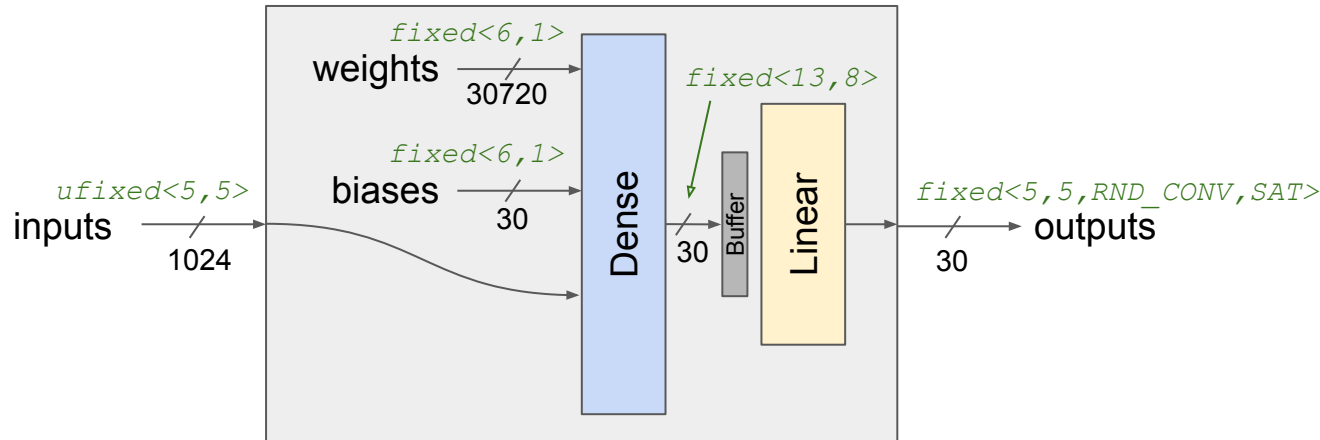
PCA Algorithm: C++ Realization



Huang, P., Deng, J., Noonan, D., Tran, N., Fahim, F., & Jacobsen, C. (2020). Supplementary Material: Matrix Factorization approaches to lost compression of ptychographic data. *Journal of synchrotron radiation*, Preprint, 43-50

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

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

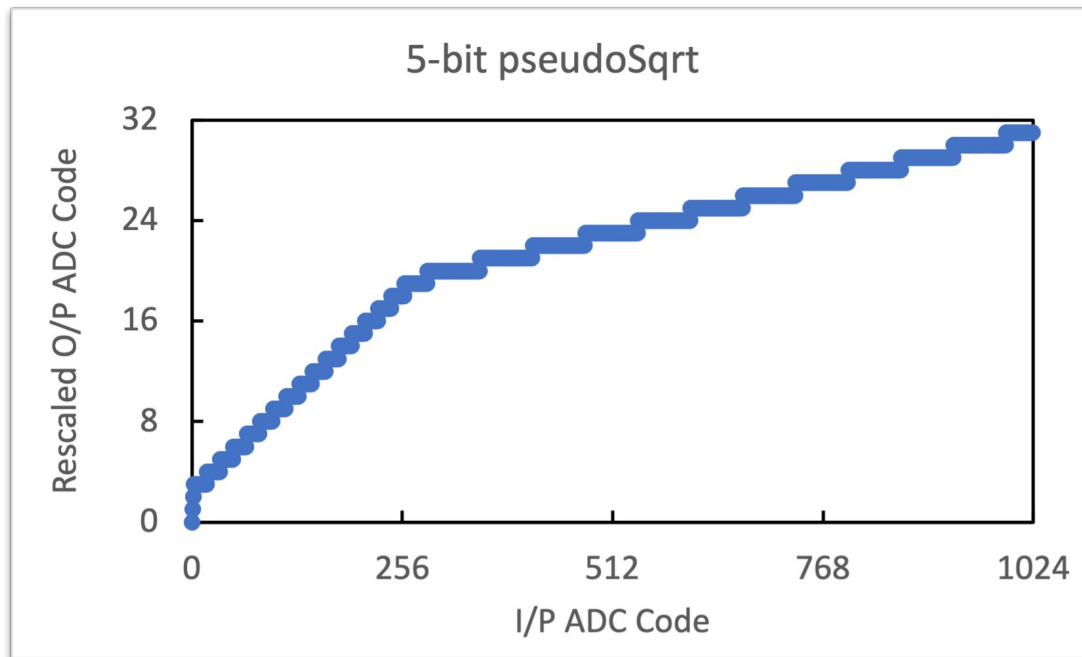
Output Size: 30 → based on factor of compression, here 70x

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

AutoEncoder (AE) Algorithm

Pre-Processing Inputs

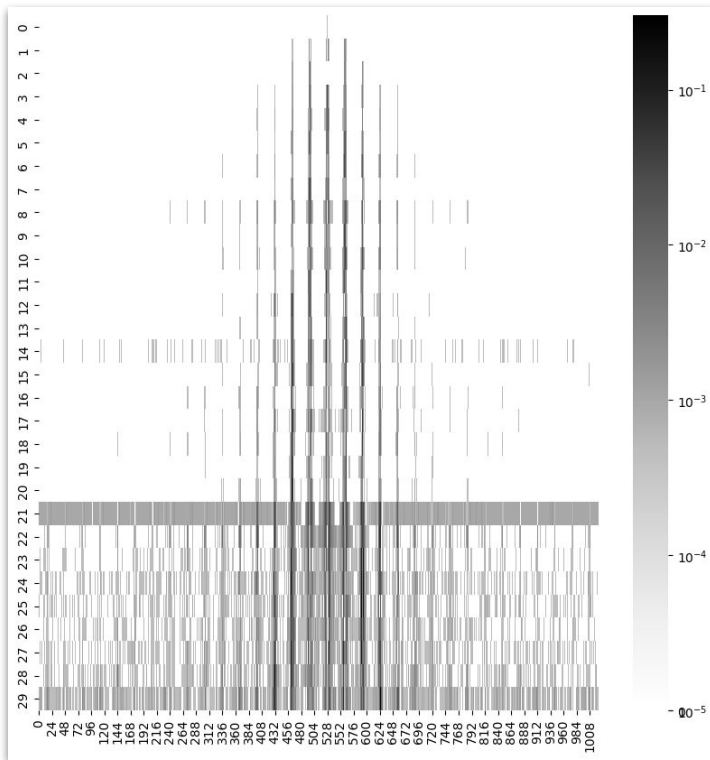
- reduce input precision (10'b ADC output) to 5'b: re-scaling
- Goal: Improve training of weights
 - Via reducing full-scale and thus increasing occupancy
- 5'b “Pseudo-square root” achieved by creating 3 gain regions: $\times 32$; $\times 2$; $\div 2$



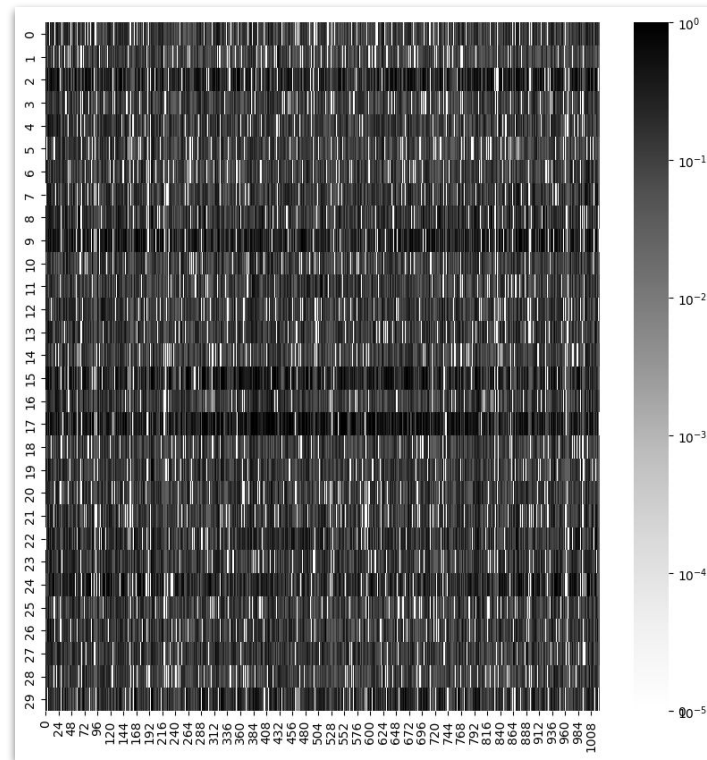
Weight Comparison: AE vs. PCA

Sparsity Difference in weights

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



PCA Calculated Weight Distribution

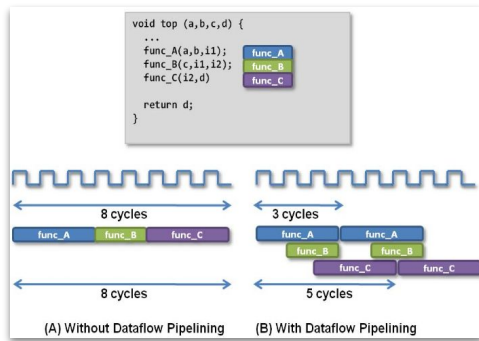


AE Post-Training Weight Distribution

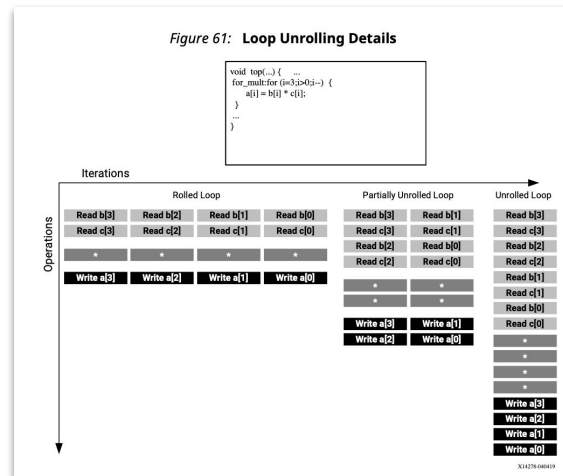


High-Level Synthesis Implementation

HLS Directives - Experiments



Xilinx: **Pipelining**



Niansong Zhang: **Unrolling**

```

foreach $opt(version) {
    "v02" {
        directive set /$ACCELERATOR/core/Second_dimension -UNROLL yes
        directive set /$ACCELERATOR/core/Third_dimension -UNROLL yes
        directive set /$ACCELERATOR/core/result_out -UNROLL yes
        directive set /$ACCELERATOR/core/accum -UNROLL yes
    }
    "v03" {
        directive set /$ACCELERATOR/core/Second_dimension -UNROLL no
        directive set /$ACCELERATOR/core/Third_dimension -UNROLL no
        directive set /$ACCELERATOR/core/result_out -UNROLL yes
        directive set /$ACCELERATOR/core/accum -UNROLL yes
    }
    "v04" {
        directive set /$ACCELERATOR/core/Second_dimension -UNROLL yes
        directive set /$ACCELERATOR/core/Third_dimension -UNROLL yes
        directive set /$ACCELERATOR/core/result_out -UNROLL no
        directive set /$ACCELERATOR/core/accum -UNROLL no
    }
    "v05" {
        directive set /$ACCELERATOR/core/Second_dimension -UNROLL 16
        directive set /$ACCELERATOR/core/Third_dimension -UNROLL no
        directive set /$ACCELERATOR/core/result_out -UNROLL yes
        directive set /$ACCELERATOR/core/accum -UNROLL yes
    }
    "v06" {
        directive set /$ACCELERATOR/core/Second_dimension -UNROLL 16
        directive set /$ACCELERATOR/core/Third_dimension -UNROLL no
        directive set /$ACCELERATOR/core/result_out -UNROLL yes
        directive set /$ACCELERATOR/core/accum -UNROLL no
    }
}
    
```

```

} else {
    directive set /$ACCELERATOR/core/local_sum:rsc -MAP_TO_MODULE {[Register
]]}






    directive set /$ACCELERATOR/core/blkRes:rsc -MAP_TO_MODULE {[Register]}
    directive set /$ACCELERATOR/core/Res:rsc -MAP_TO_MODULE {[Register]}
}
    
```

Trade latency for extra resources

Inherent Data Patterns

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






HLS Experiments: Early Results






Solution 	Latency...	Latency...	Throug...	Throug...	Slack	Total Area
 solution.v1 (new)						
 mmult.v02 (allocate)	31	4843.75	32	5000.00		930782.35
 mmult.v03 (extract)	30719	4799843.75	30720	4800000.00	144.22	459875.92
 mmult.v04 (extract)	991	154843.75	992	155000.00	149.74	862568.28

PCA:

- Unrolling: Full (2) vs Partial (3,4)

Report: General



Solution	Latency Cycles ▾	Latency Time	Throughput Cycles	Throughput Time	Slack	Total Area
 myproject.v02 (compile)	30720	4800000.00	30720	4800000.00	150.24	37447.20
 myproject.v04 (extract)	1024	160000.00	1024	160000.00	152.25	41857.96
 myproject.v05 (extract)	960	150000.00	960	150000.00	129.17	74553.44
 myproject.v03 (extract)	30	4687.50	30	4687.50	141.78	492316.58
 myproject.v01 (extract)	0	0.00	0	0.00	143.00	1144014.73

AE:

- Combinational (1) vs Sequential (2+)
- Unrolling: 2-4
- Initial Hierarchical: 5

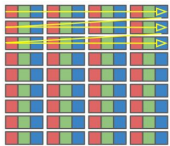
Hierarchical Design: Logic Structure

NHWC vs. NCHW Representation

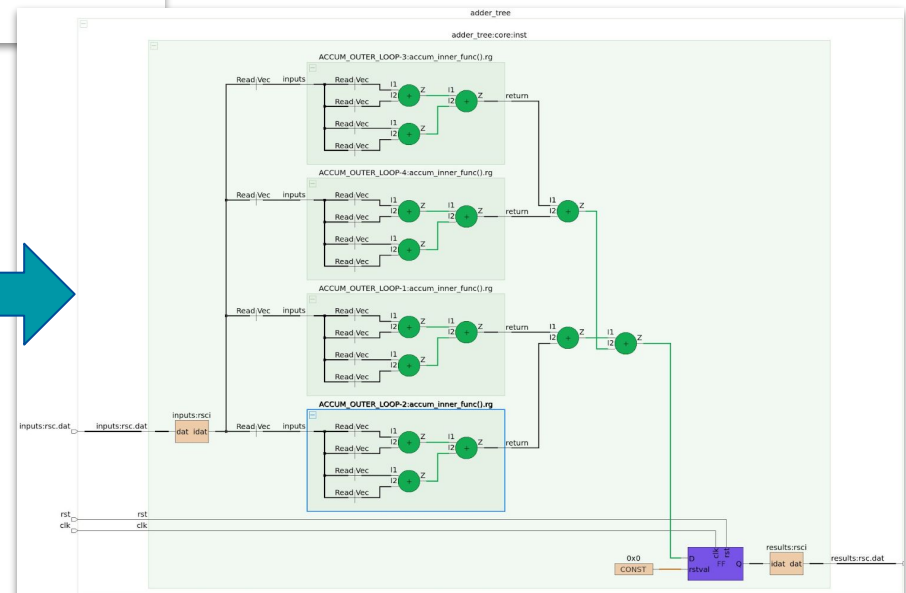
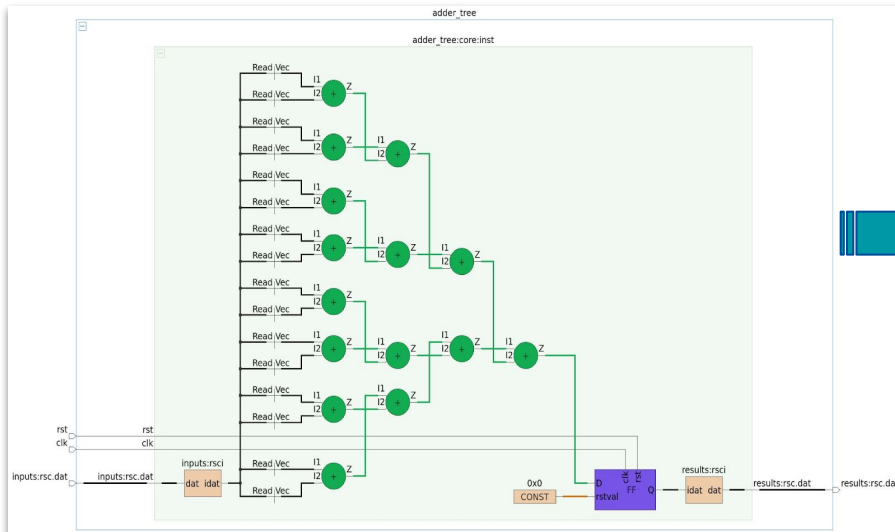
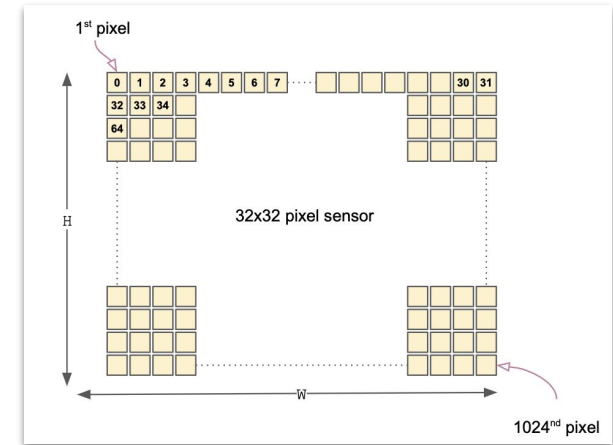
Channel last Channel first

- Batch, Number of images
- Height
- Width
- Channels
- TensorFlow uses NHWC (channel last) representation
 - [Transpose script](#)
- Representation affects performance (CPU, GPU, ...)

NHWC



NCHW



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



Overcoming Congestion



RTL+Logic Synthesis Level

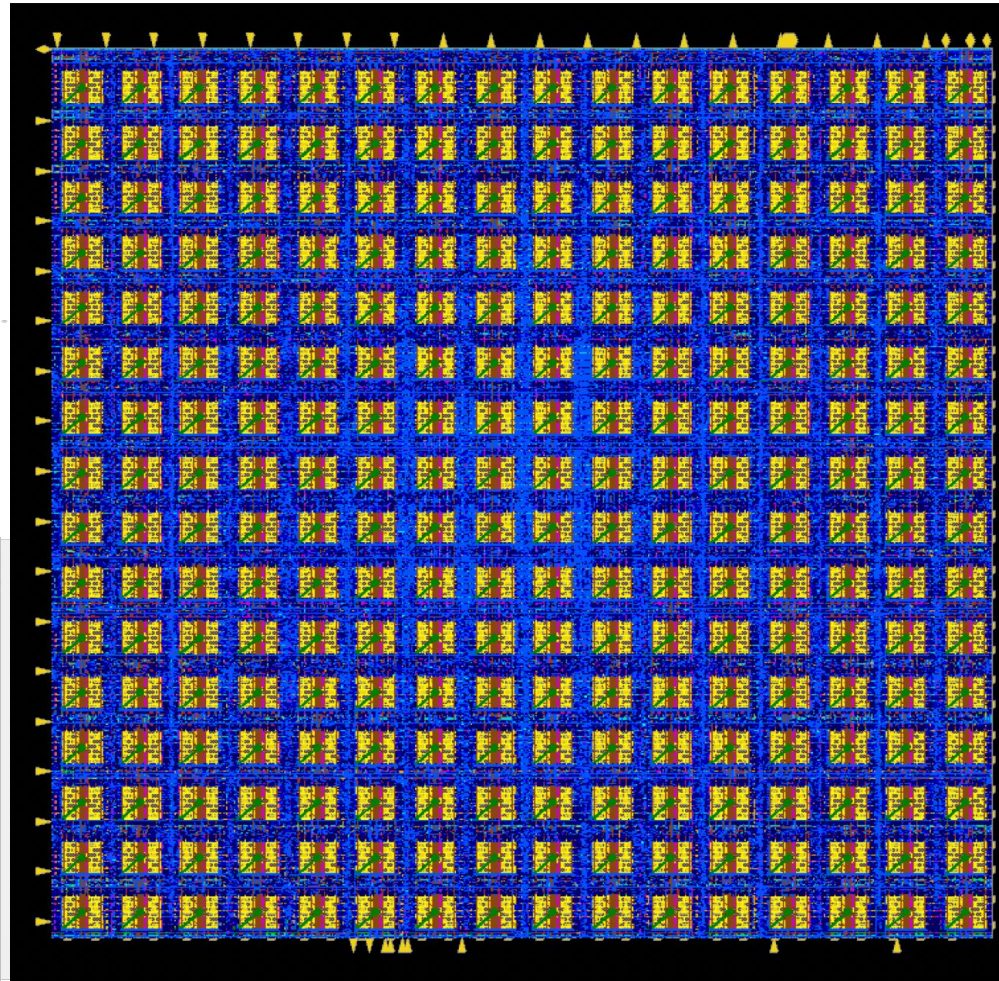
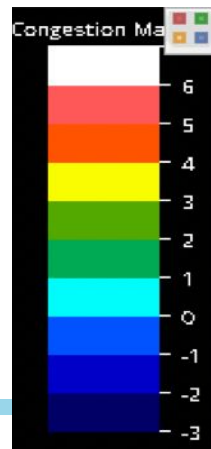


Physical Design Level

Congestion: Causes

As pixel number grows...

- more MACs needed for MatMults
- routing wire lengths increase
- unwaivable DRC violations during PnR
- distance between pixels must increase
BUT pushing limits of total chip area



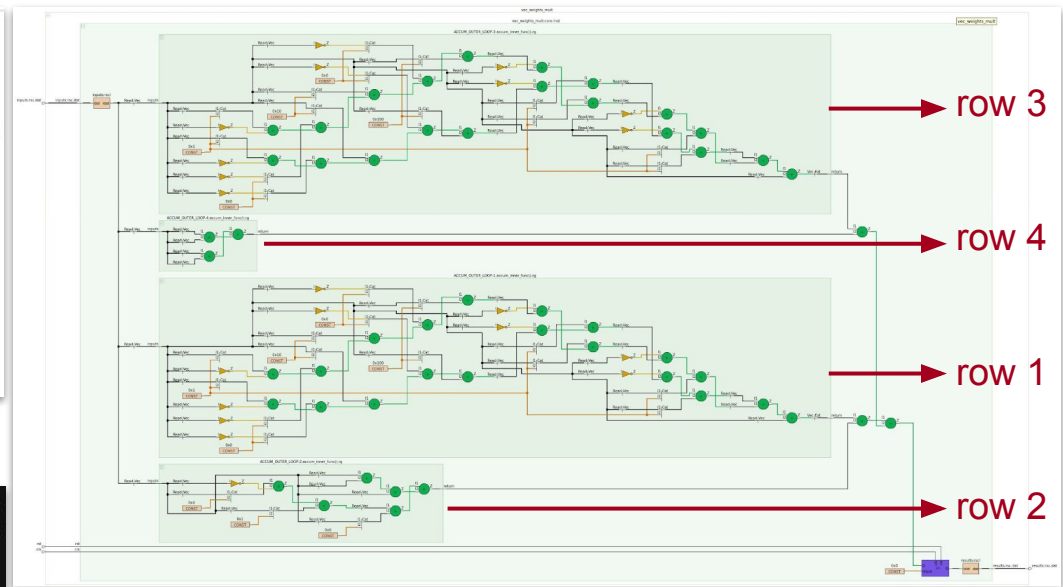
Hierarchical Design

```
N = 32
B = 4
acc = 0
ac_fixed_pd block_func(ac_fixed_pd block[B])
    block_acc = 0
    for (j = 0; j < B; j++)
        block_acc += block[j]
}

for (i = 0; i < N/B; i++)
    block_func(In + i*B)
```

“Soft” Blocks during C++ spec

```
static data_t weights[N_IN] = {
0.025, 0.025, 0.125, 0.0, row 1
0.125, 8.25, 4.75, 0.25, row 2
0.75, 12.575, 8.50, 0.125, row 3
1., 1., 1., 1. row 4
};
```



https://github.com/GiuseppeDiGuglielmo/AlinPixel_hls/tree/main/workspace_c_atapult/adder_tree_16

Post-Implementation of Hierarchical Design in HLS

- Balance between logic-heavy blocks in the center *and* sparsity at the edges
- Carried forward through to synthesis, “fencing” during physical design

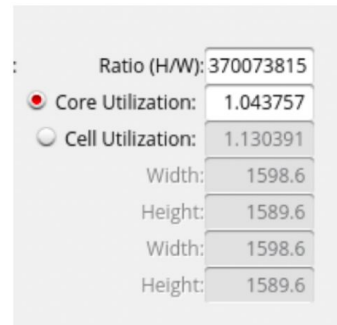
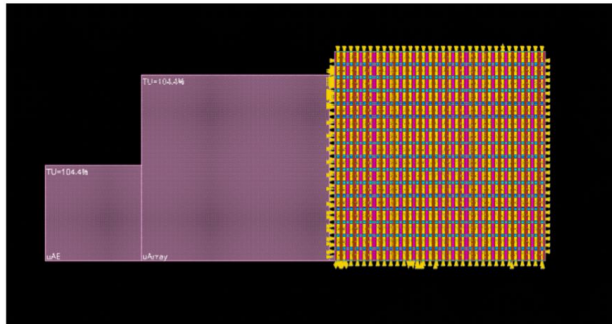
Overcoming Congestion

- Fully-Dense Architecture for AE → Echo PCA's structural implementation
- 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



Pitch 100 Floorplan
Initial (unplaced) density: **104% (!!)**

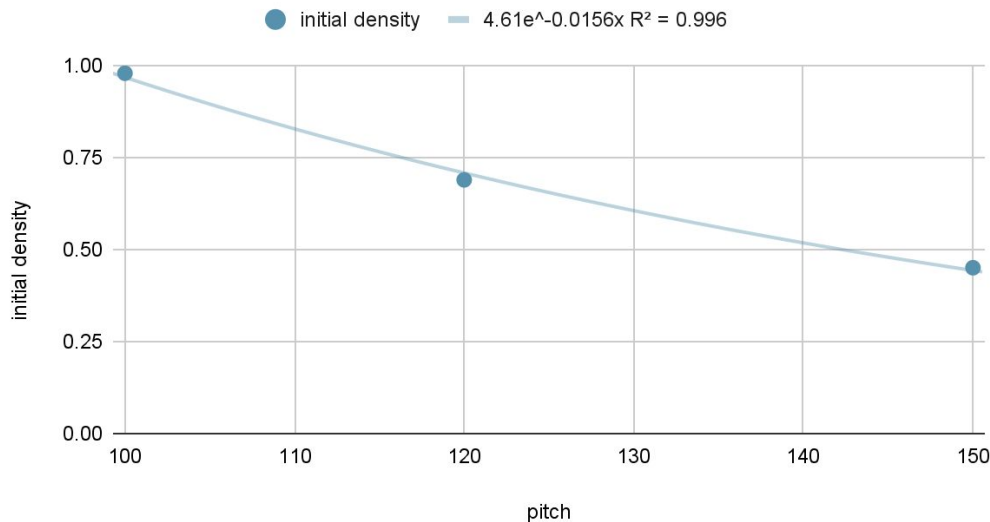
Closeup of Congestion Map: zoomed around the chip's central regions

Physical Design

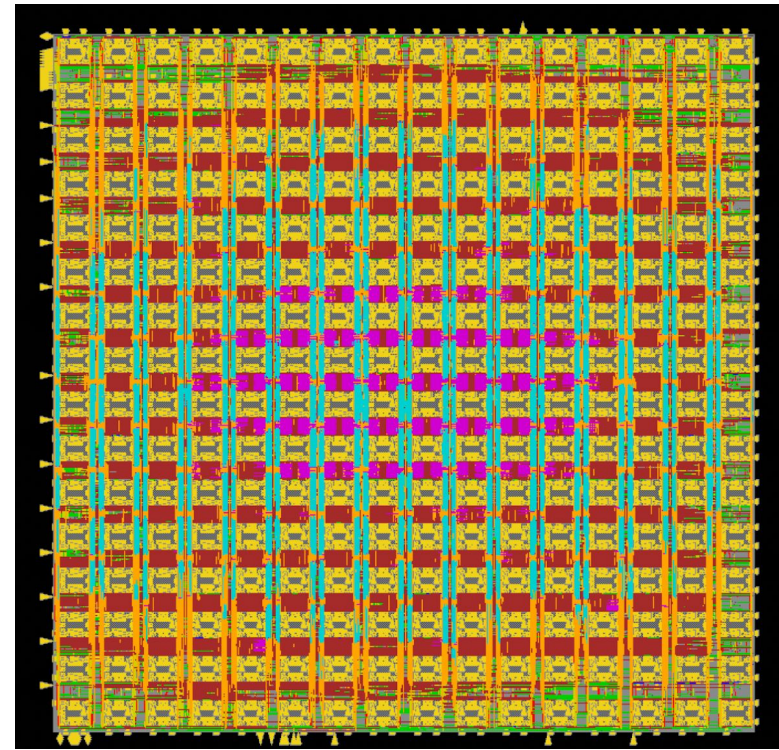
Three Branches:

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

initial density vs. pitch



80% occupancy cutoff @pitch=112 → proceed with 120



With pitch=120, post-Place&Route:

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

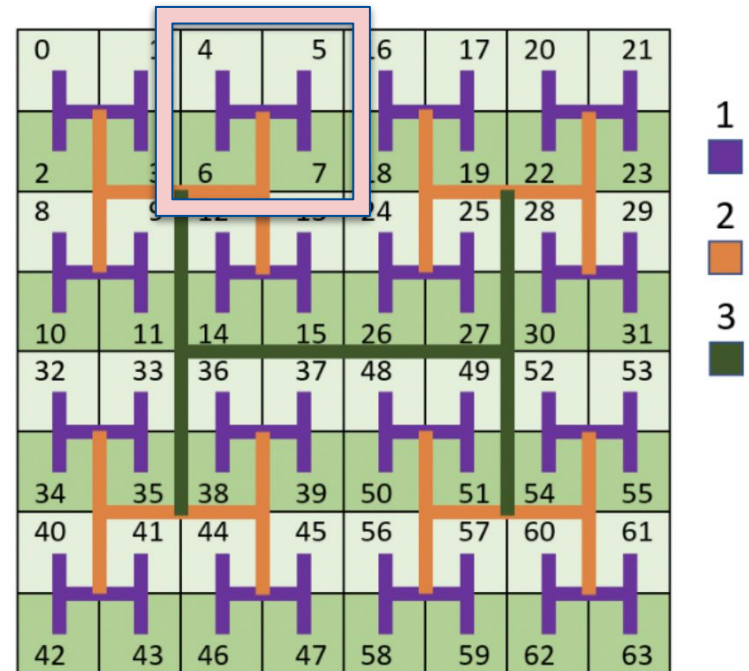
Physical Design

Three Branches:

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



Initial S-Tree Logic Flow



Fahim, F., Joshi, S., Ogren-Memik, S., & Mohseni, H. (2020). A Low-Power, High-Speed Readout for Pixel Detectors Based on an Arbitration Tree. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(2), 576-584.

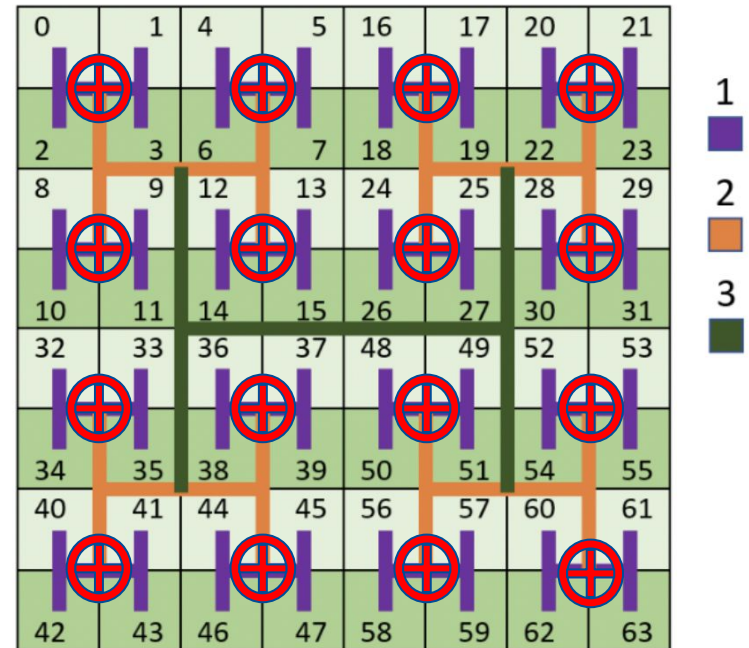
Physical Design

Three Branches:

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



Initial S-Tree Logic Flow



Fahim, F., Joshi, S., Ogresci-Memik, S., & Mohseni, H. (2020). A Low-Power, High-Speed Readout for Pixel Detectors Based on an Arbitration Tree. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(2), 576-584.

Lessons learnt for in-pixel implementation of AI

Can we comment on a design approach that can be followed for other pixel projects

Future Work

- Tapeout expected submission: Dec 7
- Comparative analysis: Among *all* quadrants
- Increase speed of front-end for 1Mfps proof of concept
- Move to reconfigurable weights from “golden weights”
 - Try: reloadable systolic array

Design Insights

- Growing pixel size can payoff for overall bandwidth reduction
 - 20% pixel size increase solved data throughput issue
- Pixel addressing schemes to leverage locality
 - at algorithm, logic synthesis, *and* floorplan levels
 - H-tree approach solved the congestion issue

References

- Fahim, F., Joshi, S., Ogrenci-Memik, S., & Mohseni, H. (2020). A Low-Power, High-Speed Readout for Pixel Detectors Based on an Arbitration Tree. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(2), 576-584. [8930978]. <https://doi.org/10.1109/TVLSI.2019.2953871>
- Moeys DP, Corradi F, Li C, Bamford SA, Longinotti L, Voigt FF, Berry S, Taverni G, Helmchen F, Delbruck T. A Sensitive Dynamic and Active Pixel Vision Sensor for Color or Neural Imaging Applications. *IEEE Trans Biomed Circuits Syst*. 2018 Feb;12(1):123-136. doi: 10.1109/TBCAS.2017.2759783. PMID: 29377801.
- Di Guglielmo, G., Fahim, F., Herwig, C., Valentin, M. B., Duarte, J., Gingu, C., ... & Tran, N. (2021). A reconfigurable neural network ASIC for detector front-end data compression at the HL-LHC. *IEEE Transactions on Nuclear Science*, 68(8), 2179-2186.
- 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. <https://doi.org/10.1107/S1600577520013326>
- Huang, P., Deng, J., Noonan, D., Tran, N., Fahim, F., & Jacobsen, C. (2020). Supplementary Material: Matrix Factorization approaches to lost compression of ptychographic data. *Journal of synchrotron radiation*, Preprint, 43-50.



In-Pixel AI: From Algorithm to Accelerator

Priyanka Dilip^{1,3}

on behalf of:

Manuel Valentin², Danny Noonan¹, Giuseppe di Guglielmo¹, Panpan Huang²,
Chris Jacobsen², Seda Memik², Nhan Tran^{1,2}, Farah Fahim^{1,2}

¹Fermilab, ²Northwestern University, ³Stanford University

Coordinating Panel for Advanced Developers (CPAD) Workshop

Solid State Detectors and ASICs

November 2022