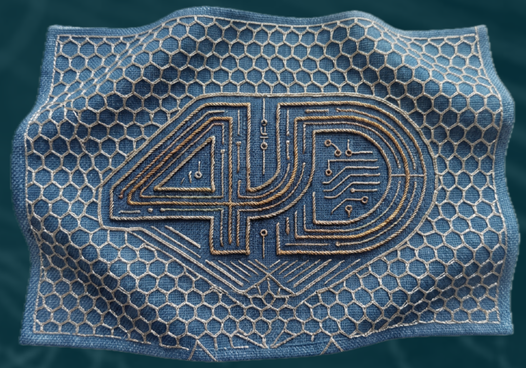
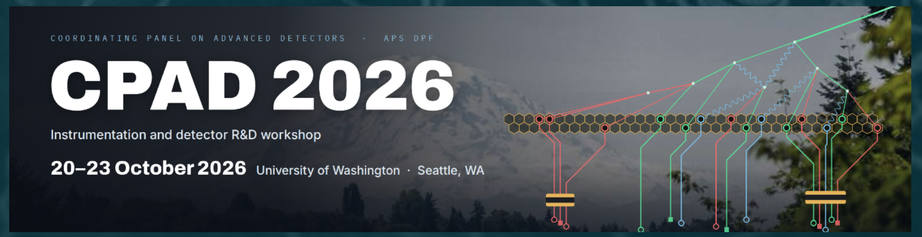


BERKELEY LAB

4D Fabric: a NIFTY solution to 4D Pixel Detectors

21.10.26



Timon Heim, Graham Anderson, Rebecca Carney,
Caleb Ceravolo (UW), Maurice Garcia-Sciveres,
Deion Fellers, Alexis Mulski - LBNL

Drivers

BERKELEY LAB

ECRP (FY26-FY30)

- **Goal**: develop 4D Pixel detector readout chip technology
- Aiming to add O(50ps) timing precision to Pixel detectors
- Leverage work done in LDRD in FY22/23, specifically Big Rock AFE
- Exploring collaboration with SLAC/FNAL
- Co-design with detector simulation -> directly understand physics performance impact

NIFTY (FY26-FY28)

- **Goal**: develop Pixel detector readout chip technology with on-chip intelligent networks
- Improve on classic column based readout by implementing mesh networking
- Emerging intelligence from simple routing rule set
- Looking for scalable architecture to full reticule size or even to wafer level

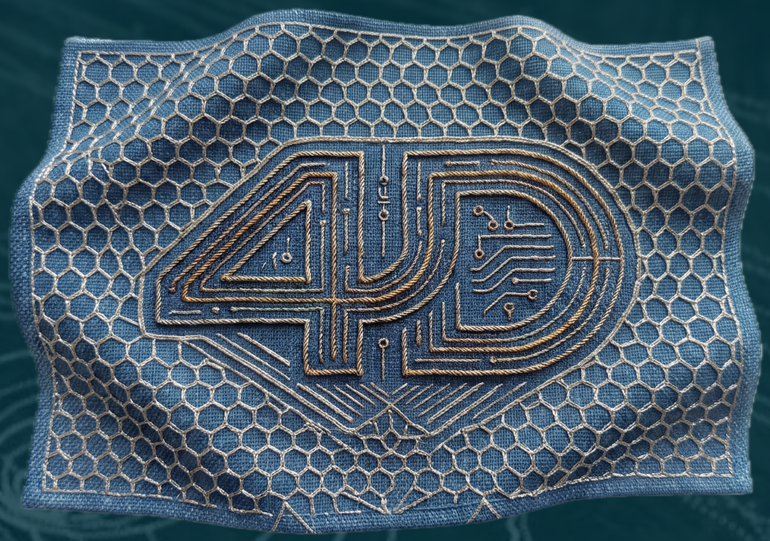
CMOS + X

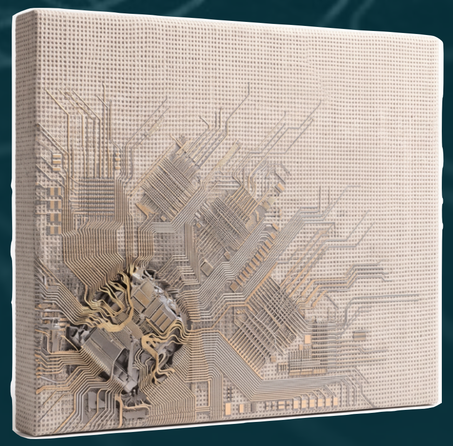
- **Goal**: develop Pixel detector readout chip technology suitable for direct deposition of thin film sensors
- Awards 28nm shuttle run in *December 2026*
- 4mm by 4mm size
- Can receive full 12” wafers
- Enables: industrialization of process at wafer scale

The Compromise

4D Fabric - Chip “PixCanvas”

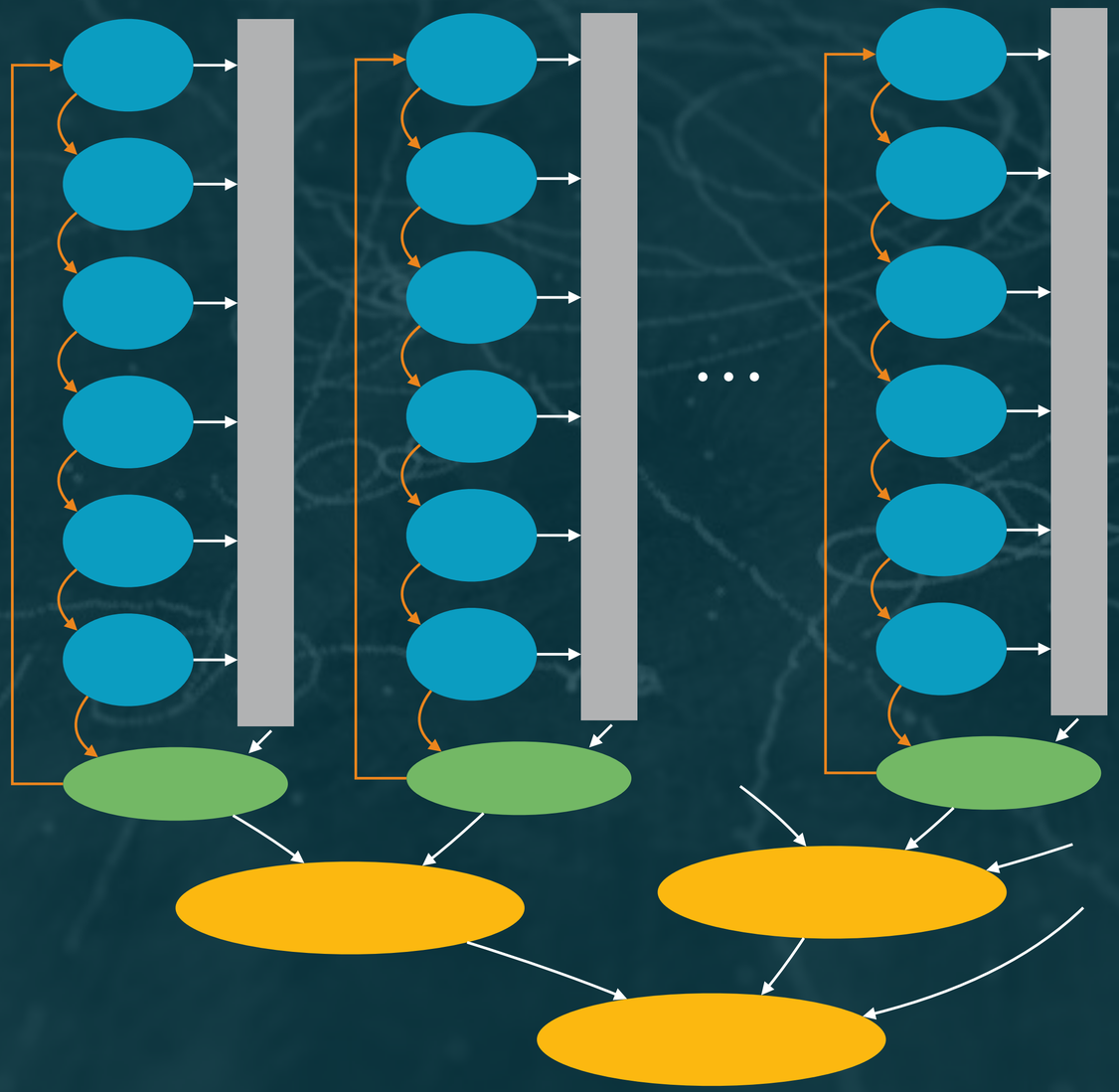
- **Goal**: develop a Pixel detector readout chip in 28nm CMOS for a December 2026 submission
- 4mm by 4mm chip size, 50um by 50um pixel pitch -> 64 x 64 pixel matrix
- No high-precision timing, no TDC -> 25ns time stamp but using high speed AFE
- Pixel Matrix organized in 8 x 8 pixel cores, cores organized in network mesh, cores flat synthesized as digital sea with analog islands
- Triggered readout with up to 40MHz trigger rate (streaming), ~12us buffering
- Digital-on-top design flow, verification with cocoTB, co-design against ATLAS ITK and MAIA (Muon collider) simulation





- Removal of high precision timing removes major analog work not feasible before Dec 2026
- Digital-on-top design flow is foundation to all future developments
- Emphasis on scalable and modular digital architecture to serve future chip versions and increase design iteration speed

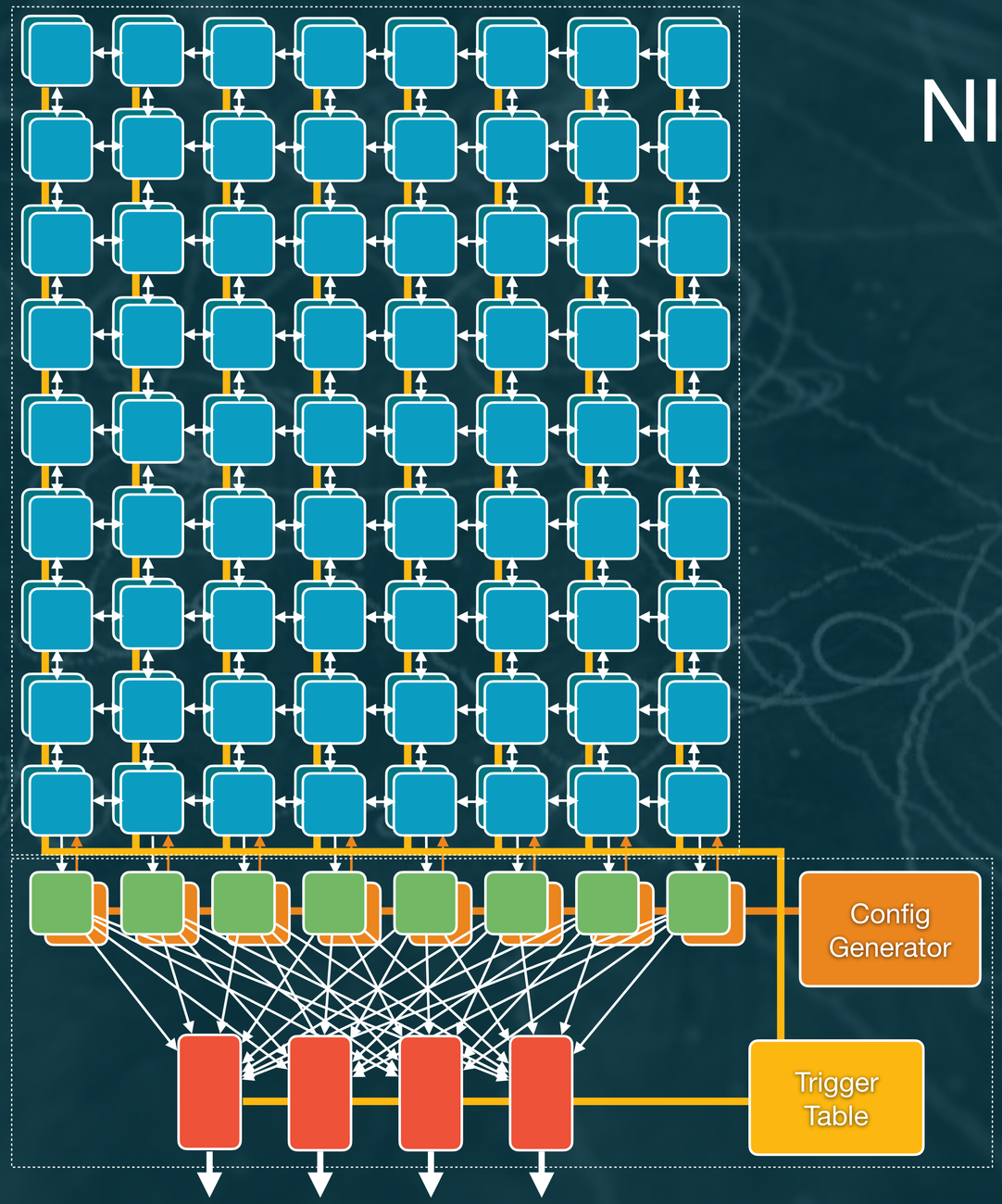
Traditional Readout Architecture



- Pixel Matrix organized in columns of Cores (8x8 pixels) (e.g ITkPix is 48 by 50 cores)
- Cores interconnected only within column (up/down)
- One shared column bus for data transmission to end-of-column buffer
- Token bus to arbitrate usage of column bus and return of token signals all hits from one event have been read
- One core column processes only one trigger at a time in chronological order, different core columns can work on different triggers
- 2-stage Core column concentrator reads events in order from end-of-column buffer -> event building

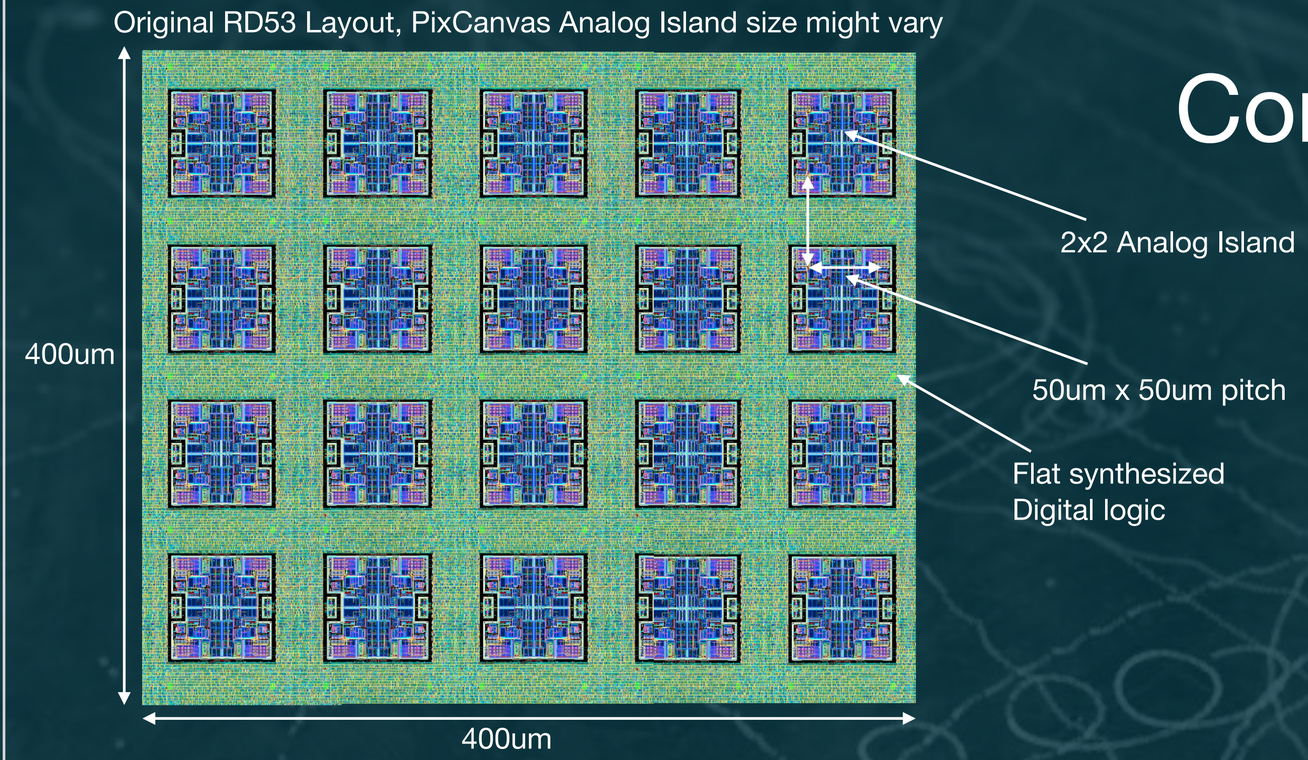
- **Pro**: highly optimized, low latency data transfer, built-in event building
- **Con**: single point failure can affect large number of pixels, no load balancing

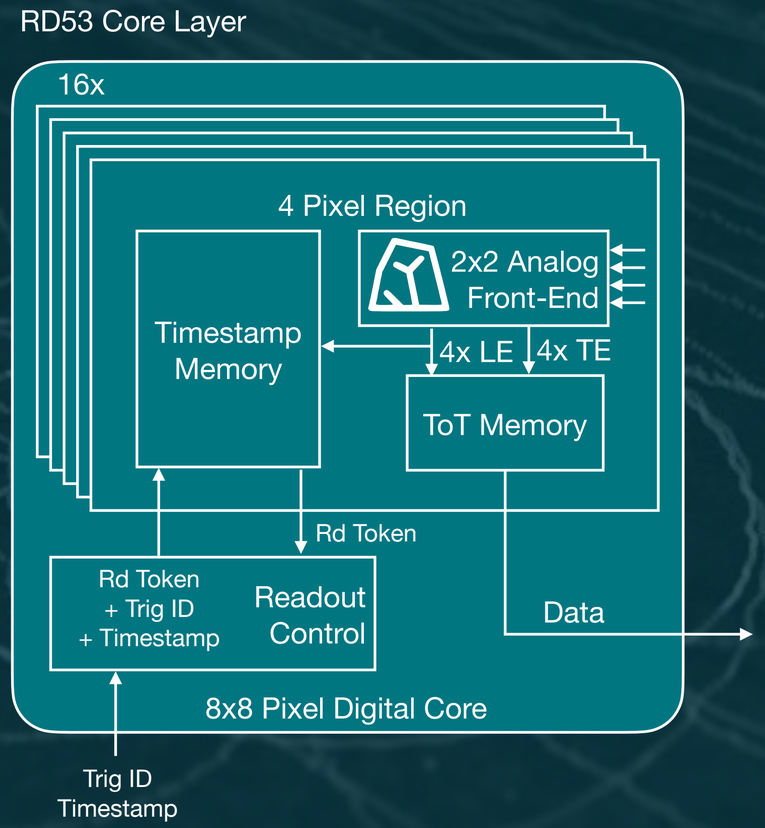
NIFTY Architecture



- Cores connected in 2D Mesh (shown here NSEW mesh)
- End-of-column concentrators become the “root” of the network, cores try to route data towards any root
- EOC concentrators are followed by a fully connected layer to events buffers
  - We are evaluating if need intermediate laters for routing or timing reasons
- Each event buffer collects data from one specific time-slice as informed by the trigger table
- Event buffers are each connected to their own data output
- Configuration is injected at the root level by the config generator

Core Block Diagram



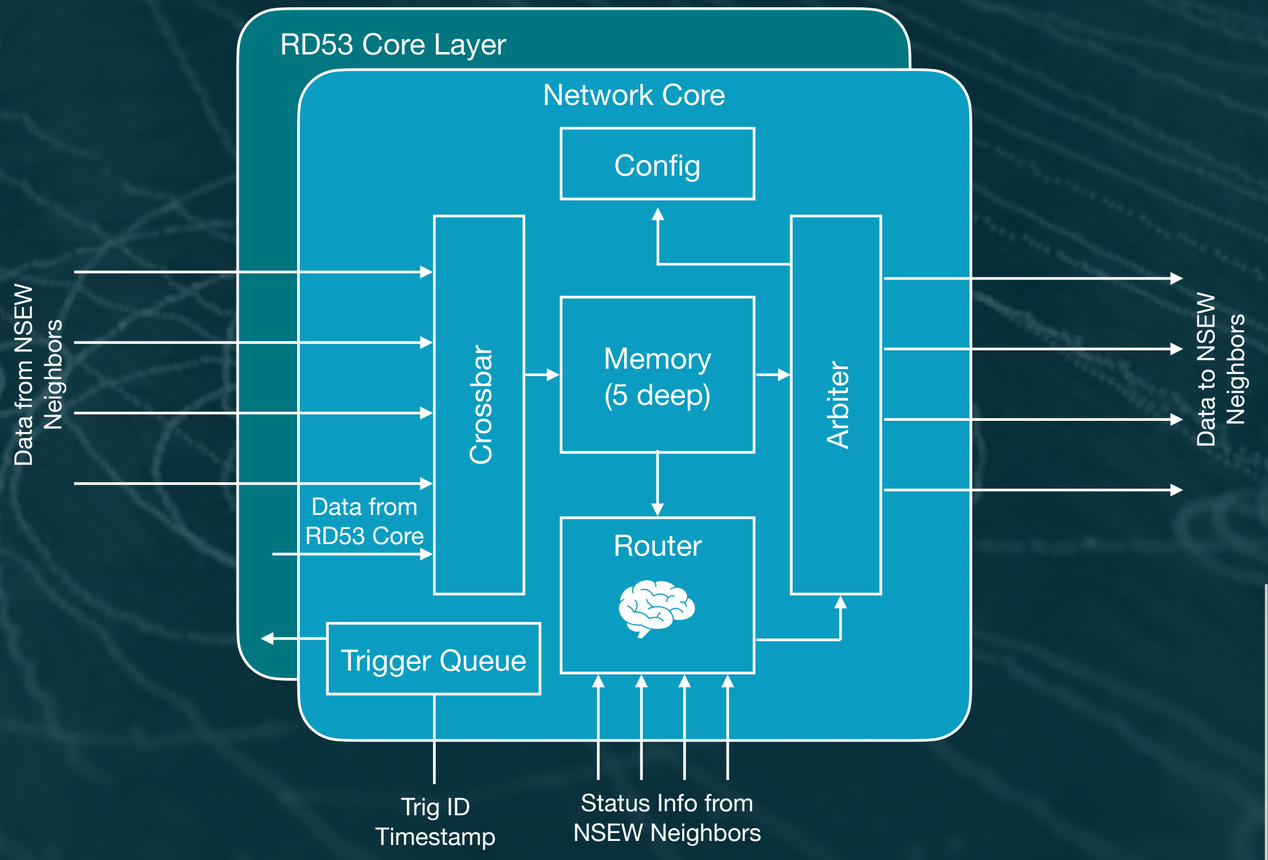


- Stripped digital Core from RD53 - ITkPixV2
- ToT of Hits is saved with associated timestamp
- Trigger of timestamp marks data to be readout
- 16 x 4 Pixel Regions read out through common data bus arbitrated through token

Analog Front-End design from Pebbles chip, see talk about it at CPAD 2023!

Core Block Diagram

- Network Core interfaces with NSEW Neighbors
- Can receive data on all edges at the same time
  - Underlying RD53 core acts as 5th input
- Send data only on one edge at a time
- Interface is bidirectional, S & W operate in positive edge domain, while N & E operate in negative edge domain
- Memory acts as FIFO
- Router contains intelligence to know where to route data, receives status info (e.g. buffer fill status, data acknowledge) of neighboring cores
  - Performs failure detection of neighbors
- Configuration also received via network
- Data contains some information related to routing to avoid loops
- Overall goal: make core as autonomous as possible, to avoid single core malfunction from impacting others



Toy Sim

Data Generation

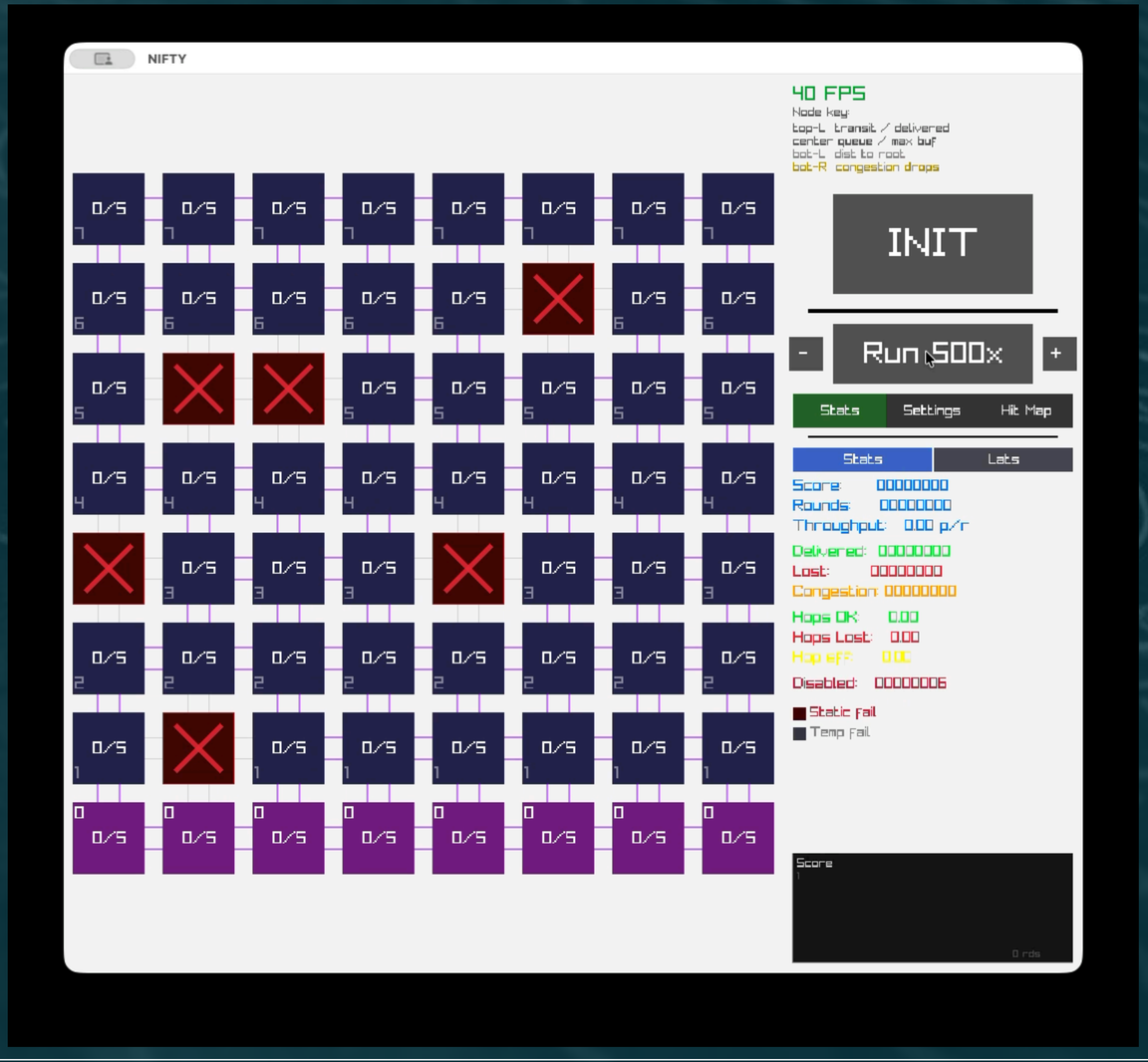
- Sources: random, ATLAS ITk simulation, MAIA simulation
- One data packet loaded into network layer per cycle (packet can hold up to 4 pixel hits)
- Emulating triggering by random sampling

Networking

- Each Core can transmit one data packet on one edge each cycle
- Each can receive from all incoming edges (including its own pixels) at the same time
- No back-pressure mechanism, data is simply dropped if network buffer overflows
- Each Core has finite memory

Routing

- Each Core has an individual router
- Router decides on which edge to transmit data for every cycle
- Router knows which edges have smaller, larger, or equal distance to the root nodes
- Router has information about buffer status from connected nodes



Framework also used for NIFTY-LArPix development! See talk by Rebecca and poster by Deion!

Failures

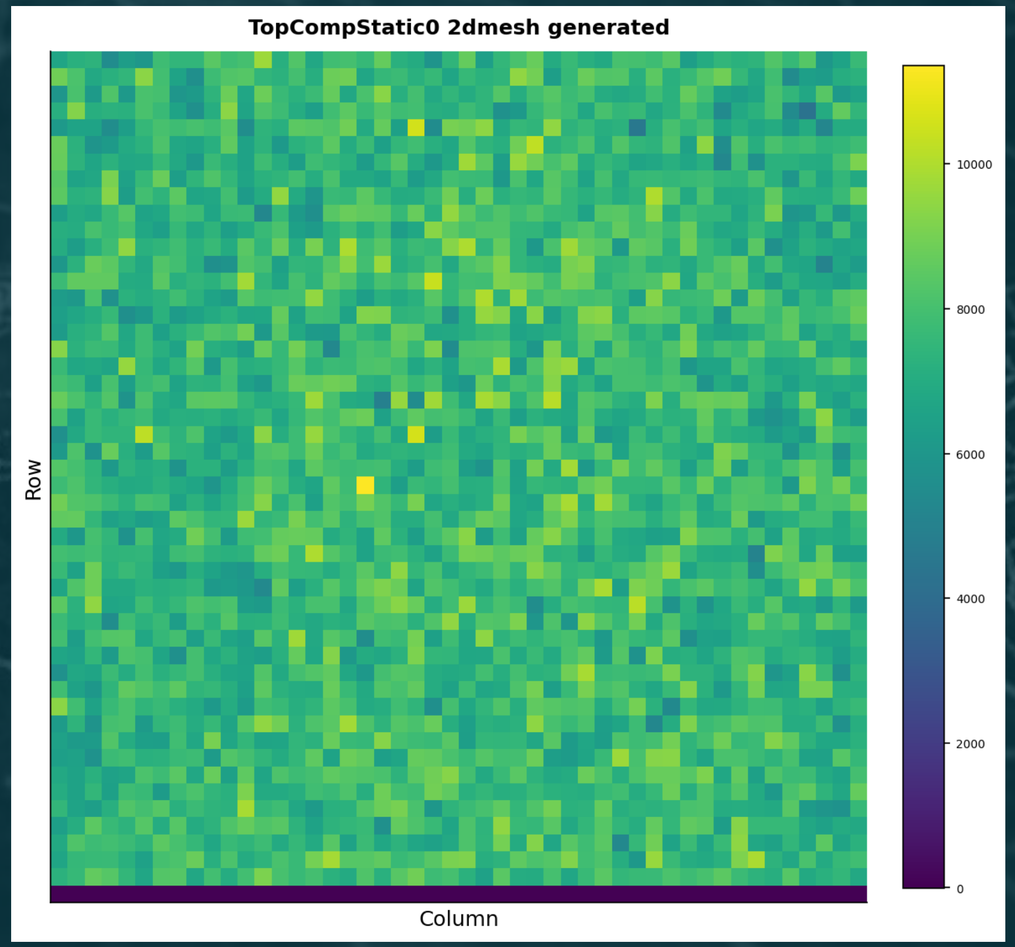
- Can simulate static failures and temporary failures
- Any data routed into a failed core will be dropped
- Data generated in failed core not counted

Optimization

- Simulation written in such a way that we can optimize each component with machine learning
- Currently though: fixed a lot of parameters to reduce phase space
- Some constraints from physical

Play with it yourself:
https://nifty.lbl.gov/www/simulation/

Heat Map: Data Generated



ATLAS ITk Layer 0 u=200

ATLAS Data

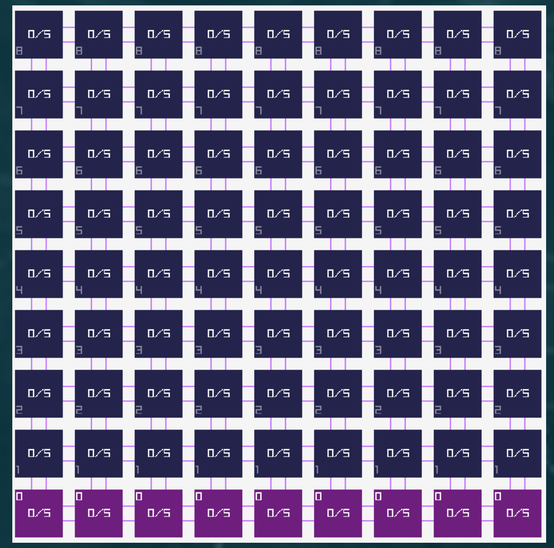
- No global structure, random
- Can contain some structure at pixel level (clustering)
- Only simulate “triggered” data, random trigger with configurable average frequency

Future Collider Data

- No global structure, random
- Can contain some structure at pixel level (clustering)
- Only simulate “triggered” data, random trigger with configurable average frequency

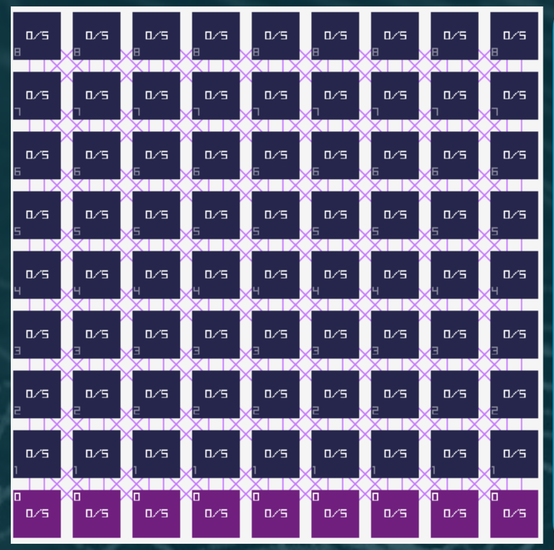
Topology Choice

BERKELEY LAB

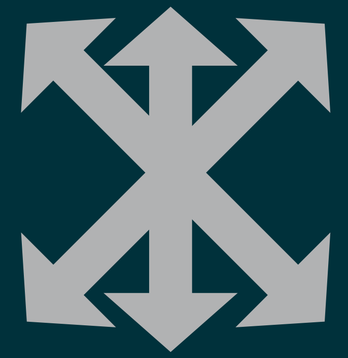


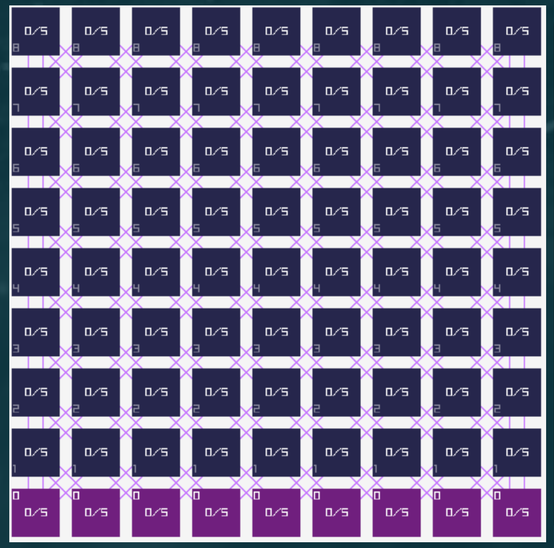
2D Mesh





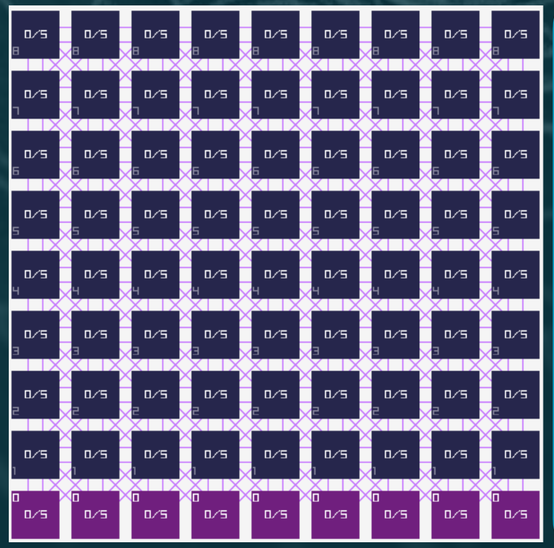
Hexagonal Mesh





Diagonal Mesh

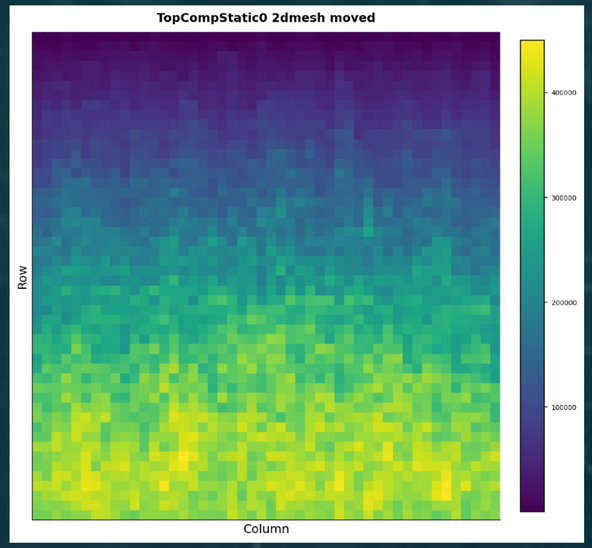




Octagonal Mesh

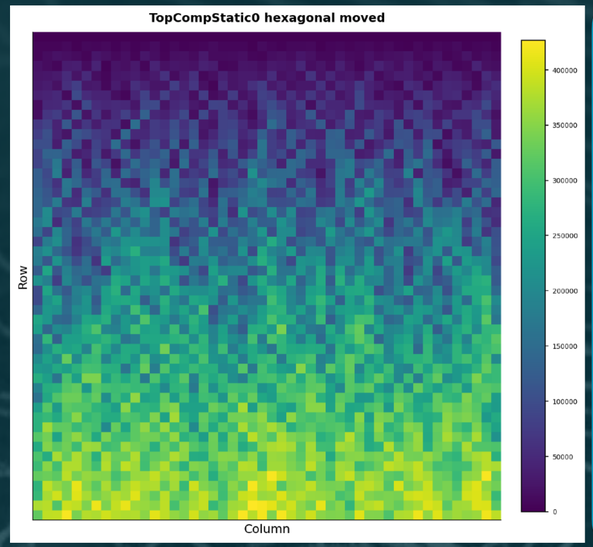


Heat Maps

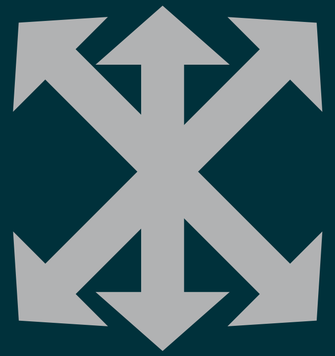


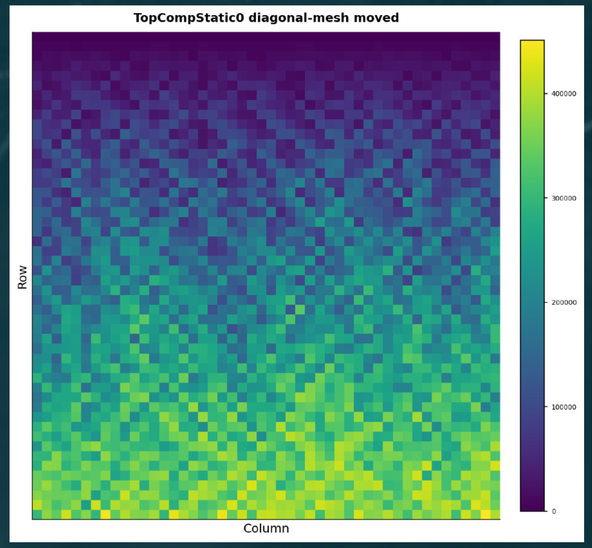
2D Mesh





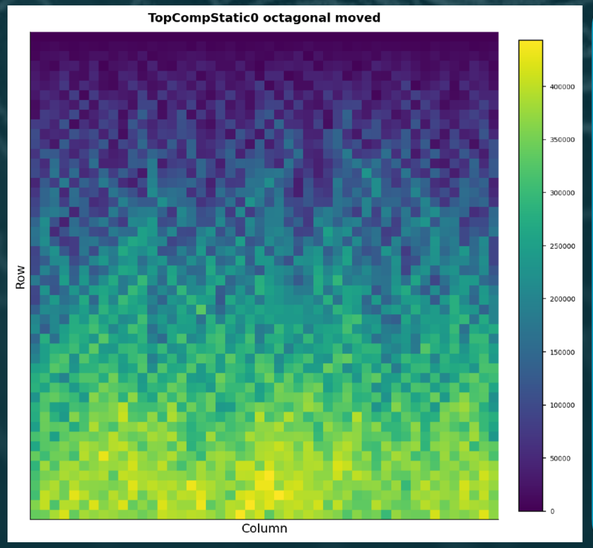
Hexagonal Mesh





Diagonal Mesh

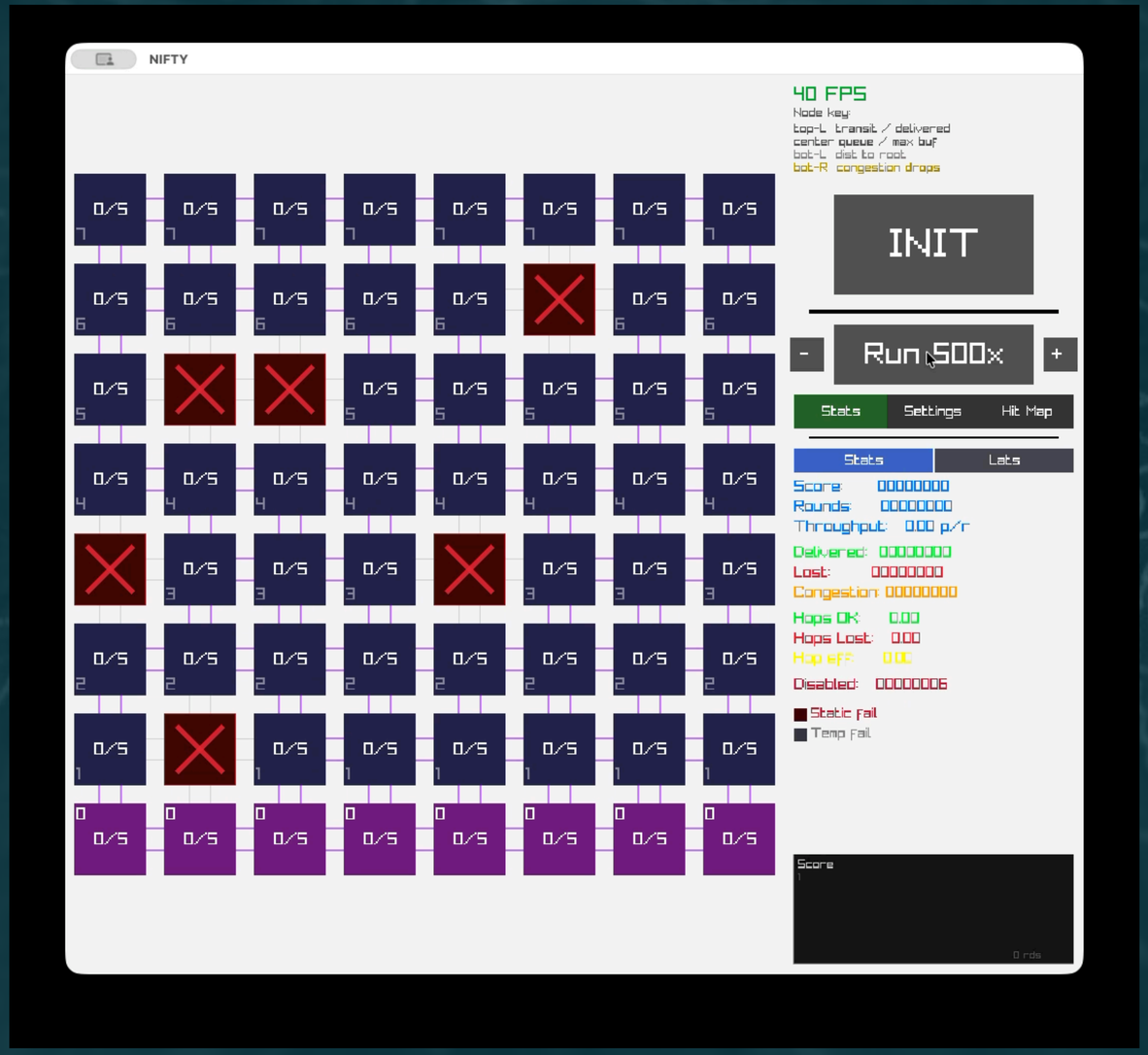


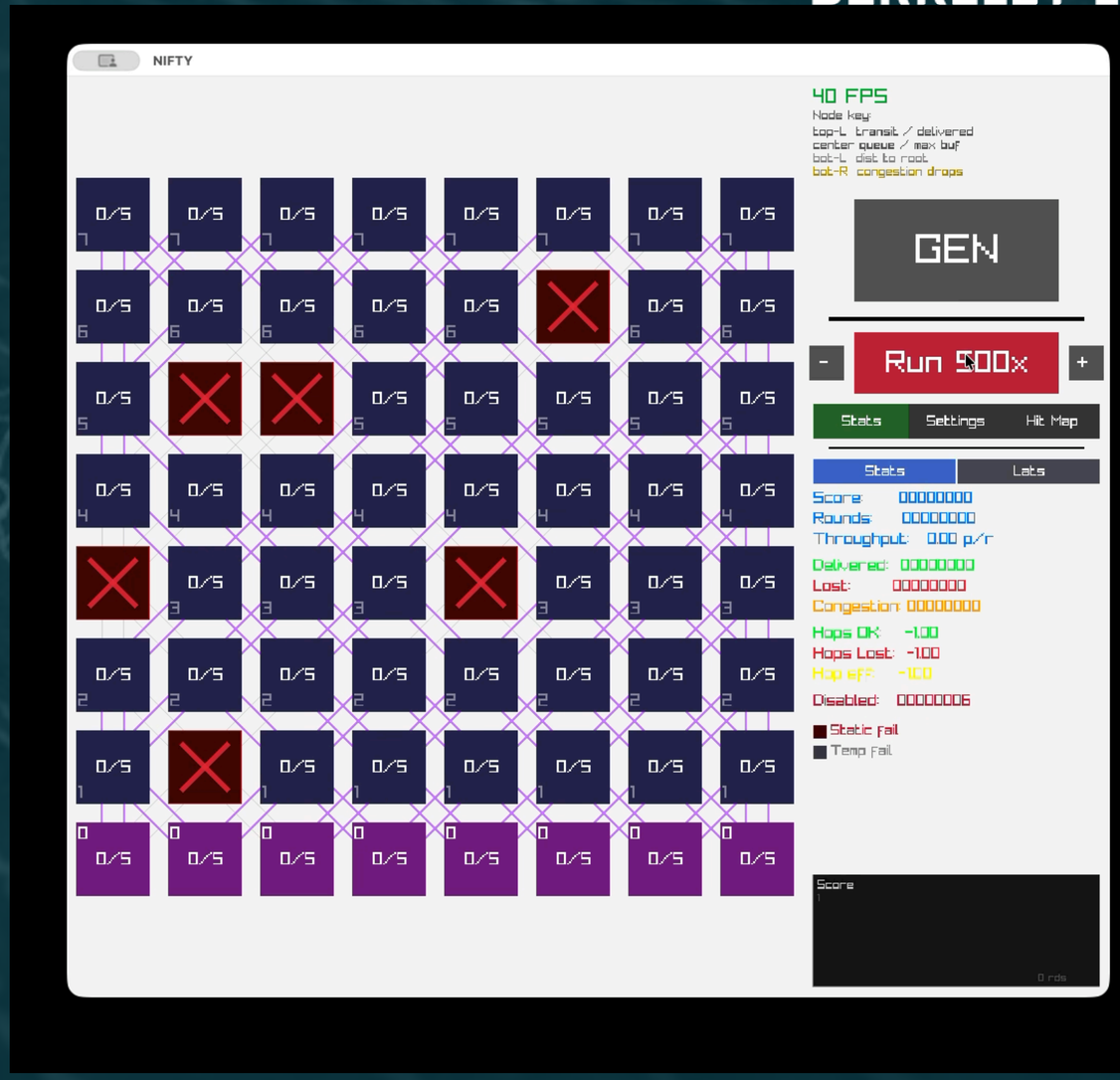


Octagonal Mesh



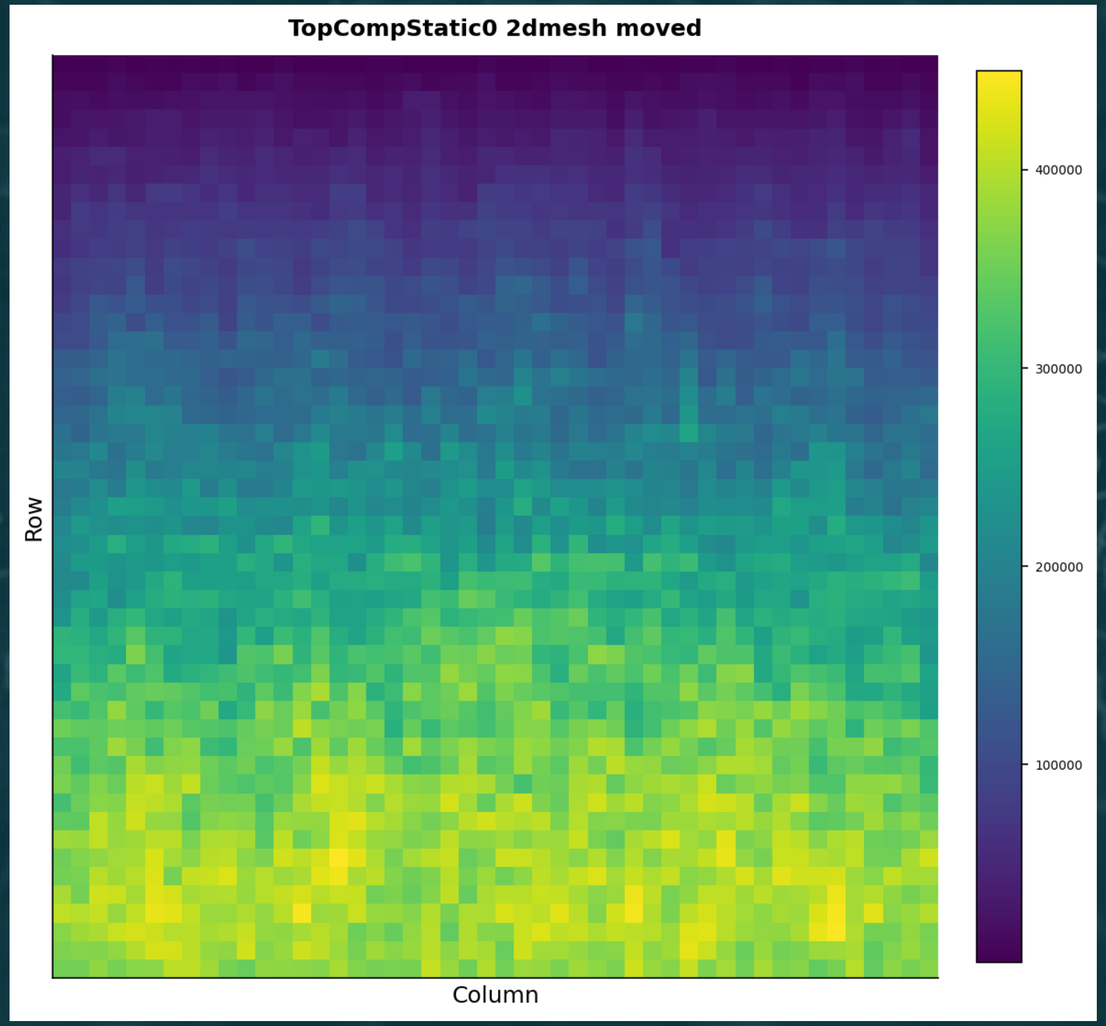
Topology Comparison

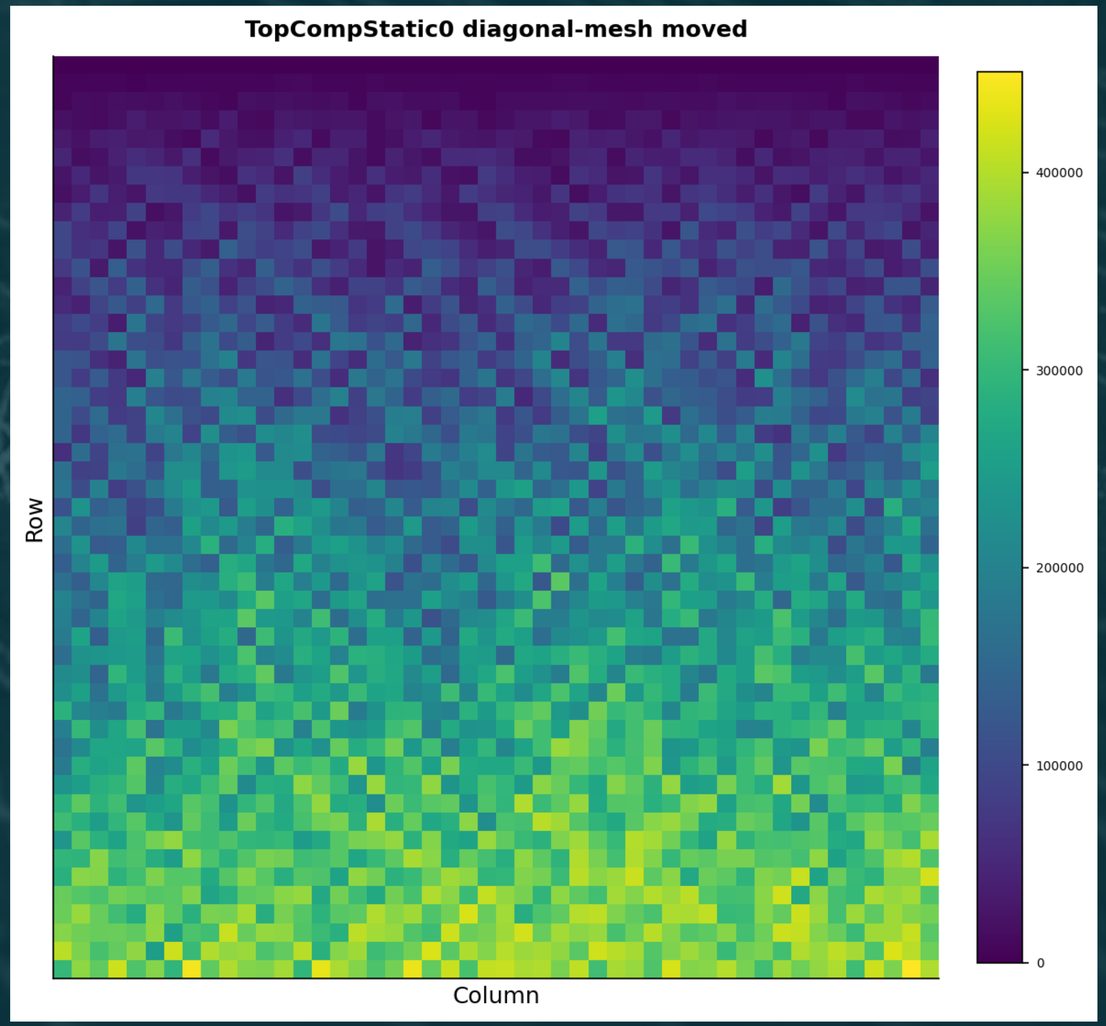




Heat Maps II

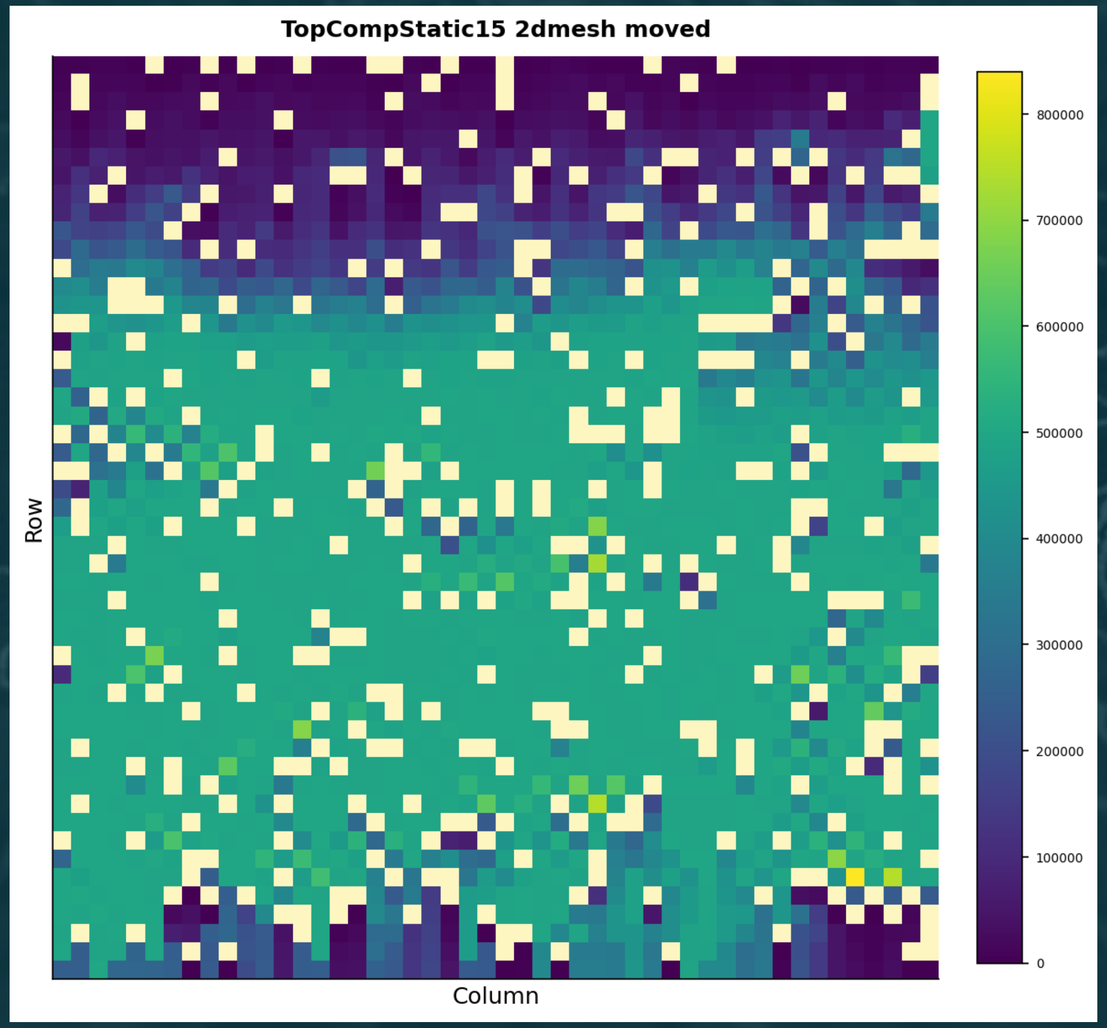
No static failures

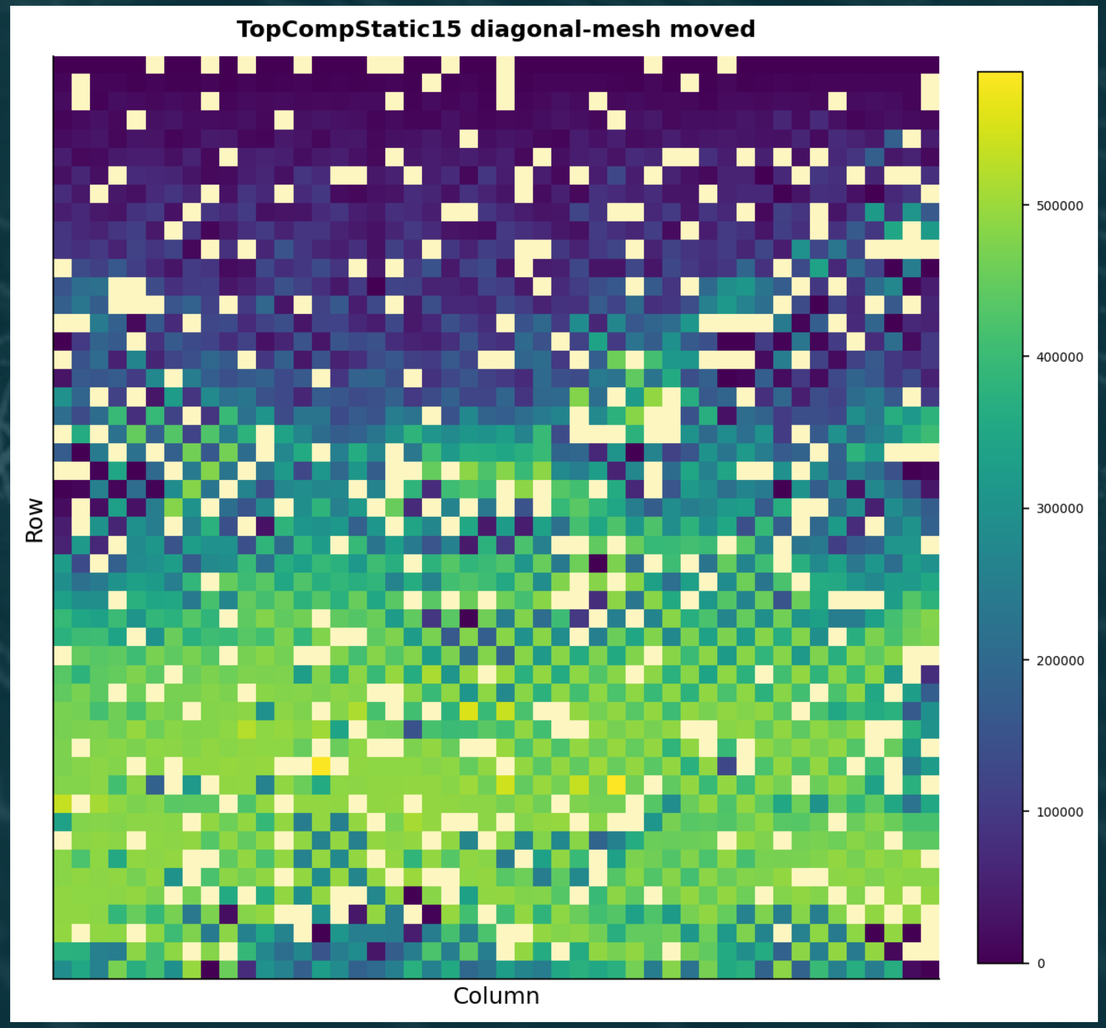




Heat Maps III

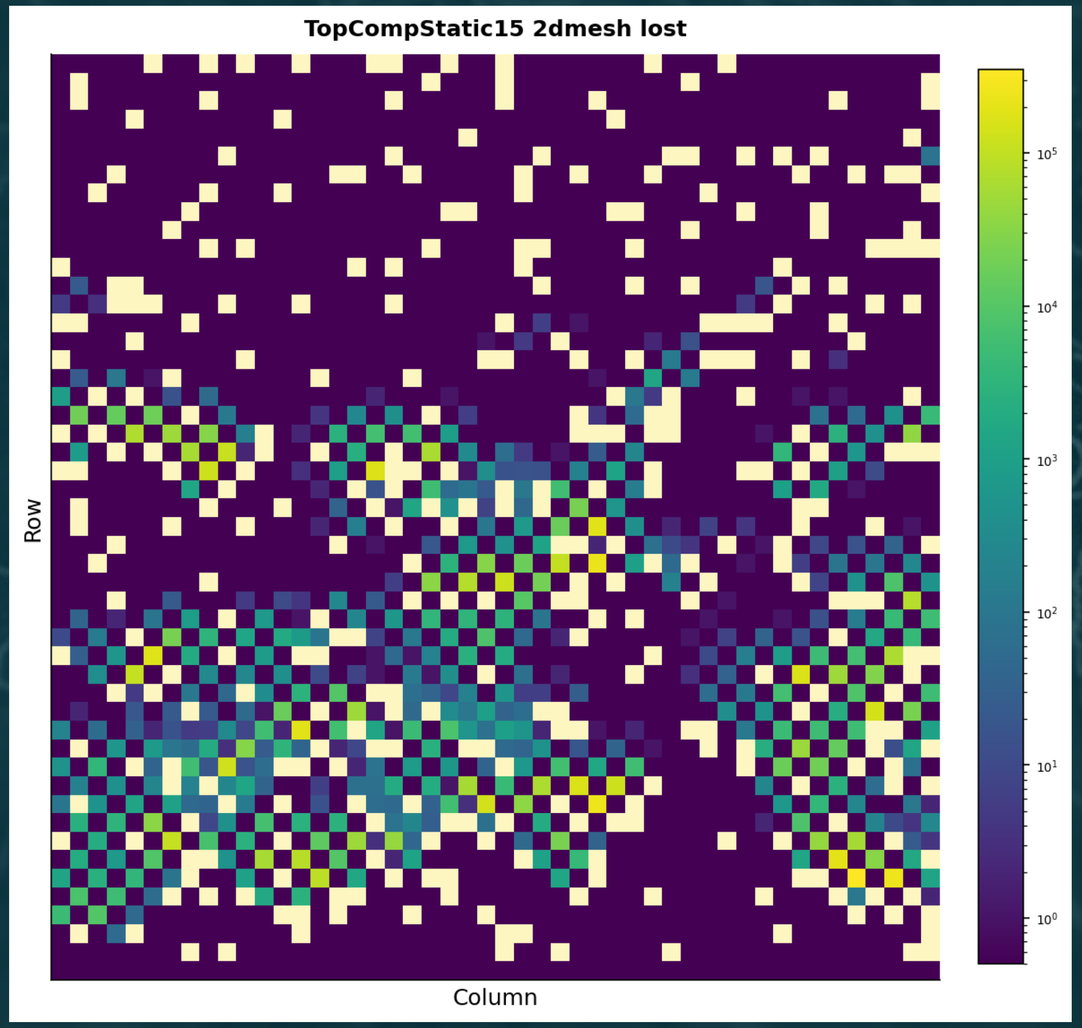
15% static failure (beige cores)

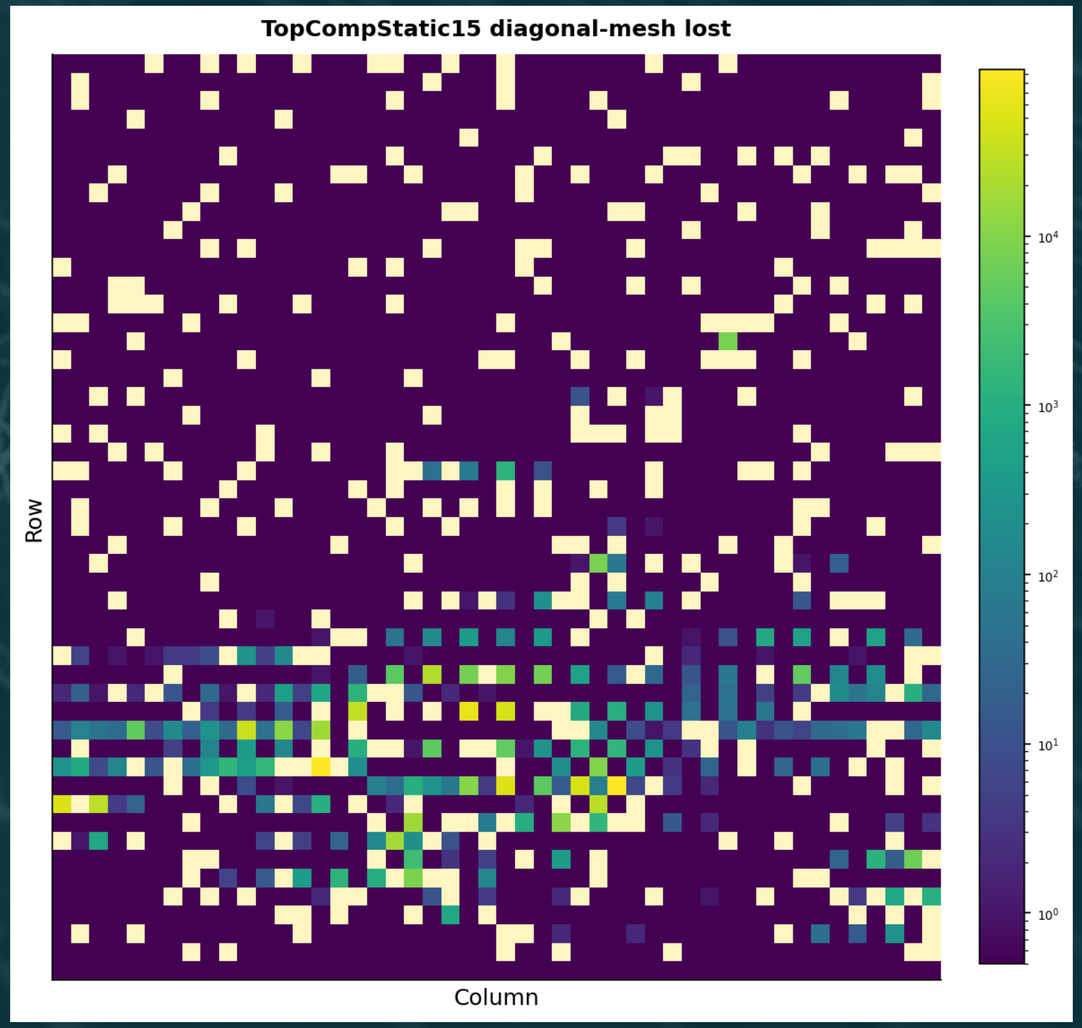




Heat Maps IV

15% static failure (beige cores)

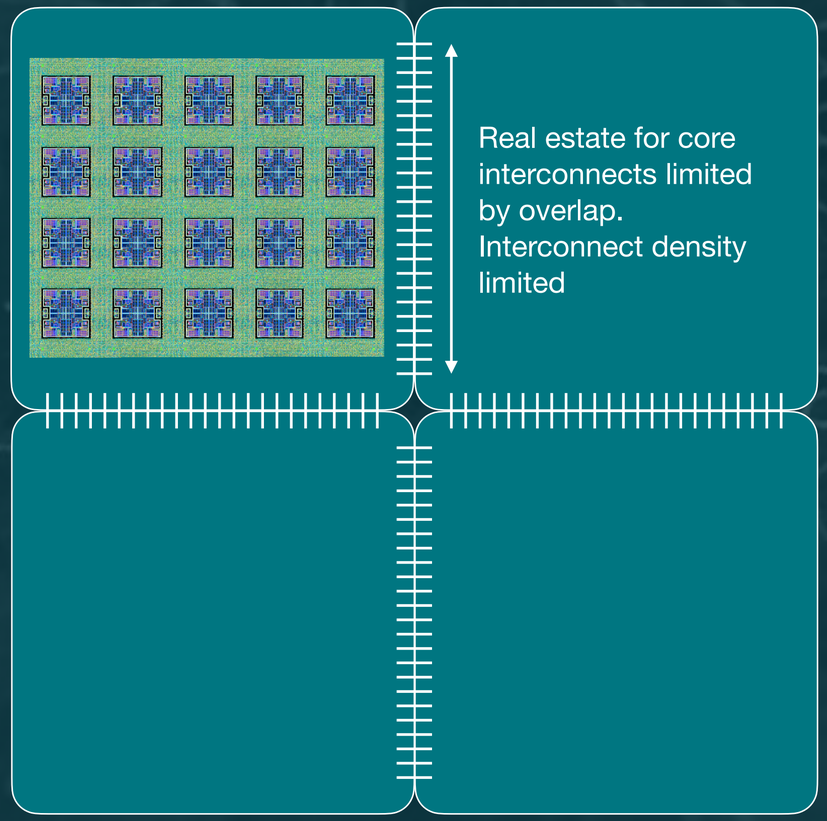




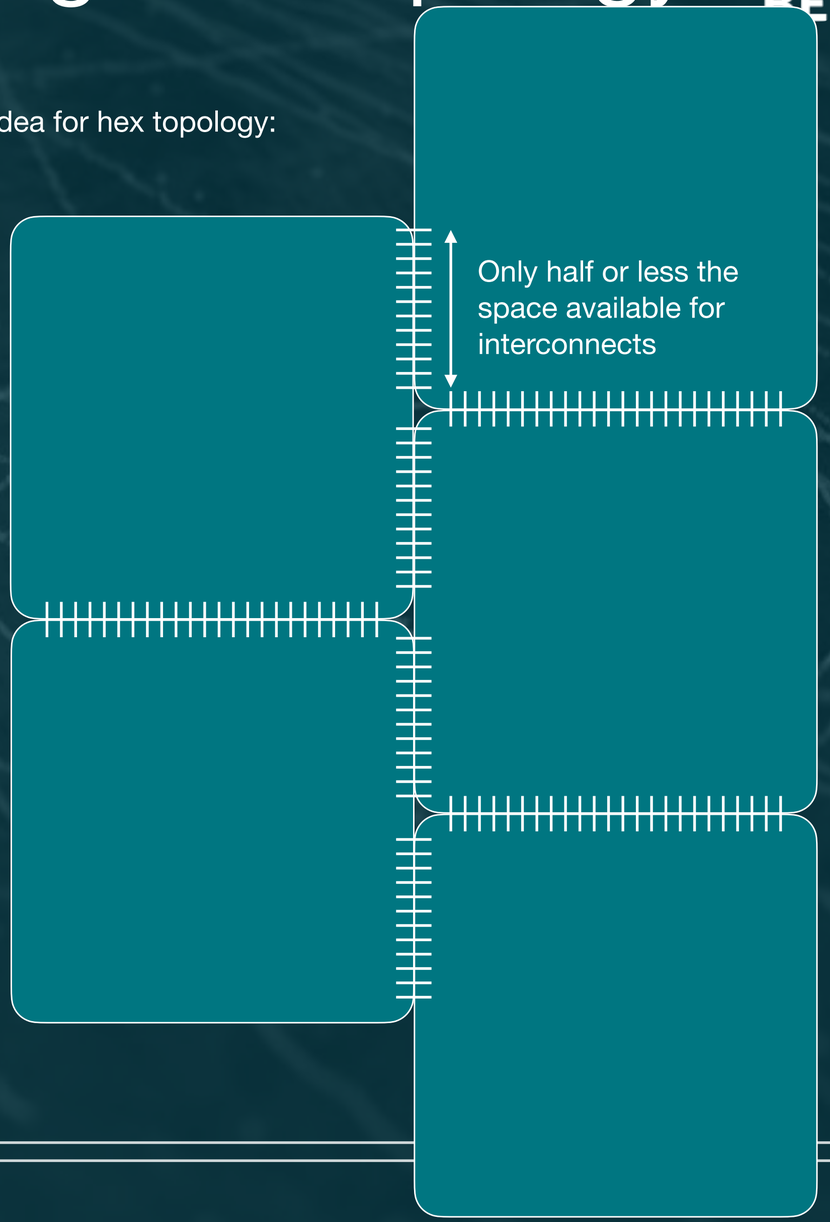
Core Tiling and Topology

Core needs to build around analog pixel island (2x2). Other than square geometry could be possible but might come at cost of routing density

2D Mesh topology:



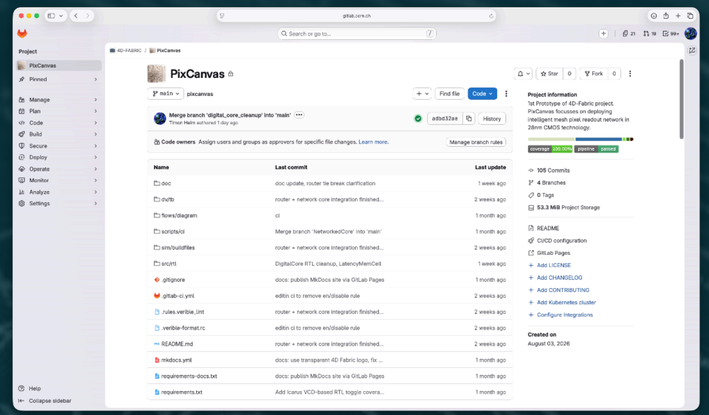
Idea for hex topology:



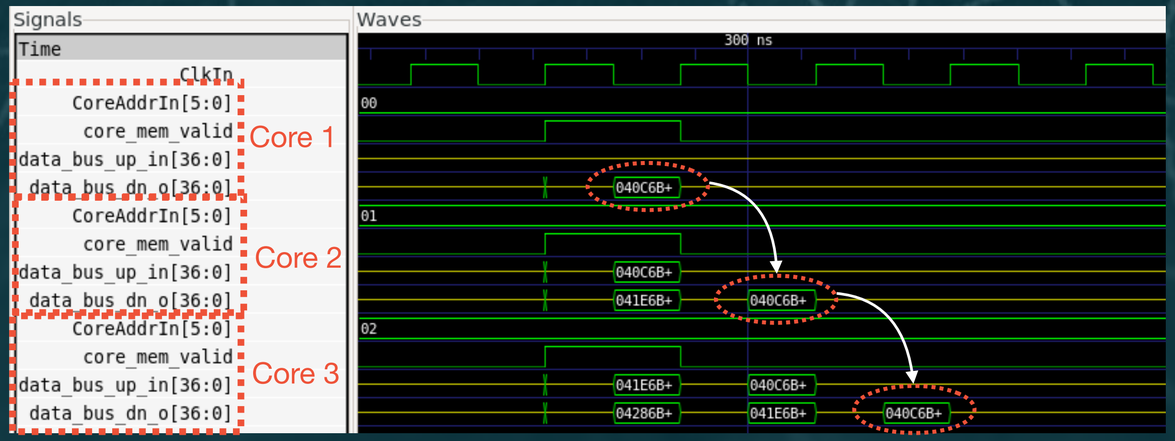
Core RTL Status

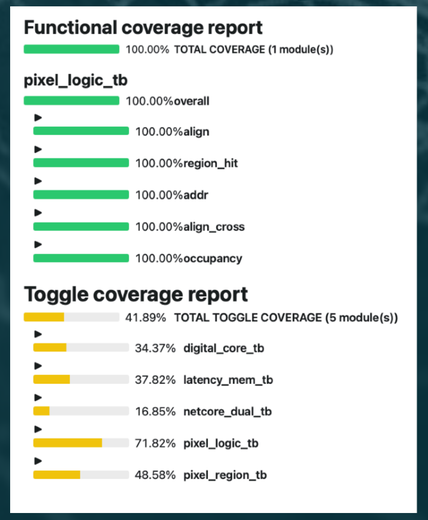
- Have rudimentary Networked Core written in RTL and interfaced with stripped RD53 core
- Writing verification at the same time as developing RTL
- Full verification run as part of CI, produces automatic coverage reports* and documents RTL hierarchy
- Some issues trying to accelerate simulation with e.g. ventilator due to aggressive clock gating to reduce power
- 3x3 Core simulation still runs fast enough to for CI (O(min)), while larger networks are feasible for milestones
- Current focus: transition from open source tooling (Icarus) to licensed tools (Questasim) to generate better coverage reports & integrate design in digital-on-top flow

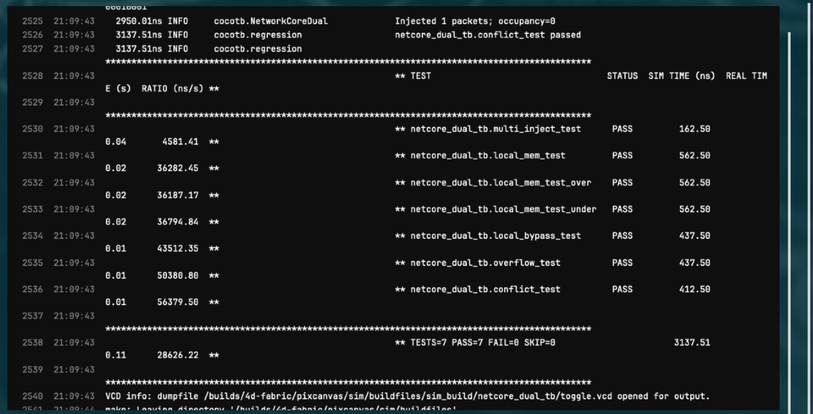
RTL GitLab Page:



Automatic generation of documentation, verification, and reports through continuous integration







End of Chip Logic - Event Building

Roots

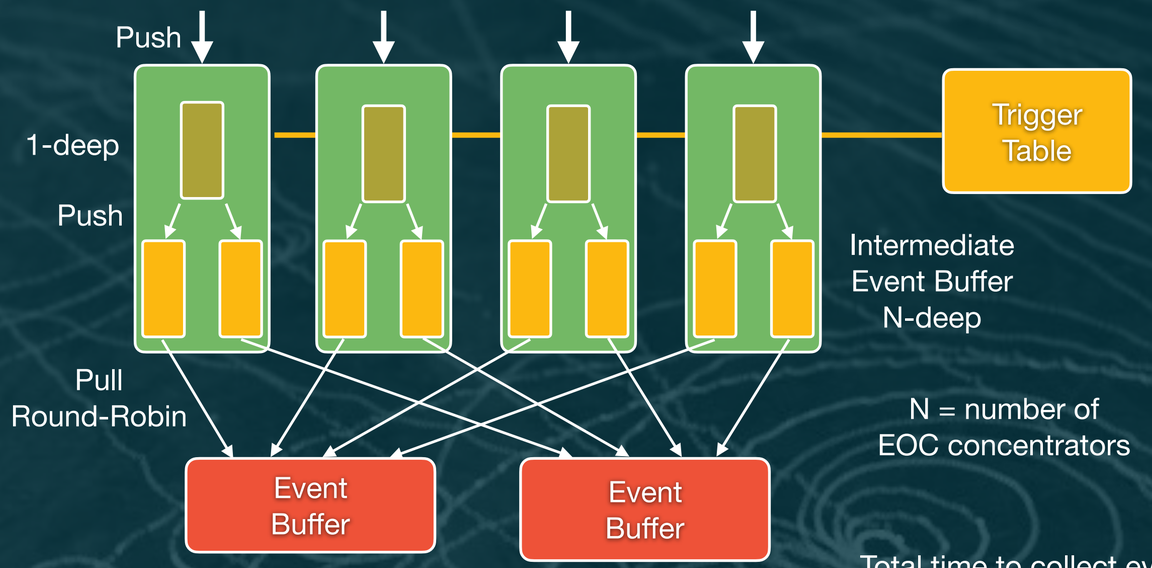
- One per core column
- Pushes data to intermediate layer depending on Trigger ID

Intermediate Layer

- Concentrates data from specific trigger ID
- Memory depth will depend on number of event buffers and Clk speed
- Could have multiple layers if intermediate buffers to overcome routing/timing constraints

Event Buffer

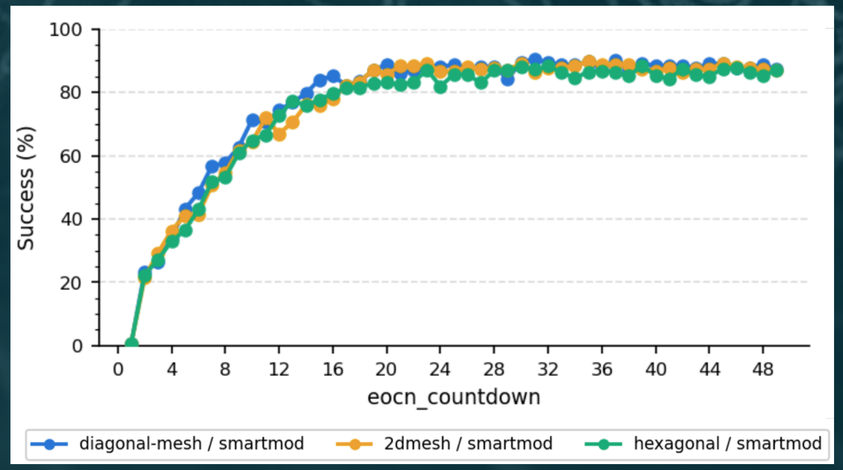
- At time of trigger is assigned an event ID (specific trigger time)
- Only collects data from that specific event ID
- Event is considered complete, after no data has been received for X number of cycles
- Data for events that cannot be assigned an event buffer will be dropped



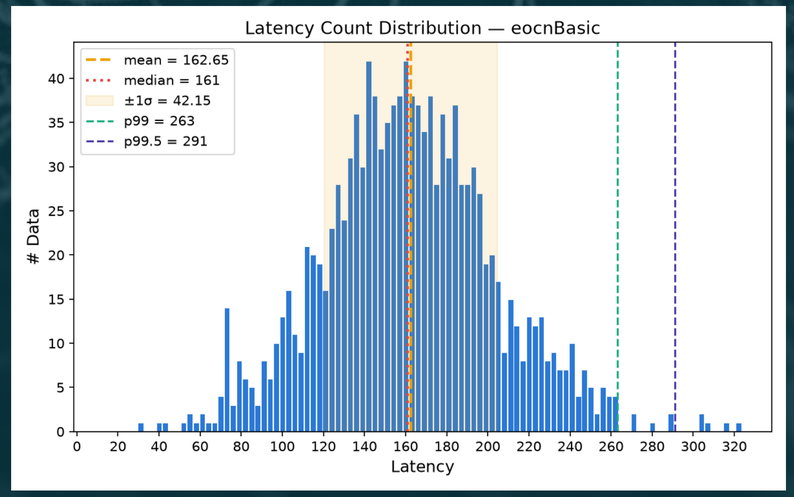
Exact number of concentrators, intermediate layers, and event buffers depends on expected hit rate and wanted trigger rate.

RTL should be parametric to adapt/scale easily!

Time to wait after last data fragment has been received:



Total time to collect event:



Summary & Plans

- **4D Fabric** project aims to ultimately develop fully fletches 4D Pixel detector technology
- Intermediate stepping stone: **PixCanvas** chip to build scalable digital architecture for pixel readout chips → the “fabric”
- Toy simulation framework helped guiding design choices or clearly points towards potential future improvements
- RTL and verification are progressing well → Now have enough meat to tackle digital-on-top design flow
- Analog designs to be leveraged from past 28nm submission
  - Recently started working on adding new analog elements (Bias DACs) with University of Hawaii
- December 2026 submission will be tight, but does not seem completely unreasonable … stay tuned
- Next iteration will aim at adding high precision timing into the design → add the “4D” to the “fabric”
  - Will need to tackle some major challenges, but network topology also opens new opportunities

Special shoutout to Graham and Caleb who did most of this work during a SULI internship and will continue as undergraduate researchers:



Graham Anderson
Undergraduate Student
UC Berkeley · Electrical Engineering

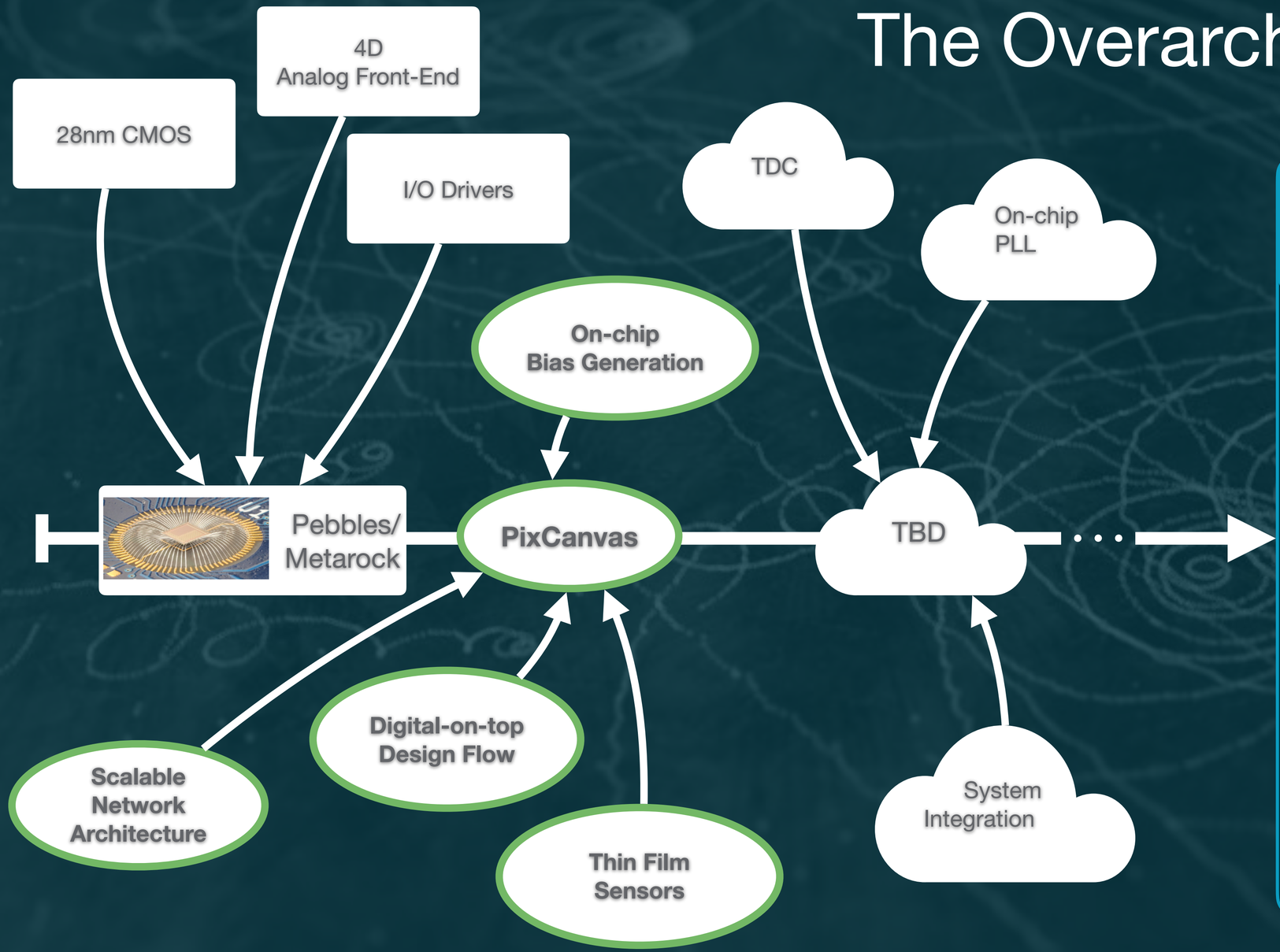


Caleb Ceravolo
Undergraduate Student
University of Washington · Electrical Engineering

Poster by Caleb

Backup

The Overarching Goal



4D Pixel Detectors for next-gen Colliders

- High temporal resolution (<30ps)
- High spatial resolution (<25um)
- 99% detection efficiency
- Low Power/Low Mass
- Extreme readout bandwidth
- On-chip data processing
- Extreme radiation tolerance
- Wafer-scale detectors



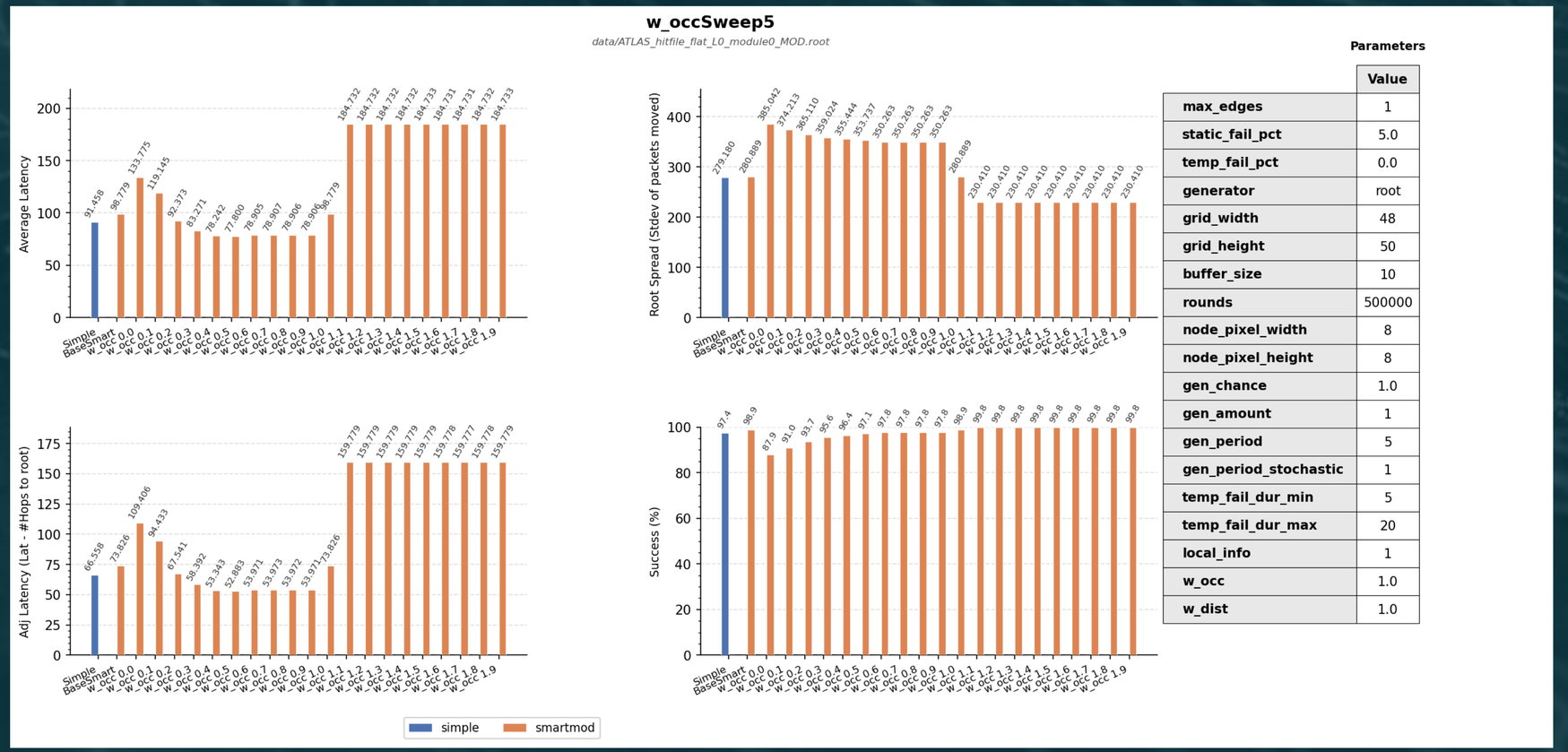
Router Optimization

Simple Router

- Ranks edges by score, picks the lowest score
- Score = distance to root + buffer status

Smart Router

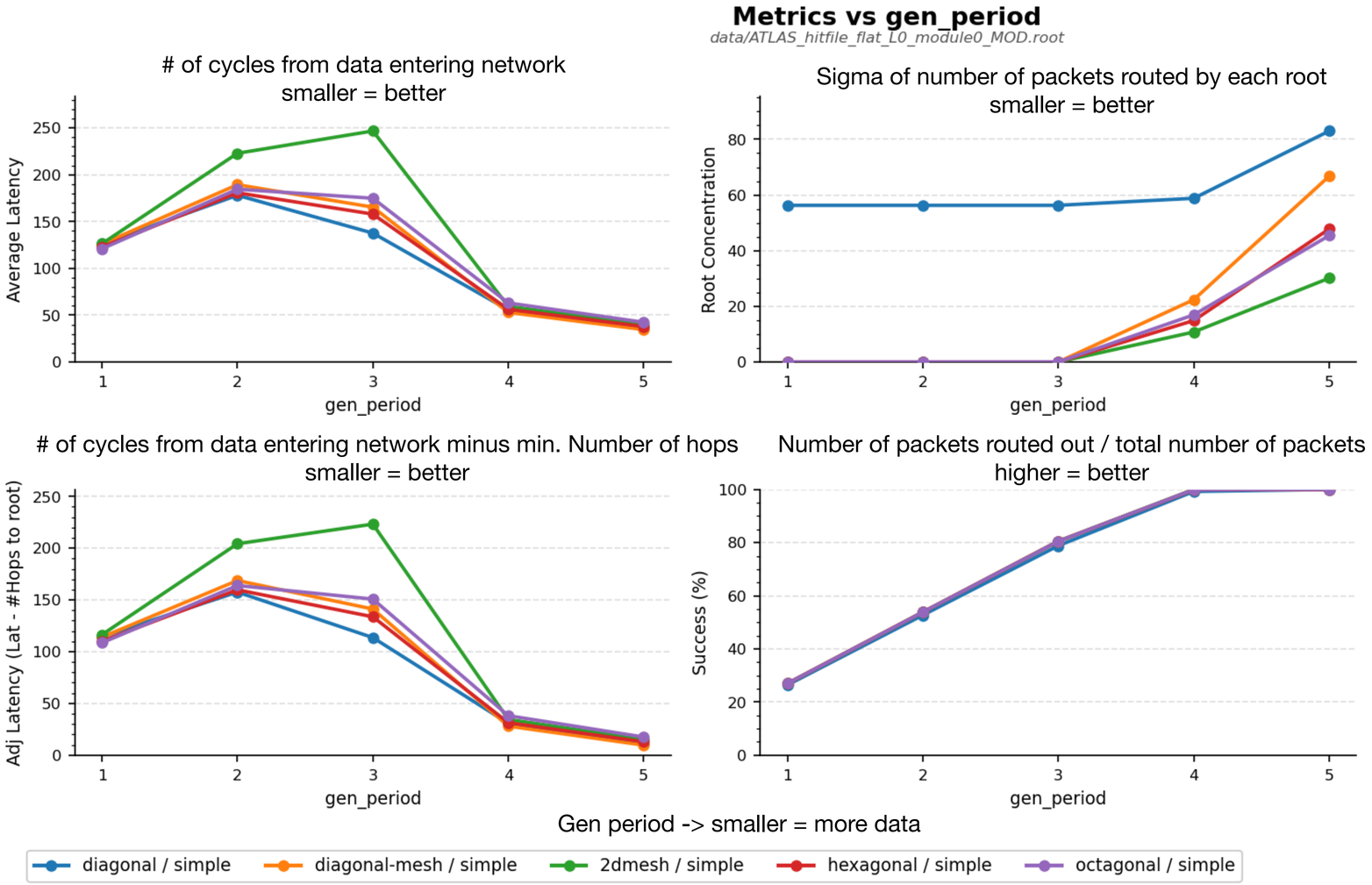
- Ranks edges by score, picks the lowest score
- Score = distance to root + W * buffer status
- Weight W to be optimized



Parameters

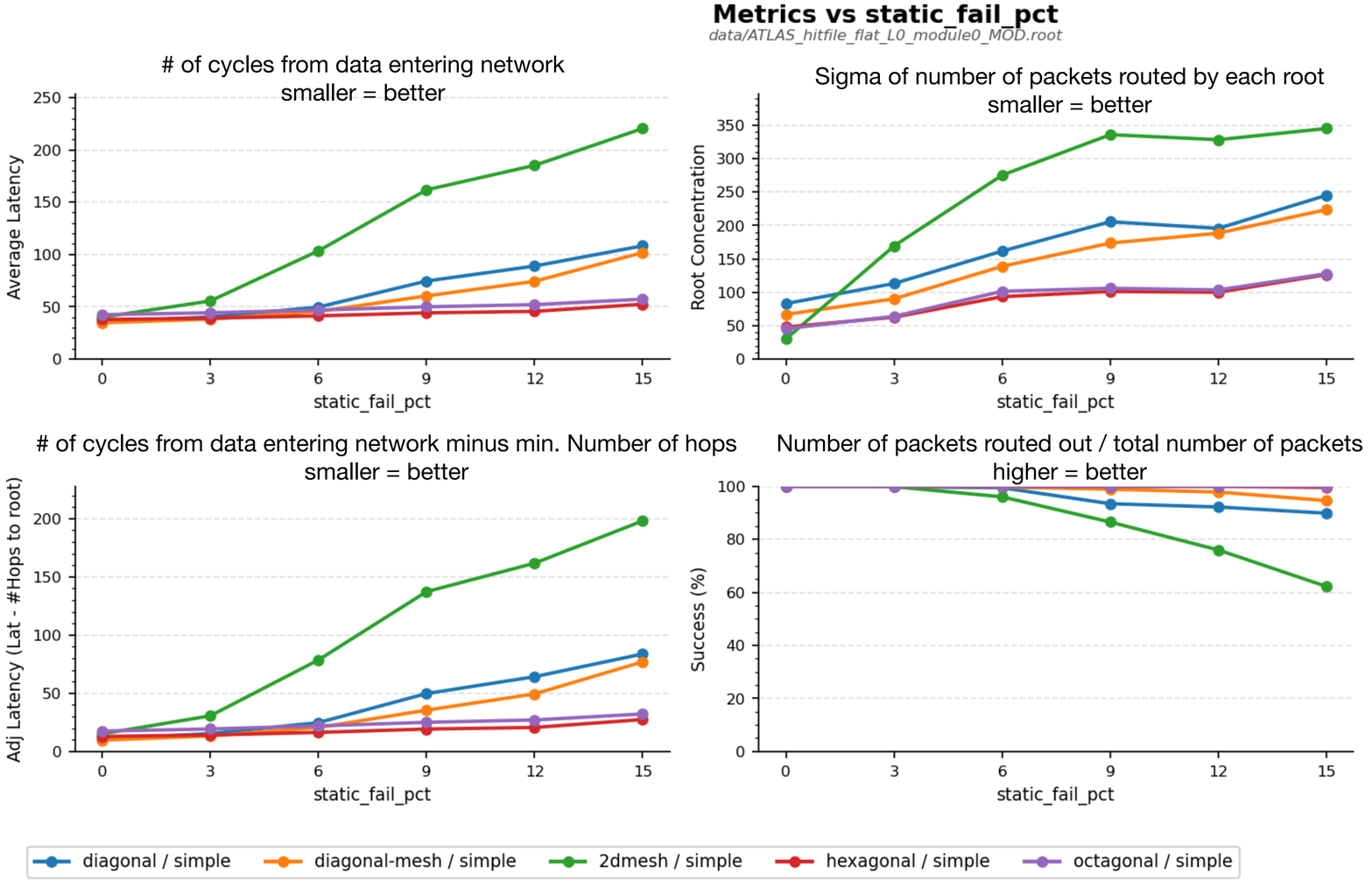
	Value
max_edges	1
static_fail_pct	5.0
temp_fail_pct	0.0
generator	root
grid_width	48
grid_height	50
buffer_size	10
rounds	500000
node_pixel_width	8
node_pixel_height	8
gen_chance	1.0
gen_amount	1
gen_period	5
gen_period_stochastic	1
temp_fail_dur_min	5
temp_fail_dur_max	20
local_info	1
w_occ	1.0
w_dist	1.0

Topology Comparison II



Parameter	Value
max_edges	1
static_fail_pct	0.0
temp_fail_pct	0.0
grid_width	48
grid_height	50
buffer_size	10
rounds	500000
node_pixel_width	8
node_pixel_height	8
gen_chance	1.0
gen_amount	1
gen_period_stoc	1
temp_fail_dur_min	5
temp_fail_dur_max	20
local_info	1
w_occ	1.0
w_dist	1.0

Topology Comparison III



Parameter	Value
max_edges	1
temp_fail_pct	0.0
grid_width	48
grid_height	50
buffer_size	10
rounds	500000
node_pixel_width	8
node_pixel_height	8
gen_chance	1.0
gen_amount	1
gen_period	5
gen_period_stoc	1
temp_fail_dur_min	5
temp_fail_dur_max	20
local_info	1
w_occ	1.0
w_dist	1.0

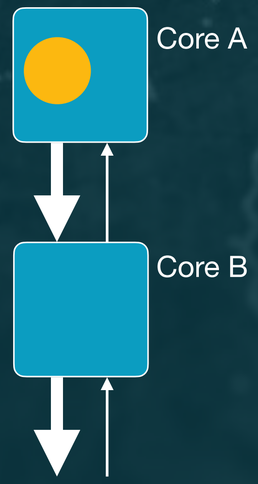
Failure Detection

BERKELEY LAB

How to identify dynamic failures?

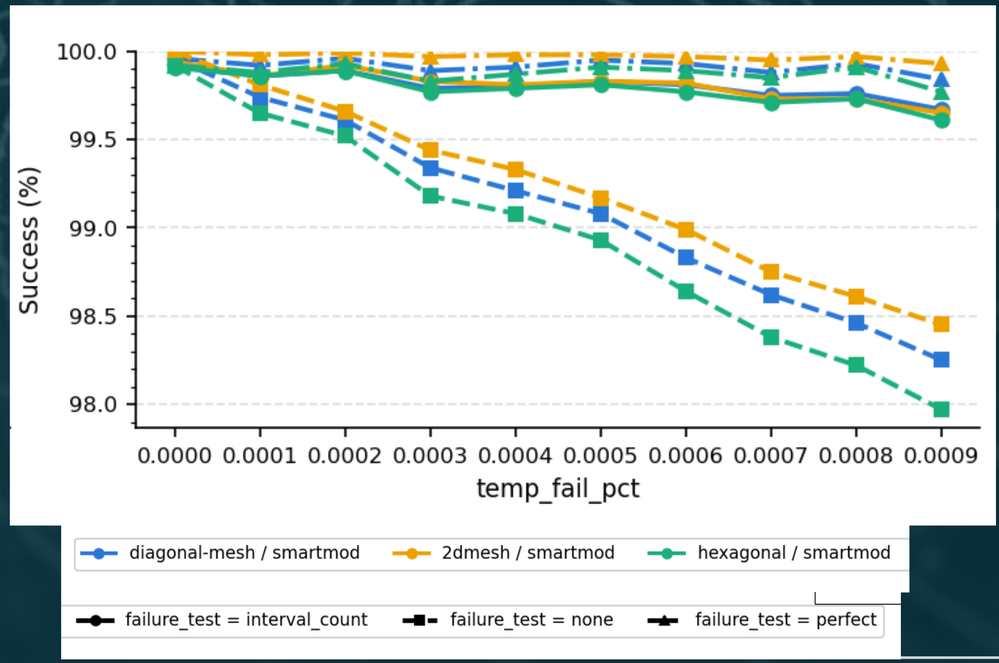
- To reduce logic/power use data to test neighboring cores
- Means protection cannot be perfect, but identification circuit becomes trivial
- Core B “acknowledge” transmission of data to the original source of the data Core A, if Core A does not receive acknowledgment -> disables transmission to Core B for some period of time
- Tests for multiple failures at the same time:
  - Data from Core A does not arrive in Core B
  - Core B does not save data correctly
  - Core B is “noisy” and flooded with other data
  - Core B cannot send data onwards

1. Data is generated in Core A
2. Core A routes Data to Core B
3. Core A starts countdown timer, Core B saves data
4. Core B routes data to next core & sends acknowledge to Core A
5. Core A receives acknowledgment in time and edge to Core B stays enables



Source of temporary failures?

- Single Event Effects (SEE) can upset logic and cause abnormal behavior until reset
- Physical damage to chip can manifest in temporary failures lasting from single event to failed until reset



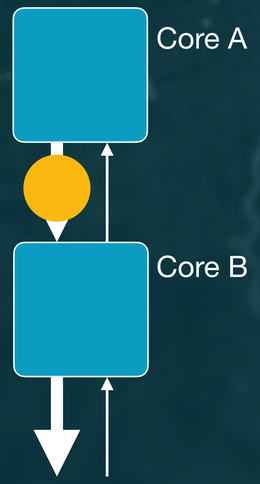
Failure Detection

BERKELEY LAB

How to identify dynamic failures?

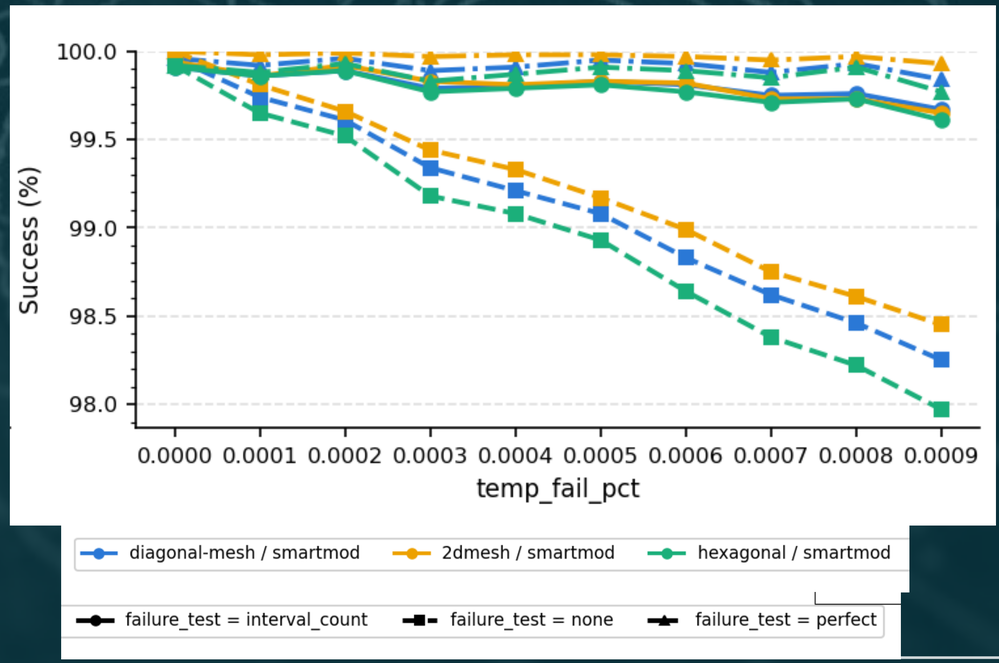
- To reduce logic/power use data to test neighboring cores
- Means protection cannot be perfect, but identification circuit becomes trivial
- Core B “acknowledge” transmission of data to the original source of the data Core A, if Core A does not receive acknowledgment -> disables transmission to Core B for some period of time
- Tests for multiple failures at the same time:
  - Data from Core A does not arrive in Core B
  - Core B does not save data correctly
  - Core B is “noisy” and flooded with other data
  - Core B cannot send data onwards

1. Data is generated in Core A
2. Core A routes Data to Core B
3. Core A starts countdown timer, Core B saves data
4. Core B routes data to next core & sends acknowledge to Core A
5. Core A receives acknowledgment in time and edge to Core B stays enables



Source of temporary failures?

- Single Event Effects (SEE) can upset logic and cause abnormal behavior until reset
- Physical damage to chip can manifest in temporary failures lasting from single event to failed until reset



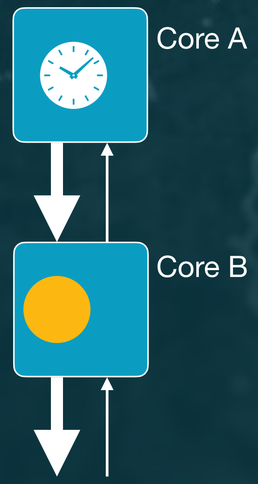
Failure Detection

BERKELEY LAB

How to identify dynamic failures?

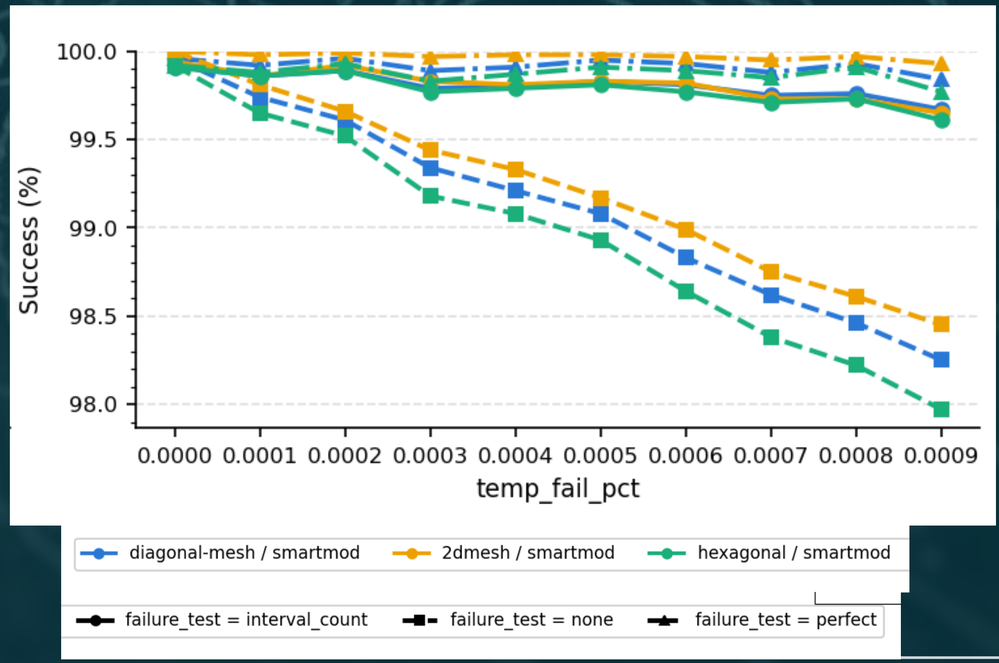
- To reduce logic/power use data to test neighboring cores
- Means protection cannot be perfect, but identification circuit becomes trivial
- Core B “acknowledge” transmission of data to the original source of the data Core A, if Core A does not receive acknowledgment -> disables transmission to Core B for some period of time
- Tests for multiple failures at the same time:
  - Data from Core A does not arrive in Core B
  - Core B does not save data correctly
  - Core B is “noisy” and flooded with other data
  - Core B cannot send data onwards

1. Data is generated in Core A
2. Core A routes Data to Core B
3. Core A starts countdown timer, Core B saves data
4. Core B routes data to next core & sends acknowledge to Core A
5. Core A receives acknowledgment in time and edge to Core B stays enables



Source of temporary failures?

- Single Event Effects (SEE) can upset logic and cause abnormal behavior until reset
- Physical damage to chip can manifest in temporary failures lasting from single event to failed until reset



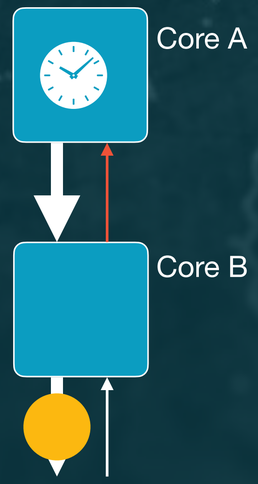
Failure Detection

BERKELEY LAB

How to identify dynamic failures?

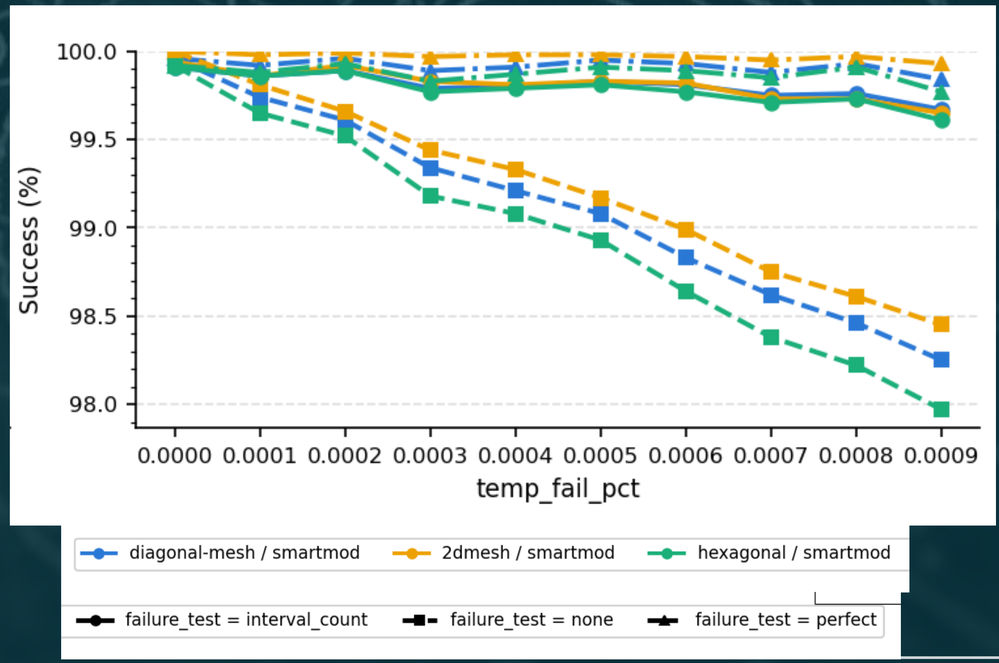
- To reduce logic/power use data to test neighboring cores
- Means protection cannot be perfect, but identification circuit becomes trivial
- Core B “acknowledge” transmission of data to the original source of the data Core A, if Core A does not receive acknowledgment -> disables transmission to Core B for some period of time
- Tests for multiple failures at the same time:
  - Data from Core A does not arrive in Core B
  - Core B does not save data correctly
  - Core B is “noisy” and flooded with other data
  - Core B cannot send data onwards

1. Data is generated in Core A
2. Core A routes Data to Core B
3. Core A starts countdown timer, Core B saves data
4. Core B routes data to next core & sends acknowledge to Core A
5. Core A receives acknowledgment in time and edge to Core B stays enables



Source of temporary failures?

- Single Event Effects (SEE) can upset logic and cause abnormal behavior until reset
- Physical damage to chip can manifest in temporary failures lasting from single event to failed until reset



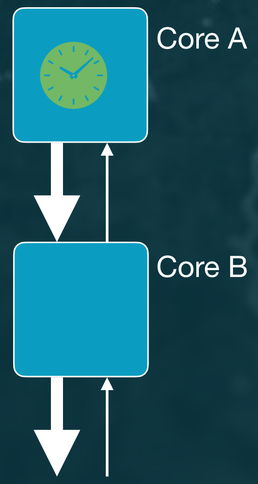
Failure Detection

BERKELEY LAB

How to identify dynamic failures?

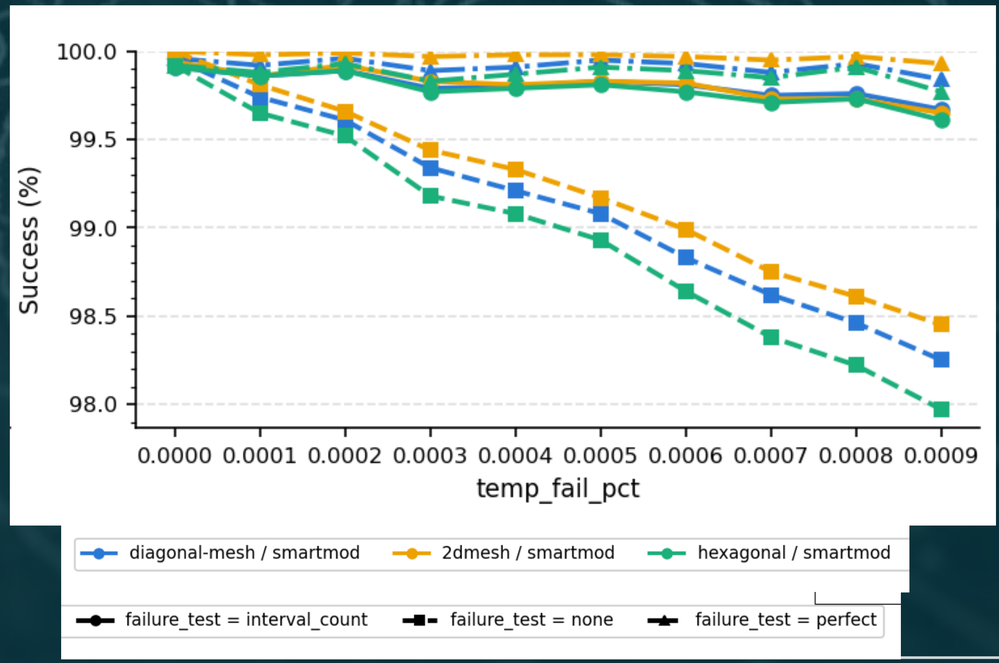
- To reduce logic/power use data to test neighboring cores
- Means protection cannot be perfect, but identification circuit becomes trivial
- Core B “acknowledge” transmission of data to the original source of the data Core A, if Core A does not receive acknowledgment -> disables transmission to Core B for some period of time
- Tests for multiple failures at the same time:
  - Data from Core A does not arrive in Core B
  - Core B does not save data correctly
  - Core B is “noisy” and flooded with other data
  - Core B cannot send data onwards

1. Data is generated in Core A
2. Core A routes Data to Core B
3. Core A starts countdown timer, Core B saves data
4. Core B routes data to next core & sends acknowledge to Core A
5. Core A receives acknowledgment in time and edge to Core B stays enables

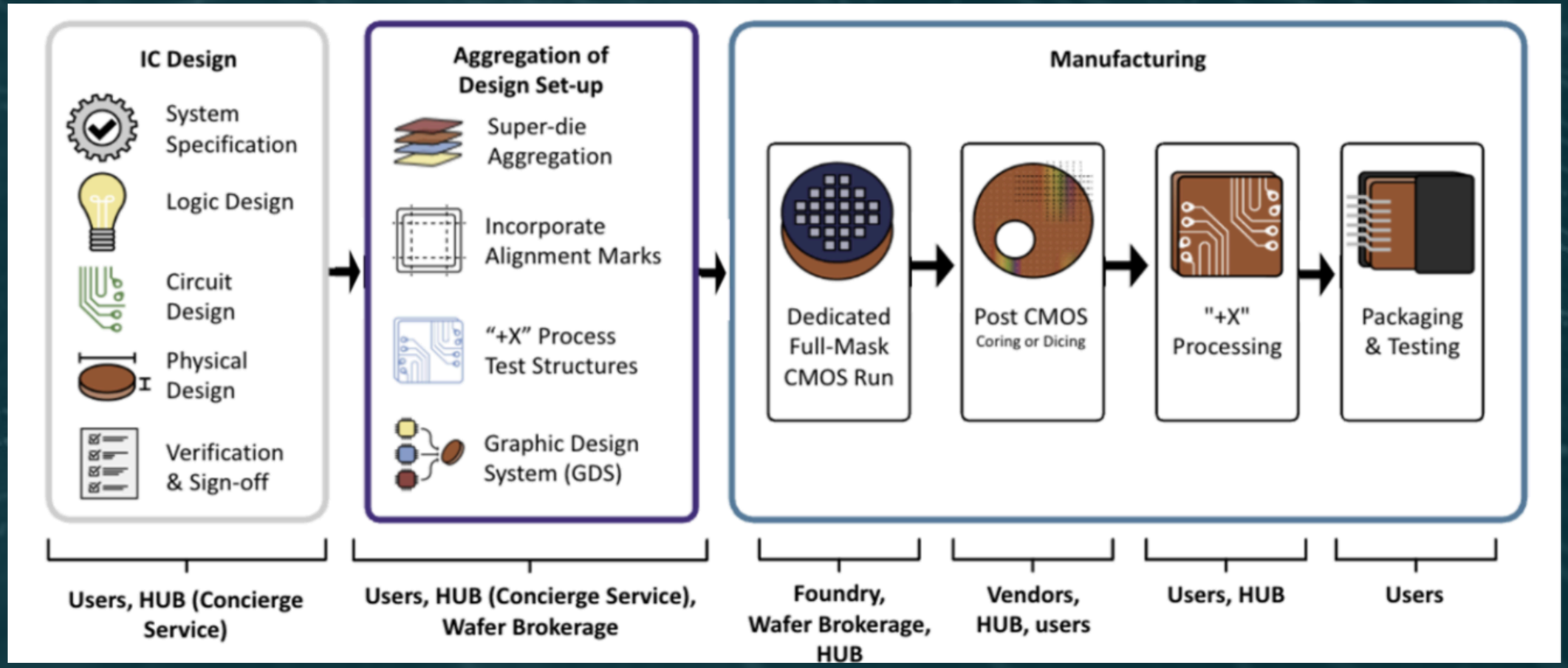


Source of temporary failures?

- Single Event Effects (SEE) can upset logic and cause abnormal behavior until reset
- Physical damage to chip can manifest in temporary failures lasting from single event to failed until reset

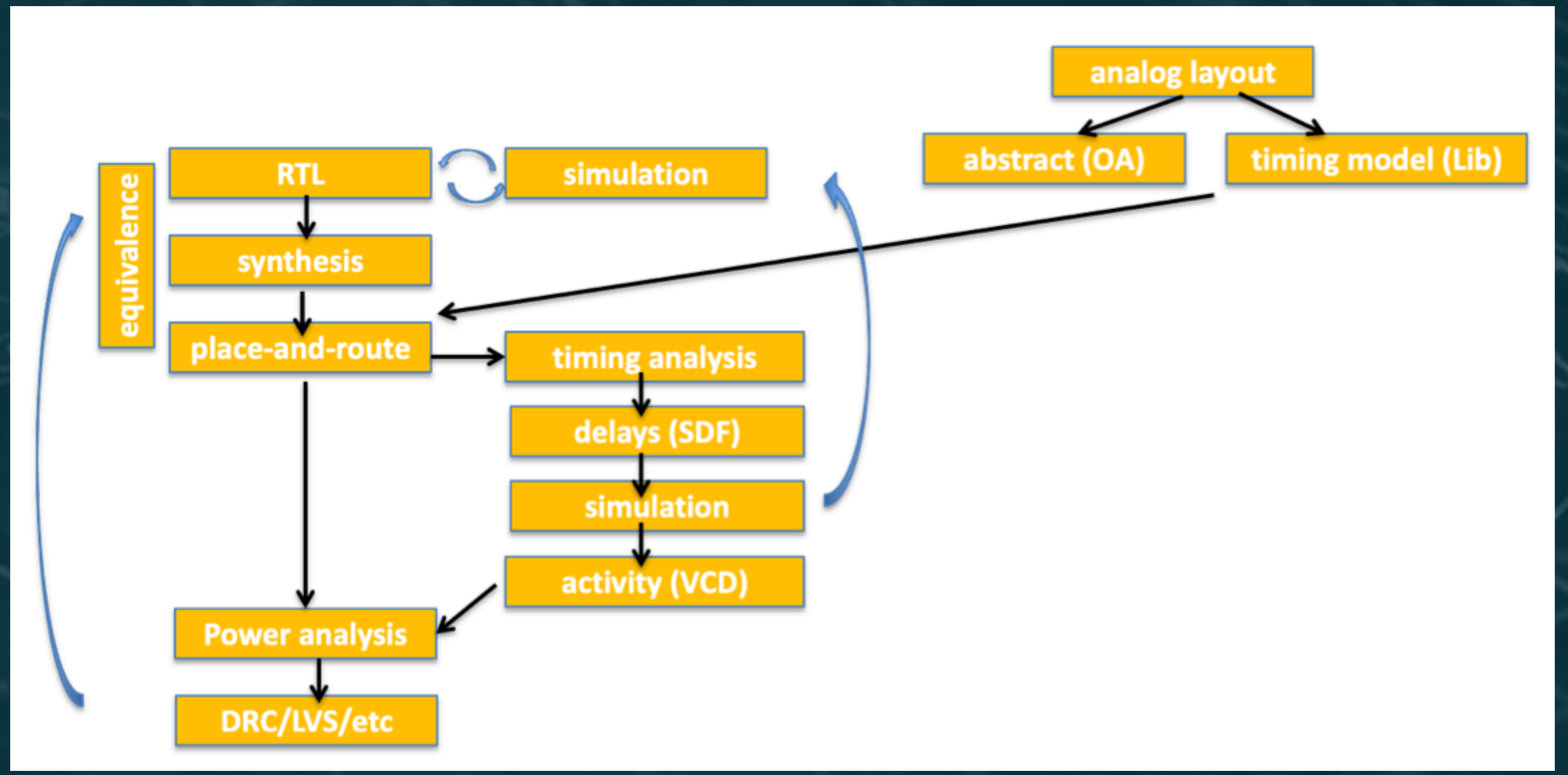


CMOS + X

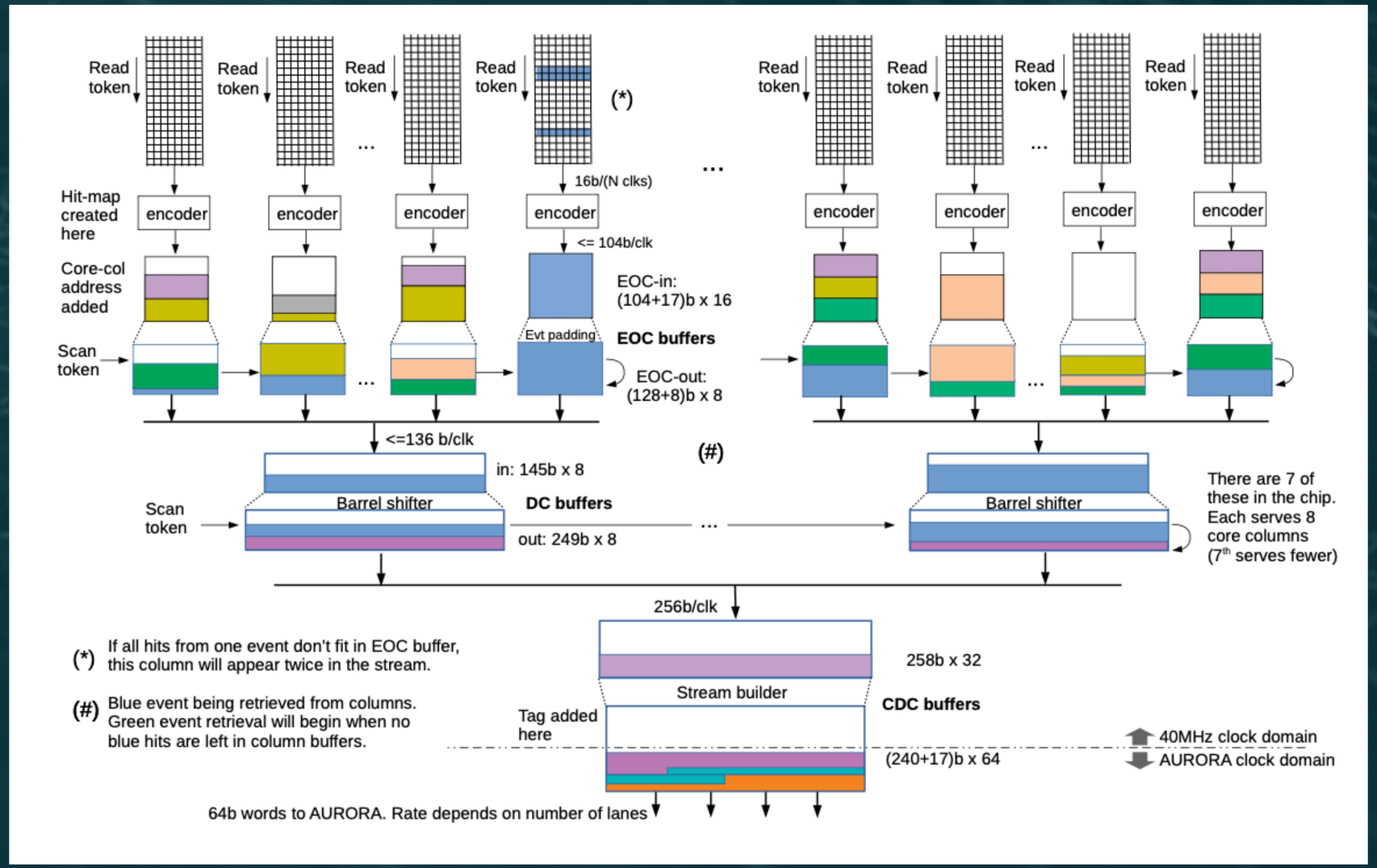


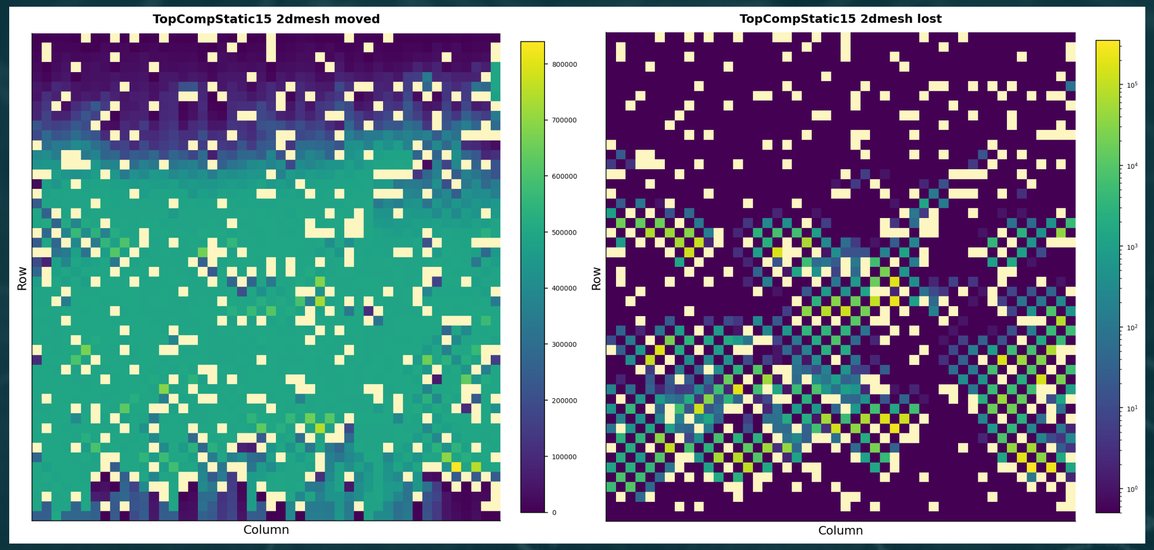
https://nw-ai-hub.org/sites/default/files/media/file/28nmshuttle-townhall-webinar-slides-mar-2026.pdf

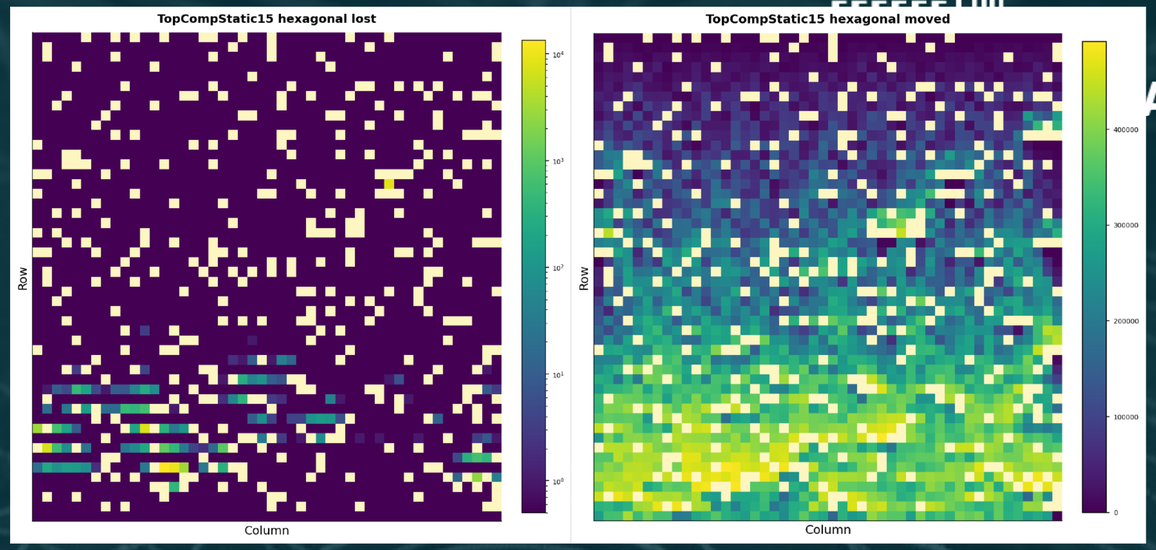
Digital-on-top Design Flow

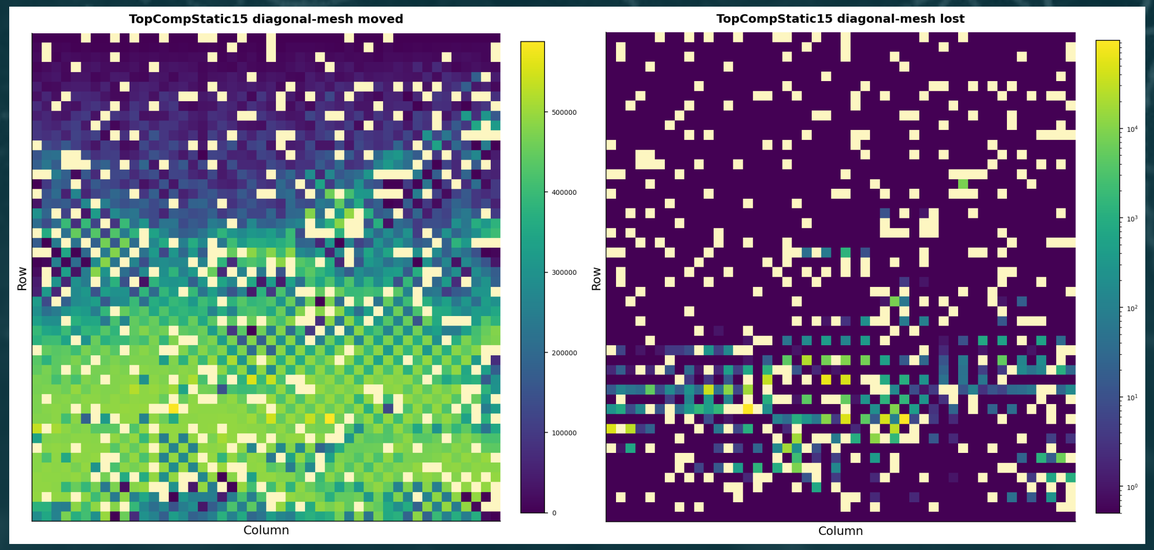


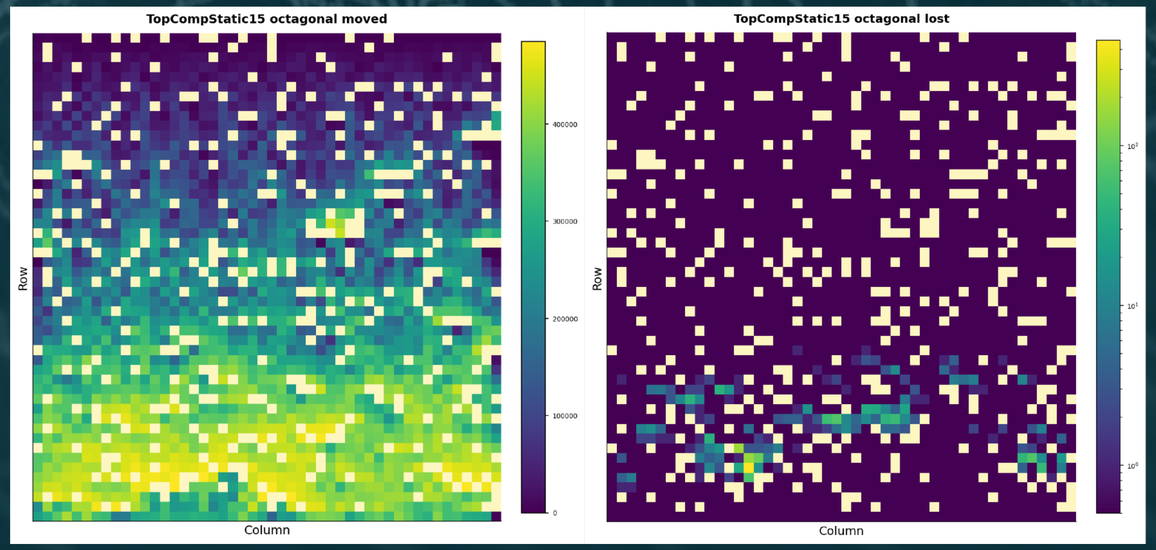
RD53 Readout Architecture



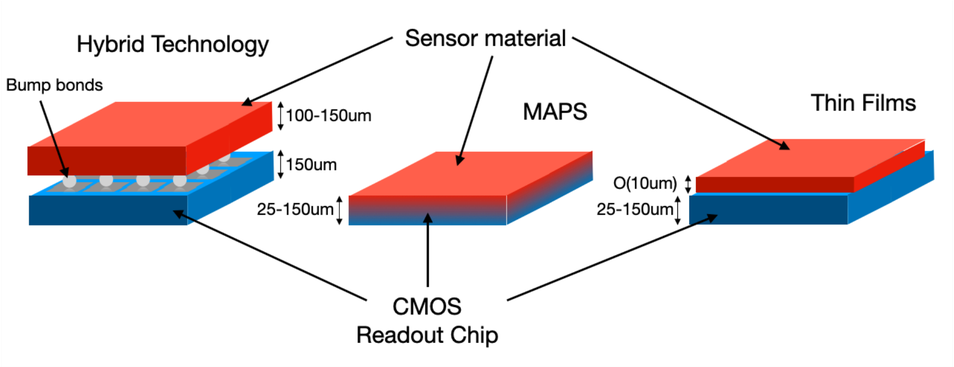

TopCompStatic15 2dmesh moved
TopCompStatic15 2dmesh lost
Row
Column
0
100000
200000
300000
400000
500000
600000
700000
800000

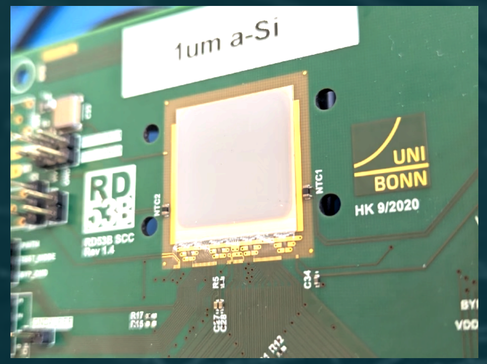

TopCompStatic15 hexagonal lost
TopCompStatic15 hexagonal moved
Row
Column
0
100000
200000
300000
400000

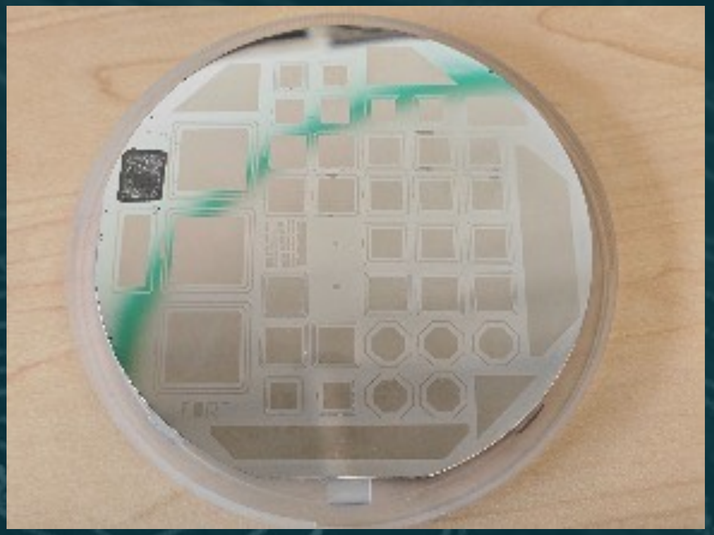

TopCompStatic15 diagonal-mesh moved
TopCompStatic15 diagonal-mesh lost
Row
Column
0
100000
200000
300000
400000
500000


TopCompStatic15 octagonal moved
TopCompStatic15 octagonal lost
Row
Column
0
100000
200000
300000
400000

Film on Readout Chip







Film on Readout Chip (FORC)

- Goal: deposit sensor material onto CMOS Pixel readout chip to enable low mass and highly scalable Pixel detectors
- Allows independent readout chip and sensors development
- LDRD in FY24/25 to investigate thin film sensors using facilities available at LBNL:
  - Semiconductor Detector Lab (SDL)
  - Molecular Foundry
  - UC Berkeley Nanolab
- Primary sensor materials investigated:
  - a-Si: known caveats but should yield functioning detector
  - IGZO (Indium Gallium Zinc Oxide): very high electron mobility and can theoretically yield equivalent amount of electron/hole pairs than crystalline silicon

Status

- Three prong investigation:
  - 1. Identify suitable materials and understand required processing/stackup
  - 2. Understand and test constraints of direct deposition on CMOS chips
  - 3. Test industrialization and scalability of process
- Do currently not have functioning prototype
- Had to learn a lot of material science, many of steps we want to do are unorthodox (>1um thick layers, substrates, deposition constraints)
- Very close to hopefully first working a-Si prototype
- Some more work needed for IGZO due to more complicated material stackup/processing

