Edge Data Processing

Ryan Herbst
TID Instrumentation
SLAC National Accelerator Laboratory

03/13/2023

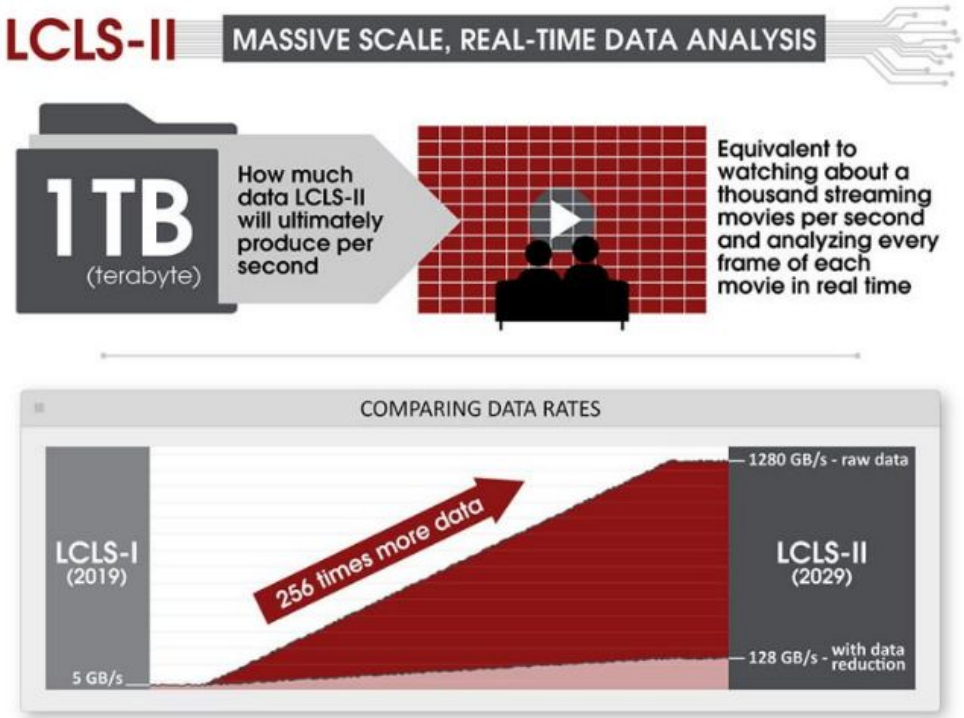
Outline

- Scientific Motivation
- Current Efforts For Edge Processing
  - FPGA
  - eFPGA in ASIC
  - Hardwired ASIC
  - DreamChip (Stanford Univ.)
- SNL Framework
  - Overview
  - Status
  - Collaboration
- Future Thoughts

Scientific Motivation

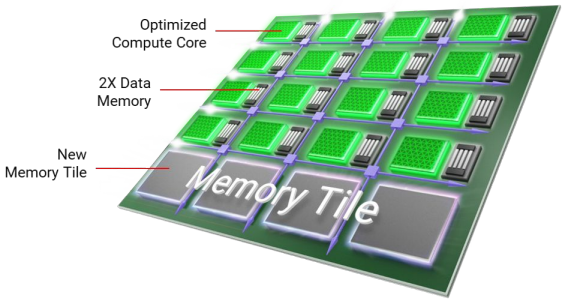
- Detectors developed for LCLS2 will generate data volumes many orders of magnitude greater than seen in LCLS
- Pre-processing and data selection at the Edge will be a powerful tool in the overall data reduction effort at SLAC
- While many groups are thinking about the algorithms to be deployed at the Edge, TID Instrumentation is working to provide the tools to deploy these engines in FPGAs and ASICs as close to the detector as possible.
- This required a close partnership between data scientists and engineers to find the appropriate algorithm for each stage of system data flow

https://www6.slac.stanford.edu/news/2021-02-17-bigger-faster-more-powerful-slacs-new-x-ray-laser-data-system-will-process-million

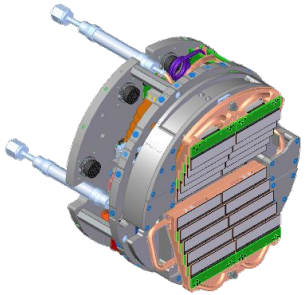


FPGA Deployments

- SLAC is working on a general library to support AI inference deployment on FPGAs (SNL) as well developing HLS based data processing
- Engines can be deployed close to the detector or at the edge of the Data Reduction Pipeline (DRP)
- On detector FPGAs are the low hanging fruit for AI and edge algorithm deployments
- Of course to get closer to the data source, ASIC deployments will be necessary…
  - See Lorenzo's talk on SparkPix-rt (compression in ASIC)



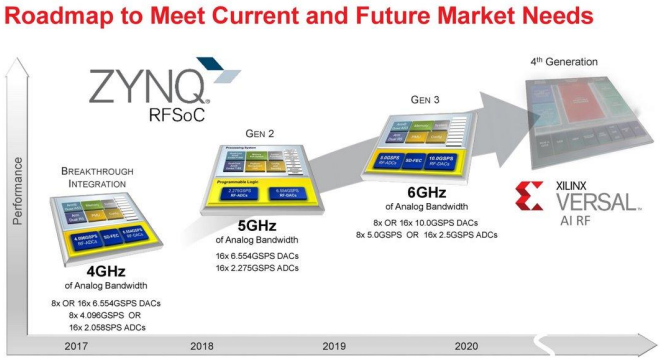
Xilinx ACAP Core



Custom Detectors (ePix Family)

Commercial Cameras

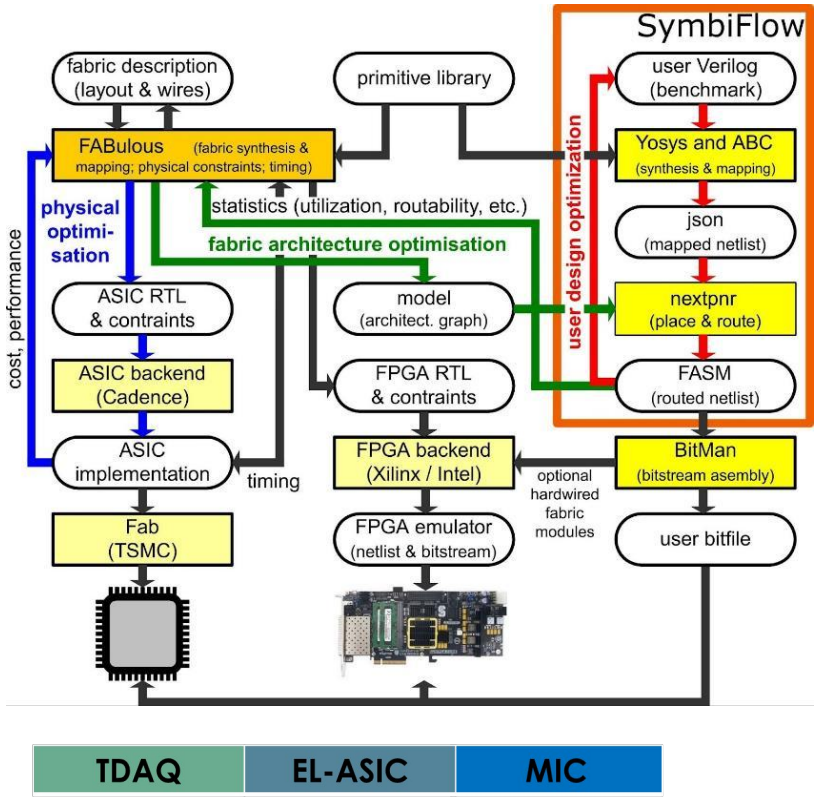
Second Stage Processing In PCIE FPGAs



Opportunity to couple RF processing & AI in ACAP + RF (Xilinx Roadmap)

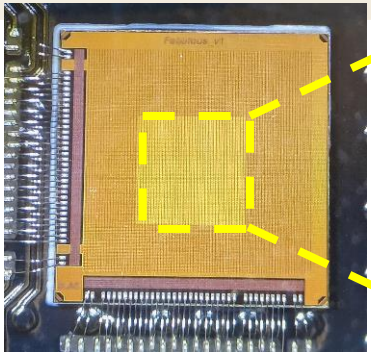
Automated data processing in ASIC/FPGA - FABulous

- Goal is to move more data processing into the front-end ASIC devices
  - Often algorithms and data processing techniques must evolve which make ASIC deployment problematic
  - Custom ASICs need to support updatable data processing pipelines
- Several popular FPGA architectures are becoming 20+ years old
  - Which means that original patents have expired
  - Includes Spartan-3 and Virtex-II FPGAs from Xilinx
- In 2021, University of Manchester has started an **open-source** project called FABulous
  - an Embedded FPGA (eFPGA) Framework
- Idea is that you put an "reconfigurable logic" in your ASIC design
- SLAC is experimenting with this approach to determine it feasibility for front end data processing, both classical and ML based

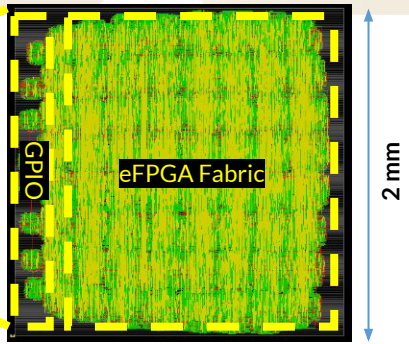


FABulous - Highlights

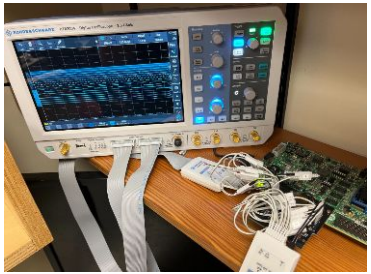
- Simple example eFPGA design to tryout the framework
  - 384 logic cells
  - 128 registers
  - 4 DSP slices
- Submitted on TSMC 130nm MPW on May 15, 2022
- ASIC wire bonded to an FMC carrier and eFPGA bitstream loaded from a Xilinx development board
- Testing started Aug 20, 2022
- **ASIC+eFPGA functionality demonstrated**
- FY2023 fabulous KA25 is doing 28nm tapeout in July 2023
  - Increases gate density and provides radiation tolerance
- **Goal:** get more familiar with the open-source framework and tools before implementing an eFPGA in a readout ASIC
  - Also determine where SLAC can contribute to this effort

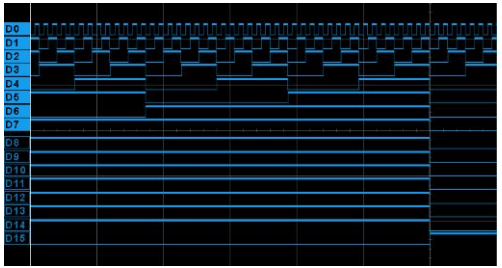


Fabulous v1 ASIC



Floorplan of eFPGA fabric after Place&Route

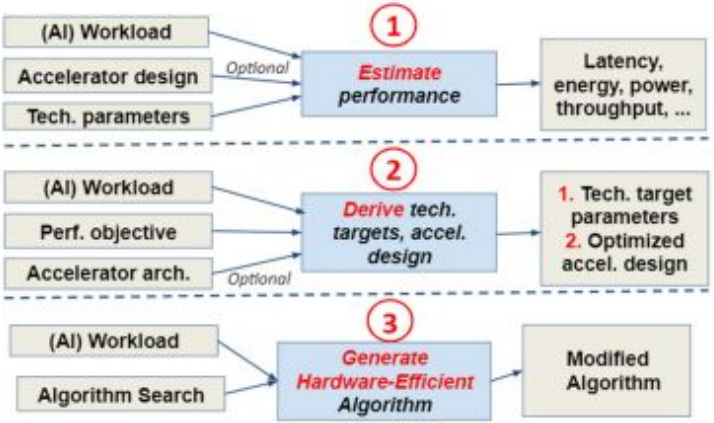




Outputs from eFPGA (16b counter) probed with oscilloscope

DreamChip (Stanford Univ.)

- Collaborative effort with Stanford's Robust Systems Group
  - Partly funded through DOE-SC Microelectronics proposal (McIntyre, et all)
- Goal is to develop a framework to optimize ASIC resources for a targeted envelope for AI & classic data reduction algorithms
  - Allows for re-synthesis and re-targeting of modified networks of a similar structure
- Output is a ASIC core along with a parameterized compiler (HLS based) for synthesising towards the custom ASIC core
- Large models and algorithms will utilize 3D integration, optimized inter-chip communication & low latency memory access
- Input to the Stanford design flow will consist of a SLAC SNL based AI model currently targeted towards FPGAs
- Optimized alternative to direct HLS -> ASIC design flow



DreamChip Development Flow

SLAC Neural Network Library (SNL) Goals

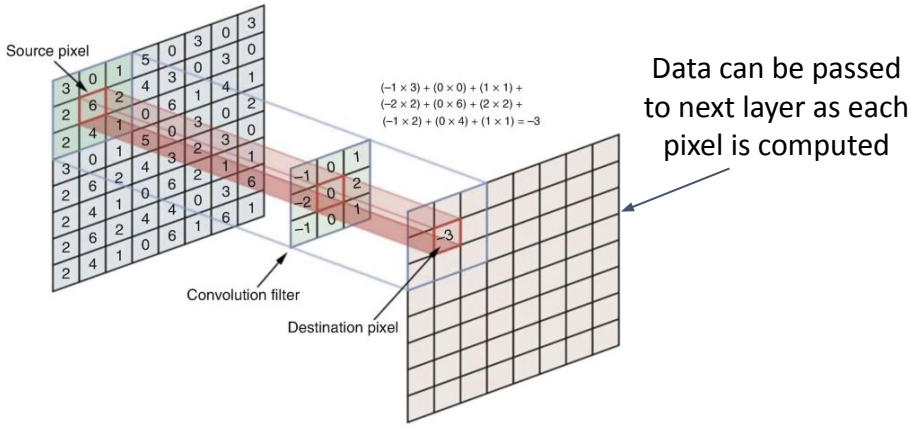
- Provide a **set of libraries** to synthesize AI inference networks into FPGAs
- Networks of a medium size,
  - **10 - 20 layers**, ~100s of thousand trainable parameters
  - Total end to end latency of **1uS** to 10ms
  - Deep networks call for AI-specific ASICs, e.g. IPU or Groq, but **FPGAs still ubiquitous** at sensor
- Pipelined implementations targeting a frame rate of **100kHz - 1MHz**
- **Dynamic reloading of weights** and biases that avoids re-synthesis
- Support a **Keras-like API** for layer definition and configuration
- Allow for a standard interface such as **HDF5** for storing weights and biases
- Allow for a modular approach with the ability to implement new and **custom layers**

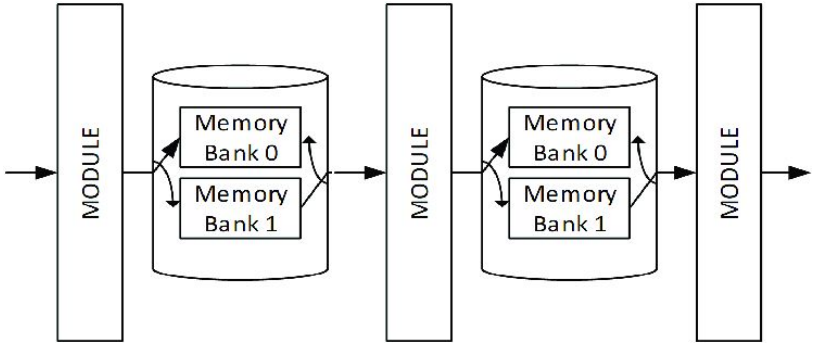
Why Dynamic Weights & Biases?

- Our SNL implementation is targeting **scientific instruments** which will continuously adapt to new data and **changing environments**
  - **High speed training** to supports this goal
  - Bias and weight **updates in real time**
- Some AI-to-FPGA frameworks take the weights and biases as an input, pruning portions of the network structure to save resources
  - **Re-synthesis is required** for each new training set
  - **Risk of the FPGA implementation failing** due to increase resources usage, timing failures or massive change in internal interconnect structure
- **Large FPGA designs can take hours to days** for the HLS -> synthesis -> place and route cycle to complete
- This is a trade off between **robustness vs. latency** and resource usage

Why Streaming & pipelining?

- Certain layer types work better with streaming based upon their **data flow** model
  - Convolution (2D & 3D) and pooling layers can process **data as it arrives**
  - **Partial data** is ready as each region of the layer completes
  - Pass to next layer can begin when indices reach roughly ½ the kernel dimensions
- Other layers such as **Dense fully connected** need all of the input data to have arrived before output can begin… this suffers a **high latency penalty**
- Idea is to trade off overall latency vs resources usage while maintaining high rate throughput



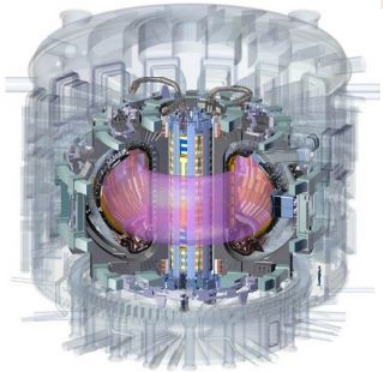


SNL Usage

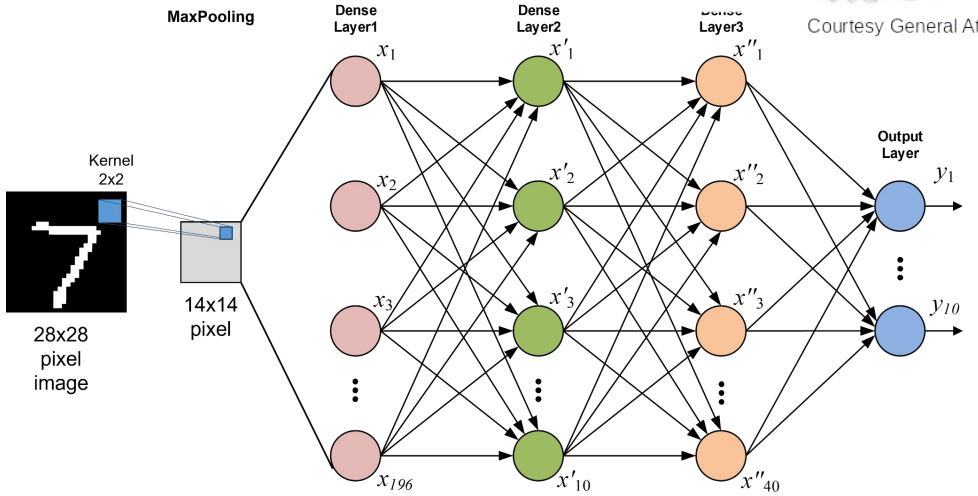
- User defines layers using a **collection of C++ templates** that define each layer type and the associated activator for each layer
- Current layer types:
  - Conv2D
  - MaxPooling
  - AveragePooling
  - Dense
  - Reservoir
- Current activators:
  - LeakyRelu
  - Relu
- Each layer interface matches closely to the Keras model
  - Allows user to use **Keras documentation as a reference**
- Additional configurations are required to control the **FPGA specific configurations**

Example Use: BES Network

- Initial example of a **small fully connected network** implemented as if a CNN (Dave Smith U. Wisc.)
- Beam Emission Spectroscopy (BES) for tokamak **disruption event identification** (Edge Localized Mode - ELM)
- Developed with intention of **forecasting ELMs**
- The BES network consists of the following layers:
  - MaxPool2D
  - Dense with LeakyReLU activator

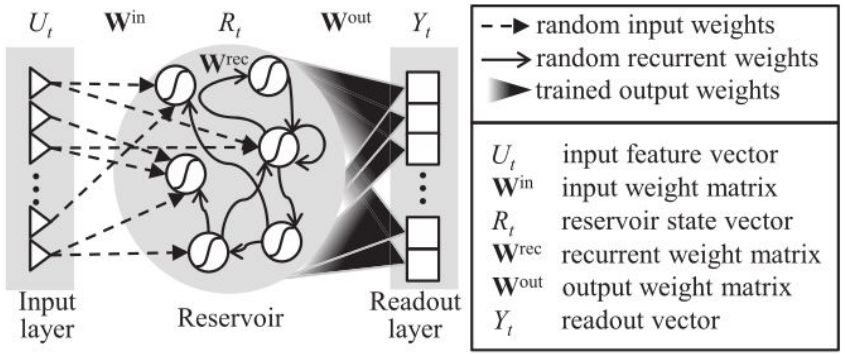


Courtesy General Atomics



Example Use: Reservoir Networks

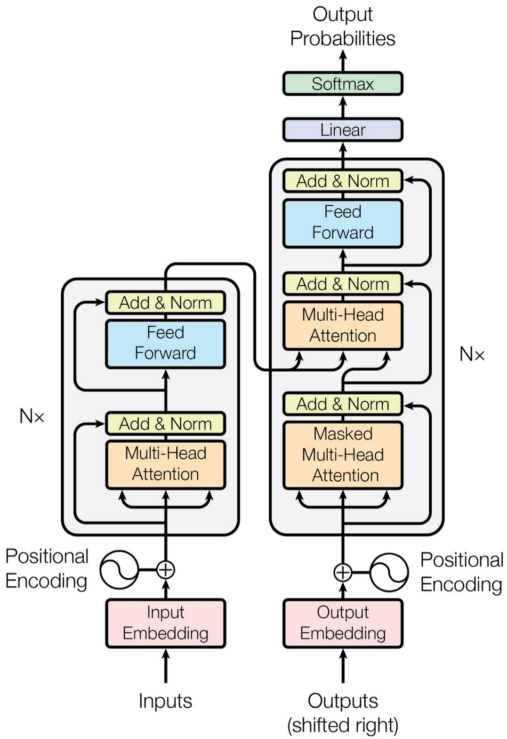
- A Reservoir network consists of a deep pool of recurrently connected neural nodes which take state from both the input and neighbor nodes on each computation cycle
  - Input and internal weights and biases are assigned randomly
  - Only output values apply weights and biases to data from each internal node
- Pros for FPGA deployment:
  - **Less trainable parameters** to download to the FPGA
  - Internal **connections do not change** with each training and so can be pruned and optimized
- Cons for FPGA deployment:
  - **Very large set of node interconnections**
- We have successfully deployed an example network
  - 2 Reservoir layers with a large vector: **~1000 nodes**
  - Results in **large usage of DSPs** and memory
  - Requires a **very large FPGA** to implement
  - **High latency** in the 100uS - 10mSec range



Azarakhsh Jalalvand *et al.* "Real-Time and Adaptive Reservoir Computing With Application to Profile Prediction in Fusion Plasma" IEEE Trans Neural Netw Learn Syst. **33**(6) :630-2641 (2022)

Example Use: Transformer Networks

- Transformer networks
  - **Conceptually similar to Reservoir** networks
  - Targeted toward using **less FPGA resources**
  - **Trainable parameters are contained** in the dense feed forward blocks (looking into Hopfield networks to replace attention)
  - Remaining interconnects and static parameters require **new interconnect types in SNL**
- Our current focus is to adapt SNL to support the deployment of these network types
  - **Domain scientists and FPGA designers** must work closely together to optimize design for FPGA deployment
  - Current workflows and models are very software oriented and should leverage the **huge potential for FPGA specific enhancements**



SNL Limitations

- The user must be aware of some limitations when making use of the SNL framework
  - SNL is **not a finished, polished product**.
- We are **continuously developing**, adding new features and improving the structures
- Only a **subset of the Keras** layers and activators are implemented
  - New layers can be added in a well defined but tedious process
  - Some layer types and **activators are not consistent** with best FPGA **data flow**
- We have **not yet implemented quantization** in the implementation
  - Future plans include this but need a clear understanding of overflow handling
- End to end optimization of the layer structure is not implemented
  - We are looking to the **Xilinx SLX design flow** as a possible solution
- We are very interested in collaborators
  - HLS experience needed!
  - Experience with targeted inference networks (not Gpt3!)

Future Work & Conclusions

- Continued development of SNL library
  - Looking for collaborators and HLS experts to assist in supporting more complex network structures
- Continued development of HLS cores for edge DATA processing
- Interest in approaches integrating custom front end ASICs with co-processing chiplets
  - (DreamChip, AI engines, FPGA dies, etc)
- Interest in analog DSPs coupled with quantized digital AI engines
  - Accelerate the processing latency and footprint of DSPs in custom ASICs
- Silicon Quantum Processors

- Challenges
  - Tiled detectors and where data can truly be processed
  - Calibration, baseline subtraction and geographic corrections
    - Also provide opportunities for AI deployment
- Potential opportunity for FW based event building
  - Network attached FPGAs coordinating data delivery
  - Boundary data passing for edge ML processing

Acknowledgement

Department of Energy, Office of Science

- Office of Basic Energy Science
  - CookieBox detector: FWP 100498 (PI Coffee) “Enabling long wavelength Streaking for Attosecond X-ray Science”
  - EdgeAI for CookieBox: FWP 100643 (PI Thayer) “Actionable Information from Sensor to Data Center”
  - DOE High Energy Physics Detector R&D

- Office of Fusion Energy Science
  - EdgeAI for Fusion Control: FWP 100636 (PIs Koleman, Coffee, Smith, Boyer, Schneider) “Machine Learning for Real-time Fusion Plasma Behavior Prediction and Manipulation.”

- SLAC-LCLS Program Development (PI Kling) “LCLS Growth”

- SLAC TID LDRD 21-007 (PI Herbst) “Edge ML for acquisition and analysis of data generated by ultra high rate detectors”