Empowering AI Implementation:
The Versatile SLAC Neural Network Library (SNL) for FPGA, eFPGA , ASIC

SLAC TID

Ryan Herbst, Ryan Coffee, J.J Russell, Abhilasha Dave, Dionisio Doering, Larry Ruckman

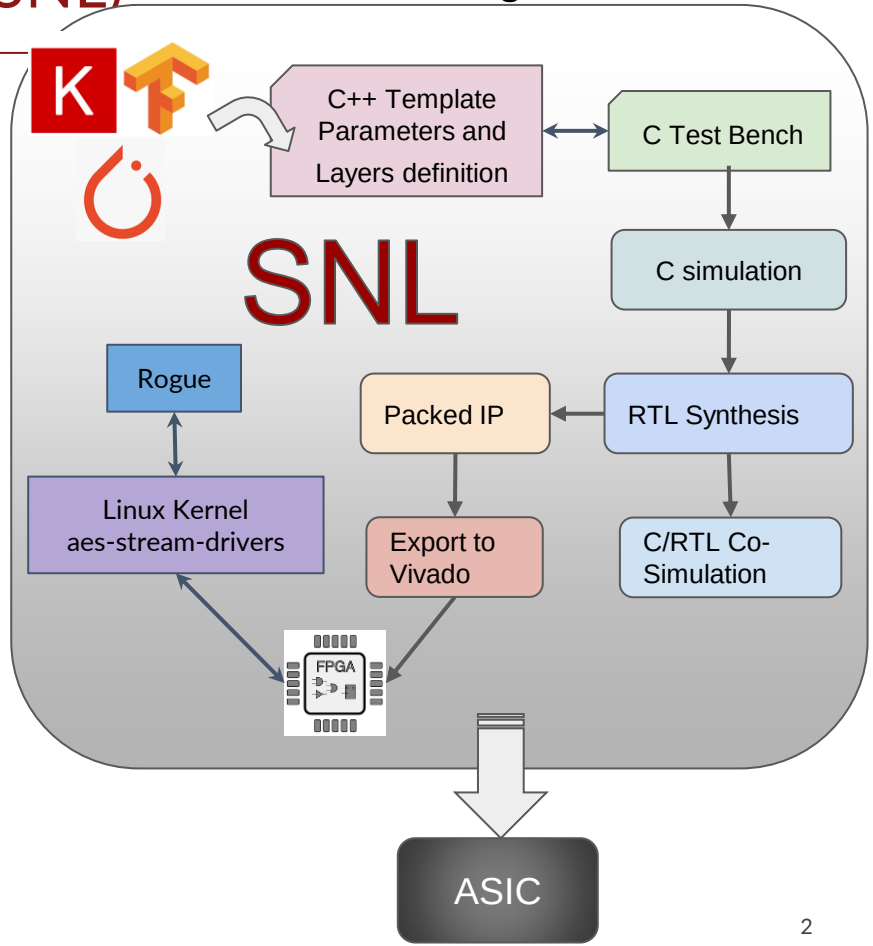
November 7 2023

SLAC Neural Network Library (SNL)

Key points:

- Provides specialized set of libraries designed in High-Level Synthesis (HLS) for deploying ML inferences on FPGAs, eFPGAs and ASICs
- At the edge of the data chain, SNL aims to create a high-performance, low-latency FPGA implementation for AI inference engines.
- Supports Keras like API for layer definition
- **Dynamic reloading** of **weights** and **biases** to avoid re-synthesis
- Supports 10s of thousands of parameters or more depending on latency requirements for the inference model
- Total end to end latency of ~couple of usec to couple of millisecond.
- Streaming interface between layers.

SNL's Design Flow



FPGAs for Latency Critical Neural Network

In context of neural network inference run ASICs, FPGAs, CPUs, GPUs comes with their own strengths and weaknesses.

CPUs

- CPU offers the **greatest programming flexibility**.
- **Lower** compute **throughput**.
- **Limited Parallelism:** CPUs are **optimized for single-threaded or limited multi-threaded performance,** which is often not sufficient for highly parallel workloads like neural network inference.

GPUs

- GPU performance is **typically much higher than a CPU** and improved further when using a **large batch number**.
- Processing **many queries in parallel**.

FPGAs

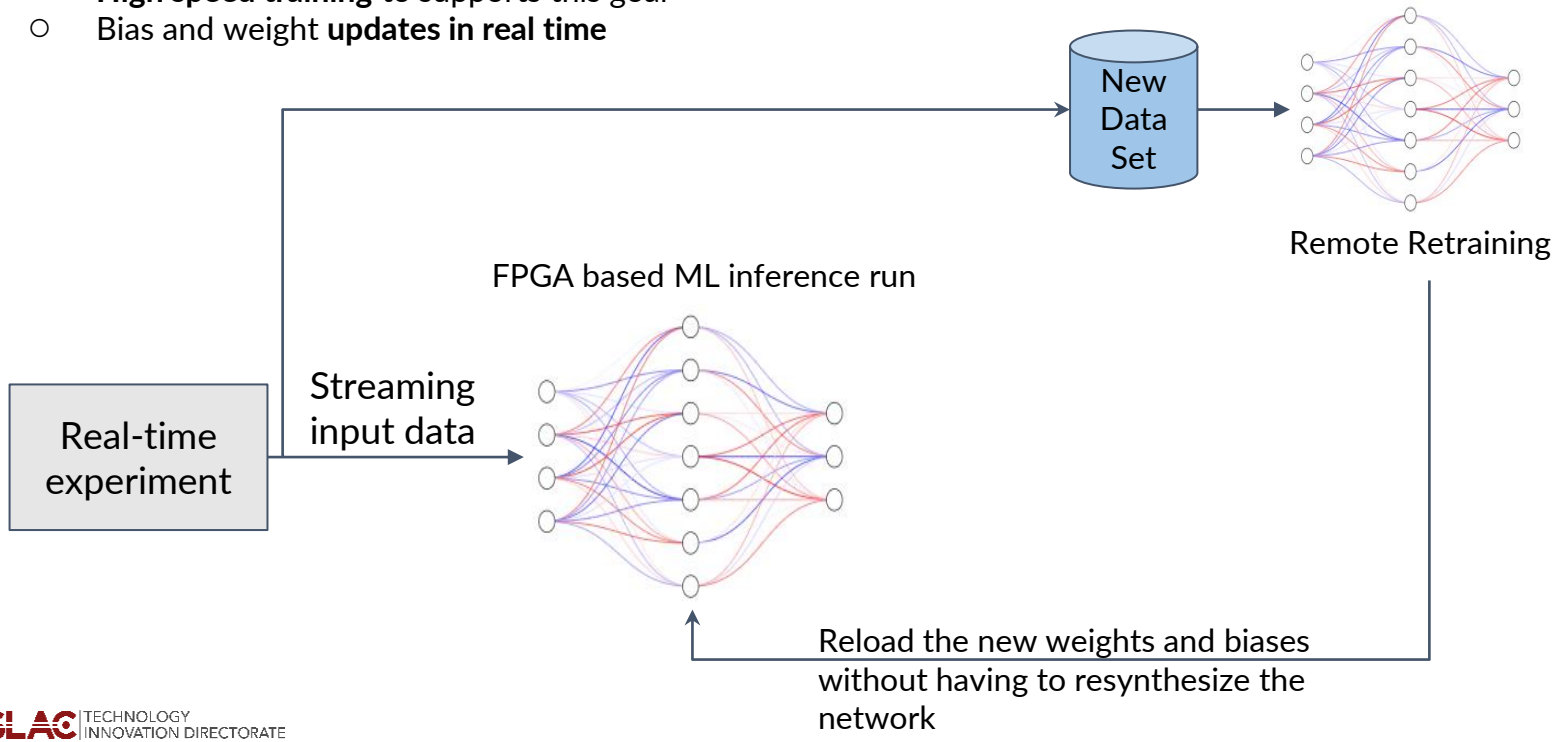
- In latency critical real-time systems, **it is not always possible to batch input data.**
- This is where FPGAs are somewhat unique, allowing neural networks to be optimized for a single query
- Still achieve a high-level compute resource utilization.
- When an ASIC does not exist, this makes FPGAs ideal for **latency critical neural network processing**.

ASICs

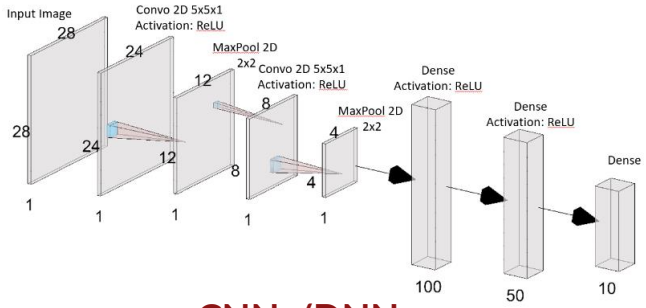
- ASICs offer the highest performance and lowest cost
- Only for targeted algorithms.
- There is no flexibility

On-the-fly Weights & Biases Loading

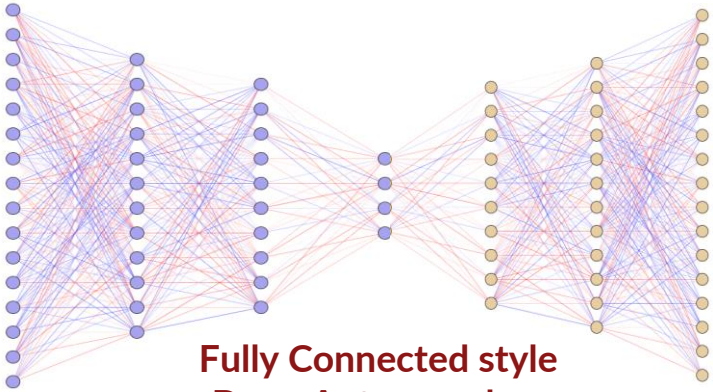
- SNL implementation is targeting **scientific instruments** which will continuously adapt to new data and **changing environments**
  - **High speed training** to supports this goal
  - Bias and weight **updates in real time**



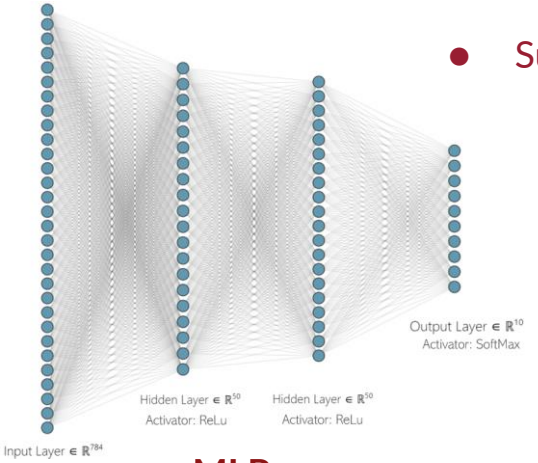
Buildable Neural Networks and Future Support



CNNs/DNNs



Fully Connected style Deep Autoencoders



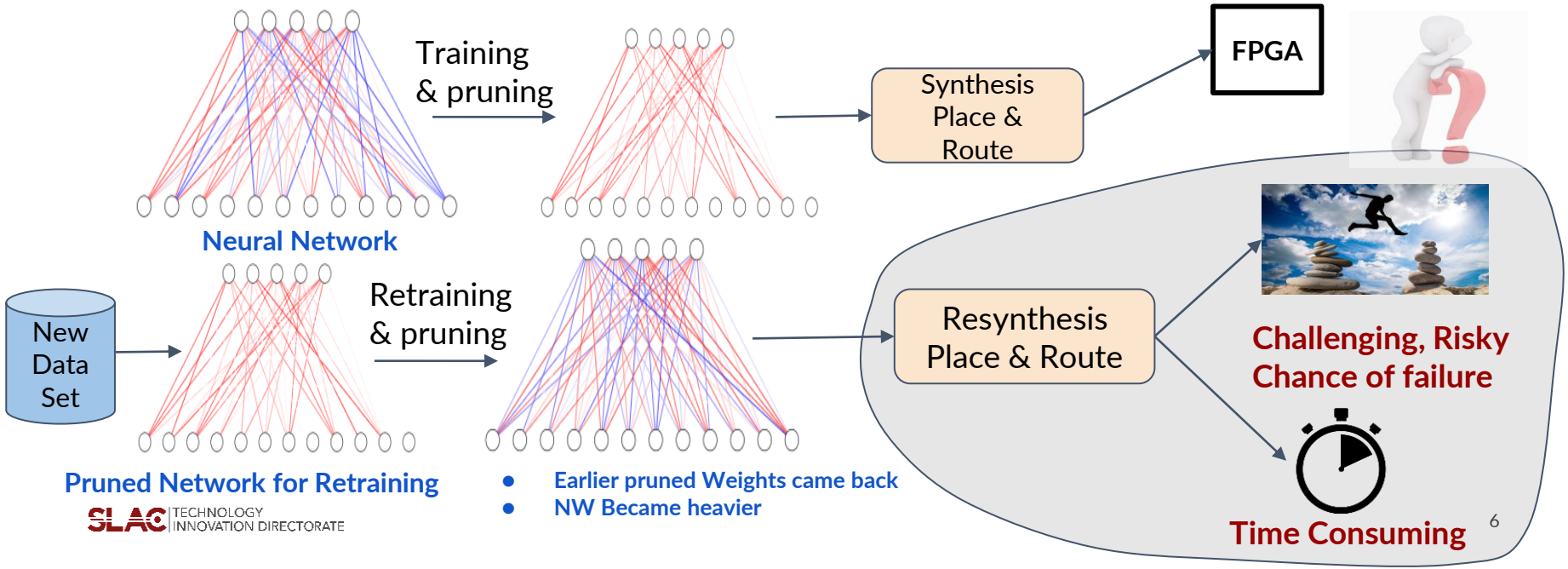
MLPs

- Supported Libraries:
  - NN Layers:Conv2D, MaxPooling2D, Average Pooling, Dense
  - Activators: LeakyRelu, Relu, SoftMax
  - Data Types: Fixed point, Integer, Floating Point

- SNL is a work in progress library
- SNL can provide future support for more libraries for variety of neural networks layer types.
  - e.g: Support to foundational Transformer Neural Network blocks
- It can also provide Novel weights loading such as binary or ternary.

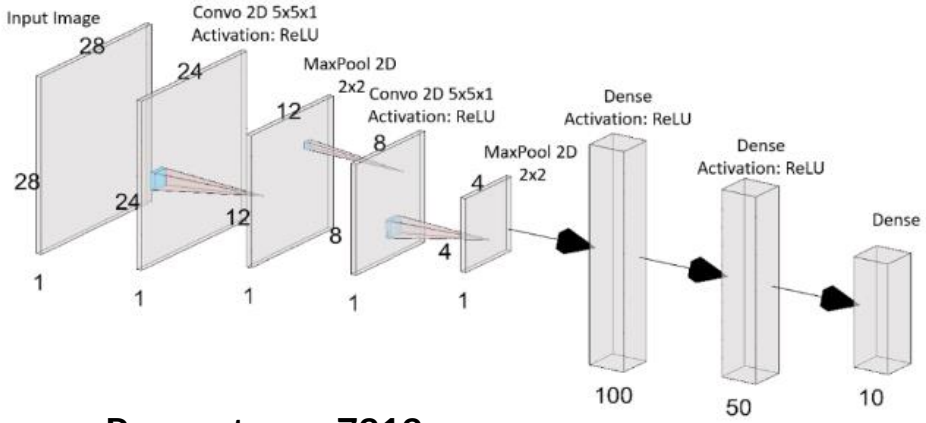
Challenges with Pruning

- Some AI-to-FPGA frameworks take the weights and biases and 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



SNL Model Example: Tiny CNN - MNIST

NW Definition and Area Consumption KCU105



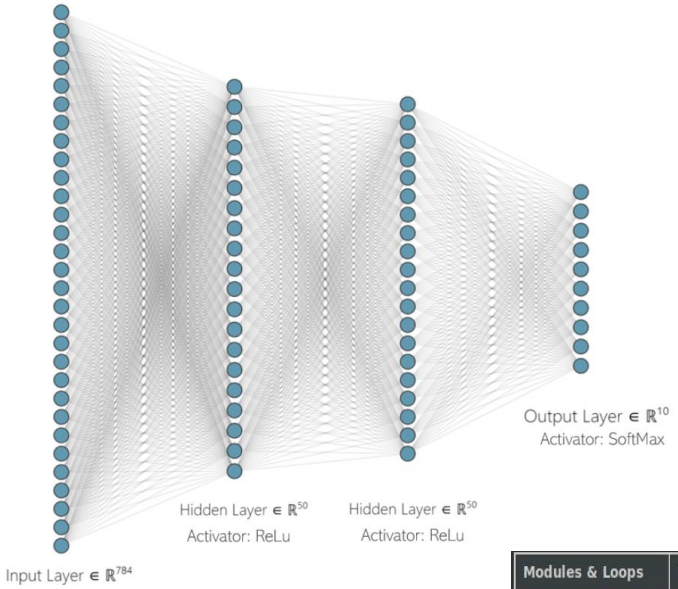
Resource	Utilization	Available	Utilization %
LUT	98953	242400	40.82
LUTRAM	10081	112800	8.94
FF	124674	484800	25.72
BRAM	100.50	600	16.75
DSP	448	1920	23.33
IO	7	520	1.35
GT	8	20	40.00
BUFG	25	480	5.21
MMCM	1	10	10.00

- **Parameters = 7312**
- **Latency = 1023 cycles**
  ~5.115us

Modules & Loops	Issue Type	Slack	Latency (cycles)	Latency (ns)
+ processNetwork*	-	0.17	1023	5.115e+03
+ process_0_s	-	0.17	1007	5.035e+03
+ process_3	-	0.17	1006	5.030e+03
+ prefill	-	1.97	125	625.000
+ prefill Pipeline conv2d window y0	-	3.15	35	175.000

SNL Model Example: MLP-MNIST

NW Definition and Area Consumption for KCU105

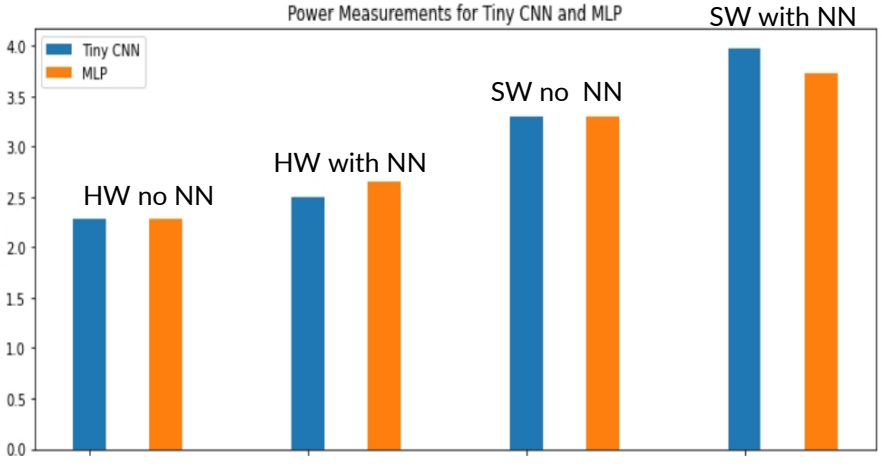


Resource	Utilization	Available	Utilization %
LUT	124740	242400	51.46
LUTRAM	24403	112800	21.63
FF	100594	484800	20.75
BRAM	132.50	600	22.08
DSP	114	1920	5.94
IO	7	520	1.35
GT	8	20	40.00
BUFG	17	480	3.54
MMCM	1	10	10.00

- **Parameters = 42310**
- **Latency = 3212 cycles**
  ~16.06us

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
processStream				-	42419	2.120E5	-	42420	-	no	88	119	75112	71852	0
construct_1				-	39203	1.960E5	-	39203	-	no	0	0	92	269	0
construct_2				-	2503	1.252E4	-	2503	-	no	0	0	76	247	0
construct				-	503	2.515E3	-	503	-	no	0	0	59	229	0
processNetwork	II Violation			-	3212	1.606E4	-	3213	-	dataflow	0	119	25110	29321	0

Further Testing with SNL: Power Comparison



	Measurement Type	Power (W)
Tiny CNN	HW Excluding NN IP	2.2745
	HW Including NN IP	2.5025
	Vivado Excluding NN IP	3.289
	Vivado Including NN IP	3.967
MLP	HW Excluding NN IP	2.2745
	HW Including NN IP	2.6545
	Vivado Excluding NN IP	3.294
	Vivado Including NN IP	3.719

- We went further and did power comparison.
- Table presents the power measurements obtained from actual KCU105 board for both CNN and MLP examples.
- **Observations**:
- There is a clear difference between power estimated by Vivado and the actual power measured on HW.
- Specifically, Vivado's power estimate for the Tiny CNN network is differs by 1.4645W when compared to the physical hardware measurement.
- Similarly, Vivado's power estimate for the MLP network differs by 1.0645W.
- For the MLP NW consumes the higher power, because it has a higher no. parameters necessitates increased memory mapping and more extensive computation between memory and logic cells, thereby leading to escalated power usage.

Challenges…

- How can we prune the model without having resynthesize the entire design?
- To target transformer style huge networks on FPGA is a complex challenging task given the FPGA resource constraints.
- It **requires Domain scientists and FPGA designers** to work closely together to optimize design for FPGA deployment
- It also requires a significant modification in transformer architecture and training processes.