CERN openlab

Graph Neural Network(GNN) Inference on FPGA

CERN openlab Lightning Talks

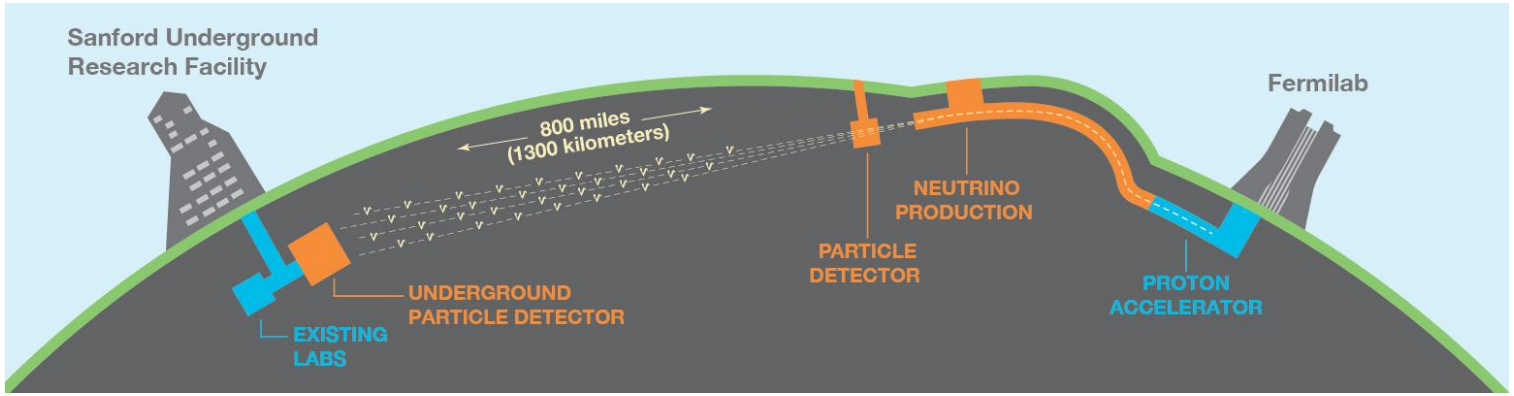
Kazi Ahmed Asif Fuad

Supervisor: **Sofia Vallecorsa**

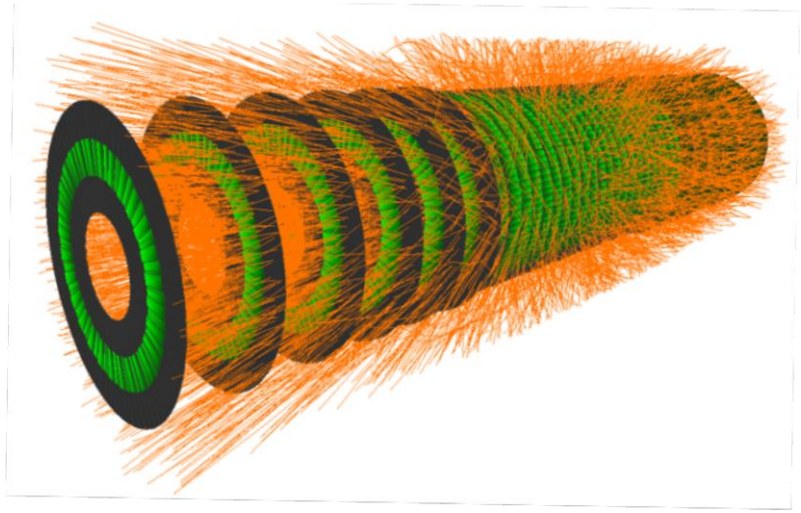
15/08/2019

Project Background

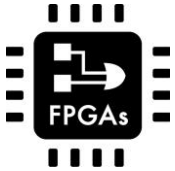
DEEP UNDERGROUND NEUTRINO EXPERIMENT



Our Objective



Field Programmable Gate Arrays



Track Reconstruction

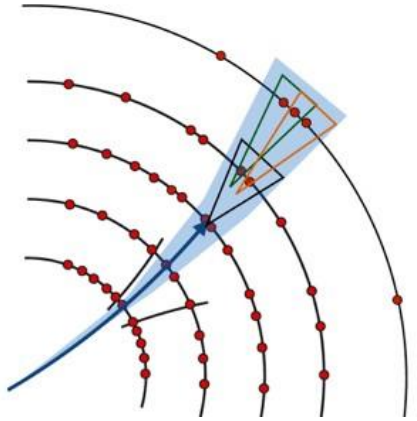
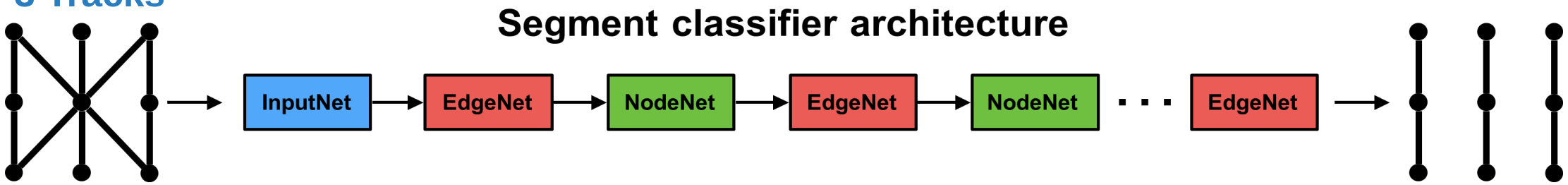


Image Based Methods → **Space-Point Representation**

https://indico.cern.ch/event/753577/contributions/3123602/attachments/1707996/2752966/acts-gnn-Aug30.pdf
https://indico.cern.ch/event/658267/contributions/2881175/attachments/1621912/2581064/Farrell_heptrkx_ctd2018.pdf
HEP.TrkX: https://heptrkx.github.io/

Graph Neural Network (GNN)

3 Layers
3 Tracks



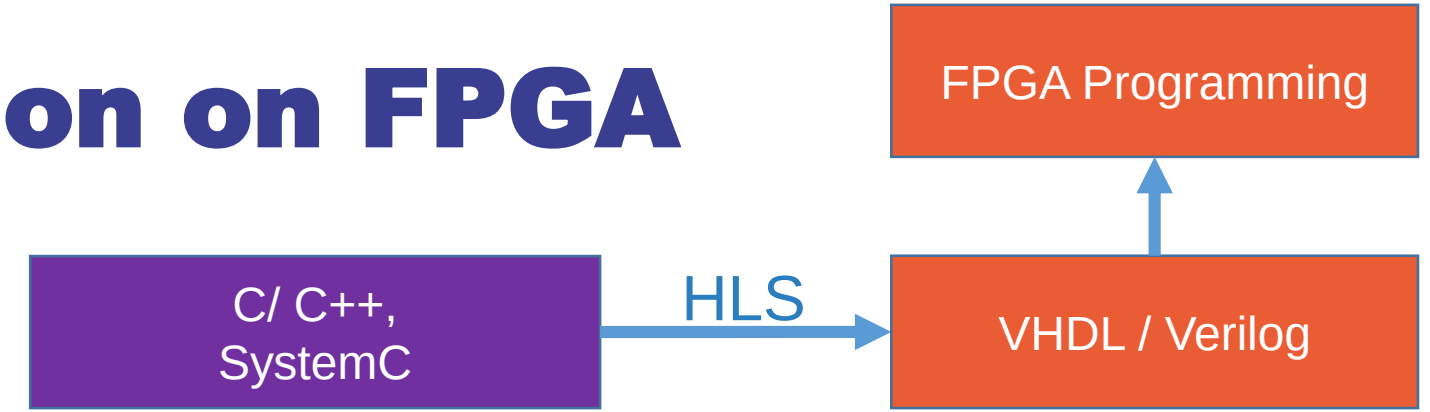
With each iteration, the model propagates information through the graph, strengthens important connections, and weakens useless ones.

InputNet: New Features tanh activations	EdgeNet: Edge Weights tanh activations sigmoid activation	NodeNet: New Features tanh activations tanh activations
1 Layer MLP	2 Layer MLP	2 Layer MLP

https://indico.cern.ch/event/753577/contributions/3123602/attachments/1707996/2752966/acts-gnn-Aug30.pdf
https://indico.cern.ch/event/658267/contributions/2881175/attachments/1621912/2581064/Farrell_heptrkx_ctd2018.pdf

Implementation on FPGA

High Level Synthesis

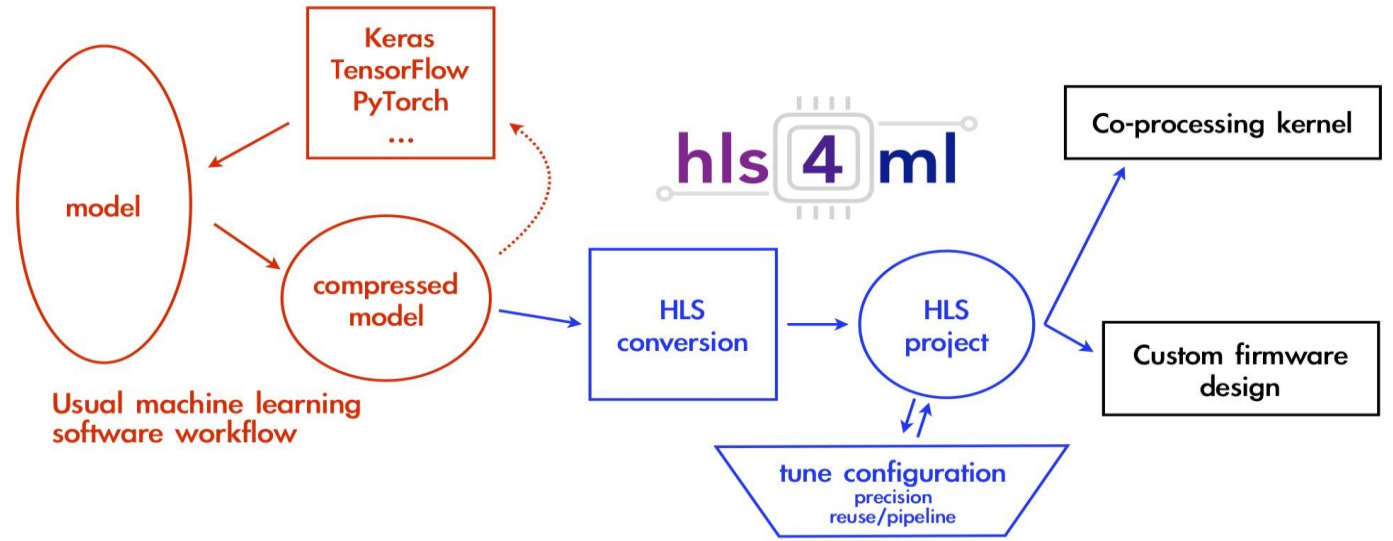


hls4ml

A package for machine learning inference in FPGAs.

FPGA

Reconfigurable
PIPELINED Operation
High Speed Inference



hls4ml: *https://hls-fpga-machine-learning.github.io/hls4ml/*

More FPGA Facts

- ✓ Basic Building Resource Blocks: LUTS, DSPs, Flip-Flops & BRAMs.
- ✓ Resource Utilization needs to be less than 100% to **fit** a design into FPGA.
- ✓ Resource Utilization of SLR less than 100% is good for the design.
- ✓ Reuse Factor means how many times the DSP(Multiplier + Adder) block will be used.
- ✓ PIPELINE Architecture is faster than Dataflow Architecture but utilizes more resources.

My HLS Implementation

For HLS implementation, I have marged following implementations.

GNN implementation of **Javier M. G. Duarte (Fermilab)**
https://github.com/hls-fpga-machine-learning/hls4ml/tree/jmgd/graph/example-prjs/graph

Reference for GNN

+

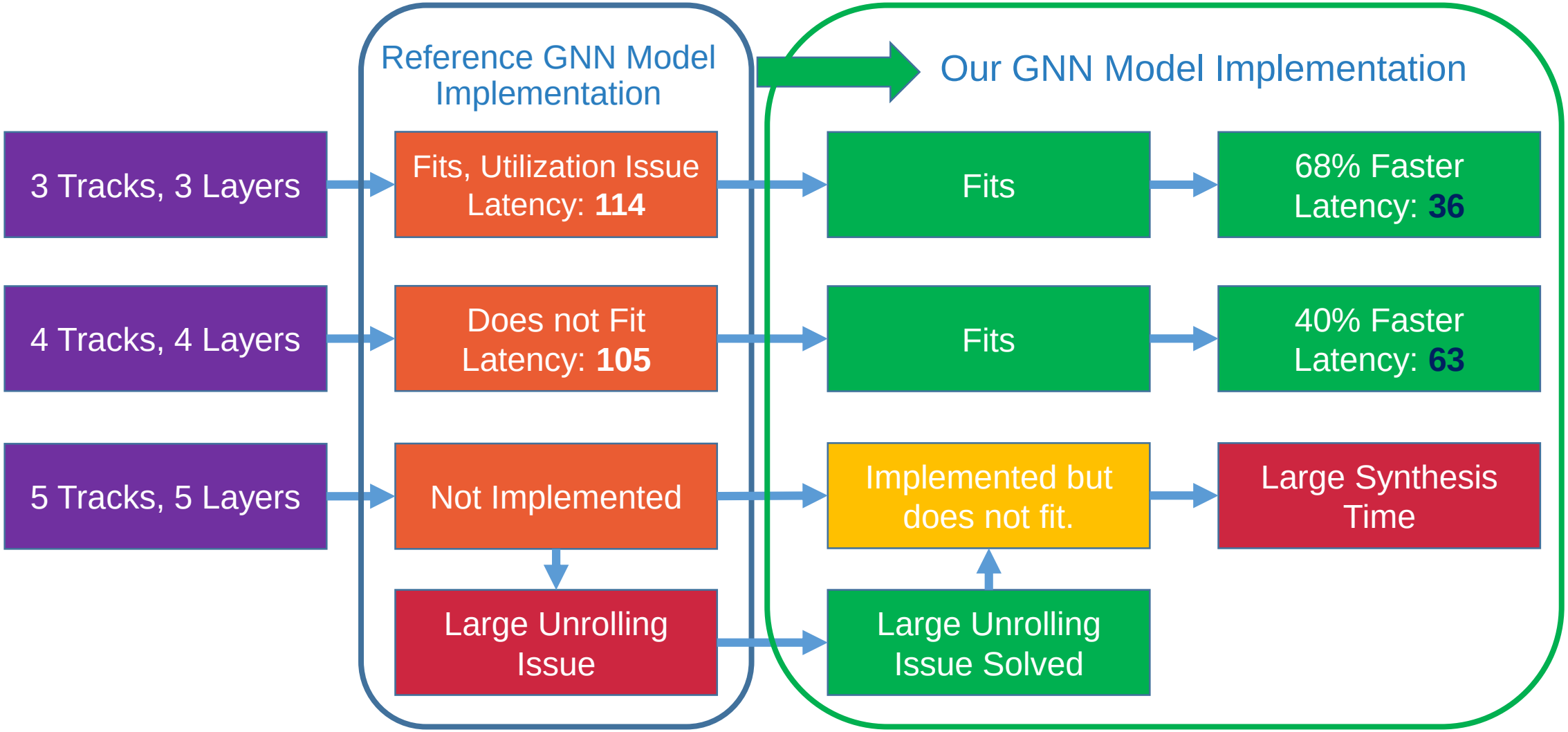
Large Dense Layers Implementation from **Vladimir Loncar** (**CERN)**:
https://github.com/vloncar/hls4ml/tree/hack6

Reference for NN

My implementations are available at:
https://github.com/belloworld/hls4ml/tree/hack6/example-prjs/GNN

Our Implementation

Results for Pipeline Architecture



Issues, We are Facing in Pipeline

- Reuse Factor Not Working
- Large Synthesis Time

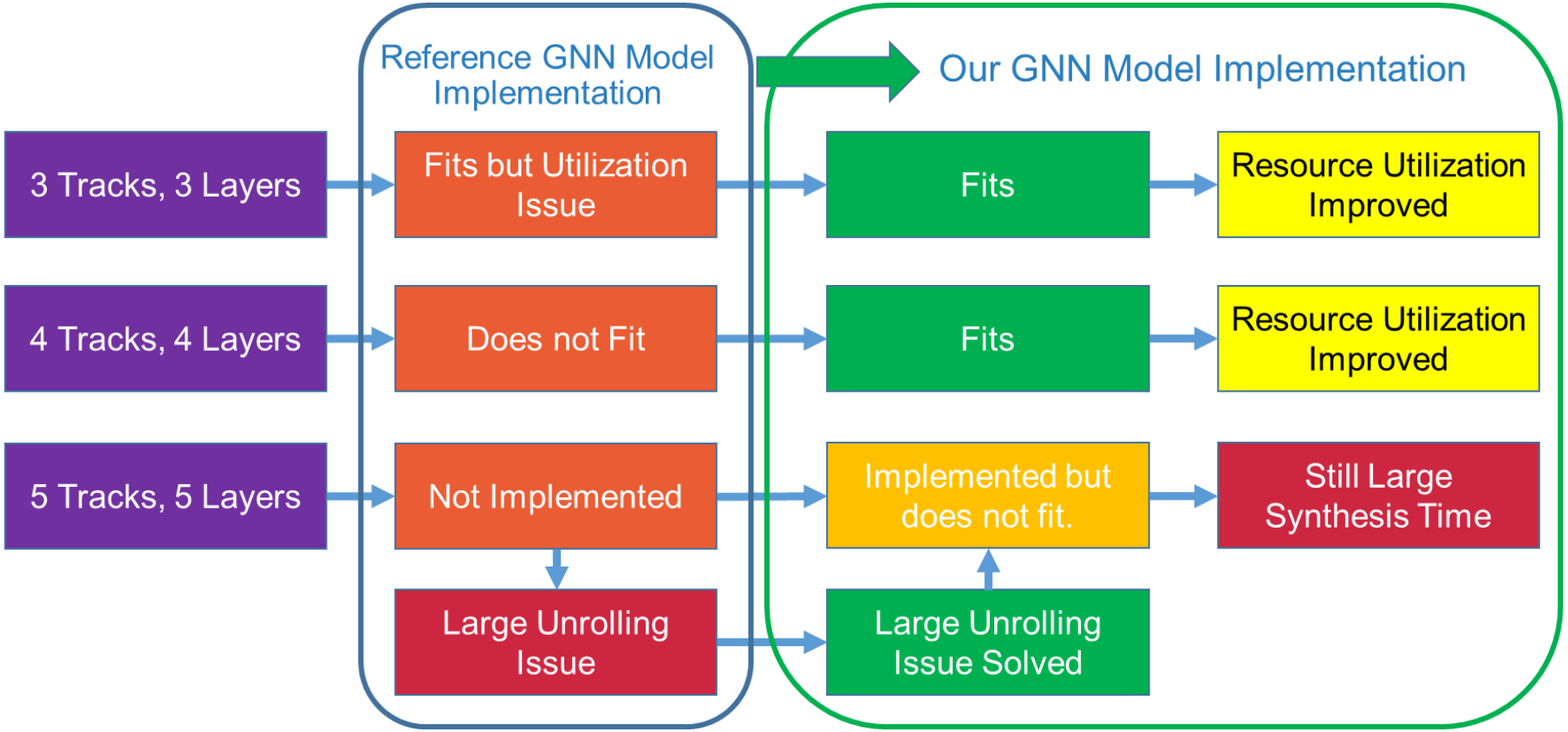
After discussions…..

Opting to…..

DATAFLOW

Results for Dataflow Architecture

REUSE Factor Works but Long Synthesis time not solved yet!



Things to do and Future Work(!)

In summary,

My 1st implemented GNNs around around **40% faster** in Pipeline architecture.

My 2nd implementations are using around **45% less resources** than the reference.

- ❑ More Investigation on the design.
- ❑ Different perspective for large unrolling issue
- ❑ Run the 3 Tracks, 3 Layers GNN on Kintex FPGA.

Ultimate Target is the 10 Tracks, 10 Layers GNN (!)

Special Thanks to...

CERN openlab

Sofia Vallecorsa

Vladimir Loncar

hls4ml

Fermilab

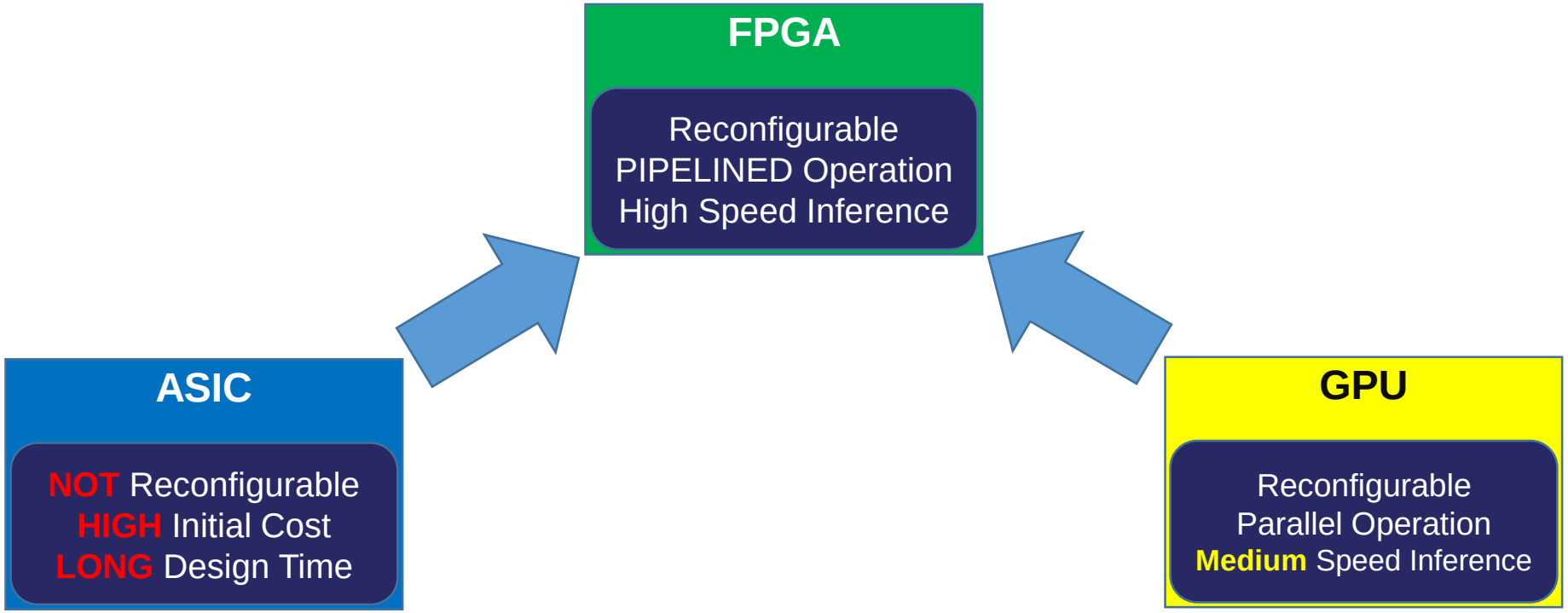
QUESTIONS?

asif.ahmed.fuad@gmail.com
https://www.linkedin.com/in/asif-fuad/

Additional Slides

Why FPGA?

FPGA
Field Programmable Gate Arrays



ASIC
Application Specific Integrated Circuit

GPU
Graphics Processor Unit

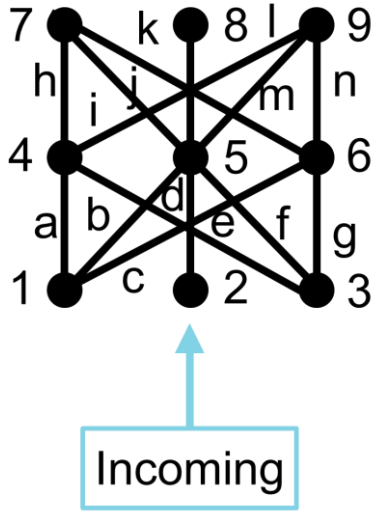
https://www.arrow.com/en/research-and-events/articles/fpga-vs-cpu-vs-gpu-vs-microcontroller
https://lancesimms.com/Microprocessors/CPU_vs_GPU_vs_FPGA.html
https://numato.com/blog/differences-between-fpga-and-asics/

A Simple Graph

A Simple 3 Layers Graph

Each layer has 3 Nodes(hits)

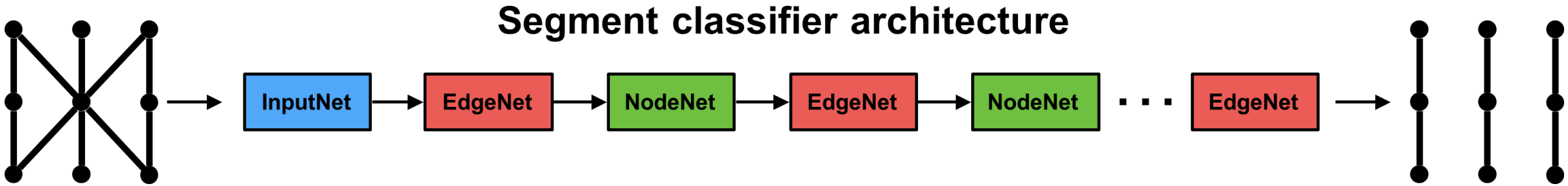
Our objective is to identify “Good” segments



https://indico.cern.ch/event/753577/contributions/3123602/attachments/1707996/2752966/acts-gnn-Aug30.pdf

https://indico.cern.ch/event/658267/contributions/2881175/attachments/1621912/2581064/Farrell_heptrkx_ctd2018.pdf

Graph Neural Network (GNN)



With each iteration, the model propagates information through the graph, strengthens important connections, and weakens useless ones.

Definitions

X ⟶ (N x D) node feature matrix

R_i ⟶ (N x E) association matrix of nodes to input edges

R_o ⟶ (N x E) association matrix of nodes to output edges

https://indico.cern.ch/event/753577/contributions/3123602/attachments/1707996/2752966/acts-gnn-Aug30.pdf
https://indico.cern.ch/event/658267/contributions/2881175/attachments/1621912/2581064/Farrell_heptrkx_ctd2018.pdf
HEP.TrkX: https://heptrkx.github.io/

Graph Neural Network (GNN)

- The edge network is a 2-layer MLP with tanh and sigmoid activations:

$$w = f_{\text{edge}}(R_i^T X, R_o^T X) \longrightarrow \text{(E) edge weight array}$$

- The node network is a 2-layer MLP with tanh activations:

$$X' = f_{\text{node}}\Big((R_i \odot w)R_o^T X, (R_o \odot w)R_i^T X, X\Big) \longrightarrow \text{(N x D) node features}$$

- Outputs
  - Node classifier: binary classifier layer gives score for each node
  - Segment classifier: final edge network application gives score for each edge

https://indico.cern.ch/event/753577/contributions/3123602/attachments/1707996/2752966/acts-gnn-Aug30.pdf
https://indico.cern.ch/event/658267/contributions/2881175/attachments/1621912/2581064/Farrell_heptrkx_ctd2018.pdf
HEP.TrkX: https://heptrkx.github.io/

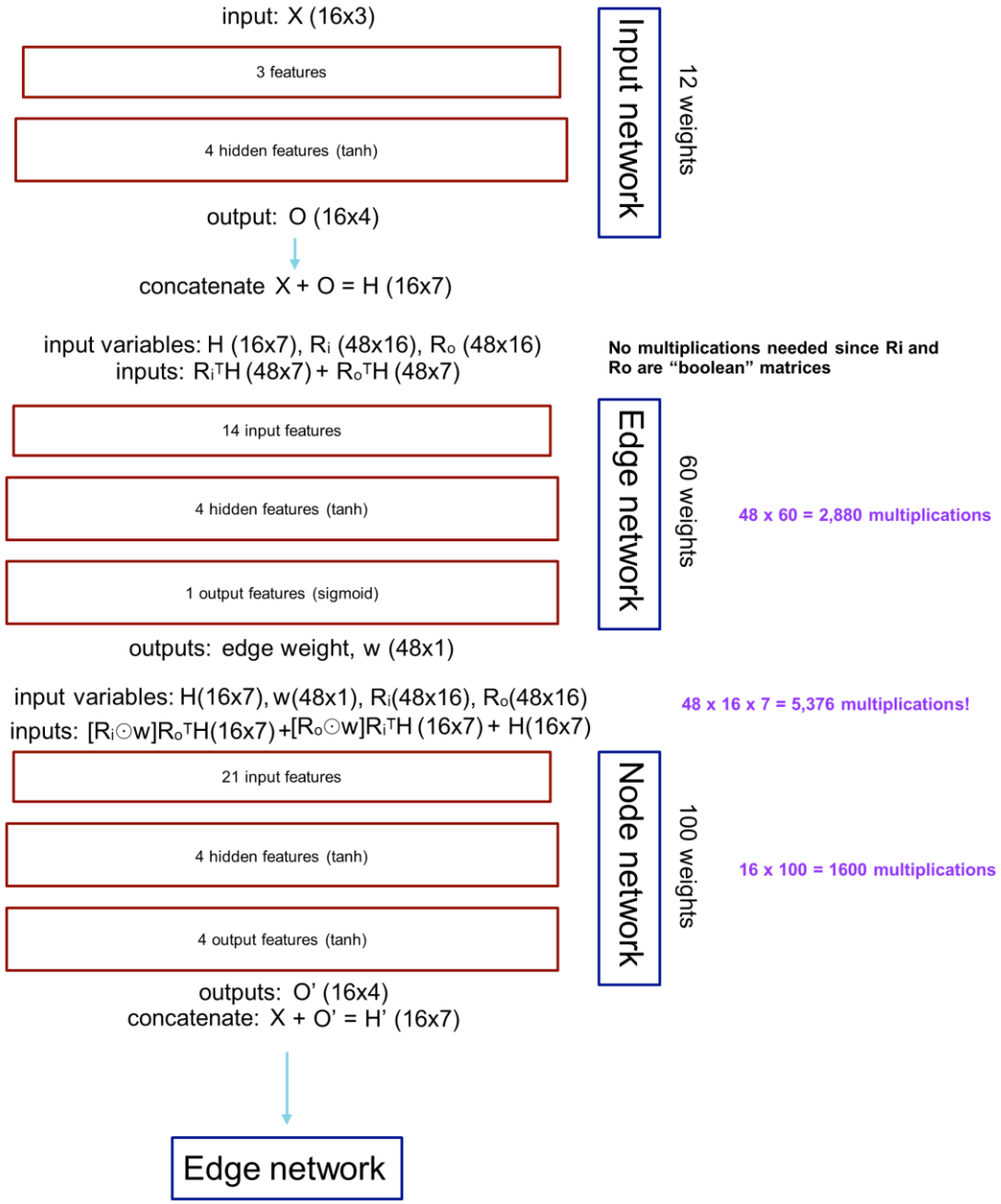
4 Tracks, 4 Layers
1 Iteration

Input Network:
12 weights

Edge Network:
60 weights
48 x 60 =2880 multiplications

48 x 16 x 7= 5,376 multiplications

Node Network:
100 weights
16 x 100 = 1600 multiplications



TIMING & RESOURCE USAGE (3 TRACKS, 3 LAYERS)

Kintex FPGA: xcku115-flva1517-1-c : Pipeline Architecture

GNN Resources Usage for Pipeline Architecture		Device: xcku115-flva1517-1-c							
		Utilization Estimates: Vivado HLS C Synthesis				Utilization Estimates: Vivado Synthesis			
		DSP48E	Change	LUT	Change	DSP48E	Change	CLB LUT	Change
	Available	**5520**	**na**	**663360**	**na**	**5520**	**na**	**663360**	**na**
	Available SLR	**2760**	**na**	**331680**	**na**	**na**	**na**	**na**	**na**
Reuse=1	Total(Used)	**5067**	**-776.64%**	**420412**	**-15.90%**	**5049**	**-773.53%**	**143112**	**60.55%**
	Utilization(%)	**91**	**-810.00%**	**63**	**-16.67%**	**91.47**	**-814.70%**	**21.57**	**60.06%**
	Utilization SLR (%)	**183**	**-815.00%**	**126**	**-15.60%**	**na**		**na**	
Reuse=7	Total(Used)	**1484**	**-156.75%**	**295398**	**18.56%**	**3309**	**-472.49%**	**106199**	**70.72%**
	Utilization(%)	**26**	**-160.00%**	**44**	**18.52%**	**59.95**	**-499.50%**	**16.01**	**70.35%**
	Utilization SLR (%)	**53**	**-165.00%**	**89**	**18.35%**				
Reuse=21	Total(Used)	**1161**	**-100.87%**	**285845**	**21.19%**	**3023**	**-423.01%**	**107392**	**70.39%**
	Utilization(%)	**21**	**-110.00%**	**43**	**20.37%**	**54.76**	**-447.60%**	**16.19**	**70.02%**
	Utilization SLR (%)	**42**	**-110.00%**	**86**	**21.10%**				

	Latency (Clock Cycles)			
	Latency			
	Min	Change	Max	Change
Reuse=1	**21**	**81.58%**	**21**	**81.58%**
Reuse=7	**36**	**68.42%**	**36**	**68.42%**
Reuse=21	**73**	**35.96%**	**73**	**35.96%**

TIMING & RESOURCE USAGE (4 TRACKS, 4 LAYERS)

Kintex FPGA: xcku115-flva1517-1-c : Pipeline Architecture

GNN Resources Usage for Pipeline Architecture		Device: xcku115-flva1517-1-c							
		Utilization Estimates: Vivado HLS C Synthesis				Utilization Estimates: Vivado Synthesis			
		DSP48E	Change	LUT	Change	DSP48E	Change	CLB LUT	Change
	Available	**5520**	**na**	**663360**	**na**	**5520**	**na**	**663360**	**na**
	Available SLR	**2760**	**na**	**331680**	**na**	**na**	**na**	**na**	**na**
Reuse=1	Total(Used)	**17616**	**-823.27%**	**1798687**	**-19.40%**	**5520**	**-189.31%**	**2285042**	**-51.68%**
	Utilization(%)	**319**	**-838.24%**	**271**	**-19.38%**	**100**	**-194.12%**	**344.46**	**-51.74%**
	Utilization SLR (%)	**638**	**-824.64%**	**542**	**-19.38%**				
Reuse=7	Total(Used)	**5664**	**-196.86%**	**1386769**	**7.95%**	**2432**	**-27.46%**	**1181929**	**21.54%**
	Utilization(%)	**102**	**-200.00%**	**209**	**7.93%**	**44.06**	**-29.59%**	**178.17**	**21.51%**
	Utilization SLR (%)	**205**	**-197.10%**	**418**	**7.93%**				
Reuse=21	Total(Used)	**4640**	**-143.19%**	**1355382**	**10.03%**	**2582**	**-35.32%**	**1025563**	**31.92%**
	Utilization(%)	**84**	**-147.06%**	**204**	**10.13%**	**46.78**	**-37.59%**	**154.6**	**31.89%**
	Utilization SLR (%)	**168**	**-143.48%**	**408**	**10.13%**				

	Latency (Clock Cycles)			
	Latency			
	Min	Change	Max	Change
Reuse=1	**23**	**78.10%**	**23**	**78.10%**
Reuse=7	**32**	**69.52%**	**32**	**69.52%**
Reuse=21	**63**	**40.00%**	**63**	**40.00%**

TIMING & RESOURCE USAGE (4 TRACKS, 4 LAYERS)

Virtex FPGA: xcvu13p-fhga2104-1-i : Pipeline Architecture

GNN Resources Usage for Pipeline Architecture		Device: xcvu13p-fhga2104-1-i			
		Vivado HLS C Synthesis		Vivado Synthesis	
		DSP48E	LUT	DSP48E	CLB LUT
	Available	12288	1728000	12288	1728000
	Available SLR	na	na	na	na
Reuse=1	Total(Used)	**17616**	**1790783**	**12288**	**991030**
	Utilization(%)	**143**	**103**	**100**	**57.35**
	Utilization SLR (%)	na	na	na	na
Reuse=7	Total(Used)	**5664**	**1380016**	**12272**	**654134**
	Utilization(%)	**46**	**79**	**99.87**	**37.85**
	Utilization SLR (%)	na	na	na	na
Reuse=21	Total(Used)	**4640**	**1355242**	**12272**	**629730**
	Utilization(%)	**37**	**78**	**99.87**	**36.44**
	Utilization SLR (%)	na	na	na	na

	Latency (Clock Cycles)	
	Latency	
	Min	Max
Reuse=1	**21**	**21**
Reuse=7	**29**	**29**
Reuse=21	**61**	**61**