Jets/MET with pileup and machine learning

Nhan Tran
Fermilab

Princeton HL-LHC Trigger Workshop
January 16, 2018

INTRODUCTION AND OUTLINE

Last talk, Giovanni:

Particle Flow - efficient combination of detector information to extract best physics performance

Building of the technology presented by Giovanni…

This talk: more advanced algorithms

- Dealing with pileup
- PUPPI proof-of-concept: jets, MET, jet substructure (?),…
- More sophistication with machine learning and `HLS4ML`

HL-LHC and Pileup

Multiple pp collisions in the same beam crossing

To increase data rate, squeeze beams as much as possible

HL-LHC AND PILEUP

Multiple pp collisions in the same beam crossing
To increase data rate, squeeze beams as much as possible

2016: <PU> ~ 20-50
2017 + Run 3: <PU> ~ 50-80
HL-LHC: 140-200

Need sophisticated techniques to preserve the physics!

PUPPI

PUPPI (PileUp Per Particle Id): based on PF paradigm

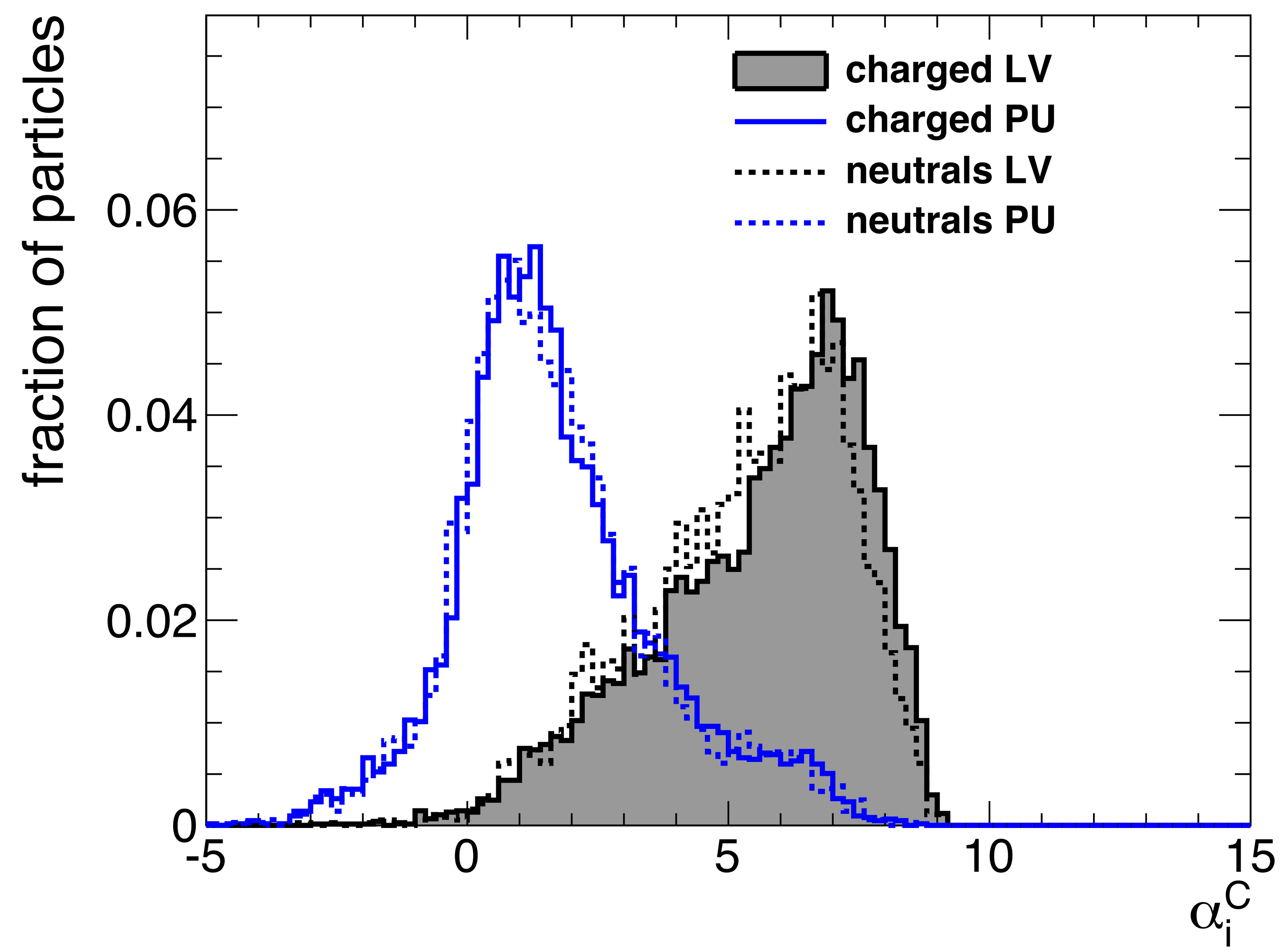
a general framework that determines, per particle, weight for **how likely** a particle is from PU

key insight: using QCD ansatz to infer neutral pileup contribution

[1] define a local discriminant, α, between pileup (PU) and leading vertex (LV)

$$\alpha_i^C = \log\left[\sum_{j\in \mathrm{Ch,LV}} \boxed{\frac{p_{T,j}}{\Delta R_{ij}}}\,\Theta(R_0 - \Delta R_{ij})\right]$$

[2] get data-driven α distribution for PU using charged PU tracks



PUPPI

PUPPI (**P**ile**U**p **P**er **P**article **Id**): based on PF paradigm

a general framework that determines, per particle, weight for **how likely** a particle is from PU

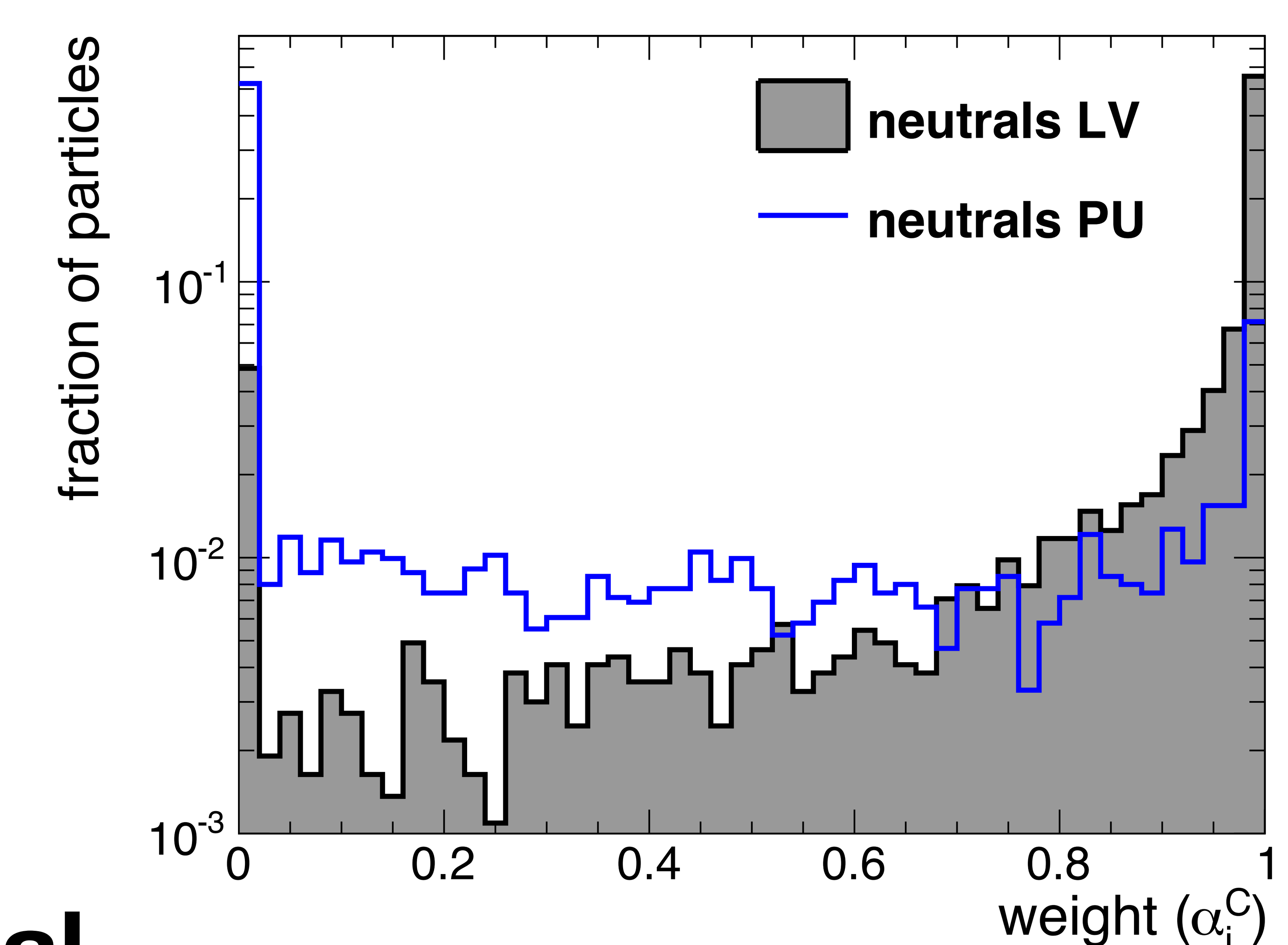
key insight: using QCD ansatz to infer neutral pileup contribution

[1] define a local discriminant, α, between pileup (PU) and leading vertex (LV)

$$\alpha_i^C = \log\left[\sum_{j\in \mathrm{Ch,LV}} \boxed{\frac{p_{T,j}}{\Delta R_{ij}}}\,\Theta(R_0 - \Delta R_{ij})\right]$$

[2] get data-driven α distribution for PU using charged PU tracks

[3] for the neutrals, ask "how un-PU-like is α for this particle?", compute a weight



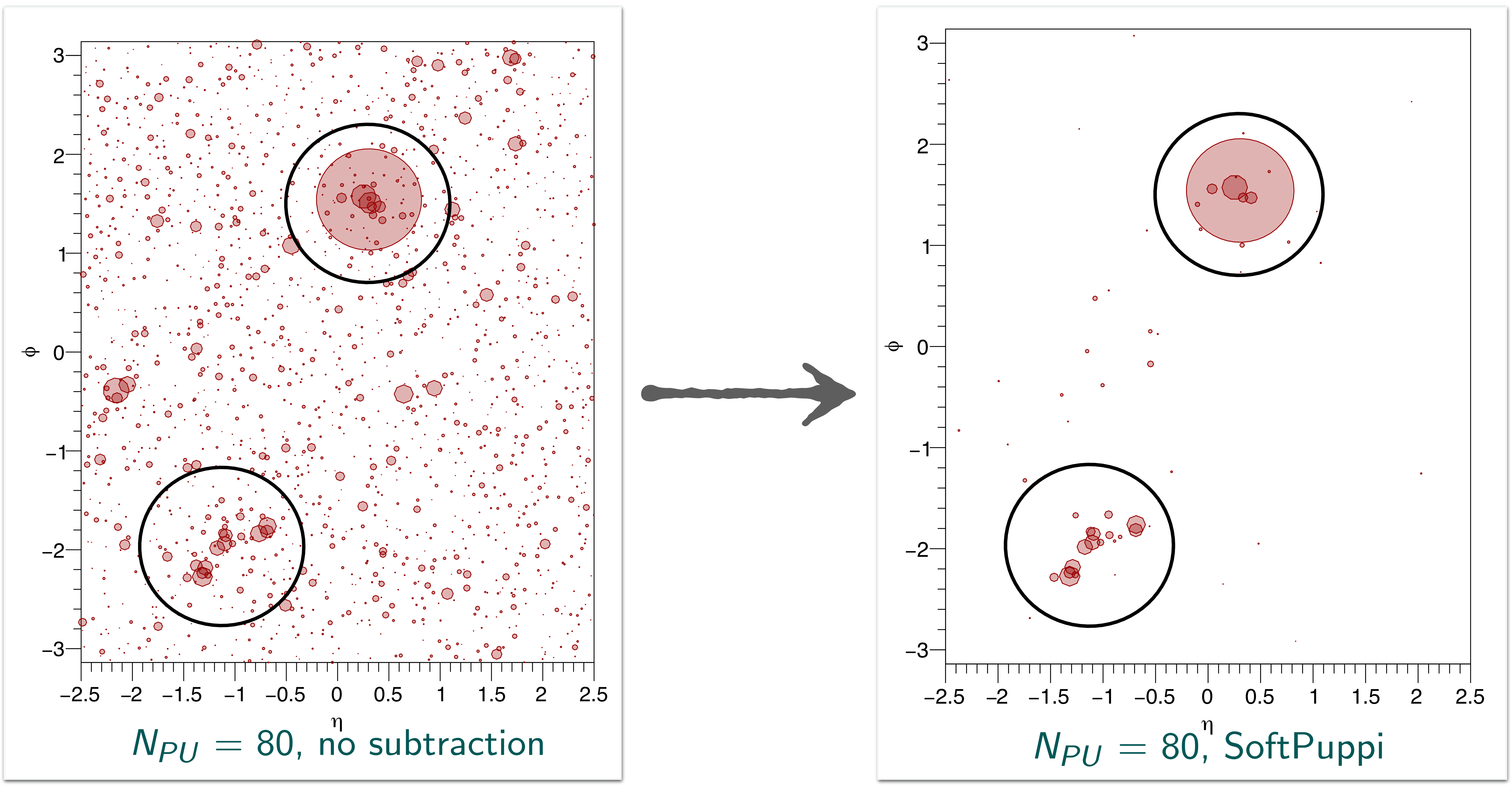
[4] reweight the four-vector of the particle by this weight, then proceed to interpret the event as usual

PUPPI

PUPPI (PileUp Per Particle Id): based on PF paradigm

a general framework that determines, per particle, weight for **how likely** a particle is from PU

key insight: using QCD ansatz to infer neutral pileup contribution

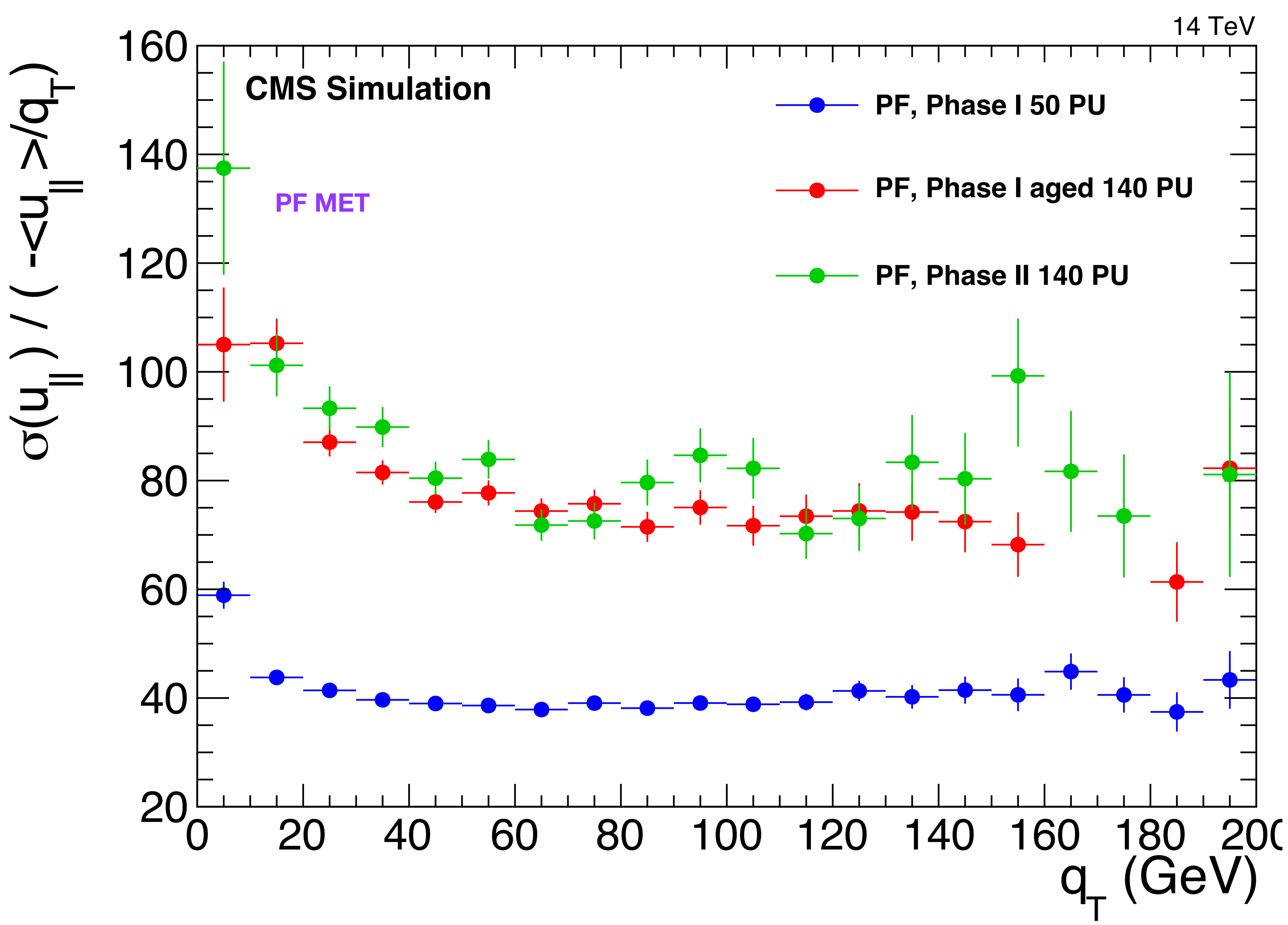


PUPPI IMPACT

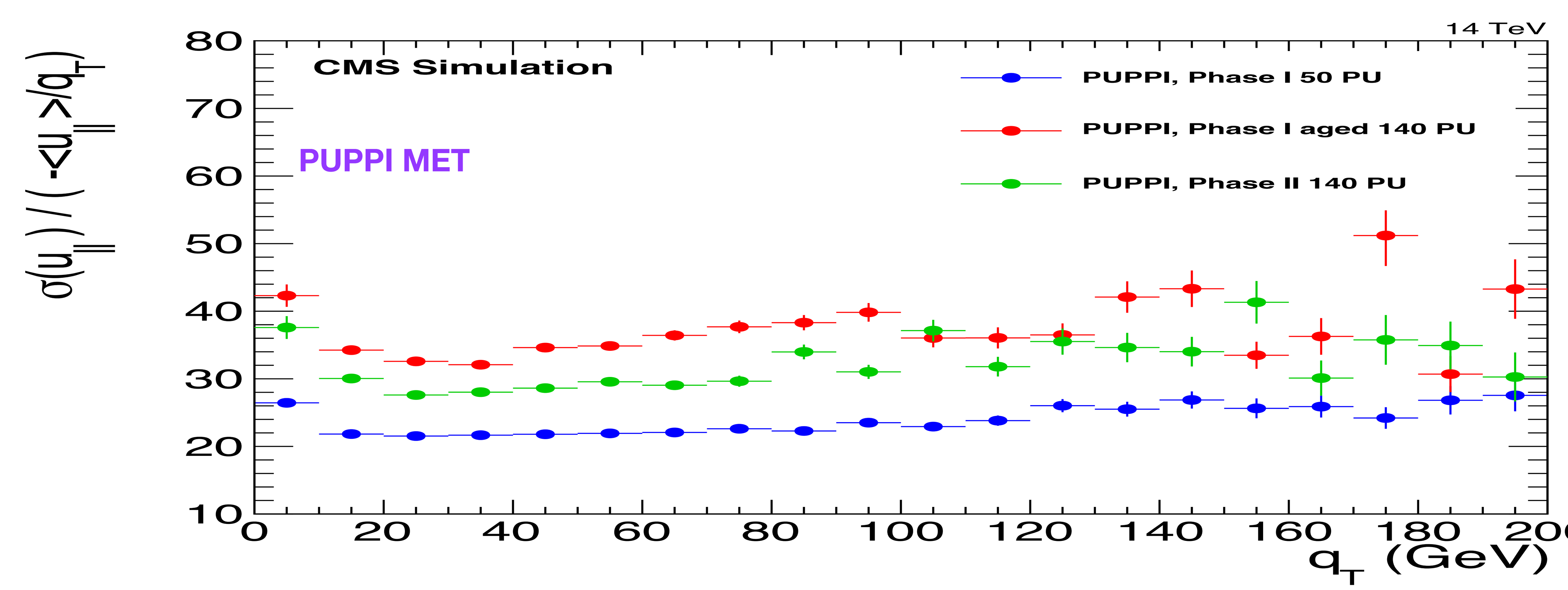
Large gains from PUPPI, especially at high PU

arXiv:1407.6013 (original), LHCC-P-008 (CMS TP), JME-14-001, CMS analyses,…

MET resolution, 140 PU
[LHCC-P-008]



many gains at high PU
jet pT resolution
fake jet rate
MET resolution
jet substructure
lepton isolation
…



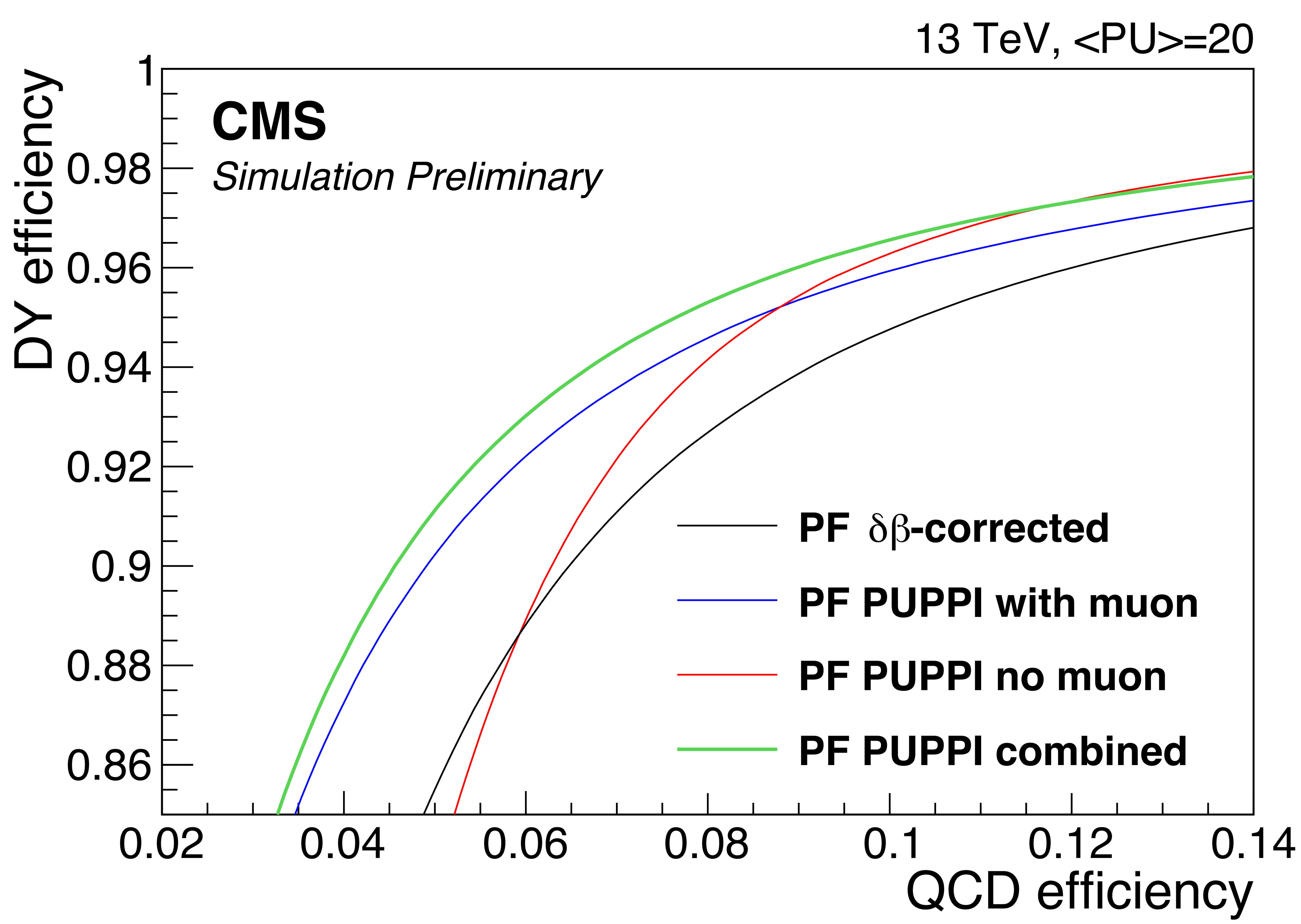
BEYOND JETS AND MET

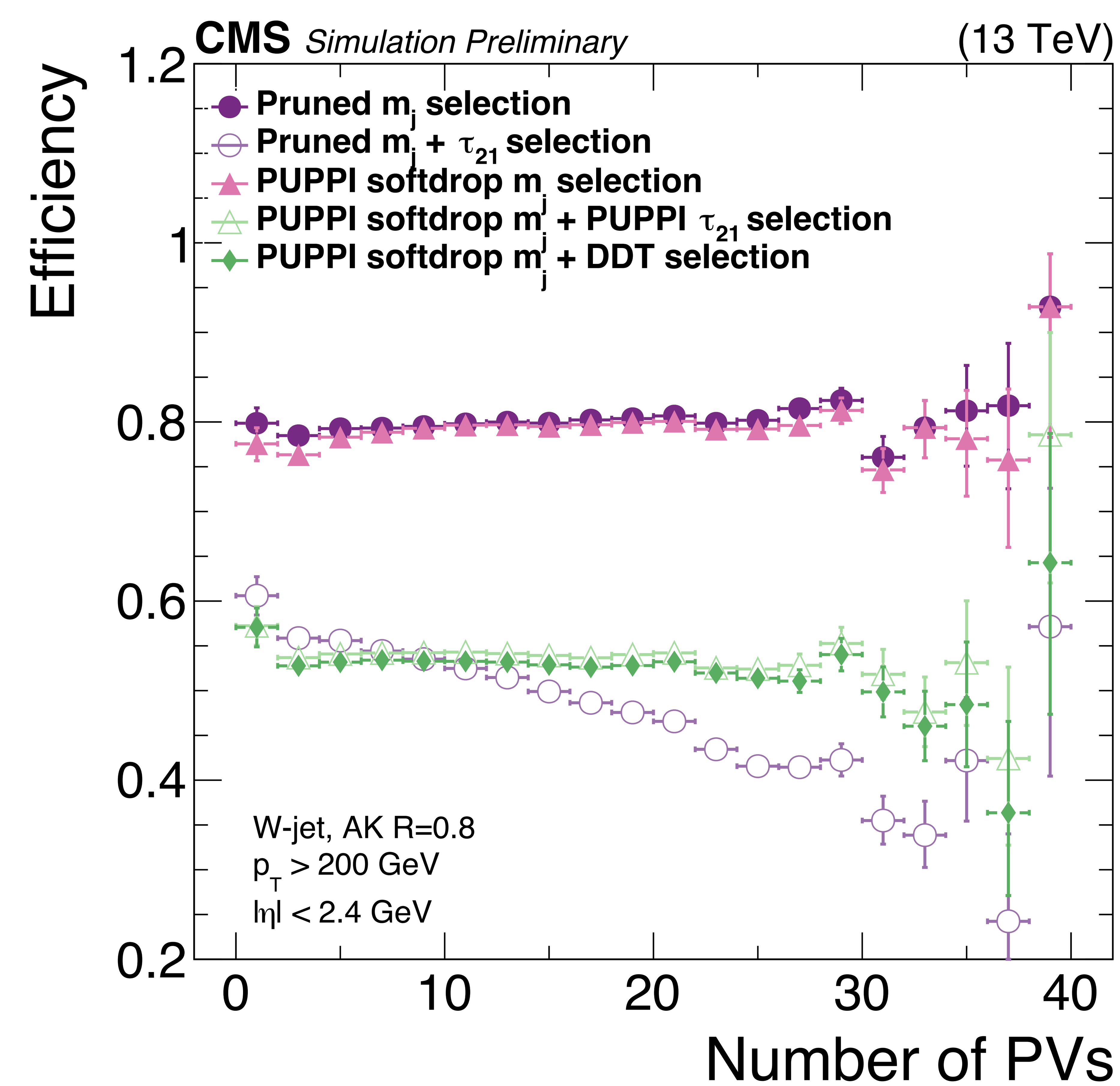
Trying to preserve soft, hidden physics

Things hidden in jets and jet substructure

Isolated, soft leptons in high PU environments

** Examples plots from offline studies*





Large gains is soft muon backgrounds and jet substructure

IMPLEMENTATION

Implementation of Puppi proof-of-concept using High level synthesis (HLS) as well

COMPUTE FOR EACH NEUTRAL

[1] define a local discriminant, α, between pileup (PU) and leading vertex (LV)

$$\alpha_i^C = \log\left[\sum_{j\in \mathrm{Ch,LV}} \boxed{\frac{p_{T,j}}{\Delta R_{ij}}}\Theta(R_0 - \Delta R_{ij})\right]$$

IMPLEMENTATION

Implementation of Puppi proof-of-concept using High level synthesis (HLS) as well

COMPUTE FOR EACH NEUTRAL

[1] define a local discriminant, α, between pileup (PU) and leading vertex (LV)

$$\alpha_i^C = \log\left[\sum_{j\in\mathrm{Ch,LV}} \boxed{\frac{p_{T,j}}{\Delta R_{ij}}}\,\Theta(R_0 - \Delta R_{ij})\right]$$

[2] get data-driven α distribution for PU using charged PU tracks

[3] for the neutrals, ask "how un-PU-like is α for this particle?", compute a weight

[4] reweight the four-vector of the particle by this weight, then proceed to interpret the event as usual

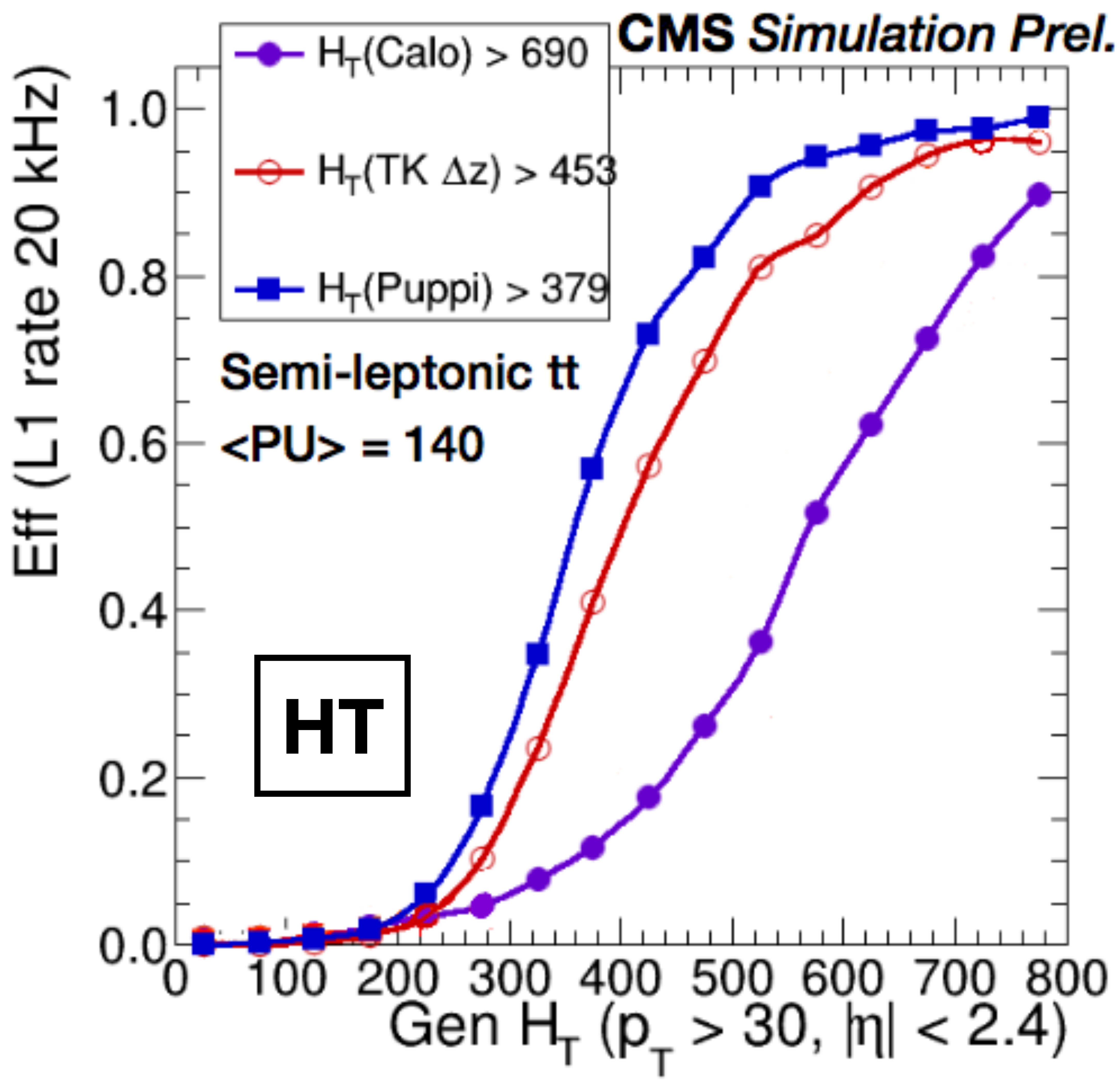
PRECOMPUTE STEP 2 OFFLINE WITH CONSTANTS (FOR GIVEN PILEUP LEVEL)

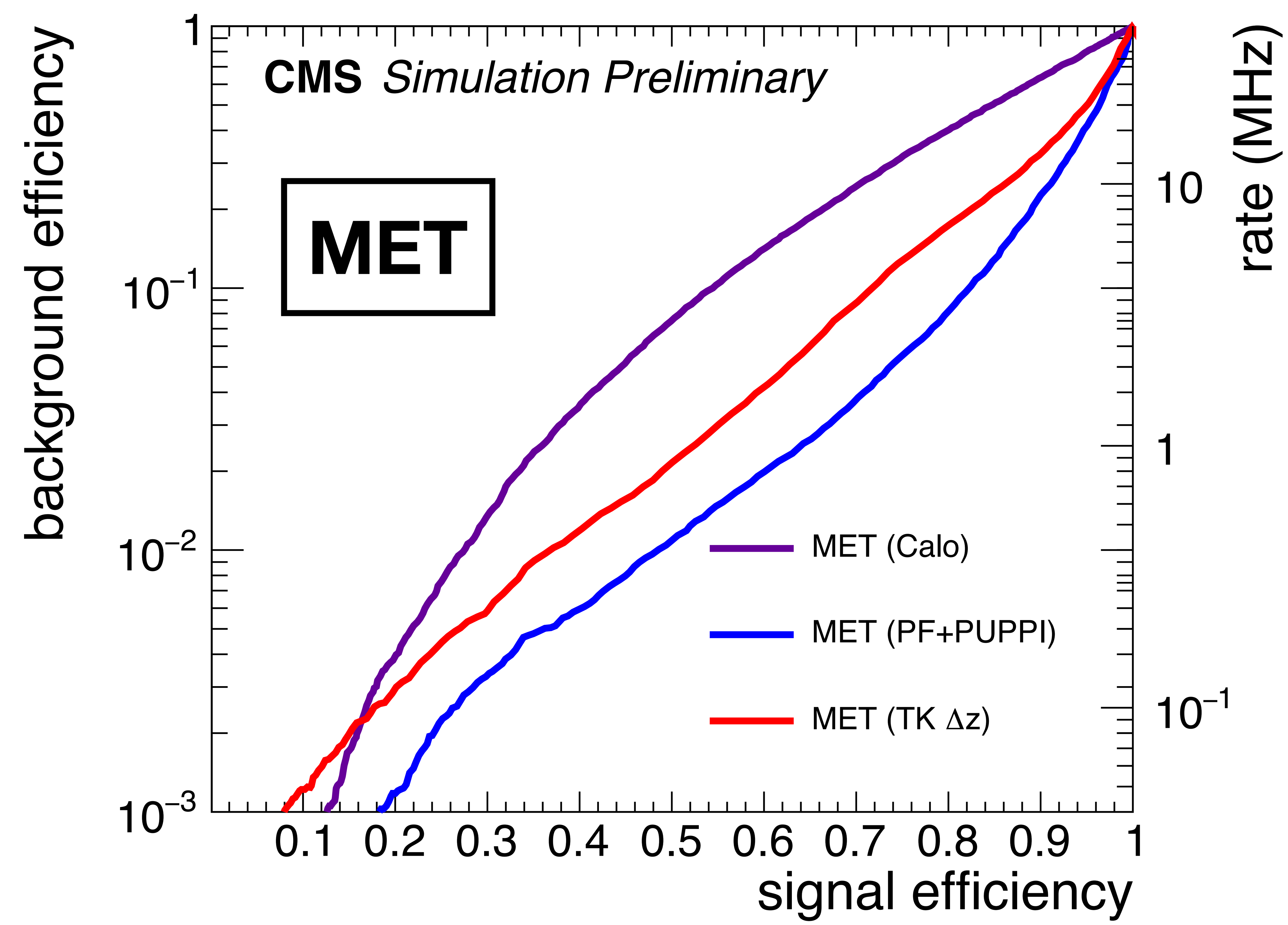
DO STEP 3/4 WITH A LOOK-UP TABLE

RESOURCE USAGE ONLY FEW % OF FPGA AND 100S OF NS LATENCY WITH LITTLE DEGRADATION IN PERFORMANCE

PERFORMANCE

First physics results on HT and MET triggers for CMS phase-2 trigger interim document



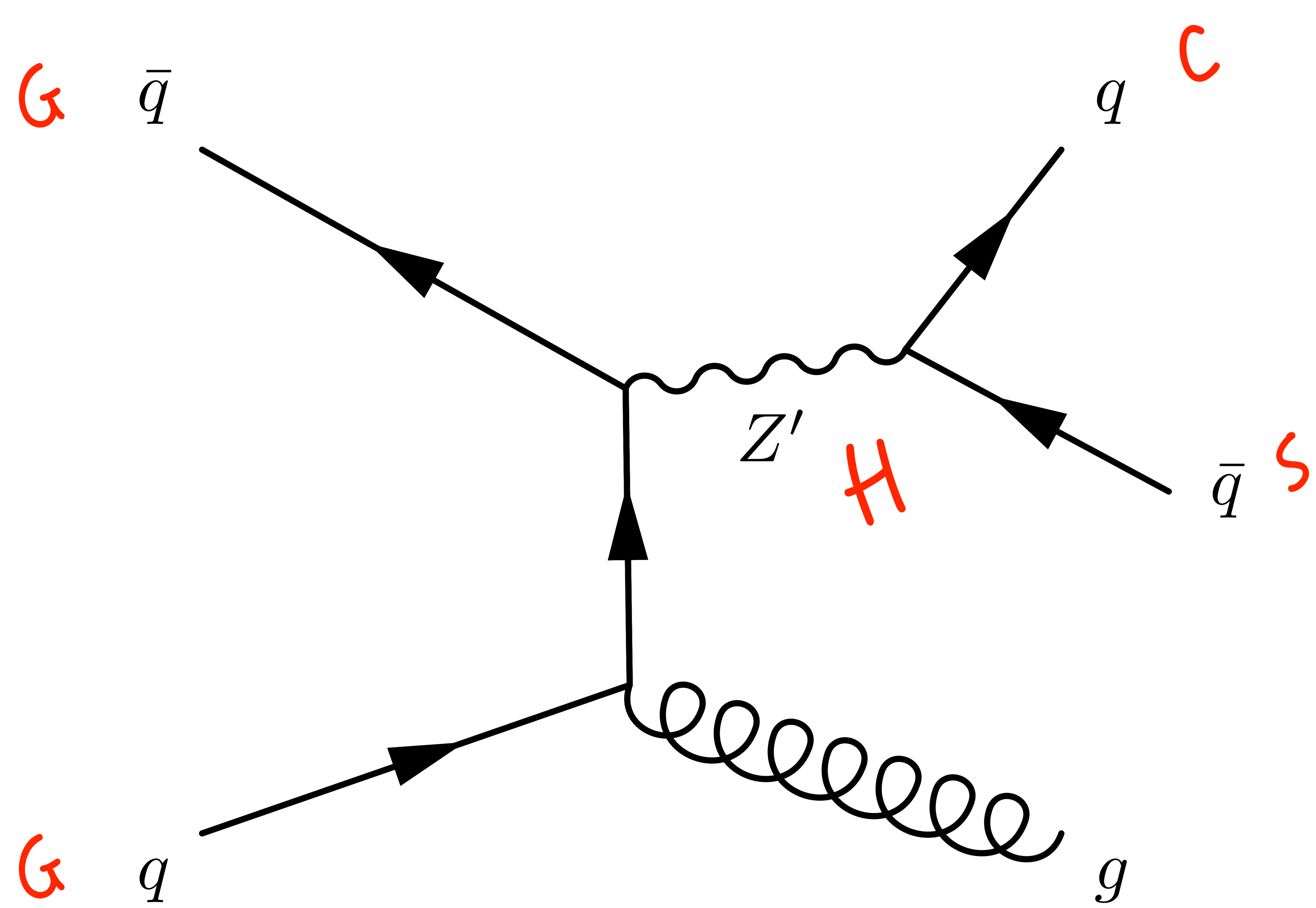


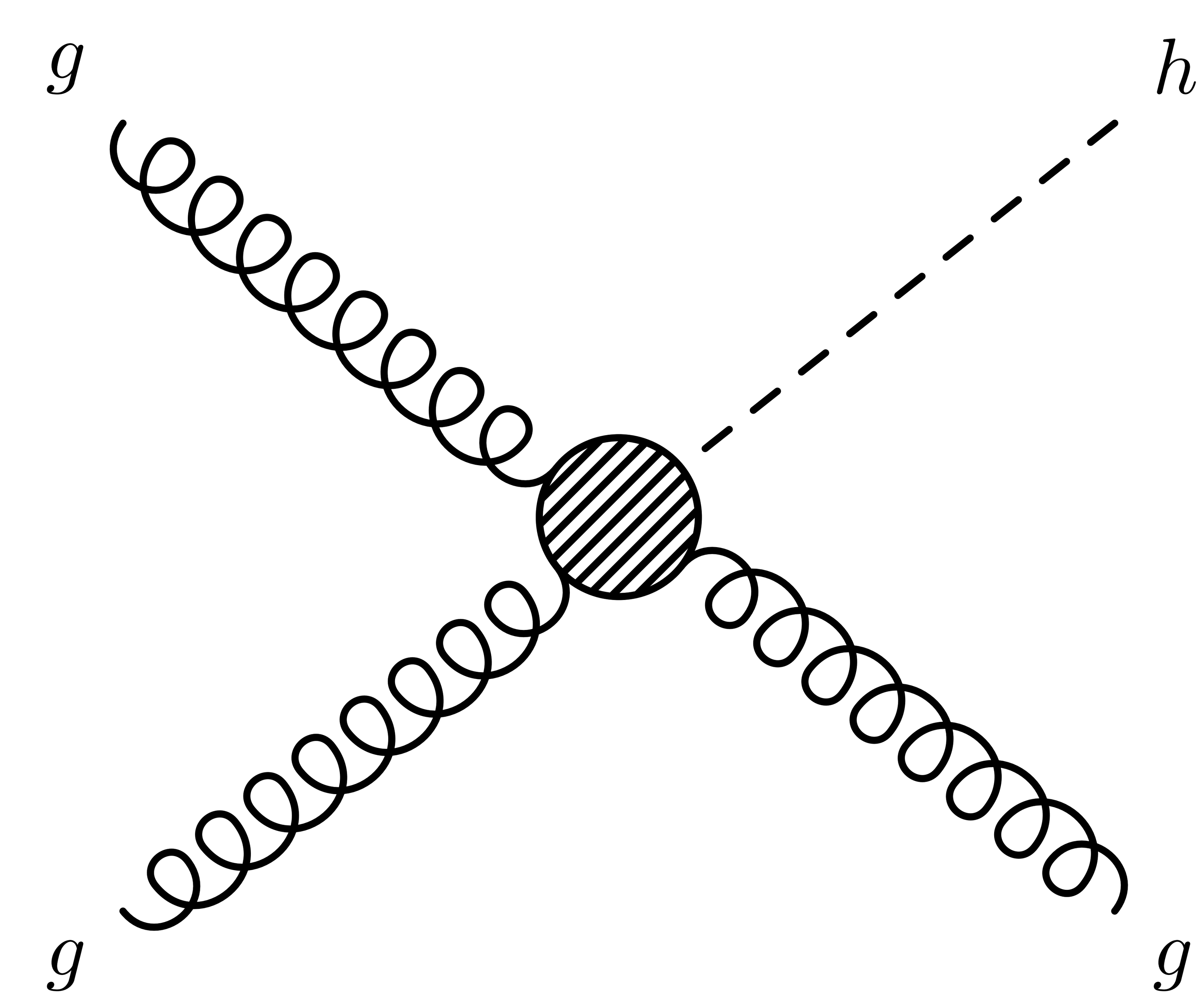
Gains in rate reduction, signal efficiency, lower thresholds

PLANS AND OUTLOOK

Bringing advanced physics algorithms to the hardware trigger!

- Proof-of-concept PF+PUPPI running at L1
- large physics gains: HT, MET, jet (substructure), lepton isolation





TRIGGER ON TOPOLOGIES LIKE THIS AT L1!

Other advanced algorithms…

- how about **machine learning**?

high level synthesis for machine learning

~~HLSFML~~

HLS4ML

Jennifer Ngadiuba, Maurizio Pierini (CERN)
Javier Duarte, Sergo Jindariani, Ben Kreis, Nhan Tran (FNAL)
Phil Harris (MIT)
Zhenbin Wu (UIC)

+ EJ Kreinar (Hawkeye 360) and Song Han (Google/Stanford)

MACHINE LEARNING IN FPGAS

Many parts of the trigger could benefit machine learning

clustering, fitting (regression), classification, anomaly detection

Not just LHC physics or triggering

DAQ, neutrino physics, intensity frontier, …

No industry solutions: LHC latency constraints are unheard of

Why HLS?

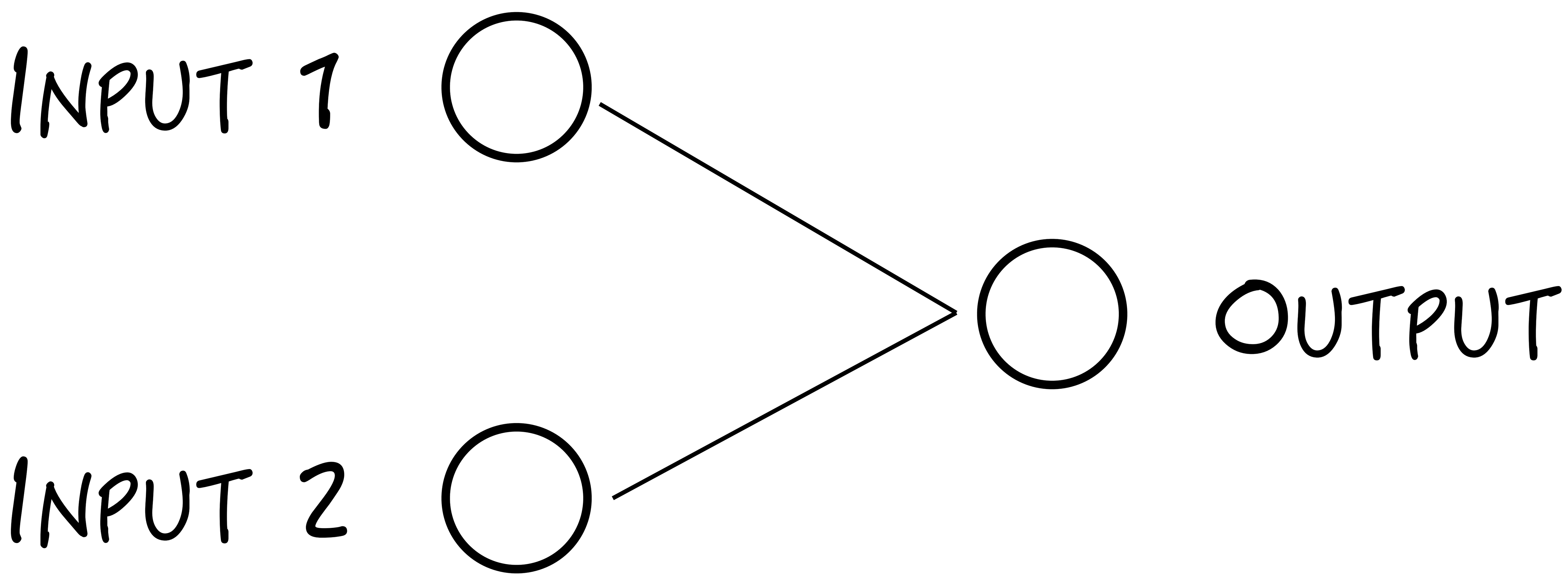
HLS allows (super)-fast algorithm development

Write a tool for machine learning ***inference**** at low latencies:

HLS4ML

* for training, GPUs remain top dog

NN INFERENCE IN A NUTSHELL

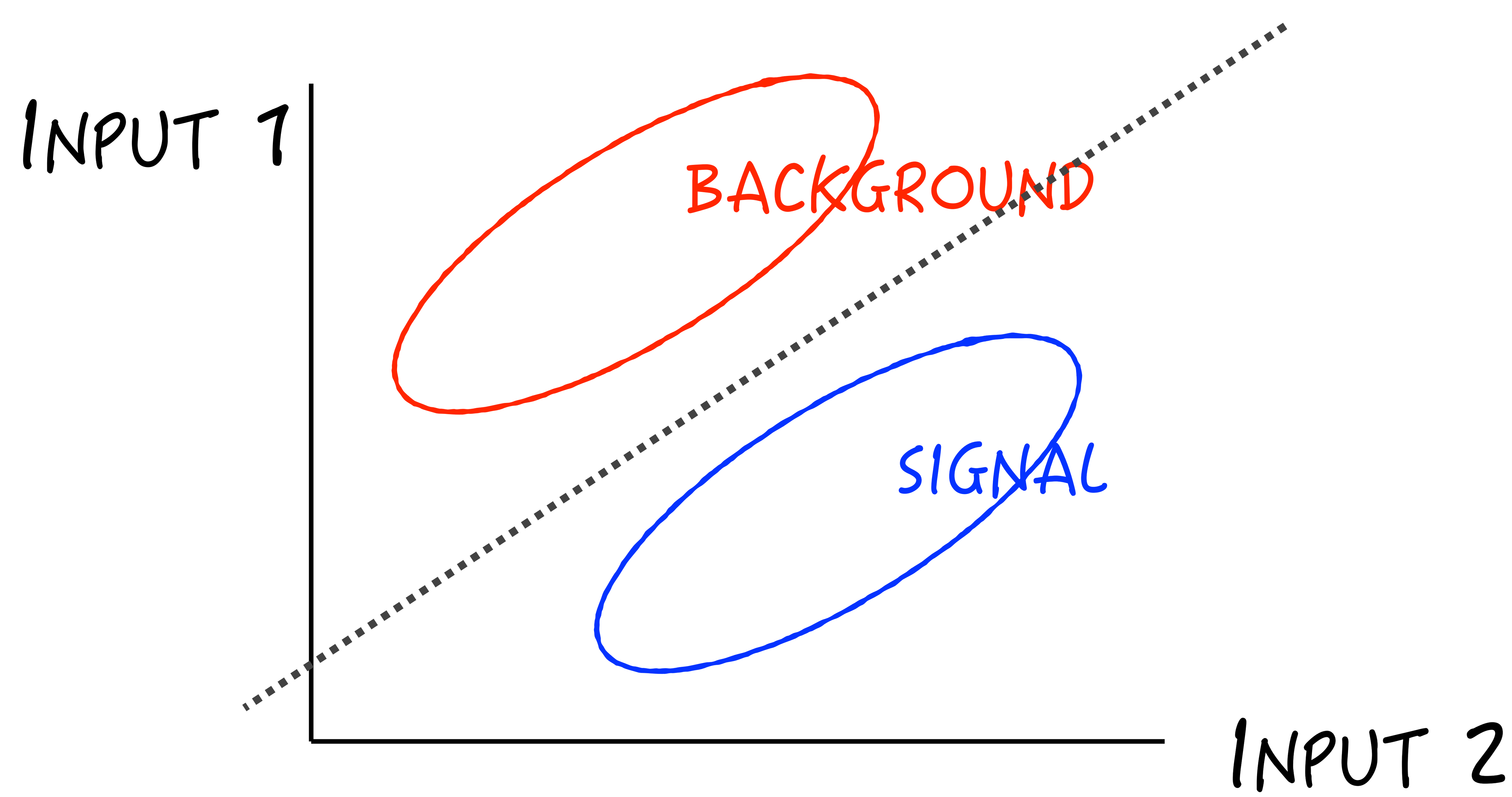


$$\vec{O}_j = \vec{I}_i \times \overleftrightarrow{W}_{ij} + \vec{b}_j$$

Simple 2 input example

(Fisher linear discriminant, linear support vector machine,…)

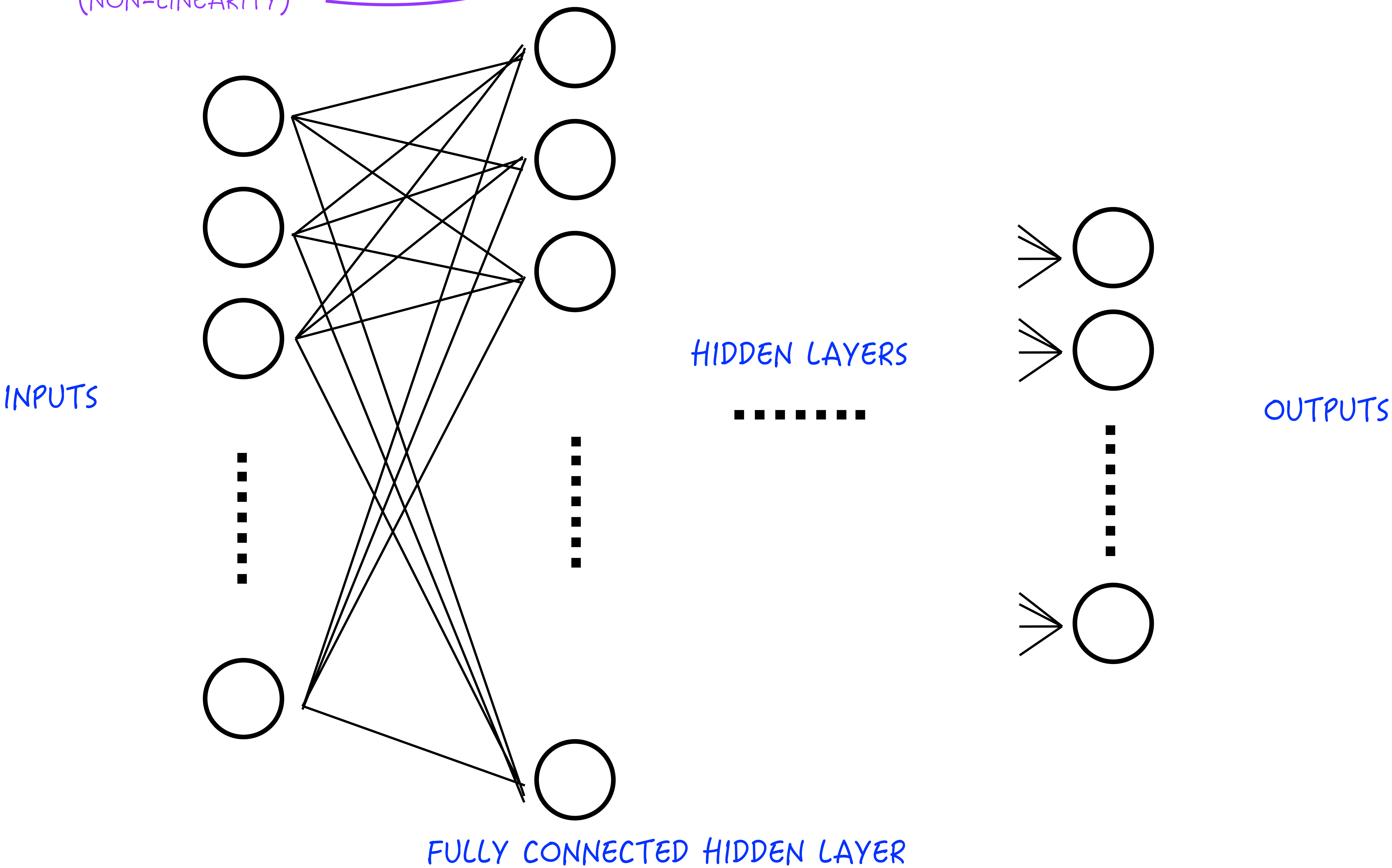
$$O_1 = I_1 \times W_{11} + I_2 \times W_{21} + b_1$$



NN INFERENCE IN A NUTSHELL

$$\vec{O}_j = \phi(\vec{I}_i \times \overleftrightarrow{W}_{ij} + \vec{b}_j)$$

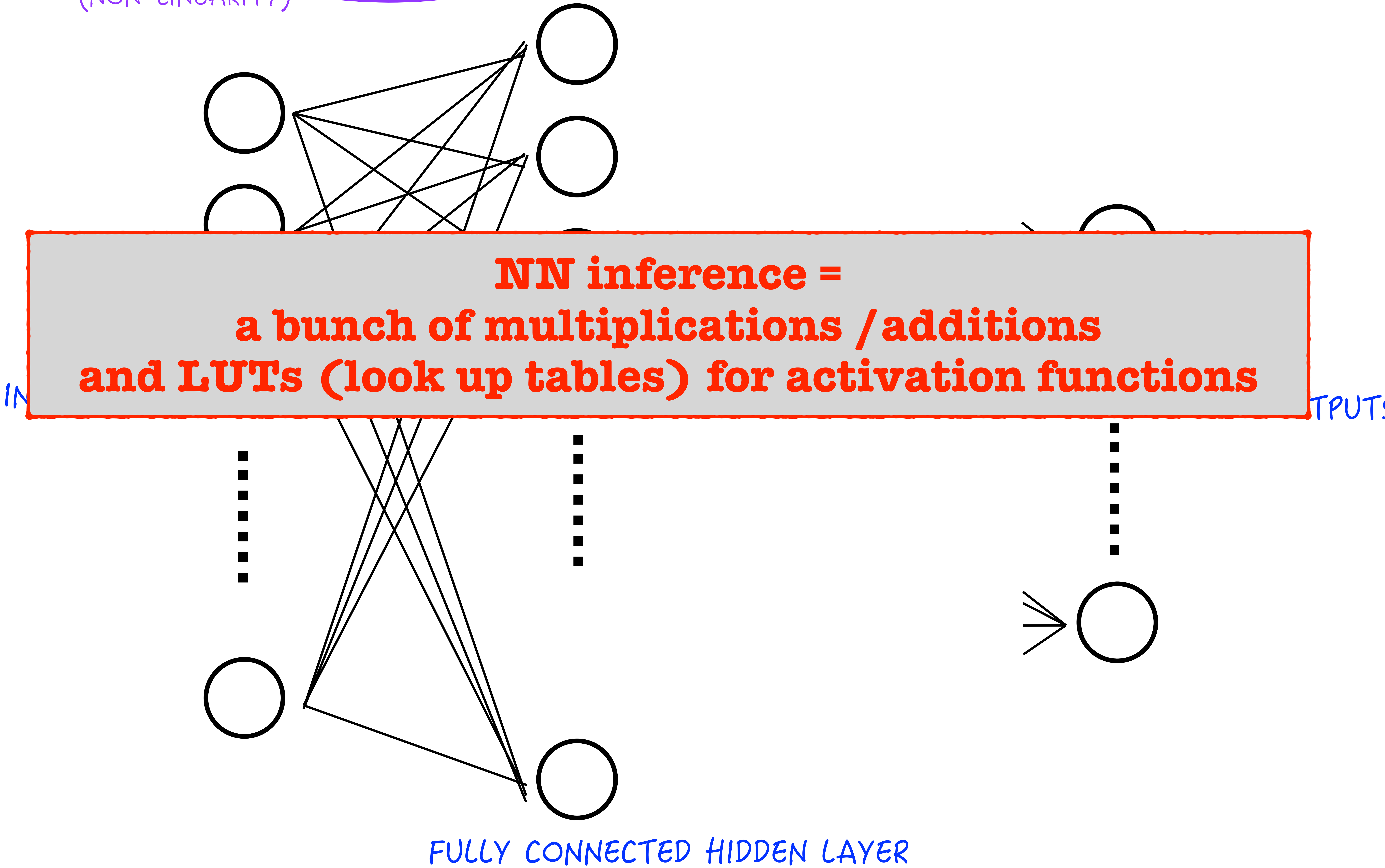
ϕ = ACTIVATION FUNCTION (NON-LINEARITY)



NN INFERENCE IN A NUTSHELL

$$\vec{O}_j = \phi(\vec{I}_i \times \overleftrightarrow{W}_{ij} + \vec{b}_j)$$

ϕ = ACTIVATION FUNCTION (NON-LINEARITY)



NN inference =
a bunch of multiplications /additions
and LUTs (look up tables) for activation functions

(ENERGY) EFFICIENT NEURAL NETWORKS

Emergent engineering field, efficient implementation of NN architecture

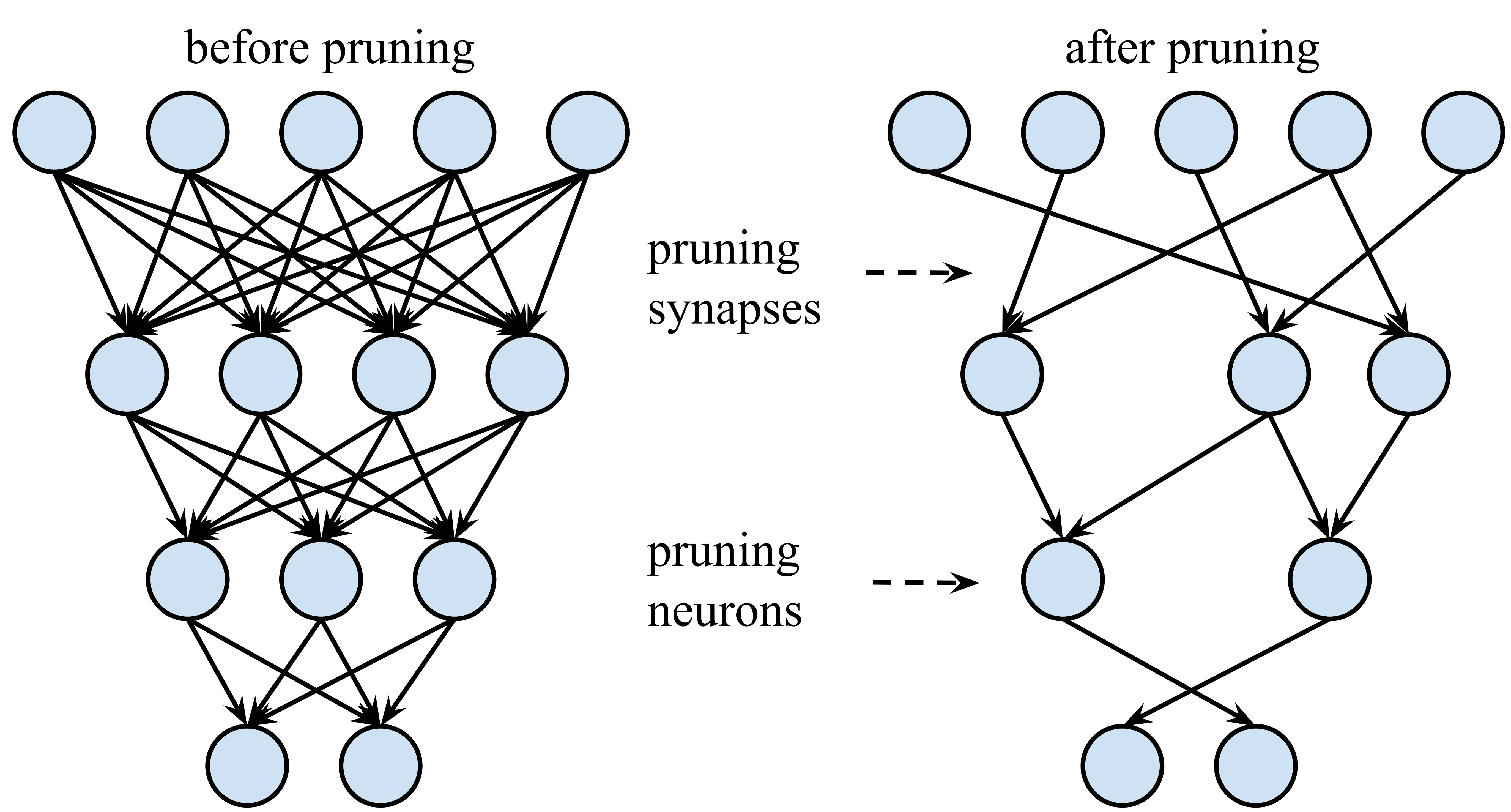
Compression/Pruning:

maintain the same performance while removing low weight synapses and neurons (many schemes)

Quantization/Approximate math:

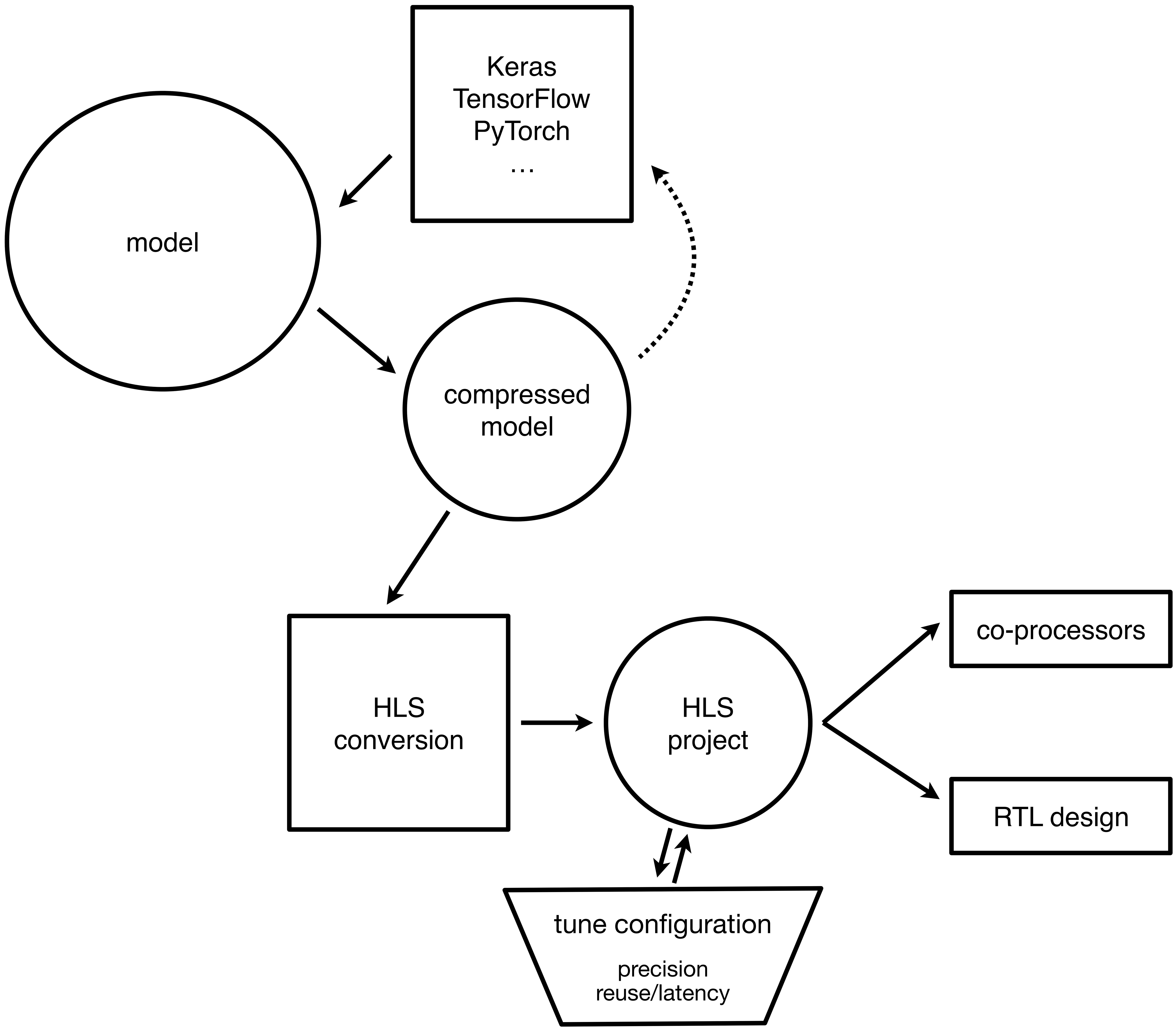
32-bit floating point math is overkill

20-bit, 18-bit, …? fixed point, integers? binarized NNs?

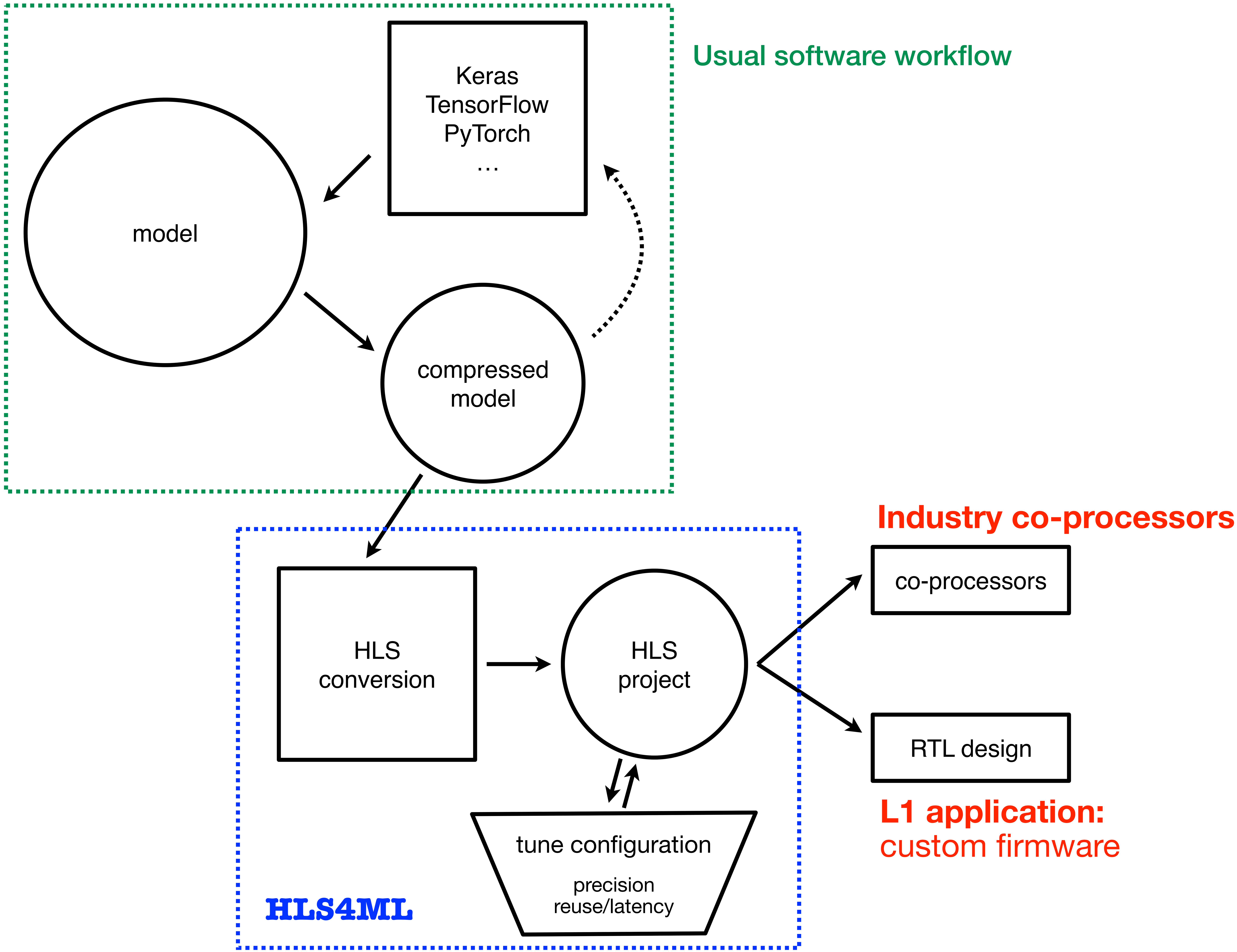


For further reading, start here: https://arxiv.org/pdf/1510.00149v5.pdf

PROJECT OVERVIEW



PROJECT OVERVIEW



HLS4ML - TRANSLATION IN ONE LINE!


```
python keras-to-hls.py -c keras-config.yml
```

```yaml
KerasJson: example-keras-model-files/KERAS_1layer.json
KerasH5:   example-keras-model-files/KERAS_1layer_weights.h5
OutputDir: my-hls-dir-test

IOType: io_parallel # options: io_serial/io_parallel
ReuseFactor: 1
DefaultPrecision: ap_fixed<18,8>
```

Keras/TF inputs

IOType: parallelize or serialize

ReuseFactor: how much to parallelize

DefaultPrecision: self-explanatory :)

EXAMPLE: JET SUBSTRUCTURE

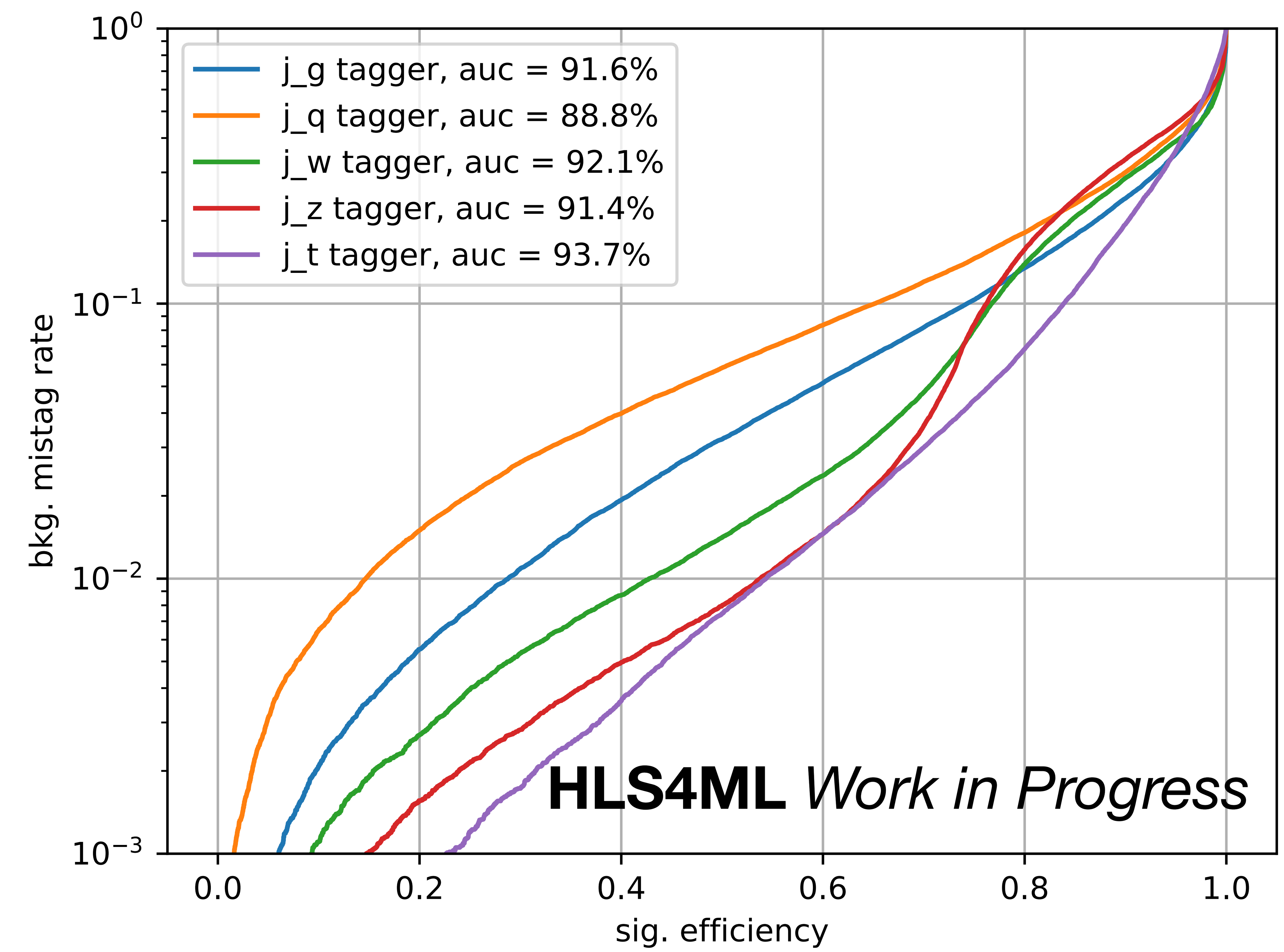
5 output multi-classifier:

Does a jet originate from a quark, gluon, W/Z boson, top quark?

Network architecture

16 expert inputs

- jet masses, multiplicity
- ECFs (β=0,1,2)



16 inputs
64 (relu)
32 (relu)
32 (relu)
5 outputs (softmax)

Fully connected deep neural network

EXAMPLE: NETWORK (NOT JET) PRUNING

Resource usage:

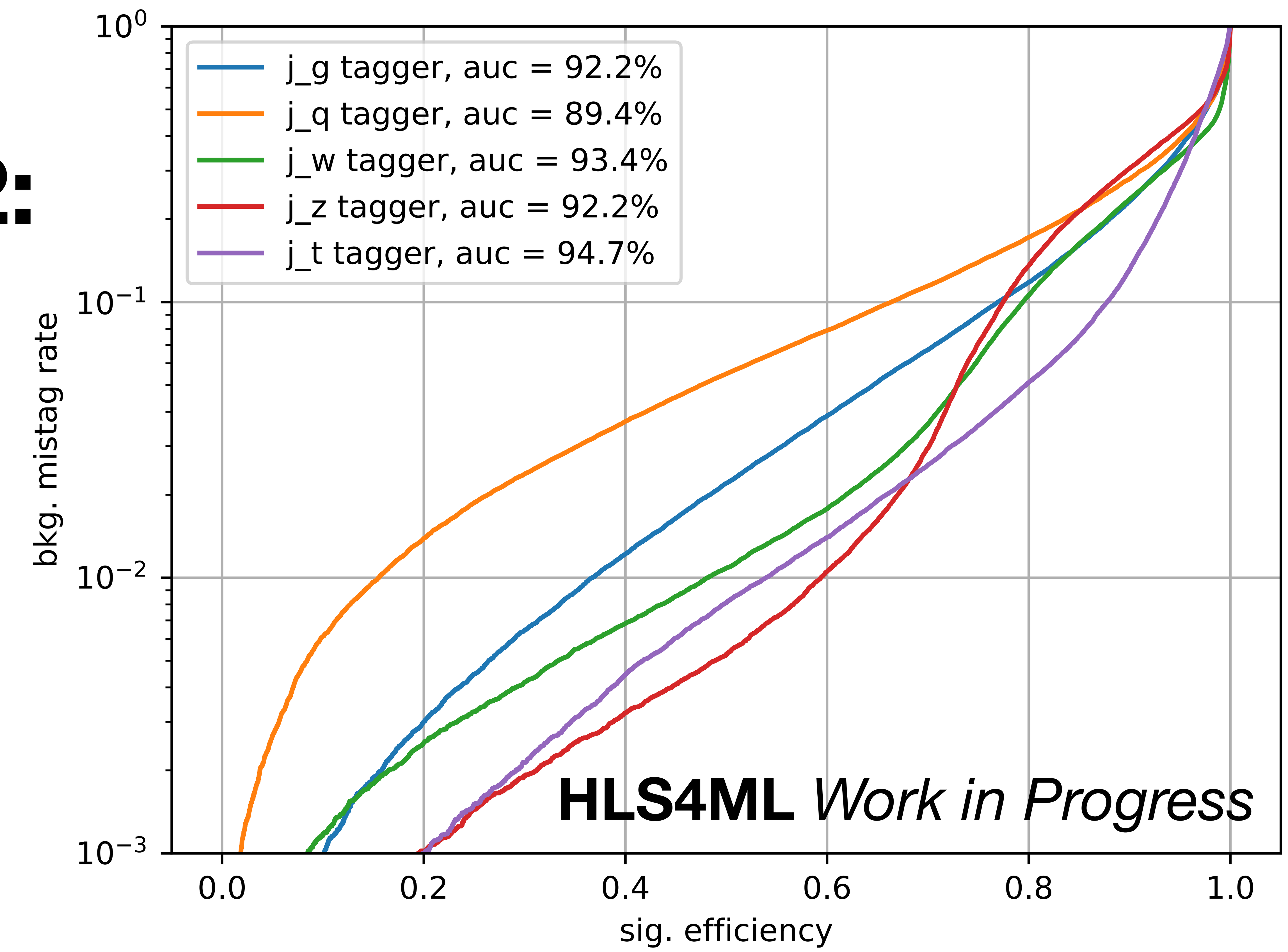
92% DSP usage for Virtex 7

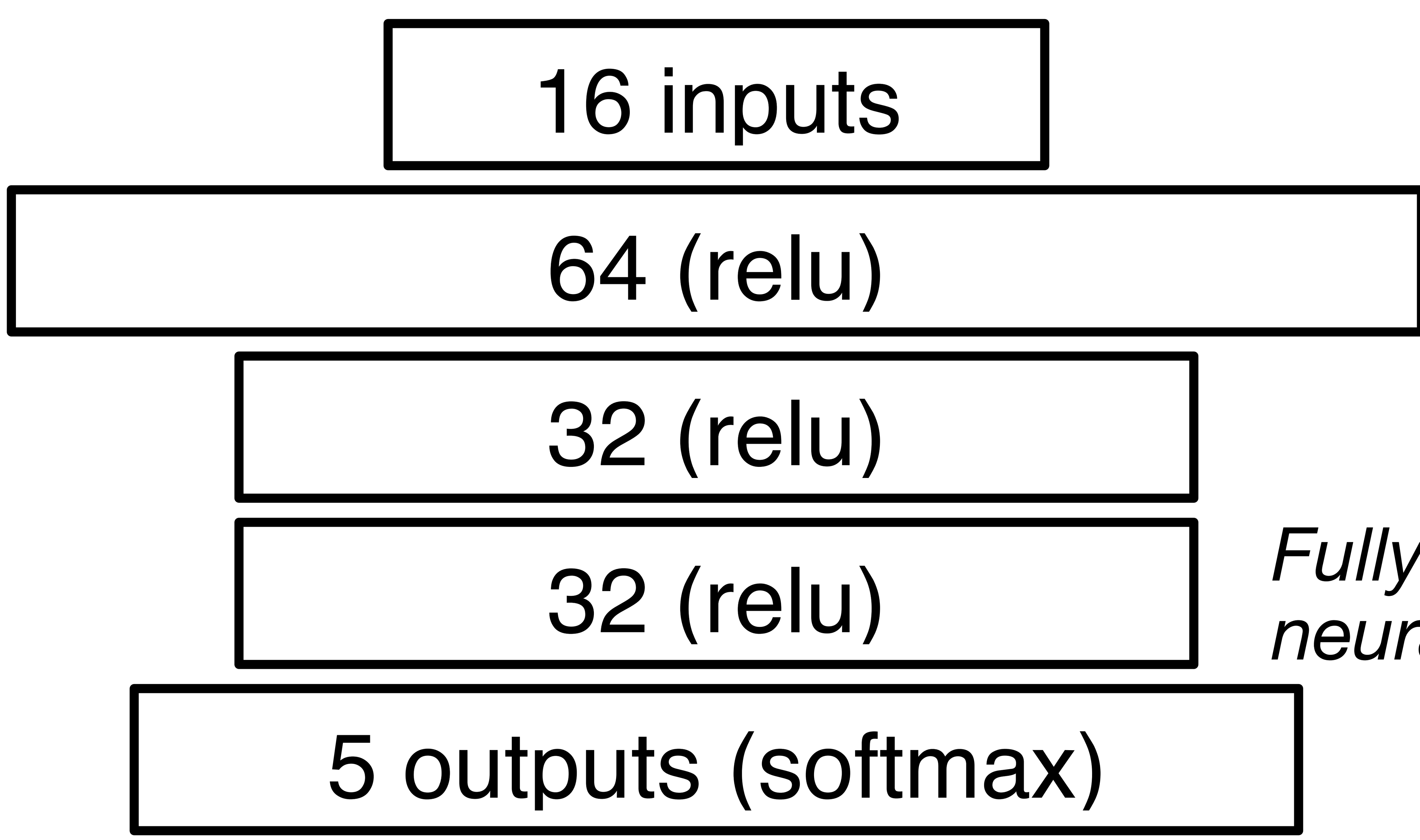
61 clocks (305 ns), Pipeline = 1

Compression (50%) + reuse = 2:

29% DSP usage for Virtex 7

60 clocks (300 ns), Pipeline = 2





Fully connected deep neural network

MINI-SUMMARY

HLS4ML

a tool to translate ML algorithms for FPGAs in minutes

highly parallelizable with user controls for resource usage and latency

tunable precision, resource reuse

very efficient network design with model compression

Work in progress

- Mapping out resource usage and latency as a function of neural network hyper parameters
- More network architectures: CNN (in progress), RNN/LSTM (tricky!), TMVA BDT (efficient?)

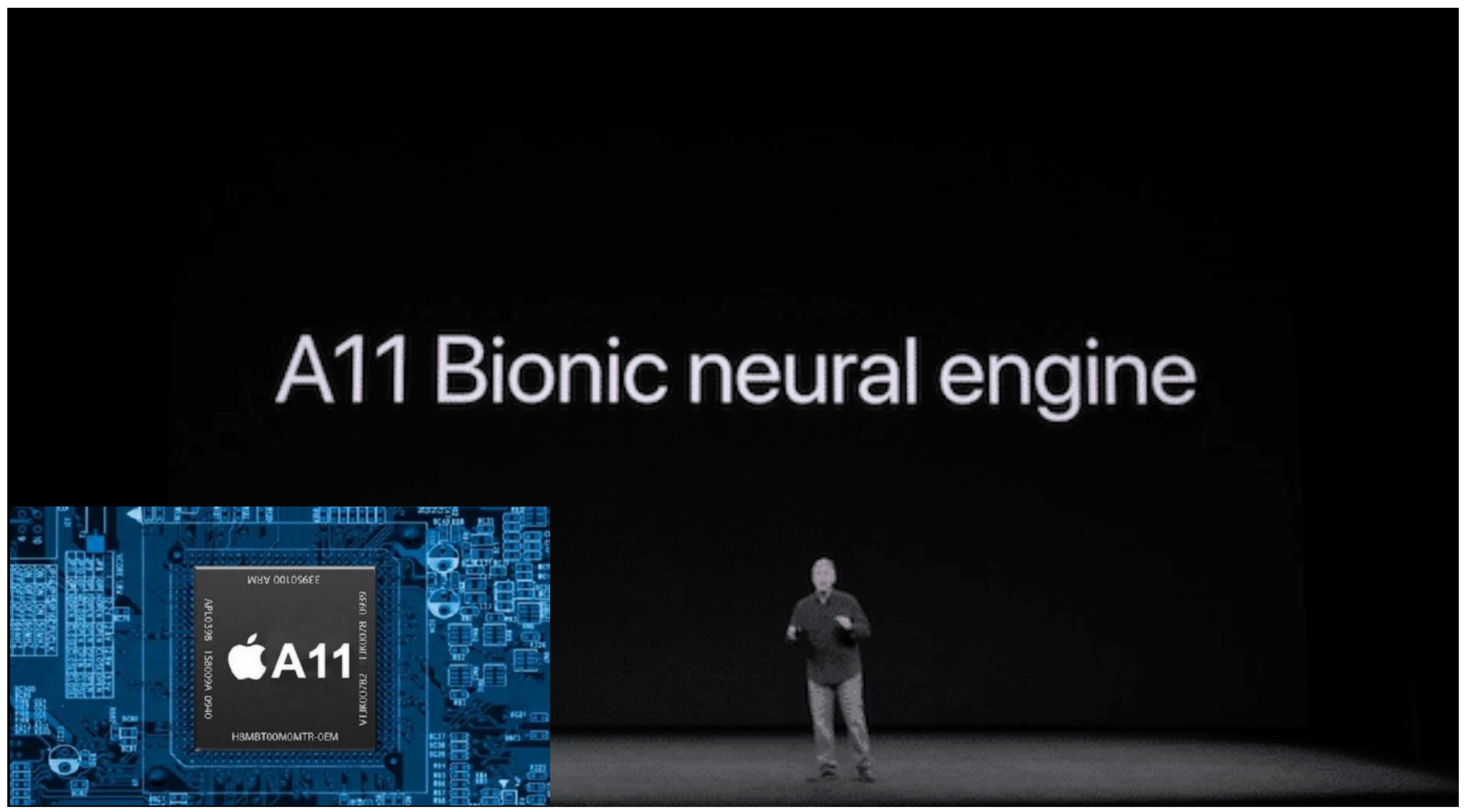
Status

- Alpha version - few weeks; Targeting March-April release of Beta version
- Please contact us if you are interested! hls4ml.help@gmail.com

one more fun thing to think about for the high level trigger (and beyond?)

#TRENDING

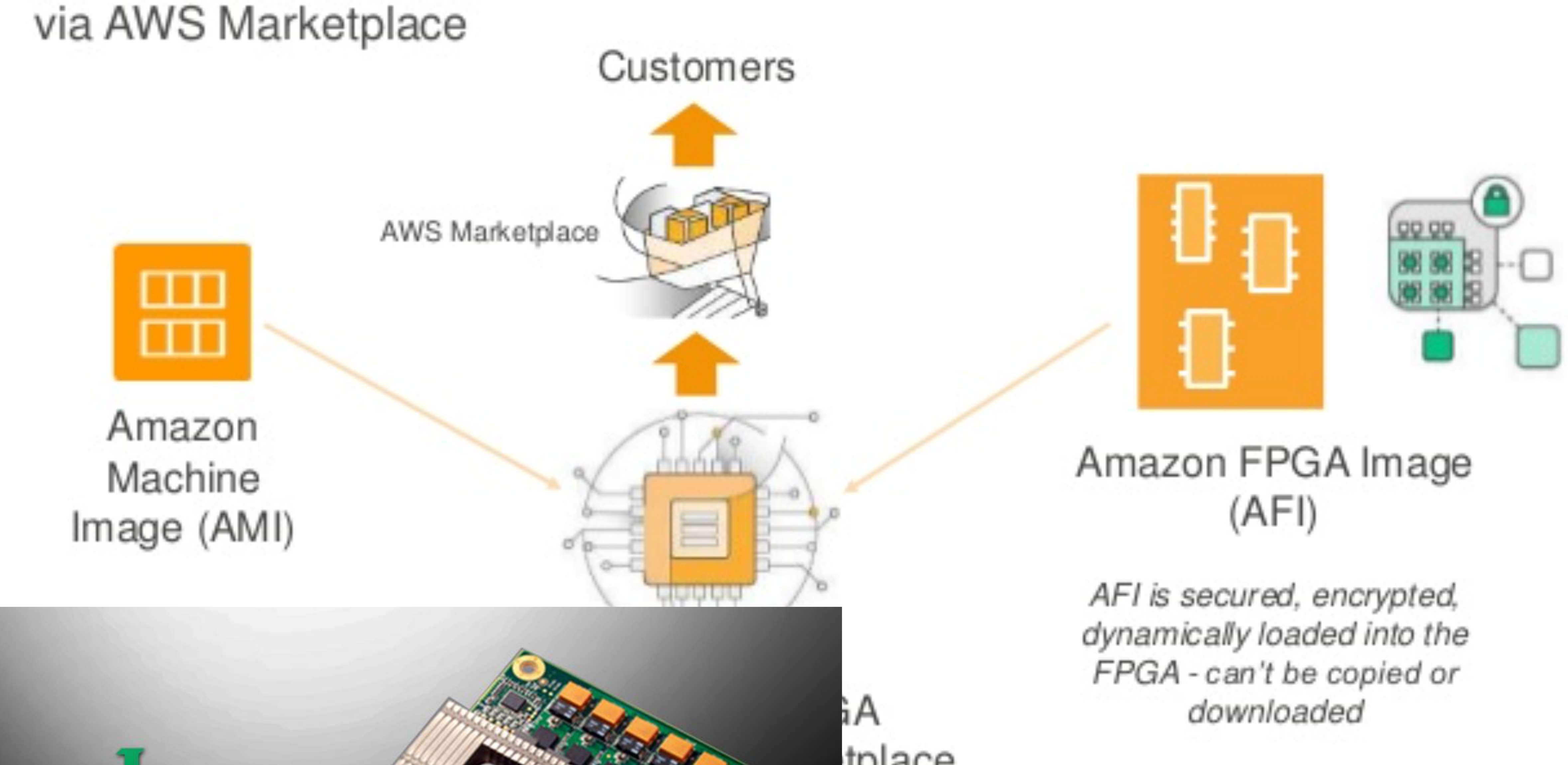
Specialized co-processor hardware for machine learning inference

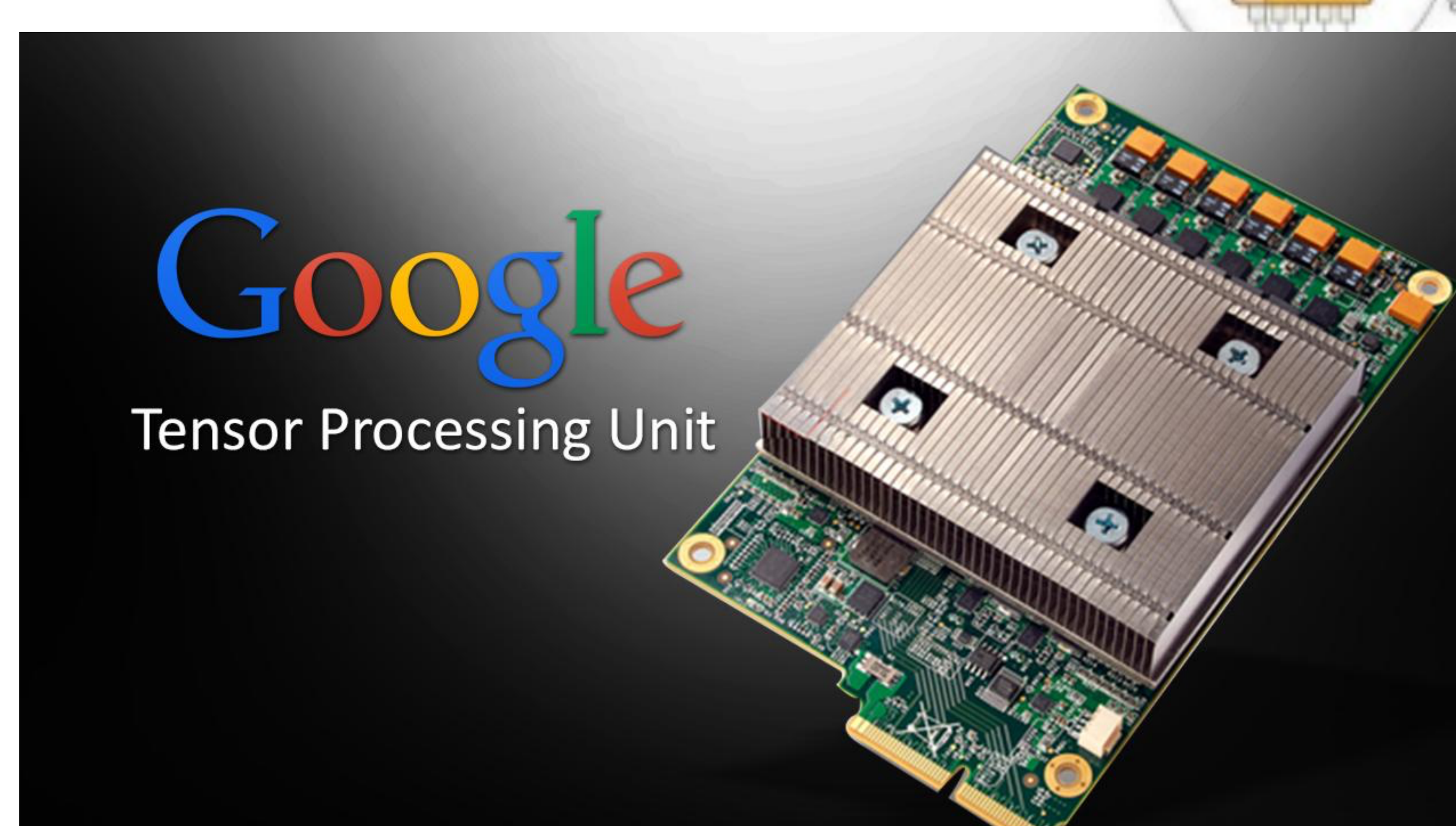


Catapult/Brainwave

Delivering FPGA Partner Solutions on AWS

via AWS Marketplace



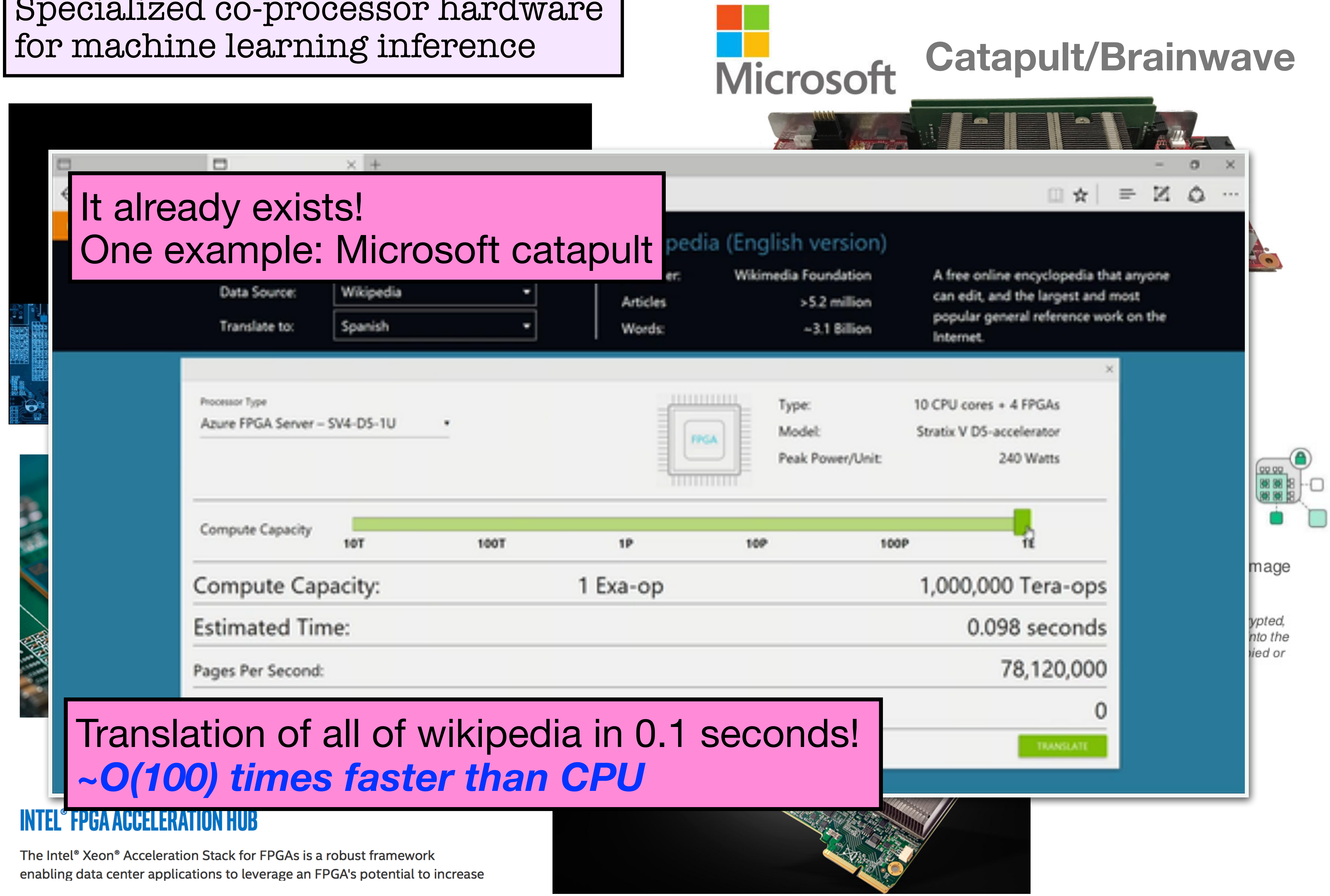


INTEL® FPGA ACCELERATION HUB

The Intel® Xeon® Acceleration Stack for FPGAs is a robust framework enabling data center applications to leverage an FPGA's potential to increase

#TRENDING

Specialized co-processor hardware for machine learning inference



It already exists!
One example: Microsoft catapult

Translation of all of wikipedia in 0.1 seconds!
~O(100) times faster than CPU

ML Babel Fish

Large gains from hardware accelerating co-processors
Industry trending towards specialized computing paradigms

Option 1

re-write physics algorithms for new hardware

Language: OpenCL, OpenMP, HLS, …?

Hardware: FPGA, GPU

PF & PUPPI @ L1 TRIGGER

Option 2

re-cast physics problem as a machine learning problem

Language: C++, Python (TensorFlow, PyTorch,…)

Hardware: FPGA, GPU, ASIC

HLS4ML/ INDUSTRY?

Why (Deep) Machine Learning?

a common *language* for solving problems
which can universally be expressed on
optimized computing hardware and follow industry trends

SUMMARY AND OUTLOOK

Recent advances in hardware and compilation/synthesis allow for sophisticated techniques at low latency

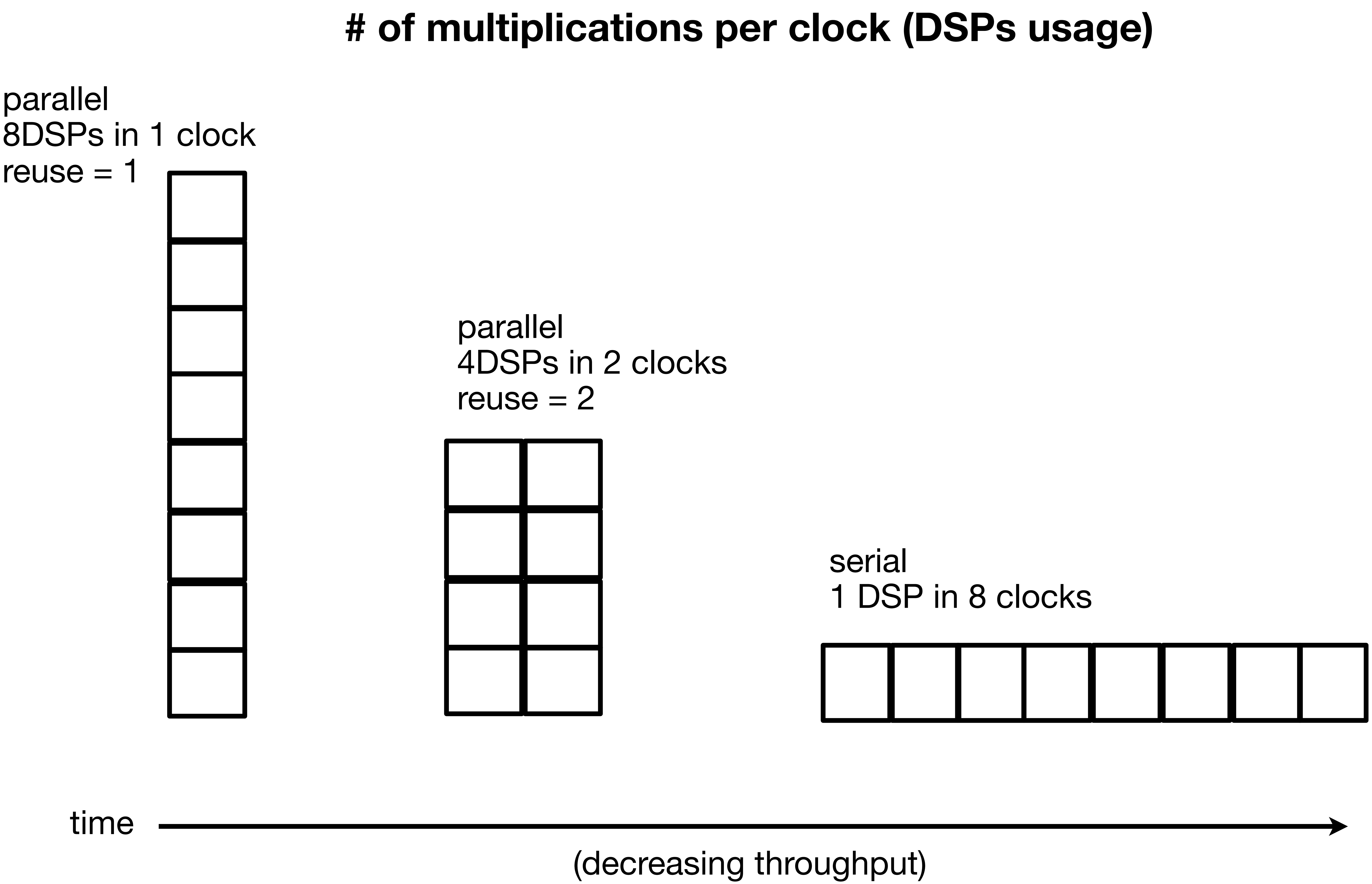
Big improvements in performance,
preserve soft and hidden signatures

Proof-of-concept holistic pileup mitigation techniques such as PUPPI
Efficient machine learning at Level-1 Trigger
New paradigms for HLT and offline?

BONUS

EXAMPLE: PARALLELIZATION

ReuseFactor: how much to parallelize operations a hidden layer

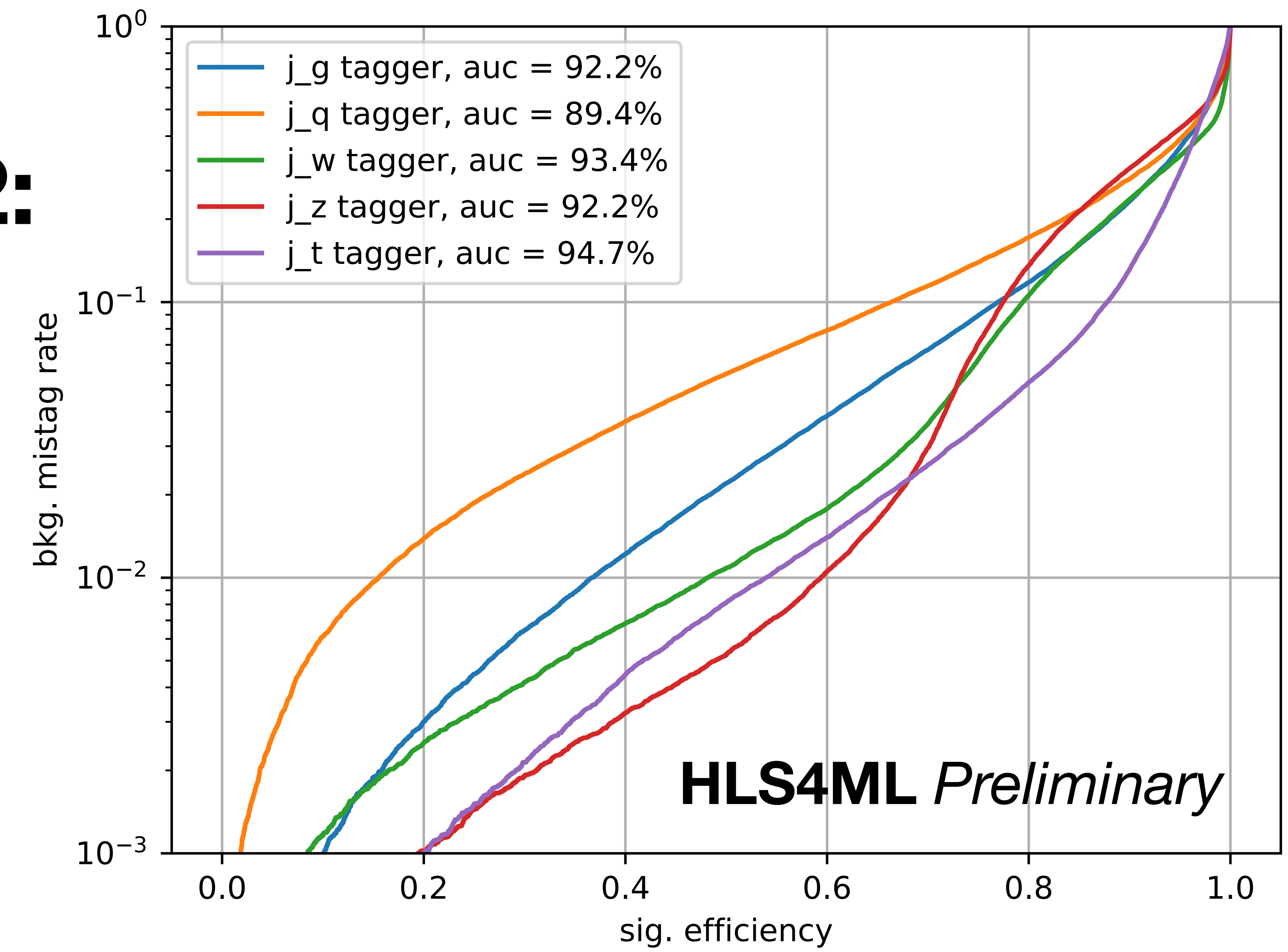


Example: Network (not jet) Pruning

Resource usage:

92% DSP usage for Virtex 7

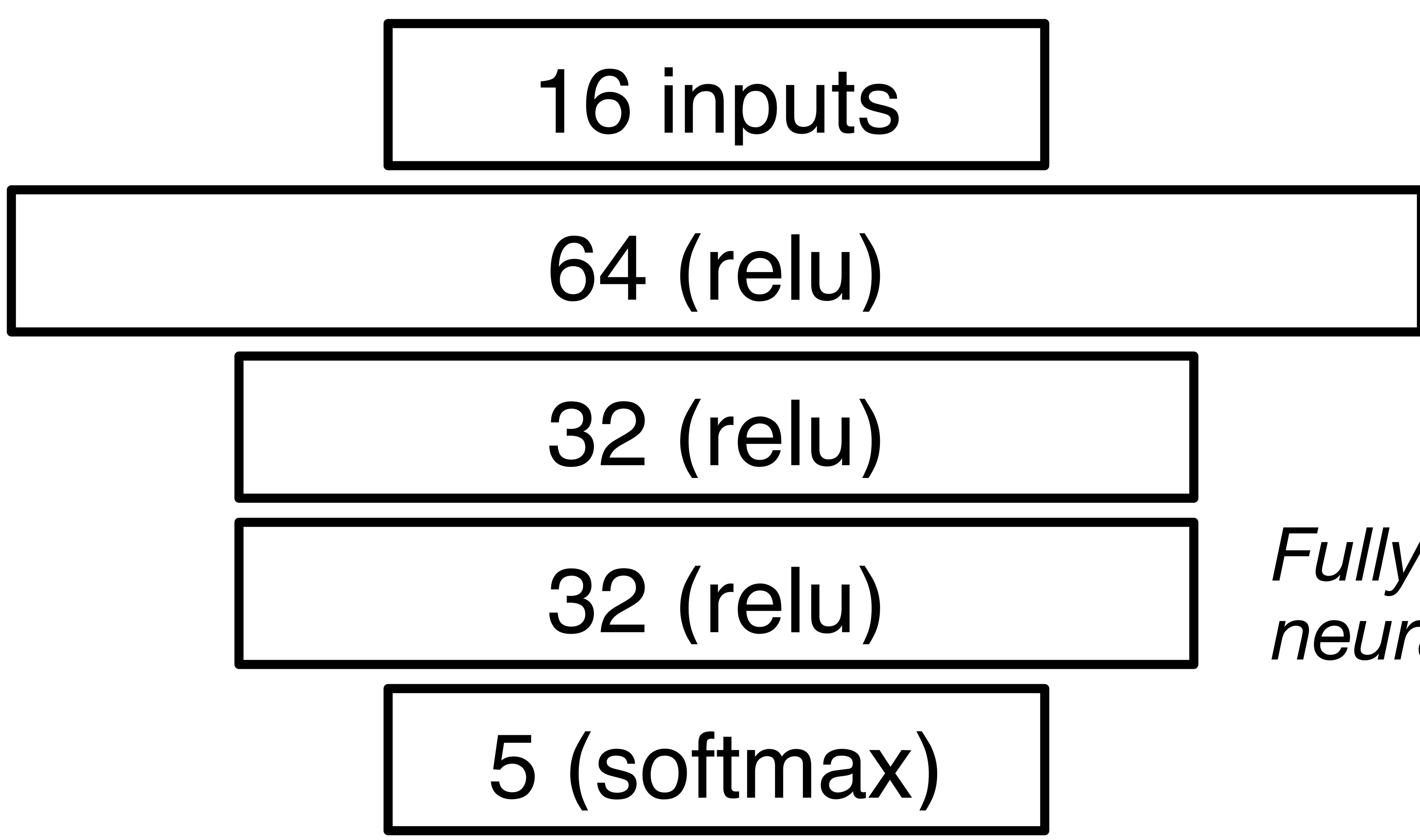
61 clocks (305 ns), Pipeline = 1



Compression (50%) + reuse = 2:

29% DSP usage for Virtex 7

60 clocks (300 ns), Pipeline = 2



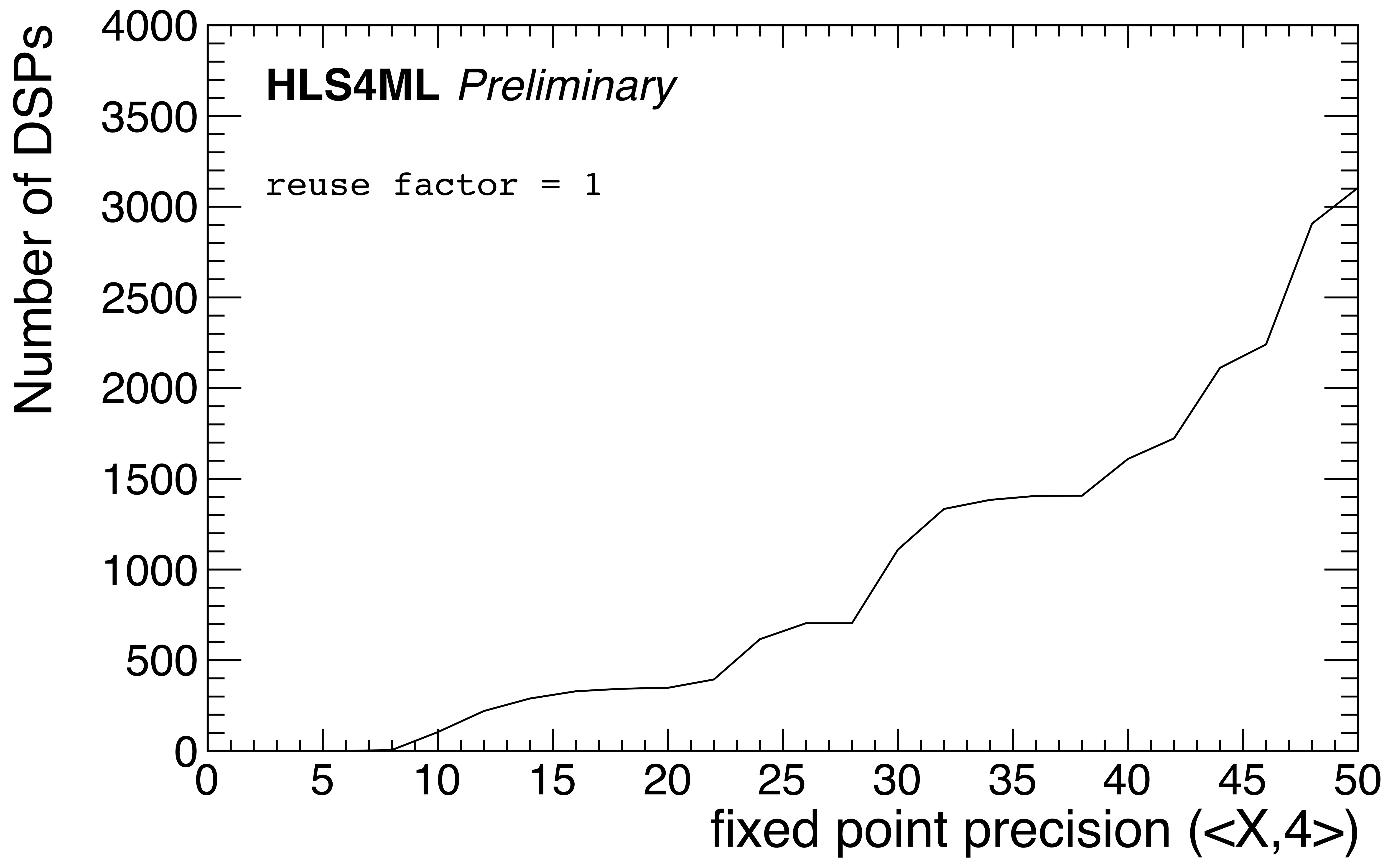
Fully connected deep neural network

EXAMPLE: QUANTIZATION

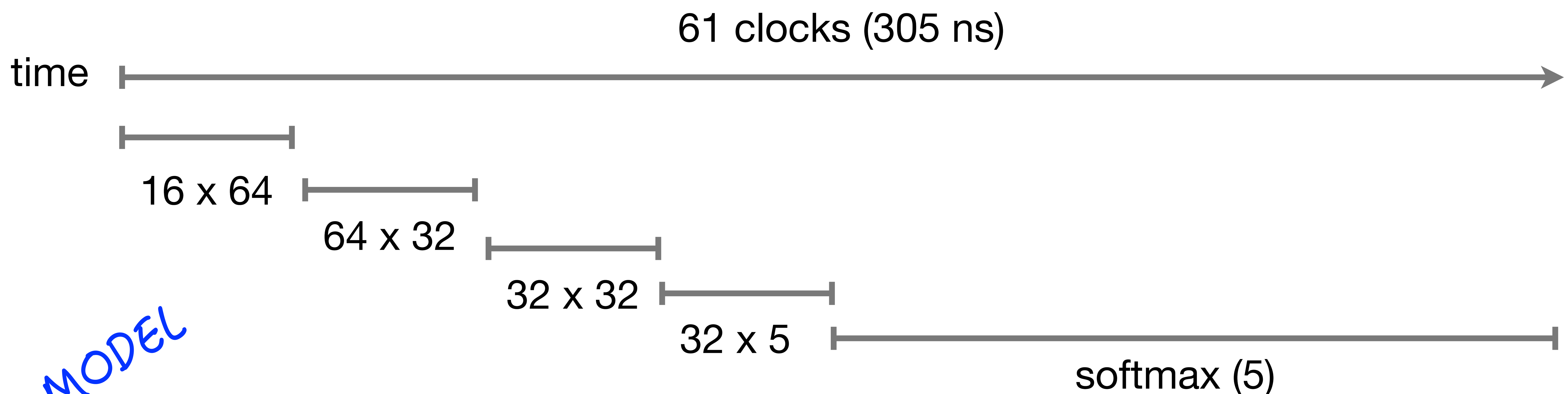
Take a simple 1-layer network and scan in input/weight precision

Reduced precision can greatly reduce resource usage

e.g. factor of 4 reduction with 18 instead of 32 bits with minimal loss in performance



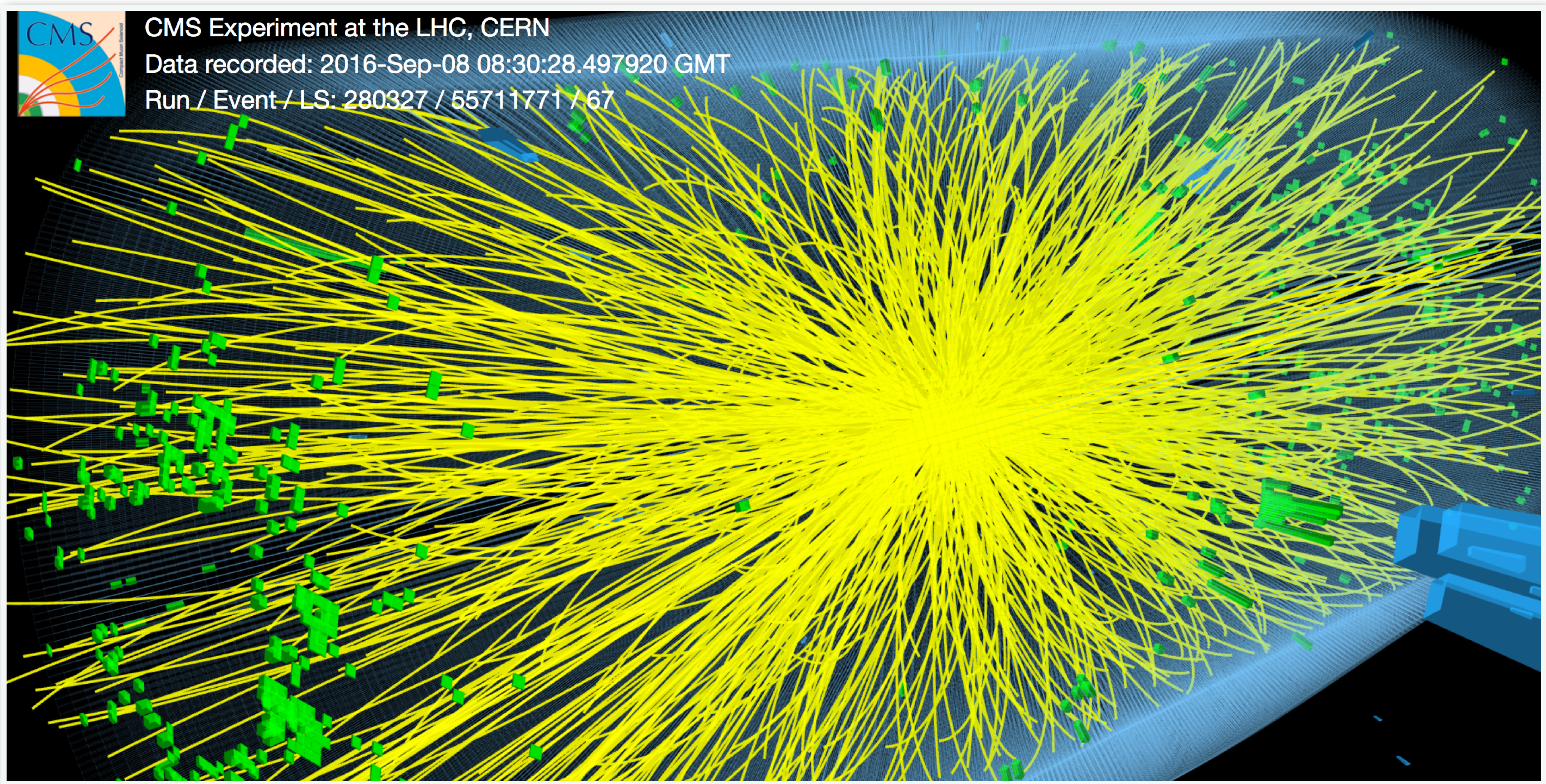
UNDER THE HOOD



ORIGINAL MODEL

	BRAM	DSP	FF	LUT
Total	**3**	**3329**	**95924**	**8127**
% Usage	**~0**	**92**	**11**	**18**

THE COMPUTING CHALLENGE

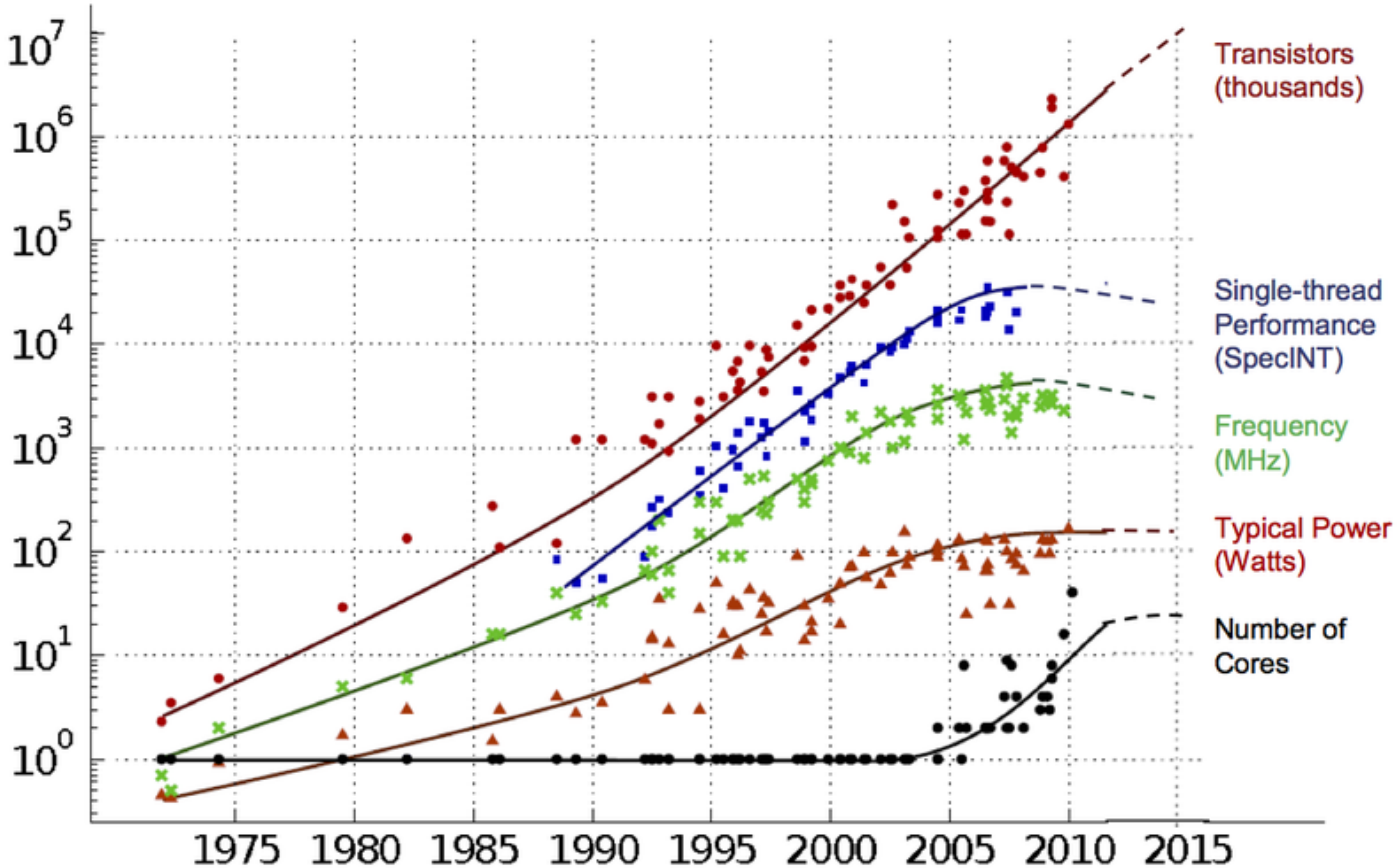


Current:
~5 minutes per HL-LHC event

100 times the data… exabytes!

Major HLT and computing challenges going forward!

MOORE'S LAW AND DENNARD SCALING

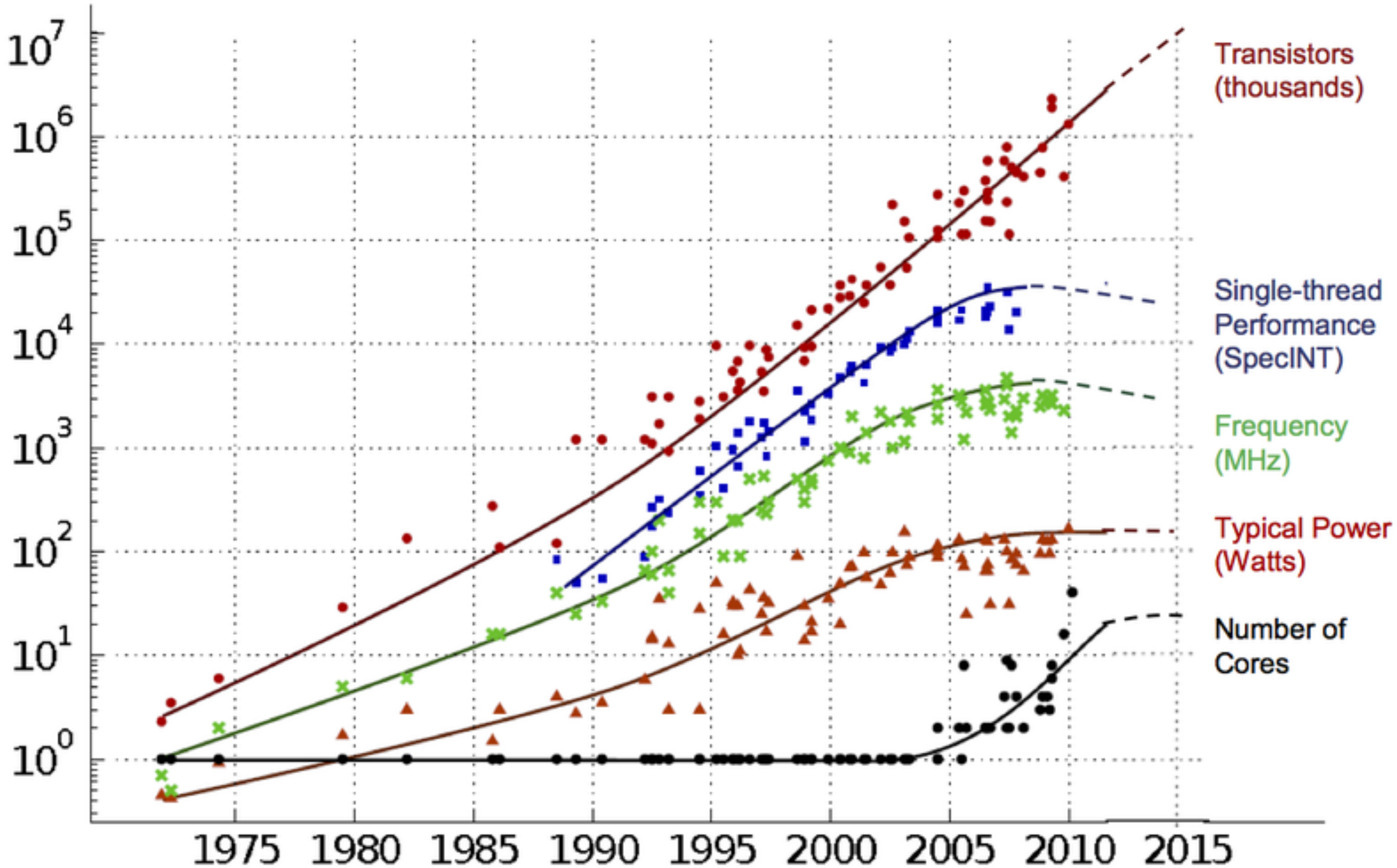


Moore's Law continues

Dennard Scaling fails

Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

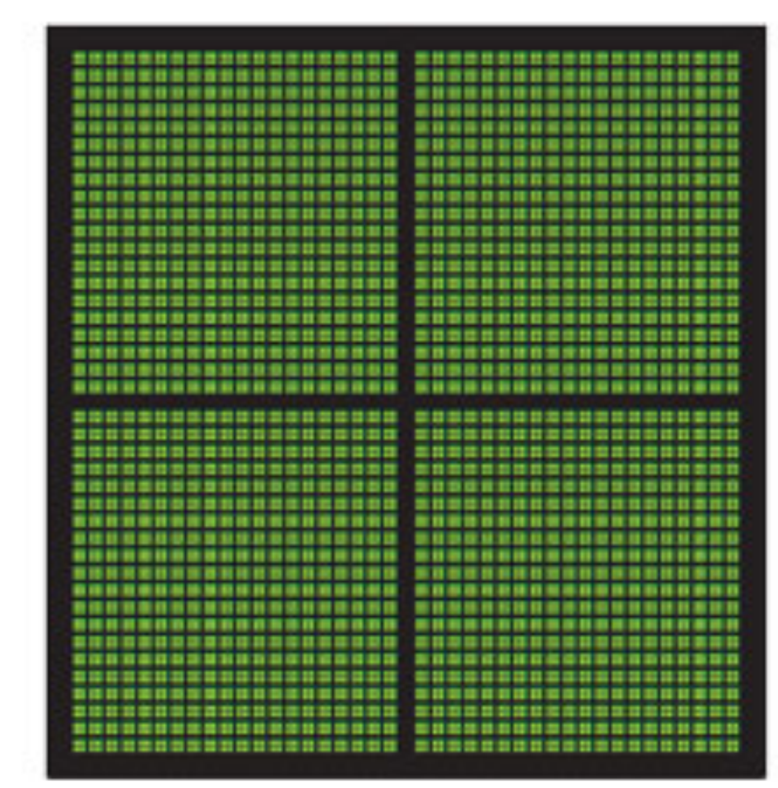
Moore's Law and Dennard Scaling



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

Moore's Law continues

Dennard Scaling fails

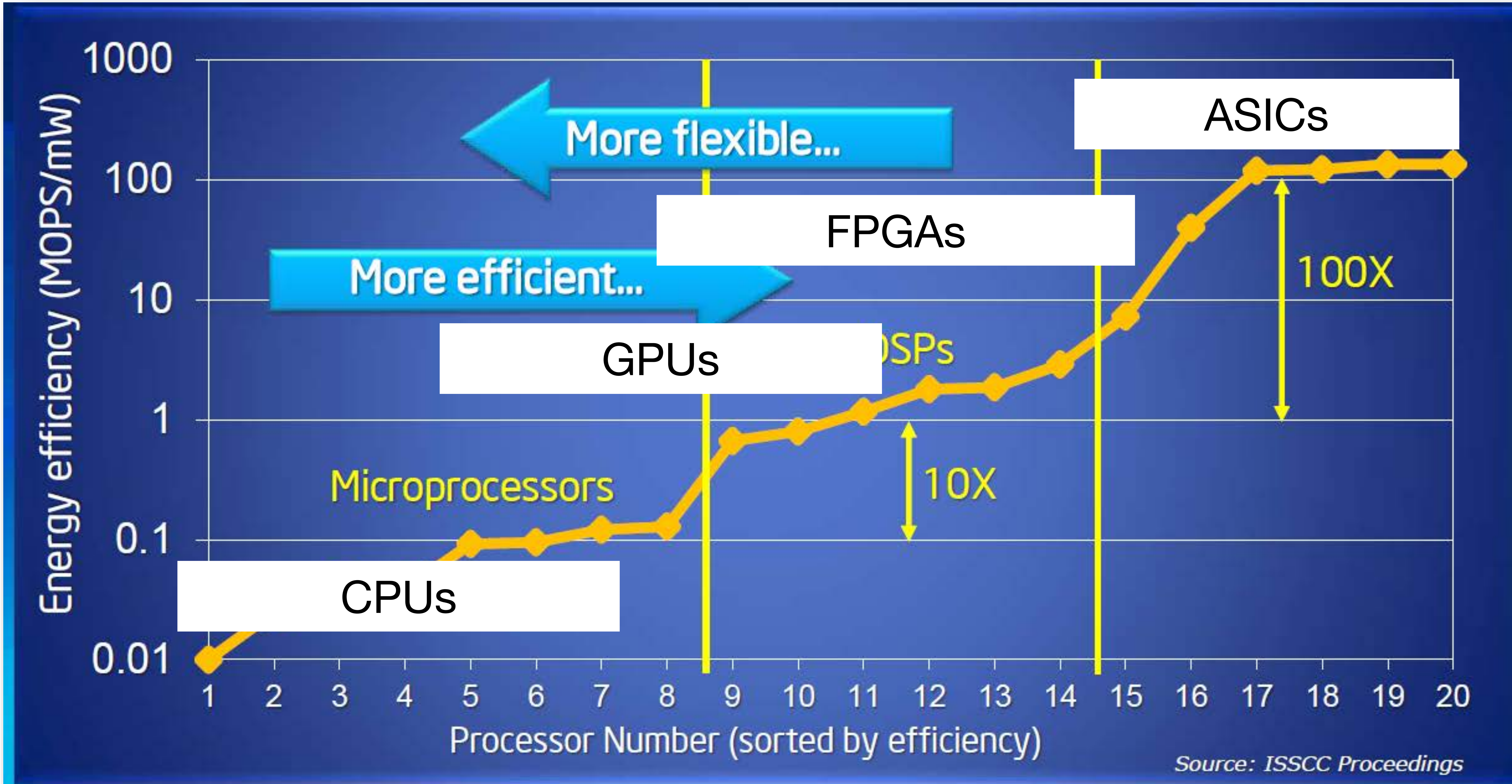


Single threaded performance not improving

Circa ~2005: "The Era of Multicore"

→ **Today: Transition to the "Era of Specialization"?** (c.f. Doug Burger)

ARCHITECTURES



Source: Bob Broderson, Berkeley Wireless group

* GPUs still best option for training
* FPGAs generally much more power efficient