Hardware Software Co-Optimization of SoftMax on Snitch

Integrated Systems Laboratory (ETH Zürich)

luca name

Run Wang runwang@student.ethz.ch

Supervisor: Gamze Islamoglu, Angelo Garofalo

@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

PULP Platform

Open Source Hardware, the way it should be!

Introduction

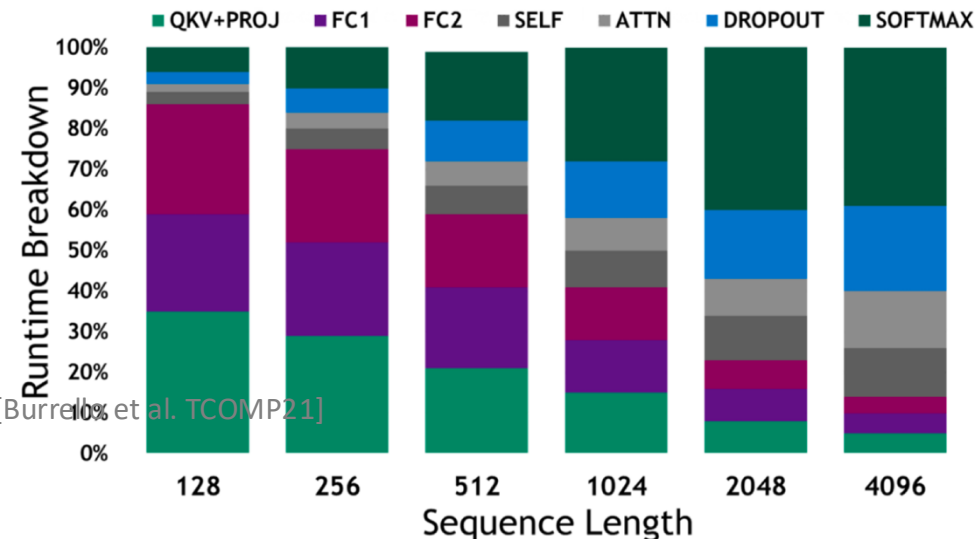
- **Attention**
  - Important for transformer based LLM
  - Quadratic complexity ($\mathrm{softmax}(QK^T)$), bad for long sequence
- **SoftMax in Attention**

softmax function

  - Non-linearity and precision requirements
  - EXP, DIV unfriendly for computing
  - Reduce max, sum bad for parallel computing

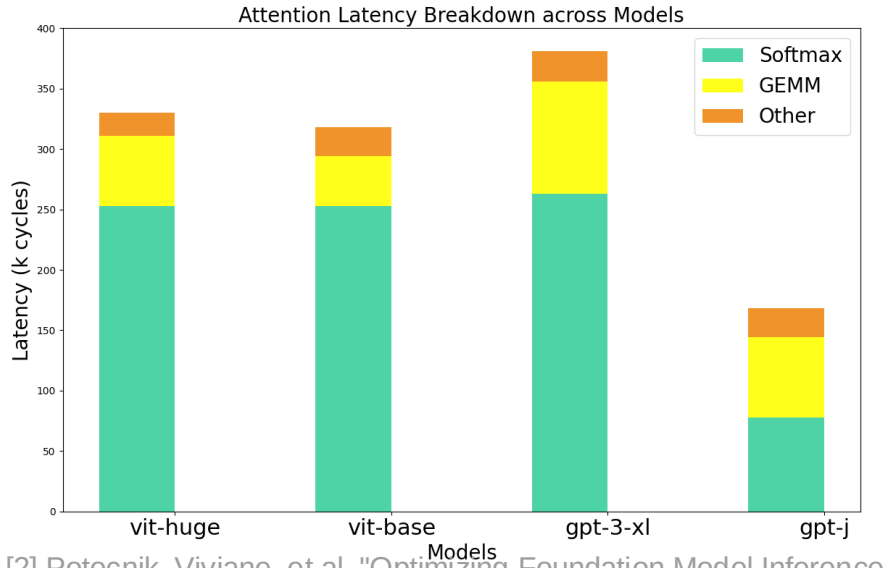
Motivation: Transformer/LLM Runtime Benchmark

- **SofterMax [1]**
  - SoftMax runtime benchmark in Attention
- **LLM on Multi Snitch Cluster [2]**
  - Attention runtime benchmark of LLM on Multi Snitch Cluster
  - Attention runtime benchmark



[Burrello et al. TCOMP21]

[1] Stevens, Jacob R., et al. "Softermax: Hardware/software co-design of an efficient softmax for transformers."



[2] Potocnik, Viviane, et al. "Optimizing Foundation Model Inference on a Many-tiny-core Open-source RISC-V Platform."

Contribution

- **Optimized the Softmax function**
  with existing instruction of Snitch and analyzed exp as a bottleneck
- **Designed a new exp instruction**
  integrated exp instruction into the Snitch cluster
- **Physical implementation**
  analyzed the hardware cost for the new instruction
- **Benchmarked the New Softmax function**
  with the new exp instruction

Background

Softmax Background: LLM, Attention, Softmax

LLM

- **MultiHead Attention**
- Other: FFN, Non-Linear, Embedding

Attention

- GEMM: QK^T, OV
- **Softmax**: Quadratic with seq len

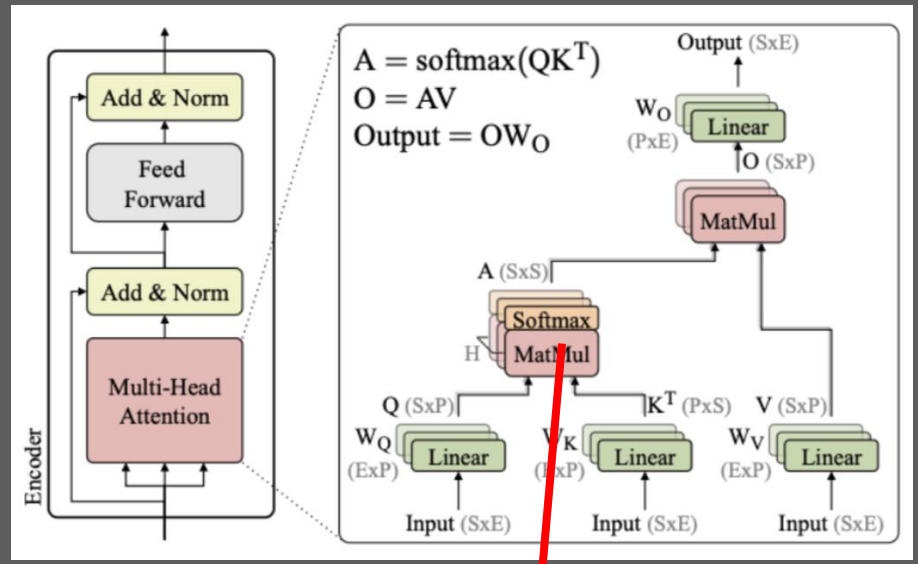
Softmax

1. Avoid overflow
2. Compatible with flashattention

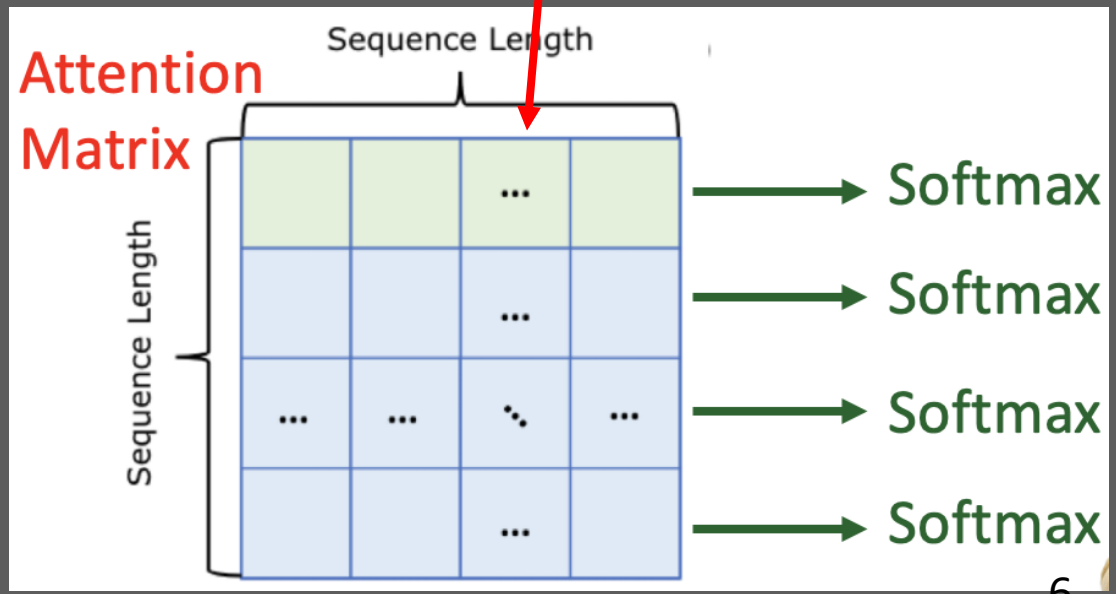
$$\text{Softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

$$\text{Softmax}(\mathbf{z})_i = \frac{e^{z_i - \max(\mathbf{z})}}{\sum_{j=1}^{N} e^{z_j - \max(\mathbf{z})}}$$

Multi-Head Attention



Softmax on Attention Matrix

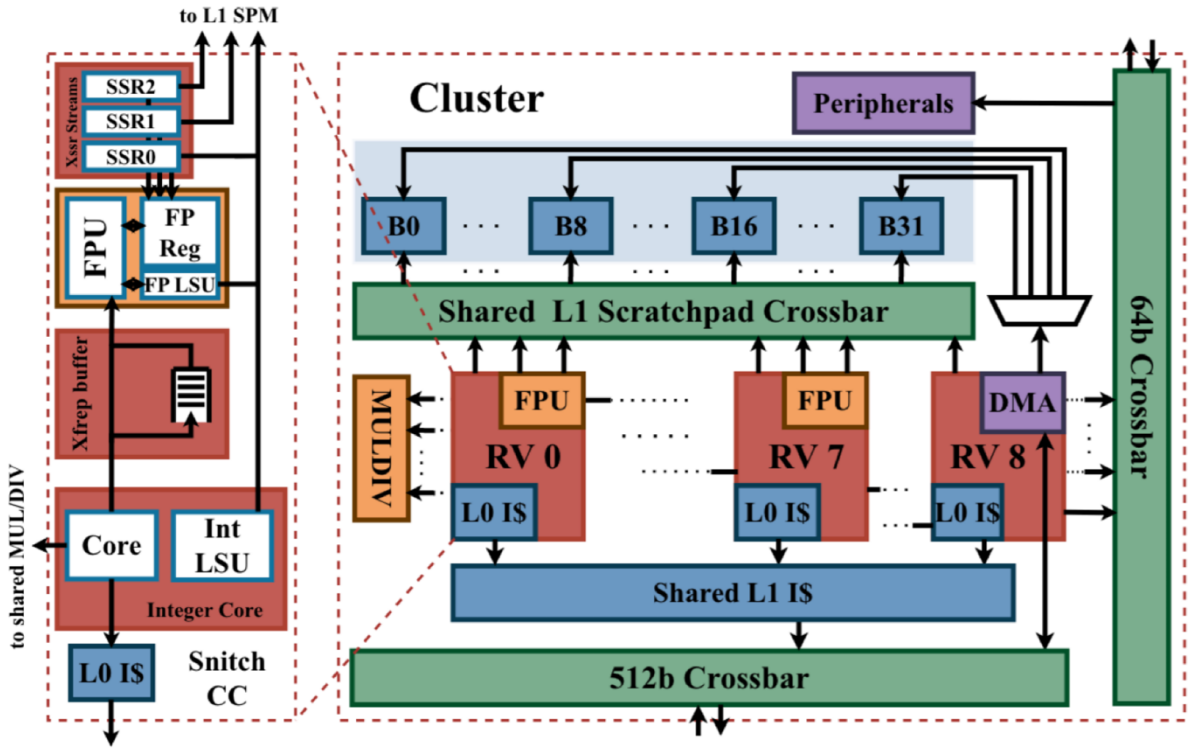


ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Background

Hardware Background: Snitch Cluster

- **Snitch Cluster 8 cores+1DMA**
  - 128KB SPM
  - FPU 64b with Group FMA, DIV, NONCAMP, CAST, DOTP
  - Frep, SSR reduce explicit load and store and branching overhead
    - SSR: loop data prefetch
    - Frep: loop repetition control



Potocnik, Viviane, et al. "Optimizing Foundation Model Inference on a Many-tiny-core Open-source RISC-V Platform."

Background

Softmax on Snitch: Baseline & Benchmark

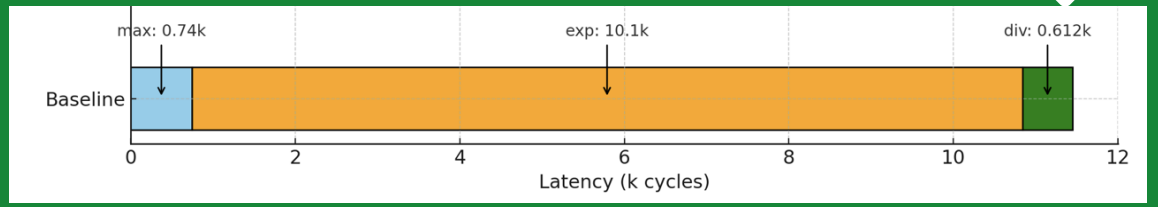
Softmax on Snitch

$$\mathrm{Softmax}(\mathbf{z})_i = \frac{e^{z_i - \max(\mathbf{z})}}{\sum_{j=1}^{N} e^{z_j - \max(\mathbf{z})}}$$

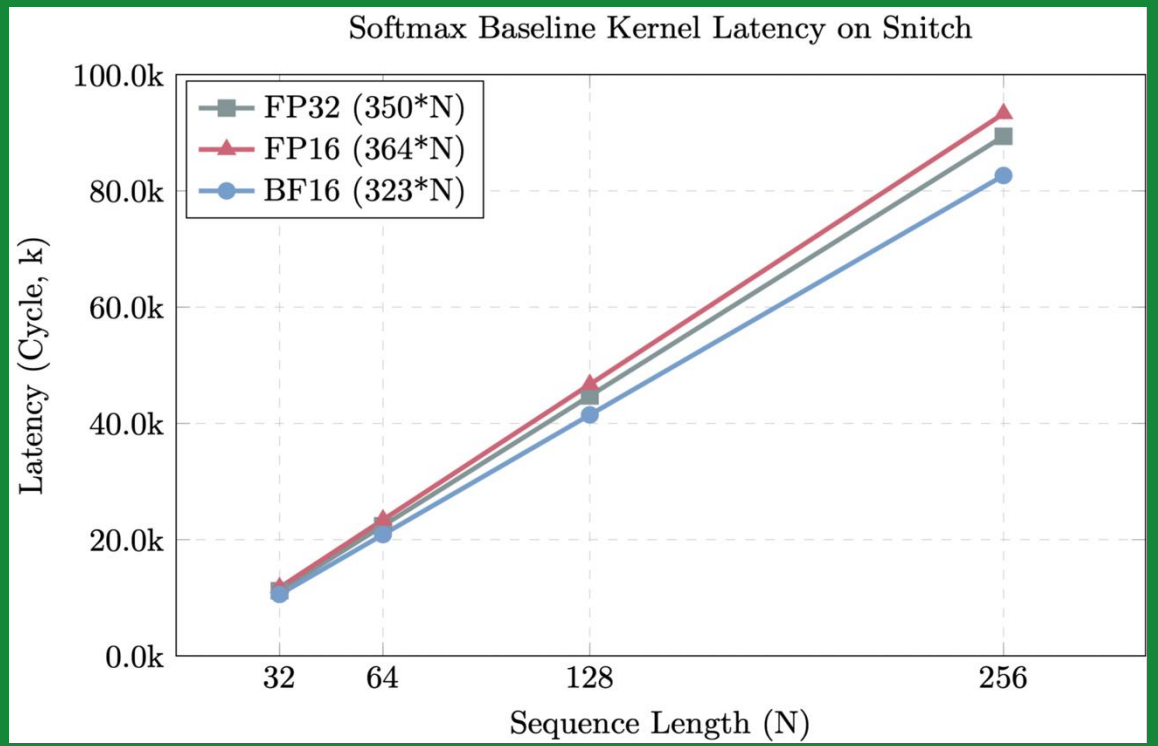
- Max Calculation
- Exponential Calculation
- Normalization (Division)

Max	C ArrayMax
Exp	C LUT + Polynomial
Div	C ArrayDiv

Latency Breakdown of Seq 32



Softmax Latency linearly increase with Sequence Length



ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

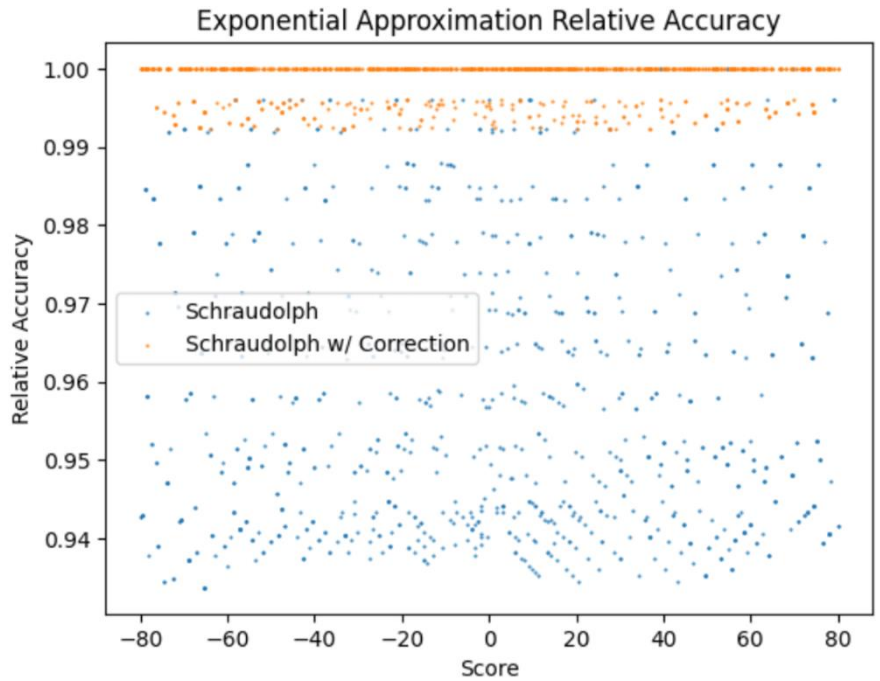
Background

Shauldolph Exp Approximation Hardware Implementation

- **Shauldolph Exp Algorithm**
- **Shauldolph Hardware**
  - Implementation on BF16 (by Andrea Delano)
  - Shauldolph Algorithm for initial calculation
  - Mantissa Correction with Second-Order Linear Approximation
- **Exp Accuracy:**
  - Avg 99.86%
  - Min 99.25%

Algorithm 1 Shauldolph Exponential Approximation

1: **Input:** `x`
2: **Output:** `approx_exp`
3: **Constants:** `a, b` {These constants are chosen for the approximation}
4: **Procedure:**
5: `approx_exp` $\leftarrow 2^{a \cdot x + b}$
6: **return** `approx_exp`



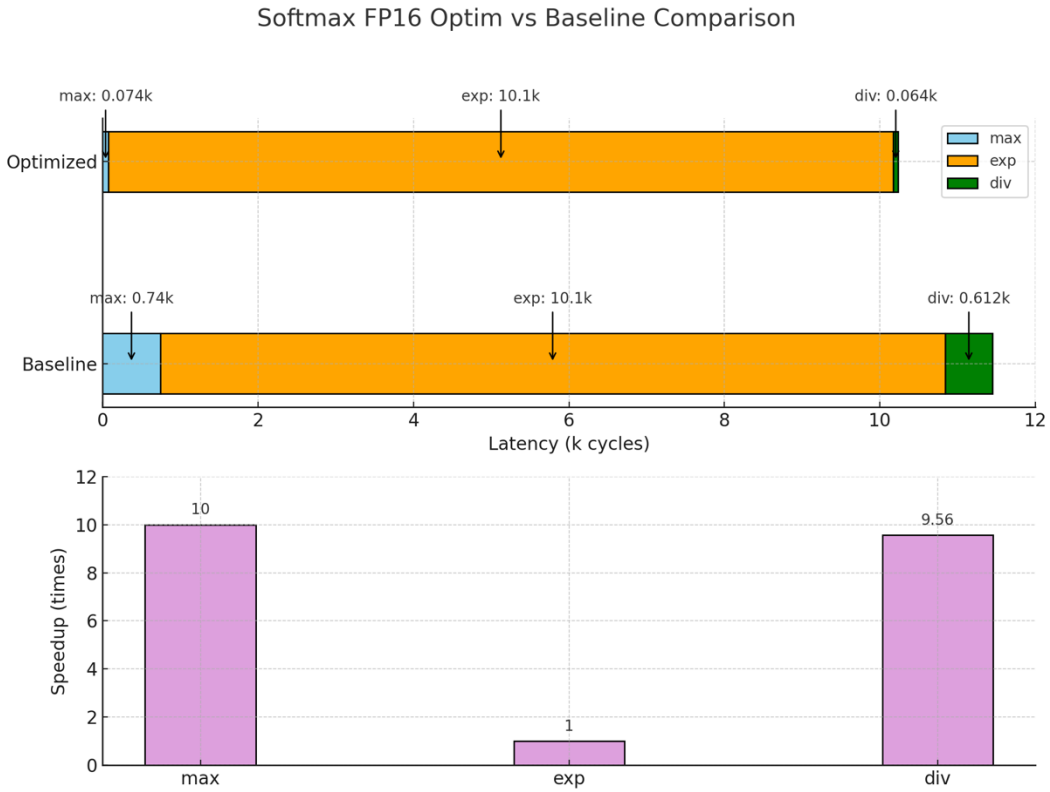
Contribution

- **Optimized the Softmax function**
  with existing instruction of Snitch and analyzed exp as a bottleneck
- **Designed a new exp instruction**
  integrated exp instruction into the Snitch cluster
- **Physical implementation**
  analyzed the hardware cost for the new instruction
- **Benchmarked the New Softmax function**
  with the new exp instruction

Snitch Softmax Optim 1.0

Softmax with SSR, FREP, SIMD

- **Snitch SSR, FREP, SIMD**
  - Exp: Softmax FP16 Seq 32
  - Bottleneck: EXP calculation

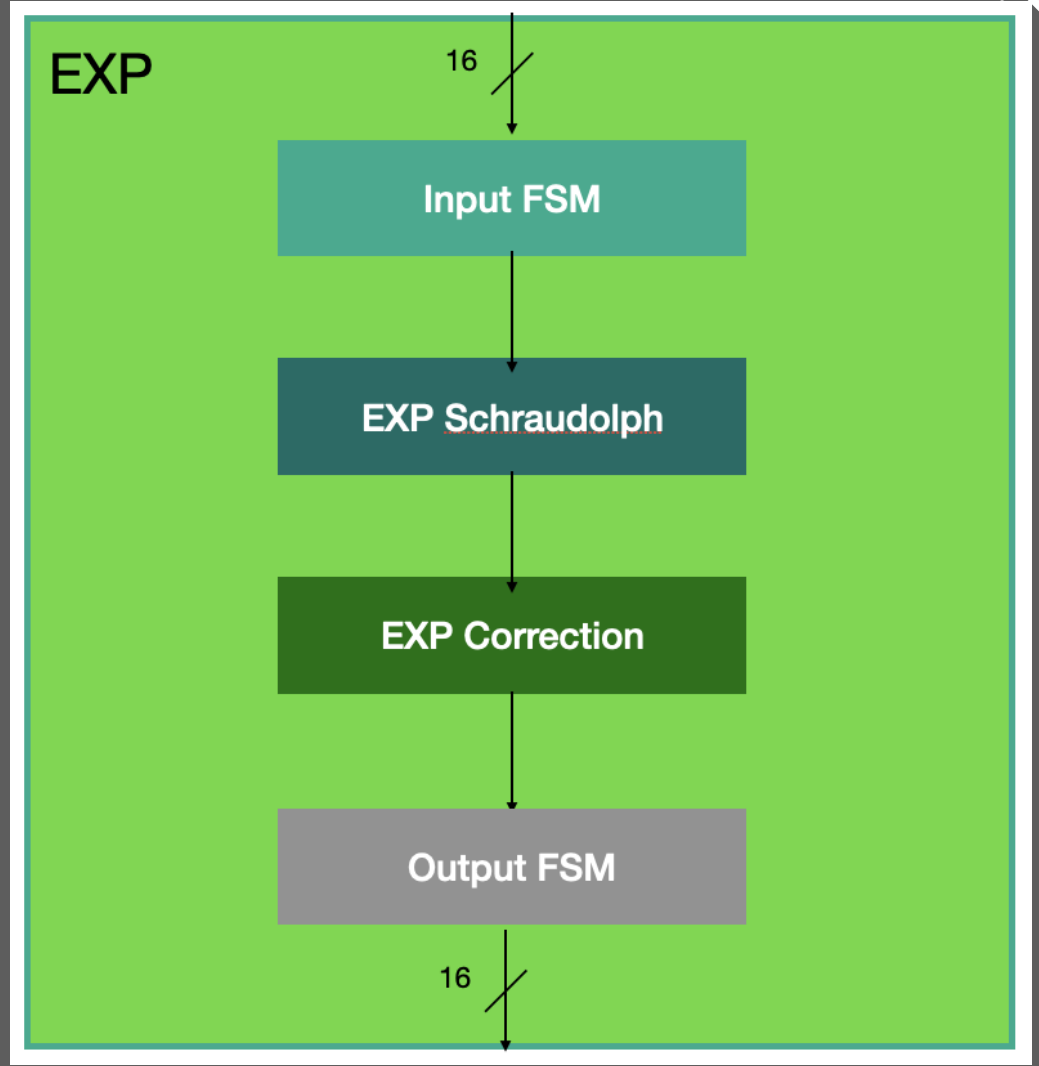


	Baseline	Optim
Max	C ArrayMax	SSR, Vfmax, FREP
Exp	C LUT + Poly	C LUT + Poly
Div	C ArrayDiv	SSR, Vfdiv, FREP

```
// Step 1: Find the maximum value
ssr config ft1 read double;
asm volatile (
"frep.o %[n_frep], 1, 0, 0"
"vfmax.h ft3,ft3,ft0")
// Step 2: exponentials and sum
// Step 3: Normalize
ssr config ft1 read double, ft2 write
asm volatile (
"frep.o %[n_frep], 1, 0, 0"
"vfdiv.h ft2,%[sum],ft0")
```

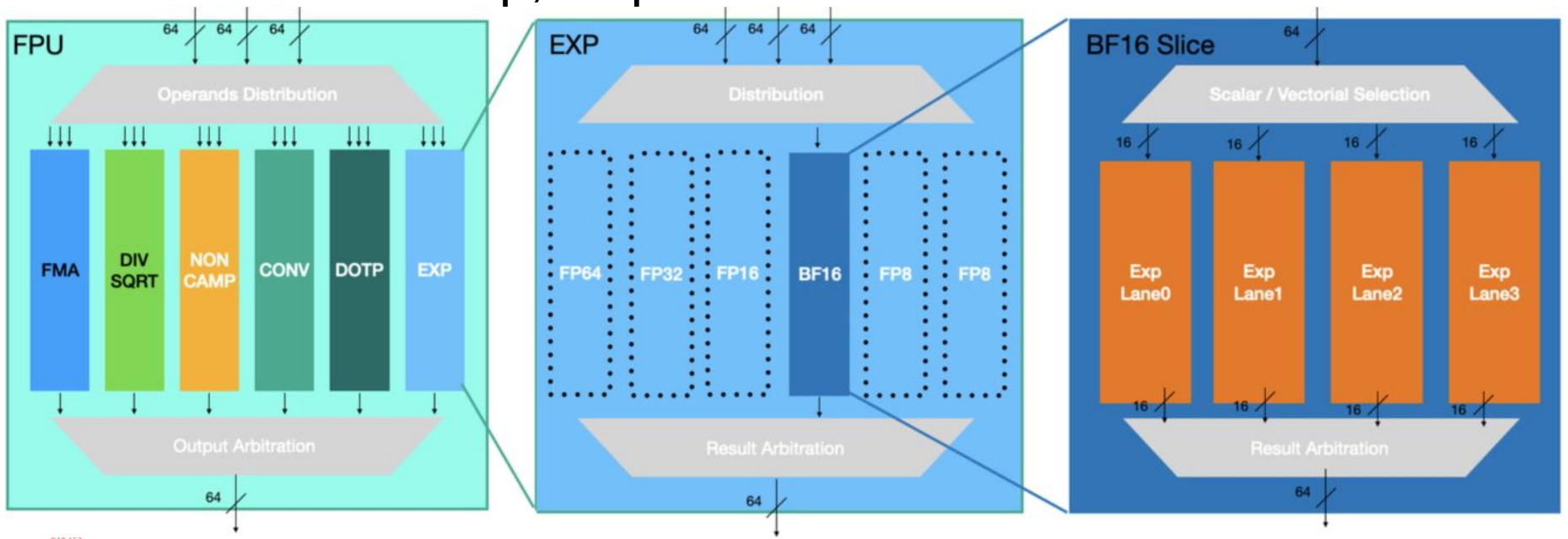
EXP block into Snitch

- **EXP BF16 in FPU**
- **Pipeline Stages**
  - Configurable input/output pipelines
  - Default **1** Input Register**, 1** Output Register
- **Core Components**
  - **Expo_Schraudolph**: initial approximation
  - **Expu_Correction**: Refines the Schraudolph approximation
- **Verification**
  - Python Golden Model



EXP block into Snitch

- **FPU**
  - Group EXP into FPU
  - Only BF16 is activated, with 4 Exp lanes in BF16 Slice
- **FP_SS decoder:** add vfexp, fexp decoder
- **Snitch decoder:** add vfexp, fexp decoder



Snitch Softmax Optim 2.0

Fexp, Vfexp integrated in Snitch and CVFPU

- **Exp**
  - Substract max
  - Exp
  - Sum
- **EXP Sum Separate Loop**
  - Exp value write back & used for sum
  - Extra memory access

	Baseline	Optim
Max	C ArrayMax	SSR, Vfmax, FREP
Exp	C LUT + Poly	SSR, Vfexp, FREP
Div	C ArrayDiv	SSR, Vfdiv, FREP

```
// Step 1: Find the maximum value
ssr config ft1 read double;
asm volatile (
"frep.o %[n_frep], 1, 0, 0"
"vfmax.h ft3,ft3,ft0")

// Step 2: exponentials and sum
ssr config double ft1 read, ft2 write;
asm volatile (
"frep.o %[n_frep], 2, 0, 0"
"vfsub.ah ft3, ft1, %[max]"
"vfexp.ah ft2, ft3")
//sum
ssr config double ft1 read, ft2 write;
asm volatile (
"frep.o %[n_frep], 1, 0, 0"
"vfadd.ah %[sum],%[sum] ft1, )

// Step 3: Normalize
ssr config ft1 read double
asm volatile (
"frep.o %[n_frep], 1, 0, 0"
"vfdiv.h ft4,%[sum],ft0")
```

Snitch Softmax Optim 2.0+

Loop Unroll, Exp_Sum, Div->Mul

- **Loop Unroll:**
  - existing extra float registers and frep entries
- **Exp Sum:** to save memory access
- **Div**: Multi-cycle path, replaced by mul

	Baseline	Optim
Max	C ArrayMax	Loop Unroll, SSR, Vfmax, FREP
Exp	C LUT + Poly	Loop Unroll, SSR, Vfexp, FREP
Div	C ArrayDiv	Loop Unroll, SSR, Vfmul, FREP

- **Loop Unroll**

```
"frep.o %[n_frep],16"
"vfmax.h ft3,ft3,ft0"
"vfmax.h ft4,ft4,ft0"
"vfmax.h ft5,ft5,ft0"
…
```

- **Exp Sum Combine** with redundant instruction

```
"frep.o %[n_frep], 16"
"vfsub.ah ft3, ft1, %[max]"
"vfexp.ah ft3, ft3"
"vfsgnj.ah ft2, ft3"
"vfadd.ah ft4, ft3"
```

- **Div replaced by mul**

```
"frep.o %[n_frep], 1"
"vfmul.h ft4,%[1/sum],ft0"
```

Snitch Softmax Summary

Baseline

```
for (int i = 0; i < N; i++) {
if(x[i] > max_val)
max_val = x[i]
}

for (int i = 0; i < N; i++) {
y[i] = exp(x[i]- max_val)
sum += y[i];}

for (int i = 0; i < N; i++) {
y[i]/=sum;
}
```

360 Cycle/N

Softmax Optim1

```
ssr config ft1 read double;
asm volatile (
"frep.o %[n_frep], 1, 0, 0"
"vfmax.h ft3,ft3,ft0" )

for (int i = 0; i < N; i++) {
y[i] = exp(x[i]- max_val)
sum += y[i];}

ssr config ft1 read double
asm volatile (
"frep.o %[n_frep], 1, 0, 0"
"vfdiv.h ft2,%[sum],ft0" )
```

329 Cycle/N

Softmax Optim2

```
ssr config ft1 read double;
asm volatile (
"frep.o %[n_frep], 1, 0, 0"
"vfmax.h ft3,ft3,ft0" )
ssr
asm volatile (
"frep.o %[n_frep], 2, 0, 0"
"vfsub.ah ft3, ft1, %[max]"
"vfexp.ah ft2, ft3")
ssr
asm volatile (
"frep.o %[n_frep], 1, 0, 0"
"vfadd.ah %[sum],%[sum] ft1, )
ssr config ft1 read double
asm volatile (
"frep.o %[n_frep], 1, 0, 0"
"vfdiv.h ft2,%[sum],ft0" )
```

4 Cycle/N

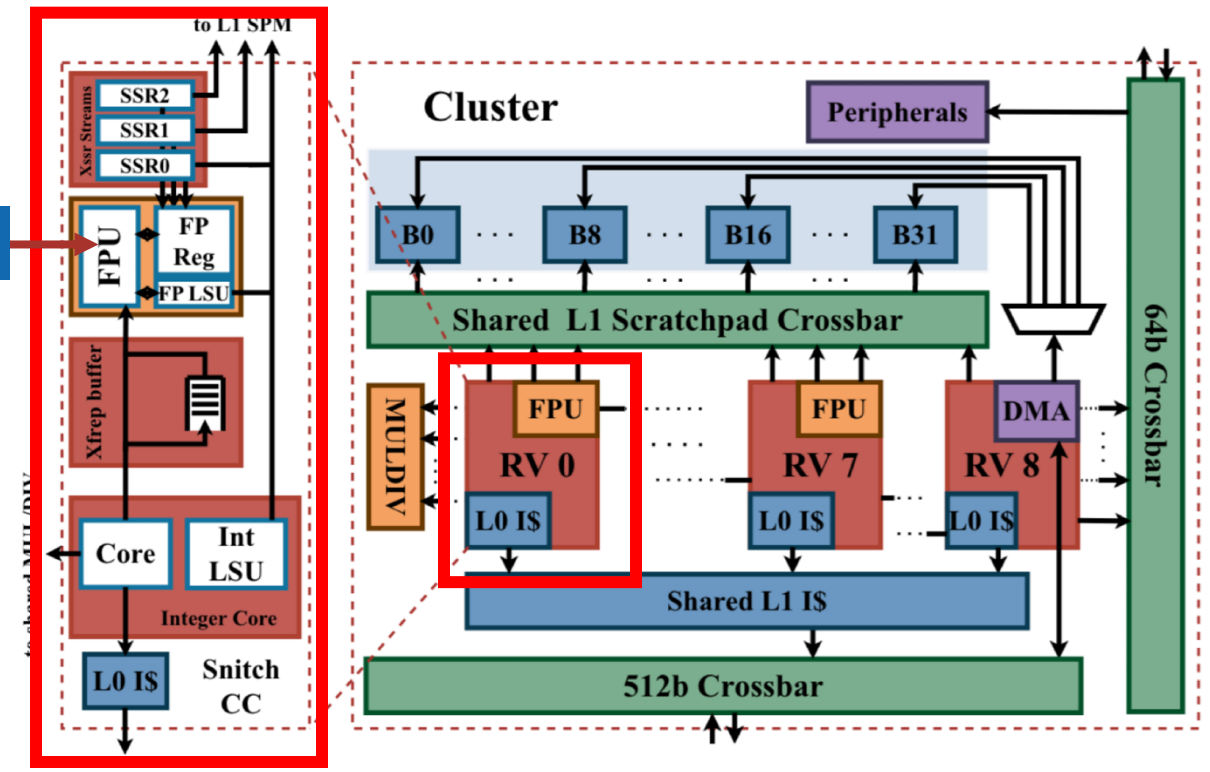
Softmax Optim2+

```
ssr
"frep.o %[n_frep],16"
"vfmax.h ft3,ft3,ft0"
"vfmax.h ft4,ft4,ft0"
"vfmax.h ft5,ft5,ft0"
…
ssr
"frep.o %[n_frep], 16"
"vfsub.ah ft3, ft1, %[max]"
"vfsub.ah ft4, ft1, %[max]"
"vfexp.ah ft3, ft3"
vfexp.ah ft4, ft4"
"vfsgnj.ah ft2, ft3"
"vfsgnj.ah ft2, ft4"
"vfadd.ah ft3, ft3"
"vfadd.ah ft4, ft4"
ssr
"frep.o %[n_frep], 1"
"vfmul.h ft4,%[1/sum],ft0"
"vfmul.h ft4,%[1/sum],ft0"
"vfmul.h ft4,%[1/sum],ft0"
.....
```

2.125 Cycle/N

Physical Implementation Setup

- Fusion Compiler 2022.03
- GF12nm
- Full synthesis and backend
- Snitch_cc with Exp Group, with Div Group
- Fre: 1GHz
- Worst-case corner at 0.72V and 125˚C or -40˚C

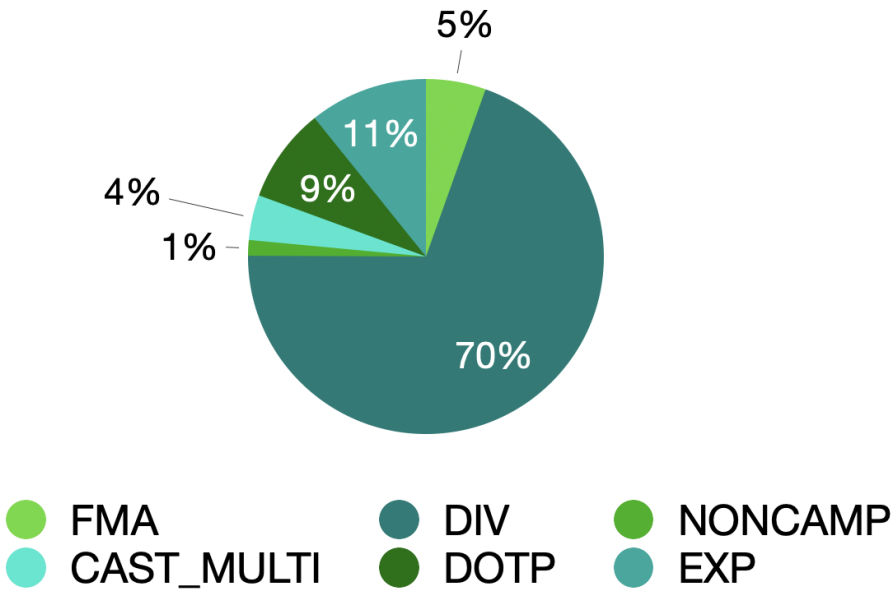


Backend Result

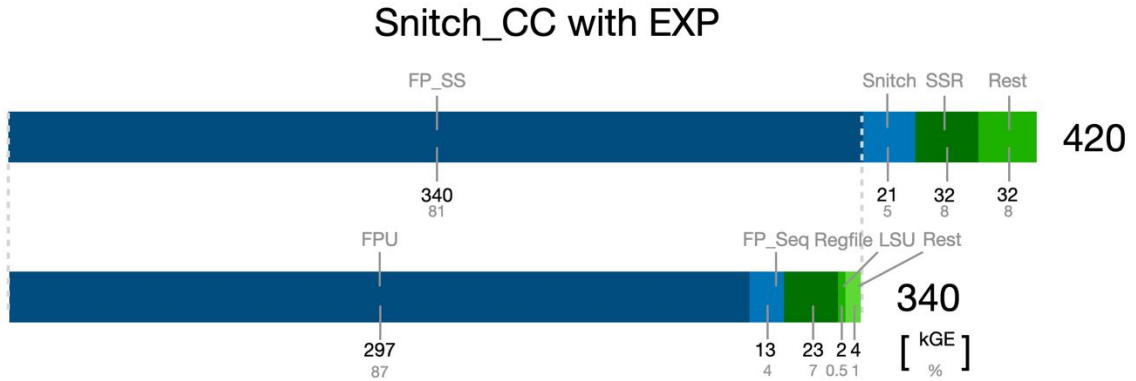
- **Timing for snitch_cc with exp**
  - **SS:** 0.72V, 125˚C
    - Critical path: fma
    - Frequency: 957MHz
  - **TT:**
    - Critical path: sdotp
    - Frequency: 1.31GHz
- **Timing for snitch_cc baseline**
  - **SS:** 0.72V, 125˚C
    - Critical path: fma
    - Frequency: 957MHz
  - **TT:**
    - Critical path: sdotp
    - Frequency: 1.31GHz

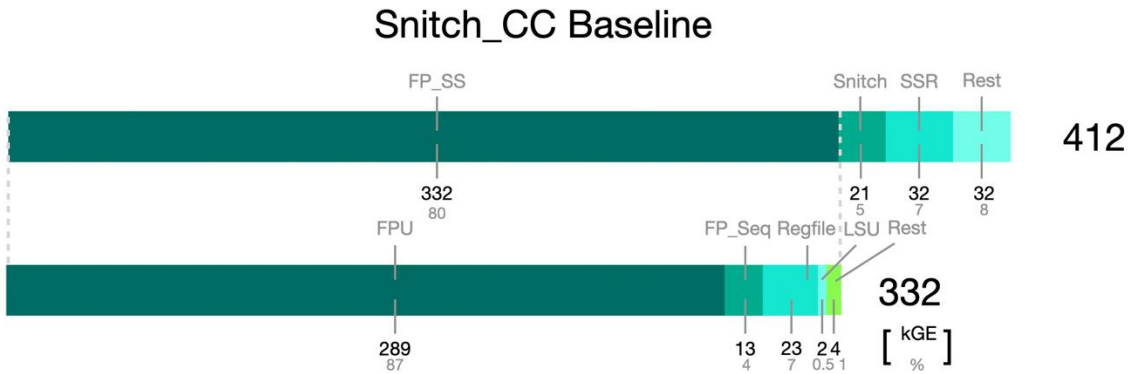
Backend Result

- **Area**
  - Exp block: 8kGE
  - Snitch_cc increase by 1.9%



FPU Group Area Breakdown

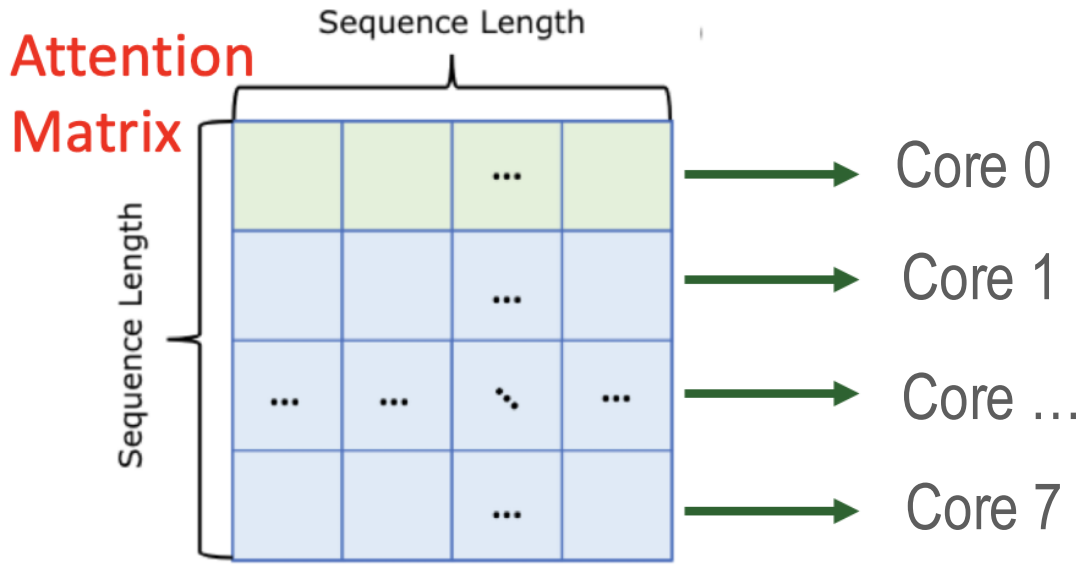




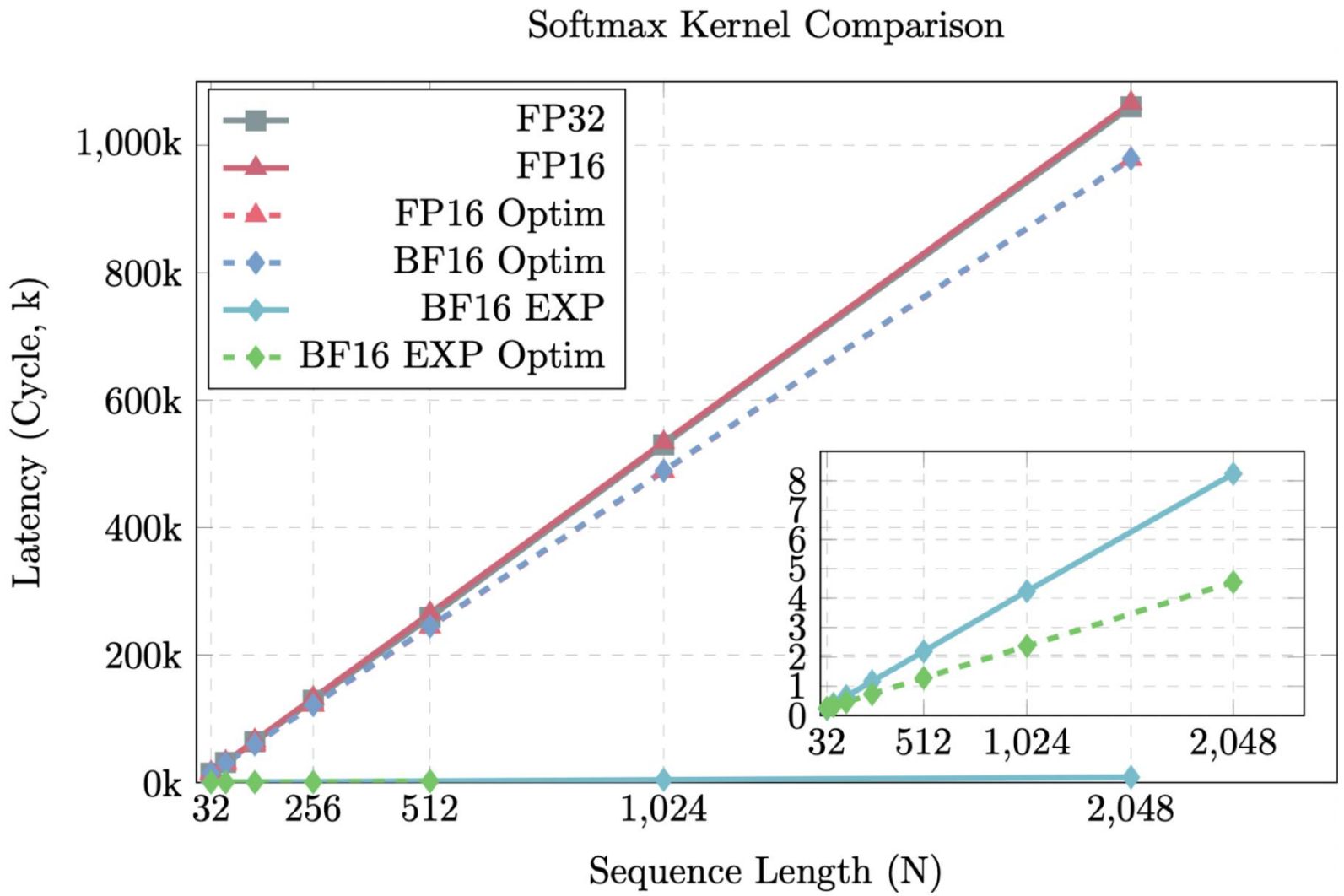
Softmax Kernel Benchmark

- **Exp setup**
  - 8 cores: each core 1 row
  - Softmax relevant Error < 0.01
  - icache preheat

FP32	C Baseline
FP16	C Baseline
FP16 Optim	FREP, SSR, SIMD
BF16 Optim	FREP, SSR, SIMD
BF16 EXP	FREP, SSR, SIMD, Vfexp
BF16 EXP Optim	FREP, SSR, SIMD, Vfexp, optim

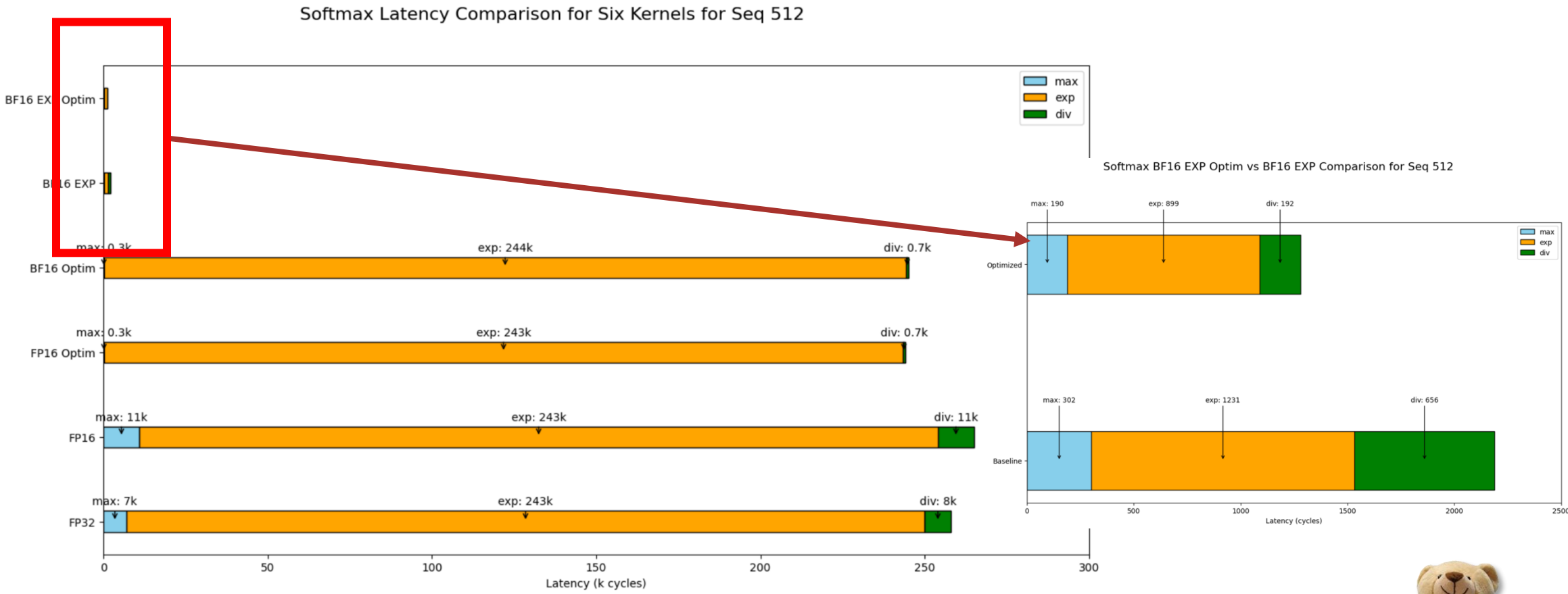


Softmax Kernel Benchmark Result



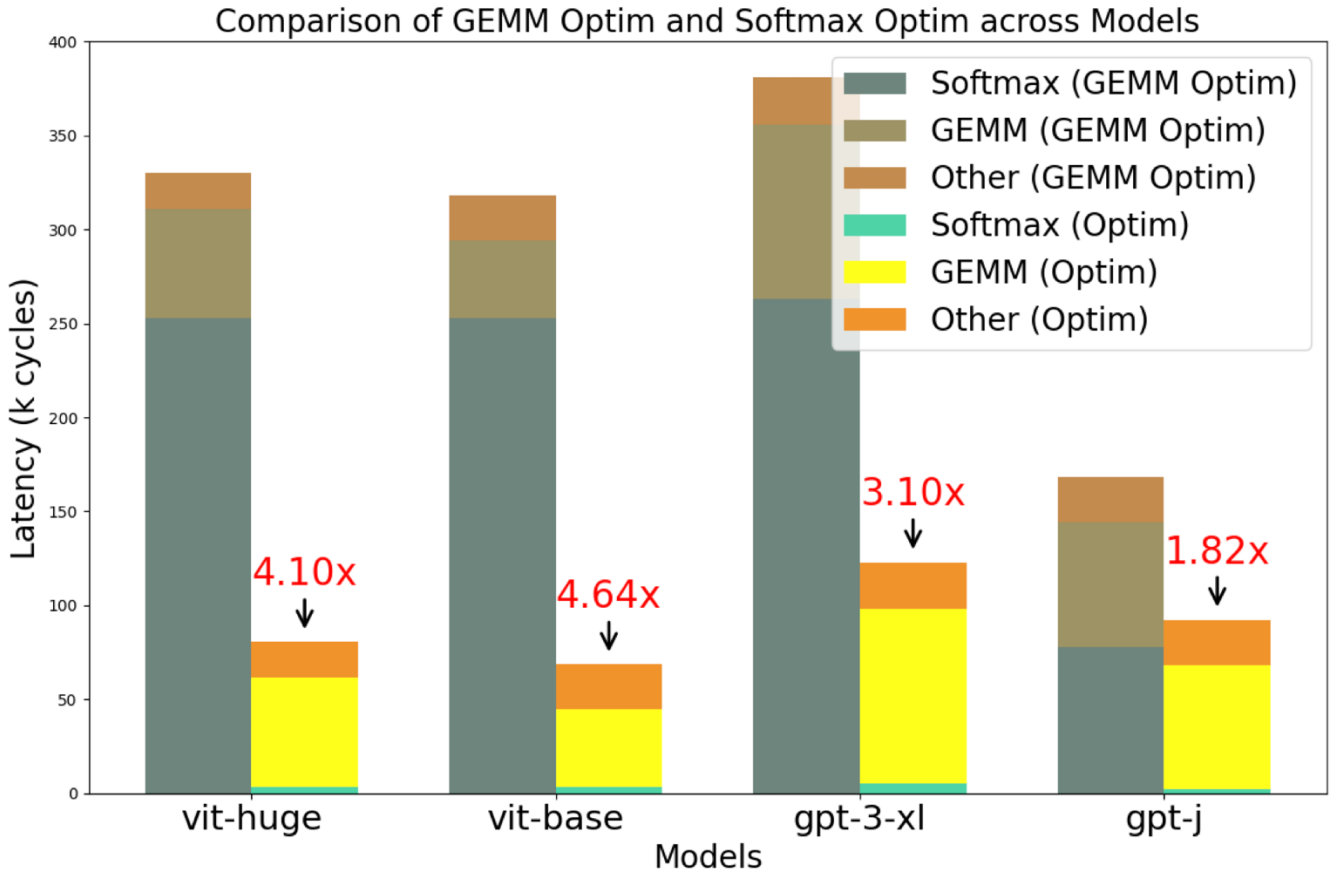
Softmax Kernel Benchmark Result

- **Softmax bf16 optim 100+x vs FP32 for Seq 512**



Flashattention New Benchmark (Socdaml)

- Config: Vit-Huge, Vit-base, GPT-3-xl, GPT-j
- Modelsim, LLVM O3
- flashattention2 fp32 kernel
- Simulation on one snitch cluster



Thank you!