VEXP: A Low-Cost RISC-V ISA Extension for Accelerated Softmax Computation in Transformers

Run Wang *, Gamze Islamoglu *, Andrea Belano†, Viviane Potocnik*, Francesco Conti†, Angelo Garofalo†, Luca Benini*†

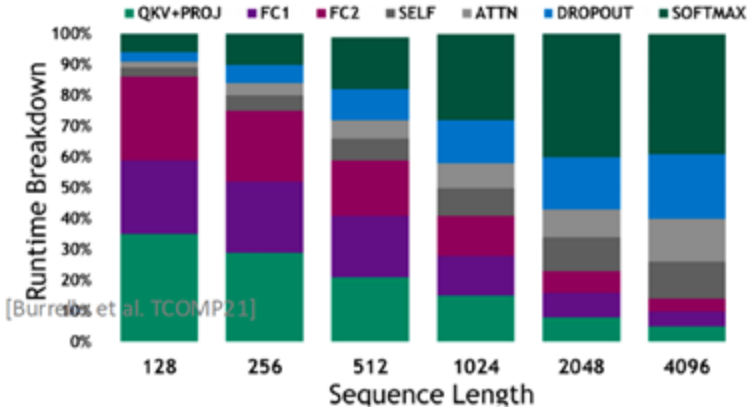
*Integrated Systems Laboratory, ETH Zürich
†DEI, University of Bologna

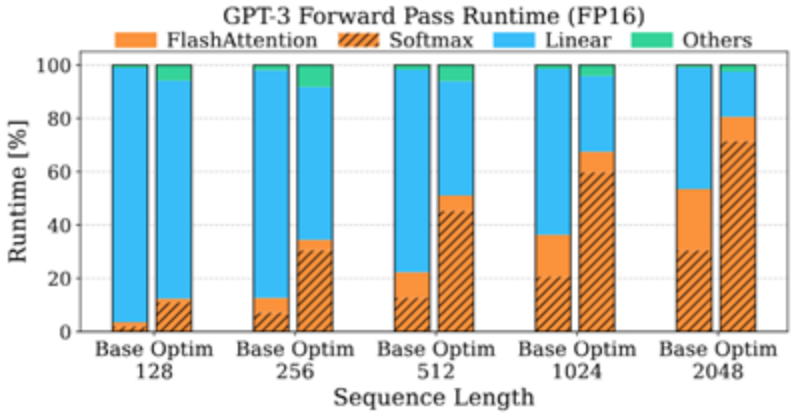
PULP Platform
Open Source Hardware, the way it should be!

@pulp_platform
pulp-platform.org
youtube.com/pulp_platform

Introduction and Motivation

- **Transformers are the main models driving the evolution of modern Artificial Intelligence**
  - Both in **perceptive** task and in **generative** applications
- **However, this performance uplift comes at a cost**
  - Each layer of a Transformer is more complex, featuring multi-head self-attention (MHSA) and additional projections
  - **Softmax** becomes a **bottleneck** when linear operations are accelerated

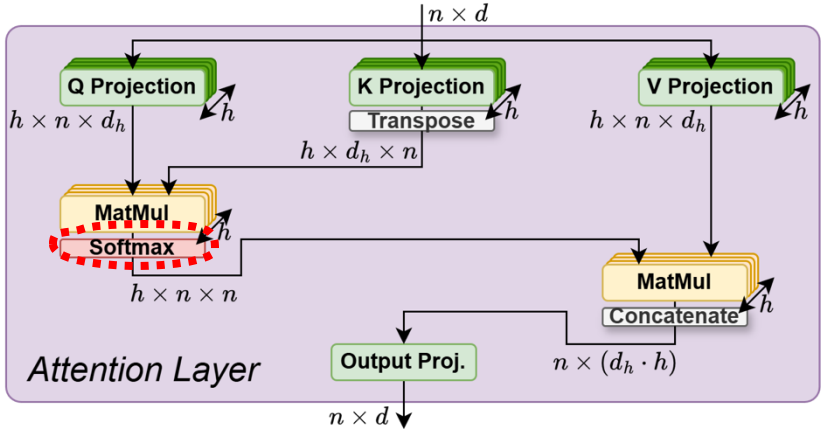




Source: Stevens, J.R., et al., "Softermax: Hardware/software co-design of an efficient softmax for transformers", DAC 2021.

The Attention Mechanism & Softmax

- **Attention consists of multiple GEMMs + softmaxs**
  - Unlike CNNs, softmax is applied multiple times every layer
- **It is fundamental to also accelerate softmax if we target Transformer-based models**
- **What's the deal with this function?**
  - It is based on the **exponential** function
  - It is **NOT** a **point-to-point** function

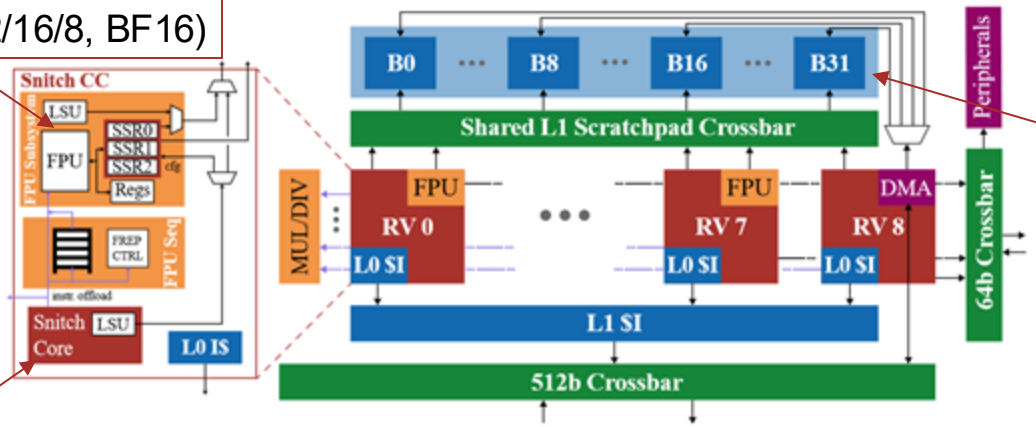


$$Softmax(x_i) = \frac{e^{x_i - x_{max}}}{\sum e^{x_j - x_{max}}}$$

Snitch Cluster Architecture

- **Tiny energy-efficient RISC-V core paired with a floating-point unit, optimized for high-performance data-parallel workloads.**
- **Advanced ISA extensions**:
  - FREP: Hardware loop controller (repeats FPU ops autonomously)
  - SSR: Stream-based memory access with affine patterns (no explicit load/store)

64-bit SIMD-capable FPU (supports FP64/32/16/8, BF16)



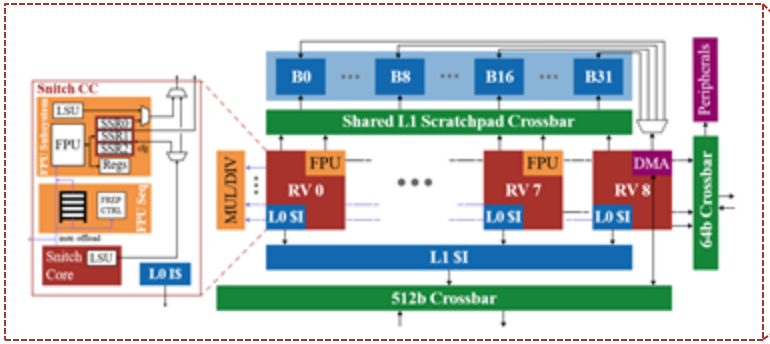
128 KB shared SPM with single-cycle logarithmic interconnect

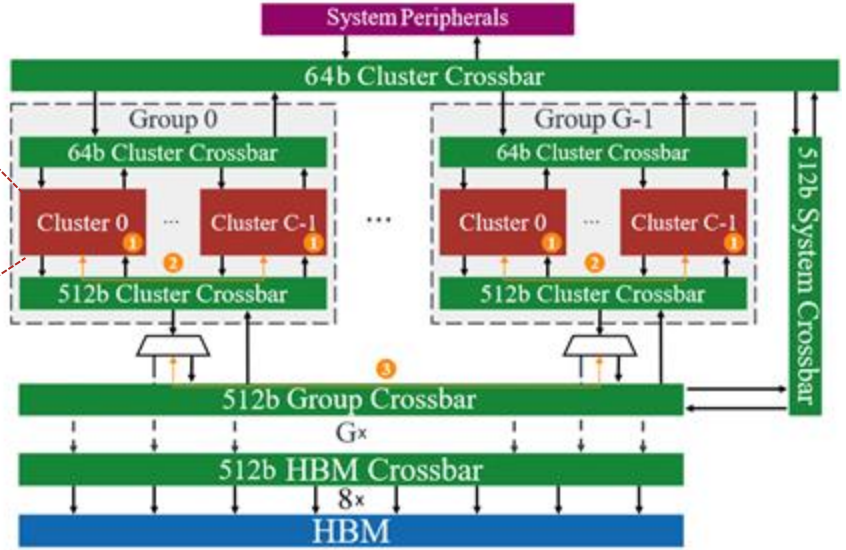
8× RISC-V RV32IMAFD cores

Zaruba, F., Schuiki, F., Hoefler, T., Benini, L., "Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floating-point intensive workloads", *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1845–1860, 2020.

Multi-Snitch Cluster Architecture

- **Snitch clusters can scale into a multi-cluster system, as demonstrated in Occamy**



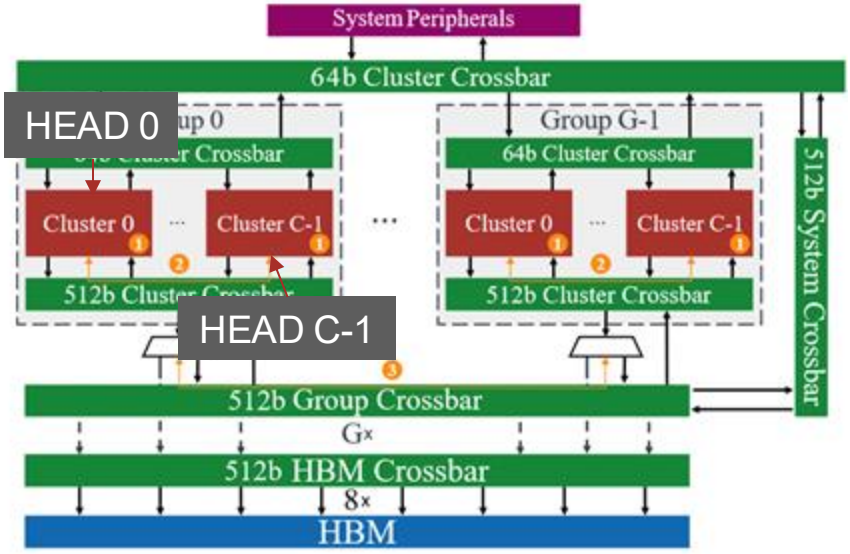


Paulin, G., Scheffler, P., Benz, T., et al., "Occamy: A 432-Core 28.1 DP-GFLOP/s/W 83% FPU Utilization Dual-Chiplet, Dual-HBM2E RISC-V-Based Accelerator for Stencil and Sparse Linear Algebra Computations with 8-to-64-bit Floating-Point Support," In Proc. IEEE Symposium on VLSI Technology and Circuits, 2024.

FlashAttention on Snitch

FlashAttention with Head-Level Parallelism

- **Partial Softmax:**
  - Process Q/K/V in blocks
  - Track row-wise maximum and normalization denominator incrementally
  - Compute the attention output without calculating the full attention matrix
- **Head-Level Tiling across Snitch Clusters:**
  - **Each Snitch cluster handles one attention head**.
  - Compute Q/K/V tiles **of the assigned head**



Potocnik, V., Colagrande, L., Fischer, T., et al., "Optimizing Foundation Model Inference on a Many-tiny-core Open-source RISC-V Platform," IEEE Transactions on Circuits and Systems for Artificial Intelligence, 2024.

Naive Softmax

- **Basic softmax implementation on Snitch**
  - No SSR, FREP, or packed SIMD instructions

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

Max	C ArrayMax
Exp	C LUT + Polynomial
Div	C ArrayDiv

Exp from Arm Libmath: SW LUT + Polynomial Approximation 319 cycle/item

Baseline C code

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

Baseline Assembly

```
MAX Loop for N:
flh ft1,0(a2)
fmax.h max,ft1,max
addi a2,a2,2
addi a3,a3,-1
bnez a3, loop
```

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

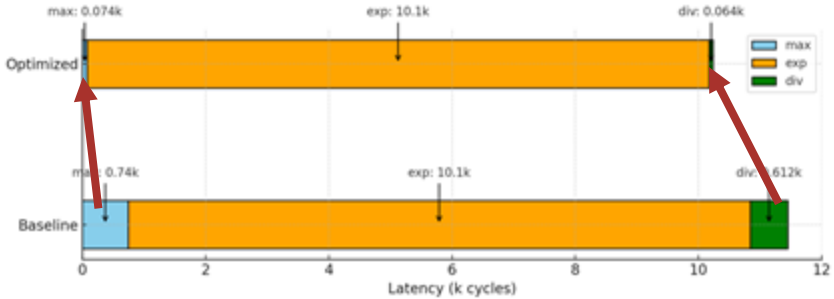
```
EXP Loop for N:
#Initialization
flh      ft0, 0(a0)
fsub.h   ft1, ft0, ft5
...
#Exp approximation
srli   a2, ft1, 20
andi   a2, a2, 2047
bgeu a2,1067,overflow
fmul.d   ft2, const1, ft1
fadd.d   ft2, ft2, const2
fmul.d   ft2, ft2, const3
fcvt.h.d ft1, ft2
#Update y[i], sum
#Solve overflow
...
```

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

```
NORM Loop for N:
flh   ft1, 0(a2)
fdiv.h ft2, ft1, sum
fsh   ft2, 0(a2)
addi a2, a2, 2
addi a3, a3, -1
bnez a3, loop
```

Softmax SW Optim

- **Optimize softmax with SSR, FREP, and packed SIMD instructions**
  - Bottleneck: EXP calculation

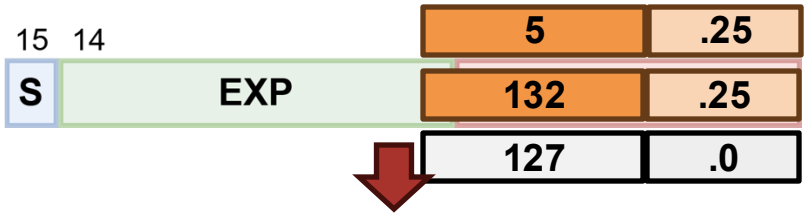


	Baseline	Optim
Max	C ArrayMax	SSR, SIMD, FREP
Exp	C LUT + Poly	Libm exp
Div	C ArrayDiv	**SSR, SIMD, FREP**

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

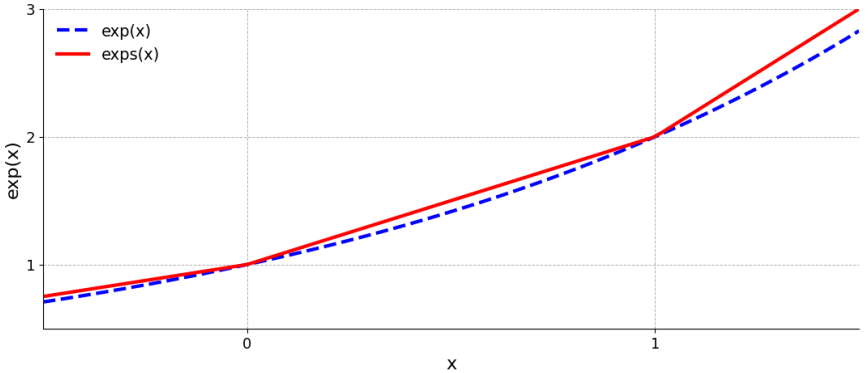
A Fast Exp Approximation – Schraudolph's Method

- **Exploit how floats are stored to approximate the base-2 exp**
  - Add a fixed-point number the bias and replace the exponent with the integer part
- **What happens to the result?**
  - The integer part is **perfectly exponentiated**
  - The fractional part becomes the mantissa
- **We get a linear interpolation of the 2 nearest integer powers of 2**



$$\text{value} = (-1)^0 \cdot 2^{132-127} \cdot (1 + 0.25) = 2^5 \cdot 1.25$$

$$2^x \approx 2^{\text{int}(x)} \cdot \big(1 + \text{frac}(x)\big) \coloneqq \text{exps}(x)$$

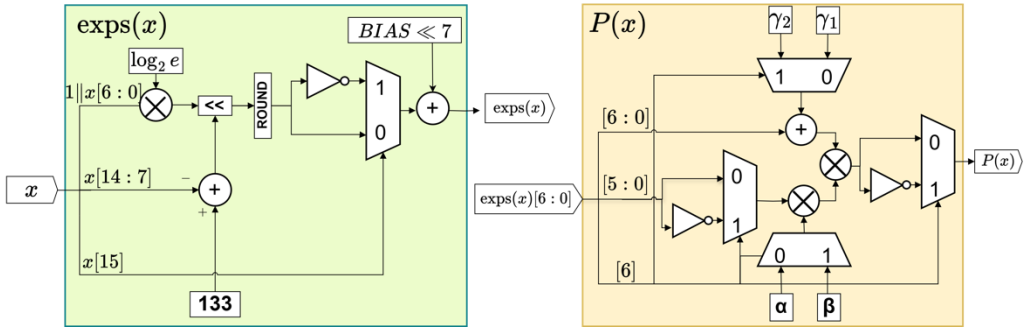


Enhancing Schraudolph's Method

- **Enhance Schraudolph's exponential with an approximate second-order polynomial**
  - Improve the accuracy of the approximation by processing the mantissa only
  - Replace $\mathrm{frac}(x)$ with a polynomial $P\big(\mathrm{frac}(x)\big)$ that approximates $2^{\mathrm{frac}(x)}$
  - Implemented in BF16 format

$$\exp(x) \approx 2^{\mathrm{int}(x')} \cdot \left(1 + P\left(\mathrm{frac}(x')\right)\right). \qquad P(x) = \begin{cases} \alpha x\,(x+\gamma_1)\,, & x \in [0, 0.5), \\ \mathrm{not}\left(\beta\,\mathrm{not}(x) \cdot (x+\gamma_2)\right), & x \in [0.5, 1). \end{cases}$$

- **Exponential accuracy:**
  - Avg 99.86%
  - Min 99.25%

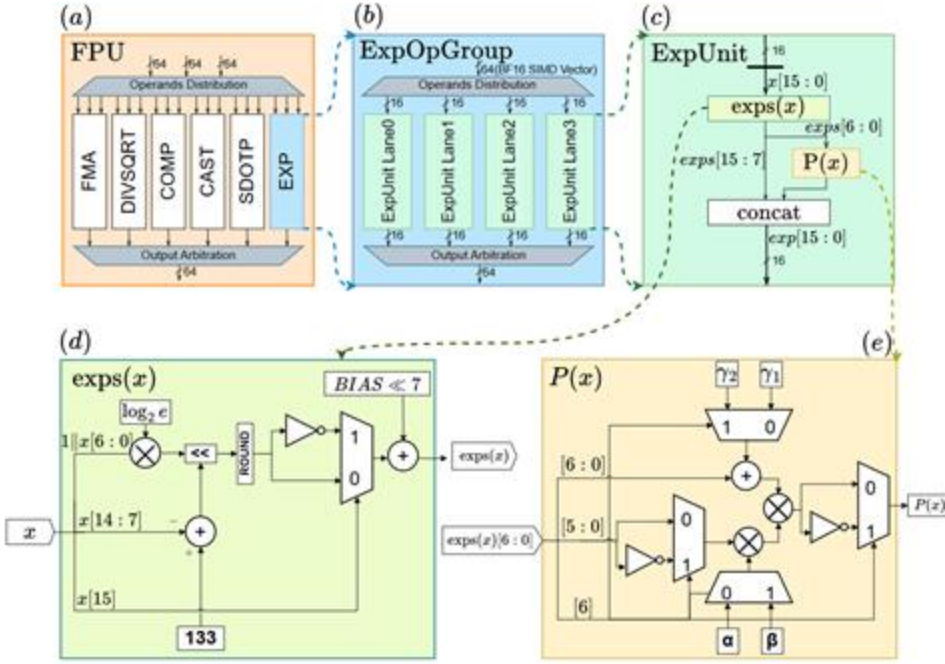


Belano, A., Tortorella, Y., Garofalo, A., et al., "A Flexible Template for Edge Generative AI with High-Accuracy Accelerated Softmax & GELU," arXiv preprint arXiv:2412.06321, 2024.

Integrating EXP Into the Snitch FPU

Snitch FPU Integration

- **Modular, multi-format FPU**
- **ExpOpGroup:**
  - Accepts 64-bit vectors (4×BF16)
  - **4 ExpUnit** lanes, 2-cycle latency, pipelined
- **2 new instructions:**
  - **FEXP**: Scalar (1 active lane)
  - **VFEXP**: packed SIMD (4 active lanes)



Softmax with VFEXP

- **Replace the libm exp with the new VFEXP instruction**

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

Loop Unrolling, Div->Mul

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

```
// Step 1: Find the maximum value
// Step 2: exponentials and sum
ssr config ft1 read double, ft2 write
asm
"frep.o %[n_frep], 4"
"vfsub.ah ft3, ft1, %[max]"
"vfexp.ah ft2, ft3"
"vfadd.ah ft4, ft2"

// Step 3: Normalize
```

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;
}
```

VFMAX, VFDIV

```
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")
```

VFEXP

```
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")
```

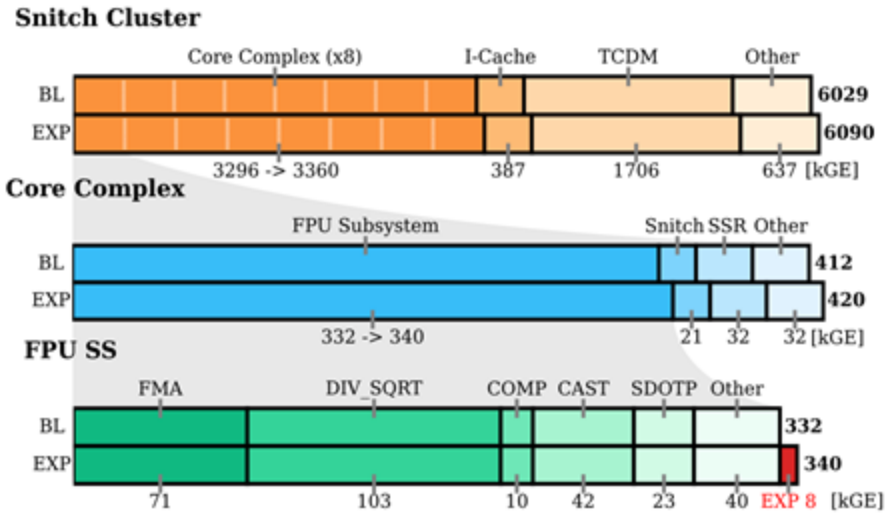
Loop Unrolling

```
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"
"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"
.....
```

350 Cycles/N → 329 Cycles/N → 4 Cycles/N → **2.125 Cycles/N**

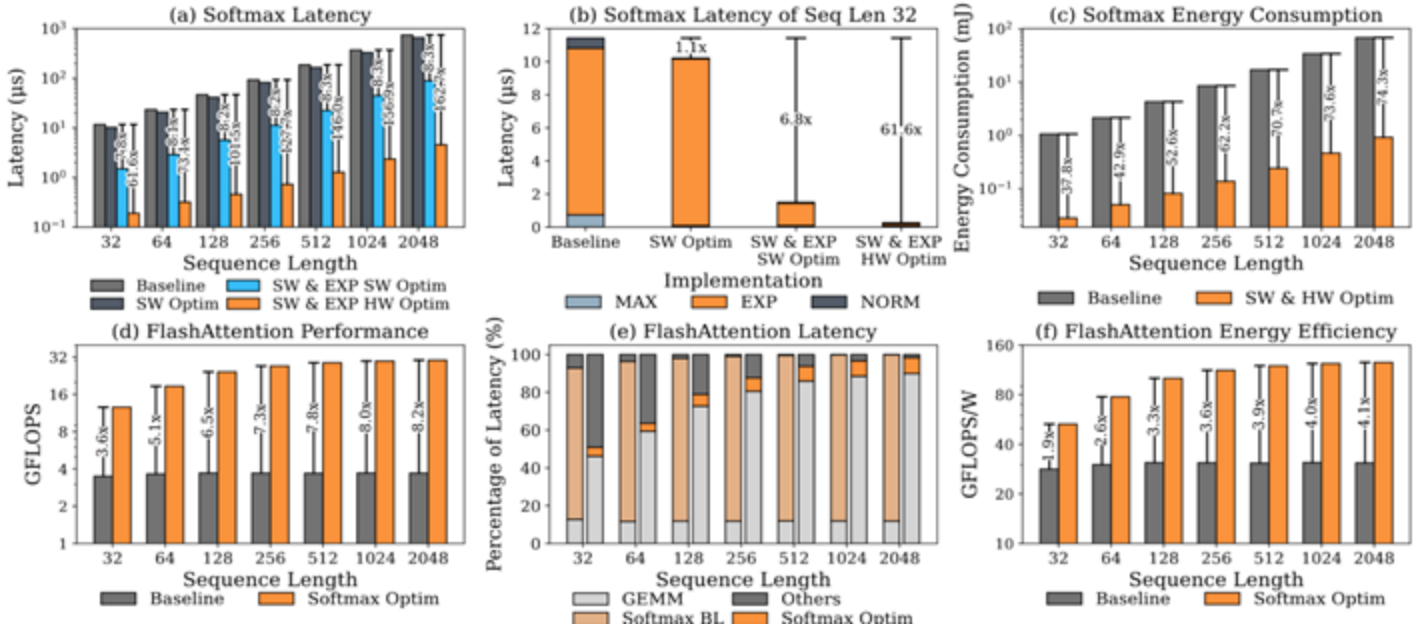
Backend Results: Timing, Area, Power

- **Technology**: GlobalFoundries 12LP+
- **Core**: Snitch Core Complex with Exp Group
- **Timing**: No degradation with Snitch CC (1.15 GHz @ TT, 941 MHz @ SS)
- **Area Overhead**: +1.0% of Snitch Cluster
- **Power Overhead**: +1.8% at peak GEMM utilization
- **Enegy per Exponentiation**:
  W/o VFEXP: 3433 pJ/op
  W/ VFEXP: 6.39 pJ/op



Softmax & Attention Performance and Energy

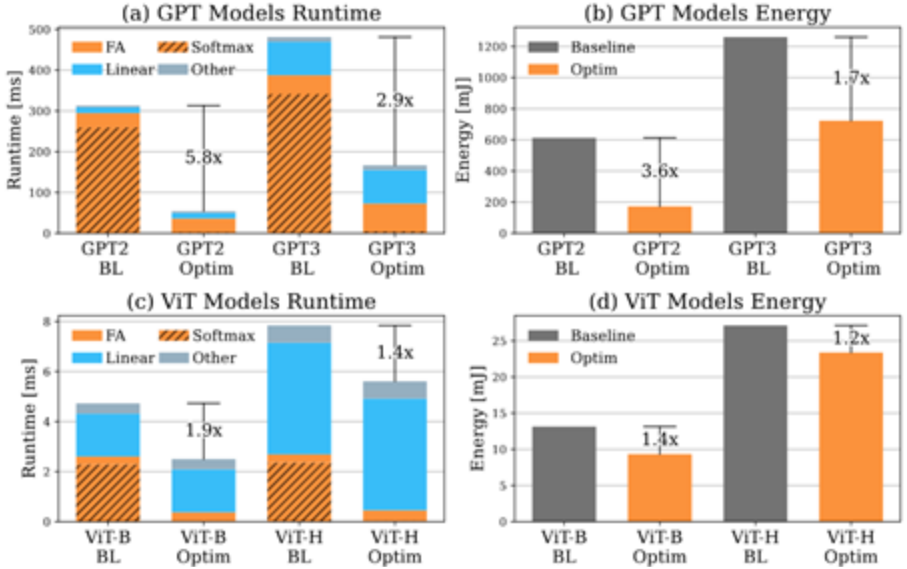
- **Softmax kernel: up to 162.7× latency reduction and 74.3× less energy consumption**
- **FlashAttention-2: up to 8.2× faster and 4.1× less energy**



Scalability: GPT & ViT

Multi-Snitch Cluster Evaluation

- **16 Snitch clusters** configuration
- Tested on **GPT2 Small** and **ViT-Base**
- **Performance**:
  - Up to **5.8× latency reduction**
  - Up to **3.6× energy reduction**
- **Accuracy loss** within the margin of error



Model	Dataset	Metric	FP32	BF16	BF16 EXP
GPT-2	WikiText	Perplexity (↓)	37.4	37.8	37.8
	ArcEasy	Accuracy (↑)	43.8	42.9	43.7
ViT-B	ImageNet	Accuracy (↑)	80.3	80.3	80.3
	CIFAR-10	Accuracy (↑)	98.5	98.5	98.5

Potocnik, V., Colagrande, L., Fischer, T., et al., "Optimizing Foundation Model Inference on a Many-tiny-core Open-source RISC-V Platform," IEEE Transactions on Circuits and Systems for Artificial Intelligence, 2024.

Conclusion

- Designed a lightweight BF16 exponential block with **1.0% area** and **1.8% power** overhead.
- Introduced scalar and SIMD exponential ISA extensions, achieving **162.7× latency** and **74.3× energy** reduction for Softmax.
- Integrated with FlashAttention-2, **achieving 8.2× speedup** and **4.1× energy** reduction.
- Scaled to a **16-cluster** system, up to **5.8× lower latency** and **3.6× energy reduction** with minimal accuracy loss.

Thank you!