

LLM Inference on RISC-V Embedded CPUs

Frank Yueh-Feng Lee , Yi-Jui Chu, Chih-Chung Huang, Heng-Kuan Lee
Andes Technology

Outline



- SW Frameworks for LLM Inference on RISC-V
- LLM Inference on Embedded RISC-V Board
- LLM Inference with RISC-V P Extension
- Optimizing and tuning llama.cpp for RISC-V
- Performance results
- Conclusions

SW Frameworks for LLM Inference on RISC-V



■ Generic vs LLM-specific framework

■ LLM inference

- Matrix / vector multiplication intensive
- New operations, KV cache
- New quantization formats, 4-bit, block-based

■ Llama cpp

- LLM-specific
- Plain C/C++ implementation
- x86 / ARM / RISC-V / GPU ...
- Support RISC-V scalar and V extension
- 2-bit to 8-bit integer quantization formats
- Support state-of-the-art models

- ☑ LLaMA 🐼
- ☑ LLaMA 2 🐼🐼
- ☑ LLaMA 3 🐼🐼🐼
- ☑ [Mistral 7B](#)
- ☑ [Mixtral MoE](#)
- ☑ [DBRX](#)
- ☑ [Falcon](#)
- ☑ [Chinese LLaMA / Alpaca](#) and [Chinese LLaMA-2 / Alpaca-2](#)
- ☑ [Vigogne \(French\)](#)
- ☑ [BERT](#)
- ☑ [Koala](#)
- ☑ [Baichuan 1 & 2](#) + [derivations](#)
- ☑ [Aquila 1 & 2](#)
- ☑ [StarCoder models](#)
- ☑ [Refact](#)
- ☑ [MPT](#)
- ☑ [Bloom](#)
- ☑ [Yi models](#)
- ☑ [StableLM models](#)
- ☑ [Deepseek models](#)
- ☑ [Qwen models](#)
- ☑ [PLaMo-13B](#)
- ☑ [Phi models](#)
- ☑ [GPT-2](#)
- ☑ [Orion 14B](#)
- ☑ [InternLM2](#)
- ☑ [CodeShell](#)
- ☑ [Gemma](#)
- ☑ [Mamba](#)

LLM Inference on Embedded RISC-V Board



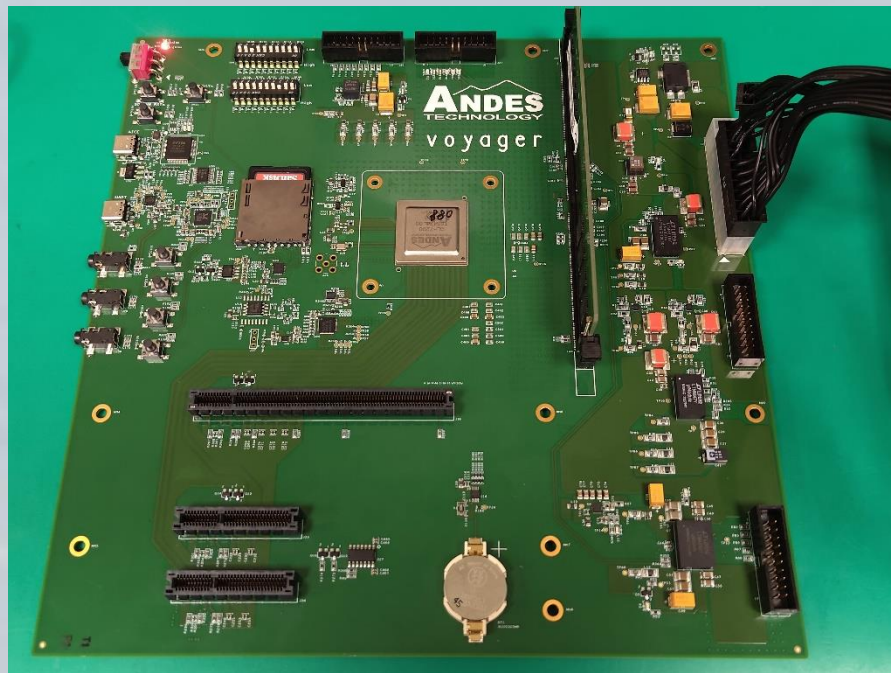
■ QiLai SoC test chip

- AX45MP 4c 2MB L2 (RVP)
- NX27V 1c 512K L1 (RVV VLEN 512)

■ Voyager board

- Micro ATX form factor
- For IP evaluation and internal application development

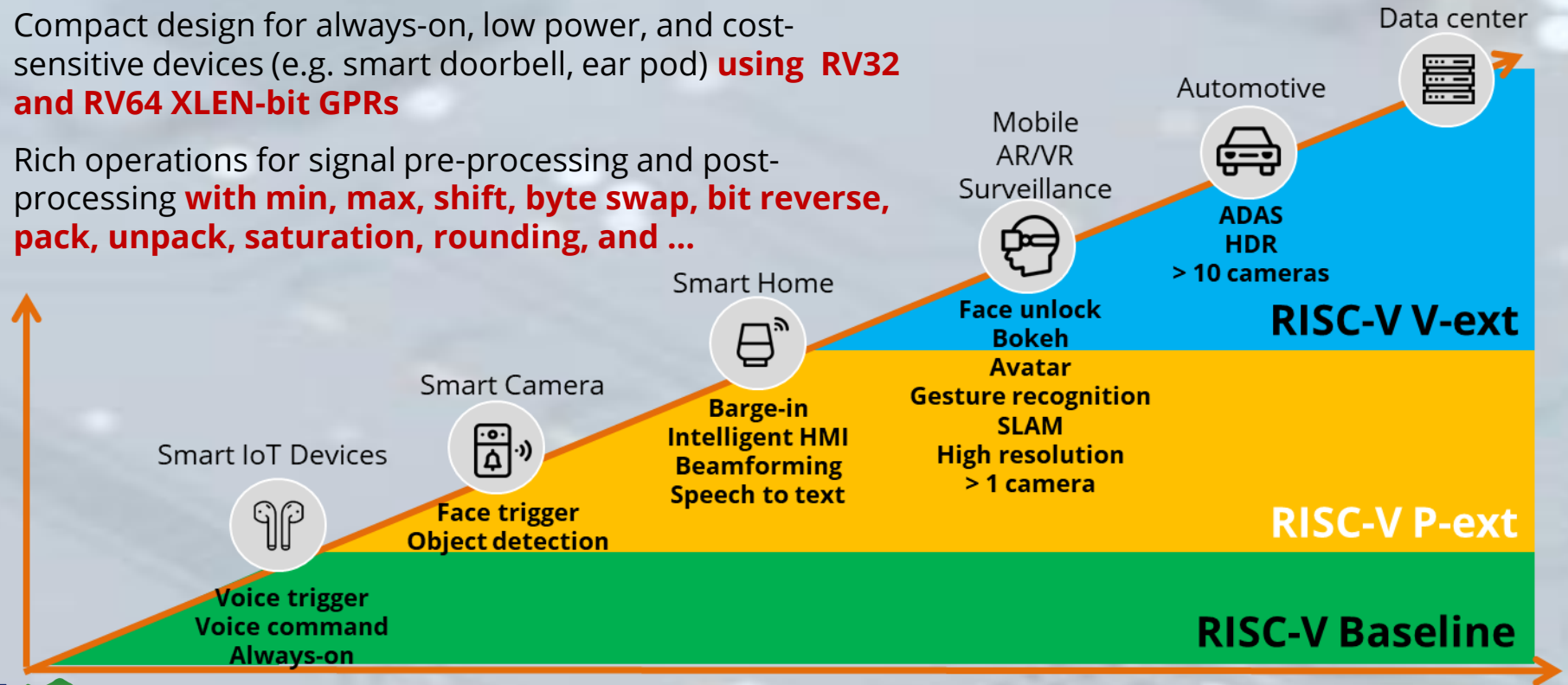
QiLai SOC		
CPU Core	Peripherals	Power
AX45MP @1.6 Ghz, 32 KB L1 I&O cache, 2MB L2 Cache (K4)	UART x2	External Power Supply
NX27V @1.1 Ghz, 32 KB L1 I cache, 512KB L1 D Cache	PWM x2	Clock
	I2C x2	32KHz Xtal (ext.)-For future use
	GPIO-16	25Mhz Xtal (ext.)
	I2S	
	SD	
Memory I/F	System	NOC
DDR4 up to 3200	PLL1- 1.2 Ghz ~2.4Ghz	Arteris Flex NOC
QSPI x2	PLL2 - 1.0 Ghz ~1.5 Ghz	
Memory	PLL3 - 800 Mhz	Host Connectivity
SRAM (512 KB) x4	PLL4 - 800 Mhz (DDR4)	PCIe DM Gen4 x4, PCIe DM Gen4 x2, PCIe DM Gen4 x2,
64 bit DRAM 16 GB	RTC	
	Power Mgmt.	
	WDT	



What RISC-V P-extension Can Bring



- Efficiency for small-scale data parallel computation (e.g. voice/face trigger) **with 8b, 16b, and 32b SIMD**
- Compact design for always-on, low power, and cost-sensitive devices (e.g. smart doorbell, ear pod) **using RV32 and RV64 XLEN-bit GPRs**
- Rich operations for signal pre-processing and post-processing **with min, max, shift, byte swap, bit reverse, pack, unpack, saturation, rounding, and ...**



LLM Inference with RISC-V P Extension



■ Llama.cpp basic computation form

- Micro kernel-based
- Convert GEMM or GEMV to vector dot product (vecdot) by default
- Various types of vecdot micro-kernels

■ RVP advantages

- Lower-cost
- vecdot friendly instruction

■ RVP LLM challenges

- SIMD width up to 64-bits (RV64P)
- Integer data types only: INT8/16/32
- => smaller LLM, 1B

■ Before optimization

- Upstream llama.cpp provides basic RVV intrinsics optimization
- No RVP optimization available
- Voyager AX45MP 4c TinyLlama 1.1B Q8_0 scalar result
 - Prompt: 1.10 token / sec
 - Eval: 0.85 token / sec

```
Sep 23 14:47:50 
```

1 Serial: Silicon Labs 0001  

 /dev/ttyUSB0 (38400)

Change baud rate  Unpin

```
llama_new_context_with_model: KV self size = 44.00 MiB, K (f16): 22.00 MiB, V (f16): 22.00 MiB
llama_new_context_with_model: CPU output buffer size = 0.12 MiB
llama_new_context_with_model: CPU compute buffer size = 148.01 MiB
llama_new_context_with_model: graph nodes = 710
llama_new_context_with_model: graph splits = 1

system_info: n_threads = 4 / 4 | AVX = 0 | AVX_VNNI = 0 | AVX2 = 0 | AVX512 = 0 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | AVX512_BF16 = 0 | FMA = 0 | NEO
N = 0 | SVE = 0 | ARM_FMA = 0 | F16C = 0 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 0 | SSE3 = 0 | SSSE3 = 0 | VSX = 0 | MATMUL_INT8 = 0 | LLAMAFILE = 1
|
sampling:
  repeat_last_n = 64, repeat_penalty = 1.100, frequency_penalty = 0.000, presence_penalty = 0.000
  top_k = 40, tfs_z = 1.000, top_p = 0.950, min_p = 0.050, typical_p = 1.000, temp = 0.000
  mirostat = 0, mirostat_lr = 0.100, mirostat_ent = 5.000

sampling order:
CFG -> Penalties -> top_k -> tfs_z -> typical_p -> top_p -> min_p -> temperature
generate: n_ctx = 2048, n_batch = 512, n_predict = -1, n_keep = 1

Building a website can be done in 10 simple steps:\nStep 1:
```

Optimizing and Tuning llama.cpp for RISC-V



■ Possible approaches

- Vecdot using SIMD instructions
 - RVP: SMAQA: 8-bit multiply with 32-bit add instructions
- Adjust quantization formats
 - 4-bit, 8-bit, or hybrid
- Adjust KV cache parameters
- GEMM / GEMV using SIMD instructions
- Custom block format

■ => Keep Perplexity (PPL) as low as possible

<https://github.com/jzhang38/TinyLlama>

<https://github.com/jzhang38/TinyLlama>

5.09x
4.40x

```

1 Serial: Silicon Labs 0001
+ 
+ /dev/ttyUSB0 (38400)
Change baud rate Unpin

llama_new_context_with_model: KV self size = 88.00 MiB, K (f32): 44.00 MiB, V (f32): 44.00 MiB
llama_new_context_with_model: CPU output buffer size = 0.12 MiB
llama_new_context_with_model: CPU compute buffer size = 148.01 MiB
llama_new_context_with_model: graph nodes = 710
llama_new_context_with_model: graph splits = 1

system_info: n_threads = 4 / 4 | AVX = 0 | AVX_VNNI = 0 | AVX2 = 0 | AVX512 = 0 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | AVX512_BF16 = 0 | FMA = 0 | NEO
= 0 | SVE = 0 | ARM_FMA = 0 | F16C = 0 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 0 | SSE3 = 0 | SSSE3 = 0 | VSX = 0 | MATMUL_INT8 = 0 | LLAMAFILE = 1
|
sampling:
  repeat_last_n = 64, repeat_penalty = 1.100, frequency_penalty = 0.000, presence_penalty = 0.000
  top_k = 40, tfs_z = 1.000, top_p = 0.950, min_p = 0.050, typical_p = 1.000, temp = 0.000
  mirostat = 0, mirostat_lr = 0.100, mirostat_ent = 5.000

sampling order:
CFG -> Penalties -> top_k -> tfs_z -> typical_p -> top_p -> min_p -> temperature
generate: n_ctx = 2048, n_batch = 512, n_predict = -1, n_keep = 1

Building a website can be done in 10 simple steps:\nStep 1:

```

Operation latency and Perplexity

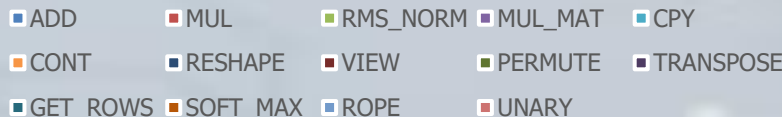
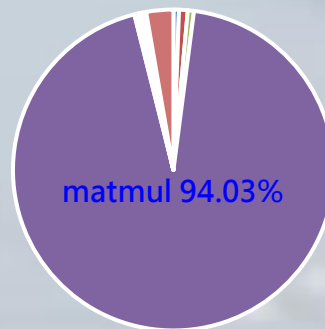
TinyLlama:

<https://github.com/jzhang38/TinyLlama>



Operation	Percentage
ADD	0.56%
MUL	0.88%
RMS_NORM	0.62%
MUL_MAT	94.03%
CPY	0.32%
CONT	0.07%
RESHAPE	0.00%
VIEW	0.10%
PERMUTE	0.05%
TRANSPOSE	0.03%
GET_ROWS	0.02%
SOFT_MAX	0.32%
ROPE	0.39%
UNARY	2.73%
Total	100%

RVP Optimized



Perplexity (PPL)	RVP Optimized	x86 Reference
8-bit	8.4508 (+/- 0.05333)	8.4504 (+/- 0.05332)
4-bit	8.8354 (+/- 0.05577)	8.8347 (+/- 0.05577)

PPL: lower is better

Conclusions



- Andes Voyager board with AX45MP 4c 1.6 GHz RVP optimization
- llama.cpp TinyLlama 1.1B
 - Prompt: 8.29 token / sec
 - Evaluation: 4.80 token / sec
- llama.cpp Llama 2 7B
 - Prompt: 1.27 token / sec
 - Evaluation: 0.90 token / sec
- RVP 4c 1.6 GHz is capable of running 1B LLM model
- 7B+ models requires more powerful platforms (RVV + multicore)



Thank You !!