芯来科技
NUCLEI

芯来科技 舒卓

芯来科技简介

芯来科技
NUCLEI

- 芯来科技成立于2018年，是中国大陆本土专业的**RISC-V IP、子系统IP及SoC解决方案**提供商，赋能下游各类应用场景

5G通信
工业控制
AI
汽车电子
物联网
存储
MCU
网络安全

- 总部位于**上海**，在**华北、华东、华中**及**华南**均能够提供本地技术支持

北京办公室
武汉办公室
武汉市洪山区木香路2号领创楼
上海办公室
上海市浦东新区张江路505号
展想中心8楼
深圳办公室

- 从0到1自主研发**全系列RISC-V CPU IP**，拥有一流的特性和优势

丰富的可配置性
稳健的编码格式
完整的信息安全方案
良好的扩展性
健全的功能安全方案
灵活的商业模式及技术支持

- **累计出货量达数亿颗**，是国内本土RISC-V IP领军企业

>300家正式授权客户

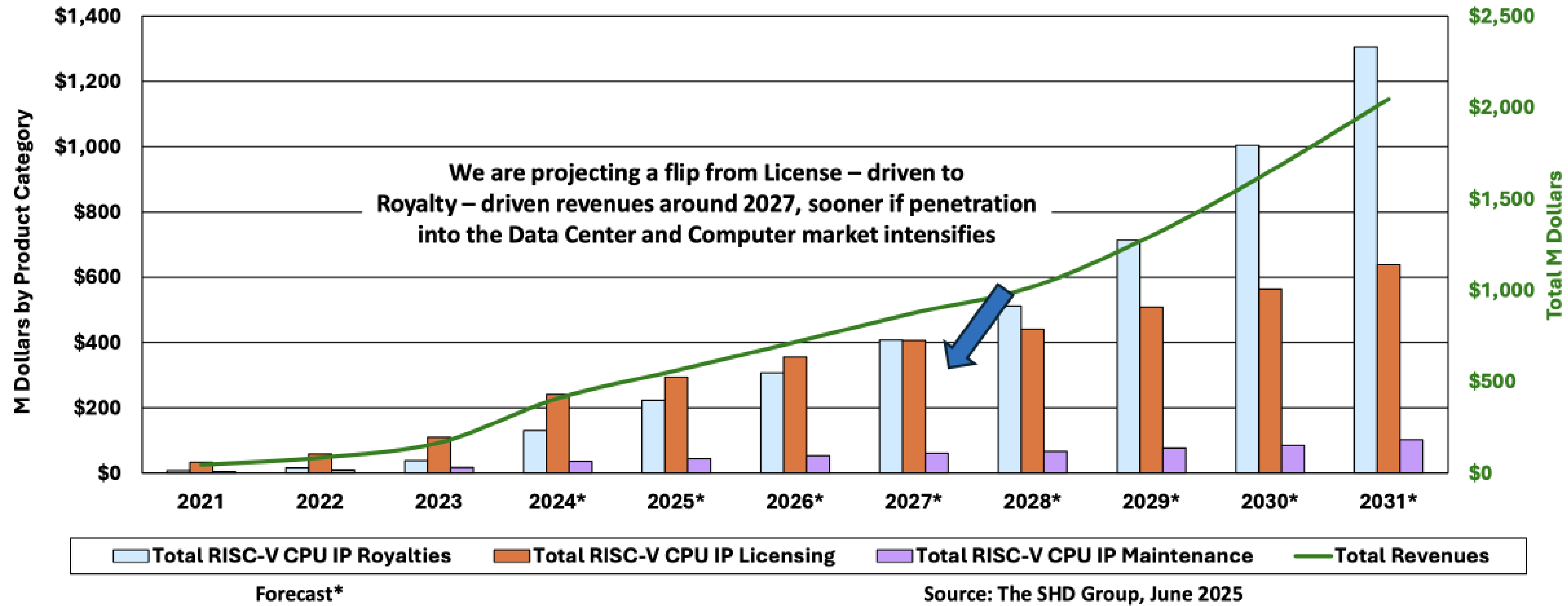
加特兰
LOGIC
安路科技
Amlogic 晶晨半导体
ASR
翱捷
GigaDevice
兆易创新
中国移动 China Mobile
方寸微电子
VeriSilicon
芯原
FISILINK
飞思灵
AISTARTEK
智芯科
WITINMEM
ChipIntelli
启英泰伦
GeoforceChip
BinarySemi
二进制
SILERGY
矽力杰
西南集成
CHIPWAYS
琪埔维
JOULWATT
杰华特

Public@2025 Nuclei System Technology Co., Ltd.

芯来科技，从本土创业，从零开始，跻身行业第一梯队

芯来科技
NUCLEI

RISC-V IP Market Will Continue Accelerating



Forecast*

Source: The SHD Group, June 2025

RISC-V IP Market Leaders (alphabetical order)

- Andes – Worldwide market share leader based on current estimates
- Codasip
- DIY (home-grown RISC-V)
- MIPS
- Nuclei Systems –– **China Market Share Leader based on current estimates**
- SiFive

Newer entrants into RISC-V IP (alphabetical order)

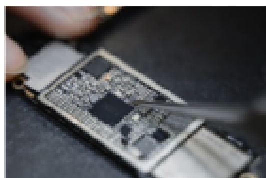
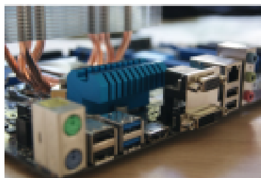
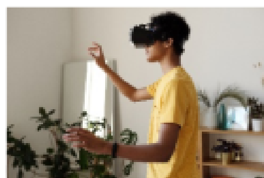
- Akeana
- Condor Computing
- Synopsys
- Tenstorrent
- Ventana

中国地区Market Share Leader

Public@2025 Nuclei System Technology Co., Ltd.

芯来科技RISC-V处理器IP产品图

芯来科技
NUCLEI

	通用处理器产品线				专用处理器产品线		
	N 级别 32位架构 MCU, AIoT, 安全	**U 级别** 32位架构+MMU Linux, 边缘计算	**NX 级别** 64位架构 存储, AR/VR	**UX 级别** 64位架构+MMU Linux, 数据中心, 网络	**NS 级别** 高安全性场景, 金融支付 SIM卡，物联网安全	**NA 级别** ISO26262功能安全 汽车电子	**NI 级别** 人工智能，自动驾驶 通信计算，视频处理
							
1000 系列 Out-of-Order 3/4/6-Wide Decode				UX1000 (SMP)		NA1000	NI1000
900 系列 9-Stage Pipeline Dual-Issue	N900 (SMP)	U900 (SMP)	NX900 (SMP)	UX900 (SMP)		NA900	NI900
600 系列 6-Stage Pipeline Single-Issue	N600 (SMP)	U600 (SMP)	NX600 (SMP)	UX600 (SMP)	NS600		
300 系列 3-Stage Pipeline Single/Dual-Issue	N300				NS300	NA300	
200 系列 2-Stage Pipeline Single-Issue	N200						
100 系列 2-Stage Pipeline Single-Issue	N100				NS100		

Public@2025 Nuclei System Technology Co., Ltd.

Contents：

- 背景介绍
- Nuclei AI Library
- Nuclei BF16 扩展
- Nuclei 矩阵扩展

Public@2025 Nuclei System Technology Co., Ltd.

背景介绍

芯来科技
NUCLEI

嵌入式 AI 框架： 由于嵌入式资源算力有限，所以嵌入式常采用 **训练-推理分离**，即在服务器(多核CPU或GPU)上训练模型，然后在嵌入式设备执行模型推理。

硬件架构：嵌入式 AI 通常采用**通用+专用**架构。

专用：专用的DSA或NPU，算力强，但不灵活

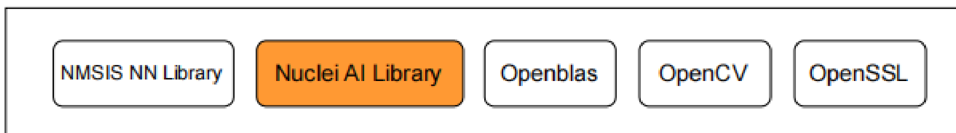
通用：使用RISC-V V扩展加速，可随算子演进而升级

简而言之，业内一般采用NPU来进行专用加速，用VPU进行通用加速。

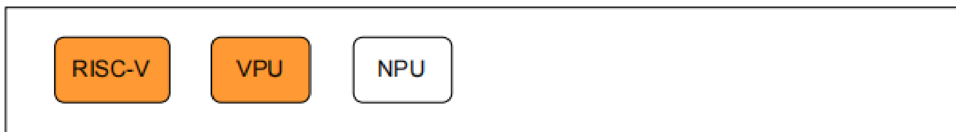
模型训练

模型部署

库及算子



硬件支持



Public@2025 Nuclei System Technology Co., Ltd.

Nuclei AI Library

芯来科技
NUCLEI

为了方便开发者使用VPU加速算子，Nuclei AI Library 对常见的AI算子都进行了RVV优化：

- 已优化实现数十种常用 AI 算子的 RVV 加速，覆盖 int8/int16/fp16/bf16/fp32 等多种数据格式
- 提供不同运行环境支持，可在 RISC-V 的裸机、RTOS 或者 Linux 环境下运行

算子列表（局部）
MatMul
BatchNormalization
Con2D
Elu
Clamp
Cos
Exp
Erf
Rsqrt
非标准或自定义算子

参考链接：

- **Nuclei AI Library**: 使用 V 扩展优化常见的AI算子
  链接：https://github.com/Nuclei-Software/nuclei-ai-library
- **Nuclei NN Library**：基于 CMSIS-NN 进行深度 P/V 扩展优化
  链接：https://github.com/Nuclei-Software/NMSIS
- **Nuclei AI demo**：基于 Nuclei SDK 适配，上手简单，并提供丰富示例
  链接：https://github.com/Nuclei-Software/npk-tflm
  https://github.com/Nuclei-Software/npk-tinymaix

Public@2025 Nuclei System Technology Co., Ltd.

在Nuclei Evalsoc上实测的提升效果-Nuclei AI Library

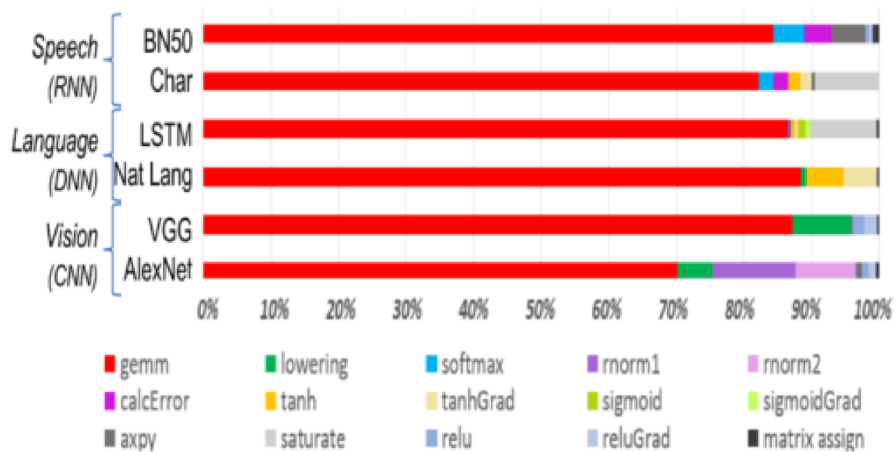
芯来科技
NUCLEI

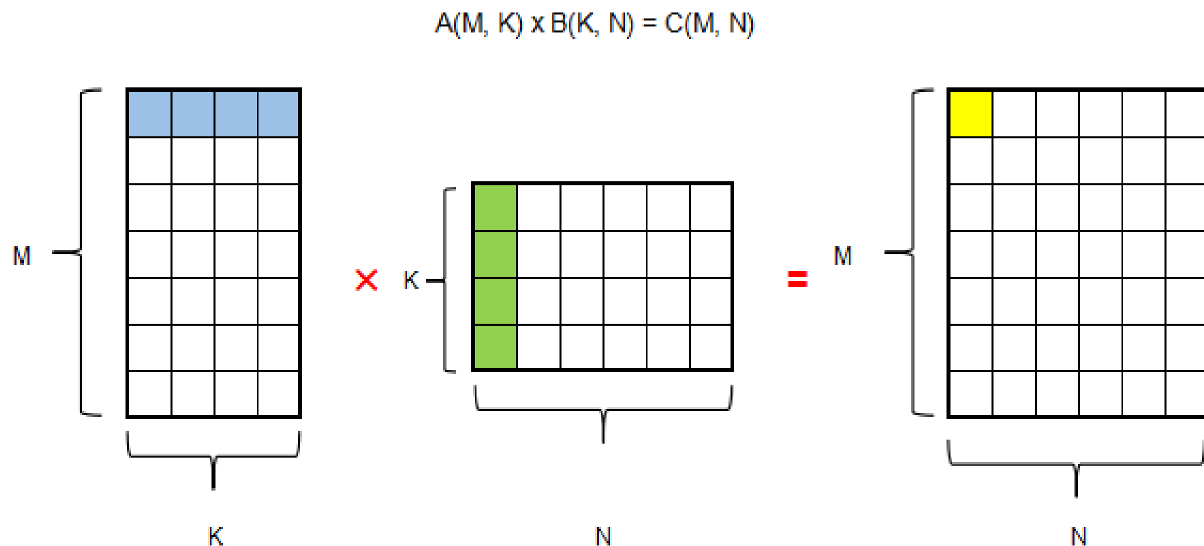
下图在 Nuclei NI900fdv 上实测的RVV提升倍数（局部）：

Public@2025 Nuclei System Technology Co., Ltd.

使用 V 扩展优化GEMM算子

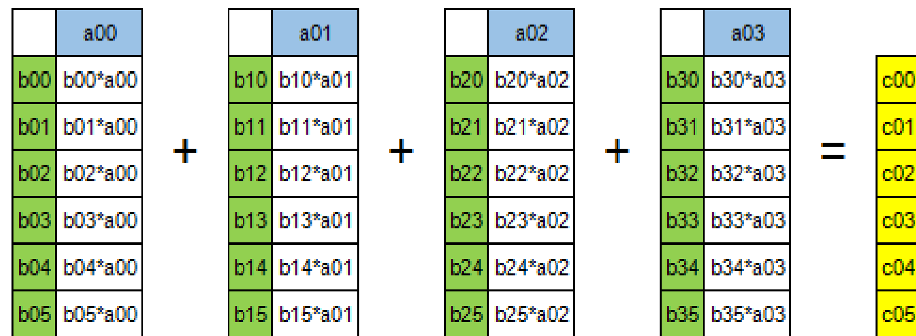
GEMM算子：即通用矩阵乘，右图引用的是IBM的一项研究，统计不同算子在模型中的占比，可见GEMM 是算力占比最高的算子。





使用RVV对GEMM算子进行优化：

1. 应尽量避免使用Reduction 这种效率较低的指令；
2. 充分"榨取"已经load的数据，减少 load 操作；
3. 尽量用满V数据寄存器



Public@2025 Nuclei System Technology Co., Ltd.

在Nuclei Evalsoc上实测的提升效果-GEMM

芯来科技
NUCLEI

下图是在 Nuclei nx900fdv 上实测 GEMM 算子 RVV 优化提升倍数：

```
/* ch = 7, mul = 4 */
for (jj = m / 7; jj > 0; jj--) {
    px = c;
    pInB = b;

    for (ii = n; ii > 0; ii -= l) {
        l = __riscv_vsetvl_e32m4(ii);

        pInA = a;

        vres0m4 = __riscv_vfmv_v_f_f32m4(0.0, l);
        vres1m4 = __riscv_vmv_v_v_f32m4(vres0m4, l);
        vres2m4 = __riscv_vmv_v_v_f32m4(vres0m4, l);
        vres3m4 = __riscv_vmv_v_v_f32m4(vres0m4, l);
        vres4m4 = __riscv_vmv_v_v_f32m4(vres0m4, l);
        vres5m4 = __riscv_vmv_v_v_f32m4(vres0m4, l);
        vres6m4 = __riscv_vmv_v_v_f32m4(vres0m4, l);
        for (kk = 0; kk < k; kk++) {
            va0m4 = __riscv_vle32_v_f32m4(pInB + kk * ldb, l);
            vres0m4 = __riscv_vfmacc_vf_f32m4(vres0m4, *(pInA + 0), va0m4, l);
            vres1m4 = __riscv_vfmacc_vf_f32m4(vres1m4, *(pInA + lda), va0m4, l);
            vres2m4 = __riscv_vfmacc_vf_f32m4(vres2m4, *(pInA + 2 * lda), va0m4, l);
            vres3m4 = __riscv_vfmacc_vf_f32m4(vres3m4, *(pInA + 3 * lda), va0m4, l);
            vres4m4 = __riscv_vfmacc_vf_f32m4(vres4m4, *(pInA + 4 * lda), va0m4, l);
            vres5m4 = __riscv_vfmacc_vf_f32m4(vres5m4, *(pInA + 5 * lda), va0m4, l);
            vres6m4 = __riscv_vfmacc_vf_f32m4(vres6m4, *(pInA + 6 * lda), va0m4, l);
            pInA++;
        }
        __riscv_vse32_v_f32m4(px, vres0m4, l);
        __riscv_vse32_v_f32m4(px + ldc, vres1m4, l);
        __riscv_vse32_v_f32m4(px + 2 * ldc, vres2m4, l);
        __riscv_vse32_v_f32m4(px + 3 * ldc, vres3m4, l);
        __riscv_vse32_v_f32m4(px + 4 * ldc, vres4m4, l);
        __riscv_vse32_v_f32m4(px + 5 * ldc, vres5m4, l);
        __riscv_vse32_v_f32m4(px + 6 * ldc, vres6m4, l);
        px += l;
        pInB += l;
    }
    a += lda * 7;
    c += ldc * 7;
}
```

Note:

1. 16x16 表示 A(16, 16) x B(16, 16) = C(16, 16)，等等
2. 上图示例unrool 7次，目的是为了充分利用32个RVV数据寄存器

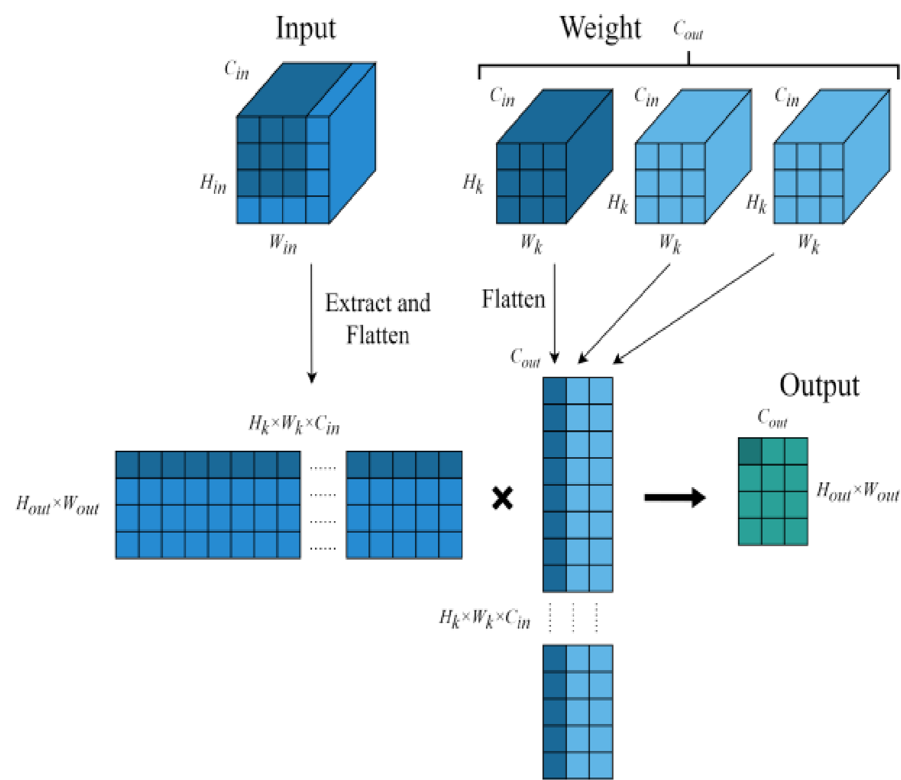
Public@2025 Nuclei System Technology Co., Ltd.

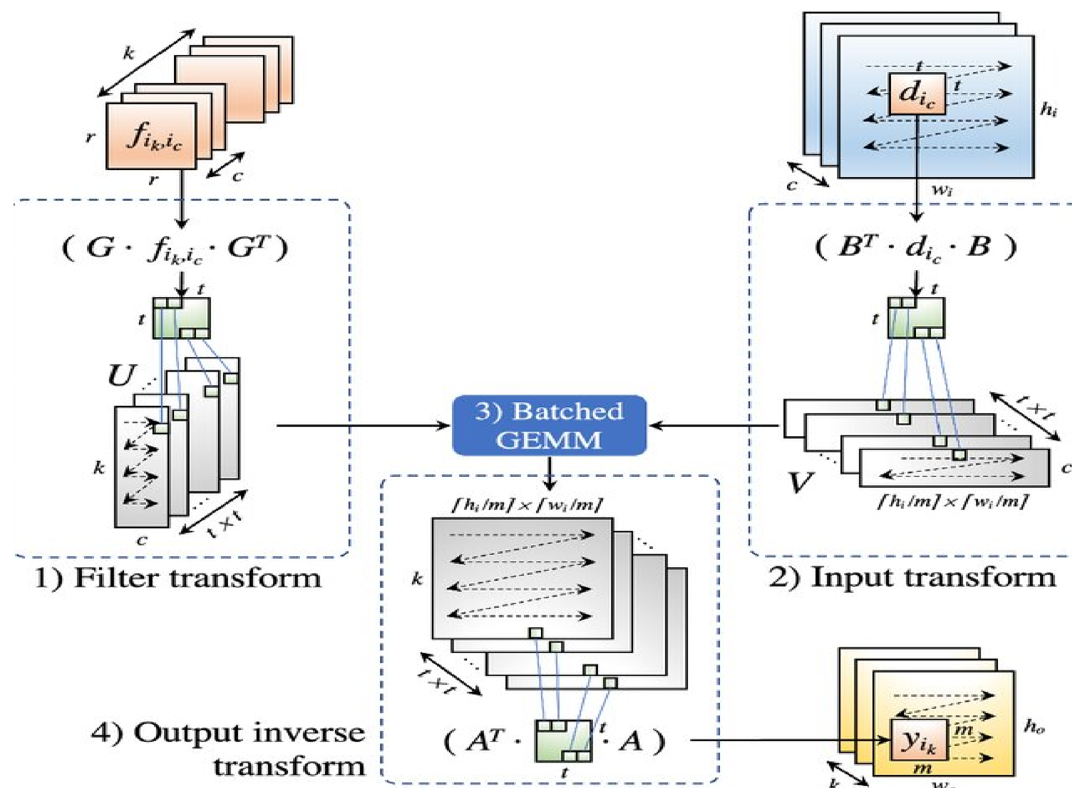
使用 V 扩展优化CON2D算子

CNN 网络中如何高效地进行卷积层计算是提升深度学习推理性能的关键点。

使用RVV优化CONV2D算子常用方法为：

- 方法1： 使用Im2col + GEMM
- 方法2： 使用Winograd + GEMM 加速小尺寸卷积核

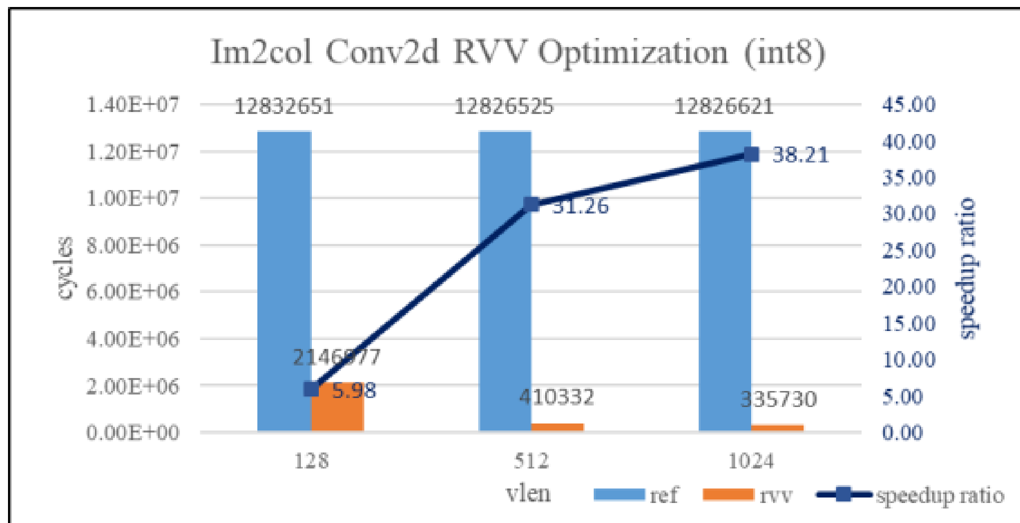


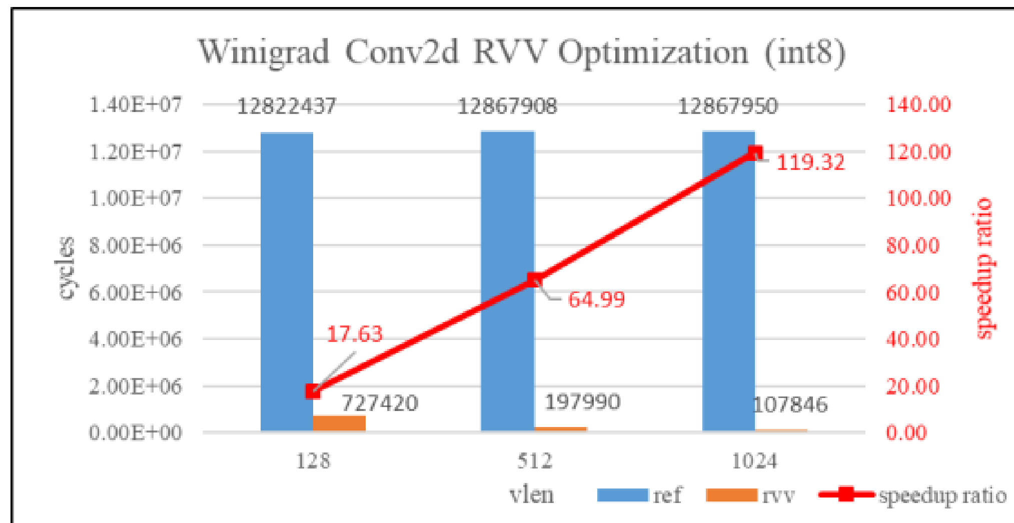


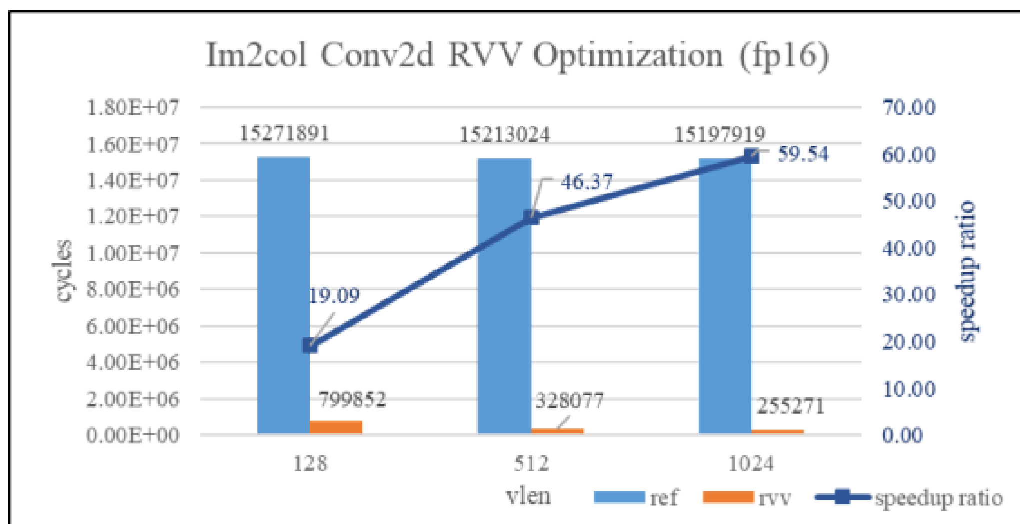
备注：右图摘自《Efficient and portable Winograd convolutions for multi-core processors》

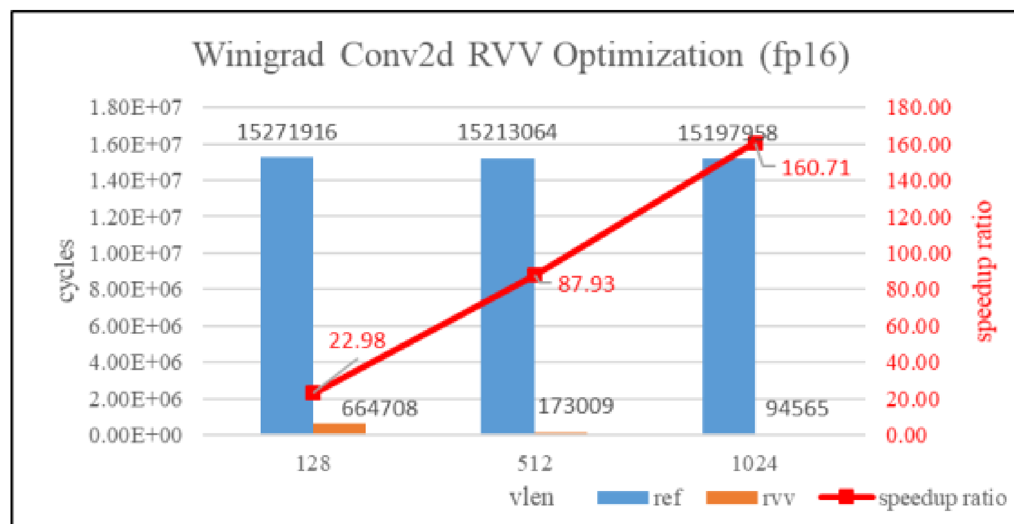
Public@2025 Nuclei System Technology Co., Ltd.

在Nuclei Evalsoc上实测的提升效果-CONV2D算子









Public@2025 Nuclei System Technology Co., Ltd.

Nuclei BF16 扩展

芯来科技
NUCLEI

BFloat16格式：Google提出BFloat16格式（Brain Floating - Point 16），它包含1位符号位，8位指数位，7位尾数位。BFloat16格式有如下特点：

- **动态范围**：BFloat16保留了与FP32相同的8位指数宽度，也即具有与FP32相同的动态范围；
- **精度**：虽然BFloat16精度比FP32有所下降，但在许多深度学习应用场景中，这种降低对结果的影响较小；
- **计算效率**：BFloat16比FP32位宽减半，这样提高内存带宽利用率，如果使用SIMD指令优化意味着计算效率也可成倍提升

RISC-V 官方BF16扩展

Nuclei BF16 扩展

RISC-V官方目前定义了基本的BF16转换指令和向量乘加指令(zvfbfmin扩展)，但需要将BF16转换为FP32后才能进行其他计算，这降低了计算效率和带宽利用率。

Nuclei通过硬件与工具链的协同优化，兼容官方BF16指令；自定义BF16 rvv intrinsic function，生成与FP16相同的指令，通过设置不同寄存器值来切换硬件行为，这样避免了转换，充分发挥了BF16 的算力。

Public@2025 Nuclei System Technology Co., Ltd.

Nuclei BF16 扩展

芯来科技 NUCLEI

Nuclei自定义BF16 扩展有如下特点：

- 生成的BF16指令与F16保持一致，通过CSR寄存器配置来动态决定硬件处理行为；
- 提供专用intrinsic API，完整支持BF16标量和向量运算，充分发挥了 BF16 的算力。

详细可参考：https://doc.nucleisys.com/nuclei_tools/toolchain/gnu/nuclei_bf16.html

```c
/* RISC-V 官方 _zfbfmin_zvfbfmin 扩展 */
void Add_bfloat16_rvv(bfloat16_t *pSrcA, bfloat16_t *pSrcB, bfloat16_t *pDst,
                      uint32_t blockSize)
{
    size_t blkCnt = blockSize;
    size_t vl;
    vbfloat16m4_t vx, vy, vz;
    vfloat32m8_t vx1, vy1, vz1;
    for (; (vl = __riscv_vsetvl_e16m4(blkCnt)) > 0; blkCnt -= vl) {
        vx = __riscv_vle16_v_bf16m4(pSrcA, vl);
        vy = __riscv_vle16_v_bf16m4(pSrcB, vl);
        pSrcA += vl;
        pSrcB += vl;
        vx1 = __riscv_vfwcvtbf16_f_f_v_f32m8(vx, vl);
        vy1 = __riscv_vfwcvtbf16_f_f_v_f32m8(vy, vl);
        vz1 = __riscv_vfadd_vv_f32m8(vx1, vy1, vl);
        vz = __riscv_vfncvtbf16_f_f_w_bf16m4(vz1, vl);
        __riscv_vse16_v_bf16m4(pDst, vz, vl);
        pDst += vl;
    }
}
 return go(f, seed, [])
}
```

```c
/* Nuclei 自定义 _xxlvfbf扩展 */
void Add_bfloat16_rvv(bfloat16_t *pSrcA, bfloat16_t *pSrcB, bfloat16_t *pDst,
                      uint32_t blockSize)
{
    size_t blkCnt = blockSize;
    size_t vl;
    vbfloat16m8_t vx, vy, vz;
    for (; (vl = __riscv_vsetvl_e16m8(blkCnt)) > 0; blkCnt -= vl) {
        vx = __riscv_vle16_v_bf16m8(pSrcA, vl);
        vy = __riscv_vle16_v_bf16m8(pSrcB, vl);
        pSrcA += vl;
        pSrcB += vl;
        vz = __riscv_xl_vfadd_vv_bf16m8(vx, vy, vl);
        __riscv_vse16_v_bf16m8(pDst, vz, vl);
        pDst += vl;
    }
}
```

备注：左图为官方_zfbfmin_zvfbfmin扩展，右图为Nuclei _xxlvfbf 扩展，实测右图比左图性能提升1倍以上

Nuclei 矩阵扩展

矩阵运算在AI应用中占据大量计算资源。为提升矩阵计算性能，Nuclei GCC工具链引入了定制化的VPU扩展Xxlvqmacc：

- 遵循IME group规范设计
- 实现高效的整数矩阵乘加指令，并提供intrinsic functions
- 支持8位整数输入值扩展至32位精度

详细可参考：https://doc.nucleisys.com/nuclei_tools/toolchain/gnu/nuclei_vpu.html

```c
/*
 * Case dose matrix multiply-add like below:
 *   C[j] += A * B[j], for j in [0, vl/16)
 */

void normal_case(int8_t *addr_in1, int8_t *addr_in2, int32_t *addr_out, int32_t data_cnt)
{
  int8_t *pin1 = addr_in1;
  int8_t *pin2 = addr_in2;
  int32_t *pout = addr_out;
  int sum;
  int array_cnt = 0;

  while (data_cnt) {
    for (int32_t ii = 0; ii < 4; ii++) {
      for (int32_t jj = 0; jj < 4; jj++) {
        sum = 0;
        for (int32_t kk = 0; kk < 4; kk++) {
            sum += pin1[ii * 4 + kk] * pin2[kk * 4 + jj];
        }
        pout[ii * 4 + jj] += sum;
      }
    }
    pin2 += 16;
    pout += 16;
    data_cnt -= 16;
  }
}
```

```c
void vpu_case(int8_t *addr_in1, int8_t *addr_in2, int32_t *addr_out, int32_t data_cnt)
{
  int8_t *pin1 = addr_in1;
  int8_t *pin2 = addr_in2;
  int32_t *pout = addr_out;
  size_t vl;

  vint8m1_t vin1;
  vint8m1_t vin2;
  vint32m4_t vout;
  for (; (vl = __riscv_vsetvl_e8m1(data_cnt)) > 0; data_cnt -= vl) {
    vin1 = __riscv_vle8_v_i8m1(pin1, vl);
    vin2 = __riscv_vle8_v_i8m1(pin2, vl);
    vout = __riscv_vle32_v_i32m4(pout, vl);
    vout = __riscv_xl_vqmacc_4x4x4_i32m4(vout, vin1, vin2, vl);
    __riscv_vse32_v_i32m4(pout, vout, vl);
    pin2 += vl;
    pout += vl;
  }
}
```

Public@2025 Nuclei System Technology Co., Ltd.

芯来科技
NUCLEI

芯来科技公众号

芯来科技业务联络

欢迎大家来芯来展台-C18！ 谢谢您！