芯来科技
NUCLEI

RISC-V架构下的音频算法优化实践

芯来科技 裘剑东

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

- 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

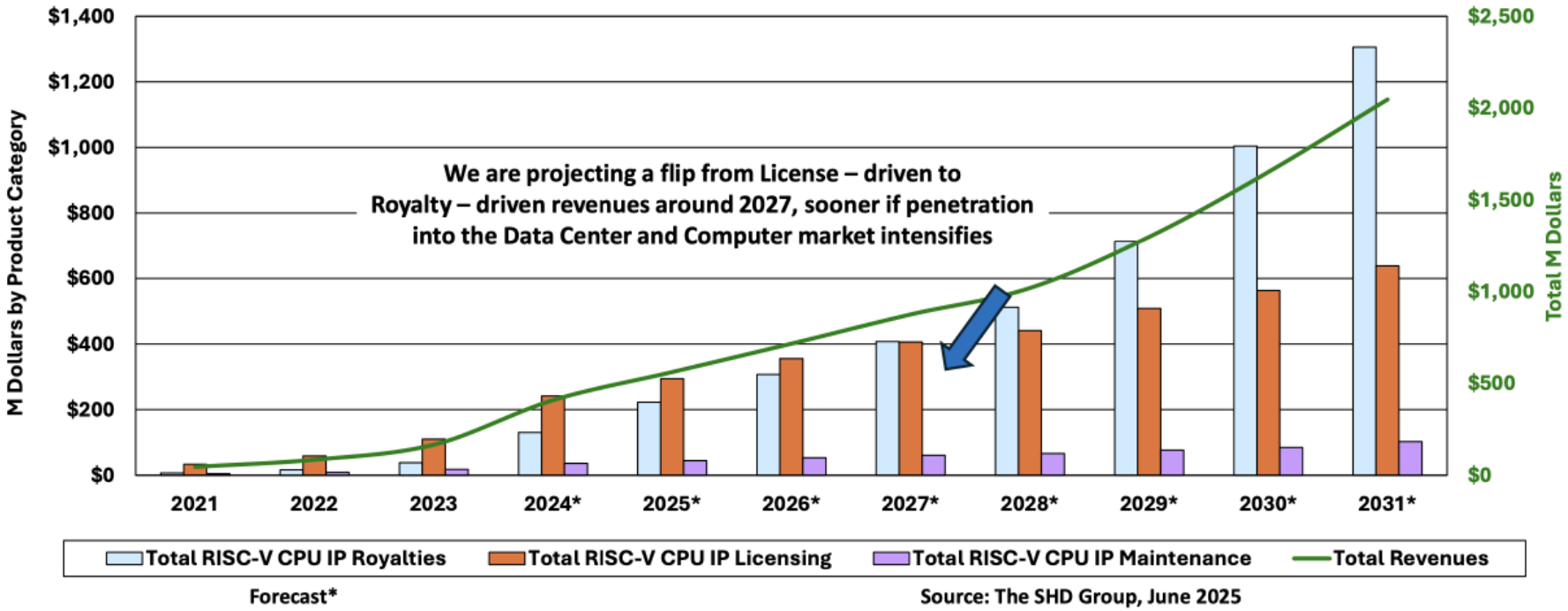
- **累计出货量达数亿颗**,是国内本土RISC-V IP领军企业

>300家正式授权客户

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



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

the SHDgroup

中国地区Market Share Leader

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

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

专用处理器产品线

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		

背景与挑战

芯来科技 NUCLEI

- **RISC-V的崛起与普及：**
  - 开放标准带来的设计自由与生态活力
- **边缘音频处理的核心需求：**
  - 智能设备无处不在，**高质量、低延迟**的音频处理成为刚需
- **当前挑战：**
  - **SIMD 扩展标准化不完整：**RISC-V “P” Extension 仍处于 0.5.4 draft 版本
  - **工具链与编译器优化不足:** 需要手动调用 intrinsic，开发效率低
  - **需要适配的情况繁多：**多种不同的Core和扩展带来维护难度的提升
  - **对性能存疑：**经过适配优化后的算法性能是否与能够达到目前主流方案的水准?

- 各种挑战导致RISC-V与ARM、x86、其它DSP处理器之间，**在音频软件方面存在软件生态的差距！**

常见的语音/音频处理算法

芯来科技 NUCLEI

- **语音（Voice）处理的算法：**ADPCM, AMR-NB, AMR-WB, EVS, Opus
- **音频（Audio）处理的算法：**LC3plus, MP3, SBC, Opus

音频算法	描述
ADPCM	Adaptive Differential Pulse-Code Modulation. IMA-ADPCM codec
AMR-NB	Adaptive Multi-Rate Narrowband Codec
AMR-WB	Adaptive Multi-Rate Wideband. Encoder & Decoder
EVS	Enhanced Voice Services Codec
LC3plus	Low Complexity Communication Codec
MP3	Helix MP3 Decoder
SBC	Sub-Band Codec
SpeexDSP	Noise Reduction, Acoustic Echo Cancellation, Automatic Gain Control ...
Opus	Opus Interactive Audio Codec

- 上述算法均已完成**裸机环境的移植适配**，实现**部分优化**

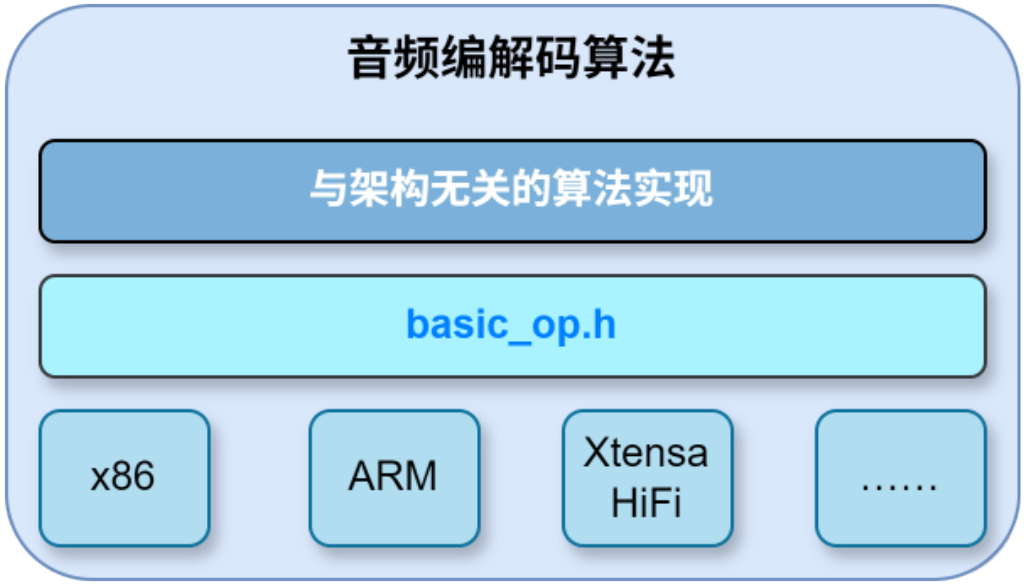
标准化的软件风格

芯来科技 NUCLEI

- 如何让软件算法适用于不同的处理器架构?
- **国际电信联盟电信标准分局**（ITU-T, ITU Telecommunication Standardization Sector）制定标准[1]
  - **G.723.1** Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s
  - **G.Imp723.1** Implementors' Guide for G.723.1
  - **G.729** Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)
  - ……
- 制定的标准实现中将一些基础运算封装成函数隔离上层软件与底层架构，**后续许多算法沿用这一风格**

```
/*-------------------------------------------------------------------------*
 |                                                                         |
 |   Operators prototypes                "basic_op.h"                      |
 |                                                                         |
 *-------------------------------------------------------------------------*/

Word16 sature(Word32 L_var1);             /* Limit to 16 bits,    1 */
Word16 add(Word16 var1, Word16 var2);     /* Short add,           1 */
Word16 sub(Word16 var1, Word16 var2);     /* Short sub,           1 */
Word16 abs_s(Word16 var1);                /* Short abs,           1 */
Word16 shl(Word16 var1, Word16 var2);     /* Short shift left,    1 */
Word16 shr(Word16 var1, Word16 var2);     /* Short shift right,   1 */
Word16 mult(Word16 var1, Word16 var2);    /* Short mult,          1 */
Word32 L_mult(Word16 var1, Word16 var2);  /* Long mult,           1 */
Word16 negate(Word16 var1);               /* Short negate,        1 */
Word16 extract_h(Word32 L_var1);          /* Extract high,        1 */
Word16 extract_l(Word32 L_var1);          /* Extract low,         1 */
Word32 L_mac(Word32 L_var3, Word16 var1, Word16 var2); /* Mac,    1 */
Word32 L_msu(Word32 L_var3, Word16 var1, Word16 var2); /* Msu,    1 */
Word32 L_macNs(Word32 L_var3, Word16 var1, Word16 var2);/* Mac without sat, 1*/
Word32 L_msuNs(Word32 L_var3, Word16 var1, Word16 var2);/* Msu without sat, 1*/
```



[1] ITU-T G series Recommendation: https://www.itu.int/rec/T-REC-G/en

“P” 扩展指令替换

芯来科技 NUCLEI

- 单条指令实现原先多条指令才能完成的运算，**精简指令数**
- 精简后的实现可以内联（inline），进一步**节省函数调用的开销**

指令	描述	计算过程
KHM16 rt,ra,rb [1]	Q15 Signed Saturating Multiply	rt.H[x] = SAT.Q15((ra.H[x]s* rb.H[x]) >> 15) (RV32: x=1..0, RV64: x=3..0)
KSLRAW.u rt, ra, rb [1]	Shift Left Logical with Q31 Saturation or Rounding Shift Right Arithmetic	if (rb[5:0] < 0) rt = RUND(ra s>> -rb[5:0]); if (rb[5:0] > 0) rt = SAT.Q31(ra << rb[5:0]);

KHM16 指令替换

```
static_vo Word16 mult (Word16 var1, Word16 var2)
{
    Word16 var_out;
#if defined(SUPPORT_DSP_STD)
    var_out = __RV_KHM16(var1, var2);
#else
    Word32 L_product;
    L_product = (Word32) var1 *(Word32) var2;
    L_product = (L_product & (Word32) 0xffff8000L) >> 15;
    if (L_product & (Word32) 0x00010000L)
        L_product = L_product | (Word32) 0xffff0000L;
    var_out = saturate (L_product);
#endif
    return (var_out);
}
```

KHM16 intrinsic

KSLRAW.u 指令替换

```
static_vo Word32 L_shr_r (Word32 L_var1, Word16 var2)
{
    Word32 L_var_out;
    if (var2 > 31)
    {
        L_var_out = 0;
    }
    else
    {
#if defined(SUPPORT_DSP_STD)
        L_var_out = __RV_KSLRAW_U(L_var1, -var2);
#else
        L_var_out = L_shr (L_var1, var2);
        if (var2 > 0)
        {
            if ((L_var1 & ((Word32) 1 << (var2 - 1))) != 0)
            {
                L_var_out++;
            }
        }
#endif
    }
    return (L_var_out);
}
```

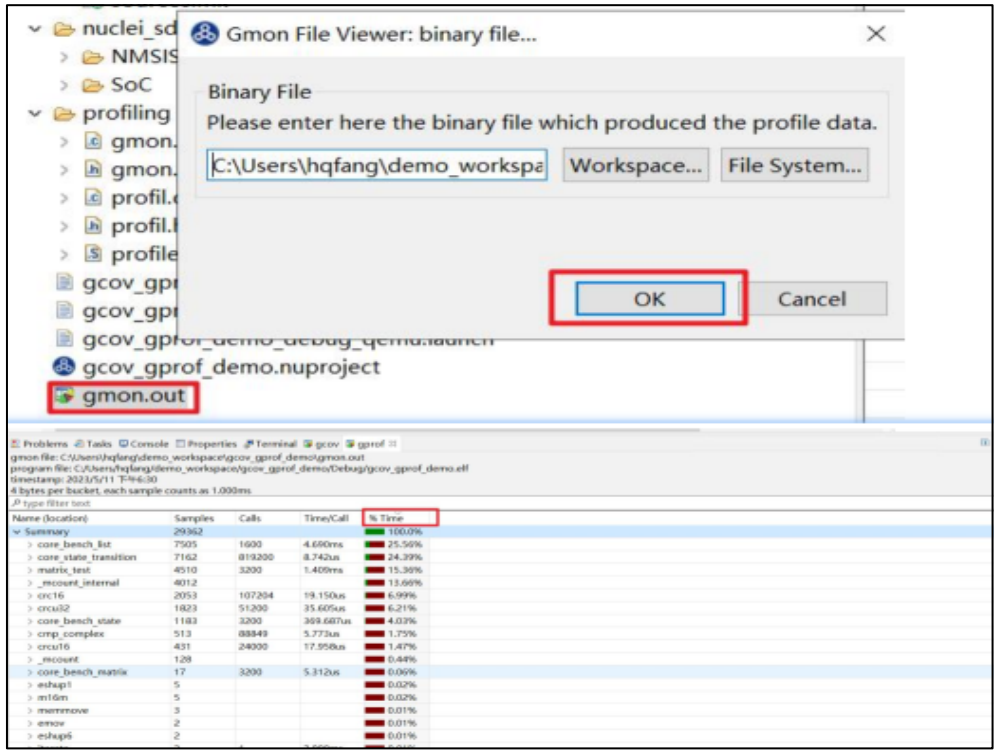
KSLRAW.u intrinsic

[1] riscv/riscv-p-spec: RISC-V Packed SIMD Extension: https://github.com/riscv/riscv-p-spec

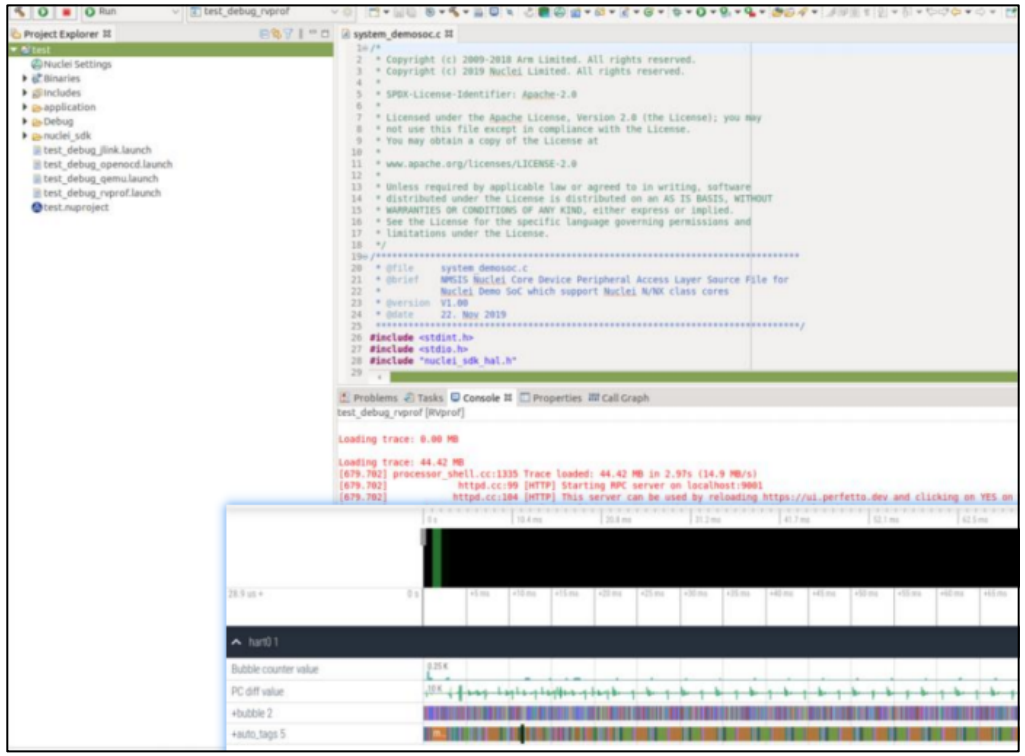
分析定位热点函数

芯来科技 NUCLEI

- 针对 basic_op.h 的优化可能不足以满足需求，需要进一步分析定位**热点函数**
- 利用 **Nuclei Studio** 提供的 **Profiling** 功能进行分析
  - Profiling via **Gprof**：硬件定时采样，结果**精确可靠**
  - Profiling via **Nuclei Model**：模型模拟，操作**方便快捷**



Performance Profiling via Gprof



Performance Profiling via Nuclei Model

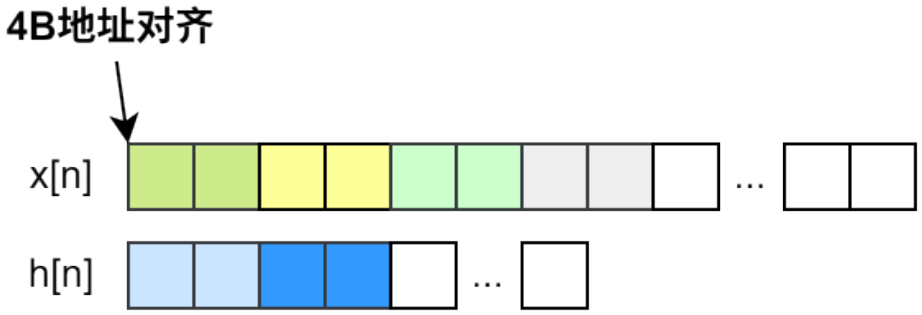
FIR滤波的SIMD实现（1）

芯来科技 NUCLEI

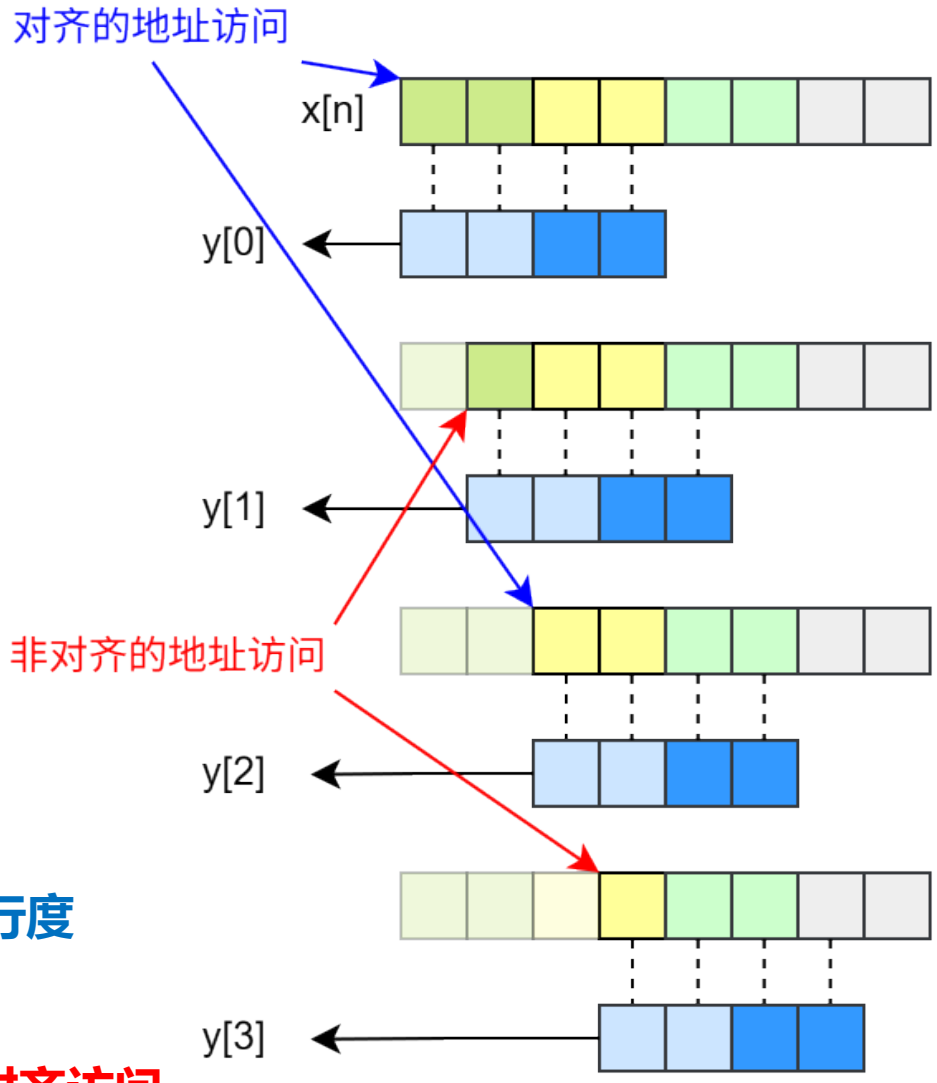
- FIR滤波器在音频算法中非常常见，输出$y[n]$是输入信号$x[n]$和系数$h[n]$的**线性卷积**：

$$y[n] = \sum_{k=0}^{N-1} h[k] \cdot x[n-k]$$

- 音频数据通常为 **int16** 类型



- FIR计算的优化措施
  - 地址对齐的数据采用 **LW** 指令连续加载两个数据
  - 利用 **SMALDA** 同时计算两对 int16 数据的乘累加，**提升并行度**
  - 同时计算 **4 个** 输出结果，**数据复用**
  - 地址非对齐的数据采用 **PKTB16** 拼接两个数据，**避免了非对齐访问**



FIR滤波的SIMD实现（1）

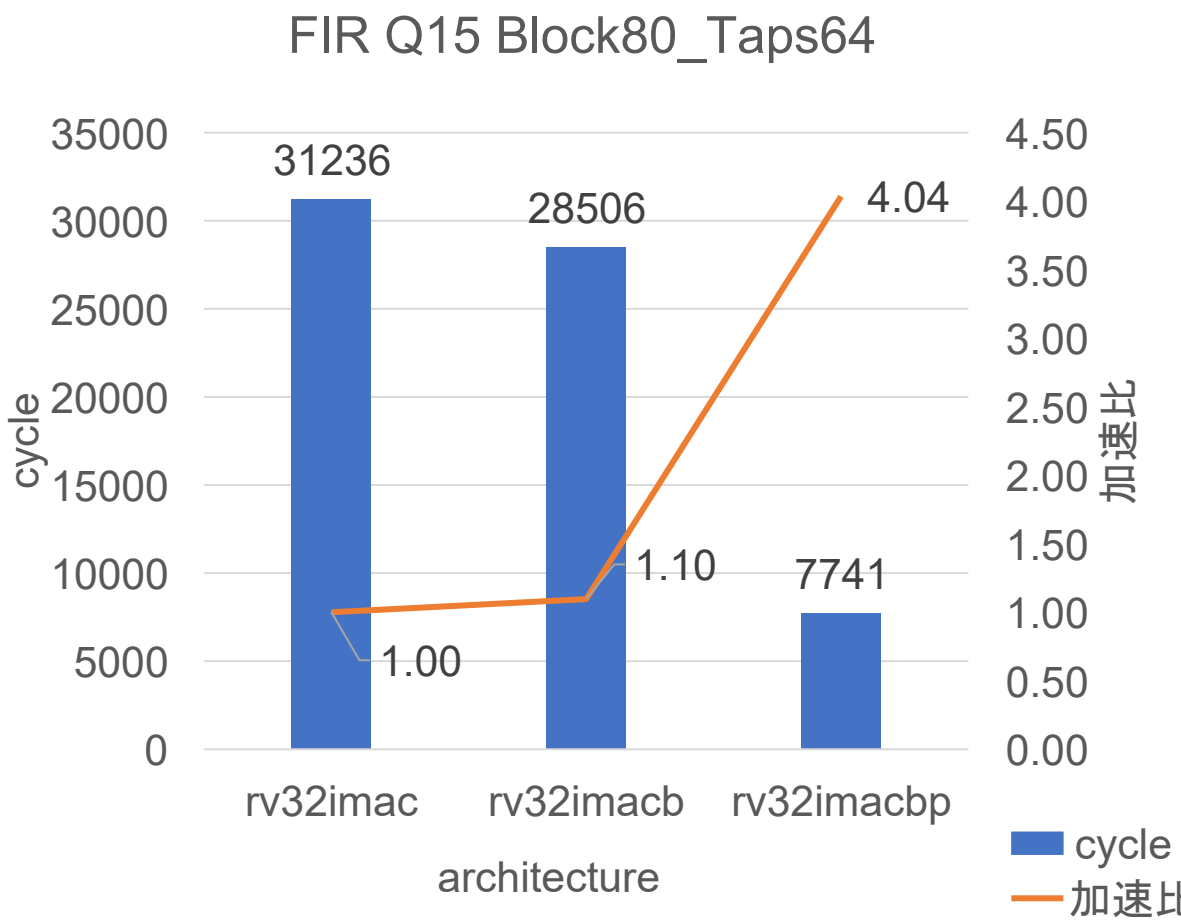
芯来科技 NUCLEI

- FIR滤波器在音频算法中非常常见，输出$y[n]$是输入信号$x[n]$和系数$h[n]$的**线性卷积**：

$$y[n] = \sum_{k=0}^{N-1} h[k] \cdot x[n-k]$$

- Taps = 64, Block Size = 80, **提速4倍！**

Architecture	Cycle	加速比
rv32imac	31236	1.00
rv32imacb	28506	1.10
rv32imacbp	7741	**4.04**

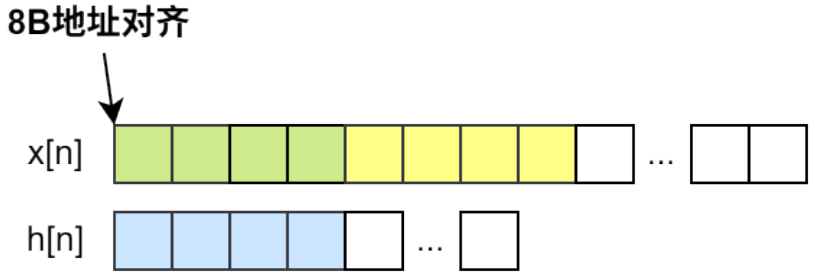


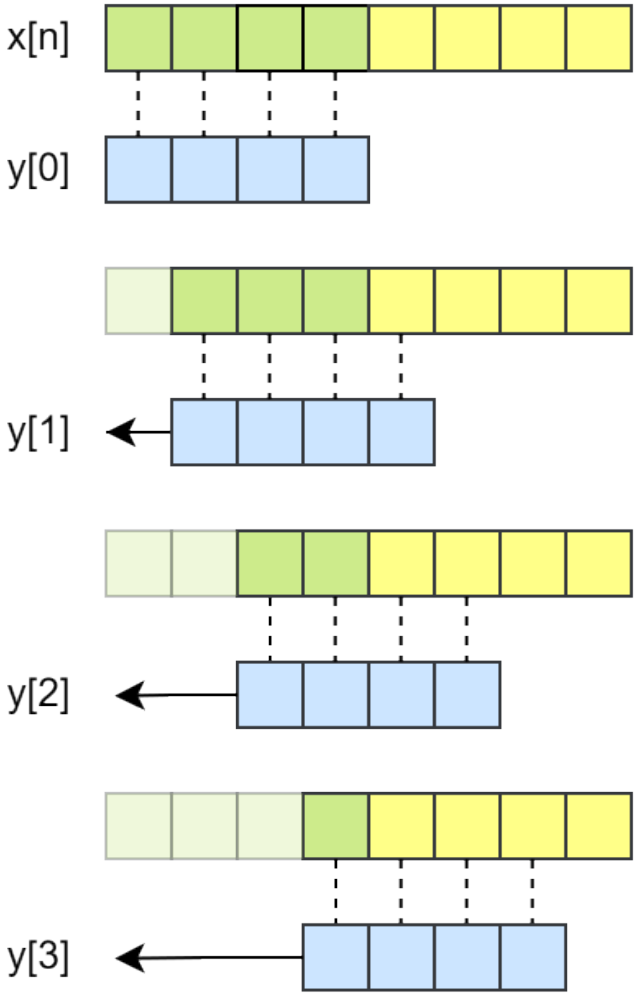
FIR滤波的SIMD实现（2）

芯来科技 NUCLEI

- 软硬件结合的进一步优化

Packed SIMD 扩展	**SMALDA**	同时计算 **2** 对 int16 数据乘累加
Nuclei N3 DSP 扩展	**DSMALDA**	同时计算 **4** 对 int16 数据乘累加
Zilsd扩展	LD/SD	令**RV32**支持直接64bit访存





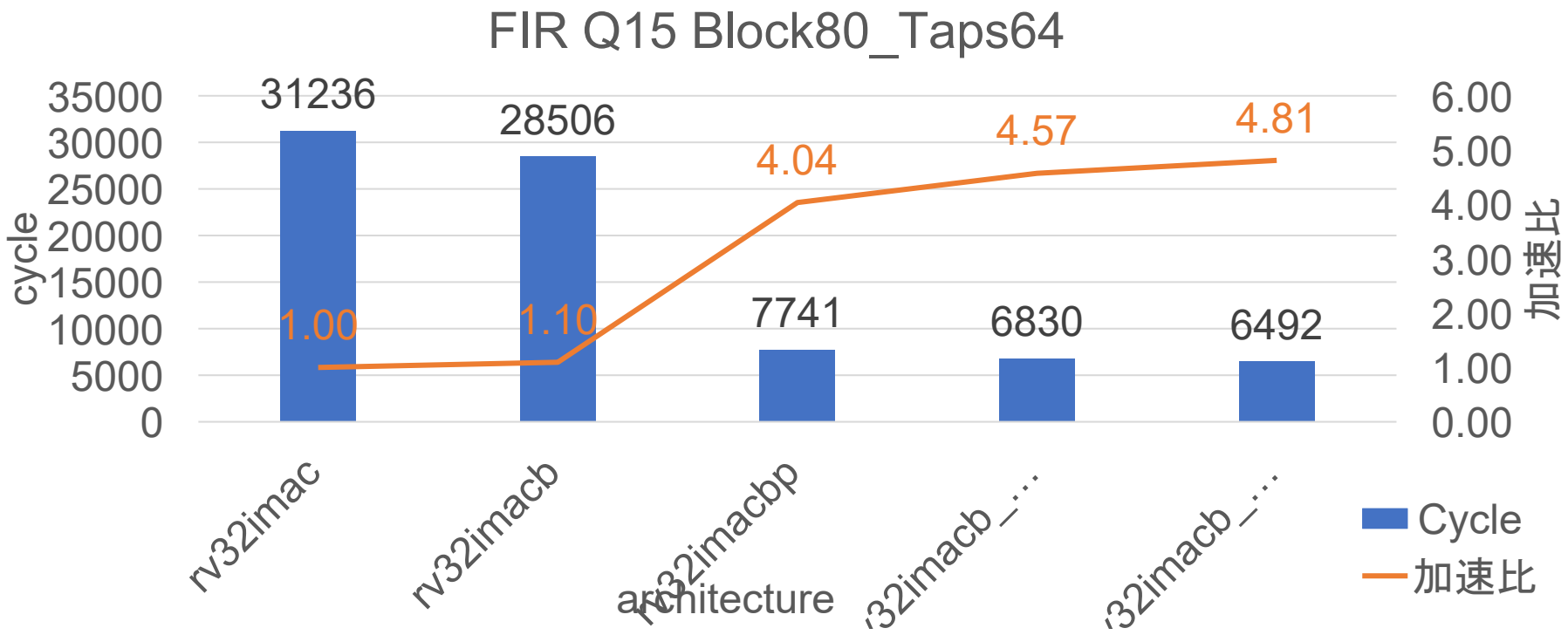
- 利用 B 扩展、zilsd扩展和 Nuclei N3 DSP 扩展优化FIR
  - 利用 **DSMALDA** 同时计算 **4 对** int16_t 数据的乘累加
  - 同时计算 **4 个** 输出结果，**数据复用**
  - 利用 **zilsd 扩展**数据采用 **LD** 指令连续加载两个数据
  - **避免非对齐访问**，利用 PACK 指令腾挪寄存器中的数据

FIR滤波的SIMD实现（2）

芯来科技 NUCLEI

- 利用 B 扩展、zilsd扩展和 Nuclei N3 DSP 扩展优化FIR
  - 利用 **DSMALDA** 同时计算 **4 对** int16_t 数据的乘累加
  - 利用 **zilsd 扩展**数据采用 **LD** 指令连续加载两个数据
  - **避免非对齐访问**，利用 PACK 指令腾挪寄存器中的数据
- Taps = 64, Block Size = 80, **提速4.8倍!**

Architecture	Cycle	加速比
rv32imac	31236	1.00
rv32imacb_xxldspn3x	6830	4.57
rv32imacb_xxldspn3x_zilsd	6492	4.81



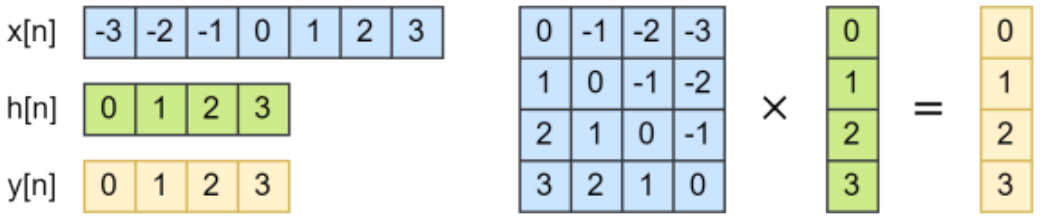
FIR滤波的RVV优化

芯来科技 NUCLEI

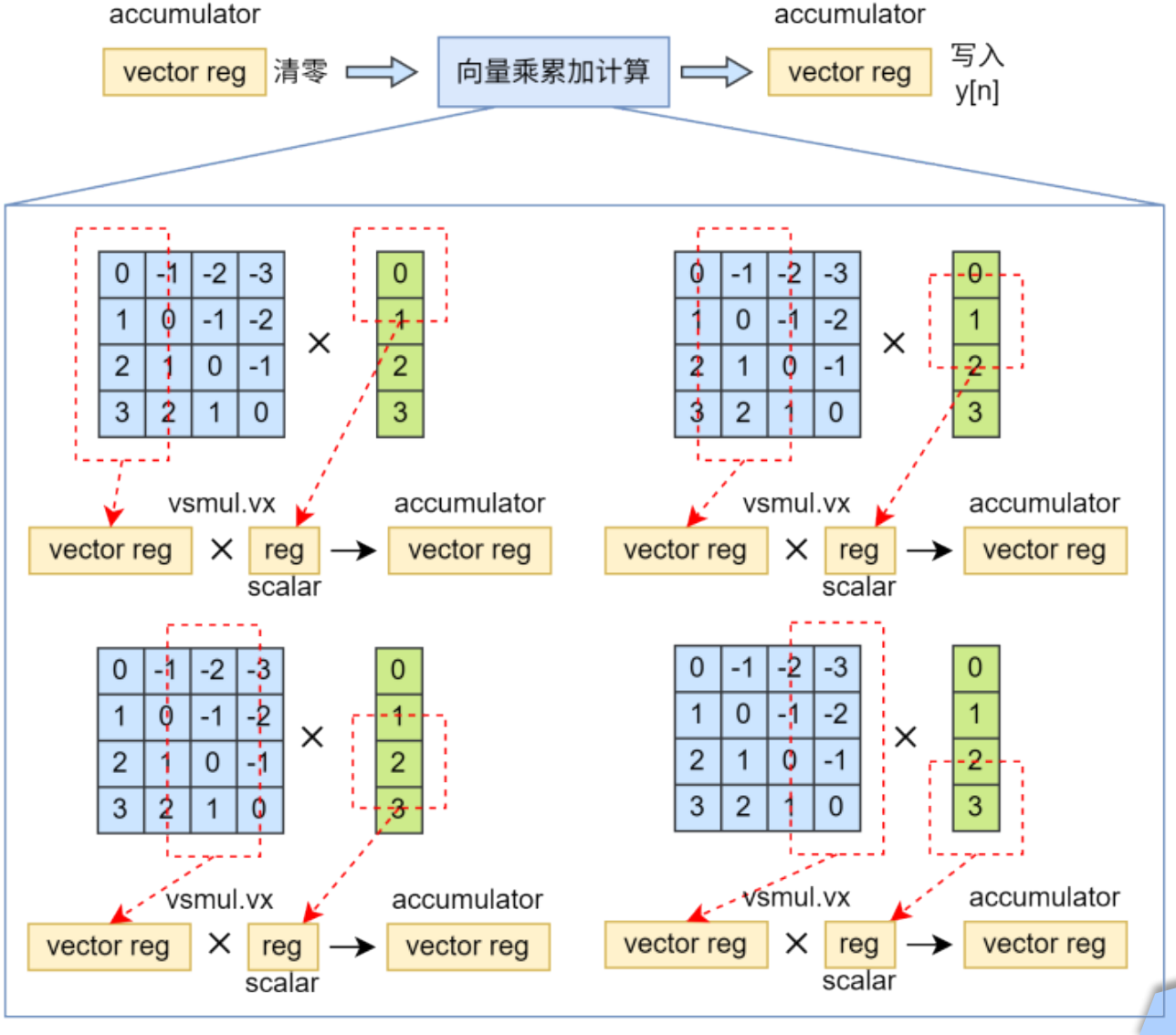
- RISC-V Vector 扩展可以实现更大规模的SIMD数据并行

$$y[n] = \sum_{k=0}^{N-1} h[k] \cdot x[n-k]$$

- 以时间序列x[n] 为例，FIR可**等效为GEMV**



- Vector扩展支持 **vwmacc** 计算乘累加
- 向量寄存器中的数据**呈滑动变化**，利用 **vslide** 指令避免多次访存



FIR滤波的RVV优化

芯来科技 NUCLEI

- 利用V扩展优化FIR计算，在VLEN=128和VLEN=256的条件下分别可以加速 **9倍** 和 **23倍** 以上!
- Block Size = 512, 数据为Q15格式

Taps	Scalar (cycle)	Vlen128 (cycle)	Vlen256 (cycle)	vlen128 加速比	vlen256 加速比
8	21147	2134	913	9.91	23.16
16	37448	3922	1606	9.55	23.32
32	70228	7519	3036	9.34	23.13
64	139855	14671	5825	9.53	24.01
96	201298	21839	8641	9.22	23.30
128	266816	29007	11457	9.20	23.29

普遍认同的FFT计算方案

芯来科技 NUCLEI

- FFT变换在EVS音频编解码，乃至其它音频算法中经常用到。FFT的算法优化已经相对比较成熟。
- 实数FFT可以用1/2点的**复数FFT**计算，计算$x[n]$的傅里叶变换$X[r]$

$$X[r] = F[r] + w_N^r G[r]$$ ，其中$G[r]$, $F[r]$分别是$x[n]$的**奇偶序列**的傅里叶变换结果

- FFT的逆变换可以**沿用正变化**的计算过程，只需要额外对输入数据和输出数据取复共轭。

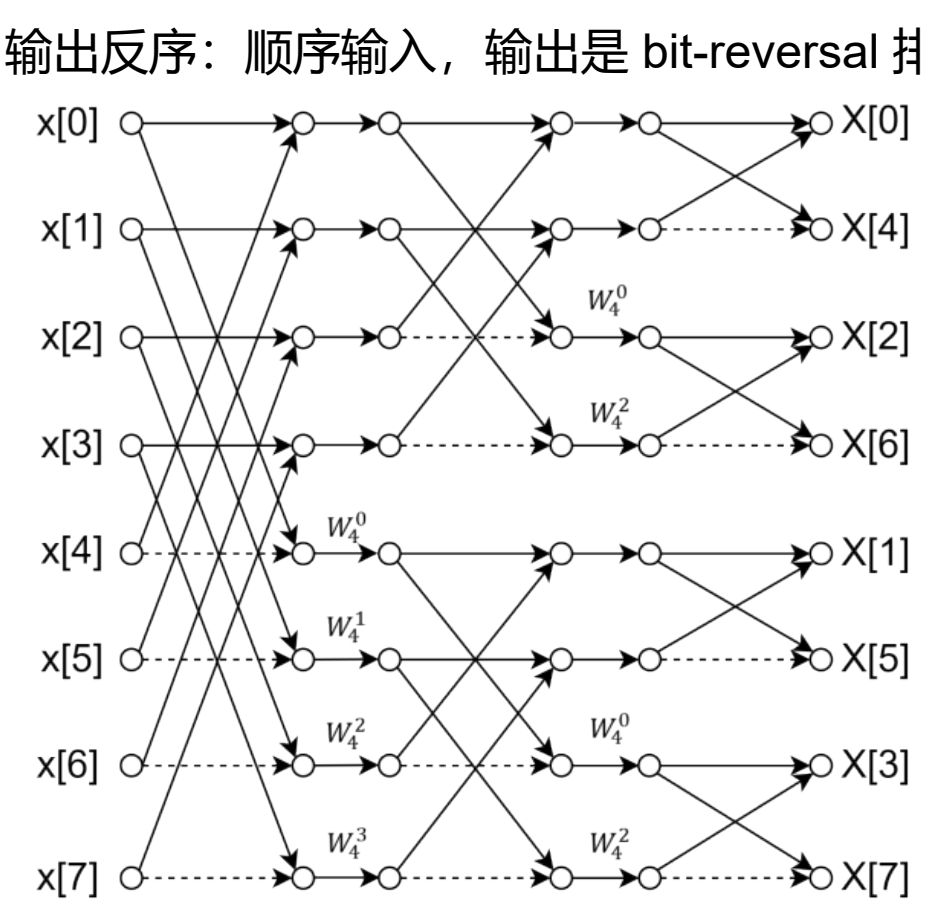
$$iDFT(X[r]) = \sum_{k=0}^{N-1} X[r] w_N^{-kn} = \overline{\sum_{k=0}^{N-1} \overline{X[r]} w_N^{kn}} = \overline{DFT(\overline{X[r]})}$$

- **复数FFT的正变换**性能足以表征处理器计算不同类型FFT的能力

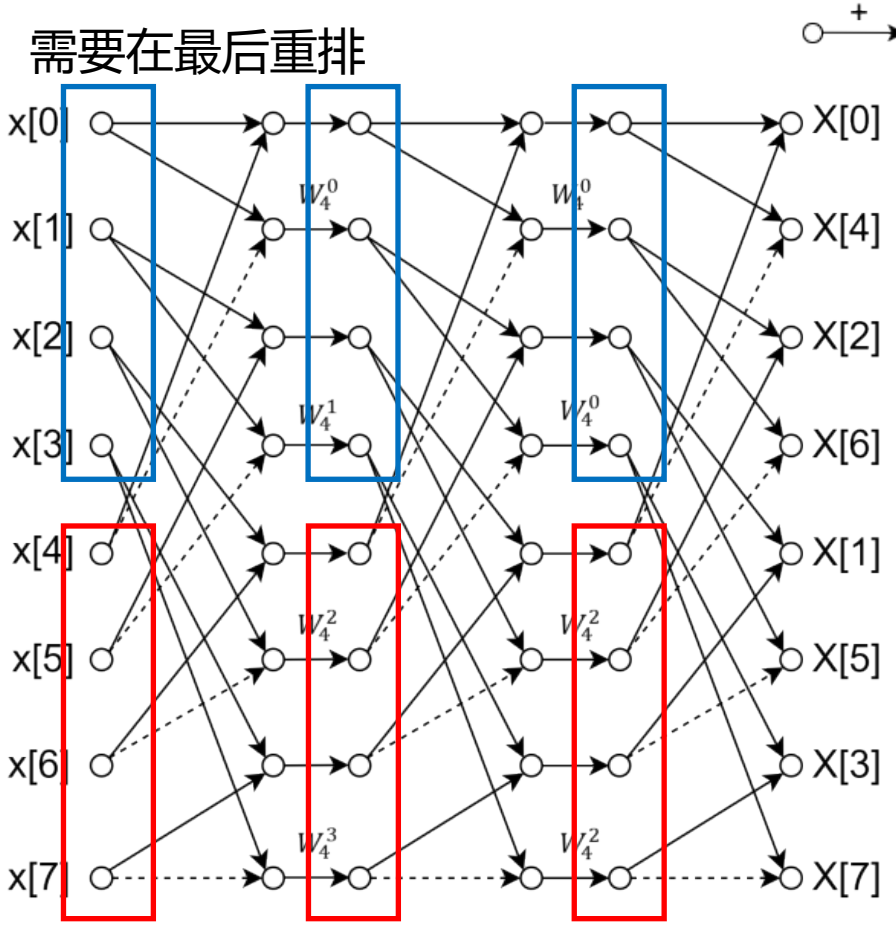
适用于RVV优化的FFT实现方案

芯来科技 NUCLEI

- **Pease FFT** [1]是一种 用于实现快速傅里叶变换（FFT） 的算法结构，由 M. C. Pease 在 1968 年提出。
  - 适合并行处理：**每层运算都能以N/2的向量长度进行SIMD计算**（N是FFT的点数）
  - 输出反序：顺序输入，输出是 bit-reversal 排列，需要在最后重排



8点DIF FFT计算图



8点Pease FFT计算图

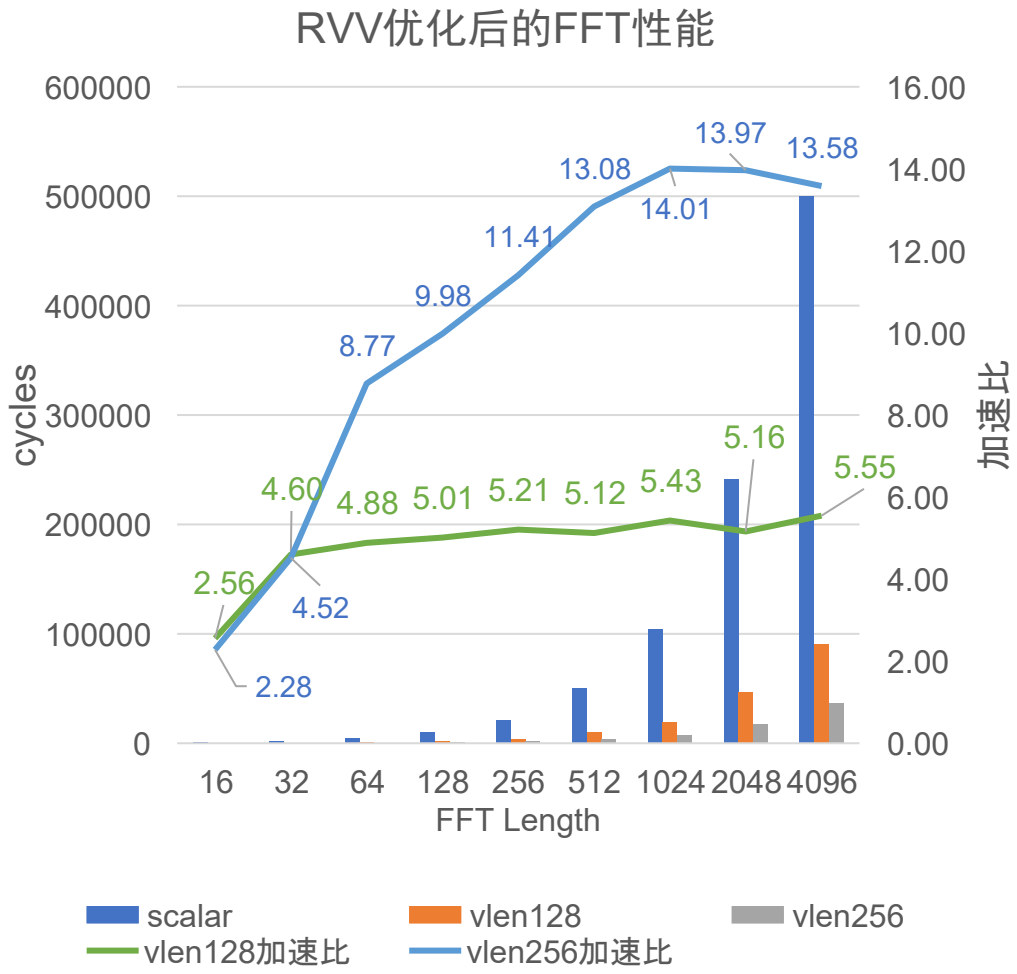
[1] An Adaptation of the Fast Fourier Transform for Parallel Processing | Journal of the ACM: https://dl.acm.org/doi/abs/10.1145/321450.321457

适用于RVV优化的FFT实现方案

芯来科技 NUCLEI

- VLEN128/256情况下利用RVV优化FFT的加速效果
- 统计**CFFT正变换**需要的Cycle数

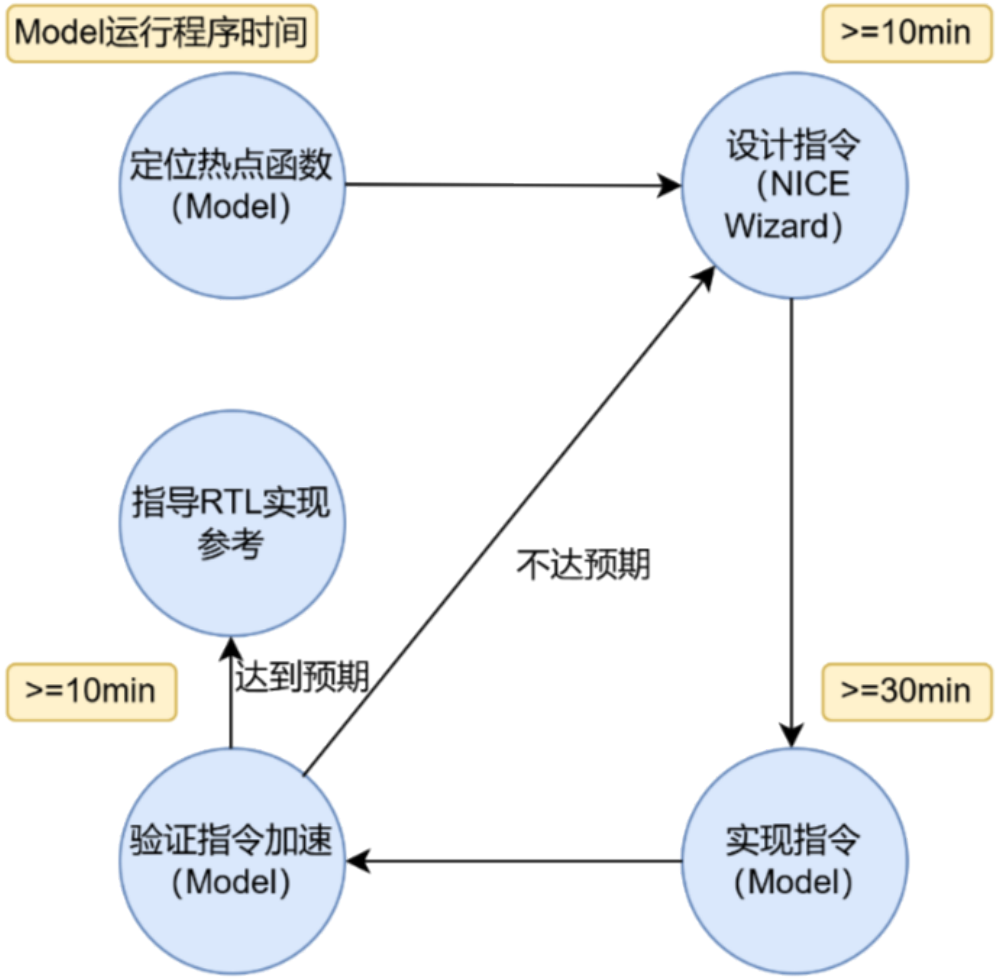
FFT Length	Scalar (cycle)	Vlen128 (cycle)	Vlen256 (cycle)	vlen128 加速比	vlen256 加速比
16	987	386	433	2.56	2.28
32	2006	436	444	4.60	4.52
64	4279	876	488	4.88	8.77
128	10122	2019	1014	5.01	9.98
256	21049	4042	1845	5.21	11.41
512	50156	9788	3834	5.12	13.08
1024	104352	19222	7450	5.43	14.01
2048	241252	46734	17273	5.16	13.97
4096	499830	90100	36803	5.55	13.58



Nuclei Model自定义指令加速算法优化

芯来科技 NUCLEI

- 对于极致性能需求的场景，可以考虑采用**自定义指令**优化
- **Nuclei Model** 结合 Nuclei Studio 的 **NICE Wizard** 功能，**快速评估算法优化后的性能**



	传统流程	Nuclei Model流程
运行环境	RTL & FPGA & IDE	IDE
人力消耗	IC设计/验证工程师 & FPGA工程师 & 软件工程师	**软件工程师 Only**
总耗时	>=2 weeks	**>= 1hour**
容错率	低	**高**

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！