PerfXLab

Optimization of the FFT algorithm on RISC-V CPUs

Xianyi Zhang

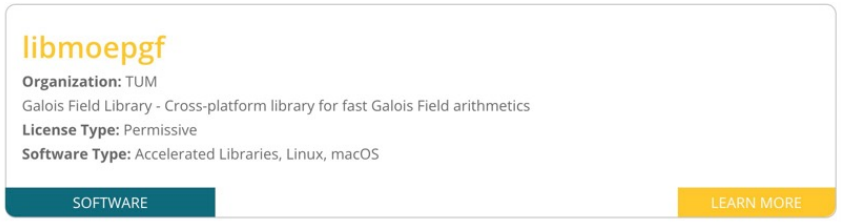
PerfXLab Technologies

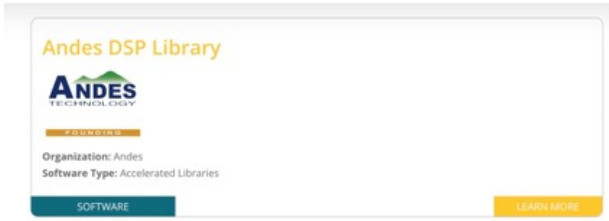
xianyi@perfxlab.com

2023.5

Background – RISC-V & HPC

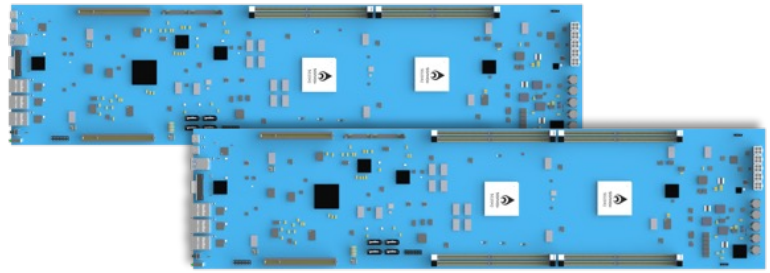
- RISC-V Vector Extension
  - RVV 0.7.1 chips (e.g. T-head C906, C910V)
  - RVV 1.0 (e.g. AX45MPV, C908)
  - RVV Intrinsic ??
- Multicores
  - 64 cores : SG2042
- Accelerated Libraries from RISC-V website
  - A few compute libraries support RISC-V and RVV





64 cores

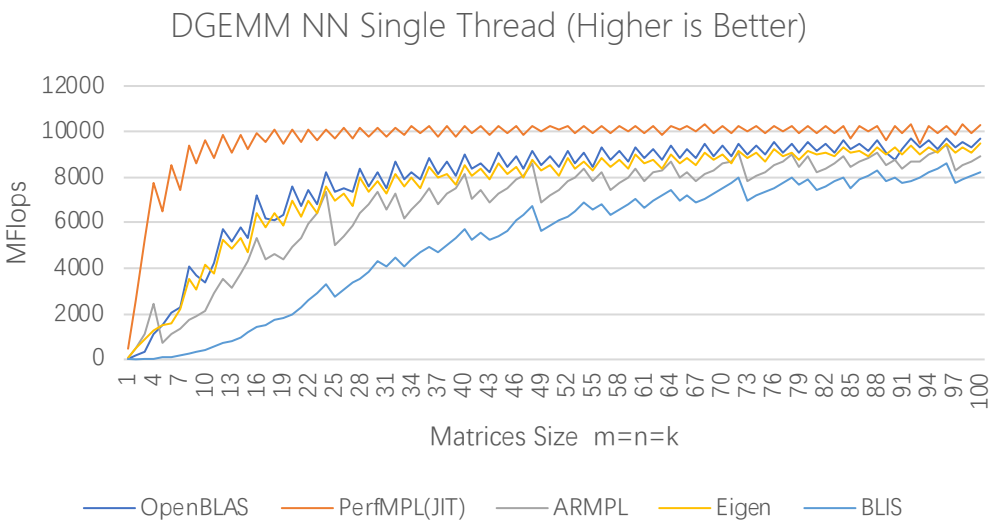
SG2042
- RISC-V@2.0Ghz
- Vector 0.7.1



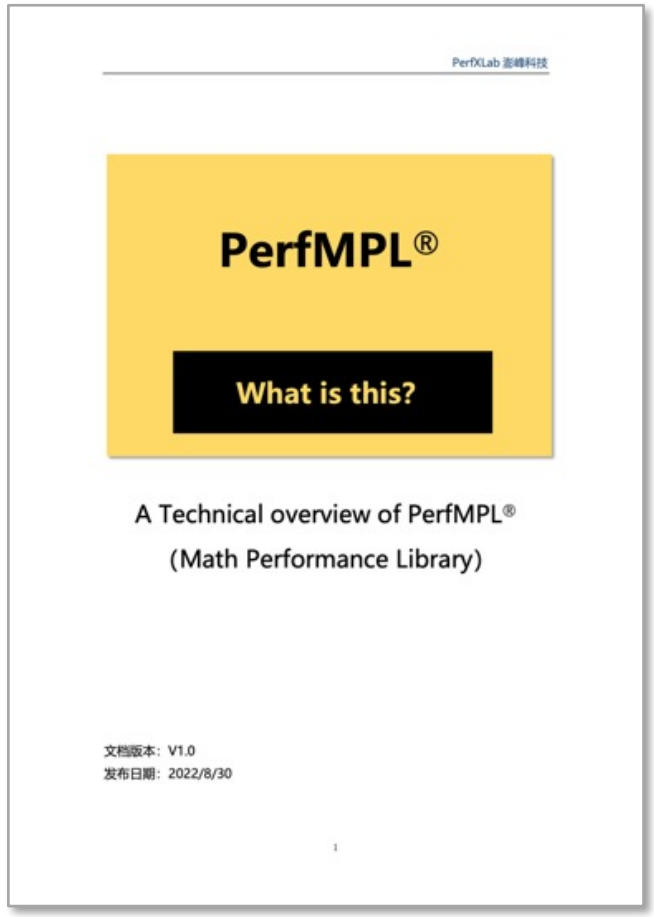
HS-1 RISC-V Compute Server
- 2 x Mainboards (2 x SG2042 , PCIe 4.0)
- Created by PerfXLab & WIAT
- Bringup on June, 2023

PerfMPL compute library

- Supports X86, ARM, GPU (OpenCL)
- WIP: Porting RISC-V
- Includes
  - OpenBLAS
  - OpenFFT
  - OpenVML
  - …



DGEMM on Kunpeng 920 ARMV8 CPU



Porting FFT

- Stockham Butterfly Algorithm
  - SIMD friendly
  - Mixed Radix

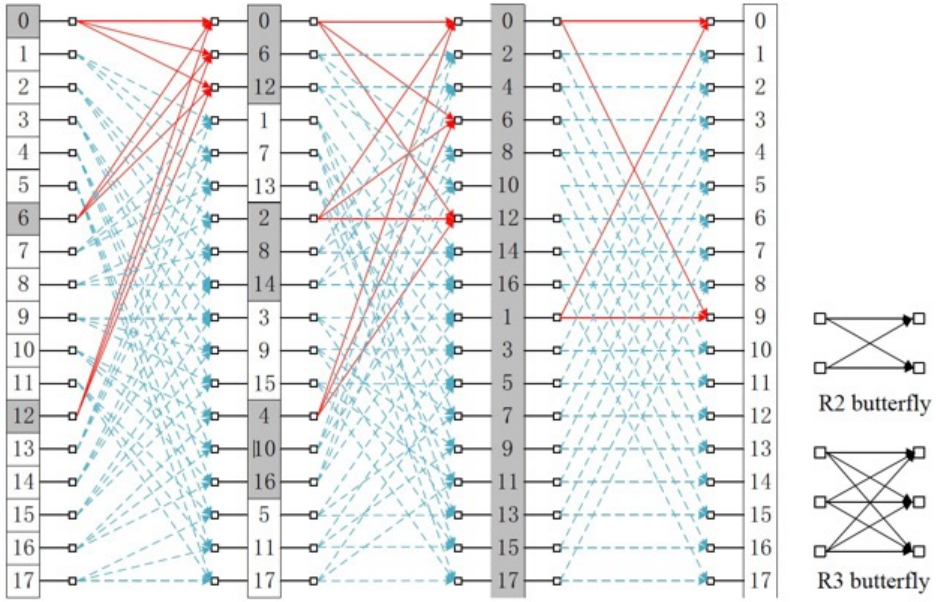


Fig. 1. The Stockham butterfly network diagram for the FFT of size eighteen. The red line represents a butterfly of the current stage.

Optimizing FFT Kernel

- Small DFT
- SIMD vectorization

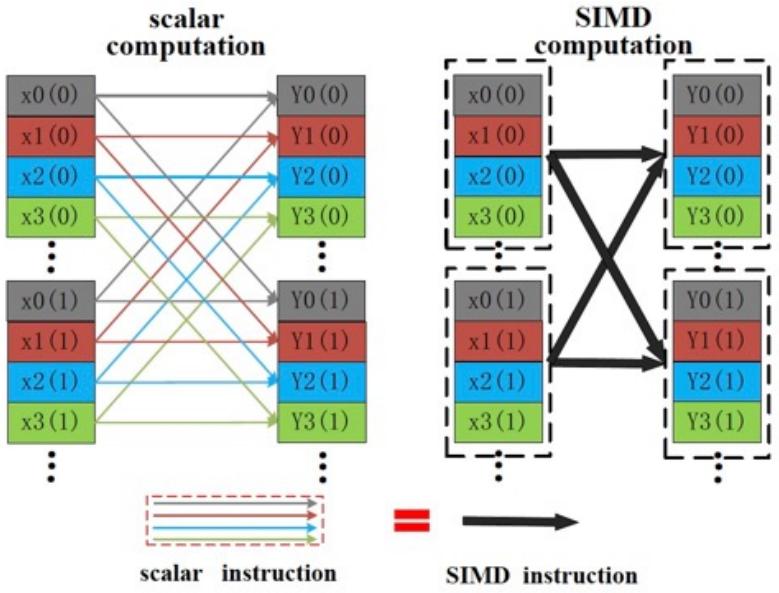


Fig. 2. R2 butterfly vectorization calculation with parallelism 4.

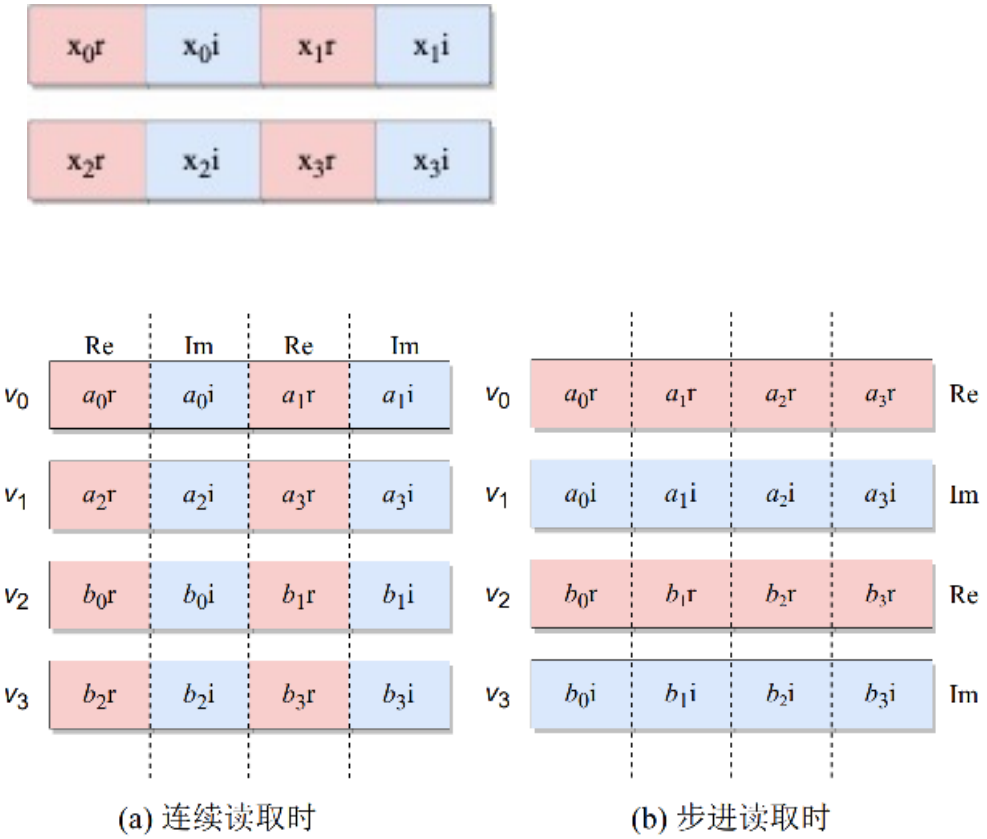
- Loop unrolling / Register Group for small radix
- E.g. R2
  - Unrolling 8x
  - Or, using RVV Register Group LMUL, instead

Optimizing FFT Kernel

- Data access and RVV instruction selection
  - Split real and imaginary number in vector

```
vlseg<nf>e.v vd, (rs1)
# offset = sew/8
# for (k = 0; k < nf; k++)
#   V_(d+k)[i] = mem[rs1 + k*offset + i*nf*offset]
```

```
v0[0]=mem[x5],       v0[1]=mem[x5+4],
v0[2]=mem[x5+8],     v0[3]=mem[x5+12]
v1[0]=mem[x5+16],    v1[1]=mem[x5+16+4],
v1[2]=mem[x5+16+8],  v1[3]=mem[x5+16+12]
```



- Use vfmacc.vv / vfnmsac.vv
- Register allocation

Performance Evaluation

- C910V IP on FPGA
- PerfMPL-FFT compared to C implemation
  - Single complex: 3.67X
  - Double complex: 3.60X

CPU	C910MP	**Register Number**	32
Architecture	RISC-V	**Register Length**	128
Vector Extension	Version 0.7.1	**GCC Version**	5.5.0
Frequency	30MHz	**FFTW**	3.3.10

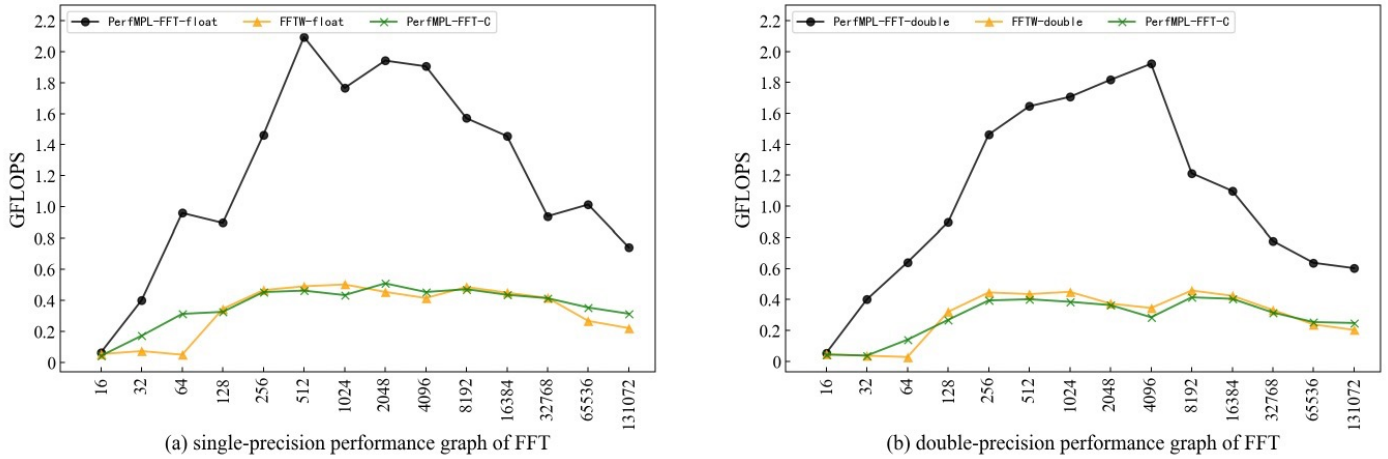


Fig. 3. (a) and (b) are the DFT processing performance of PerfMPL-FFT SIMD code, PerfMPL-FFT C code, and FFTW algorithm library on C910MP CPU for single-precision and double-precision data, respectively.

Conclusion and Future works

PerfXLab

- Porting PerfMPL to RISC-V
- PerfMPL-FFT compared to C implemation
  - Single complex: 3.67X
  - Double complex: 3.60X

- Future works
  - Optimized on HS-1 RISC-V compute server
  - Other numerical libraries, frameworks

PerfXLab

Thank you

Email: Xianyi@perfxlab.com

Wechat