Title: An FFT library optimized for VLBI software correlators

Authors: Takeuchi Hiroshi*, Chikada Yoshihiro**, Koyama Yasuhiro*
e-mail: ht@nict.go.jp
Affiliation: *:NICT/Japan **:NAOJ/Japan

Abstract:
An FFT software library, which is optimized for FX correlators, was developed. In the library, same-tick data from multiple different streams (different baseband channels or different stations) are stored adjacently in the memory space and be processed simultaneously. It guarantees an efficient sequential access to the memory in butterfly computations of the FFT. This library was written in x86 assembly language, and SSE3 instruction set is used to accelerate its performance. To evaluate the performance of the library, we compared its processing speed against that of FFTW3.0 which is known as one of the fastest FFT libraries for many processors. As a result, it runs about 10-30% faster than the FFTW. If this library is applied to the multi-channel VLBI data stream, much more speed-up can be expected.

1. Introduction

Conventionally, digital-backend instruments in radio interferometer such as correlators or digital spectrometers are implemented with custom-built hardware to process high-speed video band signals in real-time. In the mean time, performance of commodity PCs has increased so that we could use them in the digital-backend system of VLBI. In order to boost the performance of PC-based instruments, we developed an FFT library optimized for processing VLBI data streams.

2. Algorithm

While sequential access speed to the memory is much higher than random access speed in recent CPUs, non-sequential accesses are inevitable in FFT algorithm especially in the bit-reverse permutations in the last stage of FFT. To solve this problem, we developed a new FFT library in which multiple different streams are transformed simultaneously. Because same time-index data from different streams are stored adjacently in the buffer area of the library, memory accesses in the most inner loops are guaranteed to be sequential.

Here we suppose n-sets of N-point FFT input streams,

A[i,j] (i=1,…,n, j=1,…,N).

In the normal way, these data sets are transformed in a sequential order of **i** as follows:

```
For i=1 to n do
…
        B=A[i,*]
        For j=1 to N do
                …
                For k=1 to K do
                        j[k]=g(j,k)
                        J[k]=G(j,k)
                enddo
                (B[J[1]],…, B[J[K]])=f(B[j[1]],…, B[j[K]])
                …
        enddo
        …
enddo
```

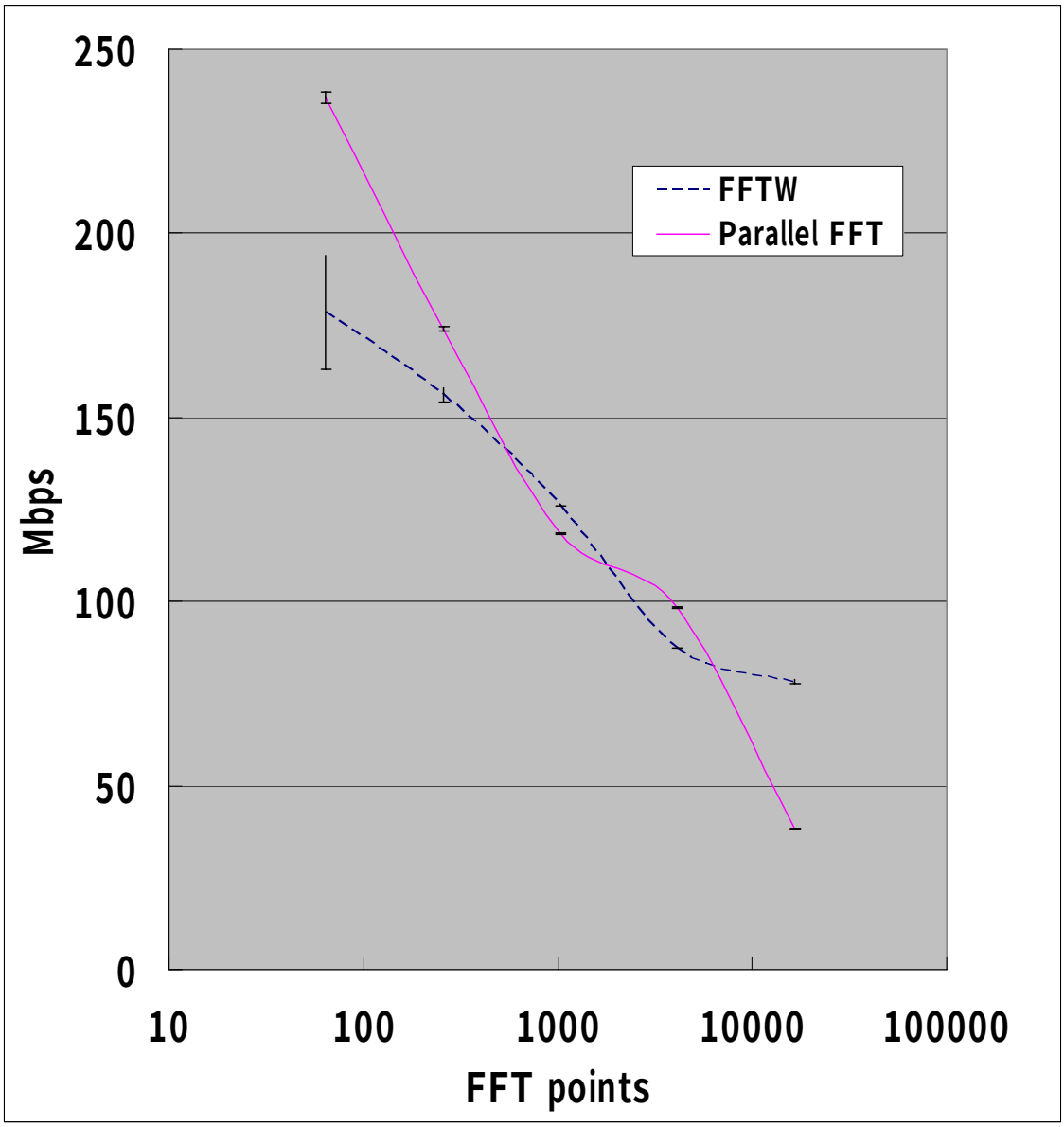
Note that memory accesses in the most inner loop of the above pseudo-code are not sequential. Though there are some ways to realize a nearly-sequential access by using out-of-place algorithms, complete sequential access is impossible due to the FFT data-flow topology.

Here, we propose a parallel FFT algorithm, in which same time-index data from different streams are stored adjacently. In this scheme, **A[i,j]** are transposed and **i**-loop and **j**-loop are interchanged as follows:

```
For j=1 to N do
        …
        For k=1 to K do
                j[k]=g(j,k)
                J[k]=G(j,k)
        enddo
        For i=1 to n do
                …
                (A[i,J[1]],…, A[i,J[K]])=f(A[i,j[1]],…, A[i,j[K]])
                …
        enddo
        …
enddo
```

Note that there are **n**-contiguous sequential memory accesses in the most inner loop of above code. Moreover, the number of times FFT coefficients are loaded for the butterfly operations becomes **1/n.** On the other hand, required buffer space for this algorithm becomes **n**-times. If the required buffer size exceeds the L2-cache size of the system, the performance will be degraded significantly.

3. Benchmark



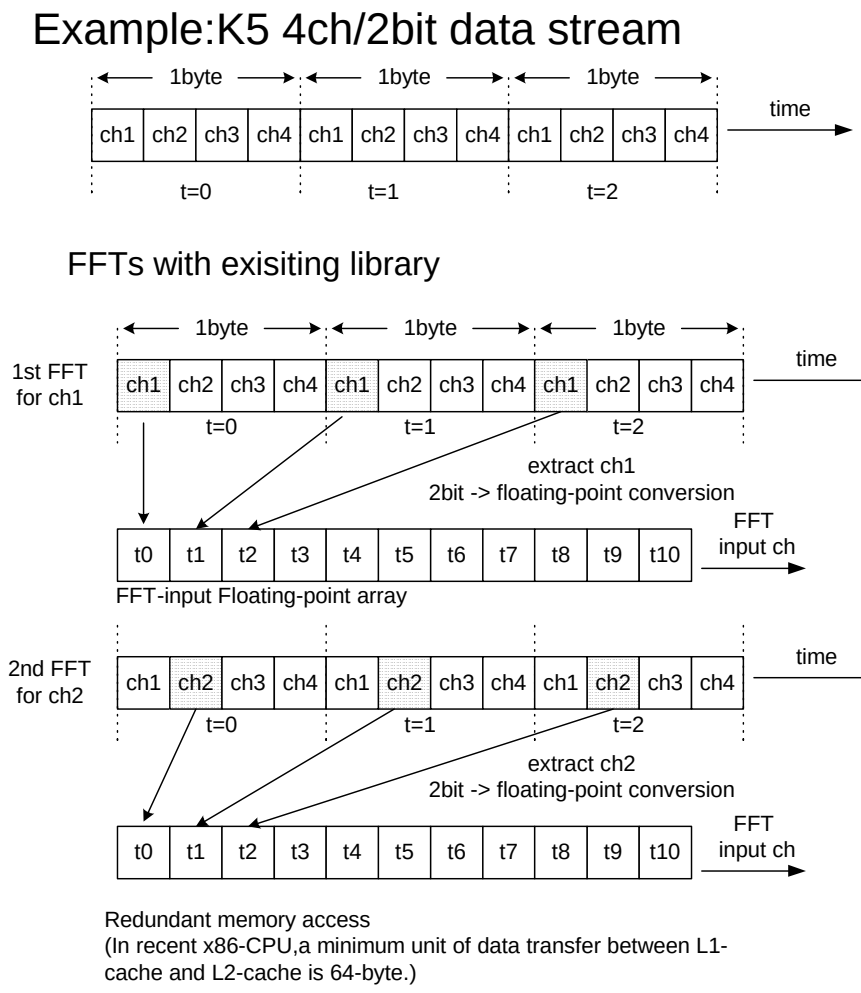
- Environment: Intel Xeon 3.40 GHz, L1 cashe:8KB,L2 cache:1024KB,Main memory 4GB,Using only 1CPU (IA32)
- Benchmark test data length: 10G samples * 5 times(error bar in the graph is based on the standard deviations of five results)
- FFTW3.0:complex input, single-precision, compiled with GCC -O3, using SSE functions. FFTW_EXHAUSTIVE option was used.
- Parallel FFT library: 4-streams parallel operation, complex input, single-precision, written in assembly language (nasm), using SSE and SSE3 functions, radix-4, Cooley-Tukey, out-of-place(dual buffer).

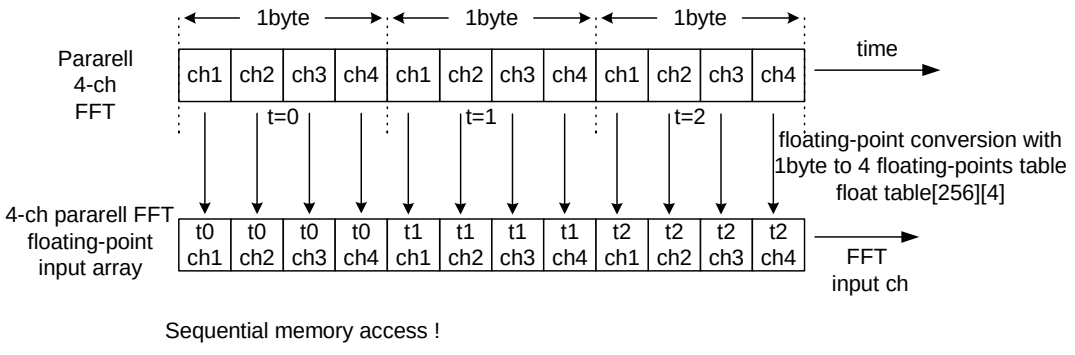
4. Why original FFT library? (Other advantages for software correlators)

- Fringe rotation can be included in butterfly operations in the last stage of FFT operation. Trigonometric functions table for fringe rotation can be shared with the FFT coefficient table.
- Conversion from input bit streams to floating-point values can be included in the first stage of the FFT library.
- If each input stream of parallel FFT is allocated to stream from different station, same frequency channels from multiple different stations are adjoining in memory

space after the FFT. As a result, 'X' part's operation in the FX correlator can be performed effectively without any redundant memory accesses.

5. Application(K5/2bit data stream)





6. Library Download

You can obtain the parallel FFT library from

http://www2.nict.go.jp/ka/radioastro/people/takeuchi/fft/index.html

If you have any questions, please feel free to contact the author (ht@nict.go.jp).

7. References

M. Frigo and S. G. Johnson, FFTW: An adaptive software architecture for the FFT, *Proc. ICASSP 1998* **3**, p. 1381