전파천문 상관처리를 위한 최적 코딩 방법에 관한 연구
A Study on Optimum Coding Method for Correlation Processing of Radio Astronomy

신재식*, 오세진*, 염재환*, 노덕규*, 정동규*, 오충식*, 황주연*, 소요환**
Jae-Sik Shin*, Se-Jin Oh*, Jae-Hwan Yeom*, Duk-Gyoo Roh*, Dong-Kyu Chung*, Chung-Sik Oh*, Ju-Yeon Hwang*, Yo-Hwan So**

요 약

본 논문에서는 한일공동VLBI상관기를 위한 소프트웨어 상관기의 성능개선을 위해 공개된 라이브러리를 이용한 최적 코딩 방법에 대해 제안한다. VLBI 관측데이터를 상관처리하는 시스템은 관측국 수에 따라 계산량이 기하급수적으로 증기하기 때문에 일반적으로 ASIC 또는 FPGA를 이용하여 하드웨어로 구축한다. 하지만 최근에는 컴퓨터 성능의 발전에 따라 소프트웨어를 이용하여 클러스터와 같은 대용량 서버에서 구축하고 있다. 하드웨어로 구축한 VLBI 상관처리 시스템은 대부분 관측시간대비 준실시간 또는 실시간으로 데이터처리를 수행하기 때문에, 소프트웨어 상관처리 시스템도 이와 같은 성능을 갖기 위해서는 코딩 작업에서 데이터 처리를 최적으로 수행할 수 있어야 한다. 따라서 본 논문에서는 상관처리 시스템에서 가장 중요한 과정인 FFT 처리과정에서 공개된 fftw 라이브러리를 활용하여 최적코딩을 수행할 수 있도록 fftw 라이브러리를 이용한 일반적인 방법, SSE를 활용한 방법, 공유메모리를 사용한 방법, OpenMP를 활용한 방법, 그리고 제시한 것들을 병합한 방법을 적용한 경우에 대해 상관처리 실험을 수행하였다. 상관처리실험을 통하여 본 논문에서 제안한 fftw와 공유메모리, 그리고 OpenMP를 적용한 최적 코딩 방법이 상관처리 시간을 단축할 수 있어서 소프트웨어 상관기의 성능개선에 유효함을 확인하였다.

ABSTRACT

In this paper, the optimum coding method is proposed by using open library in order to improve the performance of a software correlator developed for Korea-Japan Joint VLBI Correlator(KJJVC). The correlation system for VLBI observing system is generally implemented with hardware using ASIC or FPGA because the computational quantity is increased geometrically according to the participated observatory number. However, the software correlation system is recently constructed at a massive server such as a cluster using software according to the development of computing power. Since VLBI correlator implemented with hardware is able to conduct data processing with real-time or quasi real-time compared with mostly observational time, software correlation has to perform optimal data processing in coding work so as to have the same performance as that of the hardware. Therefore, in this paper, the experimental comparison was conducted by open-source based fftw library released in FFT processing stage, which is the most important part of the correlator system for performing optimum coding work in software development phase, such as general method using fftw library or methods using SSE(Streaming SIMD Extensions), shared memory, or OpenMP, and method using merged techniques listed above. Through the experimental results, the proposed optimum coding method for improving the performance of developed software correlator using fftw library, shared memory and OpenMP is effectively confirmed by reducing correlation time compared with conventional method.

Keywords : Optimum coding method, fftw, SSE, Shared memory, OpenMP, Software correlator

I. 서 론

최근 컴퓨터의 성능이 발전함에 따라 대용량 데이터처리를 위해 클러스터 형태를 활용한 컴퓨터 시스템이 자료처리에 활용되고 있다. 전파천문학 분야 중 전파간섭계인 VLBI[1] 관측데이터는 계산 전용 칩을 제작하는 ASIC[2] 또는 프로그래밍 언어 형태의 FPGA[3]를 이용하여 데이터처리를

* 한국천문연구원 ** 한남대학교
투고 일자 : 2015.9.10 수정완료일자 : 2015.11.1
게재확정일자 : 2015.11.8

1) Very Long Baseline Interferometry
2) Application Specific IC

위한 상관시스템과 같은 전용 하드웨어를 구축하는 것이 일반적이었으나, 최근에는 하드웨어가 발전함에 따라 대용량 컴퓨터 시스템을 활용하여 소프트웨어로 작성한 소프트웨어 상관기를 구축하여 활용하고 있다[1]. 전용 칩을 이용한 하드웨어 상관기 시스템을 구축하는 경우 VLBI 관측데이터에 대한 상관처리만을 위해 전용으로 제작하기 때문에 고속으로 데이터처리가 가능하지만, 전용 칩의 설계와 구축에 비용이 많이 들며, 관측시스템이 변경될 경우 상관처리 시스템을 변경하는 것이 어려운 점이 있다. 최근에는 전용 칩을 제작하는 것의 보완으로 프로그래밍을 할 수 있는 FPGA 칩을 활용하여 하드웨어 상관시스템을 구축하고 있으며, 전용 칩을 이용한 경우보다 구축비용 측면과 시스템의 변경에 대응할 수 있는 유연성이 높아졌다고 할 수 있다.

VLBI 관측에서 소프트웨어를 이용한 상관처리 시스템의 경우 VLBI 관측국 수가 적거나 데이터의 기록속도가 낮은 경우에 활용되고 있으며, 소프트웨어는 하드웨어에 비하여 다양한 구성이 가능하기 때문에 유연성이 매우 높은 장점이 있다[2]. 또한 어떤 경우에는 실시간 데이터처리가 필요하지 않은 경우에도 소프트웨어를 활용한 상관처리 시스템 구축에 활용되고 있다. 하지만 VLBI 관측과 같이 관측국 수가 많아지면 처리해야하는 기선의 수는 {n(n-1)/2, n 관측국수}로 늘어나기 때문에 소프트웨어 상관기를 이용하여 많은 관측국 수와 관측기록속도가 높은 경우 실시간 데이터처리를 위한 경우에는 아직 컴퓨터 시스템의 성능이 따라 오지 못하고 있는 실정이다[3]. 하지만, 현재 무어(Moore)의 법칙[4]에 따라서 컴퓨터 시스템의 발전 추이에 따라서 빠른 시일에 이러한 문제점을 소프트웨어를 이용하여 VLBI 관측에 대한 상관처리가 하드웨어 상관기를 대신할 수 있을 것으로 기대한다.

앞에서 기술한 것과 같이 하드웨어 상관기의 경우 대량의 데이터를 초고속으로 계산할 수 있는 장점이 있지만 주어진 규격에 한정하여 시스템을 설계 제작하기 때문에 규격에 대한 유연성이 떨어진다. 반면 소프트웨어 상관기는 운영하는 서버의 성능에 따라 규격 등을 변경하고 시험할 수 있으며 하드웨어 상관기에 비해 lag의 수를 관측모드에 따라 적절히 조절할 수 있으며, 일정한 시간간격 범위 내에서 Gbit 데이터의 기선수를 임의로 조절하여 최적으로 상관처리를 수행할 수 있는 장점이 있다.

소프트웨어 상관시스템을 구축할 때 여러 가지 장점을 이용하여 본 논문에서는 기존에 구축한 소프트웨어 상관기의 성능을 개선하기 위해 표준 C 언어 기반의 라이브러리인 fftw[5]를 활용한 최적 코딩방법에 대해 제안하고자 한다. 본 연구에서 fftw를 선택한 이유는 상관처리 알고리즘의 대부분이 입력신호에 대한 주파수 분석처리에 많은 시간이 소요되기 때문이다. 소프트웨어 상관기는 기선의 증가에 따라 계산량이 기하급수적으로 증가하기 때문에 컴퓨터의 성능을 최대한 활용해야 동일 예산 대비 기선의 수를 늘이거나 동일 데이터 대비 빠른 출력 결과를 얻을 수 있으므로 최적화된 방법을 찾아서 계산 속도를 높이는 것이 매우 중요하다. 소프트웨어로 구축한 상관시스템에서 최적화된 코딩 방법을 찾기 위해 fftw 라이브러리를 이용한 일반적인 방법, SSE(Streaming SIMD Extensions)를 활용한 방법, 공유메모리(Shared memory)를 사용한 방법, 그리고 OpenMP(Open Multi-Processing)를 활용한 방법으로 기존에 구축한 소프트웨어 상관기에 대해 본 논문에서 제안하는 최적 코딩 방법을 적용한 각각 상관 알고리즘에 대한 시스템의 성능 비교 실험을 수행하였으며, 제안 방법에 따른 실험결과에 대해 고찰하고자 한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 하드웨어 및 소프트웨어 상관기에 대해 간략히 소개하고, Ⅲ장에서는 본 논문에서 제안하는 최적코딩 방법들에 대해 소개하고, Ⅳ장에서는 실험결과에 대해 고찰한 후, 마지막으로 Ⅴ장에서 본 논문의 결론을 맺는다.

Ⅱ. 하드웨어 및 소프트웨어 상관기

한일상관센터에서는 한국천문연구원과 일본국립천문대가 2006년부터 공동으로 개발한 하드웨어 상관기인 한일공동VLBI상관기(KJJVC, Korea-Japan Joint VLBI Correlator, 이하 대전상관기)[4]와 DiFX[1] 소프트웨어 상관기를 2010년부터 운영하고 있다. 대전상관기는 총 16기의 전파망원경 120기선(Baseline)에 대해 기당 8192 Mbps의 속도로 8192 상관출력(분광채널)을 갖는 상관처리를 수행할 수 있다. 대전상관기는 FX 형식(입력신호에 대해 FFT[6]를 먼저 수행하고 상관적분을 수행하는 방법)의 설계구조를 갖는다. 그림 1은 한일상관센터 내에 설치된 대전상관기를 보여준다. DiFX 소프트웨어 상관기는 한국우주전파관측망(Korean VLBI Network, KVN)으로 VLBI 관측한 데이터의 상관처리를 담당하고 있다. 또한 관측의 정확성을 높이기 위해 VLBI 관측 전에 사전관측을 수행한 후 관측데이터를 상관센터로 전송하는 e-VLBI(electronic VLBI)로 상관처리를 수행하여 결과를 확인하는데도 활용하고 있다.

그리고 하드웨어 상관기인 대전상관기를 개발하면서 성능검증을 위해 동일한 설계규격으로 개발한 소프트웨어 상관기도 상관처리 결과를 비교·검토할 때 활용하고 있다[2]. 이 소프트웨어 상관기는 개발초기, 하드웨어 상관기와 설계규격을 동일하게 설정하고 VLBI 관측 데이터에서 1~수초정도의 관측 자료만 비교·검토하는 용도로 개발하였으므로 대용량 관측 자료를 처리할 때 속도 면에서는 많이 미흡하였다.

본 논문은 기 구축한 소프트웨어 상관기[2]의 성능을 개선하여 향후 하드웨어 상관기를 대신할 수 있을 정도로 만들기 위해서는 많은 부분에서 개량할 필요가 있으며, 특히

3) Field Programmable Gate Array
4) http://www.mooreslaw.org/
5) http://www.fftw.org
6) Fast Fourier Transform

상관알고리즘에서 가장 중요한 FFT 처리방법에 대해 공개되어 있고 잘 설계된 fftw 라이브러리를 본 연구의 소프트웨어 상관기에서도 채용하고 있다.

따라서 이 fftw 알고리즘을 기본으로 하여 하나의 기능으로서 최적인 방법이 어떤 것이 있는지 조사하고 각각의 방법들을 병합한 최적 코딩 방법을 본 연구에서는 제안하고자 한다. Ⅲ장에서 최적 코딩 방법에 대해 살펴보고 Ⅳ장에서 각 방법을 결합하여 소프트웨어 상관기에 적용하여 실제 상관처리 실험을 수행하고자 한다.

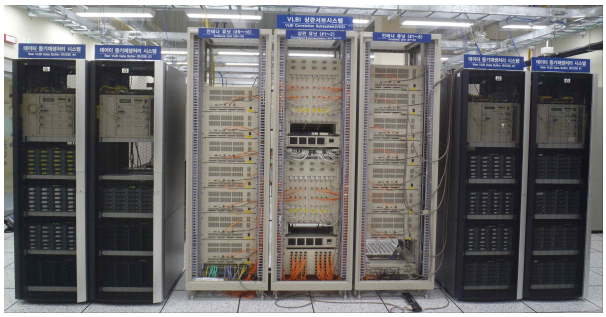

그림 1. 한일공동VLBI상관기의 설치모습.
Fig. 1. Korea-Japan Joint VLBI Correlator at Korea Japan Correlation Center.

Ⅲ. 제안하는 최적 코딩 방법

3.1 fftw[7] 라이브러리

fftw는 1차원 또는 다차원 DFT[8]를 계산하기 위한 C 언어 기반의 라이브러리이다. 라이브러리 패키지는 MIT[9]에 의해 개발되었으며, 실수 또는 복소수를 모두 계산할 수 있다. fftw 라이브러리는 일반에게 무료로 제공하고 있으며, 특정 서버나 워크스테이션을 기준으로 다른 웹사이트에서 제공하는 라이브러리와 함께 계산의 정확성과 속도를 측정한 그 결과를 그래프로 제공하고 있는데, 그림 2에 나타내었다.

그림 2의 비교 실험에 사용한 컴퓨터는 2개 코어 인텔 Xeon 3GHz CPU, 4MB L2 캐시를 장착하고 있으며 리눅스 커널 2.6.17 64비트 모드를 기본으로 하고 있다. 코드를 컴파일하기 위한 컴파일러는 인텔 C/C++ 컴파일러 9.1.043, 인텔 Fortran 컴파일러 9.1.037, 인텔 MKL[10] 8.1.1, 그리고 SSE(Streaming SIMD[11] Extensions), SSE2, SSE3를 갖고 있는 인텔 IPP[12] 5.1.1이다. fftw 라이브러리를 사용하기 위한 컴파일 플래그 옵션은 다음과 같다.

C : icc -no-gcc -O3 -xT
C++ : icc -Kc++ -no-gcc -O3 -xT
Fortran: ifort -O3 -WB -xT

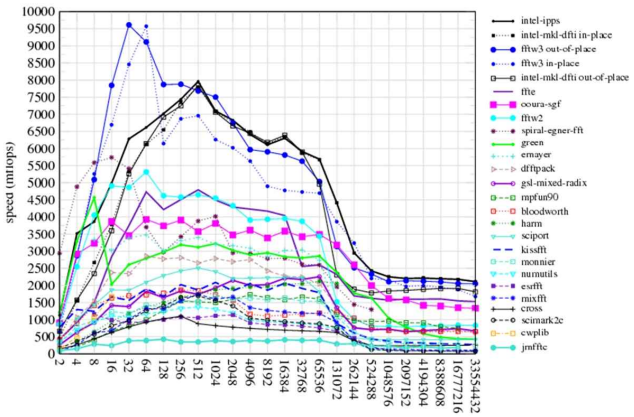

그림 2. 인텔 Xeon 3GHz CPU 서버에서의 fftw, 인텔 IPP 등을 활용한 DFT 연산 비교 결과[13] 그래프
(1차원 64비트 복소 실수 연산)
Fig. 2. Comparison result graph of DFT calculation using fftw and Intel IPP in Intel Xeon 3GHz CPU
(one dimension 64 bit complex float calculation)

DFT의 경우 2,048번째 채널 미만에서는 fftw의 연산 속도가 빠르며, 2,048번째 채널 이상에서는 인텔의 IPP를 활용한 경우 연산 성능이 약간 상회함을 볼 수 있다. 인텔의 IPP는 상업용 소프트웨어이기도 하지만, 본 논문에서는 DFT의 2,048채널 미만에서는 효율이 뛰어나고 기타 부분에서는 비슷한 성능을 갖고 있기 때문에, 그림 3의 비교결과를 바탕으로 소프트웨어 상관기의 최적 코딩 방법으로 적용하였다.

fftw 라이브러리 함수를 정리하면 다음과 같다.

-fftw_malloc: FFT 연산을 위한 복소 실수(complex float) 변수 메모리를 할당한다.
-fftw_plan_dft_xx: FFT 연산을 위한 계획(plan)을 수립한다.
-fftw_execute: 수립된 계획에 따라 FFT 연산을 실행한다.
-fftw_destroy_plan: 수립된 계획을 해제한다.
-fftw_free: 할당된 변수 메모리를 해제한다.

그림 3은 최적코딩을 위해 fftw 라이브러리 함수를 사용하여 2,048채널을 갖는 1차원 FFT 연산 예를 나타낸 것이다.

7) http://www.fftw.org
8) Discrete Fourier Transform
9) Massachusetts Institute of Technology
10) Math Kernel Library
11) Single Instruction Multiple Data
12) Integrated Performance Primitives
13) http://www.fftw.org/speed/CoreDuo-3.0GHz-icc64

```
#include <fftw3.h>
int main(void){
    fftw_complex* ina;
    fftw_complex* result;
    fftw_plan plana;
    int nch = 2048;

    ina = (fftw_complex*)fftw_malloc(sizeof(fftw_complex)*nch);
    result = (fftw_complex*)fftw_malloc(sizeof(fftw_complex)*nch);
    plana = fftw_plan_dft_1d(nch, ina, result, fftw_FORWARD,
fftw_ESTIMATE);

    fftw_execute(plana);
    fftw_destroy_plan(plana);
    fftw_free(ina);
    fftw_free(result);
    return 0;
}
```

그림 3. fftw 라이브러리를 이용한 2048 채널의 1차원 FFT 예
Fig. 3. Example of 1-dimension FFT with 2048 channel using fftw library

3.2 SSE[14] 라이브러리를 활용한 방법

SSE는 x86 아키텍처에 대한 단일 명령 다중 데이터(Single Instruction Multiple Data, SIMD) 명령어 집합의 확장으로 병렬연산이 가능하다. x86(32비트) 아키텍처에는 128비트 XMM0 – XMM7 까지 8개의 연산 레지스터가 있으며, x86-64에는 XMM8 – XMM15 까지 8개의 연산 레지스터가 추가되어 있다. SSE2 라이브러리에서는 연산 레지스터를 128비트로 확장함으로써 다음과 같은 병렬 연산을 한 번의 연산 주기에 수행할 수 있는 장점이 있다.

- 4개의 32비트 단정도 부동 소수점(Single precision floating point) 연산
- 2개의 64비트 배정도 부동 소수점(Double precision floating point) 연산
- 2개의 64비트 정수(Integer) 연산
- 4개의 32비트 정수 연산
- 8개의 16비트 정수 연산
- 16개의 8비트 정수 연산

연산 명령어는 MOVSS, MOVAPS, ADDSS 등 70여 가지의 기계어 코드를 제공한다. 그리고 C/C++ 코드에서는 컴파일러 내장 함수를 이용하여 코딩을 간단한 형태로 작성할 수 있는 장점이 있다.

SSE 라이브러리에서 __m128 변수 형태는 SSE 처리 유닛에 있는 XMM 레지스터와 직접 대응되는 변수 형태이다. __m128 변수 형태는 float(32비트), short(16비트), char(8비트) 변수를 한 번에 다중으로 처리할 수 있다. 여기서, 변수는 메모리에서 16바이트씩 정렬된 상태로 위치해 있어야 한다. 변수를 생성할 때, _aligned_malloc()함수를 이용하거나 정렬되지 않은 상태의 변수라면 _mm_loadu_ps() 명령어를 사용할 수 있다.

3.3 OpenMP[15] 라이브러리를 활용한 방법

OpenMP(Open Multi-Processing)는 공유메모리 다중 처리 프로그래밍 API[16]이다. 1997년에 포트란용 API가 공개되었으며 2002년에 C/C++ 규격으로 제안되었다.

지난 수십 년간 컴퓨터의 성능을 나타내는 CPU는 기술 개발을 통하여 많은 발전을 거듭하였다. 그러나 최근 회로 집적 기술의 한계로 인하여 개발 속도가 점점 늦어지고 있는 실정이다. 만약 전자 한 개를 이용하여 트랜지스터를 구성할 수 있다면, 집적회로의 집적도 및 처리 속도를 개선하는데 획기적인 계기가 될 수 있을 것이다. 앞에서 설명한 것과 같이 기술 한계를 극복하기 위해 많은 반도체 회사들은 다중 코어(Multi-core)를 장착한 CPU를 생산하고 있다. 다중 코어를 채용한 CPU를 사용하고 병렬처리 연산을 수행함으로써 기존 단일 코어 CPU보다 몇 배나 빠른 시간 안에 계산 결과를 확인할 수 있게 되었다.

다중 코어를 지원하는 다중 쓰레드(Multi-thread) 코드를 작성하기 위한 적응 다중 쓰레드 프로그램(Adaptive Multi Thread Programming) 방법으로 OpenMP가 일반에게 공개되어 있다. OpenMP는 손쉽게 코드에 추가하여 사용할 수 있기 때문에 고급 프로그램 과정을 학습하지 않고도 쉽게 최적코딩을 통하여 병렬처리 연산을 수행하는 것이 가능하다.

OpenMP의 병렬처리 연산을 활용하여 기본적으로 사용하는 CPU 코어의 개수와 상관없이 쓰레드의 수를 늘리고 싶다면 omp_set_num_threads(int) 함수를 사용하거나 OMP_NUM_THREADS 라는 환경변수(예. export OMP_NUM_THREADS=16)를 적용하면 쉽게 코딩하는 것이 가능하다.

3.4 공유메모리[17]를 사용한 방법

공유메모리(Shared memory) 방법은 프로세스(Process) 사이 또는 스레드(Thread) 사이의 데이터 전송을 목적으로 사용한다. 각각의 프로세스 또는 스레드에서 필요한 메모리를 각각 생성하여 사용할 경우에 메모리 관리자의 효율이 나빠져 시스템 성능의 저하를 초래할 수 있다. 이를 방지하고자 프로세스나 스레드 사이에 하나의 메모리만을 생성하여 서로 공유하여 사용할 수 있다. 동작 방식은 열쇠(Key)를 가진 프로세스가 메모리를 지배하고 나머지 프로세스는 접근할 수 없는 비교적 단순한 동작을 한다.

14) https://en.wikipedia.org/wiki/Streaming_SIMD_Extensions

15) http://www.openmp.org
16) Application Programming Interface
17) https://en.wikipedia.org/wiki/Shared_memory

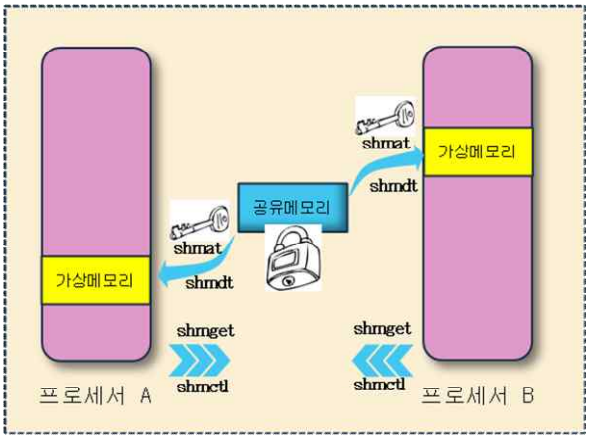

그림 4. 공유메모리와 함수의 관계
Fig. 4. Relation between shared memory and function

그림 4는 프로세스 A와 프로세스 B 사이에 생성된 공유메모리와 함수의 관계에 대해 공유메모리 사용 시퀀스를 그림으로 나타낸 것이다.

공유메모리를 생성하여 접근하는 함수는 다음과 같다.

- shmget : 공유메모리를 만들고 제어할 수 있는 핸들(Handle) 값을 가져온다.
- shmctl : 공유메모리의 상태 값을 제어하거나 제거한다.
- shmat : 공유메모리를 현재 프로세스에서 사용할 수 있도록 연결(attach)한다.
- shmdt : 공유메모리의 사용이 종료되었으면 연결을 해지(detach)한다.

Ⅳ. 실험 및 결과

4.1 상관처리 흐름도

본 연구에서 기존에 개발한 소프트웨어 상관기는 입력된 관측 데이터를 정렬하는 비트스트림 블록, 위치에 따른 기하학적인 지연을 보정하는 지연 추적 블록, 지구의 공전, 자전 및 도플러 편이(Doppler effect)를 보정하는 프린지 회전 블록, 시간 영역의 데이터를 주파수 영역의 데이터로 변환하는 복소 FFT(complex-FFT) 블록, 두 개의 관측국으로부터 수신된 신호를 곱하는 곱셈블록 및 적분 블록으로 구성된다. 그림 5는 A와 B 관측국에 대한 VLBI 상관처리 순서를 예로 들어 소프트웨어 상관기의 데이터처리 순서도를 나타낸 것이다. 상관처리 기선의 수는 관측국이 만약 n개 이면 n(n-1)/2의 수이며 그 만큼 상관처리를 수행한다.

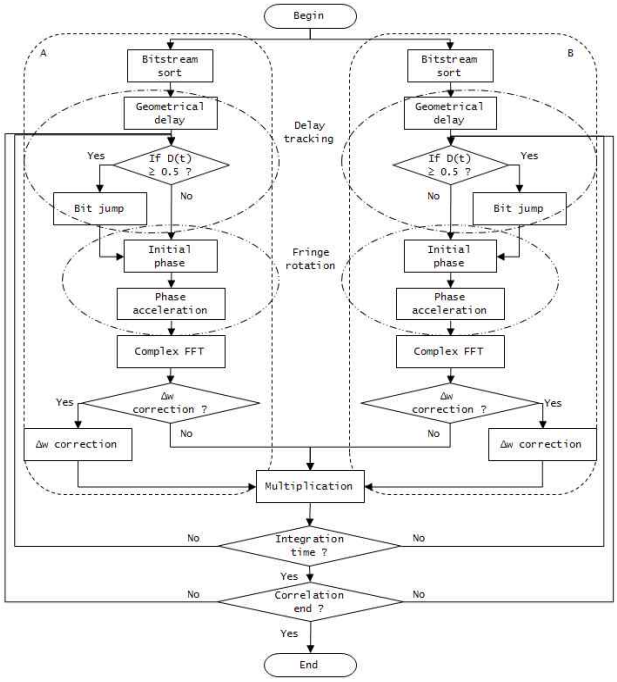

그림 5. 상관처리 흐름도[2]
Fig. 5. Correlation processing flowchart

4.2 실험 서버 및 관측데이터

그림 6에 나타낸 것과 같이 실험에 사용한 서버는 12개 코어 인텔 코어 i7 4960X 3.6GHz, 15MB L2 캐시 CPU를 장착하고 있다. 마더보드(Mother Board)는 ASRock X79 extreme 11, 16GB RAM, ARECA ARC-1882ix-12 RAID 컨트롤러, 스토리지는 Seagate NAS 3TB 하드디스크를 장착하고 있는 STARDOM ST8-S2P 24TB로 구성되어 있다.

그림 6. 실험에 사용된 서버
Fig. 6. Server system used in experiment

본 연구의 제안방법을 검증하기 위해 실험에 사용된 관측데이터의 관측명은 r14056b이며, KVN 3개 관측국(연세, 울산, 탐라)에서 2014년 56일째에 관측하였다. 관측미디어에 기록된 데이터는 1.024Gbps 속도로 저장되어 있으며

Mark5B[5] 형식으로 구성되어 있다. KVN DAS(Data Acquisition System)[6]의 디지털 필터 시스템에서 16스트림(stream)으로 분할되어 있다. 전파망원경으로 수신된 신호는 초고속 샘플러(High-speed sampler)에서 1024 MHz, 2-bit로 샘플링 되었으며, 각 스트림의 대역폭은 16MHz이다. 소프트웨어 상관처리를 위한 적분 시간은 0.4096초이고 상관처리를 수행하는 시간은 15초이다. VLBI 관측은 대상 천체에 대해 22 GHz 주파수로 관측하였으며, 16개 각 스트림의 주파수는 22.091 GHz부터 16MHz 대역씩 증가하는 설정으로 구성되며, 표 1에 자세히 나타내었다.

표 1. r14056b의 주파수 테이블
Table 1. Frequency table for r14056b

스트림 (stream)	중심 주파수[GHz]	주파수 밴드	대역폭 [MHz]
1	22.107	LSB	16
2	22.123	USB	16
3	22.139	LSB	16
4	22.155	USB	16
5	22.171	LSB	16
6	22.187	USB	16
7	22.203	LSB	16
8	22.219	USB	16
9	22.235	LSB	16
10	22.251	USB	16
11	22.267	LSB	16
12	22.283	USB	16
13	22.299	LSB	16
14	22.315	USB	16
15	22.331	LSB	16
16	22.247	USB	16

상관처리 실험에서 각 비트의 대응 값은 표 2와 같다. 이번 실험에 사용된 천체의 각각 비트 분포(Bit distribution)는 1:3:3:1을 갖는다.

표 2. 상관처리에서 대응되는 비트 값
Table 2. Bit value applied in correlation processing

비트	00	01	10	11
값	-1.0	-0.25	0.25	1.0

4.3 실험결과

그림 7은 본 연구에서 제안한 최적 코딩 방법을 적용한 소프트웨어 상관기의 처리속도를 확인하기 위해 수행한 상관처리 실험의 예를 나타낸 것이다. 그림 7의 실제 상관처리를 각각의 제안방법으로 구현한 상관 알고리즘을 소프트웨어 상관기에 적용하여 실험을 수행하였다.

그림 8은 상관처리 실험에 사용된 r14056b 관측에 대한 상과처리 결과를 보여준다. 관측에 참가한 관측국은 KVN 3 관측국(연세, 울산, 탐라)이다. 그림 8의 좌측부터 1열은 연세 관측국의 자기상관스펙트럼, 2열은 울산 관측국의 자기상관스펙트럼, 3열은 탐라 관측국의 자기상관스펙트럼 결과이다. 그림 8의 좌측부터 4열은 연세-울산 기선의 상호상관스펙트럼, 위상(phase), 그리고 프린지(fringe) 검출 결과를 보여준다. 5열은 연세-탐라 기선의 결과이고, 6열은 울산-탐라 기선 간의 상관처리 결과를 보여준다.

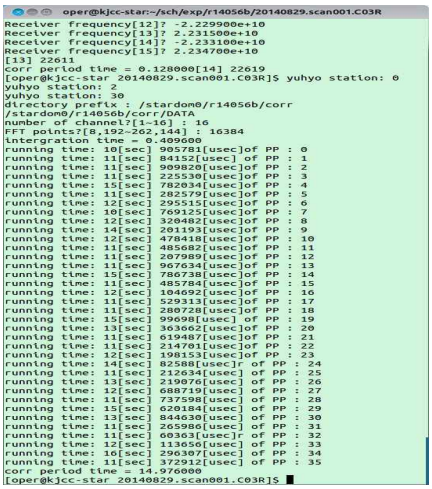

그림 7. 최적 코딩 방법을 적용한 소프트웨어 상관기의 처리속도를 확인하기 위한 상관처리 예
Fig. 7. Example of correlation processing for confirming the processing speed of software correlator applied by optimum coding method

그림 8. 상관처리 실험에 사용된 r14056b의 상관처리 결과 이미지(좌측 3열: 자기상관스펙트럼, 우측 3열: 상호상관스펙트럼과 프린지)
Fig. 8. Correlation result image of r14056b used in correlation processing experiment(left 3 column : auto correlation, right 3 column : cross correlation and fringe)

4.3.1 fftw 라이브러리만 사용한 실험

본 연구에서 제안한 최적 코딩 방법 중에서 기본이 되는 방법으로 fftw 라이브러리만을 상관 알고리즘에 적용하여 상관처리 속도에 대한 실험을 수행하였으며, 그 결과를 그림 9에 나타내었다. 실험결과 KVN 3관측국의 3개 자기상

관(auto correlation)과 3개 상호상관(cross correlation)에 대한 1Gbps, 16스트림, 16,384 FFT 채널, 15초 분량의 상관처리에 걸린 시간은 총 449.11초이다. 상관처리 중에 0.4096초의 적분시간에 소요된 평균 처리 시간은 12.48초이다.

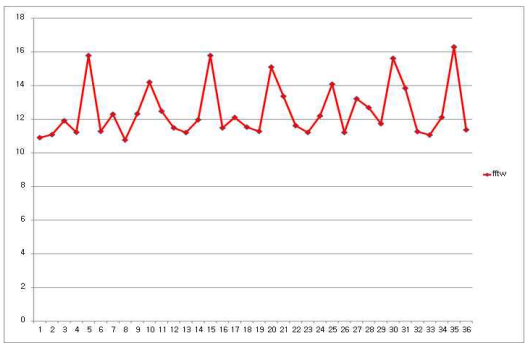

그림 9. fftw 라이브러리만을 사용한 실험결과
Fig. 9. Experimental result by using only fftw library

4.3.2 fftw와 SSE 라이브러리를 사용한 실험

두 번째는 기본이 되는 fftw와 병렬 연산을 수행하는 SSE 라이브러리를 결합한 최적 코딩 방법을 상관 알고리즘에 적용하여 수행한 상관처리 실험결과를 그림 10에 나타내었다. 실험결과 KVN 3관측국의 3개 자기상관과 3개 상호상관에 대한 1Gbps, 16스트림, 16,384 FFT 채널, 15초 분량의 상관처리에 걸린 시간은 총 638.73초이다. 0.4096초의 상관적분시간에 소요된 상관처리 시간은 17.74초이다. fftw만을 사용한 경우와 비교하여 평균 처리시간이 증가한 결과를 보였다.

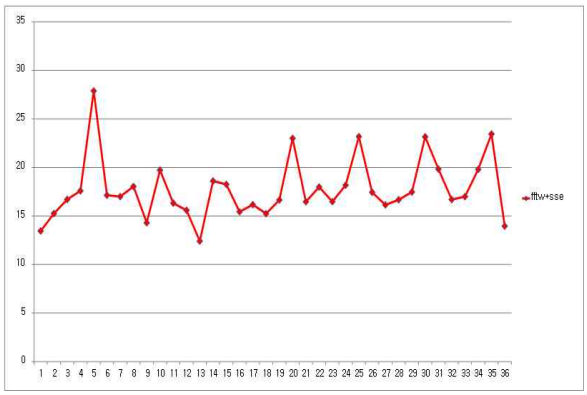

그림 10. fftw와 SSE 라이브러리를 사용한 실험결과
Fig. 10. Experimental result by using fftw and SSE library

4.3.3 fftw, SSE 라이브러리와 OpenMP를 사용한 실험

세 번째는 기본 알고리즘인 fftw와 병렬연산을 지원하는 SSE, 그리고 병렬 프로세스를 지원하는 OpenMP를 결합하여 상관 알고리즘을 적용한 후 수행한 상관처리 실험결과를 그림 11에 나타내었다. 실험결과 KVN 3관측국의 3개 자기상관과 3개 상호상관에 대한 1Gbps, 16스트림, 16,384 FFT 채널, 15초 분량의 상관처리에 걸린 시간은 총 657.23초이다. 앞의 실험에서 사용한 관측데이터 대해 0.4096초의 상관적분시간에 소요된 처리시간은 18.26초이며, 처리시간이 조금씩 증가하는 결과를 보이고 있다.

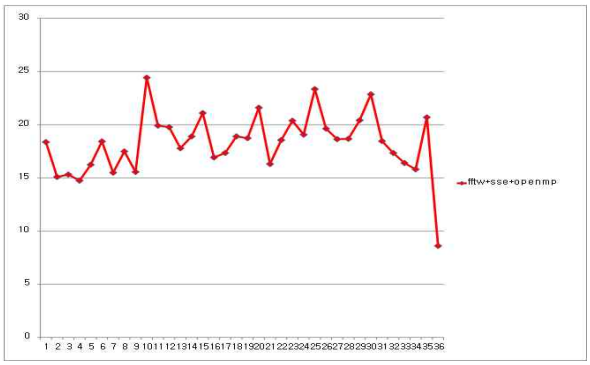

그림 11. fftw, SSE 라이브러리와 OpenMP를 사용한 실험결과
Fig. 11. Experimental result by using fftw, SSE library and OpenMP

4.3.4 fftw 라이브러리와 공유메모리를 사용한 실험

네 번째는 공유메모리를 적용한 방법의 유효성을 확인하기 위해 기본 알고리즘인 fftw 라이브러리와 공유메모리만을 적용하여 동일한 관측데이터를 대상으로 상관처리 실험을 수행하였으며, 그 결과를 그림 12에 나타내었다. 실험결과 KVN 3관측국의 3개 자기상관과 3개 상호상관에 대한 1Gbps, 16스트림, 16,384 FFT 채널, 15초 분량의 상관처리에 걸린 시간은 총 335.34초이다. 기본 알고리즘인 fftw 라이브러리와 공유메모리를 적용한 경우 동일 관측데이터에 대해 0.4096초의 적분시간에 소요된 상관처리시간이 9.31초로 앞에서 수행한 방법들에 비하여 시간이 많이 단축되는 것을 알 수 있다. 이 결과를 바탕으로 공유메모리 방법이 상관알고리즘에 적용하면 상관처리 시간을 줄이는데 매우 효과적일 것으로 판단되어 기본 알고리즘인 fftw 라이브러리와 병렬처리 프로세스를 지원하는 OpenMP와 병합하여 그 유효성을 확인하기 위해 실험을 수행하였으며, 그 결과는 다음 절에서 기술한다.

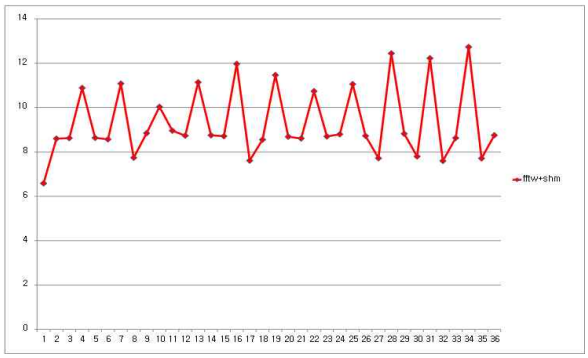

그림 12. fftw 라이브러리와 공유메모리를 사용한 실험결과
Fig. 12. Experimental result by using only fftw library and shared memory

4.3.5 fftw 라이브러리, 공유메모리와 OpenMP를 사용한 실험

앞에서 소개한 것과 같이 그림 13은 fftw 라이브러리, 공유메모리 방법, 그리고 병렬 프로세스를 지원하는 OpenMP를 사용한 실험결과를 나타낸 것이다. 동일한 실험데이터를 대상으로 한 실험한 결과 15초 분량의 관측데이터를 상관처리하는데 소요된 총 시간은 329.82초이며, 평균 9.16초가 소요되었다. 본 연구에서 제한한 최적 코딩 방법들 중에서 fftw와 공유메모리를 적용한 방법보다 약 0.15초 상관처리 시간을 줄일 수 있음을 실험으로 확인하였다.

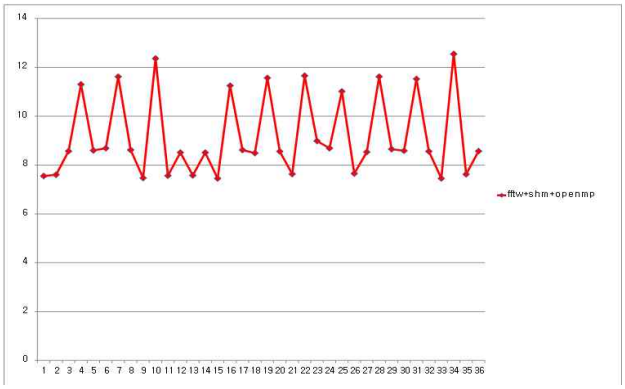

그림 13. fftw 라이브러리, 공유메모리와 OpenMP를 사용한 실험결과
Fig. 13. Experimental result by using fftw library, shared memory and OpenMP

4.4 결과 고찰

본 연구에서 제안한 소프트웨어 상관기의 상관 계산 코드의 최적 코딩 방법의 유효성을 확인하기 위해 5가지 상관처리 실험에 대한 결과를 그림 14에 나타내었다. 우선 소프트웨어 상관기에서 FFT 연산을 수행하기 위해 도입한 기본 알고리즘인 fftw 라이브러리만을 적용한 경우 상관처리 시간은 평균 12.48초가 소요되었다. 그리고 실험 가운데 상관처리 시간이 가장 단축된 최적 코딩 방법은 fftw 라이브러리, 공유메모리와 OpenMP를 적용한 경우이고, 상관처리 시간이 가장 긴 경우의 방법은 fftw와 SSE 라이브러리 및 OpenMP를 적용한 경우이다. 병렬연산을 지원하는 SSE 라이브러리를 적용한 경우, 본 연구의 실험에서는 성능이 저하되는 결과를 보인 이유는 상관처리 알고리즘의 상관계산이 합성곱(Convolution)으로 수행되기 때문이다. 즉, SSE는 16바이트로 정렬되어 메모리에 할당된 변수를 각각 독립적으로 연산하는 방법에는 적합하지만, 합성곱 연산은 바로 이웃하는 샘플과의 곱셈과 덧셈 연산을 수행하기 때문에 메모리에 쓰고 읽는 부하가 많이 걸리게 된다. 그리고 병렬처리 프로세스를 지원하는 OpenMP를 적용한 경우의 성능이 미미한 것은 fftw 계산 함수가 프로세서의 코어를 모두 사용하기 때문이다. fftw 라이브러리 역시, 계산속도의 향상을 위해 다중 스레드(Multi-thread)를 사용하는 방향으로 구현되어 있기 때문이다. OpenMP의 성능을 극대화하고자 한다면, 현재보다 더 많은 코어를 갖는 프로세서가 장착된 시스템을 선택하면 성능이 개선될 것으로 판단된다.

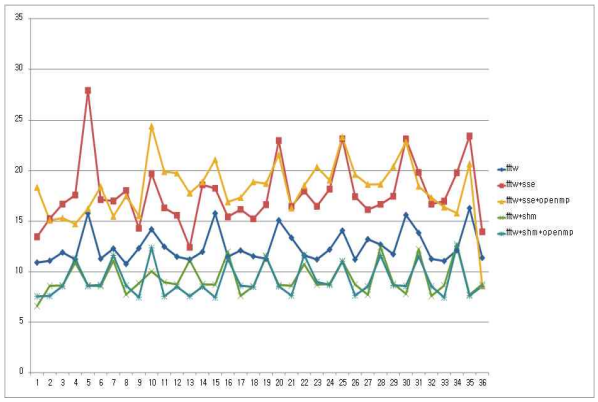

그림 14. 각 실험에 대한 비교결과
Fig. 14. Comparison result according to each experiments

그림 14에 나타낸 것과 같이 15초 분량의 관측데이터를 최적 코딩 방법을 적용한 소프트웨어 상관기로 각각 상관처리하는데, 평균적으로 fftw 라이브러리만을 사용한 경우 12.48초, SSE 라이브러리를 추가한 경우 17.74초, SSE에 OpenMP를 적용한 경우 18.26초의 시간이 소요되었다. fftw 라이브러리에 공유메모리를 추가한 경우에는 9.31초, 공유메모리에 OpenMP를 추가한 경우에는 9.16초가 상관처리 시간이 소요되었다. 상관적분시간 0.4096초를 계산하는데 걸린 평균 시간은 그림 15에 나타내었다.

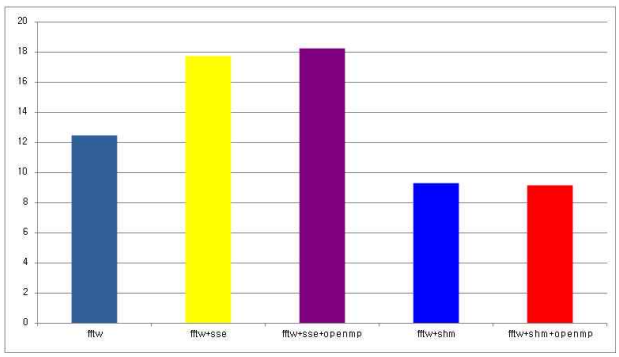

그림 15. 0.4096초 상관적분에 소요된 각 실험의 평균 시간
Fig. 15. Average used time for 0.4096sec correlation integration of each experiment

표 3. fftw라이브러리만을 사용한 상관처리 방법 대비 각 실험의 연산속도 비
Table 3. Processing speed ratio of each experiments compared with correlation method used by only fftw library

	fftw	fftw+sse	fftw+sse +openmp	fftw+s hm	fftw+shm +openmp
fftw대비 연산속도	1	1.422	1.463	0.747	0.734

표 3에 나태난 것과 같이 fftw 라이브러리만을 사용한 경우를 1로 설정하였을 때, SSE를 사용한 경우에는 42.2%의 자원(resource)을 더 소모하였고, SSE와 OpenMP를 함께 사용한 경우에는 46.3%의 자원을 더 사용하였다. fftw와 공유메모리를 사용한 경우에는 25.3%의 자원을 아낄 수 있었으며, 공유메모리와 OpenMP를 함께 사용한 경우에는 26.6%의 자원을 아낄 수 있었다. SSE와 OpenMP를 함께 사용할 경우에는, OpenMP를 적용하는 메모리 변수를 서로 독립적으로 연산되도록 설계를 해야 하지만, SSE를 사용하면서 메모리 변수 사이의 독립성이 없어져 효율이 가장 나빠졌을 것으로 판단한다. 따라서 서로 독립된 메모리 변수를 사용할 경우에는 SSE와 OpenMP를 함께 사용하는 것이 연산 효율이 개선될 것으로 생각된다. 본 논문에서 제안한 최적 코딩 방법 중에서 fftw 라이브러리에 공유메모리와 OpenMP를 사용한 경우가 가장 성능이 우수한 결과를 나타내었다. 향후 OpenMP 병렬 처리 연산의 성능을 높이고자 할 경우에는 프로세서 코어가 많이 장착된 서버를 활용하면 좋을 것으로 판단한다.

Ⅴ. 결 론

본 논문에서는 소프트웨어로 구축한 상관시스템의 성능 개선을 위해 공개된 fftw 라이브러리를 이용한 최적 코딩 방법을 제안하였다. VLBI 관측데이터를 상관처리하기 위해 하드웨어로 구축한 시스템과 비교하여 관측시간대비 준실시간 또는 실시간으로 수행하는 소프트웨어 상관기를 구축하기 위해 최적성능을 갖는 코딩방법을 적용할 필요가 있으며, 기존에 개발된 공개 알고리즘을 활용하여 최적인 시스템을 구성하면 시간과 비용을 줄일 수 있다. 따라서 본 논문에서는 상관처리 시스템에서 가장 중요한 과정인 FFT 처리과정에서 공개된 fftw 라이브러리를 활용하여 최적코딩을 수행할 수 있도록 fftw 라이브러리를 이용한 일반적인 방법, 병렬연산을 수행하는 SSE를 활용한 방법, 공유메모리를 사용한 방법, OpenMP를 활용한 방법, 그리고 제시한 것들을 병합한 방법을 적용한 경우에 대해 상관처리 실험을 수행하였다. 상관처리실험을 통하여 본 논문에서 제안한 최적 코딩 방법 중에서 fftw 라이브러리와 공유메모리, 그리고 OpenMP를 상관처리 알고리즘에 적용한 경우 상관처리 시간을 단축할 수 있어서 소프트웨어 상관기의 성능개선에 유효함을 확인하였다.

향후에는 OpenMP 방법의 성능을 최대화하기 위해 프로세서 코어가 많이 장착된 시스템을 도입하여 본 논문에서 제안한 최적 코딩 방법을 소프트웨어 상관기에 적용하여 시스템의 성능을 개선하고자 한다.

참 고 문 헌

[1] Deller, A. T., Brisken, W. F., Phillips, C. J., et al., DiFX-2: A More Flexible, Efficient, Robust, and Powerful Software Correlator, PASP, Vol. 123, pp. 275, 2011.

[2] 염재환, 오세진, 노덕규 외 4명, “한일공동VLBI상관기를 위한 소프트웨어 상관기의 개발,” 한국우주과학회지 제26권 4호, pp. 567-588, 2009.

[3] 노덕규, 오세진, 염재환 외 15명, “2008년도 한일공동 VLBI상관기 및 수신기 개발 결과보고서,” 한국천문연구원, pp. 3-100, 2008.

[4] 오세진, 노덕규, 염재환 외 6명, “VLBI상관서브시스템 본제품의 제작현장 성능시험,” 신호처리시스템학회 논문지 제12권, 제4호, pp. 322-331, 2011.

[5] 오세진, 노덕규, 김광동 외 5명, “관측 데이터의 고속기록을 위한 대용량 저장시스템,” 천문학논총, 제19권, pp. 83~92, 2004.

[6] Se-Jin Oh, Duk-Gyoo Roh, Kiyoaki Wajima et. al., “Design and Development of a High-Speed Data Acquisition System for the Korean VLBI Network,” Publication of Astronomical Society of Japan, Vol. 63, pp. 1229-1242, 2011 December 25.

신 재 식(Jae-Sik Shin)

1989년 2월 한남대 회계학과(학사)
2014년 2월 한남대 멀티미디어과(석사수료)
1991년 ~ 2003년 소프트웨어 개발
2004년 ~ 2013년 한국천문연구원 전산팀장
2014년 ~ 현재 한국천문연구원 책임기술원
※주관심분야 : VLBI 관측 프로그래밍, VLBI-WEB 인터페이스 개발

오 세 진(Se-Jin Oh)

正會員
1996년 2월 영남대 전자공학과(학사)
1998년 2월 영남대 전자공학과(석사)
2002년 2월 영남대 전자공학과(박사)
2001년 9월~2002년 12월 대구과학대학 교수
2010년 6월~2011년 5월 한국천문연구원 그룹장
2002년 12월 ~ 2014년 12월 한국천문연구원 선임연구원
2015년 1월 ~ 현재 한국천문연구원 책임연구원
※주관심분야 : 디지털신호처리, VLBI상관기 개발, 천문관측기기개발

염 재 환(Jae-Hwan Yeom)

正會員
2005년 8월 한양대 정밀기계공(석사)
2005년~현재 한국천문연구원 선임연구원

※주관심분야 : 디지털신호처리, VLBI상관기 개발

노 덕 규(Duk-Gyoo Roh)

正會員

1985년 2월 서울대 천문학과(이학사)
1994년 8월 동경대 천문학과(이학석사)
1997년 8월 동경대 천문학과(박사수료)
1985년 4월~ 현재 한국천문연구원 책임연구원
2005년 11월~2009년 3월 한국천문연구원 그룹장
※주관심분야 : 전파천문, VLBI상관기 개발

정 동 규(Dong-Kyu Jung)

2004년 8월 충남대 천문학과(이학사)
2006년 8월 충남대 천문학과(석사수료)
2012년 1월 ~ 현재 한국천문연구원 연구원
※주관심분야 : VLBI상관처리, 천문관측기기 개발

오 충 식(ChungSik Oh)

2002년 2월 서울대 천문학과(이학사)
2006년 3월 동경대 천문학과(이학석사)
2009년 3월 동경대 천문학과(이학박사)
2009년 4월-2010년 11월 한국천문연구원 박사후 연수원
2010년 12월 - 현재 한국천문연구원 선임연구원
※주관심분야 : 전파천문, Astrometry, VLBI상관처리

황 주 연(Ju-Yeon Hwang)

2006년 2월 전남대 자원공학과(공학사)
2014년 ~ 현재 한국천문연구원 연구원
※주관심분야 : VLBI상관처리, 천문관측기기 개발

소 요 환(Yo-Hwan So)

1995년 홍익대학교 서양화전공(석사)
1998년 미국 New York Institute of Technology필름/애니메이션전공(석사)
2003년~현재 한남대학교 멀티미디어학부 교수
※주관심분야 : 3D애니메이션, 가상현실