논문 2016-11-29

대칭형 멀티코어 커널에서 DBS(Doppler Beam Sharpening) 알고리즘 실시간 구현

(Real-Time Implementation of Doppler Beam Sharpening in a SMP Multi-Core Kernel)

공 영 주*, 우 선 걸
(Young-Joo Kong, Seon-Keol Woo)

Abstract: The multi-core technology has become pervasive in embedded systems. An implementation of the Doppler Beam Sharpening algorithm that improves the azimuth resolution by using doppler frequency shift is possible only in multi-core environment because of the amount of calculation. In this paper, we design of multi-core architecture for a real time implementation of DBS algorithm. And based on designed structure, we produce a DBS image on P4080 board.

Keywords : Multi-core, Doppler beam sharpening, Symmetric multiprocessing, Real-time, Doppler processing

Ⅰ. 서 론

복수의 코어를 하나의 칩에 집적하여 병렬적으로 성능을 높이는 기술인 멀티코어 기술이 임베디드 환경에서 보편화되고 있다. 멀티코어 환경에서는 알고리즘 성능 및 속도 향상을 기대할 수 있기 때문이다 [1]. 그러나 성능의 증대는 어떤 운영체제에 사용하느냐에 따라서 달라지기도 하며, 코어에 Task할당을 어떻게 하느냐에 따라 달라지기도 한다. 따라서 멀티코어를 효율적으로 사용하기 위해서는 적절한 운영체제에 맞는 멀티코어 스케줄러가 매우 중요하다.

DBS(Doppler Beam Sharpening, 이하 DBS) 알고리즘은 SAR(Synthetic Aperture Radar, 이하 SAR)의 한 기법으로 도플러 주파수 천이 개념을 이용하여 방위각 해상도를 향상시키는 기술이다. 그림 1처럼 같은 거리에 표적과 클러터 성분이 동시에 존재할 경우 기존의 파형으로는 표적과 클러터 성분이 겹쳐져서 보이기 때문에 구분이 힘들지만

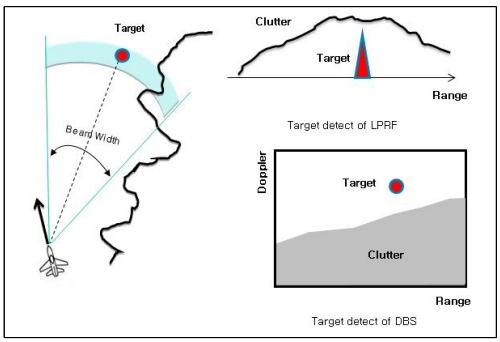

그림 1. Low PRF와 DBS
Fig. 1 Low PRF and DBS

DBS 알고리즘을 이용하며 DBS 2-D 영상으로 이두 성분의 구분이 가능하다 [2, 3]. 하지만 DBS 알고리즘은 많은 연산이 필요하기 때문에 DSP 혹은 싱글코어로는 구현이 힘들다. 멀티코어 환경이 보편화된 지금에서야 DBS 알고리즘 구현이 현실화되고 있는 실정이다.

본 논문에서는 임베디드 운용환경 중 SMP (Symmetric Multi Processing, 이하 SMP) 멀티코어 환경에서 효율적으로 DBS 알고리즘을 설계하는 방법과 그에 따른 DBS 알고리즘이 수행되는 과정을 소개한다. 특히 기존의 싱글코어 혹은 멀티코어

*Corresponding Author (yjkong16@lignex1.com)
Received: 2 June 2016, Revised: 11 July 2016, Accepted: 21 July 2016.
Y.J. Kong, S.K. Woo: LIG Nex1

ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2016.11.4.251

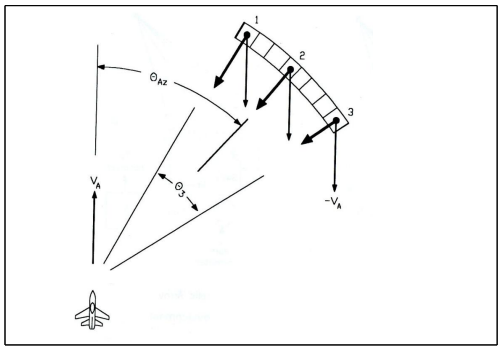

그림 2. DBS 원리
Fig. 2 DBS Principle

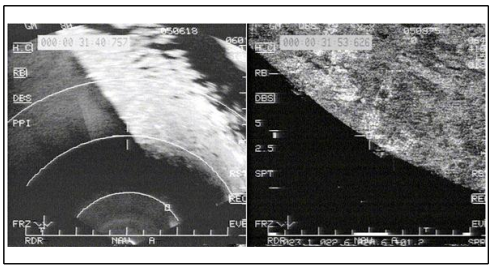

그림 3. 레이더 영상(왼쪽)과
DBS 영상(오른쪽) 비교
Fig. 3 Radar image(Left) and
DBS image(Right)

환경에서 DBS 영상을 생성 시 사용하지 않았던 세 마포어 동기 및 메모리 공유를 이용하여 연산시간을 단축하였고 알고리즘 단순화를 이루어냈다. 마지막으로 제시된 알고리즘 순서도에 따른 시뮬레이션 결과를 보여준다.

Ⅱ. 본 론 (1)

1. DBS 알고리즘 원리

DBS 알고리즘은 SAR의 한 기법으로 레이더의 도플러 필터를 이용하여 각도 해상도를 향상시키는 기법이다. 일반적으로 방위각 해상도를 향상시켜 표적 구분 능력을 증대시키는데, 특히 일정한 속도와 방향으로 움직이는 물체에 효과적이다. 그림 2와 같이 3개의 표적이 하나의 빔 폭 안에 들어오게 되어 같은 거리 셀에 위치하게 되면 일반적인 레이더에서는 하나의 표적으로 인식하게 되어 세 표적에 대한 구분이 힘들어진다.

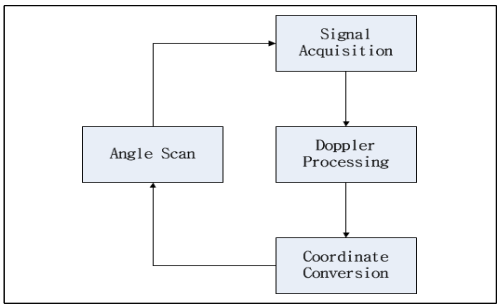

그림 4. DBS 알고리즘 순서도
Fig. 4 DBS Algorithm Flowchart

그러나 DBS 알고리즘을 이용하면 각 표적과의 상대적 각도는 표적마다 미세하게 차이가 발생한다. 이러한 미세하게 차이가 나는 각도는 상대적 속도의 차이를 만들게 되며, 이로 인해 도플러 천이 현상이 발생한다 [4]. DBS 알고리즘은 이런 도플러 천이 현상을 이용하여 방위각 방향으로 표적을 분리하게 된다. 충분한 해상도가 주어진다면 이렇게 분리된 표적들은 2-D DBS 영상으로 구분이 가능하다.

그림 3과 같이 일반적인 레이더의 영상과 DBS 영상을 비교하면 DBS 영상이 일반 레이더 영상보다 선명도가 훨씬 뛰어남을 알 수 있다[5]. 이로 인해 표적 식별 및 추적 능력이 향상될 수 있다. 이점이 DBS 영상을 사용하는 주요 이유이다.

2. DBS 알고리즘 순서도

DBS 영상을 실시간 생성을 위하여 알고리즘을 단순화가 필요하다. 본 논문에서는 DBS 2-D 영상을 만들기 위해서 그림 4와 같은 단순화된 알고리즘 순서도를 구성하였다. 매 펄스마다 신호획득으로 얻은 Raw Data를 이용하여 FFT 및 부엽신호 제거를 위한 Window 과정이 포함된 도플러프로세싱, R-D맵으로 생성된 영상으로 절대좌표 X-Y맵으로 변환하는 좌표변환 과정이 포함된다.

Look time 시간동안 획득한 Raw Data는 다음 Look time 동안 모든 DBS 처리가 이루어져야 한다. 그렇지 못하다면 실시간성을 보장하지 못하여 정확한 DBS 2-D 영상을 얻지 못하게 된다. 멀티코어 구조를 이용하여 병렬적으로 도플러프로세싱 및 좌표변환을 처리하도록 한다.

3. 좌표변환 방안

그림 5과 같이 변환할 절대좌표 위치 (X_c, Y_c),

현재 위치 (X_m, Y_m), 안테나 각도(θ), 현재 속도(V_m)이라고 한다면 절대좌표 위치에서 변환되어 R-D맵 좌표 값에 해당되는 좌표 index값은 아래 식들로부터 구해질 수 있다 [6].

$$\cos\theta = \frac{\overrightarrow{U_m} \bullet \overrightarrow{U_p}}{\left|\overrightarrow{U_m}\right|\left|\overrightarrow{U_p}\right|} \quad (1)$$

$$V_P = V_m \cos\theta \quad (2)$$

$$f_i = \frac{2V_p}{\lambda}\frac{1}{D_{resolution}} \quad (3)$$

$$R_i = \sqrt[2]{(X_c - X_m)^2 + (Y_c - Y_m)^2} / R_{resolution} \quad (4)$$

여기서,

$\overrightarrow{U_m}$: 진행방향에 대한 단위벡터

$\overrightarrow{U_p}$: 절대좌표위치에 대한 단위벡터

V_p : 상대속도

R_i : R-D맵에서 거리 index

f_i : R-D맵에서 도플러 index

위 식들로 구해진 거리/도플러 index 값으로부터 절대좌표 (X_c, Y_c)에 해당되는 도플러프로세싱 결과의 R-D맵 값을 얻을 수 있다. 전체 X-Y맵의 좌표위치에 대하여 R-D맵의 값 매칭하는 것을 멀티코어에 나누어서 할당한다. 변환되는 연산수는 절대좌표 해상도에 따라 달라지는데 해상도가 높으면 변환되어야 좌표 수는 늘어나게 되고 연산량이 많아지는 반면, 해상도가 낮으면 연산량은 줄어들지만 DBS 2-D 영상이 선명하지 않아 표적 식별 능력이 떨어지게 된다. 따라서 적절한 해상도의 선택이 매우 중요하다.

Ⅲ. 본 론 (2)

1. 멀티 코어 구조 설계

도플러프로세싱은 M(거리) * N(도플러) FFT 및 부엽신호 제거를 위한 Window 과정으로 이루어진다. FFT 연산시간이 도플러프로세싱 과정의 대부분을 차지하기 때문에 연산시간 단축을 위하여 K개의 멀티코어를 이용하여 각각 코어에 M/K개의 거리방향으로 나누어 할당하여 M/K개의 N FFT 이루어지게 하였다.

좌표변환은 K개의 멀티코어에서 모두 도플러프로세싱이 끝난 후 순차적으로 진행되도록 설계하였다.

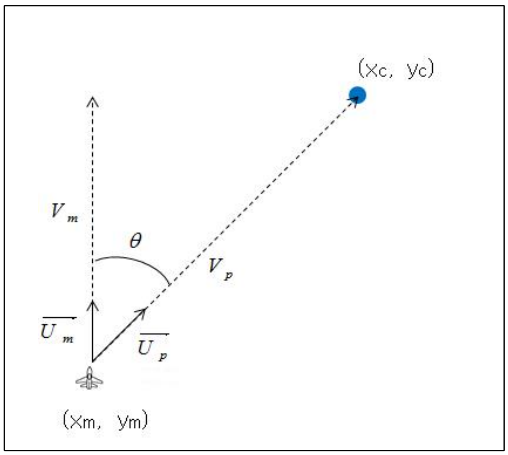

그림 5. 좌표변환 방안
Fig. 5 Coordinate Conversion Plan

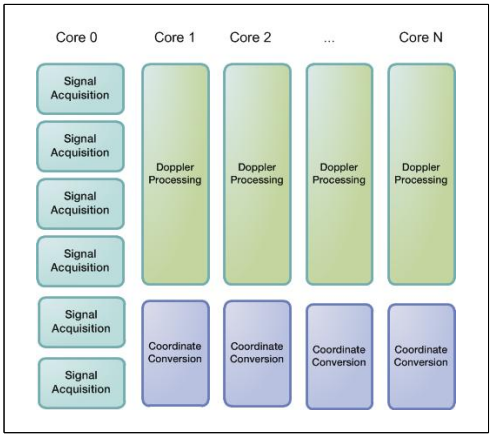

그림 6. 멀티코어 Task 할당 계획
Fig. 6 Multi-core Task assignment plan

K개의 멀티코어에서 나누어져 실행된 도플러프로세싱이 모두 끝난 후 좌표변환도 또한 L개의 멀티코어에 나누어 할당하여 실행되게 하였다. 좌표변환과정은 도플러프로세싱에서 구해진 R-D맵을 절대좌표인 X-Y맵으로 바꾸는 과정이다.

본 논문에서는 R-D맵의 모든 좌표를 X-Y맵으로 바꾸는 대신에 절대좌표인 X-Y맵의 좌표를 역계산하여 R-D맵의 값을 얻는 방식으로 진행하였다. 이렇게 함으로서 겹쳐지는 좌표 계산을 최대한 줄여 연산시간이 단축효과를 얻을 수 있다. 따라서 X-Y맵의 좌표에서 X축으로 L개로 나누어서 코어에 할당한다. 안테나 각도, 현재 위치 정보, 속도 등을 이용하여 하나의 코어에서 X/L-Y의 좌표변환 과정

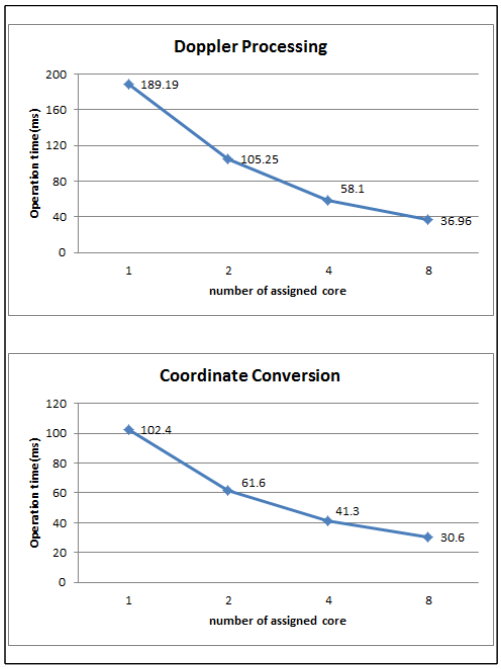

그림 7. 할당된 코어 수에 따른 연산시간
Fig. 7 Operation time by number of assigned core

이 이루어지며 된다. 좌표변환 과정은 도플러프로세싱 과정이 모두 끝난 후 동시에 시작되도록 동기화 시켰다.

그림 6과 같은 멀티코어 Task 할당 계획을 이용하여 DBS 알고리즘을 실행하였다. 신호획득 안정성을 위하여 신호획득 코어를 가장 높은 우선순위를 가지게 하였고 실험데이터 분석하기 위한 데이터 저장 기능을 하나의 코어에 할당하였다. 이 모든 알고리즘은 VxWorks 6.9로 설계되었다.

본 논문에서는 SMP 운영환경에서 이루어졌기 때문에 코어간의 동기는 세마포어를 이용하였다 [7]. 그리고 도플러프로세싱과 좌표변환 과정을 멀티코어로 나누어 할당하기 위하여 코어 간 메모리를 공유하는 방식을 선택하였다 [8, 9].

2. 시뮬레이션 결과

매 PRF마다 신호획득을 얻어진 Raw Data의 거리 256셀, 도플러 1024셀을 이용한 도플러프로세싱 및 좌표변환을 실행하였다. 실시간 구현이 가능한 각 기능별 연산시간은 표 1과 같다.

도플러 프로세싱을 수행하기 위하여 이 기능에 할당된 코어 수에 따른 연산시간은 그림 7과 같이

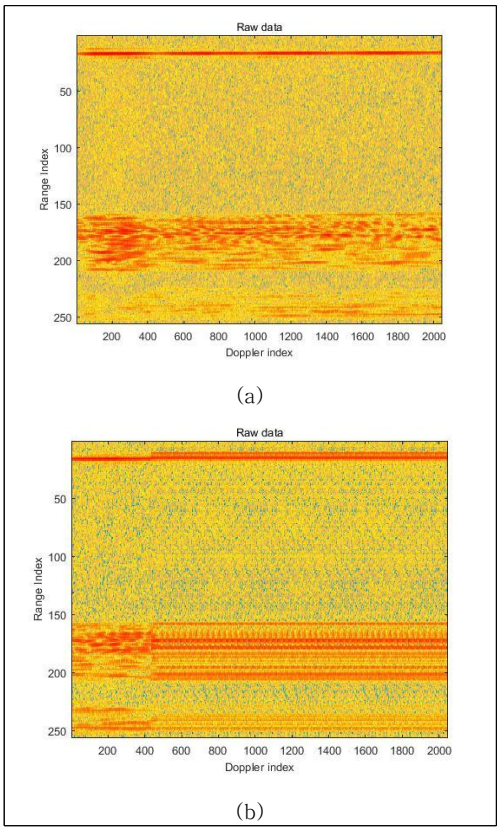

그림 8. 실시간 구현 유무에 따른 신호획득
Fig. 8 Signal acquisition in absence of a realtime implementation

표 1. 기능별 연산시간
Table 1. Operation Time of the Function

	Doppler Processing	Coordinate Conversion
P4080	98ms	34ms

나타내었다. 그림 7과 표 1에 보듯이 하나 또는 두 개의 멀티코어로 도플러 프로세싱을 수행할 경우 실시간 구현이 불가능함을 알 수 있다. 도플러 프로세싱을 실시간으로 구현하기 위해서는 최소한 4개의 멀티코어를 할당하여 수행하여야 한다. 그리고 좌표변환은 최소 8개의 멀티코어로 수행하여야 실시간 구현이 가능함을 알 수 있다. 여기서는 실시간 구현이 가능하도록 도플러 프로세싱 및 좌표변환을 8개의 멀티코어로 수행하였다.

그림 8에서는 도플러 프로세싱 및 좌표변환이 실시간 구현이 이루지 않았을 경우 신호획득이 어

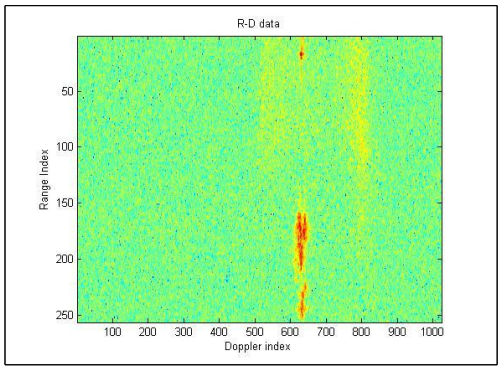

그림 9. 도플러프로세싱 결과
Fig. 9 Doppler Processing result

긋남을 보여준다. 그림 8의 (b)는 (a)와는 다르게 일정 시점부터 신호획득이 제대로 이루어지지 않고 같은 신호가 반복됨을 알 수 있다. 이것은 도플러 프로세싱 및 좌표변환이 CPI(Coherent Process Interval)동안에 이루어지 못하여 다음 신호 획득 시에도 이 기능을 담당하는 코어가 수행되어 신호 획득 코어는 한동안 pending상태가 되어 신호획득이 제대로 이루어지지 않았기 때문이다. 신호획득, 도플러프로세싱 및 좌표변환이 제대로 수행되기 위해서는 적절한 멀티코어가 할당되어 실시간성이 구현되어야한다.

8개의 멀티코어를 할당하여 실시간이 구현이 가능하였을 때 신호획득 데이터를 이용하여 수행된 도플러프로세싱 시뮬레이션 결과는 그림 9와 같다. 본 논문에서는 부엽신호 제거를 위하여 chebyshev 윈도우를 사용하였다. 도플러프로세싱 과정에서 얻은 R-D맵을 절대좌표 변환하기 전에 적절한 Threshold를 취해주어야 하며, 40dB를 Threshold 값으로 하여 수행되었다.

목표 지점을 중심으로 한 6km(X축)*4km(Y축) 거리를 좌표변환 수행한다. 거리해상도에 따라 연산시간 및 DBS 영상이 달라진다. 본 논문에서는 그림 9의 도플러프로세싱 결과를 현재 위치 정보, 안테나 각도 및 속도 등을 이용하여 10m/20m/100m 거리 해상도에 따른 좌표변환 결과를 그림 10에 나타내었다. 그림 10에 보듯이 거리해상도가 커질수록 DBS 영상의 선명도는 떨어짐을 알 수 있다. 대신 연산량은 줄어드는데 사용되는 시스템에 따라서 표적 식별이 가능한 해상도 선택이 중요함을 알 수 있다.

그리고 한번 Look time으로 얻을 수 있는 DBS 영상의 영역은 안테나 빔 폭에 의하여 제한된다.

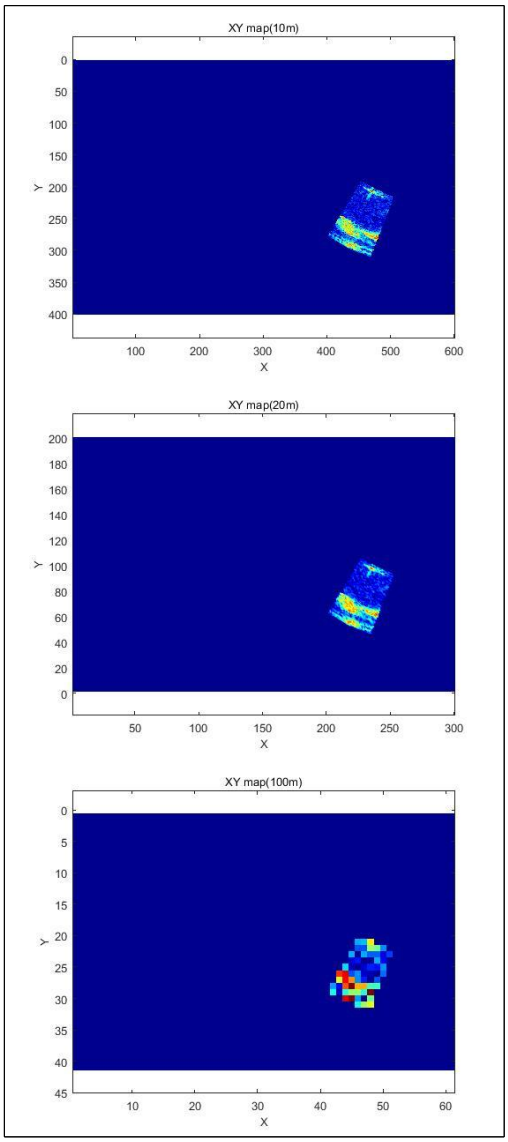

그림 10. 좌표변환 결과
Fig. 10 Coordinate Conversion Result

절대좌표 모든 영역을 채우기 위해서는 많은 Look time이 필요하다. 그렇기 위해서는 다음 Look time 동안 안테나는 이동해야 하며, 속도에 의한 거리 보상도 필요하다.

그림 11은 여러 Look time동안 좌표변환 결과를 누적한 영상이다. 누적된 영상은 한 번 Look time에서 얻은 영상보다 더 넓은 영상을 보여주고 있으며, 또한 겹쳐지는 부분의 평균화로 인하여 영상이 더 선명해짐을 알 수 있다.

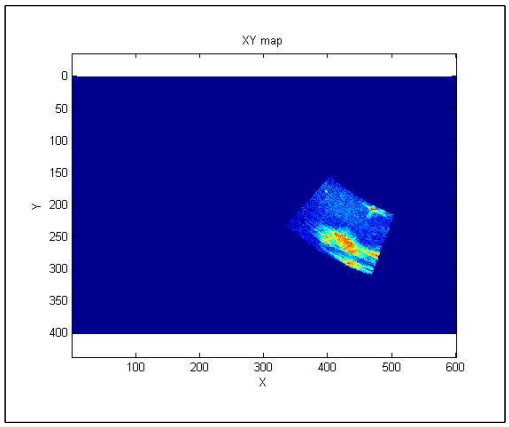

그림 11. 누적된 DBS 영상
Fig. 11 Accumulated DBS image

Ⅳ. 결 론

본 논문은 DBS 알고리즘의 중요기능인 도플러 프로세싱 및 좌표변환이 실시간으로 수행되기 위하여 멀티코어 커널을 설계하는 방법과 DBS 알고리즘 수행되는 과정을 보였다. 최대한 연산시간을 줄여 실시간성을 보장하기 위하여 복잡한 DBS 알고리즘을 단순화시켰고 메모리 공유 및 세마포어 동기화를 통하여 멀티코어 구조를 설계하였다. 실시간 구현이 가능한 코어 수를 확인하여 해당 기능에 맞는 코어 할당을 하였다. 그리고 마지막으로 실제로 P4080보드로 구현된 DBS 알고리즘을 시뮬레이션하여 결과를 나타내었다.

임베디드 시스템 환경에서 DBS 알고리즘을 실시간 구현을 위해서는 많은 연산시간이 소요되는 도플러프로세싱 및 좌표변환의 연산시간을 단축시켜야 한다. 여기서 우리는 연산시간을 단축을 위하여 도플러프로세싱 및 좌표변환 기능을 8개의 멀티코어에 나누어서 할당하여 동시에 수행하게 하였다. 그리고 좌표변환 연산량을 줄이기 위하여 좌표 역계산법을 적용하여 실시간으로 DBS 알고리즘을 구현이 가능하도록 하였다.

사용되는 시스템 환경 및 운용에 따라 멀티코어 설계 방법은 달라지며, 이에 따라 실시간 처리가 가능케 하는 알고리즘 방안도 달라질 것이다. 따라서 실제 사용하는 환경에 맞는 알고리즘 순서도 및 멀티코어 구조 설계가 필요할 것이다.

향후에는 실제 영상과 DBS 알고리즘으로 얻은 영상은 계산오차로 인하여 차이를 보이므로 영상을 보정하는 방안을 개발해 나가야 할 것으로 보인다. 추후 DBS 영상을 보정하여 실제 영상과 비교하는 방안을 설계할 예정이다.

References

[1] J. Liu, M. Yang, “Task scheduling of real-time systems on multi-core embedded processor,” Proceedings of International Conference on Intelligent Systems and Knowledge Engineering, pp. 580-583, 2010.

[2] G.W. Stimson, Introduction to Airborne RADAR, 2nd edition, 1998.

[3] K. Kim, S. Kim, J. Ui, “Detection of ship targets near coastline by using Doppler Beam Sharpening technique,” Proceedings of 3rd International Asia-Pacific Conference on Synthetic Aperture Radar, pp. 1-4, 2011.

[4] E. Byron, Radar : principles, technology, applications, Prentice-hall, 1993.

[5] M.A. Richards, Principles of Pulse-Doppler RADAR, 2011.

[6] W. Sun, A. Chen, C. Zhang, “Range-Doppler Approach for Calibration and Location of Air-borne SAR Image,” Proceedings of CIE International Conference on Radar, pp. 16-19, 2006.

[7] WindRiver VxWorks Platforms 6.9 documentation, http://www.windriver.com

[8] M. Gschwind, “Synergistic Processing in Cell's Multicore Architecture,” IEEE Micro. Vol. 26, No. 2, pp. 10-24, 2006.

[9] J.M. Calandrino, J. Anderson, “On the design and implementation of a cache-aware multicore real-time scheduler,” Proceedings of the 21st Euromicro Conference on Real-Time Systems, pp. 194-204, 2009.

Young-Joo Kong (공 영 주)

He received his B.S degree in Electronic Engineering from Sung-kyonkwan University, in 2009. He is currently working as a researcher for LIG Nex1. His Current research interests includes embedded Software, Radar signal processing.

Email: yjkong16@lignex1.com

Seon-Keol Woo (우 선 걸)

He received his Ph.D degree in Electric Wave Engineering from Kwang-woon University, in 2007. He is currently working as a researcher for LIG Nex1. His Current research interests includes embedded Software, Radar signal processing.

Email: wooseonkeol@lignex1.com