İşlem Yükü Yüksek Sinyal İşleme Algoritmalarının İşlemci (CPU) ve Grafik İşlemci (GPU) Üzerinde Paylaşılarak Gerçeklenmesi

Sevda Erdoğdu[1], Burcu Sağel Tawk[1], Erdal Mehmetcik[1]

[1]Gömülü ve Gerçek Zamanlı Yazılım Tasarım Müdürlüğü, SST Sektör Bşk.
ASELSAN A.Ş.

{erdogdu, bsagel, emehmetcik}@aselsan.com.tr

Özet. Birçok algılayıcı içeren sistemlerde, sistemin amacının gerçekleştirilebilmesi için, bu algılayıcılardan elde edilen büyük boyutlardaki veriler üzerinde işlem yükü yüksek sinyal işleme algoritmalarını kabul edilebilir bir performans sağlayacak şekilde gerçek zamanlı olarak uygulamak gerekmektedir. Geliştirilen sinyal işleme algoritmalarının paralel ya da ardışık işlenebilir yapıda olmasına göre, grafik işlemci (GPU) ya da işlemci (CPU) gibi daha uygun olan donanımlar üzerinde işlenmesi, işlem süresinin azaltılmasını sağlayacaktır. Bu makale kapsamında, algılayıcı olarak birçok hidrofon kullanan, çevredeki olası tehditleri tespit etmeyi amaçlayan bir sonar sisteminin sinyal işleme algoritma dizisinin, işlemci (CPU) ve grafik işlemci (GPU) üzerinde paylaşılarak ve bu yapılara özel kütüphaneler kullanılarak gerçeklenmesi anlatılacaktır.

Anahtar Kelimeler: Sinyal İşleme Algoritmaları Gerçekleme, GPU, CPU

Abstract. For systems including multi sensors, it is necessary to apply complex signal processing algorithms in real time with an acceptable performance to achieve the system goals. Depending on the nature of the algorithms to be implemented (suitable for parallel or sequential processing), implementation of the algorithms on the appropriate hardware, for instance graphical processing units (GPU) or central processing units (CPU), would decrease the computation time. In this paper, implementation of a sonar system (uses multi hydrophones to detect possible threats) signal processing algorithm sequence by distributing algorithms on CPU and GPU and using libraries specific to these architectures will be explained.

Keywords: Implementation Of Signal Processing Algorithms, GPU, CPU

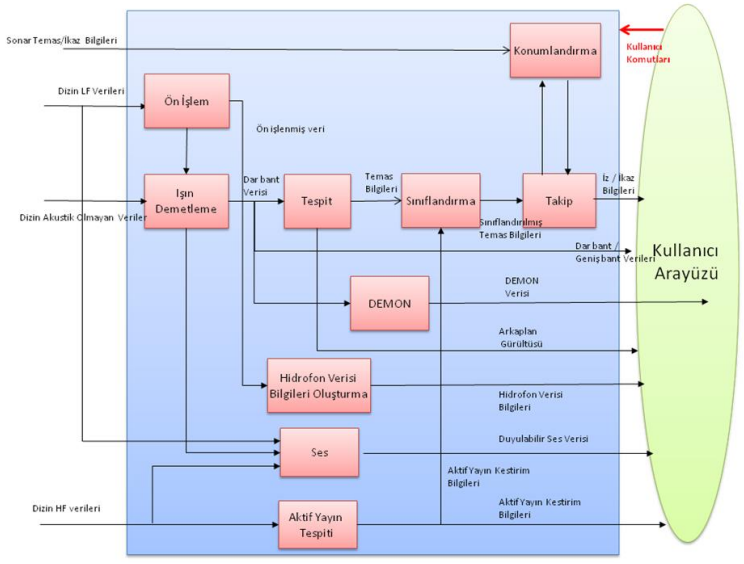
1 Giriş

Birçok algılayıcı içeren sistemlerde, sistemin amacının gerçekleştirilebilmesi için, bu algılayıcılardan elde edilen büyük boyutlardaki veriler üzerinde işlem yükü yüksek sinyal işleme algoritmalarını kabul edilebilir bir performans sağlayacak şekilde gerçek zamanlı olarak uygulamak gerekmektedir. İşlem yükü yüksek sinyal işleme algoritmalarının (görüntü işleme, sonar sinyal işleme, vs.) gerçeklenmesi için güçlü donanımlara (grafik işlemci vs.) ya da işlem yükünün (algoritmaların) farklı donanımlara dağıtılmasına ihtiyaç olduğu görülmüştür.

Bu makale kapsamında, algılayıcı olarak birçok hidrofon kullanan örnek bir pasif sonar sisteminin sinyal işleme algoritma dizisinin, işlemci (CPU) ve grafik işlemci (GPU) üzerinde paylaşılarak ve bu yapılara özel kütüphaneler kullanılarak gerçeklenmesi ve gerçeklenme sırasında yapılan adımlar anlatılacaktır. Bir algoritmanın gerçekleneceği işlemci seçiminde izlediğimiz yoldan da bahsedilecektir.

2 Pasif Sonar Sistemi

Bu makalede anlatılan örnek pasif sonar sistemi, düşük frekanslı hedef özgürültüsü ve DEMON (“Detection Of Envelope Modulation On Noise”) bilgisi ile her türlü sualtı hedefini tespit etme, sınıflandırma, konumlandırma, takip etme kabiliyetlerine sahip bir sistemdir. Aynı zamanda hedefin yüksek frekanslı aktif yayınını tespit ederek kullanıcıya sınıflandırma bilgisi olarak da sunmaktadır.



Şekil 1 Sinyal İşleme Algoritma Akışı

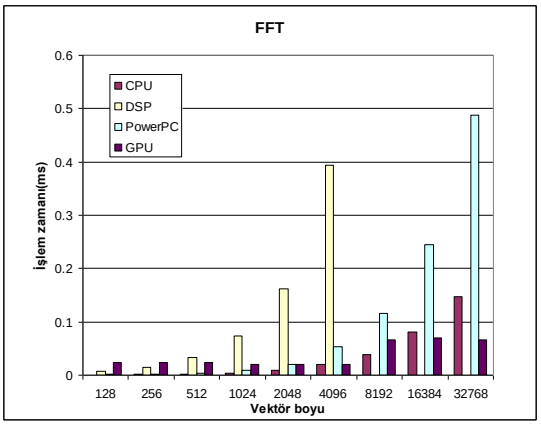
Sistemde düşük frekanslardaki hedef öz gürültüsünün tespiti için doğrusal dizin yapısında bir hidrofon dizini bulunmaktadır. Bu dizin triplet adı verilen sensör istasyonlarından oluşmaktadır. Bunun dışında dizinde hareket esnasında meydana gelen ve sinyal işlemeyi etkileyebilecek geometrik değişiklikleri algılayabilmek için akustik olmayan sensörler ve yüksek frekans bandında hedef aktif yayınlarının dinlenmesi için tüm yönlü yayın (intersept) algılayıcıları yer almaktadır.

Tüm hidrofonlardan ve akustik olmayan sensörlerden gelen veri, ışın demetleme algoritmaları tarafından işlenir ve işlenmiş veri tespit, sınıflandırma, takip, konumlandırma algoritmalarına aktarılır. Sistemin sinyal işleme algoritma akışı ve algoritmaların birbirleri ile olan ilişkileri Şekil 1'de verilmiştir.

3 Sinyal İşleme Algoritmaları Gerçekleme

3.1 Donanım Seçimi

Sistemde düşük frekans için yaklaşık 7.5MB, yüksek frekans için yaklaşık 1 MB'lık verinin 500 ms içerisinde işlenmesi gerekmektedir. Bu büyüklükte bir verinin sinyal işleme alanında sıkça kullanılan Power PC, DSP gibi tek bir birim üzerinde gerçeklenemeyeceği yapılan testler ile görülmüştür.



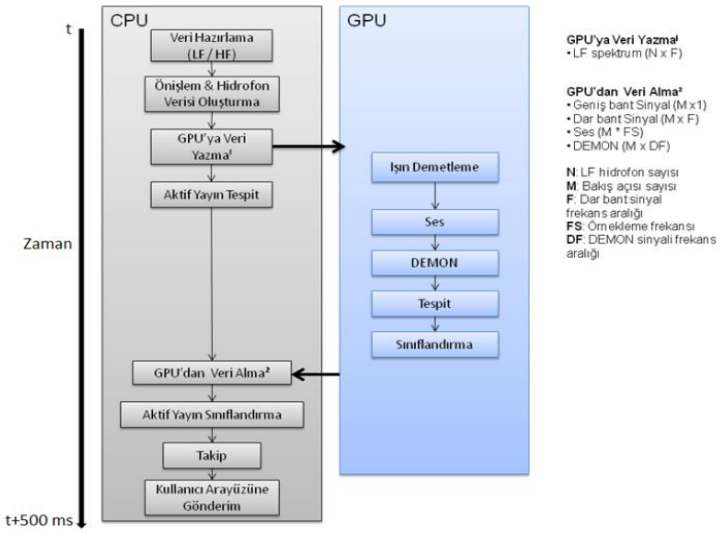
Şekil 2 FFT işlemi zamanlama değerleri

Bu nedenle ilk olarak görüntüleme amaçlı kullanılmak için tasarlanmış ama sonradan çok çekirdekli (100 çekirdek üzeri) büyük verilerle tek, basit ve tekrarlı algoritmaları gerçeklemek için kullanılır hale gelmiş GPU'lar[1] üzerinde çalıştırılarak denenmesinin uygun olacağı düşünülmüştür. Sinyal işleme algoritmalarında kullanılan temel işlemler (çıkarma, çarpma, FFT gibi) farklı büyüklükteki veriler için bir Power PC, CPU, DSP ve GPU üzerinde çalıştırılarak, çalışma süreleri karşılaştırılmıştır.

Şekil 2'de görüldüğü gibi GPU özellikle büyük verilerde en kısa sürede çıktı üreten donanımdır. Bu nedenle sistemde büyük verileri paralel olarak işlemeye olanak veren sinyal işleme algoritmalarının gerçeklenmesinde GPU kullanımına karar verilmiştir.

3.2 Algoritma Gerçekleme

GPU'lar yardımcı işlemci (co processor) olarak kullanılmaktadırlar, tek başlarına çalışamazlar. GPU'lara erişim ve kullanım CPU üzerinden olmaktadır. CPU, GPU üzerinde algoritmaların akışını gerçeklemek için kullanılmaktadır. Ayrıca tüm algoritmaların GPU üzerinde gerçeklemeye uygun olmadığı görülmüştür bu nedenle bazı algoritmalar CPU üzerinde gerçeklenmektedir. GPU üzerinde gerçeklenecek algoritmalara, algoritmanın yapısına bakılarak karar verilmektedir. Algoritma büyük veriler üzerinde ve paralel çalışabilir fonksiyonlardan oluşuyor ise GPU üzerinde, aksi durumda CPU üzerinde gerçeklenmektedir. Algoritmalar tarafından kullanılan ve akış esnasında değişmeyen tüm parametreler GPU hafızasına ilkleme esnasında yazılır ve hep burada tutulur.



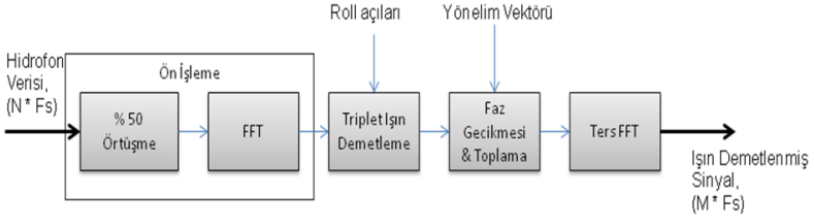
Şekil 3 Algoritma akışı ve CPU – GPU paylaşımı

CPU-GPU kullanım kararı için ilk olarak algoritmaların yapısı incelenmiştir. **Error! Reference source not found.**'te Pasif Sonar sisteminin sinyal işleme algoritma akışı ve CPU-GPU paylaşımı verilmiştir. Kullanılan algoritmalar ve neden CPU ya da GPU'da gerçeklendikleri aşağıda kısaca verilmiştir.

Ön İşleme: Ön işleme ile hidrofonlardan alınan veriler önceki verilerle %50 örtüştürülürek FFT'si alınır. Ön işleme, ön işleme çıktısı sinyalin hidrofon verileri

bilgilerinin (RMS, maksimum değer vs.) oluşturulması için de kullanıldığından CPU üzerinde yapılmaktadır.

Işın demetleme: Sistemde kullanılan pasif sonar dizini, triplet yapıya sahip algılayıcı (hidrofon) istasyonlarından oluşan doğrusal dizindir. Triplet yapı, iskele-sancak belirsizliğini anlık olarak çözmek için kullanılan bir yapıdır. Triplet sinyal işleme [2] sonucu elde edilen sinyal, dizin ışın demetleme algoritmasına girdi olmaktadır (Şekil 4Şekil *1*).



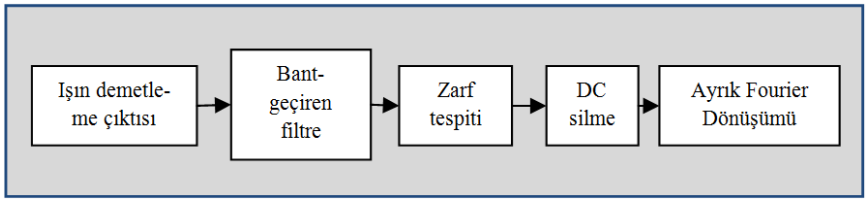
Şekil 4 Işın Demetleme Blok Şeması

Konvansiyonel ışın demetleme işlemi zaman veya frekans uzamında yapılabilir. Bu çalışma kapsamında frekans uzamında ışın demetleme yapılmaktadır. Konvansiyonel ışın demetleme olarak ifade edilen yöntem, hidrofonlardan alınan veriye gecikme uygulayarak toplama işlemine karşılık gelmektedir. Yapılan işlem her bir hidrofondan gelen sinyalin FFT'si alındıktan sonra, yönelim açısına göre uygun ağırlıklar ve fazlarla (gerçek zamanla değişen yönelim vektörü ile) çarpılıp toplanmasına karşılık gelmektedir.

Işın demetleme algoritması hem işlem yükü yoğun hem de paralel çalıştırılabilecek bir algoritmadır. GPU üzerinde çalıştırılmasının performans açısından daha verimli olacağı öngörülmüştür ancak; CPU'nun da performansını gözlemlemek ve kararın doğruluğunu teyit etmek amacıyla Intel IPP/MKL kütüphaneleri kullanılarak da gerçeklenmiştir. Karşılaştırmalar sonucunda GPU(Tesla K40c) üzerindeki performansa erişebilmek için 21 tane Intel E5 işlemcili kart üzerinde algoritmanın koşması gerekmektedir.

Oluşturulan ışın demetleme çıktısı frekans uzamındadır. Bu sinyal ters FFT alınarak zaman uzamına çevrilir ve NxFs boyutundaki ışın demetleme çıktısı sinyal oluşturulur. Ters FFT alma işlemi, hem bu işleme girdi olan sinyal GPU üzerinde olduğu için, hem de çıktısı işlem yükü yoğun DEMON algoritmasında kullanıldığı için yine GPU üzerinde yapılır.

DEMON: Hedef platform pervaneleri kavitasyona uğradığında geniş bantlı gürültü üretirler. Bu gürültü, pervanenin kanatçıkları tarafından genlik modülasyonuna uğrar ve pervanenin dönüş hızı ve kanatçık sayısı gibi bilgiler kavitasyon gürültüsünün düzgün bir şekilde demodüle edilmesiyle elde edilebilir [3].



Şekil 5 DEMON algoritması blok şeması

DEMON algoritması GPU üzerinde gerçeklenmektedir.

Tespit: Tespit algoritması, hedef tespit dizininden gelen zaman tabanlı akustik veriden ışın demetleme sonucunda oluşturulan verileri kullanarak olası bir hedefin tespitini gerçekleştirir. Sistem düşük frekanslar için, güç tabanlı geniş bant ve güç tabanlı dar bant tespit algoritmaları koşturmaktadır.

Geniş bant tespit algoritması, her bir açıdan gelen güç bilgisini kullanarak hedef bulunması olası açılarla, hedef bulunması olasılığı olmayan açıları ayırarak tespit sonucunu üretir. Bu işlemde arka plan gürültüsünün seviyesi kestirilmekte, daha sonra da hedefin bulunması olası açılar için eşik seviyesi değerleri "sabit yanlış alarm sıklığı" (CFAR, [5], [5]) yöntemi hesaplanmaktadır. Geniş bant tespit algoritması ortam gürültüsünü kestirebilen ve değişen ortam gürültüsü karşısında tespit eşik değerini otomatik olarak güncelleyebilen bir yapıya sahiptir. Dar bant tespit algoritması, geniş bant tespit algoritması ile oldukça benzerdir. Geniş bant algoritmasından farklı olarak, her bantta bağımsız bir tespit işlemi yapılır. Her bir banttan gelen ham tespit sonuçları (geniş ve dar bant) birleştirilerek, nihai tespit çıktısı oluşturulur.

Sınıflandırma: Pasif sonar sisteminde, sınıflandırma algoritmasında kullanılan özellikler, tespit bloğundan gelen tonal frekans bilgisi (dar bant tespitler) ile aktif yayın kestirim bloğundan gelen bilgilerdir.

Sınıflandırma algoritmasında, dar bant tespit bloğundan alınan hedef tonal bilgilerinin kütüphanede yer alan hedef özellikleriyle karşılaştırılması gerekmektedir. Bu işlem, her tespit kararı için, kütüphanedeki her hedef sınıfıyla bir korelasyon hesaplamasına ihtiyaç duyar. K adet tespit kararı için kütüphanede bulunan L adet sınıfla, toplamda KxL adet korelasyon hesaplaması gerekmektedir. Bu korelasyon hesaplamaları, birbirlerinden bağımsız oldukları için (herhangi birinin sonucu diğerinin hesaplaması sırasında kullanılmadığından) GPU üzerinde hesaplanmalarının daha uygundur ve bu nedenle sınıflandırma GPU üzerinde yapılmaktadır.

Aktif yayın tespitlerinin sınıflandırılması "Bayes" karar verme kuralına göre yapılmaktadır, burada kullanılan veri daha küçük olduğundan bu işlem CPU üzerinde yapılmaktadır.

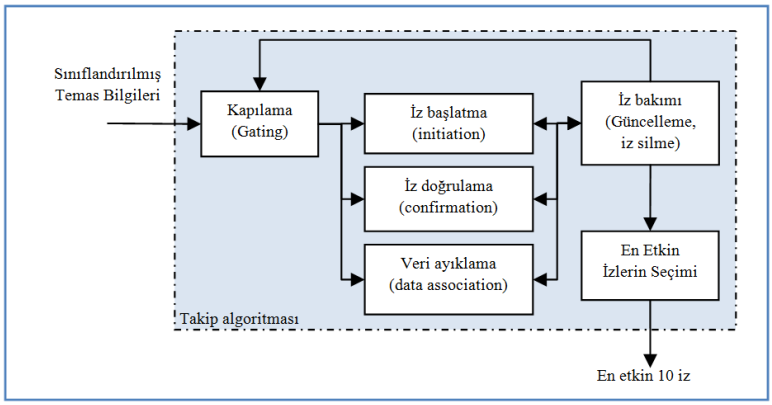
Takip: Takip algoritması, sınıflandırılmış temas (tespit) çıktılarını kullanarak bir veya birden fazla hedefin kerteriz bilgilerini çıkartmaktadır. Takip algoritması en etkin 10 hedefi izlemek üzere tasarlanmıştır. En "etkin" hedeflerin belirlenmesi, takip edilen hedeflerden sınıflandırma olanların önceliklendirilmesi ile yapılmaktadır. Bu

şekilde, tehdit potansiyeli daha yüksek ya da kullanıcı tarafından "önemli" olarak işaretlenen hedeflerin takibine öncelik verilmektedir.

Takip algoritmasının durum vektörü kerteriz ve kerteriz değişiminden oluşmaktadır ve sabit hız süreç modeli kullanılmaktadır. Veri ayıklama yöntemi olarak da "Global Nearest Neighborhood, (GNN)" seçilmiştir[6], [7].

Algoritmanın akış diyagramı Şekil 6'da gösterilmiştir. Her güncelleme sonrasında var olan izler kontrol edilerek, belli bir süre bir tespit kararıyla güncellenmeyen izler silinmektedir.

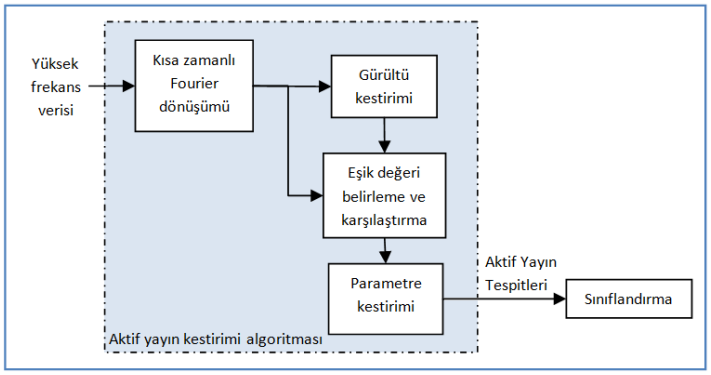
Takip algoritması sınıflandırılmış temas bilgilerini kullanmaktadır. Sınıflandırılmış temas bilgileri, temel olarak kerteriz bilgisi ile hedefin sınıflandırma bilgilerinden(veri tabanındaki bir tehdide benzeme oranı, tehdit bilgileri vs.) oluşur, bu veriler hidrofon verileri gibi büyük olmadıklarından CPU üzerinde gerçeklenmektedir.



Şekil 6 Takip Algoritması Blok Şeması

Aktif Yayın Tespit: Aktif yayın tespit algoritması, aktif yayın yapan platformların yayınlarını tespit etmekte ve bu yayınların parametrelerinin kestirimini yapmaktadır (Şekil 7). Yüksek frekans bandında alış yapabilen hidrofonlardan gelen yüksek frekanslı zaman tabanlı akustik veri, aktif yayın tespit algoritmasından geçirilerek, yayının tespiti ve yayın parametrelerinin çıkartılması sağlanmaktadır. Literatürde bu tarz algoritmalar genellikle enerji dedektörü (radyometre, [8]) ve sabit yanlış alarm olasılığı (CFAR, [4]) tabanlıdır. Sistemde bu yöntemlerin kombinasyonu kullanılmaktadır.

Aktif Yayın tespit algoritması tüm yönlü yüksek frekans hidrofonundan alınan verileri kullanılır. Bu veriler de düşük frekans verileri gibi büyük verilerdir fakat algoritma tüm veriyi tek seferde işlemez, küçük bloklar halinde işler. Bu nedenle bu algoritma CPU üzerinde gerçeklenmesi daha uygun bulunmuştur.



Şekil 7 Aktif yayın tespit algoritması blok şeması

Yüksek frekans hidrofonu ses verisinin oluşturulması da yine CPU üzerinde yapılmaktadır. Bu işlem yüksek frekanstaki verilerin duyulabilir ses frekansına çekilmesi ile gerçeklenir.

Kullanıcı arayüzüne gönderilerek ekranlarda gösterilecek tüm sinyallerin (geniş bant, dar bant, DEMON) dB (Decibel) skalasında olması gerekmektedir. Bu çevrim veri gönderimi esnasında CPU üzerinde yapılmaktadır.

Veri transferi, (hem GPU'ya veri aktarma hem de GPU'dan veri okuma) zaman alan işlemlerdir, bu nedenle algoritmaların gerçeklenmesi esnasında mümkün olduğunca az veri transferi yapılmasına dikkat edilmektedir.

Güncel algoritmaların gerçekleme zamanları Tablo 1'de verilmiştir. Algoritmaların bir kısmı önce CPU üzerinde gerçeklenmiş, algoritma gerçekleme zamanı beklenenden uzun alındığında GPU üzerine taşınmıştır. Örneğin Işın demetleme algoritması Intel IPP/MKL kütüphaneleri kullanılarak gerçeklendiğinde, GPU ile gerçeklediğimiz sürenin 21 katı civarında bir sürede ancak işlenebilmiştir.

Tablo 1 Normalize Algoritma Gerçekleme Zamanları

Algoritma	**Gerçeklendiği Yer**	**Normalize İşlem Süresi([1])**
Ön işleme	CPU	0,063258
Işın Demetleme	GPU	0,308558
DEMON	GPU	0,122506
Tespit	GPU	0,06996
Sınıflandırma	GPU	0,02963
Takip	CPU	0,00411
Aktif Yayın Tespit	CPU	0,04616
Toplam Süre	CPU + GPU	0,644182
Işın Demetleme[2]	CPU	8,00000

[1] Algoritma işlem süresinin algoritmanın gerçeklenmesi için ayrılan toplam zamana oranı

[2] IPP/MKL ile gerçeklenmiş optimize edilebilir algoritma

4 Kütüphaneler

Sistemde Intel tabanlı CPU'lar tercih edilmiş ve Intel CPU'lara özgü olan IPP (Intel Integrated Performance Primitives) ve MKL (Math Kernel Library) kütüphaneleri kullanılmıştır. Bu kütüphanelerin kullanımıyla, çok işlemcili çok çekirdekli bilgisayarlarda yapılan işlemler, işlemciler ve çekirdekler üzerinde otomatik olarak dağıtılmakta ve bu yapıların en etkin kullanımı sağlanmaktadır. Torpido tespit sistemi sinyal işleme bilgisayarı olarak çok işlemcili, çok çekirdekli güçlü bir bilgisayar tercih edilmiştir.

GPU ile çalışırken CUDA (Computer Unified Device Architecture) kütüphaneleri kullanılmış, temel CUDA kütüphanesi fonksiyonları bir araya getirilip uygun şekillerde paketlenerek CUDA Sinyal İşleme Kütüphanesi (CudaSPL) geliştirilmiş ve algoritmaların gerçeklenmesinde kullanılmıştır.

5 Sonuç ve Yapılabilecek İyileştirmeler

Pasif sonar sisteminin sinyal işleme yazılımının CPU-GPU paylaşımı ile gerçeklenmesi ile işlem yükü yoğun algoritmaların tek bir bilgisayar üzerinde ve tek bir uygulama (executable) ile gerçeklenmesi sağlanmıştır. CPU-GPU kullanımı ile algoritmik olmayan sistem fonksiyonları Intel CPU(E5, I7 v.b.) üzerinde gerçeklenirken, algoritmaların çoğu GPU üzerinde sistemin ihtiyacını sağlayacak şekilde koşturulabilmiştir. Sistemde kullanılan sinyal işleme algoritmaları, çalışmaların başladığı günden bu yana sistem performansının iyileştirilmesi için sürekli güncellenmiştir. Özellikle geliştirilen CUDA Kütüphanesi ile algoritmalardaki güncellemeler kolaylıkla algoritma gerçekleme yazılımına yansıtılabilmiştir.

Sistem performansını artırabilmek amacıyla kullanılabilecek iki yöntem vardır. Bu yöntemlerden ilki olan CPU ve GPU'ların beraber kullanımı ile GPU ve CPU üzerinde heterojen olarak algoritma parçacığı koşturmak mümkündür. Bu yöntemde asenkron CUDA fonksiyonları, fonksiyon çağrıldıktan sonra CPU'ya kontrolü geri vermekte ve böylece CPU üzerinde veri işlenebilmekte ve bundan GPU etkilenmemektedir. Bu şekilde kullanılabilen kernel çağrıları, asenkron bellek kopyalama ya da 64KB'dan küçük belleğe veri kopyalama (CPU'dan GPU'ya), bellek set etme v.b fonksiyonlardır. Heterojen veri işlemede GPU'da veri işlenirken CPU'nun da veri işlemeye en hızlı şekilde dönmesine olanak verecek şekilde kodlama yapılmalıdır.

İkinci yöntem çoklu GPU kullanımıdır. Bu yöntemde çoklu GPU'lar aynı kodu ya da farklı kodu paralel işleyecek şekilde kullanılabilmektedir. GPU'lar arasında iletişim için yeni versiyon GPU (Fermi v.b.)'larda ve güncel CUDA versiyonlarında (5.0 ve üzeri) birbirlerinin belleklerine erişim izni verilerek CPU üzerine veriyi

taşımadan birbirlerinin belleklerindeki veriyi kullanabilmektedirler. Çoklu GPU kullanımı için farklı teknikler ve altyapılar kullanılması gerekmektedir.

Önümüzdeki dönemde bu yöntemler üzerinde çalışma yapılması planlanmaktadır.

Kaynaklar

[1] B. Sagel, S. Ceylan, M. C. Akpulat, H. Türken, S. Erdogdu, "CUDA Altyapısı Kullanılarak Sinyal İşleme Yazılımı Optimizasyonu", UYMS 2011

[2] A. Baldacci, G. Haralabus, M. Van Velzen, "Cardioid Receive Array Calibration for Active Systems", UDT Europe, 2006.

[3] S. R. Silva, "Advances in Sonar Technology", In-Teh 2009.

[4] W. A. Struzinski, E. D. Lowe, "A performance comparison of four noise background normalization schemes proposed for signal detection sytems", Journal of The Acoustical Society of America, 1984.

[5] H. L. Van Trees, "Detection, Estimation and Modulation Theory, Part-1 Detection, Estimation and Linear Modulation Theory", John Wiley and Sons Inc. 2001.

[6] S. Blackman and R. Popoli, Design and Analysis of Modern Tracking Systems. Norwood, MA:Artech House, 1999.

[7] Y. Bar-Shalom and X. R. Li, Multitarget-Multisensor Tracking: Principles, Techniques. Storrs, CT: YBS Publishing, 1995.

[8] H. Urkowitz, "Energy detection of unknown deterministic signals," Proc. IEEE, vol. 55, pp. 523-531, April 1967