ANKARA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

PROGRAMLANABİLİR DONANIM ÜZERİNDE SIKIŞTIRICI ALGILAMA İLE GÖRÜNTÜ YENİDEN OLUŞTURMA ALGORİTMASININ TASARIMI

KORAY KARAKUŞ

ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA

2011

ÖZET

Yüksek Lisans Tezi

PROGRAMLANABİLİR DONANIM ÜZERİNDE SIKIŞTIRICI ALGILAMA İLE GÖRÜNTÜ YENİDEN OLUŞTURMA ALGORİTMASI TASARIMI

Koray KARAKUŞ

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Hakkı Alparslan ILGIN

Sıkıştırıcı Algılama (SA) yöntemi bir sinyal vektörünün kendi boyutundan çok daha az sayıda doğrusal ölçümlerle yeniden oluşturulabileceğini öne süren, gelişmekte olan bir tekniktir. Seyrek sinyaller algılama matrisleri yardımı ile vektörler içerisine toplanır. Eğer sinyaller yeteri kadar seyrek ise orjinal sinyal başarılı bir şekilde yeniden oluşturulabilir. Görüntü verileri genelde seyrek veya seyrek hale getirilebilir (bulunduğu bölgeden başka bir bölgeye geçirilerek) sinyallerdir. Dolayısıyla bu veriler SA konusu için en uygun örneklerdendir.

SA uygulamalarında sinyal algılama kısmı basit yöntemlerle gerçekleştirilebilirken, eksik veri setinden sinyalleri yeniden oluşturma kısmında çok yüksek işlem gücü ve karmaşık istatistiksel hesaplamalar gerekmektedir. Yeniden oluşturma algoritmaları arasında donanımda gerçeklenmeye en uygun ve diğer algoritmalara göre daha hızlı bir yöntem olan OMP (Orthogonal Matching Pursuit) kullanılmıştır. Yöntem, donanım üzerinde gerçekleştirilebilmesi için ana yapısı bozulmadan yeniden formülize edilmiştir.

Bu çalışmada, OMP algoritması Virtex-6 tipi bir FPGA (Field Programmable Gate Array) üzerinde gerçeklenmiştir. Çeşitli iyileştirmelerle tasarlanan sistemde CPU ve GPU uygulamalarına göre en az bin kat hızlanma sağlanmıştır.

Ekim 2011, 71 sayfa

Anahtar Kelimeler: Sıkıştırıcı algılama, FPGA, Görüntü yeniden oluşturma, Donanımsal hızlandırma, Paralel işleme

ABSTRACT

Master Thesis

DESIGN OF IMAGE RECONSTRUCTION ALGORITHM USING COMPRESSIVE SAMPLING IN RECONFIGURABLE HARDWARE

Koray KARAKUŞ

Ankara University
Graduate School of Natural and Applied Sciences
Department of Electronics Engineering

Supervisor: Asst. Prof. Dr. Hakkı Alparslan ILGIN

Compressive Sampling (CS) is an emerging technique that suggests the possibility of reconstruction of a signal vector using linear measurements in much smaller numbers than its dimension. Sparse signals are acquired in vectors using sensing matrices. İf the signals are sparse enough the original signal can be reconstructed successfully. Image data are generally sparse or can be made sparse by translating to another domain. Therefore image data are one of the most feasible applications of CS.

In CS applications while the signal can be acquired using basic methods, high processing power and complex statistical computations are required in reconstructing the signal using incomplete data sets. This research focuses on OMP (Orthogonal Matching Pursuit) which is a faster and more hardware-implementable reconstruction algorithm among other methods.Method is re-formulazed without breaking its main structure, in order to implement on the hardware.

In this research OMP algorithm is implemented on a Virtex-6 type FPGA (Field Programmable Gate Array). The system designed with various optimizations yielded at least thousand times faster results than CPU and GPU applications.

October 2011, 71 pages

Key Words: Compressive sampling, FPGA, Image reconstruction, Hardware acceleration, Parallel processing

TEŞEKKÜR

Tez konumun belirlenmesinden bu güne çalışmalarımın her safhasında değerli yardım ve katkıları ile beni yönlendiren, iş hayatımdaki yoğunluk sebebiyle çalışmalarımda oluşan aksamalarda büyük sabır gösteren, tez danışmanım Sayın Yrd. Doç. Dr. Hakkı Alparslan ILGIN'a (Ankara Üniversitesi Elektronik Mühendisliği Anabilim Dalı) teşekkürlerimi sunarım.

Tez çalışmalarım sırasında fikirlerini ve deneyimlerini paylaşan, iş hayatımdaki gecikmeleri anlayışla karşılayan, teknik liderim Dr. A. Neslin İSMAİLOĞLU'na, bu süreçte benden destek ve yardımlarını esirgemeyen mesai arkadaşlarım Ozan YILMAZ, H. Erdem KAZAK'a teşekkürü borç bilirim.

Manevi destekleriyle beni hiçbir zaman yalnız bırakmayan çok değerli arkadaşlarım Erdem AĞAOĞLU ve Onur ATAR'a teşekkür ederim.

Son olarak, hayatımın her anında maddi ve manevi desteklerini esirgemeyen, emekleriyle bugünlere ulaşmamı sağlayan kıymetli babam İbrahim KARAKUŞ ve sevgili annem Gülnaz KARAKUŞ'a, desteğini her an arkamda hissettiğim ve tez sürecinin yoğun dönemlerinde göstermiş olduğu sabır ve fedakarlıktan ötürü Derya ÇEVİK'e teşekkürlerimi sunarım.

Koray KARAKUŞ
ANKARA, Ekim 2011

İÇİNDEKİLER

KISALTMALAR DİZİNİ

ACD	Alternatif Cholesky Ayrıştırması (Alternative Cholesky Decomposition)
ASIC	Uygulamaya Özel Tümdevre (Application Specific Integrated Circuit)
BP	Taban Arayış (Basis Pursuit)
BPDN	Taban Arayışı Arındırması (Basis Pursuit Denoising)
CLB	Yapılandırılabilir Lojik Blok (Configurable Logic Block)
CPU	Merkezi İşlem Birimi (Central Processing Unit)
CD	Cholesky Ayrıştırması (Cholesky Decomposition)
DSP	Sayısal İşaret İşleme (Digital Signal Processing)
FPGA	Sahada Programlanabilen Kapı Dizisi (Field Programmable Gate Array)
GPU	Grafik İşlemci Ünitesi (Graphics Processing Unit)
HDL	Donanım Tanımlama Dili (Hardware Description Language)
IID	Bağımsız ve Özdeşçe Dağılmış(Independent and Identically Distrubuted)
IC	Tümleşik Devre (Integrated Circuit)
JPEG	Birleşik Fotoğraf Uzmanları Grubu (Joint Photographic Expert Group)
LUT	Değer Tablosu (Look-Up Table)
MGS	Modified Gram Schmidt
MMCM	Karışık Mod Saat Yöneticisi (Mixed-Mode Clock Manager)
MP	Eşleştirme Arayış (Matching Pursuit)
MRI	Manyetik Rezonans Görüntüleme (Magnetic Resonance Imaging)
OMP	Dik Eşleştirme Arayış (Orthogonal Matching Pursuit)
RTL	Yazmaç Aktarım Seviyesi (Register Transfer Level)
SA	Sıkıştıcı Algılama (Compressive Sampling)
SPD	Simetrik ve Pozitif Tanımlı (Symmetric Positive Definite)
VLSI	Çok Geniş Ölçekli Tümleştirme (Very Large Scale Integration)
VHDL	VHSIC Donanım Tanımlama Dili (VHSIC Hardware Description Language)

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Sinyallerin veya görüntülerin frekans verilerinden toplanması ve yeniden oluşturulması konusundaki klasik yöntemlerde ve yaygın uygulamalarda temel prensip olan Nyquist örnekleme teoremi kullanılır. Nyquist teoremi, bir sinyali veya görüntüyü yeniden oluşturabilmek için sahip olunması gereken Fourier örneği sayısının, sinyalin veya görüntünün istenen çözünürlüğünü karşılaması gerektiğini ifade eder (Baraniuk 2007).

Sıkıştırıcı Algılama (SA), klasik görüntü sıkıştırma yöntemlerinin verimli olmadığını, sinyallerin veya görüntülerin istenen çözünürlüğünün çok altında bir sayıda örnek alınarak yaklaşık olarak, bazı zamanlarda ise tam olarak yeniden oluşturulabilmesinin mümkün olduğunu öne sürer (Candès 2006). Bu yeni veri toplama protokolü klasik yöntemler için gerekli olduğu düşünülen sensör sayısından daha az sayıda sensörle analog bilgiyi sayısal forma çevirmeyi hedefler. Bu yeni örnekleme teorisinin altında yatan prosedür, örnekleme ve sıkıştırma işlemini eş zamanlı olarak yapmasıdır.

SA yönteminin temelindeki fikir ilk olarak 2004 yılında manyetik rezonans problemleriyle uğraşan bir matematikçi tarafından ortaya atılmıştır. Test görüntülerinin, Nyquist kriterini sağlamayacak kadar az bir veri sayısı ile yeniden oluşturulabildiğini keşfetmiştir.

Bu çalışmada, SA olarak anılan bu yeni örnekleme teorisinin temelini oluşturan önemli matematiksel görüşler ve SA teorisindeki yeniden oluşturma algoritmaları incelenmiştir. FPGA üzerinde gerçekleştirilen OMP (Dik Eşleştirme Arayış - Orthogonal Matching Pursuit) algoritması ayrıntıları ile incelenip açıklanmıştır. OMP algoritması ile daha önce yapılmış uygulamalarla arasındaki performans ve yöntem farklılıkları ortaya konulmuştur.

2. KURAMSAL TEMELLER

2.1 Sıkıştırıcı Algılama

2.1.1 Sıkıştırıcı algılama teorisi

SA teorisi bazı sinyallerin veya görüntülerin klasik yöntemlere göre çok daha az örnek veya ölçüm alınarak yeniden oluşturulabileceğini öne sürer. Bu önermenin gerçekleştirilebilmesi iki temel prensibe dayanır. Bu prensiplerden ilki seyrekliktir. Seyreklik sinyalin ilgilenilen bölgede gösterimiyle alakalıdır. İkincisi olan ilişkisizlik ise sinyalin algılama yaklaşımıyla alakalıdır.

Seyreklikle anlatılmak istenen, herhangi bir sürekli zaman sinyalinin içerdiği bilgi oranı, sinyalin bant genişliğinden çok daha az olması veya bir kesikli zaman sinyalinin kendisinin sonlu uzunluğuna kıyasla çok daha az bir değerle ifade edilebileceğidir. Daha açık bir şekilde ifade edilirse, birçok doğal sinyalin uygun bir ψ bölgesinde ifade edildiğinde bulunduğu bölgeye oranla daha kısa bir gösteriminin olması gerçeği, sinyalin seyrek olması anlamına gelir (Candès 2006).

İlişkisizlik ise zaman ve frekans arasındaki dualite kavramını vermeye çalışır ve şu fikri ifade eder: ψ bölgesinde seyrek gösterime sahip olan sinyallerin, tıpkı zaman bölgesindeki dürtü sinyallerinin frekans bölgesine yayıldığı gibi, toplanıldığı bölgeye yayılması gerekir (Haupt ve Nowak 2006).

SA teorisinde amaç, seyrek bir sinyalin içinde bulunan önemli bilgi içeriğini yakalayabilen ve bu bilgileri sinyalin orjinal miktarından daha az bir miktar veri içine sıkıştırabilen etkin bir örnekleme veya algılama protokolleri tasarlayabilmektir. Bu protokoller adaptif değildir ve seyreklik bölgesiyle ilişkisiz olan sabit bir dalga biçimi ile sinyali ilişkilendirmeye yarar (Candès ve Wakin 2008). Örnekleme protokolleriyle ilgili bilinmesi gereken önemli detaylardan biri de seyrek sinyallerin içerdiği bilgiyi sinyali anlamakla uğraşmadan bir sensör yardımıyla etkin bir şekilde yakalanmasına

izin vermesidir. Sinyalin orjinal miktarına oranla daha az miktar veri kullanılarak nümerik optimizasyon yöntemleri yardımı ile orjinal sinyalin yeniden oluşturulabiliyor olması SA teorisinin yaygınlaşmasının ana sebeplerindendir.

SA düşük oranda örnekler alan basit ve etkili bir sinyal toplama protokolüdür. İşlem gücünü ise örnekleme kısmında kullanmayıp eksik olarak ifade edebileceğimiz veri setinden sinyali tekrar oluşturmaya çalışırken kullanır (Candès ve Wakin 2008).

2.1.2 Algılama problemi

Büyük boyutlardaki bir *x* sinyali ϕ algılama matrisi yardımı ile orjinaline oranla daha küçük bir *y* matrisi içerisine toplanır.

$$y = \langle x, \phi \rangle, \tag{2.1}$$

y_k, ölçüm vektörü olarak adlandırılan *y* vektörünün *k*. elemanlarını; ϕ_k, algılama matrisi olarak adlandırılan ϕ matrisinin kolon vektörlerini gösterir ise *y* matrisinin her bir elemanı aşağıdaki gibi hesaplanabilir.

$$y_k = \langle x, \phi_k \rangle, \ k = 1,2, \dots, m. \tag{2.2}$$

Yukarıdaki denklemde *x* sinyali istenilen bir ϕ_k dalga biçimiyle y_k matrisine toplanır. Bu standart bir yapıdır. Örnek vermek gerekirse; algılama bölgesi dürtü delta fonksiyonu ise *y* ölçüm vektörü zaman veya uzay bölgesinde *x* sinyalinin örneklenmiş değerleri olabilir.

Denklem 2.3'te gösterildiği gibi *y* matrisinde toplanan *m* sayıdaki ölçümler *x* sinyalinin boyu olan *n* değerinden çok daha küçük olur.

$$m \ll n, \tag{2.3}$$

m sayıdaki ölçüm sinyalin sıkıştırılabilme derecesini gösterir ve sıkıştırma oranı olarak adlandırılabilir. Sıkıştırma oranı sinyalin seyrekliği ile orantılıdır ve seyrek bir görüntüde sıkıştırma oranı Nyquist oranının altında bir değere ulaşabilmektedir.

Bu kadar az sayıda örnekle sinyali elde etmek istenilmesinin birçok sebebi olabilir. Sensör sayısının kısıtlı olması, nötron saçılması ile görüntü işleme uygulamalarındaki gibi ölçüm almanın çok pahalı olması veya algılama işlemi MRI (Manyetik Rezonans Görüntüleme - Magnetic Resonance Imaging)'daki gibi çok yavaş olduğundan dolayı nesneden sadece birkaç kez ölçüm alabilme imkanı bulunması az sayıda örnekle sinyali elde etmek istenilmesinin sebepleri olarak sayılabilir. Bunlar gibi birçok örnek verilebilir.

Yukarıda anlatılanlar şu önemli gerçekleri beraberinde getirmektedir:

- Bir sinyali sadece *m* sayıda ölçüm ile tam olarak yeniden oluşturulabilmenin mümkün olabilirliği,
- Sadece *m* tane ölçüm alarak sinyalin bütün önemli bilgilerini yakalayabilecek etkin bir algılama mekanizması tasarlama imkanı,
- Elde edilecek az bir miktar veri setinden *x* sinyalinin geri getirilebilmesi.

Burada yapılması gereken az tanımlanmış bir doğrusal denklem sistemi çözümüdür. ϕ matrisinin *m* x *n* boyutlu, satırları $\phi_1^*,\ldots, \phi_m^*$ vektörlerinden oluşan bir algılama matrisi olduğunu düşünülürse; $m < n$ olduğu sürece *x* sinyalini denklem 2.1 yardımı ile yeniden oluşturma işleminin kötü konumlanmış olduğu görülür. Çünkü $y = \phi x$ ise ϕ matrisi $N(\phi)$ boşluk uzayında her *r* vektörü için $y = \phi (x+r)$ eşitliğini sağlar (Candès ve Wakin 2008). Fakat ileriki bölümlerde anlatılacak olan sıkıştırıcı algılama yeniden oluşturma yöntemleri ile sinyalin yeniden oluşturulabildiği görülecektir.

Shannon teoremi, *x* sinyalinin çok düşük bir bant genişliğine sahip olduğu durumlarda, az sayıda örneğin geri oluşturma için yeterli olacağını ifade etmektedir. Çalışmanın ileriki kısımlarında ise az sayıda örnekle geri oluşturmanın çok büyük bant genişliğine sahip olan sinyaller için de geçerli olacağı görülecektir.

2.1.3 Seyreklik

Birçok doğal sinyal, sinyal çeşidine uygun bir bölgede ifade edildiği sürece kısa bir gösterime sahiptir. Şekil 2.1'de görülen görüntünün Dalgacık (wavelet) dönüşümü katsayı değerleri şekil 2.2'de görülmektedir.

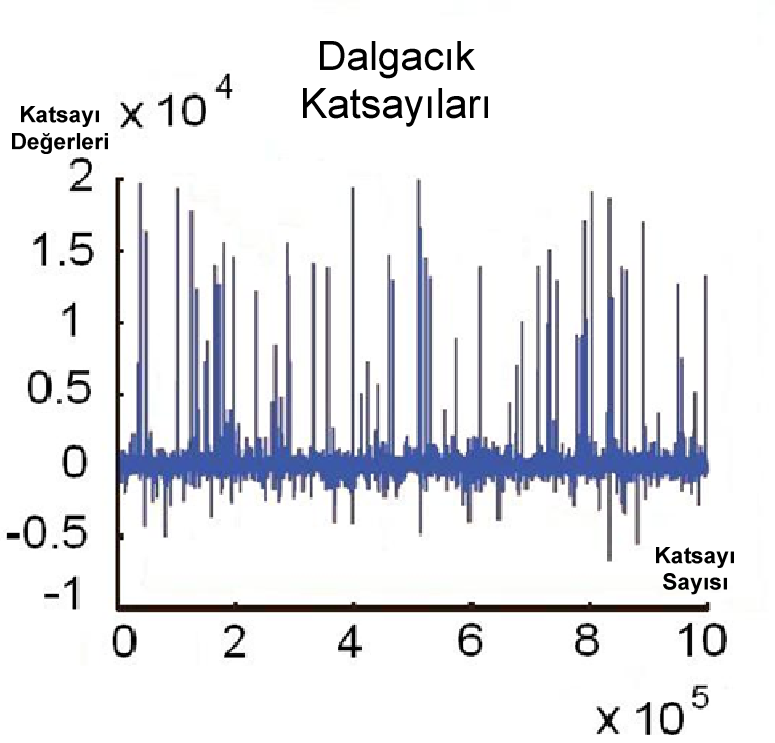
Şekil 2.1 Piksel değerleri 0 ile 255 arasında değişen (256 gri seviyeli) bir görüntü

Görüntü üzerindeki nerdeyse tüm piksellerin sıfırdan farklı değerlere sahip olmasına rağmen dalgacık katsayıları daha kısa bir gösterim imkanı sunmaktadır. Orjinal görüntüye Dalgacık dönüşümü uygulandığında elde edilen Dalgacık katsayılarının büyük çoğunluğu sıfırdır. Az sayıdaki büyük değerler bize görüntü hakkındaki bütün bilgileri vermektedir (Candes ve Romberg 2006).

Matematiksel olarak ifade edilirse elimizde bir x vektörü varsa (şekil 2.1'deki n piksellik görüntü gibi) ve bu sinyal ortonormal $\psi = [\psi_1 \psi_2 \ldots . \psi_n]$ gibi bir bölgeye yayılırsa (dalgacık bölgesi gibi) aşağıdaki şekilde bir denklem elde edilir:

$$x = \sum_{i=1}^{N} w_i \, \psi_i \tag{2.4}$$

Bu denklemdeki w_i'ler x sinyalinin katsayılar dizinidir. Yani x sinyali ψw şeklinde ifade edilebilir. Artık seyreklik ifadesini uygulamaya koymak daha anlaşılır bir hale gelmiştir. Eğer bir sinyal seyrek bir gösterime sahipse bütün küçük katsayılarının atılması algısal olarak hiçbir kayba sebep olmamaktadır. Çünkü az sayıda olan büyük katsayılar sinyal hakkındaki bütün bilgiyi içermektedir.



Şekil 2.2 Orjinal görüntünün dalgacık katsayıları

x sinyalinin uygun bir bölgede gösterildiği zaman elde edilecek w_i katsayılarından sadece S tane en büyük katsayıları alınıp kaydedilirse, oluşturulan bu vektöre x_s denilebilir. Matematiksel olarak $x_s = \psi \, w_s$ denklemiyle ifade edilebilen w_s, w_i katsayılar dizisinden sadece S tane en büyük katsayıları ifade eder. Bundan sonra S tane sıfır olmayan büyük katsayı ile ifade edilebilen ve dolayısıyla seyrek olan sinyallere S-seyrek denilecektir.

ψ bölgesi bir ortonormal bölge ise aşağıdaki şekilde bir eşitlik yazılabilir.

$$\|x - xs\|_{l2} = \|w - ws\|_{l2} \quad (2.5)$$

Sonrasında *w* katsayıları, w_s katsayıları yardımıyla orjinal değerlere çok yakın olacak şekilde tahmin edilebilir. Dolayısıyla $\|w - ws\|_{l2}$ hatası küçük olur. Başka bir şekilde ifade edilirse katsayıların büyük bir çoğunluğu bize herhangi bir kayıp getirmediğinden kolayca atılabilir. Şekil 2.3'te algısal olarak kaybın anlaşılmasının neredeyse imkansız olduğu resimin katsayılarının % 97,5'i atılarak yeniden oluşturulmuştur (Candes ve Romberg 2006).

Yukarıda bahsedilen prensip aslında JPEG2000 (Birleşik Fotoğraf Uzmanları Grubu - Joint Photographic Expert Group) gibi bütün modern sıkıştırma algoritmalarının temelinde yatar. Veri sıkıştırma için basit bir yöntem, *x* sinyalinden *w* katsayılarını hesaplayabilir ve *S* sayıda önemli katsayıların değerlerini ve sinyal içindeki konumlarını kodlayabilir. Fakat böyle bir kodlama işlemi *x* sinyali içindeki tüm *n* tane *w* katsayısı hakkında bilgi sahibi olma yükünü de beraberinde getirir. Daha genel olarak, seyreklik etkin ve temel bir sinyal işleme modelleme aracıdır. Örneğin hassas istatistiksel tahmin ve sınıflandırma, etkin veri sıkıştırma ve bunun gibi birçok dalda önemli rol oynar. Seyrekliğin veri toplama işleminde de çok etkin bir rol aldığını söyleyenebilir. Seyreklik bir kişinin ne kadar etkin bir şekilde veriyi toplayabileceğini belirler (Candès ve Wakin 2008).

Şekil 2.3 Dalgacık bölgesinde yaklaşık olarak 25000 dalgacık katsayısı ile yeniden oluşturulan görüntü

2.1.4 İlişkisizlik

SA problemlerinde ortonormal bölgede bir (ϕ, ψ) bölge çifti verildiği düşünülürse; bunlardan ilki olan ϕ bölgesi x sinyalini algılamaya yarar. İkincisi olan ψ bölgesi ise x sinyalini ifade etmeye yarar.

Algılama bölgesi ve gösterim bölgesi arasındaki ilişkiyi ifade eden denklem;

$$\mu(\phi, \psi) = \sqrt{n}.max_{1<k,\ j\leq n} |\langle \phi_k \psi_j \rangle| \tag{2.6}$$

olarak verilir (Candès ve Romberg 2006).

Sözlü olarak ifade edersek ϕ ve ψ bölgeleri gibi herhangi iki bölge arasındaki ilişkinin ölçüsü iki bölge arasındaki en büyük benzerliği olan elemanlar ile ifade edilir (Donoho

ve Huo 2001). ϕ ve ψ bölgeleri birbiriyle benzerliği olan elemanlar içeriyorsa bölgeler arasındaki ilişki büyük olur, aksi halde aradaki ilişki küçük bir değer alır.

SA teorisinde genel olarak düşük ilişkili bölge çiftleriyle ilgilenilir. Düşük ilişkili bölge çiftlerine bir örnek verilirse; ϕ bölgesi kanonik bir bölgede $\phi_k(t) = \delta(t-k)$ denklemiyle, ψ bölgesi ise Fourier bölgesinde $\psi_j(t) = n^{-1/2} e^{i.2\pi jt/n}$ denklemiyle ifade edilir. Zaman frekans çifti $\mu(\phi, \psi) = 1$ kuralına uyması gerekir. Bu kurala uyan bölge çiftleri arasında maksimum ilişkisizlik olduğu anlaşılır.

Bir diğer örnek ise ϕ algılama bölgesi için dalgacık bölgesi seçilmesi, ψ gösterim bölgesi için gürültücük (noiselet) bölgesi seçilmesidir. Dalgacık bölgesinin de çeşitleri olduğu için gürültücük bölgesi ile dalgacık bölgeleri arasındaki ilişkisizlik değerleri aşağıdaki gibidir.

- Gürültücük ile Haar Dalgacık arasındaki ilişkisizlik $\sqrt{2}$ 'dir.
- Gürültücük ile Daubechies D4 Dalgacık arasındaki ilişkisizlik 2.2 'dir
- Gürültücük ile Daubechies D8 Dalgacık arasındaki ilişkisizlik 2.9 'dur.

SA teorisinde gürültücük bölgesi ile ilgilenilmesinin iki tane ana sebebi vardır. Bunlar;

- Gürültücük bölgesi görüntü verisi veya diğer tip verilerin seyrek gösterimiyle maksimum düzeyde bağlantısız olur.
- Gürültücük bölgesinin çok hızlı bir algoritması vardır. Gürültücük dönüşümü tıpkı Fourier dönüşümü gibi *0(n)* zamanında çalışmaya başlar ve gürültücük matrisinin bir vektöre uygulanması için depolanmasına gerek yoktur (Candès ve Wakin 2008).

İlişkisizlik konusu ile ilgili son olarak rastgelelik konusuna değinilmesi gerekir. Rastgele matrisler herhangi bir sabit ψ bölgesi ile büyük oranda ilişkisizdir. Herhangi bir ϕ bölgesi seçilir ve ϕ matrisi birim çember üzerinden bağımsız ve düzgün rastgele seçilmiş n sayıda vektör örnekle ortonormalize edilir. Elde edilen ϕ ve ψ bölgeleri arasındaki ilişkisizlik yaklaşık olarak $\sqrt{2\log(n)}$ olur.

2.1.5 Seyrek örnekleme

Herhangi bir sıkıştırma yöntemi kullanılmadan *x* veri setini elde edebilmek için *x* veri setindeki *n* sayıda eleman toplanır. Fakat SA teorisinde bu elemanların bir alt kümesi kadar sayıda veri toplanır. Anlatılanlar denklemle ifade edilirse;

$$y_k = \langle x, \phi_k \rangle, \ k \epsilon M, \tag{2.7}$$

Denklem 2.7'deki *M*, *n* sayıdaki bir kümenin alt kümesidir ($M \subset \{1,2,\ldots,n\}$). Bu bilgilerden sonra *m* tane ölçümü içeren *y* vektörünü kullanarak *x* sinyali L_1 norm minimizasyon yöntemiyle yeniden oluşturulacaktır. L_1 norm minimizasyon yöntemi SA problemlerinin çözümü için önerilen ve $x^* = \psi w^*$ denklemiyle verilen yeniden oluşturma yöntemidir. Bu denklemdeki w^* değerleri konveks optimizasyon programının çözümüdür. L_1 norm minimizasyonun denklemi aşağıdaki gibidir:

$$min_{w \epsilon R} \|\tilde{w}\|_{l1} \qquad y_k = \langle \phi_k, \psi \tilde{w} \rangle, \ \forall k \epsilon M. \tag{2.8}$$

Burada katsayı dizisinin L_1 normuna göre minimum olduğu yerler bulunur. L_1 normunun seyreklik artırıcı fonksiyon olarak kullanımı çok eskilere dayanmaktadır. Bu algoritmanın kullanımına örnek olan en eski çalışmalardan biri sismolojideki yansıma problemleridir. Seyrek yansıma fonksiyonları sınırlı banttaki verilerde aranmıştır. Çözüm yolu olarak da L_1 norm minimizasyonu kullanılmıştır (Santosa 1986). L_1 norm minimizasyonu yöntemi bu tür seyrek sinyalleri yeniden oluşturmak için kullanılan tek yöntem değildir. Açgözlü (Greedy) algoritması gibi yeniden oluşturma algoritmaları da bulunmaktadır. Buradan çıkarılabilecek en önemli sonuçlardan biri sinyal yeterli derecede seyrekse ve seyreklik oranı ne kadar fazla ise L_1 norm minimizasyon yöntemi ile yeniden oluşturma işlemi o kadar başarılı olur.

2.1.6 Ölçüm miktarı

Sabit bir *x* sinyali gerçek sayılar kümesinde ve *x* sinyalinin katsayılar dizisi ψ bölgesine göre *S*-seyrek ise, *m* sayıda ölçüm değeri ϕ bölgesinden rastgele ve düzgün olarak seçilir. Eğer;

$$m \geq C\mu^2(\phi,\psi)S\log(n) \tag{2.9}$$

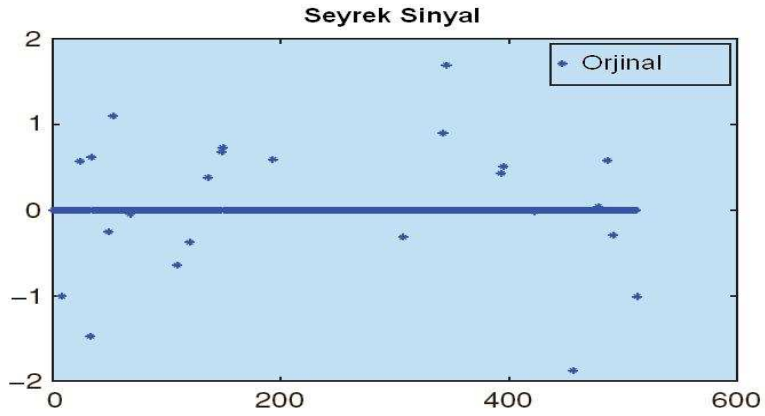
denklemi pozitif bir *C* katsayısı için sağlanıyorsa L_1 norm minimizasyonu algoritması yeniden oluşturma işlemini büyük oranda yapabilmektedir. Eğer $m \geq C\mu^2(\phi,\psi)Slog\,(n/\delta)$ ise L_1 norm minimizasyonu algoritmasının başarı oranı *(1-* δ*)* değeri kadardır. Bu denklemle ilgili aşağıdaki üç yorum daha iyi anlaşılmasını sağlayacaktır (Candès ve Romberg 2007).

- Bölge çifti arasındaki ilişki konusu gayet açıktır. Bölge çifti arasındaki ilişki ne kadar az olursa, sinyali yeniden oluşturabilmek için gereken ölçüm sayısı o kadar az olur. SA teorisi bölge çiftleri arasındaki ilişkinin az olduğu sistemlerle ilgilenir ve bu şekilde sistemler oluşturmaya çalışır. Çünkü ne kadar az ölçümle orjinal sinyal yeniden oluşturulabilirse o kadar başarılı bir iş çıkarılmış olur.
- Bölge çifti arasındaki $\mu(\phi,\psi)$ ilişkisizliği 1 veya 1'e ne kadar yakın olursa o kadar iyidir. Sinyali hiçbir bilgi kaybı olmadan yeniden oluşturmak için *n* tane örnek yerine denklemde $\mu(\phi,\psi)$ ifadesi 1 olacağı için *Slog(n)* sayıda ölçüm yeterli olacaktır.
- *x* sinyali bir konveks optimizasyon algoritmasını minimize ederek sıkıştırılmış veri setinden hiçbir kayıp olmadan yeniden oluşturulabilir. Bu işlemi yaparken sinyalin sıfır olmayan katsayıları hakkında, bu katsayıların yerleri veya genlikleri hakkında hiçbir bilgi sahibi olunması gerekmemektedir. Sadece minimizasyon algoritması çalıştırılır. Sinyal yeteri kadar seyrekse mükemmel biçimde yeniden oluşturulur.

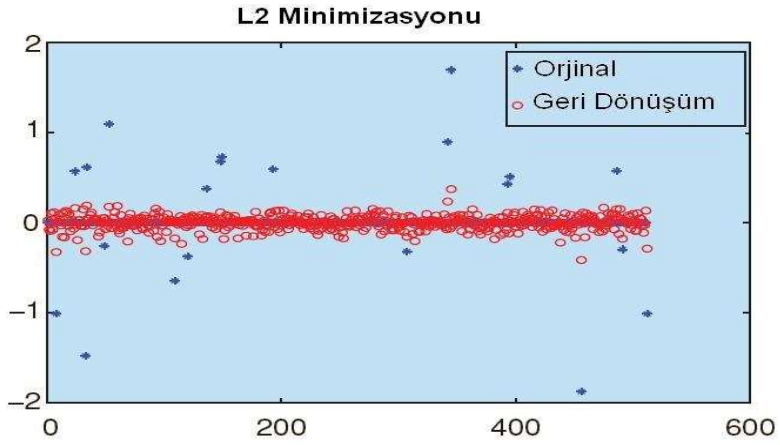
Aslında bu ilişkisiz bölgede rastgele örnekleme teoremi, seyrek sinyallerin örneklenmesi hakkında eskiye dayanan çalışmaları genişletmektedir. Rastgelelik üzerine kurulan sistemler aşağıdaki sonuçları ortaya çıkarır;

- Rastgelelik çok etkin bir algılama mekanizmasıdır.
- İyi bir ispat yapabilmek için uygun bir sistemdir. Bugüne kadar yapılmış olan birçok SA teorisi uygulamalarını tetikleyici etkide bulunmuştur.

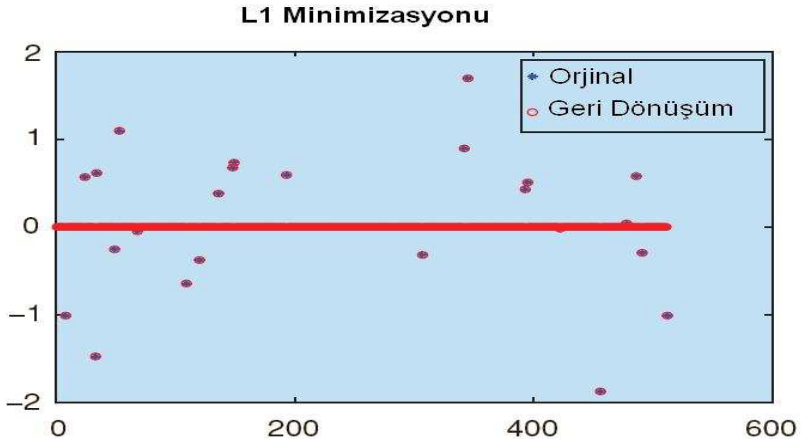
Çok geniş bantta seyrek bir sinyal örneklenmeye çalışıldığında; sinyal $f(t) = \sum_{j=0}^{n-1} x_J \ e^{i2\pi jt/n}, t = 1,2 \ldots\ldots, n-1$ formunda olduğu düşünülürse buradaki n çok büyük bir değerdir. Fakat sıfır olmayan katsayılar dizisi olan x_j, S değerinden küçük veya bu değere eşittir. Sinyalin içerdiği frekanslardan hangi frekansların aktif olduğu bilinmiyor veya aktif olan setin genlikleri bilinmiyor ise Nyquist teoremi şu açıklamasıyla soruna yardımcı olur: ''Eğer sinyalin bant genişliği önceden sınırlanamıyorsa n sayıda örnek gerekir.'' Daha önce keyfi olarak seçilen $S\log(n)$ tane örnekle sinyalin yeniden oluşturulabileceği denklemlerle gösterilmişti. Üstelik seçilen bu $S\log(n)$ tane örneğin dikkatlice seçilmesi gerekmiyor. Bu boyuttaki her örnek setinden aynı şekilde sinyal yeniden oluşturulabilir. Anlatılanları açıklayan ve L_1 minimizasyonunun etkisini gösteren bir örnek Şekil 2.4 - 2.6'da verilmektedir (Candès ve Wakin 2008).



Şekil 2.4 Gerçek değerli seyrek bir sinyal



Şekil 2.5 L_2 norm minimizasyonu ile 60 Fourier katsayısı ile yeniden oluşturulmuş veri seti



Şekil 2.6 L_1 norm minimizasyonu ile 60 Fourier katsayısı ile yeniden oluşturulmuş veri seti

2.2 Yeniden Oluşturma Algoritmaları

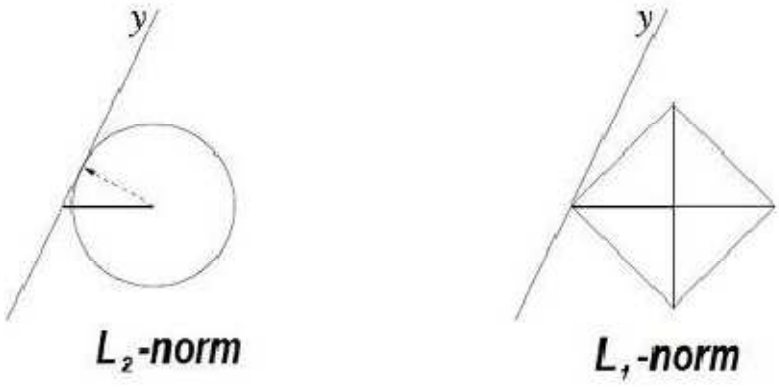
2.2.1 Taban arayış algoritması

BP (Taban Arayış - Basis Pursuit) aslında tam olarak bir yeniden oluşturma algoritması değildir. BP yöntemi kompleks optimizasyon yöntemini kullanarak aşırı tanımlı sözlüklerde sinyal gösterimlerinin bulunmasını sağlayan bir prensip olarak tanımlanabilir (Donoho vd. 1998). Yukarıda bahsedilen sözlük kelimesinden kasıt parametrelenmiş dalga biçimleridir. Dolayısıyla sinüs dalga biçimleri kullanan bir frekans sözlüğü, dalgacıklara bağlı bir zaman sözlüğü hatta bir nokta haricinde sıfır olan basit bir dalgalar topluluğu olabilir. Yöntemin karmaşıklığını ve yapılabilirliğini bunlar belirleyecektir. Bu sözlük SA problemlerine uygulandığında, algılama matrisi (ϕ) tarafından verilir. Sonuçta BP algılama matrisinin aşırı tanımlı bir sözlük olduğu ve sinyal gösteriminin orjinalinden toplanan bir sinyal (x) olduğu, SA teorisindeki yeniden oluşturma problemine bir çözüm bulur:

$$y = \phi x, \tag{2.10}$$

Genel hatlarıyla prensip, katsayıların küçültülüp seyrekleştirilerek sinyalin bir gösteriminin bulunmasıdır. Gösterimin bulunabilmesi için, en seyrek çözümün tekil özelliği olan L_1 norm içinde en küçük hale getirilmesi gerekir. L_2 norm ile çözüm hiçbir zaman seyrekliği koruyamaz. Bu durum deneysel olarak aşağıdaki gibi açıklanmıştır;

Şekil 2.7'de L_2 norm çözümü, en küçük öklid topu ve alt uzay y'nin temas noktası gösterilmektedir. Buna karşın L_1 norm minimizasyonu için, bu top sekiz yüzlü bir cisim şeklini almaktadır. L_1 norm minimizasyonunun çözümü ise alt uzay ile bu sekiz yüzlü cismin herhangi bir köşesinin temas noktasıdır. Bunun yanı sıra sekiz yüzlü cismin herhangi bir köşesi seyrek bir çözüm olacaktır. Yöntem en uygun çözümü bulduğu gibi aynı zamanda bulunan çözümün seyrek olduğunu da garantiler. Bütün bunlar aslında eksik tanımlanmış doğrusal denklem sistemlerindeki minimal L_1 norm çözümünün aynı zamanda en seyrek çözüm olması gerçeğine dayanır (Rudelson ve Vershynin 2005).



Şekil 2.7 Minimizasyon problemine deneysel yaklaşım

2.2.1.1 Algoritma tanımı

Problem L_1 norm içinde bir optimizasyon olarak tanımlanırsa;

$$min\|x\|_{l1} \ ; \ \phi x = y. \tag{2.11}$$

İkinci derece olmayan, konveks bir optimizasyon problemi olur ve Denklem 2.12'deki gibi bir doğrusal programlama problemine dönüştürülür.

$$minc^T X \ ; \ AX = bX \geq 0. \tag{2.12}$$

Burada;

$$X \in \Re^m, \ X := (u; v), c := (1; 1), A := (\phi, -\phi), b := y \tag{2.13}$$

ve toplanan sinyal denklem 2.14 yardımıyla bulunur.

$$x := u - v \tag{2.14}$$

Süreç boyunca, sıfır olmayan katsayılar A matrisinin m sütunu ile ilişkilendirilir ve bunlar $\Re^m$ için bir temel oluşturur. Çözüm, bu temel tarafından, tekrarlamalı bir süreç ile yapılan optimizasyonlar sonrasında oluşacaktır. Bu süreç, X çözümünü minimize

eden kombinasyonun bulunması için, temelin sütunlarını değiştirmeyi içerir. Doğrusal programlama probleminin çözümü BP sürecidir.

Tüm doğrusal programlama algoritması içinde, BP çözümü için en ilginç olanlar karakteristiklerinden ötürü Simpleks (simplex) yöntemi ve İç nokta (Interior) yöntemidir.

İlk yöntem olan BP-Simpleks, *A* matrisinin doğrusal bağımsız bir sütunundan başlar. Tekrarın her bir adımında kullanılan değiştirme, hedef fonksiyonu en fazla iyileştiren olacaktır. Bu konudaki çalışmalar (Candès ve Tao 2005) yakınsamayı garantilemek için terimlerin nasıl seçilmesi gerektiğini gösterir. Yöntem, optimal çözüm dışındaki her değiştirmede iyileştirme sağlar ve yöntemin hızını kısıtların sayısı belirler. Değişkenlerin sınırları tam olarak hesaplanır ve bu olay yönteme fazladan hesaplama yükü getirir (Jokar ve Pfetsch 2007).

Diğer taraftan BP-iç nokta yöntemi sütünları değiştirme işini yapmaya devam eder. Fakat bu yöntem her zaman en uygun değişimi yapamamaktadır. Tüm uygun noktalar konveks bir çok yüzlü olarak düşünülürse, simpleks yöntemi çokyüzlünün kenarları üzerinde gezinirken iç nokta yöntemi çok yüzlünün içerisinde gezip son tekrarda kenara ulaşarak çözümü bulur. Bu yöntem her tekrarda daha fazla bilginin işlenmesini gerektirmektedir. Bazı durumlarda ara tekrarlar mantıklı değerler vermese de sonuçta mantıklı değerler elde edilecektir (Donoho vd. 1998).

2.2.1.2 Algoritmanın karakteristiği

BP algoritmasının özgün özellikleri, algoritmayı SA problemlerinin çözümü için referans sayılabilecek algoritmalar içine taşımıştır. BP algoritmasına alternatif olabilecek birçok algoritma ortaya çıkmasına rağmen hiçbiri az sayıda örnek ile BP algoritmasının hatasızlığına ulaşamamıştır.

BP algoritması sağlam bir teorik temel üzerine kurulmuştur. Bir sinyal bazı dönüşüm bölgelerinde tam anlamıyla seyrek ise, BP algoritması hatasız olarak yeniden oluşturma

işlemini gerçekleştirebilmektedir. Gerekli örnek sayısı ve işlem süresi, sinyalin seyreklik derecesi ve matrisin özellikleri tarafından belirlenecektir.

Genel optimizasyon tabanlı bir algoritma olduğundan diğer algoritmaların başaramayacağı istikrarlılıkla çözümlemeyi başarır (Donoho vd. 1998). Diğer algoritmalara oranla daha az ölçümler kullanır. Dolayısıyla bu algoritmaya toplayıcı kısmında karmaşıklık getirir ve örnekleme oranı belirgin bir şekilde artar. Diğer algoritmaların tersine BP seyrek gösterim problemine, doğrusal programlama yaklaşımı tabanlı, sıfır olmayan katsayıların sayısını en aza indirmek yerine katsayıların mutlak değerlerinin toplamını en aza indirmeye çalışır.

BP algoritması diğer algoritmalara göre oldukça yavaş çalışmaktadır. Açgözlü algoritmalar gibi yerel optimum değerleri değil, genel optimum değerleri bulma tabanlı olduğu için işlem süresi çok uzun sürebilmektedir.

Algoritma gürültülü veriler ile denendiğinde oldukça istikrarlı olduğu görülür. Aslında yüksek değerde gürültü olduğunda bile en iyi çözümü bulabilen bir BP algoritması varyasyonu da vardır. Bu algoritmanın ismi de BPDN (Taban Arayışı Arındırması - Basis Pursuit Denoising)'dır. BP ile aynı temele dayanır. Aralarındaki fark BP algoritmasının kısıtlarının biraz iyileştirilmesine dayanır.

$$min\|x\|_{l1} \;;\; \|\phi x - y\|_{l2} \leq \sigma \tag{2.15}$$

Denklemdeki σ ifadesi gürültü seviyesinin tahmini bir değeridir. BP algoritması bu tür problemleri yüksek bir doğruluk oranıyla çözebilmektedir.

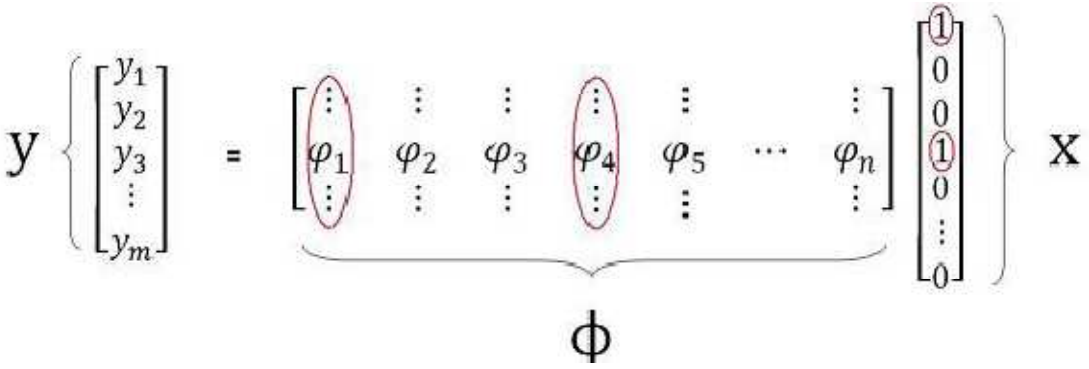
BP algoritmasının diğer algoritmalara göre dezavantajları da vardır. Bunlardan biri işlem gücü çok yüksek bir algoritma olduğu için gerçekleştirmesi çok zordur. BP algoritması birçok durumda aşırı sayıda tekrara ihtiyaç duymaktadır. İşlem süresi için limit koymak veya yaklaşık bir tahminde bulunmak çok zordur. Bu algoritmayı gerçekleyen simülasyonlar, matris ile sinyal arasındaki ilişkinin ve matrisin özelliklerinin bu algoritmanın hızını önemli ölçüde etkileyebileceğini göstermiştir.

Matrisin izometri özelliklerine uyması gerekir. Sınırlı izometri değerlerine tam olarak uyabilen bir matris algoritmanın performansını büyük ölçüde yukarı taşıyabilir. Eğer matrisin elemanları iid (Bağımsız ve Özdeşçe Dağılmış - Independent and Identically Distributed) değerlerden oluşuyor ise isometri şartlarını çok büyük bir olasılıkla sağlıyor demektir. Buna karşın daha öncede belirtildiği üzere seyreklik de büyük öneme sahiptir. Bir matris ne kadar yoğun ise, işlem süresi o kadar uzun olur.

2.2.2 Dik eşleştirme arayış algoritması

Örnekleme matrisini işlemenin hesaplama yükünün fazla olması nedeniyle, sadelik ve hız gereksinimlerini karşılamak için Doğrusal Programlama'ya alternatif bir yöntem ihtiyacı duyulmaktadır. Yoğun matrisler bazı gerçek zamanlı uygulamaların yetişemeyeceği kadar hesaplama yükü getirecektir. Hızı ve uygulama kolaylığı açısından en büyük alternatifi OMP (Dik Eşleştirme Arayış – Orthogonal Matching Pursuit) yöntemidir. Bu yöntem optimizasyon tabanlı olmayan alternatif bir yöntemdir. Herhangi bir optimizasyon hedefini amaçlamaz fakat örnekleme matrisinin hangi bileşenlerinin sıfırdan farklı elemanlarıyla ilişkili olmadığını belirleyerek seyrek sinyali oluşturur. OMP algoritması ölçüm sayısı olan m, k sayısına göre büyük olduğu sürece k-seyrek bir sinyalden yeniden oluşturma özelliğini başarılı bir şekilde yapabilir (Tropp ve Gilbert 2007). Ancak algoritmanın problemi çözmesi için sinyalin seyreklik derecesi bilgisine ihtiyacı vardır.

OMP algoritması boş bir matris ile algoritmayı çalıştırmaya başlar. Örnekleme matrisinin sıfırdan farklı katsayılarla ilişkili olduğunu saptadığı kolonlar ile bu boş matrisi doldurmaya başlar. Her tekrarda bir tane kolon seçerek bu matrisi oluşturur. OMP algoritması açgözlü bir algoritmadır. Çünkü kolonları açgözlü bir yöntemle seçer. Açgözlülük bir sonraki adımı düşünmeden, o anda bulunan seçeneklerden en iyisini seçmektir. Algoritma her tekrarda örneklenmiş sinyal y ile en çok ilişkili kolonu seçer. Bu yöntemle oluşturulan matris Şekil 2.8'de görülmektedir.



Şekil 2.8 Sıfırdan farklı katsayılarla ilişkili kolonlar(OMP)

Algoritma *k* tekrardan sonra orjinal seyrek sinyalin sıfırdan farklı katsayılarıyla ilişkili kolonlar seçilmiş olur.

2.2.2.1 Algoritma tanımı

Algoritmayı matematiksel olarak ifade edersek; *r* değeri artık vektörü belirtir, s_t değeri her *t.* tekrarda elde edilen y sinyali ile en büyük ilişkisi olan kolonun çarpım değerini ifade eder, S_t ise yaklaşık *x* sinyalinin seçilmiş kolonlarının indis değerlerini tutan bir vektördür. Seçilmiş kolonlardan oluşturulmuş matris *k* tekrardan sonra elde edilmiş olur.

İlk tekrarda değişkenler başlangıç değerlerine alınır:

$$r_0 = y, S_t = \emptyset \; ve \; t = 1 \tag{2.16}$$

İlk kolon aşağıdaki ilişki sayesinde toplanan sinyal yardımıyla seçilir:

$$s_t = argmax_{i=1,\dots,n} |\langle r_{t-1}, \phi_i \rangle| \tag{2.17}$$

Eğer en yüksek ilişkili birden fazla kolon varsa, yöntem bir tanesini belirlenimci bir şekilde seçer (Tropp ve Gilbert 2007). Sonrasında ise seçilen kolonun indeks değeri S_t vektörüne seçilen kolonun kendisi ise *A* matrisine eklenir.

$$S_t = S_{t-1} \cup s_t \ , \ A_t = [A_{t-1}\, \phi_{s_t}] \tag{2.18}$$

Bu aşamadan sonra, yaklaşık sinyali bulmak için yapılması gereken işlem en küçük kareler problemini çözmektir. Bu yaklaşık sinyal son tekrardan sonra nihai çözümü verecektir.

$$x_t = argmin_x \|y - A_t\, x\|_{l2} \tag{2.19}$$

Bu, çözüm için *y* vektörünü tüm kolonlar üzerine yansıtır

$$x = \mathrm{A}_\mathrm{t}^\dagger y \tag{2.20}$$

Bu denklemdeki $\mathrm{A}_\mathrm{t}^\dagger$ ifadesi QR Veya Cholesky faktörizasyonu yöntemiyle hesaplanmış matris tersi işlemidir. Buradaki katsayı vektörü ise sinyalin o andaki tekrarda seçilmiş elemanlarının üzerine ortogonal projeksiyonudur. Bu özellik yönteme adını veren ve algoritmanın her iterasyonda yeni bir eleman seçmesini sağlar. Eğer son tekrar değilse artık vektörün güncellenmesi sağlanır.

$$r_t = y - A_t x \tag{2.21}$$

Sonrasında artık vektör ile en fazla ilişkili olan kolonlar tekrar seçilir. Orjinal sinyalin sıfırdan farklı eleman sayısına karşılık gelen son tekrara kadar aynı adımlar tekrarlanır.

Daha öncede belirtildiği gibi; bütün bu işlemlerden sonra oluşan yaklaşık sinyal orjinal sinyal olan *x*'in S_t vektörünün içerdiği pozisyonlardaki değerleri olacaktır.

2.2.2.2 Algoritmanın karakteristiği

BP yöntemine alternatif bir yöntem olan OMP yöntemi hem teorik hemde deneysel olarak daha hızlıdır. Daha önceki çalışmalarda gösterildiği gibi OMP daha hızlı bir algoritmaya sahiptir ve gerçeklenmesi L_1 norm minimizasyonuna göre daha kolaydır (Elad 2007). OMP, ölçüm vektörü olan *y*'nin enerjisinin büyük bir bölümünü içerebilen,

örnekleme matrisinin kolonlarını tekrarlamalı olarak seçer. Her tekrardaki seçim işlemi, örnekleme matrisinin kolonları ile artık vektörün iç çarpımına bakılarak yapılır. Artık vektör daha önce seçilmiş kolonların y vektöründen çıkarılmış halidir (Takhar vd. 2006). Yöntem k sayıda tekrarda sonlanır. Her tekrar, mxn boyutlarında ϕ matrisi ile çarpma ve en fazla mxk boyutlarında en küçük kareler problemi çözme işini içerir.

Her tekrarın sadeliğinin yanında, tekrarlar sınırlıdır. OMP yönteminin her tekrarda doğru elemanı seçtiği ve k tekrardan sonra doğru bir çözümle sona erdiği kanıtlanmıştır. OMP algoritması için k tekrarda çözüm özelliği yeterli olmasına rağmen en seyrek çözüm ile başarılı bir şekilde yeniden oluşturabilmek için gerekli bir koşul olmadığını bu konudaki çalışmalar kanıtlamıştır (Donoho ve Tsaig 2006). Tekrarların sınırlı olması OMP algoritmasının karmaşıklığını diğer LP yöntemlere göre önemli ölçüde azaltır. Özellikle sinyalin seyreklik seviyesinin düşük olduğu durumlarda algoritma hem daha az adımda biter hemde karmaşıklık göreceli olarak azalır (Dai ve Milenkovic 2008).

OMP algoritmasının diğer algoritmalara göre basit ve hızlı olduğundan bahsedilmiştir. Fakat OMP algortimasının diğer en seyrek çözümü bulmayı hedefleyen hızlı algoritmaların karşısındaki başarısının temelinde yatan ortogonalliğidir. Kullanılan ortogonal projeksiyon sayesinde artık vektör olan r_t herzaman daha önce seçilmiş olan elemanlara ortogonaldir ve bu elemanlar bu sayede birdaha seçilemezler (Blumensath ve Davies 2007). MP (Eşleştirme Arayış - Matching Pursuit) gibi yöntemler veriyi açıklayan bir çözüme yaklaşabilirler fakat bulunan çözümün en seyrek çözüm olduğunu garanti edemezler. OMP yönteminde, artık vektörün seçilen diğer vektörlere göre ortogonal olması algoritmanın karmaşıklığını azalttığı gibi performansını da artırır. OMP yöntemi MP yöntemine göre daha iyi yeniden oluşturmayı garanti eder. Ayrıca bugüne kadar yapılan denemeler sonucunda OMP algortimasının MP algoritmaları ailesinde en iyi performansa sahip olduğu kanıtlanmıştır (Boufounus vd. 2007).

OMP yönteminin bazı dezavantajları da vardır. OMP yöntemi diğer yöntemler kadar istikrarlı bir yöntem değildir. Eğer OMP algoritması bir adımda yanlış bir eleman seçerse orjinal sinyali hiçbir zaman yeniden oluşturamayabilir. Bazı çalışmalar OMP algoritmasının yanlış elemanlar seçtiğini göstermiştir (Tropp ve Needell 2008). Diğer

açgözlü algoritmalar gibi OMP düzgün bir yeniden oluşturmayı garanti edemez. OMP algoritmasının güvenilirliği konusunda sağlam kuramsal temelleri yoktur. Fakat şu ana kadar yapılan yeniden oluşturma deneylerinin %99 'unda çalıştığı görülmüştür.

Her adımda yerel optimum seçimler yapılarak bir seferde bir yaklaşım değerinin bulunması, genel optimizasyon seçimleri kullanan L_1 minimizasyonuna göre hata yapma olasılığı olduğunu gösterir.

OMP algoritmasının bir başka dezavantajı ise problemi çözerken ortaya çıkmaktadır. Algoritmayı gerçeklemek için sinyalin seyreklik seviyesi olan k değerinin de bilinmesi gerekir. Çünkü tekrar sayısının sonlanması gereken yerin bilinmesi gerekmektedir. Bilinmemesi durumunda algoritma sinyalin orjinalinde bulunmayan sıfırdan farklı eleman ararken yanlış yapabilmektedir.

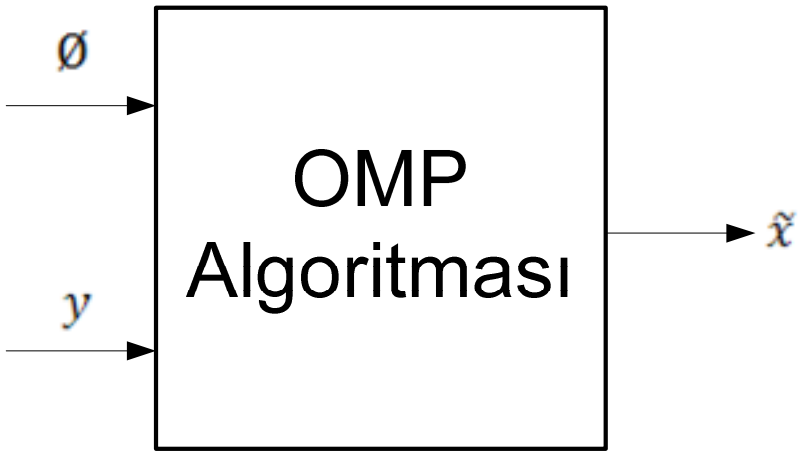
OMP algoritmasını MP algoritması ile başka bir yönden de karşılaştırılması gerekir. QR veya Cholesky faktörizasyonu yöntemiyle matris tersi alma işlemi OMP yönteminde daha zorlu bir işlem olduğu söylenebilir. Fakat algoritmanın her tekrarda yeni bir eleman seçmesi, o ana kadar seçilen elemanlar setinin minimum hata ile seçilmiş olduğunu garantiler. QR ve Cholesky faktörizasyonu gibi matris tersi alma işlemleri depolama kapasitesine ihtiyaç duyarlar. Küçük ölçekli problemler için çok önemli bir yük getirmese de büyük ölçekli problemlerde büyük bir sorun teşkil eder. Bazı durumlarda A matrisi depolanamayacak kadar büyük olabilir. Bu soruna çözüm bulmak için hızlı ve yaklaşık değerler bulan OMP algoritması varyasyonları geliştirmek üzere çalışmalar bulunmaktadır (Blumensath ve Davies 2007).

Son olarak gürültülü verilere karşı performansı, gürültü çok yüksek olmadıkça BP gibi algoritmalarla aynı seviyededir. Fakat gürültü seviyesi çok yüksek değerlere ulaşırsa OMP yönteminin performansı, yöntemin istikrarsız olmasından dolayı, kötü bir hal almaktadır (Tropp ve Needell 2008).

OMP algoritması, diğer sıkıştırıcı algılama algoritmalarına göre yüksek hızı ve sadeliği yönünden çok daha iyidir. Diğer algoritmalara kıyasla yüksek verimli hesaplamalar yapabilmektedir.

2.3 OMP Algoritmasının Yeniden Formülasyonu

OMP algoritması bölüm 2.2.2'de detaylı bir şekilde anlatılmıştır. Bu bölümde algoritmanın bir donanım üzerinde uygulanabilmesi için yeniden formülasyonunun akış basamakları anlatılacaktır. Bu tezde OMP algoritmasının kullanılmasının amacı yüksek hızı ve sadeliği açısından donanım üzerinde gerçekleştirmeye en elverişli algoritma olmasıdır. Şekil 2.9'da OMP algoritmasının basit bir bloğu görülmektedir.



Şekil 2.9 OMP algoritması blok diyagramı

x sinyali m seyrek bir sinyal ise, y ölçüm vektörüne etkisi bulunan ϕ örnekleme matrisinin m sayıda kolonunu bulmamız gerekmektedir. Algoritmanın her tekrarında y ölçüm vektörünün kalan kısmı ile en çok ilişkili örnekleme matrisi kolonu seçilir. MGS (Modified Gram Schmidt) yöntemi kullanılarak kolonun y ölçüm vektörüne katkısı hesaplanır ve y vektöründen çıkartılır. Bir sonraki tekrarda yukarıdaki anlatılanlar kalan vektörü kullanılarak tekrar yapılır. y ölçüm vektörüne etkisi bulunan ϕ örnekleme matrisinin m sayıda kolonu bulunduktan sonra orjinal sinyal olan x vektörünün yaklaşık

değeri, aşırı tanımlı en küçük kareler denklemi çözülerek bulunur. OMP yeniden oluşturma algoritmasının işleyişi aşağıda adım adım açıklanmıştır:

a. Başlangıç aşamasında kalan vektörü $R_0 = y$, indis seti $S = \emptyset$, ve tekrar sayacı $t = 1$ değerlerini alır.

b. Basit optimizasyon problemini çözen s_t değeri bulunur. Denklemdeki ϕ_j , örnekleme matrisi olan ϕ matrisinin *j*. kolonunu ifade eder.

$$s_t = arg\ max_{j=1,\dots,N} \left| \langle R_{t-1}, \phi_j \rangle \right| \tag{2.22}$$

c. İndis setine b adımında bulunan s_t değeri yazılır.

$$S_t = S_{t-1} \cup \{s_t\} \tag{2.23}$$

d. ϕ matrisinin s_t. kolonunu kullanılarak MGS yöntemi uygulanır ve q değerleri bulunur.

e. Aşağıdaki denklem kullanılarak yeni kalan vektörü bulunur.

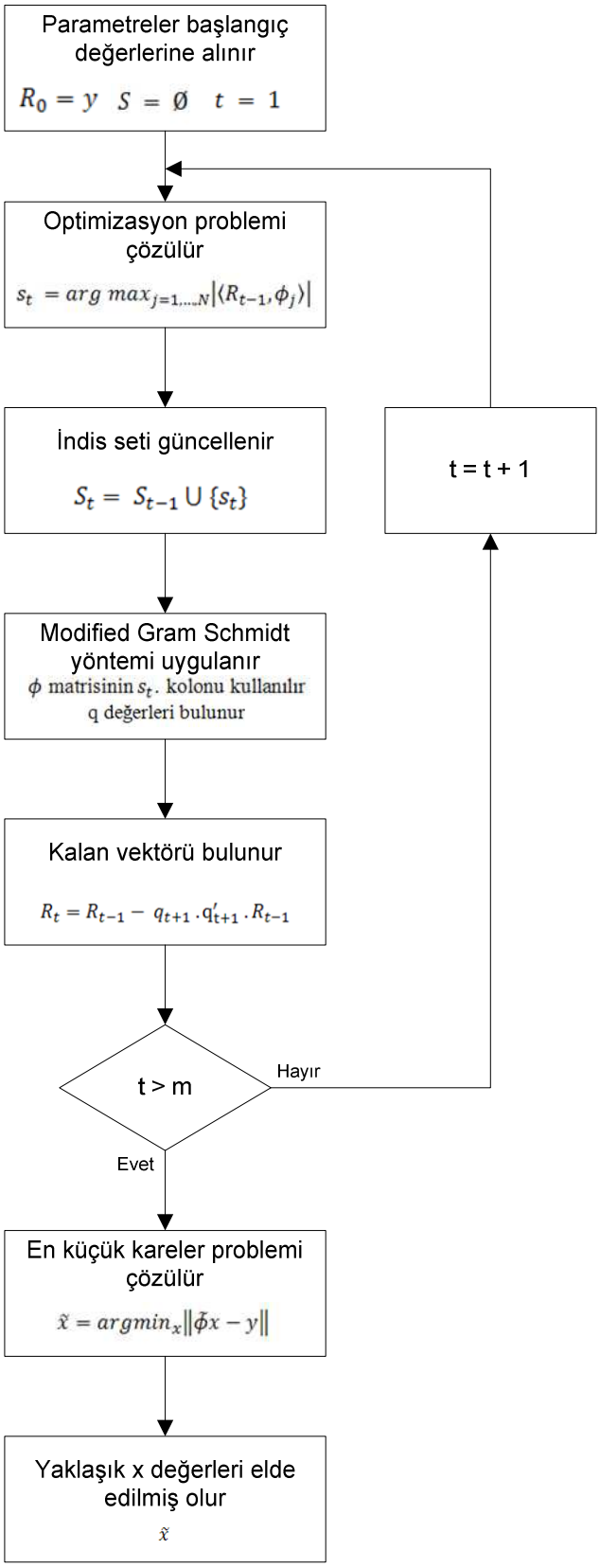
$$R_t = R_{t-1} - q_{t+1} \cdot \mathrm{q}'_{t+1} \cdot R_{t-1} \tag{2.24}$$

f. Tekrar sayacı *t*, seyreklik seviyesi *m*'den küçük ise *t* tekrar sayacı 1 artırılır ve b adımına dönülür. Büyük ise g adımına geçilir.

g. *S* indis setinde bulunan s_t indislerine karşılık gelen ϕ matrisini kolonları yardımı ile aşağıdaki en küçük kareler problemi çözülür. Bu denklemdeki $\tilde{\phi}$, örnekleme matrisinin *m* sayıdaki kolonlarından oluşmaktadır.

$$\tilde{x} = argmin_x \left\| \tilde{\phi} x - y \right\| \tag{2.25}$$

OMP yeniden oluşturma algoritmasının akış diyagramı Şekil 2.10'da verilmiştir.



Şekil 2.10 OMP yeniden oluşturma algoritmasının blok diyagramı

Algoritmanın a adımında kalan vektörü olarak adlandırdığımız $R_t$vektörüne giriş olarak *y* ölçüm vektörü verilir.

$$R_0 = y \tag{2.26}$$

R_t vektörünü kalan vektörü olarak adlandırılmasının sebebi algoritmanın tekrarlarında *y* ölçüm vektörüne etkisi bulunan ϕ örnekleme matrisinin s_t. kolonu R_t vektöründen çıkarılmasıdır. Çıkarma işlemi sonucu elde ettiğimiz kalan vektörü R_t vektörüne giriş olarak verilerek algoritmanın tekrar etmesi sağlanır. İndis seti olarak adlandırdığımız *S* vektörü başlangıç adımında boş kümedir.

$$S = \emptyset \tag{2.27}$$

b adımındaki optimizasyon problemi sonucu bulunan s_t değerleri her iterasyonda ϕ örnekleme matrisinin bir kolonunun indisini vererek bu vektörü dolduracaktır. Tekrar sayacı olarak adlandırdığımız *t* değeri algoritmanın başlangıcında 1 değerini alacaktır:

$$t = 1 \tag{2.28}$$

Algoritmanın her tekrarında bu değer 1 artırılacaktır. Bu sayacın en son aldığı değer, orjinal *x* sinyalinin *m* seyreklik sayısına eşit olmalıdır. Parametreler başlangıç değerlerine alındıktan sonra b adımına geçilir.

Algoritmanın b adımında basit bir optimizasyon probleminin çözülmesi gerekir:

$$s_t = arg\ max_{j=1,\dots,N} \left| \langle R_{t-1}, \phi_j \rangle \right| \tag{2.29}$$

Algoritmanın *t*. adımına göre atanan R_{t-1} vektörü ile ϕ örnekleme matrisinin 128 kolonu tek tek iç çarpım işlemine sokulur. Bu iç çarpım işlemi sonuçları arasındaki en büyük değer seçilir. En büyük çarpım değerini oluşturan *j* değeri bu problemin çözümüdür ve bu değer s_t değişkenine atanır. Bu adımın amacı şu şekilde açıklanabilir. Orjinal sinyal olan *x* sinyali *m* seyrek bir sinyal olduğuna göre *m* sayıda sıfır olmayan

katsayısı vardır. x sinyali yeniden oluşturma algoritmasına başlamadan önce ϕ örnekleme matrisi ile çarpılmış ve y ölçüm vektörü elde edilmişti. x sinyalinin sıfır olmayan katsayısı ile matris çarpımı işleminde bu sıfır olmayan katsayı ile çarpılacak ϕ örnekleme matrisinin kolonunun çarpımı sonucu y vektörününün ilgili satırında büyük bir değer olarak yer alır. Sıfır olan katsayıların ise matris çarpımı işleminde bu sıfır olan katsayı ile çarpılacak ϕ örnekleme matrisinin kolonunun çarpımı sonucu y vektörünün ilgili satırında 0 olarak yer alır. Dolayısıyla denklem 2.29'daki optimizasyon probleminde y vektörü ile ϕ örnekleme matrisinin j. kolonunu iç çarpımını en yüksek yapan j değeri, x vektörünün sıfır olmayan katsayısının satır numarasını verir.

Algoritmanın c adımında, b adımında bulunan s_t değerleri indis seti vektörüne eklenecektir. İndis seti vektörü algoritmanın ilerleyen adımlarında kullanılacaktır.

Algoritmanın d adımında ϕ örnekleme matrisinin j. kolonunun QR açılımı ile q değerleri bulunacaktır. QR açılımı yöntemi olarak MGS algoritması kullanılmıştır. MGS algoritması ile q değerleri aşağıdaki verilen denklemler yardımıyla hesaplanır.

MGS algoritması ile q değerlerinin hesaplanması:

Adım 1: Ortogonal vektörler hesaplanır. $\{ z_1, z_2, \dots, z_m \}$

$\phi_{s_t} = x_1$, $\phi_{s_{t+1}} = x_2$, ... , $\phi_{s_{t+m}} = x_k$ olmak üzere,

$$z_1 = x_1$$

$$z_2 = x_2 - proj_{z_1}(x_2) = x_2 - \frac{\langle x_2, z_1 \rangle}{\langle z_1, z_1 \rangle} z_1$$

$$z_3 = x_3 - proj_{z_1}(x_3) - proj_{z_2}(x_3) = x_3 - \frac{\langle x_3, z_1 \rangle}{\langle z_1, z_1 \rangle} z_1 - \frac{\langle x_3, z_2 \rangle}{\langle z_2, z_2 \rangle} z_2$$

$$z_k = x_k - \ proj_{z_1}(x_k) - \cdots - proj_{z_{k-1}}(x_k)$$

$$= \ x_k - \frac{\langle x_k, z_1 \rangle}{\langle z_1, z_1 \rangle} z_1 - \cdots - \frac{\langle x_k, z_{k-1} \rangle}{\langle z_{k-1}, z_{k-1} \rangle} z_{k-1}$$

Adım 2 : Vektörler normalize edilir.

$$q_1 = \frac{z_1}{\|z_1\|}, \quad q_2 = \frac{z_2}{\|z_2\|}, \quad \ldots \quad , q_k = \frac{z_k}{\|z_k\|}$$

Elde edilen q değerleri e adımında kullanılarak kalan vektörlerinin hesaplanması sağlanacaktır.

Algoritmanın e adımında, d adımında elde ettiğimiz q değerleri ve bir önceki algoritma tekrarının e adımında oluşturulan artık vektör kullanılarak denklem 2.30 ile yeni artık vektör hesaplanır.

$$R_t = R_{t-1} - \ q_{t+1} \cdot \mathrm{q}'_{t+1} \cdot R_{t-1} \tag{2.30}$$

Bu denklem ile algoritmanın b adımında bulunan s_t kolonlarının R_t matrisine katkısı çıkarılmaktadır. Dolayısıyla yeni oluşturulan R_t vektörü ile algoritma bir sonraki tekrarında b adımına girdiğinde, yeni bir $s_t$değeri bulacaktır.

Algoritmanın f adımında tekrar sayacı *t*, seyreklik seviyesi *m*'den küçük ise tekrar sayacı 1 artırılır ve b adımına dönülür. Tekrar sayacı *m* değerine eşit veya büyük ise g adımına geçilir.

Algoritmanın g adımında *S* vektöründeki değerlere karşılık gelen ϕ matrisini kolonları ile orjinal sinyal *x*'in yaklaşık değerleri aşağıdaki en küçük kareler problemi ile bulunacaktır.

$$\tilde{x} = argmin_x \|\tilde{\phi} x - y\| \tag{2.31}$$

Bu problem genelikle Moore–Penrose sözde ters alma yöntemi kullanılarak çözülmektedir. Yöntem denklemlerle aşağıda gösterilmektedir.

$$\tilde{\phi}^{\dagger} = (\tilde{\phi}^{\mathrm{T}}\tilde{\phi})^{-1}\,\tilde{\phi}^{\mathrm{T}} \tag{2.32}$$

Eğer $(\tilde{\phi}^{\mathrm{T}}\tilde{\phi})^{-1}$ ifadesine *C* dersek;

$$\tilde{\phi}^{\dagger} = \mathrm{C}^{-1}\,\tilde{\phi}^{\mathrm{T}} \tag{2.33}$$

Denklem 2.33'ü çözebilmek için *C* matrisinin tersini almamız gerekir. *C* matrisi 5x5'lik bir matristir. Aynı zamanda *C* matrisi SPD (Simetrik ve Pozitif Tanımlı – Symmetric and Positive Definite) bir matristir. SPD matrislerin tersini alma işleminde genellikle CD (Cholesky Ayrıştırması - Cholesky Decomposition) kullanılır. Fakat bu yöntem kök alma işlemi içerdiği ve kök alma işlemi hem zaman hemde alan açısından verimsiz olduğu için Cholesky Decomposition yönteminin alternatif bir formu kullanılacaktır. ACD (Alternatif Cholesky Ayrıştırması - Alternative Cholesky Decomposition) yöntemi olarak bilinen bu yöntem kök alma işlemi içermemektedir. CD yöntemi matrisi denklem 2.34'deki gibi alt üçgensel matris ve bu matrisin transpozu şeklinde ikiye ayırır.

$$C = LL^{\mathrm{T}} \tag{2.34}$$

L matrisinin elemanlarının bulunmasında ise denklem 2.35 uygulanır.

$$L_{i,j} = \frac{1}{L_{j,j}}\,(C_{i,j} - \sum_{k=1}^{j-1} L_{i,k}\,L_{j,k}),\ i > j\ ise \tag{2.35}$$

CD yöntemine tercih edilecek ACD yönteminde ise denklem 2.36'da görüldüğü üzere matrisi köşegenlerinde 1 değeri bulunan alt ügensel matris, köşegen matris ve alt üçgensel matrisin transpozu şeklinde üçe ayırır.

$$C = LDL^{\mathrm{T}} \tag{2.36}$$

Bu yöntemdeki L ve D matrislerinin elemanlarının bulunmasında ise denklem 2.37 ve denklem 2.38 uygulanır.

$$L_{i,j} = \frac{1}{D_{j,j}} \left(C_{i,j} - \sum_{k=1}^{j-1} L_{i,k}\, L_{j,k}\, D_{k,k}\right),\ i > j\ ise \tag{2.37}$$

$$D_{i,i} = C_{i,j} - \sum_{k=1}^{i-1} {L_{i,k}}^2\, D_{k,k} \tag{2.38}$$

Denklem 2.37 ve 2.38'nin bağımlılık grafikleri Şekil 2.11'de verilmiştir.

C matrisinin tersini alma işlemi ACD yöntemi aracılığı ile denklem 2.39'daki gibi yazılabilir.

$$C^{-1} = (L^{\mathrm{T}})^{-1} D^{-1} L^{-1} \tag{2.39}$$

Bu denklem sayesinde C matrisinin tersini alma işlemi L matrisinin tersini alma ve D matrisinin tersini alma işlemine dönüşmüştür. Denklem 2.40'daki matris özelliğinden dolayı L^{T} matrisinin ayrıca tersini almamıza gerek yoktur. L matrisinin tersine alma işlemi sonucu bulduğumuz L^{-1} matrisinin transpozunu alabiliriz.

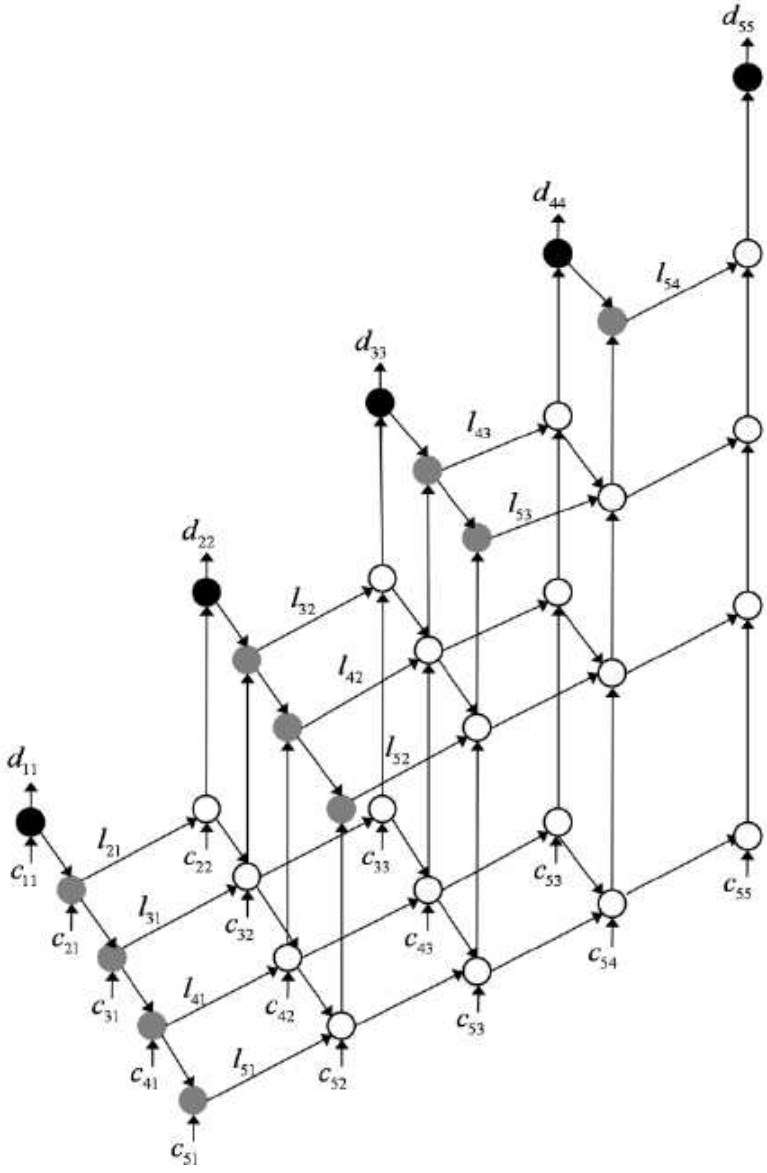
$$(L^{\mathrm{T}})^{-1} = (L^{-1})^{T} \tag{2.40}$$

D matrisi köşegen matris olduğu için matrisin tersi, 1 sayısının matrisin elemanlarına bölünmesiyle bulunur. Bu işlemi denklem 2.37'de yaptığımız için D matrisinin tersi bulunmuştur. Dolayısıyla C^{-1} matrisini bulmak için yapmamız gereken tek şey L^{-1} matrisini bulmaktır. L^{-1} matrisini bulmak için uygulamamız gereken prosedür aşağıda verilmiştir.

for j = 1 to n **do**
  $L_{inv}[j,j] = 1/L[j,j]$
  for i = j + 1 to n **do**
    $L_{inv}[i,j] = -L[i,j:(i-1)]\ L_{inv}[j:(i-1),j]/L[j,j]$
  end for
end for

Bir alt üçgensel matrisin tersi yine bir alt üçgensel matristir. Bu bilgiyi de dikkate alarak işlemlerde matrisin üst üçgensel kısmının indisleri kullanılmaz.

C^{-1} matrisini bulmak yerine denklem 2.39 çözülerek C^{-1} matrisinin değerine ulaşılır. Denklem 2.33'de C^{-1} değeri yerine konularak bulunan sonuç, y matrisi ile çarpılırsa x sinyalinin yaklaşık değer matrisi ($\tilde{x}$) bulunur.



Şekil 2.11 ACD yöntemi bağımlılık grafiği

3. MATERYAL VE YÖNTEM

3.1 Materyal

3.1.1 FPGA

FPGA (Sahada Programlanabilir Kapı Dizileri - Field Programmable Gate Array) programlanabilir mantık blokları ve bu bloklar arasındaki ara bağlantılardan oluşan sayısal tümleşik devrelerdir. FPGA aygıtları üretim sonrasında müşteri veya tasarımcı tarafından ihtiyaç duyulan mantık işlevlerini gerçekleştirme amacına yönelik olarak tasarlanmışlardır. Dolayısıyla FPGA aygıtlarının içeriği kullanıcı tarafından düzenlenebilmektedir. Birçok FPGA aygıtı programlanabilir mantık blokları ve ara birimlere ek olarak hafıza birimlerini de bulundurur. Sahada programlanabilir ismi verilmesinin nedeni, mantık bloklarının ve ara bağlantılarının imalat sürecinden sonra sahada yani müşteri tarafından programlanabilir olmasıdır.

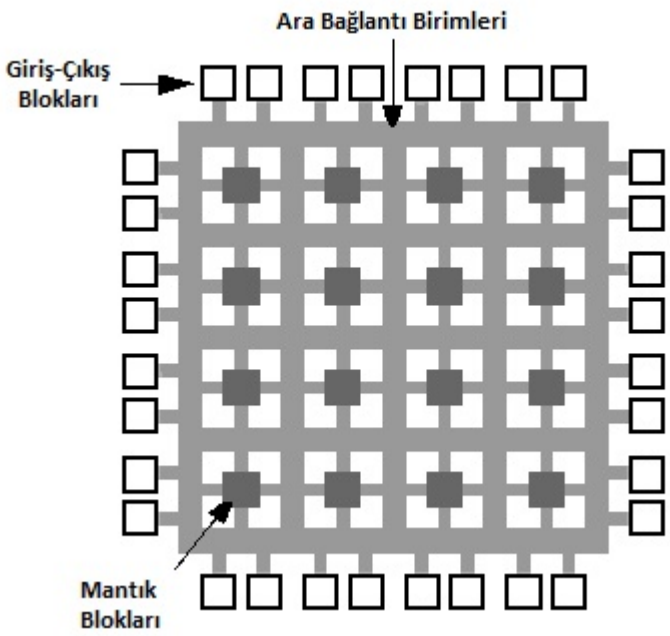
FPGA aygıtları ASIC (Application Specific Integrated Circuit – Uygulamaya Özel Tümleşik Devre) tasarımlara oranla daha yavaş çalışmalarına rağmen tekrar programlanabilme özelliği dolayısıyla daha fazla tercih edilmektedir.

FPGA aygıtları genellikle HDL (Donanım Tanımlama Dili - Hardware Description Language) olarak adlandırılan dillerle yapılandırılırlar. Yazılım programlama dillerinde işlemler ardışıl yapılırken, HDL dillerinde işlemler paralel olarak yapılmaktadır. Bu alanda en yaygın olarak kullanılan diller Verilog ve VHDL (VHSIC Donanım Tanımlama Dili - VHSIC Hardware Description Language) donanım tanımlama dilleridir.

3.1.1.1 FPGA'ların yapısı

FPGA aygıtları programlanabilir mantık blokları, bu blok dizisini çevreleyen giriş-çıkış blokları ve ara bağlantılar olmak üzere düzenlenebilir üç ana bölümden oluşur. Programlanabilir mantık blokları, ara bağlantılar içerisine gömülü olarak bulunur.

Programlanabilir mantık bloklarının yapılandırılması ve bu bloklar arasındaki iletişim ara bağlantılar sayesinde gerçekleşir. Giriş-çıkış blokları, ara bağlantılar ile bütünleşmiş devrenin paket bacakları arasındaki ilişkiyi sağlar. Şekil 3.1'de bir FPGA'nın yapısı görülmektedir.

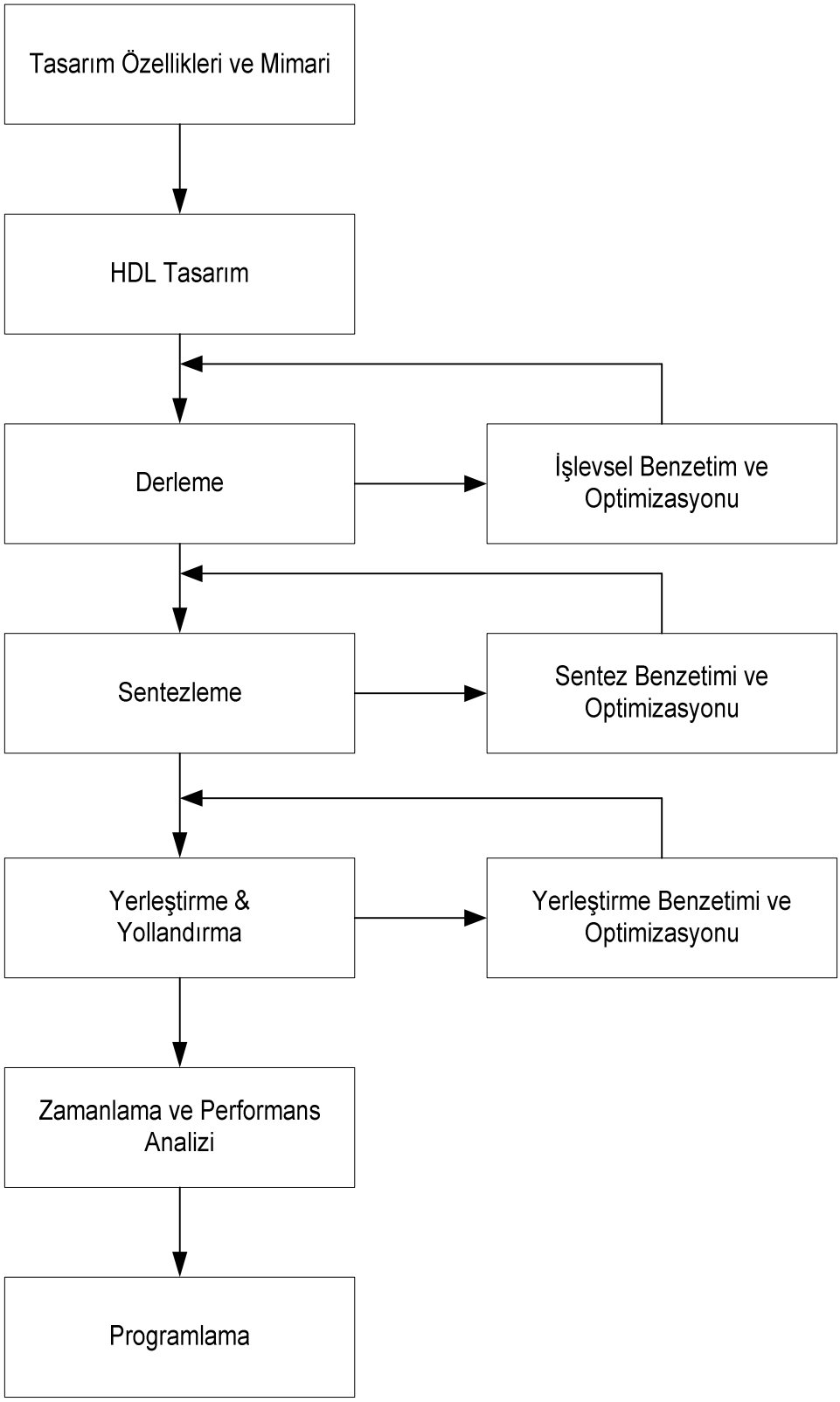


Şekil 3.1 FPGA'nın yapısı

3.1.1.2 Tasarım süreci akışı

FPGA'ların programlanmasına giden tasarım akışında ilk olarak tasarlanacak sistemin tasarım özellikleri ve mimarisi verilmelidir. Daha sonra yüksek seviyeli HDL kullanılarak sistemin tasarımı yapılır. HDL dilleri sadece modelleme amaçlı geliştirildiği için, HDL ile tanımlanan her sistem sentezlenebilir değildir. Devrede ne şekilde işleyeceği düşünülerek yazılan bir kodun tasarımın ileriki aşamalarında büyük kolaylık sağlayacağı bilinmelidir. Tasarım sonrası derleme işlemine geçilir. Derleme işleminde tasarıma ait standart bağlantılar oluşturulur. Derleme işleminden sonra yapılan tasarımın istenilen özellikleri yerine getirip getirmediği işlevsel benzetim yapılarak test edilir. Benzetim sonucuna göre, tasarımda değişiklikler yapılarak istenen sonuç elde edilene kadar tekrara devam edilir. İşlevsel yeterlilik sağlandığı takdirde

sentezleme adımına geçilir. Bu adımda HDL kodlarının yanı sıra kullanılacak FPGA, asgari çalışma frekansı, azami sinyal gecikmesi gibi tasarıma ait ek bilgilere de ihtiyaç duyulmaktadır. Bunlar kullanıcı kısıtlama dosyası içinde sisteme girdi olarak verilir. Sentezleyici istenilen fonksiyonların en iyi şekilde gerçeklenebilmesi için gerekli optimizasyonları yaptıktan sonra elde edilen lojik fonksiyonların FPGA içerisindeki lojik bloklara eşleştirilmesi işlemi yapılarak kapı seviyesinde bir bağlantı listesi oluşturulur. Sentezleyici eşleştirme işlemi sırasında kullanıcı kısıtlama dosyasını da kullanılarak kullanıcının verdiği kısıtlar içerisinde kalmaya çalışır. Sentezleme işleminden sonra sentez benzetimi ve optimizasyonu yapılır. Sentez benzetimi adımında iştenilen sonuçlara ulaşılırsa yerleştime ve yollandırma adımına geçilir. Bu adımda devre fonksiyonları ile eşleştirilmiş lojik bloklar arasındaki bağlantılar oluşturulur. Birbiriyle ilişkili lojik bloklar, lojik bloklar arasındaki yolların daha kısa olması için birbirine yakın yerleştirilir. Bu aşamadan sonra yerleştirme benzetimi yapılır. Bu benzetimde yollandırma bağlantılarına ait gecikmelerde hesaplandığı için gerçek sistemin performansını neredeyse tam olarak göstermektedir. İstenilen performans seviyesine ulaşıldığı takdirde programlama adımına geçilir. FPGA'nın programlanması aşamasında kullanılacak bit dizisi üreticinin sağladığı yazılımla elde edilir. FPGA'nın uygun donanım kullanılarak programlanmasıyla tasarım tamamlanmış olur. Programlanması tamamlanan FPGA üzerinde tasarımın doğruluğunu gösteren testler yapılır ve sistem donanım üzerinde gerçeklenmiş olur. Şekil 3.2'de tasarım sürecine ilişkin akış diyagramı verilmiştir.



Şekil 3.2 HDL tasarım akış diyagramı

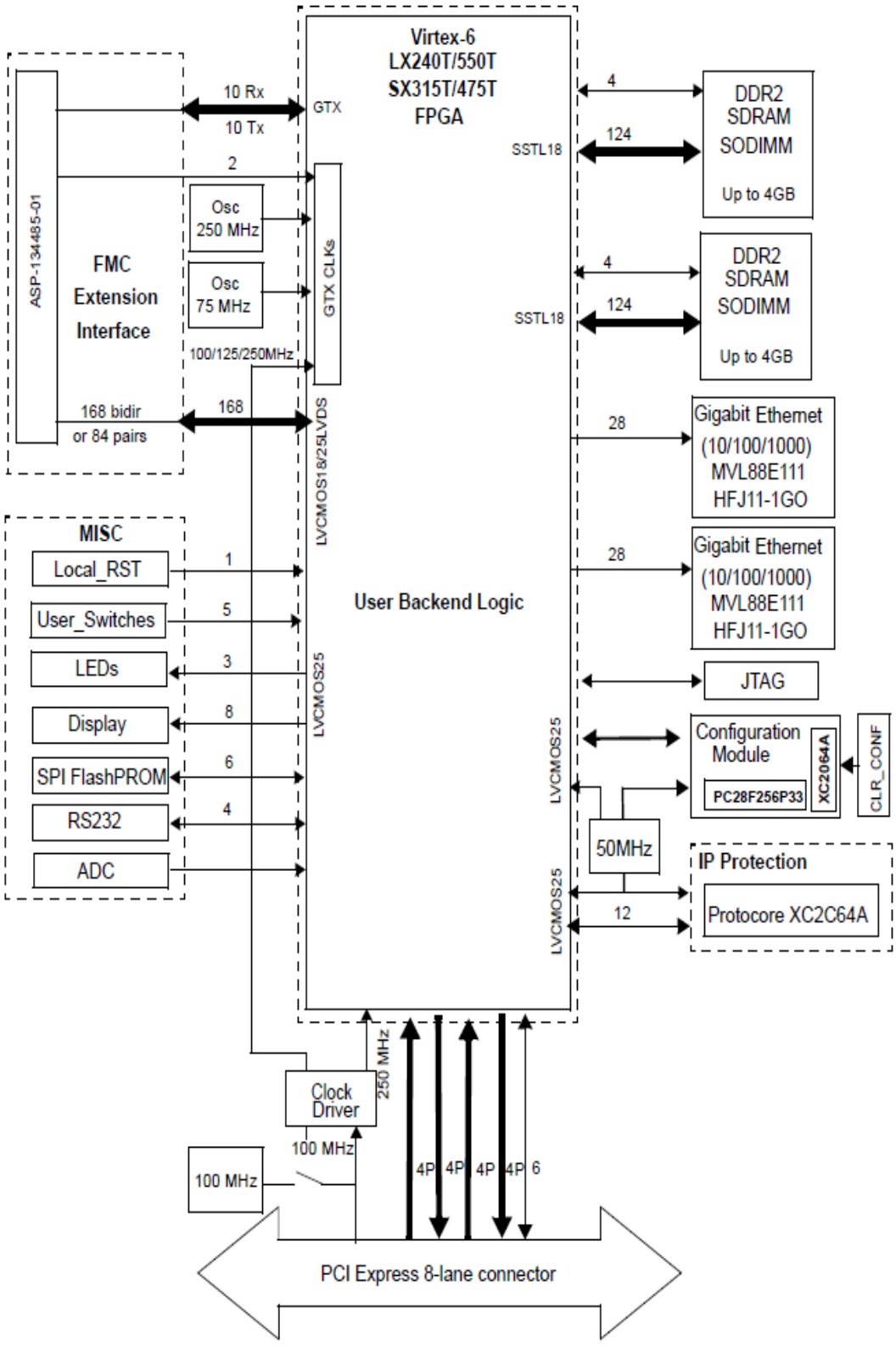
3.1.2 Uygulama geliştirme kartı

Bu çalışmada PLDA firmasının ürettiği XpressV6-550 uygulama geliştirme kartı kullanılmıştır. Bu kart üzerinde

- 550.000 lojik işlem birimine sahip Xilinx Virtex-6 XC6VLX550T-2FFG1759C FPGA tümdevresi,
- PCI Express® 2.0 x1, x4, x8 protokolü,
- 2 x 4 GByte DDR2 SDRAM,
- 1 MB SPI FlashPROM,
- 2 Gigabit Ethernet arayüzü,
- 160 Giriş-Çıkış pinleri,
- 3 adet LED, 1 adet 7 parçalı gösterge, 1 adet RS232 seri link, 2 adet RST düğmesi bulunmaktadır.

Xpress V6-550 uygulama geliştirme kartı Şekil 3.3'de, kartın blok şeması ise Şekil 3.4'de verilmiştir.

Şekil 3.3 Xpress V6-550 uygulama geliştirme kartı



Şekil 3.4 Xpress V6-550 uygulama geliştirme kartı blok şeması

3.1.2.1 Xilinx Virtex-6

Kullanılan geliştirme kartı üzerinde bulunan FPGA, Xilinx firması tarafından üretilmiş Virtex-6 XC6VLX550T-2FFG1759C'dir. Virtex-6 yapısındaki ana lojik bloğuna CLB (Yapılandırılabilir Lojik Blok - Configurable Logic Block) denir. Her CLB iki dilim (slice)'den oluşur. Bir dilim ise 6 girişli LUT (Değer Tablosu - Look-Up Table) ve 8 adet D-flip-flop'tan oluşur.

40 nm üretim teknolojisine sahip olan Virtex-6 LX550T FPGA üzerinde 549.888 standart lojik hücreye karşılık gelen 85.920 dilim bulunmaktadır. Bu lojik birimlerinin dışında 22.752 Kb blok ram, 18 adet MMCM (Karışık Mod Saat Yöneticisi – Mixed Mode Clock Manager), 576 adet DSP (Sayısal İşaret İşleme - Digital Signal Processing) bloğu bulunmaktadır. Üreticinin verdiği bilgilere göre bu FPGA tümdevresi 600 MHz ve üzeri hızlarda çalışabilmektedir.

Kullanıcıya ait 1200 adet tek yönlü giriş-çıkış pini ve 600 adet diferansiyel giriş-çıkış pin çifti bulunmaktadır.

3.1.3 Diğer materyaller

Bir firmanın ürettiği FPGA'nın üzerinde tasarım adımlarını (derleme, sentezleme, yerleştirme ve yollandırma) gerçekleştirebilmek için yine aynı firmanın "tümleşik tasarım/geliştirme ortamı" olarak adlandırılan yazılımlarının kullanılması gerekir. Bu çalışmada Xilinx firması tarafından üretilen Xilinx ISE 13.2 yazılımı kullanılmıştır.

Tasarım adımları sonrasında yapılan benzetimler için FPGA üreticisinin geliştirdiği tasarım/geliştirme ortamına entegre edilebilen bir benzetim programı kullanılması gerekir. Bu çalışmada benzetim programı olarak Modelsim PE 10.0 kullanılmıştır.

Benzetim sonuçlarının gerçekten doğru olup olmadığını saptamak amacıyla bu çalışma bir yazılım ortamında gerçeklenmiştir. HDL ile kodlanıp benzetim programıyla elde edilen sonuçlar, daha önce bilgisayar ortamında yazılım programı ile elde edilen

sonuçlar ile karşılaştırılmıştır. Çalışmanın bilgisayar ortamında gerçeklenmesi için MATLAB R2008a yazılımı kullanılmıştır.

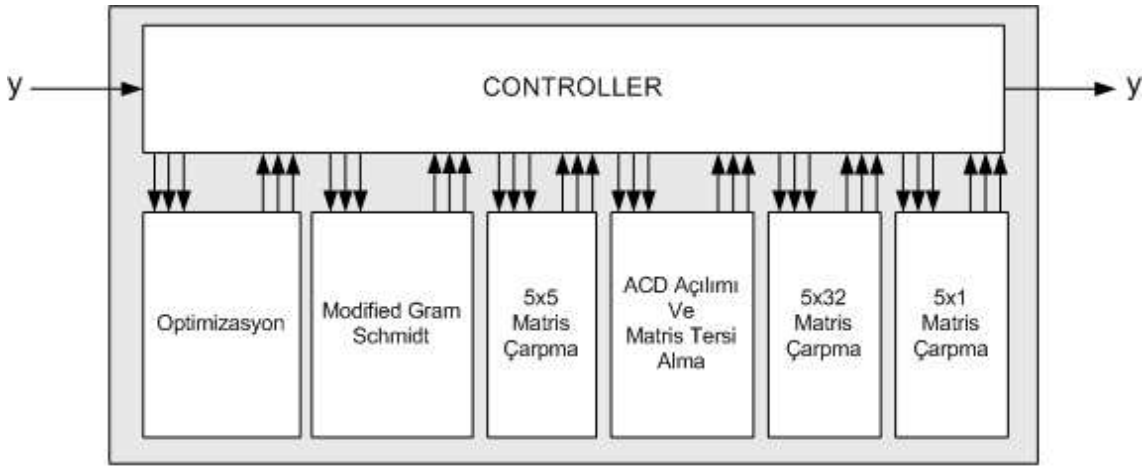
3.2 Yöntem

3.2.1 FPGA gerçekleştirimi

Tezin bu bölümüne kadar gerçekleştirimi yapılacak algoritma ve hedeflenen gerçekleştirim platformu olan FPGA'lar hakkında temel bilgiler verilmiştir. Bu bölümde ise, gerçekleştirimin nasıl yapıldığına ilişkin detaylar sunulmaktadır.

3.2.1.1 Sistemin genel donanım yapısı

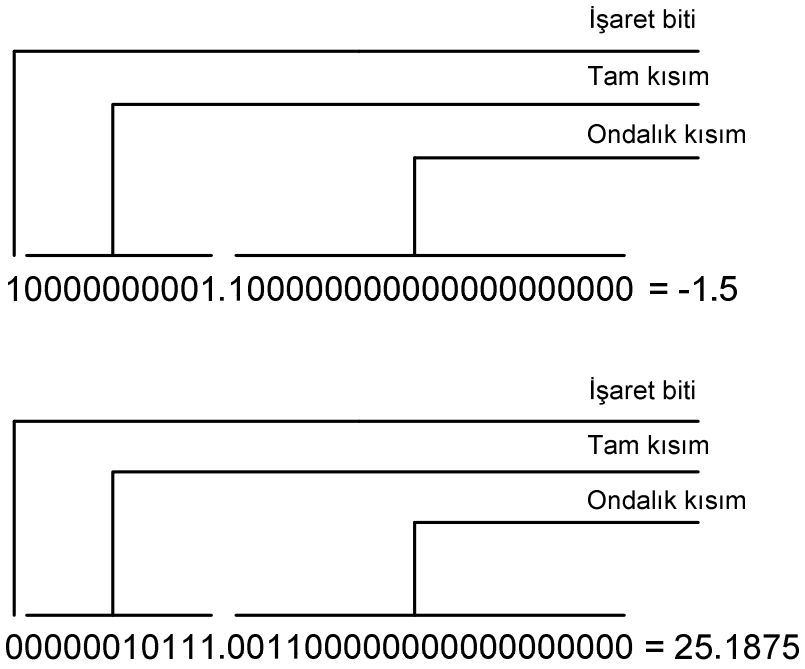
Sistemin genel blok diyagramı Şekil 3.5'de gösterilmiştir. Algoritmanın bütün blokları başlıklar altında tek tek anlatılacaktır.



Şekil 3.5 OMP algoritması sisteminin genel blok diyagramı

Algoritmanın giriş pinine verilen y matrisi ve start sinyali ile algoritma çalışmaya başlar. Algoritma sonunda done sinyali ile birlikte $\tilde{x}$ matrisi elde edilir. Algoritmada kullanılan sayılar gerçel sayılardır. Bir sayıyı ifade edebilmek için donanımda 32 bit kullanılmıştır. Şekil 3.6'da gösterildiği gibi, otuz iki bitlik sayıların en soldaki biti işaret

biti olarak kullanılmaktadır. En soldaki bit 1 ise söz konusu sayı negatif, 0 ise pozitiftir. Bu bitten sonraki on bit ile sayının tam kısmı gösterilir. Noktadan sonraki yirmi bir bit ise, sayının ondalık kısmını göstermek amacıyla kullanılır.

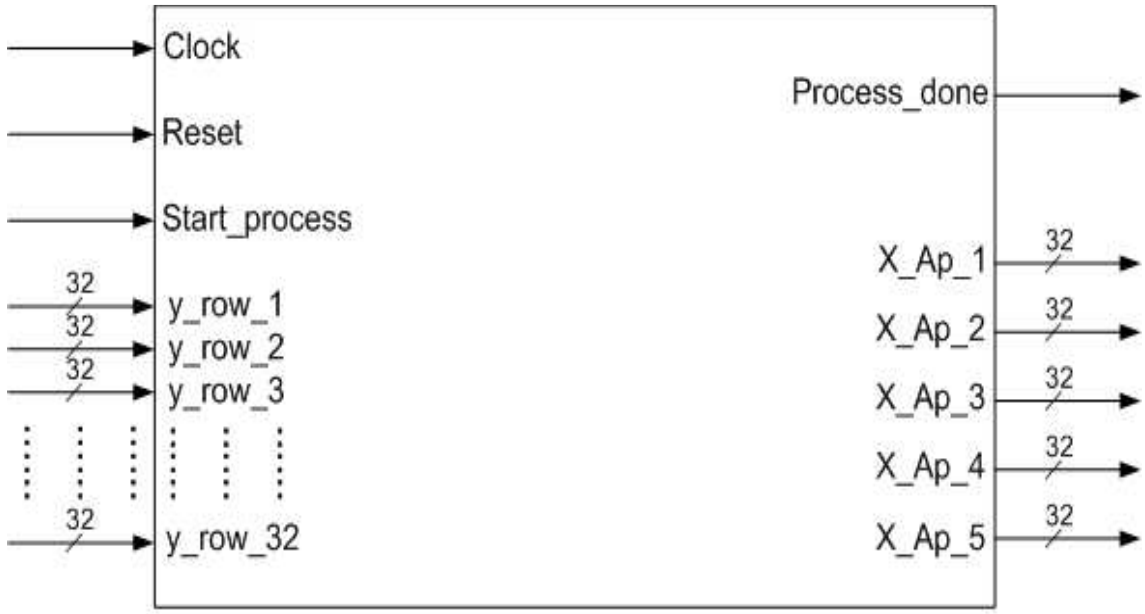


Şekil 3.6 Sayıların donanımda ifade edilmesi

3.2.1.2 Kontrol (Controller) modülü

Controller modülü bütün sistemi kontrol eden, sistemdeki bütün modüllere sırasıyla başlama sinyalini gönderen, modüller işlemlerini bitirdikten sonra ürettikleri verileri toplayıp başka modüllere giriş olarak veren, algoritmanın tekrar sayısını hesaplayan, kısacası algoritmayı yöneten bir modüldür. Öncelikle OMP algoritması ile yeniden oluşturma yapacağımız y ölçüm vektörünü giriş olarak alır. Sistem dışından Start_process girişine verilen 1 değeri algoritmayı başlatır. Başlaması gerektiğini anlatan sinyali aldıktan sonra optimizasyon modülüne y ölçüm vektörünü giriş olarak verir ve başla komutunu iletir. Controller modülünün verdiği giriş değerleriyle işleme başlayan optimizasyon modülü, işlemini bitirdiği zaman bu bilgiyi Controller modülüne iletir. Optimizasyon modülünün işlemini bitirdiğini anlayan Controller modülü bu

modülden aldığı verileri MGS modülüne verir ve bu modüle başla komutunu verir. MGS modülü işlemini bitirdiği zaman Controller modülüne işlemi bitirdiğini iletir. MGS modülünün işlemi bitirdiğini öğrenen Controller modülü bu modülden aldığı verileri bazı işlemlerden geçirdikten sonra yeniden optimizasyon modülüne verir ve optimizasyon modülününe başla komutu verir. Controller modülü bu döngünün orjinal sinyalin seyreklik derecesi olan m kere tekrar edilmesini sağlar. Döngü *m* kere tekrar edildikten sonra gerekli verileri 5x5 matris çarpma modülüne verir. Bu modül işlemlerini bitirdikten sonra 5x5 matris çarpma modülünden aldığı verileri matris tersini alma modülüne verir ve bu modüle başlama komutunu verir. Matris tersini alma işlemi bittikten sonra bu modülden aldığı verileri 5x32 matris çarpma modülüne verir ve başlaması gerektiğini bildirir. 5x32 matris çarpma modülü de işlemlerini bitirdiğinde Controller modülüne işlemini bitirdiğini bildirir. Bu modülün de işlemi bitirdiğini öğrenen Controller modülü bu modülden aldığı verileri son işlem modülü olan 5x1 matris çarpma modülüne verir ve bu modüle işleme başla komutunu iletir. Son işlem modülü olan 5x1 matris çarpma modülü işlemi bitirdiğinde orjinal sinyal olan *x* sinyalinin yaklaşık değerlerini üretmiş olur. Bu verileri alan Controller modülü verileri sistem dışına çıkış olarak verir ve algoritmanın tamamlandığını Process_done sinyalini 1 değerine çekerek bildirir. Controller modülü yöneten konumda olduğu ve bir işlem modülünü başlatabilmesi için bir önceki işlem modülünün işlemi bitirmesini beklediğinden dolayı, işlemleri bitirme süresi verilemez. Optimizasyon modülünün giriş çıkış pinlerinin detaylı bilgileri Çizelge 3.1'de, modülün blok diyagramı ise Şekil 3.7'de verilmiştir.



Şekil 3.7 Controller modülü blok diyagramı

Çizelge 3.1 Controller modülü giriş-çıkış pinleri

Pin İsmi	Giriş/Çıkış	Bit Sayısı	Açıklama
Clock	Giriş	1 bit	Clock
Reset	Giriş	1 bit	Reset Sinyali
Start_process	Giriş	1 bit	Modülün işleme başlamasını bildiren sinyal
y_row_1 y_row_2 ..... y_row_32	Giriş Giriş ..... Giriş	32 bit 32 bit .. 32 bit	Ölçüm vektörünün 1. satır değeri Ölçüm vektörünün 2. satır değeri ..... Ölçüm vektörünün 32. satır değeri
Process_done	Çıkış	1 bit	Modülün işlemi bitirdiğini belirten sinyal
X_Ap_1 X_Ap_2 X_Ap_3 X_Ap_4 X_Ap_5	Çıkış Çıkış Çıkış Çıkış Çıkış	32 bit 32 bit 32 bit 32 bit 32 bit	x sinyalinin yaklaşık değeri 1. satırı x sinyalinin yaklaşık değeri 2. satırı x sinyalinin yaklaşık değeri 3. satırı x sinyalinin yaklaşık değeri 4. satırı x sinyalinin yaklaşık değeri 5. satırı

3.2.1.3 Çarpma ve bölme modülü

Bu modül iki adet 32 bitlik sabit noktalı sayıyı Op_a ve Op_b girişleri aracılığı ile alır. Mode girişine 1 değeri verilir ise bölme işlemi, 0 değeri verilir ise çarpma işlemi yapar. İşlem sonucunu Out çıkışından verir. Bu modül için başlamasını belirten start ve bitirdiğini belirten done giriş/çıkışları kullanılmamıştır. Bu modülden MGS modülünde 32 tane, 5x1 matris çarpma modülünde 32 tane, ACD ve matris tersi alma modülünde 10 tane kullanılmıştır. Modül yapılması gereken işlemi 1 saat darbesinde (clock cycle) gerçekleştirir. Çarpma ve bölme modülünün giriş çıkış pinlerinin detaylı bilgileri Çizelge 3.2'de, modülün blok diyagramı ise Şekil 3.8'de verilmiştir.

Şekil 3.8 Çarpma ve bölme modülü blok diyagramı

Çizelge 3.2 Çarpma ve bölme modülü giriş-çıkış pinleri

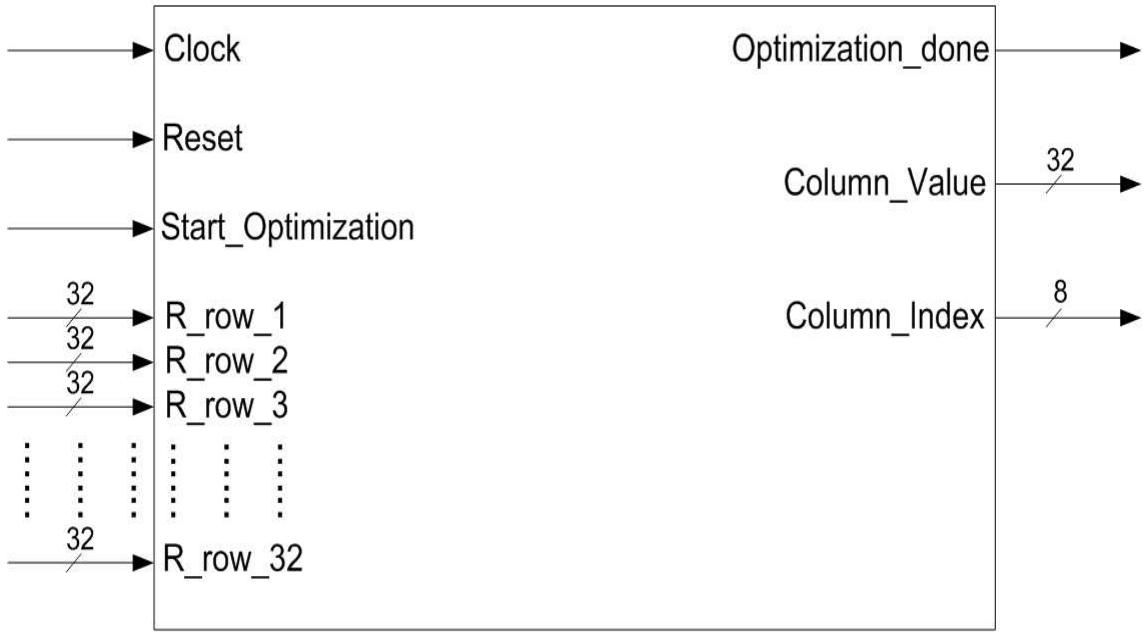
Pin İsmi	Giriş/Çıkış	Bit Sayısı	Açıklama
Op_a	Giriş	32 bit	Operatör(Bölme işlemi ise bölünen)
Op_b	Giriş	32 bit	Operatör(Bölme işlemi ise bölen)
Mode	Giriş	1 bit	Modülün çarpma veya bölme yapmasını sağlar. (0 = çarpma, 1=bölme)
Out	Çıkış	32 bit	Çarpma veya bölme işleminin sonucunu verir.

3.2.1.4 Optimizasyon modülü

Bu modül aşağıda verilen basit optimizasyon problemini çözmek için kullanılır.

$$s_t = arg\ max_{j=1,\ldots,N} \left| \langle R_{t-1}, \phi_j \rangle \right| \tag{3.1}$$

x sinyali *m* seyrek bir sinyal ise *y* ölçüm vektörüne etkisi bulunan ϕ örnekleme matrisinin *m* sayıda kolonunu bulunması gerekmektedir. Dolayısıyla algoritmanın her tekrarında *y* ölçüm vektörünün kalan kısmıyla en çok ilişkili örnekleme matrisi kolonu seçilir. Algoritma *m* sayıda tekrar edeceğinden algoritma sonucunda elimizde *m* sayıda kolon bulunur. Bu kolonların indisleri aynı zamanda orjinal sinyal olan *x* sinyalinin sıfır olmayan değerlerinin kaçıncı satırı olduğunu göstermektedir. 32 x 128 boyutlarındaki örnekleme matrisi bu modül içerisinde depolanır. Controller modülünün Start_Optimization girişini 1 değerine çekmesi ile modül işleme başlar. Denklem 3.1'de görüldüğü üzere örnekleme matrisinin 128 kolonunun herbirinin, kalan vektörü ile matris iç çarpım işlemine sokulur. Algoritmaya başlandığında kalan vektörü olarak ölçüm vektörü verilir. Algoritmanın daha sonraki tekrarlarında modülün girişine kalan vektörleri verilir. Çarpım işlemleri sonunda her kolona ait bir değer olmak üzere 128 adet değer elde edilir. Bu değerlerin en büyüğü, değerler birbiri ile karşılaştırılarak bulunur. Bulunan en büyük değer matrisin kaçıncı kolonunun değeri ise, o kolonun kaçıncı kolon olduğu bilgisi Column_Index çıkışına verilerek Controller modülüne bildirilir. Bildirilen kolonun içeriği ise Column_Value çıkışına verilerek Controller modülüne bildirilir. Modül işlemi bitirdiği zaman Optimization_done sinyalini 1 değerine çekerek işlemi bitirdiğini Controller modülüne bildirir. Optimizasyon modülü bütün bu işlemi 9 saat darbesinde (clock cycle) gerçekleştirir. Optimizasyon modülünün giriş çıkış pinlerinin detaylı bilgileri Çizelge 3.3'de, modülün blok diyagramı ise Şekil 3.9'da verilmiştir.



Şekil 3.9 Optimizasyon modülü blok diyagramı

Çizelge 3.3 Optimizasyon modülü giriş çıkış pinleri

Pin İsmi	Giriş/Çıkış	Bit Sayısı	Açıklama
Clock	Giriş	1 bit	Clock
Reset	Giriş	1 bit	Reset Sinyali
Start_Optimization	Giriş	1 bit	Modülün işleme başlamasını bildiren sinyal
R_row_1 R_row_2 ..... R_row_32	Giriş Giriş ..... Giriş	32 bit 32 bit .. 32 bit	Kalan vektörünün 1. kolon değeri Kalan vektörünün 2. kolon değeri ..... Kalan vektörünün 32. kolon değeri
Optimization_done	Çıkış	1 bit	Modülün işlemi bitirdiğini belirten sinyal
Column_Value	Çıkış	32 bit	Örnekleme matrisinin ilgili kolonunun değeri
Column_Index	Çıkış	8 bit	Örnekleme matrisinin kaçıncı kolonunun max değer olduğu bilgisi

3.2.1.5 Modified Gram Schmidt modülü

Bu modül optimizasyon modülünde yapılan işlemler sonucu bulunan y ölçüm vektörünün kalan kısmıyla en çok ilişkili örnekleme matrisi kolonuna ait q değerlerinin bulunması görevini üstlenmektedir. q değerlerinin bulumasının amacı aşağıdaki denklemi çözebilmektir.

$$R_t = R_{t-1} - q_{t+1} \cdot \mathrm{q}'_{\mathrm{t+1}} \cdot R_{t-1} \tag{3.2}$$

Bu denklem, kalan vektörü ile en çok ilişkili örnekleme matrisi kolonunun kalan matris üzerine etkisinin, kalan matrisinden çıkarılmasını sağlar. Controller modülü, modül işleme başlamadan önce algoritmanın kaçıncı tekrarı olduğunu Step girişi aracılığı ile bu modüle bildirmesi gerekir. Controller modülü Step bilgisinin yanında optimizasyon modülü işlemi bitirdiğinde, optimizasyon modülünden aldığı Column_Index bilgisini bu modülün Column_Ind girişine, Column_Value bilgisini ise Column_Val girişine verir. Bu girişlere gerekli bilgileri verdikten sonra Start_Gram_Sch girişini 1 değerine çekmesi ile modül işleme başlar. Bu modül 3. bölümde OMP yeniden oluşturma algoritmasının d adımında anlatılan formüller aracılığı ile verilen kolon değerlerinin q değerlerini hesaplar. Herhangi bir adımda hesapladığı q değerlerini bir sonraki adımda da kullanacağı için bütün q değerlerini saklar. İşlemi bitirdiği zaman o adımda verilen kolonun 32 satırının hesaplanan q değerlerini Q_value_1, Q_value_2, Q_value_3, ... , Q_value_32 isimli 32 adet çıkışa verir ve Gram_Sch_done sinyalini 1 değerine çekerek işlemi bitirdiğini Controller modülüne bildirir. MGS modülünün altında modülün içerisindeki 32 bit sabit noktalı çarpma ve bölme işlemlerini yapabilmek için 32 adet çarpma ve bölme modülü bulunmaktadır. MGS modülü bu işlemi ilk iki adımda 3, daha sonraki adımlarda 6 saat darbesinde (clock cycle) gerçekleştirir. MGS modülünün giriş çıkış pinlerinin detaylı bilgileri Çizelge 3.4'de, modülün blok diyagramı ise Şekil 3.10'da verilmiştir.

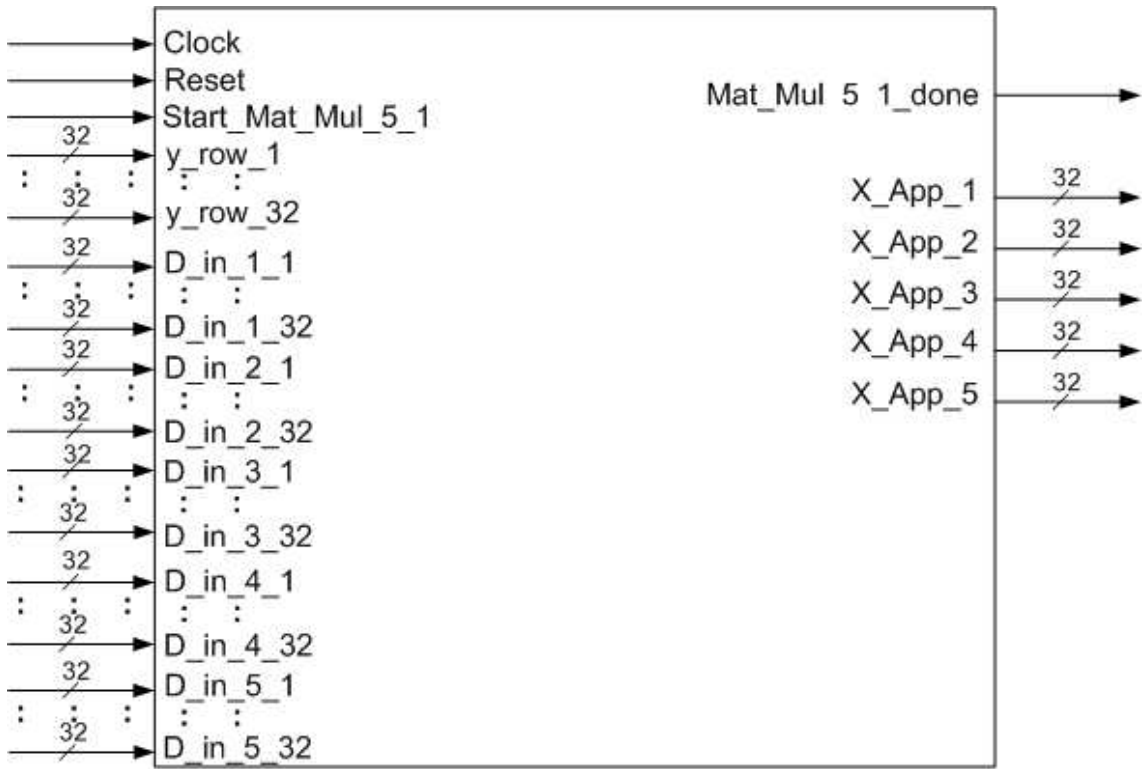
Şekil 3.10 Modified Gram Schmidt modülü blok diyagramı

Çizelge 3.4 Modified Gram Schmidt modülü giriş çıkış pinleri

Pin İsmi	Giriş/Çıkış	Bit Sayısı	Açıklama
Clock	Giriş	1 bit	Clock
Reset	Giriş	1 bit	Reset Sinyali
Start_Gram_Sch	Giriş	1 bit	Modülün işleme başlamasını bildiren sinyal
Step	Giriş	3 bit	Algoritmanın tekrar sayısını gösterir.
Column_Ind	Giriş	8 bit	Optimizasyon probleminin sonucu olan kolonun kaçıncı kolon olduğu bilgisi
Column_Val	Giriş	32 bit	Optimizasyon probleminin sonucu olan kolonun değeri
Q_value_1 Q_value_2 ..... Q_value_32	Çıkış Çıkış ..... Çıkış	32 bit 32 bit .. 32 bit	Mod. Gram Sch. işlemini sonucunun 1. satır değeri Mod. Gram Sch. işlemini sonucunun 2. satır değeri ..... Mod. Gram Sch. işlemini sonucunun 32. satır değeri
Gram_Sch_done	Çıkış	1 bit	Modülün işlemi bitirdiğini belirten sinyal

3.2.1.6 5x1 matris çarpım modülü

Bu modülün isminin 5x1 matris çarpım modülü olmasının nedeni 5x32 lik bir matris ile 32x1 lik bir matrisi, matris çarpım işlemine sokması ve sonucunda 5x1 lik bir matris elde etmesidir. Bu modül bütün algoritma sırasıyla işledikten sonra 5x32 lik *D* matrisi ile 32x1 lik *y* ölçüm vektörünü matris çarpma işlemine sokarak *x* sinyalinin yaklaşık değerlerini bulan son işlem modülüdür. Controller modülü bu modülün çalışması için gerekli olan verileri girişlerine verdikten sonra Start_Mat_Mul_5_1 girişine 1 değerini vermesi ile modül işleme başlar. Modül matris çarpma işlemini bitirdiğinde Mat_Mul_5_1_done çıkışını 1 değerine çekerek işlemi bitirdiğini controller modülüne bildirir. 5x1 matris çarpım modülü bu işlemi 7 saat darbesinde (clock cycle) gerçekleştirir. 5x1 matris çarpım modülünün giriş çıkış pinlerinin detaylı bilgileri Çizelge 3.5'de, modülün blok diyagramı ise Şekil 3.11'de verilmiştir.



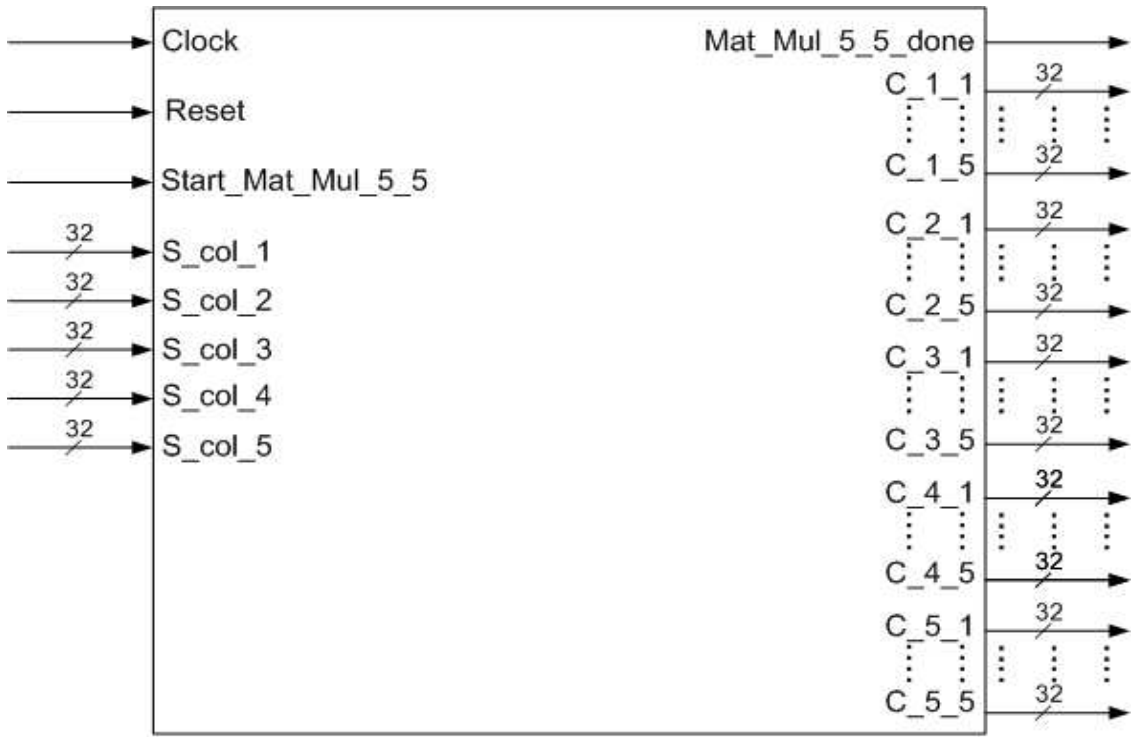
Şekil 3.11 5x1 matris çarpım modülü blok diyagramı

Çizelge 3.5 5x1 matris çarpım modülü giriş çıkış pinleri

Pin İsmi	Giriş/Çıkış	Bit Sayısı	Açıklama
Clock	Giriş	1 bit	Clock
Reset	Giriş	1 bit	Reset Sinyali
Start_Mat_Mul_5_1	Giriş	1 bit	Modülün işleme başlamasını bildiren sinyal
y_row_1 y_row_2 ..... y_row_32	Giriş Giriş ..... Giriş	32 bit 32 bit .. 32 bit	Ölçüm vektörünün 1. satır değeri Ölçüm vektörünün 2. satır değeri ..... Ölçüm vektörünün 32. satır değeri
D_in_1_1 D_in_1_2 ..... D_in_1_32	Giriş Giriş ..... Giriş	32 bit 32 bit .. 32 bit	D matrisinin 1. kolonunun 1. satır değeri D matrisinin 1. kolonunun 2. satır değeri ..... D matrisinin 1. kolonunun 32. satır değeri
D_in_2_1 D_in_2_2 ..... D_in_2_32	Giriş Giriş ..... Giriş	32 bit 32 bit .. 32 bit	D matrisinin 2. kolonunun 1. satır değeri D matrisinin 2. kolonunun 2. satır değeri ..... D matrisinin 2. kolonunun 32. satır değeri
D_in_3_1 D_in_3_2 ..... D_in_3_32	Giriş Giriş ..... Giriş	32 bit 32 bit .. 32 bit	D matrisinin 3. kolonunun 1. satır değeri D matrisinin 3. kolonunun 2. satır değeri ..... D matrisinin 3. kolonunun 32. satır değeri
D_in_4_1 D_in_4_2 ..... D_in_4_32	Giriş Giriş ..... Giriş	32 bit 32 bit .. 32 bit	D matrisinin 4. kolonunun 1. satır değeri D matrisinin 4. kolonunun 2. satır değeri ..... D matrisinin 4. kolonunun 32. satır değeri
D_in_5_1 D_in_5_2 ..... D_in_5_32	Giriş Giriş ..... Giriş	32 bit 32 bit .. 32 bit	D matrisinin 5. kolonunun 1. satır değeri D matrisinin 5. kolonunun 2. satır değeri ..... D matrisinin 5. kolonunun 32. satır değeri
Mat_Mul_5_5_done	Çıkış	1 bit	Modülün işlemi bitirdiğini belirten sinyal
X_App_1 X_App_2 X_App_3 X_App_4 X_App_5	Çıkış Çıkış Çıkış Çıkış Çıkış	32 bit 32 bit 32 bit 32 bit 32 bit	x sinyalinin yaklaşık değeri 1. satırı x sinyalinin yaklaşık değeri 2. satırı x sinyalinin yaklaşık değeri 3. satırı x sinyalinin yaklaşık değeri 4. satırı x sinyalinin yaklaşık değeri 5. satırı

3.2.1.7 5x5 matris çarpım modülü

Bu modülün isminin 5x5 matris çarpım modülü olmasının nedeni 5x32 lik bir matris ile 32x5 lik bir matrisi, matris çarpma işlemine sokması ve sonucunda 5x5 lik bir matris elde etmesidir. y ölçüm vektörüne etkisi bulunan ϕ örnekleme matrisinin m sayıda kolonu ϕ örnekleme matrisinin içinden optimizasyon modülü aracılığı ile seçilip alındığı anlatılmıştı. Controller modülüne iletilen bu kolonlar birleştirilerek yeni bir matris oluşturulur. Bu oluşturulan matris m = 5 verildiği için 32x5 lik bir matristir. Bu modül girişine verilen bu 32x5 lik matrisin transpozu ile kendisini matris çarpım işlemine sokarak 5x5 lik bir matris elde eder. Diğer modüller gibi bu modül de controller modülü tarafından Start_Mat_Mul_5_5 girişinin 1 değerine çekilmesi ile işleme başlar. Matris çarpma işlemini bitirdiğinde ise Mat_Mul_5_5_done çıkışını 1 değerine çekerek işlemi bitirdiğini controller modülüne bildirir. 5x5 matris çarpım modülü bu işlemi 4 saat darbesinde (clock cycle) gerçekleştirir. 5x5 matris çarpım modülünün giriş çıkış pinlerinin detaylı bilgileri Çizelge 3.6'da, modülün blok diyagramı ise Şekil 3.12'de verilmiştir.



Şekil 3.12 5x5 matris çarpım modülü blok diyagramı

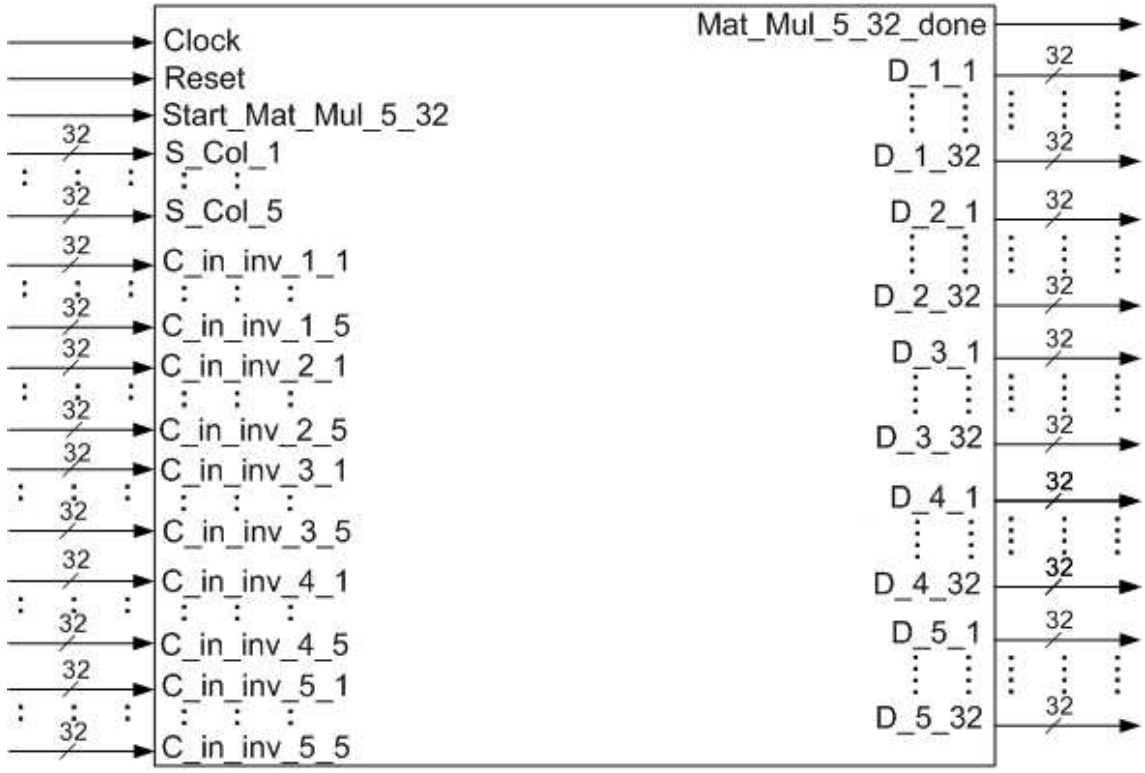
Çizelge 3.6 5x5 matris çarpım modülü giriş çıkış pinleri

Pin İsmi	Giriş/Çıkış	Bit Sayısı	Açıklama
Clock	Giriş	1 bit	Clock
Reset	Giriş	1 bit	Reset Sinyali
Start_Mat_Mul_5_5	Giriş	1 bit	Modülün işleme başlamasını bildiren sinyal
S_Col_1	Giriş	32 bit	ϕ matrisinin seçilen 1. kolonu
S_Col_2	Giriş	32 bit	ϕ matrisinin seçilen 2. kolonu
S_Col_3	Giriş	32 bit	ϕ matrisinin seçilen 3. kolonu
S_Col_4	Giriş	32 bit	ϕ matrisinin seçilen 4. kolonu
S_Col_5	Giriş	32 bit	ϕ matrisinin seçilen 5. kolonu
Mat_Mul_5_5_done	Çıkış	1 bit	Modülün işlemi bitirdiğini belirten sinyal
C_1_1	Çıkış	32 bit	Matris çarpma işlem sonucunun (1,1) elemanı
C_1_2	Çıkış	32 bit	Matris çarpma işlem sonucunun (1,2) elemanı
C_1_3	Çıkış	32 bit	Matris çarpma işlem sonucunun (1,3) elemanı
C_1_4	Çıkış	32 bit	Matris çarpma işlem sonucunun (1,4) elemanı
C_1_5	Çıkış	32 bit	Matris çarpma işlem sonucunun (1,5) elemanı
C_2_1	Çıkış	32 bit	Matris çarpma işlem sonucunun (2,1) elemanı
C_2_2	Çıkış	32 bit	Matris çarpma işlem sonucunun (2,2) elemanı
C_2_3	Çıkış	32 bit	Matris çarpma işlem sonucunun (2,3) elemanı
C_2_4	Çıkış	32 bit	Matris çarpma işlem sonucunun (2,4) elemanı
C_2_5	Çıkış	32 bit	Matris çarpma işlem sonucunun (2,5) elemanı
C_3_1	Çıkış	32 bit	Matris çarpma işlem sonucunun (3,1) elemanı
C_3_2	Çıkış	32 bit	Matris çarpma işlem sonucunun (3,2) elemanı
C_3_3	Çıkış	32 bit	Matris çarpma işlem sonucunun (3,3) elemanı
C_3_4	Çıkış	32 bit	Matris çarpma işlem sonucunun (3,4) elemanı
C_3_5	Çıkış	32 bit	Matris çarpma işlem sonucunun (3,5) elemanı
C_4_1	Çıkış	32 bit	Matris çarpma işlem sonucunun (4,1) elemanı
C_4_2	Çıkış	32 bit	Matris çarpma işlem sonucunun (4,2) elemanı
C_4_3	Çıkış	32 bit	Matris çarpma işlem sonucunun (4,3) elemanı
C_4_4	Çıkış	32 bit	Matris çarpma işlem sonucunun (4,4) elemanı
C_4_5	Çıkış	32 bit	Matris çarpma işlem sonucunun (4,5) elemanı
C_5_1	Çıkış	32 bit	Matris çarpma işlem sonucunun (5,1) elemanı
C_5_2	Çıkış	32 bit	Matris çarpma işlem sonucunun (5,2) elemanı
C_5_3	Çıkış	32 bit	Matris çarpma işlem sonucunun (5,3) elemanı
C_5_4	Çıkış	32 bit	Matris çarpma işlem sonucunun (5,4) elemanı
C_5_5	Çıkış	32 bit	Matris çarpma işlem sonucunun (5,5) elemanı

3.2.1.8 5x32 matris çarpım modülü

Bu modülün isminin 5x32 matris çarpım modülü olmasının nedeni 5x5 lik bir matris ile 5x32 lik bir matrisi, matris çarpma işlemine sokması ve sonucunda 5x32 lik bir matris

elde etmesidir. ACD ve matris tersine alma modülünün çıkışı olan 5x5 lik *C* matrisinin tersi ile 5x32 lik *y* ölçüm vektörüne etkisi bulunan ϕ örnekleme matrisinin 5 tane kolonunun transpozu bu modül aracılığı ile matris çarpım işlemine sokulur. Sonuç 5x32 lik bir matris olacaktır ve bu matris de matris çarpımı 5x1 modülüne giriş olarak verilecektir. Oluşturulacak olan bu 5x32 lik matrise *D* matrisi denilecektir. Controller modülü bu modülün çalışması için gerekli olan verileri girişlerine verdikten sonra Start_Mat_Mul_5_32 girişine 1 değerini vermesi ile modül işleme başlar. Modül matris çarpma işlemini bitirdiğinde Mat_Mul_5_32_done çıkışını 1 değerine çekerek işlemi bitirdiğini controller modülüne bildirir. 5x32 matris çarpım modülü bu işlemi 4 saat darbesinde (clock cycle) gerçekleştirir. 5x32 matris çarpım modülünün giriş çıkış pinlerinin detaylı bilgileri Çizelge 3.7'de, modülün blok diyagramı ise Şekil 3.13'de verilmiştir.



Şekil 3.13 5x32 matris çarpım modülü blok diyagramı

Çizelge 3.7 5x32 matris çarpım modülü giriş çıkış pinleri

Pin İsmi	Giriş/Çıkış	Bit Sayısı	Açıklama
Clock	Giriş	1 bit	Clock
Reset	Giriş	1 bit	Reset Sinyali
Start_Mat_Mul_5_32	Giriş	1 bit	Modülün işleme başlamasını bildiren sinyal
S_Col_1 S_Col_2 S_Col_3 S_Col_4 S_Col_5	Giriş Giriş Giriş Giriş Giriş	32 bit 32 bit 32 bit 32 bit 32 bit	ϕ matrisinin seçilen 1. kolonu ϕ matrisinin seçilen 2. kolonu ϕ matrisinin seçilen 3. kolonu ϕ matrisinin seçilen 4. kolonu ϕ matrisinin seçilen 5. kolonu
C_in_inv_1_1 ..... C_in_inv_1_5	Giriş ..... Giriş	32 bit ..... 32 bit	C matrisi tersinin 1. kolonunun 1. satır değeri ..... C matrisi tersinin 1. kolonunun 5. satır değeri
C_in_inv_2_1 ..... C_in_inv_2_5	Giriş ..... Giriş	32 bit ..... 32 bit	C matrisi tersinin 2. kolonunun 1. satır değeri ..... C matrisi tersinin 2. kolonunun 5. satır değeri
C_in_inv_3_1 ..... C_in_inv_3_5	Giriş ..... Giriş	32 bit ..... 32 bit	C matrisi tersinin 3. kolonunun 1. satır değeri ..... C matrisi tersinin 3. kolonunun 5. satır değeri
C_in_inv_4_1 ..... C_in_inv_4_5	Giriş ..... Giriş	32 bit ..... 32 bit	C matrisi tersinin 4. kolonunun 1. satır değeri ..... C matrisi tersinin 4. kolonunun 5. satır değeri
C_in_inv_5_1 ..... C_in_inv_5_5	Giriş ..... Giriş	32 bit ..... 32 bit	C matrisi tersinin 5. kolonunun 1. satır değeri ..... C matrisi tersinin 5. kolonunun 5. satır değeri
Mat_Mul_5_32_done	Çıkış	1 bit	Modülün işlemi bitirdiğini belirten sinyal
D_1_1 ..... D_1_32	Çıkış ..... Çıkış	32 bit ..... 32 bit	D matrisinin 1. kolonunun 1. satır değeri ..... D matrisinin 1. kolonunun 32. satır değeri
D_2_1 ..... D_2_32	Çıkış ..... Çıkış	32 bit ..... 32 bit	D matrisinin 2. kolonunun 1. satır değeri ..... D matrisinin 2. kolonunun 32. satır değeri
D_3_1 ..... D_3_32	Çıkış ..... Çıkış	32 bit ..... 32 bit	D matrisinin 3. kolonunun 1. satır değeri ..... D matrisinin 3. kolonunun 32. satır değeri
D_4_1 ..... D_4_32	Çıkış ..... Çıkış	32 bit ..... 32 bit	D matrisinin 4. kolonunun 1. satır değeri ..... D matrisinin 4. kolonunun 32. satır değeri
D_5_1 ..... D_5_32	Çıkış ..... Çıkış	32 bit ..... 32 bit	D matrisinin 5. kolonunun 1. satır değeri ..... D matrisinin 5. kolonunun 32. satır değeri

3.2.1.9 ACD ve matris tersi alma modülü

ACD ve matris tersi alma modülünde iki işlemin aynı anda yapılması istenilmiştir. Bu modül ACD yöntemine göre verilen matrisi LDL^{T} şeklinde 3 matrise ayırır ve L matrisi ile D matrisinin tersini alır. FPGA tipi aygıtlar CPU (Merkezi İşlem Birimi – Central Processing Unit) tipi işlemcilerden ayıran en temel özellik CPU lar birden fazla çekirdekli olmadıkları sürece aynı anda sadece bir iş yapabilmeleridir. Fakat FPGA ler istenildiği gibi tasarlanabildiği için aynı anda sayısız işlem yapabilmektedir. Bu modülde ACD ve matris tersi alma işleminin birleştirilmesinin amacı FPGA'nın bu özelliğini kullanarak ACD ile bir yandan 3 matrise ayrılmaya çalışılırken elde edilen matris elemanları ile bir yandan da bu elemanlar ile matris tersi alma işlemine başlamaktır. Dolayısıyla ACD işleminin bitmesiyle hemen hemen anı zamanda matris tersi alma işlemi de bitmektedir. Bu birleştirme işlemi Matlab kodlarıyla anlatılırsa, aşağıdaki kod 5x5 lik bir matrisin LDL açılımı için yazılmış bir koddur.

```
D = zeros(5,5); L = zeros(5,5);

L(1,1)= 1;
L(2,2)= 1;
L(3,3)= 1;
L(4,4)= 1;
L(5,5)= 1;

D(1,1) = A(1,1);

L(2,1) = (1/D(1,1))*A(2,1);
L(3,1) = (1/D(1,1))*A(3,1);
L(4,1) = (1/D(1,1))*A(4,1);
L(5,1) = (1/D(1,1))*A(5,1);

D(2,2) = A(1,1) -  (L(2,1)*L(2,1)*D(1,1));

L(3,2) = (1/D(2,2))*(A(3,2)-(L(3,1)*L(2,1)*D(1,1)));
L(4,2) = (1/D(2,2))*(A(4,2)-(L(4,1)*L(2,1)*D(1,1)));
L(5,2) = (1/D(2,2))*(A(5,2)-(L(5,1)*L(2,1)*D(1,1)));

D(3,3) = A(1,1) - ((L(3,1)*L(3,1)*D(1,1)) + (L(3,2)*L(3,2)*D(2,2)));

L(4,3) = (1/D(3,3))*(A(4,3)-((L(4,1)*L(3,1)*D(1,1)) +
(L(4,2)*L(3,2)*D(2,2))));
L(5,3) = (1/D(3,3))*(A(5,3)-((L(5,1)*L(3,1)*D(1,1)) +
(L(5,2)*L(3,2)*D(2,2))));

D(4,4) = A(1,1) - ((L(4,1)*L(4,1)*D(1,1)) + (L(4,2)*L(4,2)*D(2,2)) +
(L(4,3)*L(4,3)*D(3,3)));
```

```
L(5,4) = (1/D(4,4))*(A(5,4)-((L(5,1)*L(4,1)*D(1,1)) +
(L(5,2)*L(4,2)*D(2,2)) + (L(5,3)*L(4,3)*D(3,3))));

D(5,5) = A(1,1) - ((L(5,1)*L(5,1)*D(1,1)) + (L(5,2)*L(5,2)*D(2,2)) +
(L(5,3)*L(5,3)*D(3,3)) + (L(5,4)*L(5,4)*D(4,4)));
```

Aşağıdaki kod ise *L* ve *D* matrislerinin tersini almak için yazılan bir koddur.

```
L_inv = zeros(5,5);

L_inv(1,1)= 1;
L_inv(2,2)= 1;
L_inv(3,3)= 1;
L_inv(4,4)= 1;
L_inv(5,5)= 1;

L_inv(2,1)= - L(2,1);
L_inv(3,2)= - L(3,2);
L_inv(4,3)= - L(4,3);
L_inv(5,4)= - L(5,4);

L_inv(3,1)= L(2,1)*L(3,2) - L(3,1);
L_inv(4,2)= L(3,2)*L(4,3) - L(4,2);
L_inv(5,3)= L(4,3)*L(5,4) - L(5,3);

L_inv(4,1)= - (L(4,1) + L(4,2)*L_inv(2,1) + L(4,3)*L_inv(3,1));
L_inv(5,2)= - (L(5,2) + L(5,3)*L_inv(3,2) + L(5,4)*L_inv(4,2));

L_inv(5,1)= - (L(5,1) + L(5,2)*L_inv(2,1) + L(5,3)*L_inv(3,1) +
L(5,4)*L_inv(4,1));

D_inv(1,1) = 1/D(1,1);
D_inv(2,2) = 1/D(2,2);
D_inv(3,3) = 1/D(3,3);
D_inv(4,4) = 1/D(4,4);
D_inv(5,5) = 1/D(5,5);
```

Bu iki kod birleştirilerek sıralı bir şekilde ACD bittiği sürede matris tersi alma işlemide bitmektedir. Bu kodda aşağıda görülmektedir.

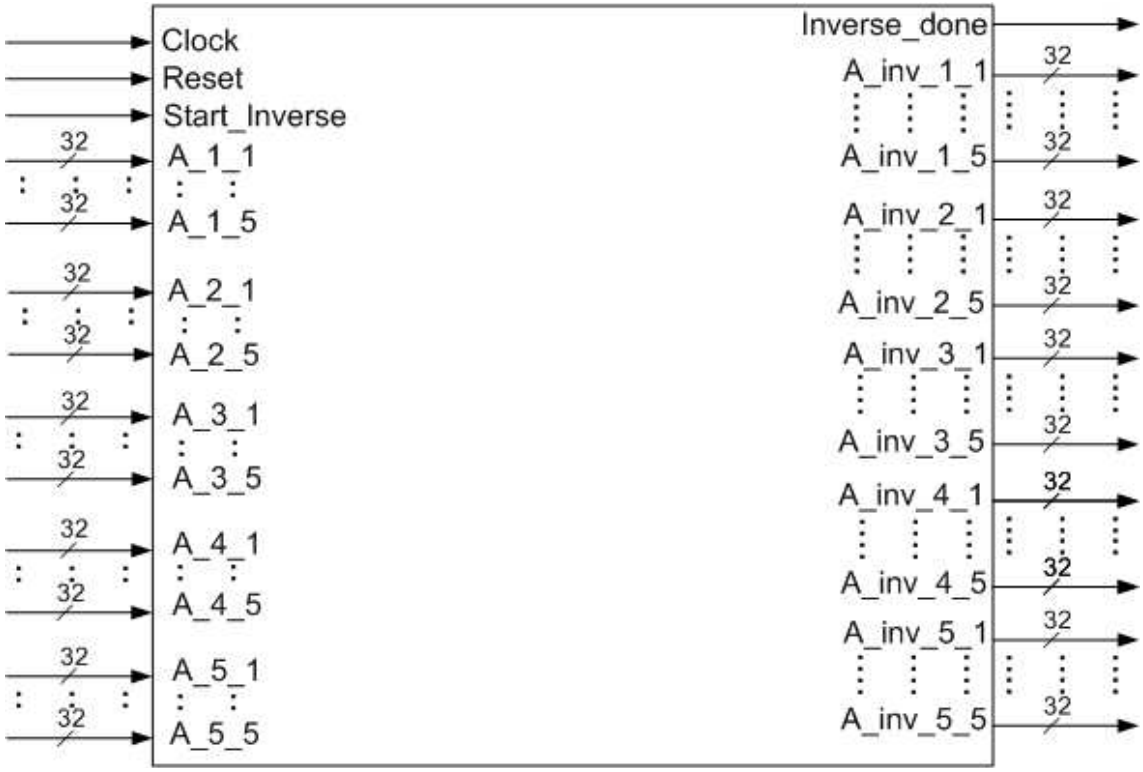
```
D = zeros(5,5);
D_inv = zeros(5,5);
L = zeros(5,5);
L_inv = zeros(5,5);
L(1,1)= 1;
L(2,2)= 1;
L(3,3)= 1;
L(4,4)= 1;
L(5,5)= 1;
L_inv(1,1)= 1;
L_inv(2,2)= 1;
L_inv(3,3)= 1;
```

```
L_inv(4,4)= 1;
L_inv(5,5)= 1;
D(1,1) = A(1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%step1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L(2,1) = (1/D(1,1))*A(2,1);
L(3,1) = (1/D(1,1))*A(3,1);
L(4,1) = (1/D(1,1))*A(4,1);
L(5,1) = (1/D(1,1))*A(5,1);
D_inv(1,1) = 1/D(1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%step2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
D(2,2) = A(1,1) -  (L(2,1)*L(2,1)*D(1,1));
L_inv(2,1)= - L(2,1);
%%%%%%%%%%%%%%%%%%%%%%%%%step3%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L(3,2) = (1/D(2,2))*(A(3,2)-(L(3,1)*L(2,1)*D(1,1)));
L(4,2) = (1/D(2,2))*(A(4,2)-(L(4,1)*L(2,1)*D(1,1)));
L(5,2) = (1/D(2,2))*(A(5,2)-(L(5,1)*L(2,1)*D(1,1)));
D_inv(2,2) = 1/D(2,2);
%%%%%%%%%%%%%%%%%%%%%%%%%step4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
D(3,3) = A(1,1) - ((L(3,1)*L(3,1)*D(1,1)) + (L(3,2)*L(3,2)*D(2,2)));
L_inv(3,2)= - L(3,2);
L_inv(3,1)= L(2,1)*L(3,2) - L(3,1);
%%%%%%%%%%%%%%%%%%%%%%%%%step5%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L(4,3) = (1/D(3,3))*(A(4,3)-((L(4,1)*L(3,1)*D(1,1)) +
(L(4,2)*L(3,2)*D(2,2))));
L(5,3) = (1/D(3,3))*(A(5,3)-((L(5,1)*L(3,1)*D(1,1)) +
(L(5,2)*L(3,2)*D(2,2))));
D_inv(3,3) = 1/D(3,3);
%%%%%%%%%%%%%%%%%%%%%%%%%step6%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
D(4,4) = A(1,1) - ((L(4,1)*L(4,1)*D(1,1)) + (L(4,2)*L(4,2)*D(2,2)) +
(L(4,3)*L(4,3)*D(3,3)));
L_inv(4,3)= - L(4,3);
L_inv(4,2)= L(3,2)*L(4,3) - L(4,2);
L_inv(4,1)= - (L(4,1) + L(4,2)*L_inv(2,1) + L(4,3)*L_inv(3,1));
%%%%%%%%%%%%%%%%%%%%%%%%%step7%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L(5,4) = (1/D(4,4))*(A(5,4)-((L(5,1)*L(4,1)*D(1,1)) +
(L(5,2)*L(4,2)*D(2,2)) + (L(5,3)*L(4,3)*D(3,3))));
D_inv(4,4) = 1/D(4,4);
%%%%%%%%%%%%%%%%%%%%%%%%%step8%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
D(5,5) = A(1,1) - ((L(5,1)*L(5,1)*D(1,1)) + (L(5,2)*L(5,2)*D(2,2)) +
(L(5,3)*L(5,3)*D(3,3)) + (L(5,4)*L(5,4)*D(4,4)));
L_inv(5,4)= - L(5,4);
L_inv(5,3)= L(4,3)*L(5,4) - L(5,3);
L_inv(5,2)= - (L(5,2) + L(5,3)*L_inv(3,2) + L(5,4)*L_inv(4,2));
L_inv(5,1)= - (L(5,1) + L(5,2)*L_inv(2,1) + L(5,3)*L_inv(3,1) +
L(5,4)*L_inv(4,1));
%%%%%%%%%%%%%%%%%%%%%%%%%step9%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
D_inv(5,5) = 1/D(5,5);
```

Controller modülünün Start_inverse girişine 1 değerini vermesi ile modül işleme başlar. Modül ACD ve matris tersi alma işlemini bitirdiğinde Inverse_done çıkışını 1 değerine çekerek işlemi bitirdiğini controller modülüne bildirir. ACD ve matris tersi alma modülü bu işlemi 46 saat darbesinde (clock cycle) gerçekleştirir. ACD ve matris tersi alma modülünün giriş çıkış pinlerinin detaylı bilgileri Çizelge 3.8'de, modülün blok diyagramı ise şekil 3.14'de verilmiştir.

Çizelge 3.8 ACD ve matris tersi alma modülü giriş çıkış pinleri

Pin İsmi	Giriş/Çıkış	Bit Sayısı	Açıklama
Clock	Giriş	1 bit	Clock
Reset	Giriş	1 bit	Reset Sinyali
Start_Inverse	Giriş	1 bit	Modülün işleme başlamasını bildiren sinyal
A_1_1 ..... A_1_5	Giriş ..... Giriş	32 bit ..... 32 bit	Tersi alınacak matrisin 1.kolon 1.satır değeri ..... Tersi alınacak matrisin 1.kolon 5.satır değeri
A_2_1 ..... A_2_5	Giriş ..... Giriş	32 bit ..... 32 bit	Tersi alınacak matrisin 2.kolon 1.satır değeri ..... Tersi alınacak matrisin 2.kolon 5.satır değeri
A_3_1 ..... A_3_5	Giriş ..... Giriş	32 bit ..... 32 bit	Tersi alınacak matrisin 3.kolon 1.satır değeri ..... Tersi alınacak matrisin 3.kolon 5.satır değeri
A_4_1 ..... A_4_5	Giriş ..... Giriş	32 bit ..... 32 bit	Tersi alınacak matrisin 4.kolon 1.satır değeri ..... Tersi alınacak matrisin 4.kolon 5.satır değeri
A_5_1 ..... A_5_5	Giriş ..... Giriş	32 bit ..... 32 bit	Tersi alınacak matrisin 5.kolon 1.satır değeri ..... Tersi alınacak matrisin 5.kolon 5.satır değeri
Inverse_done	Çıkış	1 bit	Modülün işlemi bitirdiğini belirten sinyal
A_inv_1_1 ..... A_inv_1_5	Çıkış ..... Çıkış	32 bit ..... 32 bit	A matrisinin tersinin 1.kolon 1.satır değeri ..... A matrisinin tersinin 1.kolon 5.satır değeri
A_inv_2_1 ..... A_inv_2_5	Çıkış ..... Çıkış	32 bit ..... 32 bit	A matrisinin tersinin 2.kolon 1.satır değeri ..... A matrisinin tersinin 2.kolon 5.satır değeri
A_inv_3_1 ..... A_inv_3_5	Çıkış ..... Çıkış	32 bit ..... 32 bit	A matrisinin tersinin 3.kolon 1.satır değeri ..... A matrisinin tersinin 3.kolon 5.satır değeri
A_inv_4_1 ..... A_inv_4_5	Çıkış ..... Çıkış	32 bit ..... 32 bit	A matrisinin tersinin 4.kolon 1.satır değeri ..... A matrisinin tersinin 4.kolon 5.satır değeri
A_inv_5_1 ..... A_inv_5_5	Çıkış ..... Çıkış	32 bit ..... 32 bit	A matrisinin tersinin 5.kolon 1.satır değeri ..... A matrisinin tersinin 5.kolon 5.satır değeri



Şekil 3.14 ACD ve matris tersi alma modülü blok diyagramı

4. ARAŞTIRMA BULGULARI

Tez çalışmasının bu kısmı, tasarlanan sistemin istenildiği gibi çalışmasını görmek adına dijital ortamda yapılan benzetimlerin ve donanım üzerinde yapılan testlerin sonuçlarını, karşılaştırmalarını ve yorumlarını içermektedir.

Tasarlanan sistemin teorik açıdan doğruluğunu ispatlayabilmek için algoritma öncelikli olarak MATLAB yazılımı kullanılarak bilgisayar ortamında test edilmiştir. Sistemin öncelikli olarak MATLAB yazılımı kullanılarak test edilmesinin sebebi, bir sonraki benzetim aracı olarak kullanılacak ve hata ayıklama işlemi daha karmaşık olan FPGA yazılımları için güvenilir giriş verileri ve bu giriş verileri sonucu elde edilecek çıkış verilerinin üretilmesidir. FPGA yazılımları ile tasarım aşamasında MATLAB ortamında üretilen veriler kullanılacak ve kullanılan giriş verileri sonucunda elde edilen verilerin, MATLAB ortamında elde edilen verilerle aynı olması beklenmektedir.

Önceki bölümlerde de anlatıldığı üzere tezin gerçeklenmesi aşamasında PLDA firmasının ürettiği üzerinde Xilinx firmasının üretmiş olduğu Virtex-6 LX550T-2FFG1759C tipi bir FPGA bulunan XpressV6-550 isimli uygulama geliştirme kartı kullanılacaktır. FPGA yazılımı olarak Xilinx ISE 13.2 kullanılacaktır. Benzetim aracı olarak ise Modelsim PE 10.0a kullanılacaktır.

Algoritma tasarımında ϕ örnekleme matrisi olarak 32x128 lik bir matris kullanılmıştır. Matris elemanları ikilik sistemden (binary) oluşmaktadır. Orjinal giriş sinyali 128 elemanlı bir vektör olarak tanımlanmıştır. Giriş sinyalinin seyreklik derecesi en fazla m=5 olacak şekilde gerçeklenmiştir. Tasarımda bu değerlerin kullanılmasının sebebi daha önce yapılan çalışmalarla karşılaştırma imkanı bulabilmektir.

Tasarım Virtex 6 LX550T FPGA üzerinde Xilinx ISE 13.2 yazılımı kullanılarak sentezlenmesi sonucunda FPGA'in % 97'sini kapladığı, çalışabileceği en yüksek saat frekansı olarak ~67 Mhz olduğu bilgisini vermektedir. Bu bilgiler Şekil 4.1 – 4.2'de görülmektedir. Algoritmanın en uzun saat darbesi harcayan modülü 46 saat darbesi ile ACD ve matris tersi alma modülüdür. Algoritmanın toplam işlem süresi 184 saat

darbesi olarak hesaplanmıştır. Süre cinsinden hesaplanırsa bu tezde tasarlanan algoritmanın, başlama sinyali ile bitiş sinyali arasında 2,7 mikrosaniye (µs) süre geçmektedir. Kısacası işlem süresi 2,7 mikrosaniye (µs)'dir.

```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 14.935ns (Maximum Frequency: 66.958MHz)
   Minimum input arrival time before clock: 2.286ns
   Maximum output required time after clock: 0.659ns
   Maximum combinational path delay: No path found
```

Şekil 4.1 En yüksek saat hızı bilgileri

```
Device utilization summary:
---------------------------

Selected Device : 6vlx550tff1759-2


Slice Logic Utilization:
 Number of Slice Registers:           61623  out of  687360     8%
 Number of Slice LUTs:               333720  out of  343680    97%
    Number used as Logic:            333720  out of  343680    97%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:  346461
   Number with an unused Flip Flop:   284838  out of  346461    82%
   Number with an unused LUT:         12741  out of  346461     3%
   Number of fully used LUT-FF pairs: 48882  out of  346461    14%
   Number of unique control sets:       117

IO Utilization:
 Number of IOs:                         196
 Number of bonded IOBs:                 196  out of    840    23%
    IOB Flip Flops/Latches:             161

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                3  out of     32     9%
 Number of DSP48E1s:                    848  out of    864    98%
```

Şekil 4.2 Kaynak kullanım bilgileri

Sıkıştırıcı Algılama ile yeniden oluşturma algoritmasının FPGA üzerinde gerçeklenmesi ile ilgili daha önce yapılmış bir adet tasarım bulunmaktadır. Yapılan diğer tasarımda da OMP algoritması kullanılmıştır. Virtex 5 üzerinde sentezlenen tasarım, donanım üzerinde gerçeklenmemiş sadece benzetim sonuçları verilmiştir. Diğer çalışmalarda ise FPGA dışında CPU ve GPU (Grafik İşlemci Ünitesi - Graphics Processing Unit) üzerinde de gerçekleştirilen OMP yeniden oluşturma algoritmaları bulunmaktadır. Daha önce gerçekleştirilen FPGA tasarımı ile bu tezde gerçekleştirilen tasarımın karşılaştırma yapılabilmesi için parametreleri aynı tutulmuştur. GPU VE CPU üzerinde yapılan tasarımlar ise gerekli oransal düzeltmelerle karşılaştırılabilir seviyeye getirilmiştir. Daha önceki tasarımların ve bu tezde ulaşılan sonuçların karşılaştırmalı tablosu Çizelge 4.1'de verilmiştir.

Çizelge 4.1 İşlem sürelerinin karşılaştırılması

	Süre(µs)
Önerilen Virtex-6 OMP tasarımı	2,76
Virtex-5 OMP tasarımı	24
CUBLAS GPU	37500
İntel Core i7, 920	68000

Çizelge 4.1'de görüldüğü üzere çeşitli iyileştirmeler sonucunda tasarlanan OMP yöntemi ile sıkıştırıcı algılama yeniden oluşturma algoritması diğer tasarımlara göre daha hızlıdır. GPU ve CPU uygulamalarına göre onbin kat daha hızlı olduğu gibi, diğer FPGA tasarımına göre de yaklaşık 9 kat hızlıdır. FPGA üzerinde gerçekleştirilen diğer tasarıma göre süre cinsinden hızlı olmasının iki tane ana nedeni vardır. Bunlardan ilki işlem süresinin saat darbesi cinsinden daha kısa olması, ikincisi ise algoritmanın çalışabildiği sistem frekansının daha yüksek olmasıdır.

Tasarlanan sistemin gerçeklenmesi bu bölümde iki ayrı başlık altında verilecektir. Bu başlıklardan birincisi tasarımın FPGA benzetim araçları üzerinde gerçeklenmesi, ikincisi ise tasarımın donanım üzerinde gerçeklenmesi olacaktır.

4.1 Tasarımın FPGA Benzetim Araçları Üzerinde Gerçeklenmesi

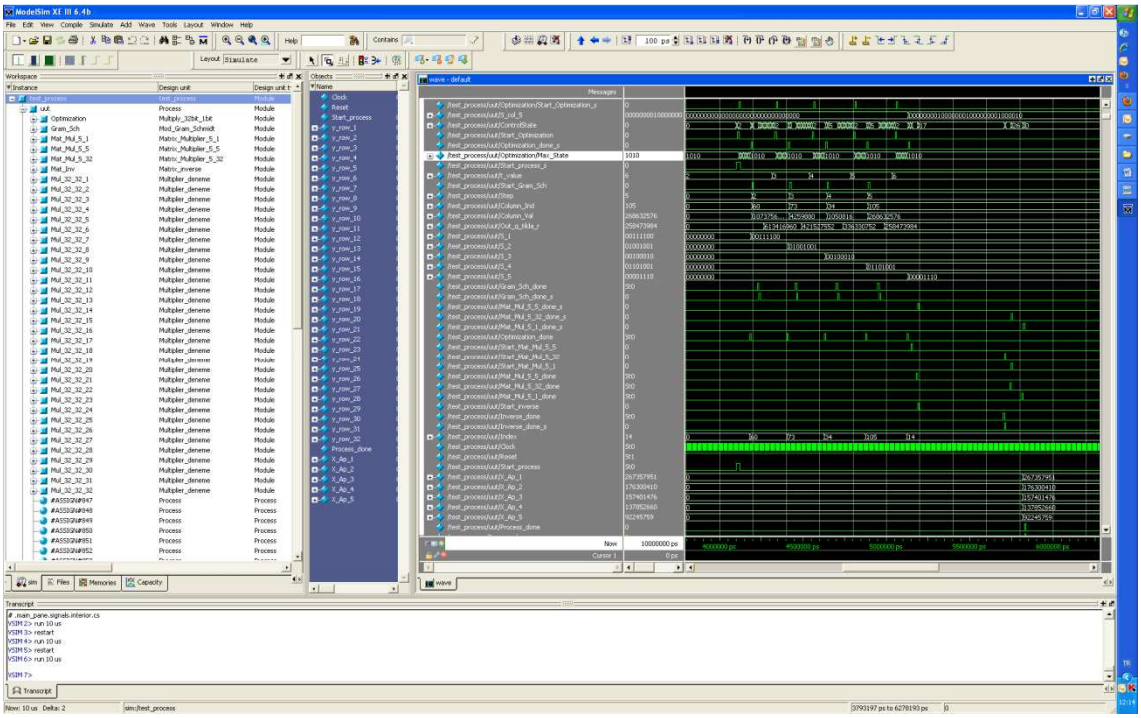
Tasarımın Modelsim PE 10.0a üzerinde benzetimleri yapılmıştır. Yapılan benzetimler sonucu ortaya çıkan veriler MATLAB programıyla gerçekleştirilen tasarımla aynı sonuçları vermiştir. OMP algoritması ile yeniden oluşturulan sinyal orjinal sinyale çok yakın değerlere yaklaşmıştır. Algoritmaya verilen sinyallerin büyük katsayıları ile yeniden oluşturulup elde edilen sinyallerin büyük katsayıları Çizelge 4.2'de verilmiştir.

Çizelge 4.2 Elde edilen çıkış verileri

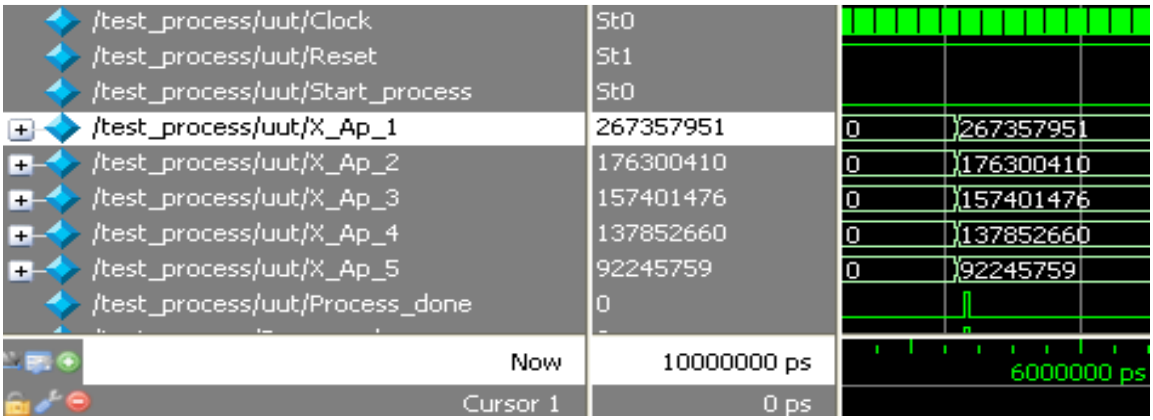
Orjinal sinyallerin büyük katsayıları	Elde edilen sinyallerin büyük katsayıları
44	43,98
73	75,05
125	127,48
82	84,03
66	65,72
101	99,28
98	97,50
41	42,94
69	70,18
74	76,26
119	121,51
39	39,94
55	54,37
83	85,64
49	48,06
114	116,84
50	51,47
121	119,12
63	65,66
87	88,54

Sıkıştırıcı algılama yöntemi ile OMP algoritması kullanılarak görüntü yeniden oluşturma problemlerinde, orjinal görüntü ϕ örnekleme matrisi ile matris çarpımı işlemine sokulduğundan orjinal görüntü verileri kayıpsız bir şekilde korunamaz. Yeniden oluşturma kısmında ise optimizasyon yöntemleri kullanılarak orjinal veriye en

yakın değerler elde edilmeye çalışılır. Çizelge 4.2'de elde edilen verilerin orjinal verilere yaklaşmasının örnekleri verilmiştir. Şekil 4.3'de Modelsim simülatöründen alınmış benzetim sonuçlarını gösteren bir ekran görüntüsü bulunmaktadır. Şekil 4.4'de ise Çizelge 4.2'de verilen çizelgenin ilk beş verisinin elde edilme benzetiminin son kısmı görülmektedir. Şekil 4.4'de görülen değerlerin ikilik sistem karşılığı Şekil 3.6'da anlatıldığı gibi çevirilirse Çizelge 4.2'de, elde edilen sinyallerin büyük katsayıları kolonundaki verilerin ilk 5 satırının benzetimlerle elde edildiği görülmektedir.



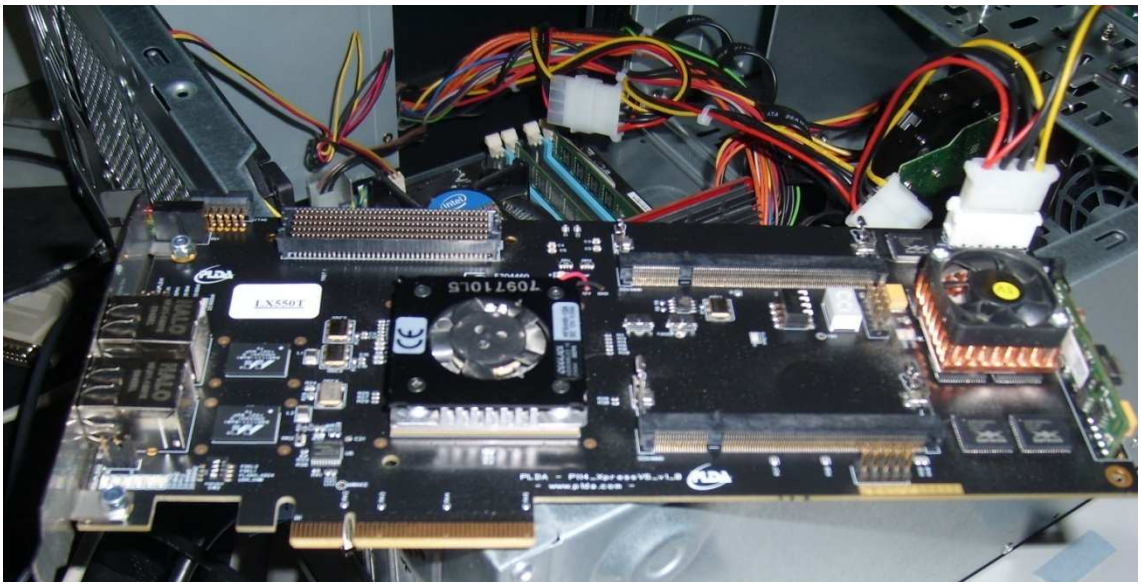
Şekil 4.3 Benzetim görüntüsü



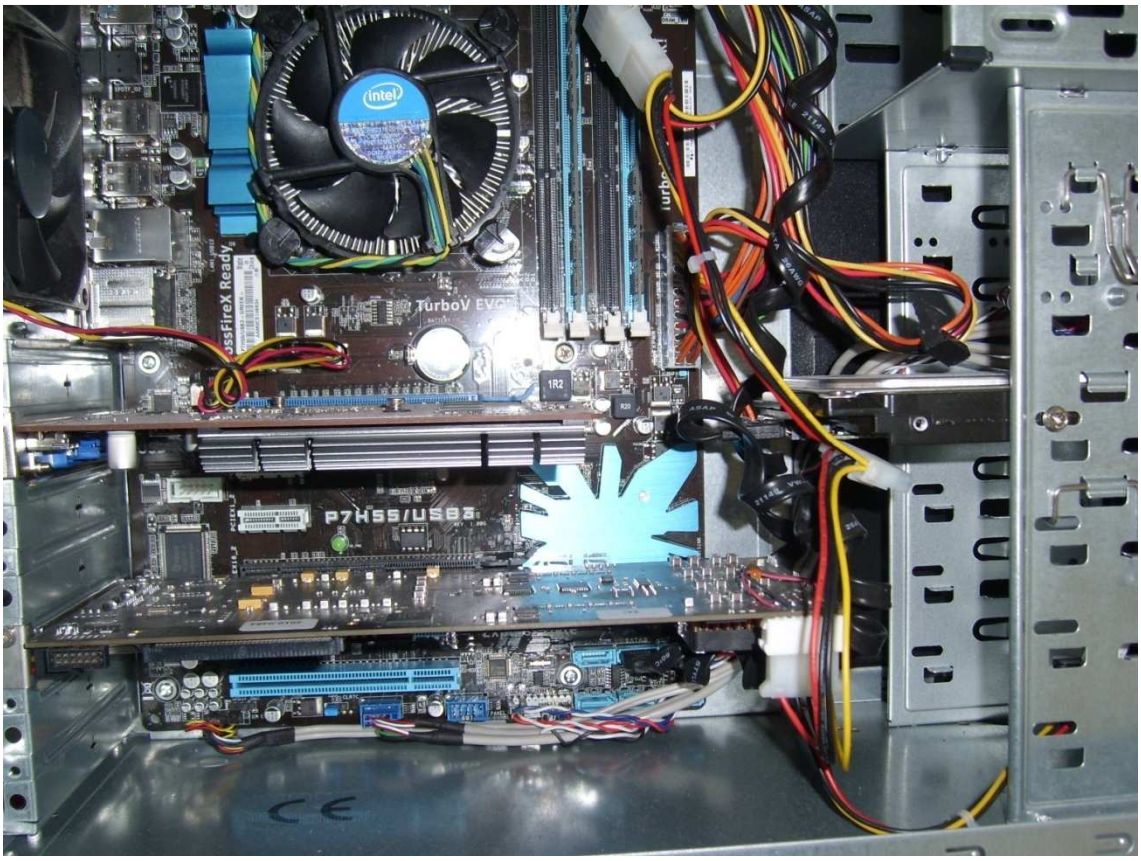
Şekil 4.4 İlk örneğin benzetim görüntüsü

4.2 Tasarımın Donanım Üzerinde Gerçeklenmesi

Sıkıştırıcı algılama yöntemi ile OMP algoritması kullanılarak görüntü yeniden oluşturma tasarımını PLDA firnasının ürettiği üzerinde Xilinx firmasının üretmiş olduğu Virtex-6 LX550T-2FFG1759C tipi bir FPGA bulunan XpressV6-550 isimli uygulama geliştirme kartı kullanılmıştır. Uygulama kartının görüntüsü Şekil 4.5'de görülmektedir. Uygulama kartının bilgisayara takılmış halde görüntüsü Şekil 4.6'da görülmektedir.



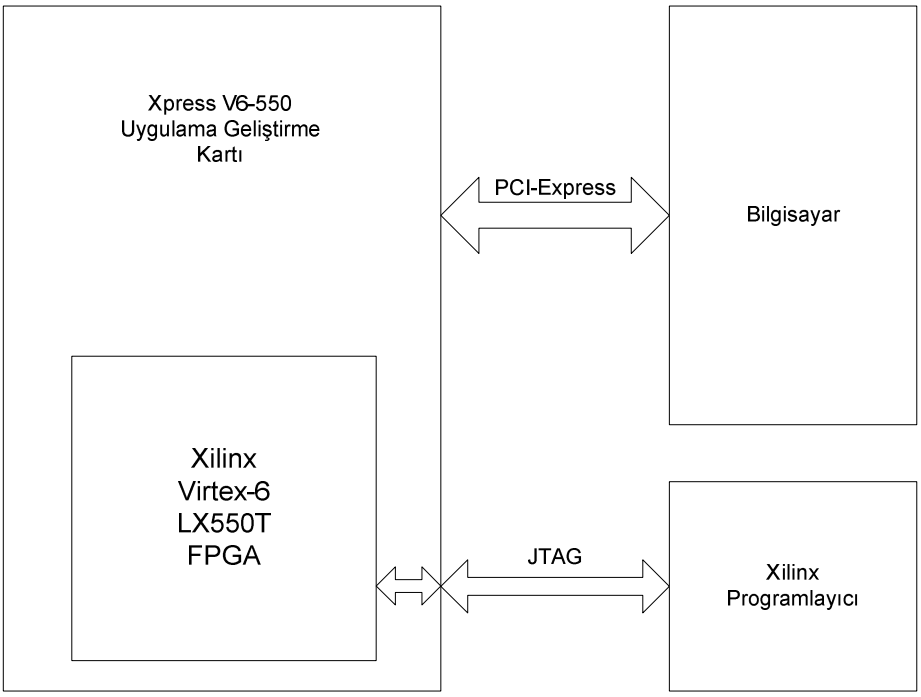
Şekil 4.5 Xpress V6-550 kartının görüntüsü



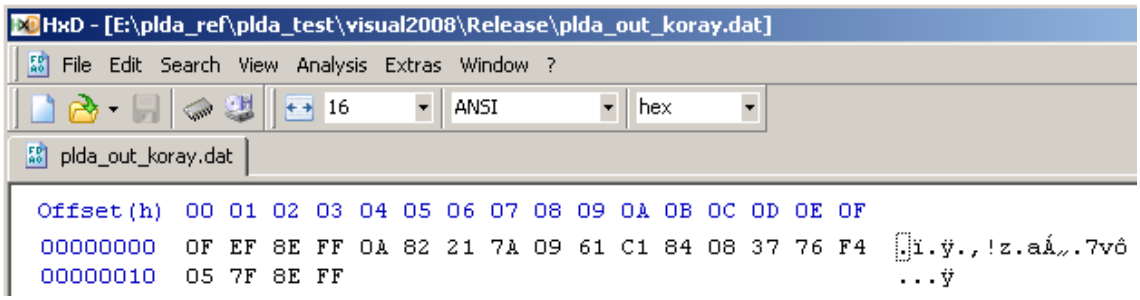
Şekil 4.6 Xpress V6-550 kartının donanım üzerine takılmış görüntüsü

OMP algoritmasının donanım üzerinde gerçeklenmesi sonucunda elde edilen veriler, algoritmanın benzetim sonuçları ile aynıdır. Algoritma donanım üzerinde de sorunsuz bir şekilde çalışmıştır. Algoritmanın donanım üzerinde gerçeklenmesini anlatan blok diyagramı Şekil 4.7'de görülmektedir. Kısaca anlatılırsa, MATLAB yardımıyla orjinal sinyal ϕ örnekleme matrisi ile matris çarpımı işlemine sokulur. Elde edilen veriler bilgisayar tarafından Xpress V6 uygulama geliştirme kartına PCI –Express üzerinden aktarılır ve başlamasını bildiren sinyal gönderilir. Gerekli verileri ve başlama sinyalini alan kartın üzerinde bulunan FPGA algoritmayı başlatır. Algoritma belirlenen süre sonunda bittiğinde algoritmanın bittiğini belirten bir sinyal ile yeniden oluşturulmuş verileri PCI-Express üzerinden bilgisayara gönderir. Böylece sıkıştırıcı algılama ile

toplanmış veriler donanım üzerinde yeniden oluşturulmuş olur. Yeniden oluşturulmuş verilerin onaltılık sistemde (hexadecimal) gösterimi Şekil 4.8'de gösterilmiştir.



Şekil 4.7 Donanım blok diyagramı



Şekil 4.8 Algoritmanın donanım üzerinde gerçeklenmesiyle elde edilen veriler

5. TARTIŞMA VE SONUÇ

SA ile görüntü yeniden oluşturma algoritmaları sinyal işleme alanında çalışan araştırmacılar tarafından ilgi çekmesine rağmen, şimdiye kadar teorik temeller üzerine çalışılmıştır. SA problemlerini ve yeniden oluşturma algoritmalarını uygulamaya geçirmek SA teorisinin gerçek hayatta kullanılmaya başlamasını sağlayacaktır. SA ile yeniden oluşturma algoritmalarından sadeliği, hızı ve donanımda gerçeklenmeye uygunluğu açısından öne çıkan OMP algoritması incelenmiş, iyileştirmeler yapılmış ve donanım üzerinde gerçeklenmiştir.

OMP algoritması üzerinde yapılan iyileştirmeler algoritmanın ana yapısını bozmadan gerçekleştirilmiştir. Geliştirilen algoritmanın doğrulanması için öncelikle MATLAB yazılımında test edilmiştir. MATLAB programıyla doğrulanan algoritma Verilog donanım tanımlama dili ile FPGA yazılımları üzerinde kodlanmıştır. Daha sonra benzetim programlarıyla benzetimi yapılarak doğrulanan algoritma, benzetim sonuçları verilerek bırakılmamış Xilinx firmasının şu anda satışta bulunan en son teknoloji FPGA'sı olan Virtex-6 üzerinde de gerçeklenmiştir.

Tasarlanan sistem CPU ve GPU uygulamalarına göre onbin kat daha hızlı, daha önce yapılan FPGA tasarımına göre ise yaklaşık 9 kat hızlıdır. Bu hızlanma, hem algoritmanın işlem süresinin saat darbesi cinsinden 184 saat darbesine kısaltılmasıyla (diğer FPGA üzerinde gerçekleme çalışmasında OMP algoritmasının işlem süresi 930 saat darbesi sürmüştür) hemde sistemin çalışabildiği en yüksek frekansın 67 MHz'e çıkarılması ile (diğer FPGA üzerinde gerçekleme çalışmasında OMP algoritmasının çalışabildiği en yüksek frekans 39 MHz olarak belirtilmiştir) elde edilmiştir.

İleride yapılabilecek çalışmalarda üzerinde birden fazla ve daha büyük FPGA'lar bulunan kartlar ile akış paralellenerek algoritma daha hızlı bir hale getirilebilir veya hız ile ilgili bir iyileştirme yapmadan algoritmanın giriş verisi daha da büyütülebilir. Tasarlanan sistem bir kamera ile entegre edilip sıkıştırılarak alınan görüntü verileri yeniden oluşturulabilir.

KAYNAKLAR

Baraniuk, R. 2007 “Compressive sensing”, IEEE Signal Processing Mag., Vol. 24, No. 4, pp. 118-121.

Baraniuk, R., Davenport, M., DeVore, R. and Wakin, M. 2008 “A simple proof of restricted isometry property for random matrices”, Constructive Approximation, Vol. 28, No. 3, pp. 258-263.

Blumensath, T. and Davies, M. 2007 “Gradient Pursuits”, IEEE Trans. Signal Processing Theory, Vol. 56, No. 6, pp. 2370-2382.

Boufounus, P., Duarte, M. and Baraniuk, R. 2007 “Sparse Signal Reconstruction from Noisy Compressive Measurements using Cross Validation”, IEEE/SP 14th Workshop on Statistical Signal Processing, pp. 299-303.

Candès, E. and Tao, T. 2005 “Decoding by lineer programming”, IEEE Trans. Inform. Theory, Vol. 51, No. 12, pp. 4203-4215.

Candès, E., Romberg, J. and Tao, T. 2006 “Robust uncertaınty principles: Exact signal reconstruction from higly incomplete frequency information”, IEEE Trans. Inform. Theory, Vol. 52, No. 2, pp. 489-509.

Candès, E. and Tao, T. 2006 “Near optimal signal recovery from random projection: Universal encoding strategies?”, IEEE Trans. Inform. Theory, Vol. 52, No. 12, pp. 5406-5425.

Candès, E. and Romberg, J. 2007 “Sparsity and incoherence in compressive sampling”, Inverse Prob., Vol. 23, No. 3, pp. 969-985.

Candès, E. and Wakin, M. 2008 “An introduction to compressive sampling”, IEEE Signal Processing Mag., Vol. 25, No. 2, pp. 21-30.

Claerbout, J.F. and Muir, F. 1973 “Robust modeling with erratic data”, Geophys. Mag., Vol. 38, No. 5, pp. 826-844.

Cohen, A., Dahmen, W. and DeVore, R. 2005 “Compressed sensing and best k-term approximation”, IEEE Trans. Inform. Theory, Vol. 53, No. 12, pp. 4203-4215.

Dai, W. and Milenkovic, O. 2008 “Subspace Pursuit for Compressive Sensing: Closing the Gap Between Performance and Complexity”, Arxiv preprint , pp. 803-811.

Donoho, D. and Stark, P. B. 1989 “Uncertainty principles and signal recovery”, SIAM J. Appl. Math., Vol. 49, No. 3, 906-931.

Donoho, D., Chen, S. and Saunders, M. 1998 “Atomic Decomposition by Basis Pursuit”, SIAM Journal on Scientific Computing, Vol. 20, pp. 33-61.

Donoho, D. 2006 “Compressed Sensing”, IEEE Trans. Inform. Theory, Vol. 52, No. 4, pp. 1289-1306.

Donoho, D. and Huo, X. 2006 “Uncertainty principles Near optimal signal recovery from random projection: Universal encoding strategies?”, IEEE Trans. Inform. Theory, Vol. 52, No. 12, pp. 5406-5425.

Donoho, D. and Tsaig, Y. 2006 “Fast Solution of L1-norm Minimization Problems when The Solution may be Sparse”, Tech. Rep., Stanford University Department of Statics.

Elad, M. 2007 “Optimized Projections for Compressed Sensing sampling and reconstruction of multiband signals”, IEEE Trans on Signal Processing.

Feng, P. and Bresler, Y. 1996 “Spectrum blind minimum rate sampling and reconstruction of multiband signals”, IEEE Int. Conf. Acoustics Speech Signal Processing, Vol. 2, pp. 1689-1692.

Haupt, J. and Nowak, R. 2006 “Signal reconstruction from noisy random projection”, IEEE Trans. Inform. Theory, Vol. 52, No. 9, pp. 4036-4048.

Jokar, S. and Pfetsch, M. 2007 “Exact and Approximate Sparse Solutions of Underdetermined Linear Equations”, ZIB-Report.

Rudelson, M. and Vershynin, R. 2005 “Geometric Approach to Error Correcting and Reconstruction of Signals”.

Santosa, F. and Symes, W.W. 1986 “Lineer inversion of bandlimited reflection seismograms”, SIAM J. Sci. Statist. Comput., Vol. 7, No. 4, pp. 1307-1330.

Takhar, D., Bansal, V., Wakin, M., Duarte, M., Baron, D., Kelly, K.F., and Baraniuk, R.G. 2006 “A compressed sensing camera: New Theory and an implementation using digital micromirrors”, Proc. Comp. Imaging IV SPIE Electronic Imaging, San Jose, CA.

Tropp, J. and Gilbert, A.C. 2007 “Signal recovery from partial information via orthogonal matching pursuit”, IEEE Trans. Inform. Theory, Vol. 53, No. 12, pp. 4655-4666.

Tropp, J. and Needell, D. 2008 “CoSaMP: Iterative Signal Recovery from Incomplete and Inaccurate Samples”, Arxiv preprint, arXiv:0803. pp. 2392.

Vetterli, M., Marzilliano, P. and Blu, T. 2002 “Sampling signals with finite rate innovation”, IEEE Trans. Signal processing, Vol. 50, No. 6, pp. 1417- 1428.

ÖZGEÇMİŞ

Adı Soyadı : Koray KARAKUŞ

Doğum Yeri : Isparta

Doğum Tarihi : 02.08.1983

Medeni Hali : Bekar

Yabancı Dili : İngilizce

Eğitim Durumu :

Lise :Isparta Anadolu Lisesi (2001)

Lisans :Ankara Üniversitesi Mühendislik Fakültesi Elektronik Mühendisliği Bölümü (2006)

Yüksek Lisans :Ankara Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Anabilim Dalı (Eylül 2008 – Ekim 2011)

Çalıştığı Kurumlar :

TÜBİTAK UZAY Teknolojileri Araştırma Enstitüsü (2007 - ...)

Yayınları :

Kapucu K., **Karakus K.**,Ismailoglu N., ve Oktem R.,"A JPEG 2000 Bit Plane Coder Implementation Combining Group of Column Skipping and Concurrent Membership Testing Strategies" Recent Advances in Space Technologies (RAST) 2011

Ismailoglu N., **Karakus K.**, Kapucu K., Yılmaz O., Mert M., Kazak E., ve Oktem R., "A Dual-Core ASIC Architecture for High Speed On-Board Image Compression with JPEG2000" Recent Advances in Space Technologies (RAST) 2011

Ismailoglu N., Vargun B., Sever R., Okcan B., **Karakus K.**, Kapucu K., ve Oktem R. ., "GEZGIN-2 Image Processing Subsystem of RASAT" Recent Advances in Space Technologies (RAST) 2011

Sunay H., Kırılmaz T., Dudak C., Sever R., Kahyaoglu N., **Karakus K**., Kapucu K., Bolucek M., Ismailoglu N.,ve Sen O.,"High Data Rate X-Band Transmitter for Low Earth Orbit Satellites" ABSCO International Symposium, Pattaya, Thailand, 2009