M.M.KÖSTEN, 2013

YÜKSEK LİSANS TEZİ

NİĞDE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

T.C
NİĞDE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALI

GERÇEK ZAMANLI UYGULAMALAR İÇİN KALMAN SÜZGECİ TABANLI TAKİP ALGORİTMALARININ FPGA ÜZERİNDE ETKİN GERÇEKLEŞTİRİLMESİ

MEHMET MUZAFFER KÖSTEN

TEMMUZ 2013

T.C.
NİĞDE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALI

GERÇEK ZAMANLI UYGULAMALAR İÇİN KALMAN SÜZGECİ TABANLI TAKİP ALGORİTMALARININ FPGA ÜZERİNDE ETKİN GERÇEKLEŞTİRİLMESİ

MEHMET MUZAFFER KÖSTEN

Yüksek Lisans Tezi

Danışman

Yrd. Doç. Dr. Fuat KARAKAYA

TEMMUZ 2013

Mehmet Muzaffer KÖSTEN tarafından **Yrd. Doç. Dr. Fuat Karakaya** danışmanlığında hazırlanan "**Gerçek Zamanlı Uygulamalar İçin Kalman Süzgeci Tabanlı Takip Algoritmalarının FPGA Üzerinde Etkin Gerçekleştirilmesi**" adlı bu çalışma jürimiz tarafından Niğde Üniversitesi Fen Bilimleri Enstitüsü **Elektrik-Elektronik Mühendisliği** Ana Bilim Dalı'nda Yüksek Lisans tezi olarak kabul edilmiştir.

Başkan : Doç. Dr. Nurettin ACIR (Niğde Üniversitesi)

Üye : Yrd. Doç. Dr. Cihan KARAKUZU (Bilecik Üniversitesi)

Üye : Yrd. Doç. Dr. Fuat KARAKAYA (Niğde Üniversitesi)

ONAY:

Bu tez, Fen Bilimleri Enstitüsü Yönetim Kurulunca belirlenmiş olan yukarıdaki jüri üyeleri tarafından …./…./20.... tarihinde uygun görülmüş ve Enstitü Yönetim Kurulu'nun …./…./20.... tarih ve ..... sayılı kararıyla kabul edilmiştir.

...../...../20...

Doç. Dr. Osman SİVRİKAYA
MÜDÜR

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin bilimsel ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Mehmet Muzaffer KÖSTEN

ÖZET

GERÇEK ZAMANLI UYGULAMALAR İÇİN
TAKİP ALGORİTMALARININ ALAN PROGRAMLANABİLİR
DİZİ KAPI ÜZERİNDE ETKİN UYARLANMASI

KÖSTEN, Mehmet Muzaffer
Niğde Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Ana Bilim Dalı

Danışman :Yrd. Doç. Dr. Fuat KARAKAYA

Temmuz 2013, 71 sayfa

Teknolojinin gelişmesi gündelik hayatta kullandığımız çoğu cihaz daha akıllı ve kullanıcı dostu haline gelmiştir. Bu gelişimin arka planına bakıldığında gelişmiş kontrol yöntemlerinin ve algoritmaların kullanılmaya başladığı görülmektedir. Özellikle bir sistemin durumunu kontrol etmek ve gözlemlemek gibi işlemlerde yoğun miktarda veri toplanmakta ve bu verilerin anlamlı bir şekilde takip edilerek yorumlanması gerekir. Fiziksel dünyadan alınan verilerin gerek ortam gürültüsü gerekse ölçüm yöntemlerinden kaynaklanan hatalar içermesi nedeni ile süzülmesi gerekmektedir. Bu gözleme ve süzme işlemi için pek çok yöntem geliştirilmiş olmakla birlikte günümüzde çoğu uygulama için Kalman Süzgeci ve onun genişletilmiş hali olan Genişletilmiş kalman süzgeci kullanılmaktadır. Bu çalışmada da bu yöntem FPGA üzerine Kayan Noktalı sayılar kullanılarak uygulanmış ve iki örnek problem üzerinde uyarlanarak FPGA üzerinde gerçekleştirilmiştir. İlk uygulama olarak eğik atış problemi seçilmiş olup, ikinci uygulama olarak da Eş Zamanlı Haritalandırma ve Konumlama probleminde kullanılan Genişletilmiş Kalman Süzgeci tercih edilmiştir. Tasarlanan yapılar Matlab üzerinde denenmiş ve daha sonra Xilinx ISE programında sentezlenmiştir. Elde edilen sonuçlar bilgisayar üzerinde MATLAB kullanılarak değerlendirilmiştir.

Anahtar Sözcükler: FPGA, Kalman Süzgeci, Genişletilmiş Kalman Süzgeci, Eş Zamanlı Haritalandırma ve Konumlama

SUMMARY

EFFECTIVE IMPLEMENTATION OF TRACKING ALGORITHMS
ON FIELD PROGRAMMABLE GATE ARRAYS FOR
REAL-TIME APPLICATION

KÖSTEN, Mehmet Muzaffer
Niğde University
Graduate School of Natural and Applied Science
Department of Electrical-Electronics Engineering

Supervisor :Assistant Professor Dr. Fuat KARAKAYA

July 2013, 71 pages

Thanks to the developments in the technology, many devices that we use in our daily life have become more useful. When we look at the backgrounds of these developments, we can see the usage of advanced control systems and algortihms. Especially, in the process of observing and controlling of a system huge amount of data may be collected and this data should be deciphered in a logical way. Because of the inconvenience of knowledge which is collected from the real world, both the noise of the environment and the errors which are caused by methods of measurement should be filtered. For this observing and filtering process, many different methods have been developed but the mostly used ones are Kalman Filtering and it's developed version Extended Kalman Filtering. And, in this study these two methods have been applied by using Floating Point Numbers on FPGA and implemented on it with the help of two example case. For the first implementation Projectile Motion Problem has been chosen and for the second implementation, Extended Kalman Filtering which is used in Simultaneous Localization and Mapping Problem has been applied. Designed structures have been tested on Matlab and then synthesized on Xilinx ISE Program. Obtained results have been evaluated with computer on Matlab.

Keywords: FPGA, Kalman Filter, Extended Kalman Filter, Simultaneous Localization and Mapping

TEŞEKKÜR

Bu tez çalışması sürecinde bana desteğini esirgemeyen, değerli danışmanım Sayın Yrd. Doç. Dr. Fuat KARAKAYA'ya; yüksek lisans eğitimim boyunca beraber çalıştığım tüm çalışma arkadaşlarıma; ve tüm bunların olmasını sağlayan maddi manevi desteklerini benden esirgemeyen aileme teşekkür ederim.

İÇİNDEKİLER

TABLO DİZİNİ

ŞEKİLLER DİZİNİ

RESİM DİZİNİ

KISALTMALAR

KS	Kalman Süzgeci
GKS	Genişletilmiş Kalman Süzgeci
FPGA	Alan Programlanabilir Kapı Dizileri
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
VHDL	VHSIC Donanım Tanımlama Dili
GPS	Global Positioning System

BÖLÜM I

GİRİŞ

1.1 Çalışmanın Amaç ve Kapsamı

Gelişen teknoloji ile birlikte gündelik yaşantımızda kullandığımız pek çok cihaz daha yetenekli, kullanımı ve yönetimi daha kolay bir hale geldi. Son kullanıcı açısından pek çok şey kolaylaşmış olsa da buna paralel olarak cihazlar giderek karmaşık hale gelmeye, daha karmaşık yönetim algoritmalarına ihtiyaç duymaya başladı. Bu algoritmalar daha verimli çalışabilmek ve daha doğru sonuçlar üretebilmek adına mümkün olan pek çok bilgi kaynağını kullanmaya dolayısıyla fiziksel dünyadan pek çok bilgiyi çeşitli algılayıcılar ve ölçüm yöntemleri ile alarak işlemeye başladılar.

Bu noktada fiziksel dünyadaki ortamların ideal olmaması, pek çok gürültü kaynağının mevcut olması ve kullanılan ölçüm yöntemlerinin ya da algılayıcıların fiziksel kısıtlamaları gibi nedenler yüzünden elde edilen verilerde hatalar oluşmaktadır. Bu sorunu aşmak adına çeşitli yöntemler denenmişse de günümüzde belki de en sık kullanılan ve neredeyse pek çok yere uygulanabilen yaklaşım 1960 yılında R. E. Kalman tarafından önerildi(Kalman 1960) Bu yaklaşım o kadar başarılı oldu ki geçtiğimiz 50 yıl içerisinde pek çok yere uyarlandı ve çeşitli geliştirmeler yapıldı.

Kalman Filtresi ilk halinde sadece doğrusal sistemler için uygundu ve gürültü modeli olarak sadece beyaz gürültü kullanılmıştı. Fakat gerçek hayat uygulamalarındaki sistemlerin çoğu doğrusal olmayan sistemlerdi. Bu ihtiyaçtan dolayı Kalman Filtresi doğrusal olmayan sistemler için uyarlandı. Bu konu ile ilgili ilk çalışmalar NASA tarafından yapıldı (Mc Gee, Schmidt, ve Smith 1962; McElhoe 1966). Kalman Süzgeci'nin doğrusal olmayan sistemlere uyarlanmış hali Genişletilmiş Kalman Süzgeci (Extended Kalman Filter) olarak isimlendirildi.

Kalman Süzgeci'nin doğrusal olmayan sistemlere de uyarlanabileceğinin gösterilmesinden sonra pek çok araştırmada GKS kullanılmaya başlandı. Fakat bu

noktada modellenen sisteme bağlı olarak GKS için gereken işlem yükü giderek artmaya başladı. Özellikle gerçek zamanlı uygulamalar söz konusu olduğunda bilgisayar üzerinde çalışan yazılımlardan ziyade donanımsal çözümler ön plana çıktı. FPGA teknolojisinin de gelişim göstermesiyle birlikte Kalman Süzgeci'nin ve Genişletilmiş Kalman Süzgeci'nin FPGA üzerinde uygulanması ile ilgili çalışmalar hız kazandı.

Dinamik işlemlerin durum uzayında kontrolünde kullanılan bir donanım önerisi Garbergs ve Sohlberg tarafından önerildi (Garbergs ve Sohlberg 1996). Daha sonra yapılan başka bir çalışmada Kalman Süzgeci'nin hızlandırılabilmesi için paralel bir mimari Lee ve Salcic tarafından önerilmiştir (Lee ve Salcic 1997). Başka bir çalışmada Turney ve diğerleri gerçek zamanlı video süzmek için kullanılacak FPGA tabanlı bir adaptif kalman süzgeci gerçekleştirmişlerdir (Turney, Reza, ve Delva 1999). Salcic ve Lee tarafından yapılan başka bir çalışmada ise FPGA tabanlı adaptif bir takip kestirim bilgisayarı gerçekleştirilmiştir (Salcic ve Lee 2001).

Charoensak ve Abeysekera ise yaptıkları çalışmada kalman süzgecini FM demodülasyon işleminde kullandıkları FPGA tabanlı bir yapı oluşturmuşlardır (Charoensak ve Abeysekera 2004; Charoensak ve Abeysekera 2005). Al-Dhaher ve diğerleri çoklu algılayıcılardan gelen verileri işleyen adaptif kalman süzgeci yapısı oluşturmuşlar ve matris çarpımları için FPGA kullanmışlardır (Al-Dhaher, Farsi, ve Mackesy 2005). Benzer bir diğer çalışmada Chappell ve arkadaşları birden fazla kaynaktan gelen verileri değerlendirmek için yine kalman süzgecinden faydalanmışlar ve işlemleri FPGA tabanlı bir sistem üzerinde gerçeklemişlerdir (Chappell ve diğerleri 2005).

Wei-Tsen Lin ve Dah-Chung Chang taşıyıcı senkronizasyonu ile ilgili yaptıkları çalışmalarda FPGA tabanlı kalman süzgeci yapılarını kullanmışlardır (Wei-Tsen Lin ve Dah-Chung Chang 2005; Wei-Tsen Lin ve Dah-Chung Chang 2006; Wei-Tsen Lin ve Dah-Chung 2007). Yang Liu ve diğerleri ise kalman süzgecinin verimli bir şekilde nasıl FPGA üzerine uyarlanabileceğini gösteren bir çalışma yapmışlardır (Yang Liu, Bouganis, ve Cheung 2007).

Kalman Süzgeci ve FPGA'nın beraber kullanıldığı bir diğer alan ise otonom mobil robotik alanıdır. Özellikle konumlama konusunda, GPS verilerinin işlenmesi konusunda Kalman Süzgeci ve Genişletilmiş Kalman Süzgeci neredeyse standart bir işlem adımı olmuştur. Bonato ve diğerleri çalışmalarında mobil robotik için Kalman Süzgecini FPGA üzerine uyarlamışlardır (Bonato, Marques, ve Constantinides 2007; Bonato ve diğerleri 2007). Zhi-Jian Sun ve Xue-Mei Liu çalışmalarında FPGA ve DSP tabanlı bir yapı kullanarak yönlendirme sisteminde kullanmışlardır (Zhi-Jian Sun ve Xue-Mei Liu 2008). Az yer kaplayan, az güç tüketen uygulamalar için FPGA ve DSP tabanlı entegre edilmiş bir GPS-INS sistemi de Agarwal ve diğerleri tarafından tasarlanmıştır (Agarwal, Arya, ve Bhaktavatsala 2009). Gömülü, araç yönlendirme sistemleri için başka bir çalışma da Islam ve diğerleri tarafından önerilmiştir (Islam, Langlois, ve Noureldin 2009). Mobil robotikte son dönemde insansız hava araçlarının (İHA) yaygınlaşması ile birlikte bu konudaki çalışmalar da hız kazanmıştır. Veera Ragavan ve diğerleri İHA da kullanılmak üzere FPGA tabanlı bir yapı oluşturmuşlar ve algılayıcılardan gelen verileri Kalman Süzgecini kullanarak işlemişlerdir (Veera Ragavan, Ganapathy, ve Xian 2009).

Bu çalışmaların yoğunlaştığı bir diğer konu ise Eş Zamanlı Haritalandırma ve Konumlama (Simultaneous Localization and Mapping- SLAM) konusu olmuştur. Araçların üzerinde bulunan algılayıcılardan elde edilen veriler genellikle Kalman Süzgeci ya da değiştirilmiş bir türeviyle işlenerek değerlendirilmekte, elde edilen bilgileri kullanılarak otonom aracın bulunduğu konumun haritasını çıkarması ve bu haritalandırdığı bölgeye göre nerede olduğunu kestirmesi sağlanmaya çalışılmaktadır. Bu sayede otonom araçların daha önceden bilinmeyen yerlerde de mümkün mertebe sorunsuz bir şekilde görevini yerine getirmesi beklenmektedir. Çok fazla değişken olması, alınan bilgilerin değerlendirilmesi için gereken işlem yükünün fazla olması ve çoğunlukla gerçek zamanlı işlenmesi gereken veriler olduğu için bu konu hala geliştirilmeye devam edilen bir alan olmuştur.

Bu konu ile ilgili öncü çalışmalar Smith ve diğerleri tarafından yapılmıştır (R. C. Smith ve Cheeseman 1986; R. Smith, Self, ve Cheeseman 1987). Bir diğer çalışma ise Leonard ve Durrant-Whyte tarafından gerçekleştirilmiştir (Leonard ve Durrant-Whyte 1991). Aradan geçen zaman rağmen SLAM algoritmalarındaki temel sorun olan

parametre sayısının artmasına bağlı olarak gereken işlem gücü gerekliliği henüz tam olarak aşılabilmiş değildir. Bu işlemleri hızlandırmak için Idris ve diğerleri paralel matris çarpımı yapan bir yapı önermişlerdir (Idris ve diğerleri 2010). Yaptıkları başka bir çalışmada ise Monocular SLAM EFK işlemleri için bir yardımcı işlemciyi FPGA ile tasarlamışlardır (Idris ve diğerleri 2012). Diğer bir çalışmada Rosa ve Bonato gömülü robotik uygulamalar için kayan noktalı sayılar kullanan kalman süzgecini sabit noktalı sayı kullanan forma dönüştürmek için bir yöntem önermişlerdir (Rosa ve Bonato 2012). Bu konu ile ilgili bir diğer gelişme ise yongada sistem çözümlerinin başarımlarının artması ile yaşanmıştır. Steux ve El Hamzaoui tinySLAM algoritmasını ARM9 mimarili bir yongada sistem çözümünde çalıştırmışlardır (Steux ve El Hamzaoui 2010). Gonzalez ve diğerleri ise benzer bir çalışmada gerçek zamanlı monocular SLAM uygulaması için geliştirdikleri yöntemi 3 farklı yongada sistem çözümü üzerinde gerçeklemiş ve karşılaştırmalarını yapmışlardır (Gonzalez, Codol, ve Devy 2011). Vincke ve diğerleri ise ARM ve DSP mimarisini beraberce kullanarak melez bir yapı tasarlamışlardır (B. Vincke, Elouardi, ve Lambert 2011; Bastien Vincke, Elouardi, ve Lambert 2012).

Bu çalışmada ise Kalman Süzgeci'nin FPGA üzerine mümkün olduğunca genelleştirilmiş bir şekilde uyarlanması üzerinde durulmuştur. Daha önce yapılan çalışmalar belli bir sorunun çözümüne yönelik olarak iyileştirilmiş yaklaşımlar sunmakla beraber genel kullanım amacından uzaklaşmaktadır. Bu bir açıdan beklenen bir sonuçtur. FPGA gibi belli miktarda kaynağın kullanılabileceği bir yapıda probleme özel çözümler üretmek kaynak kullanımında ve işlem sürelerinde iyileştirmeler sağlayabilir. Fakat söz konusu kolay uyarlanabilirlik, tekrar kullanılabilirlik ve ön çalışmaların hızlıca yapılabilmesi söz konusu olduğunda bu yöntem çok verimli olmamaktadır.

Bahsedilen kısıtlamaları aşmak için bu çalışmada mümkün olduğunca genelleştirilmiş bir yapı oluşturulmaya çalışılmıştır. Bunu sağlamak adına, oluşturulan yapı alt seviyeden tasarlanmaya başlanarak hiyerarşik bir hale getirilmiş, parçadan bütüne bir tasarım anlayışı izlenmiştir. Bu sayede farklı firmaların FPGA yapıları kullanılırken alt seviyedeki işlem birimlerinde yapılacak ufak değişikliklerle uyumluluk yakalanmış olacaktır. Bu sayede problemlerin uyarlanması ve prototiplendirme aşamasında ciddi bir

zaman kazanımı mümkün hale getirilmiştir.

Farklı durumlara uygun şekilde değiştirilebilmesi için genelleştirilmiş parametreler ile matrislerin boyutları, sayı sisteminin bit uzunluğu ve yapılacak matris işlemlerinin seri, paralel ya da tam paralel şekilde çalışması gibi işlevlerin seçimleri kullanıcıya bırakılmıştır.

1.2 Bölüm İçerikleri

Gerçeklenen yapının denenmesi için ilk olarak basit bir eğik atış problemi seçilmiştir. Daha sonra yine aynı iskelet kullanılarak , günümüzde oldukça çok kişi tarafından üzerinde çalışılan Eş Zamanlı Haritalandırma ve Konumlama (Simultaneous Localization and Mapping – SLAM) uygulamasına yönelik bir yapı oluşturulmuştur.

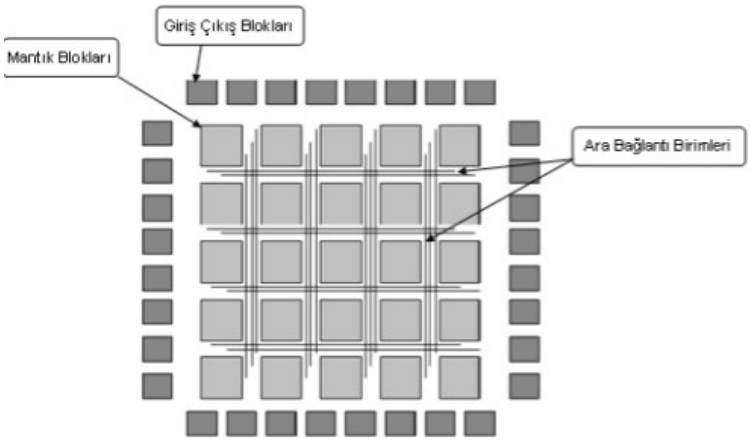
Bu çalışmanın ikinci bölümünde kullanılan deney kartı ve yazılımlar ve sayı sistemleri hakkında bilgi verilmiştir. Bir sonraki bölümde tasarım için kullanılan alt bileşenler anlatılmış olup sentez sonuçları ve mimarileri hakkında bilgilendirme yapılmıştır. Dördüncü bölümde ise takip algoritmaları hakkında teorik bilgiler verilmiş ve beşinci bölümde bu teorik bilgiye dayanarak örnek problemler donanım üzerine uygulanmıştır. Altıncı bölümde ise çalışma hakkında genel bir değerlendirme yapılmıştır.

BOLÜM II

ALTYAPI

2.1 Kullanılan Deney Kartı ve Yazılım

Bu çalışmada donanım altyapısı olarak Xilinx Virtex5 LX110-T FPGA'sı kullanılmıştır. FPGA'lar sahip oldukları esnek mimari sayesinde sayısal olarak gerçeklenebilen pek çok tasarımın kolayca uyarlanabilmesine olanak sağlamaktadırlar. Basit olarak içerisinde bulunan mantık elemanlarının istenen amaca göre birleştirilmesi ile herhangi bir sayısal devre FPGA üzerinde gerçeklenebilmektedir. Şekil 2.1'de temel anlamda bir FPGA'nın yapısı gösterilmiştir.



FPGA üzerinde tasarım yapılırken sıklıkla kullanılan iki dil mevcuttur. Bunlar VHDL ve Verilog dilleridir. Bu diller sayesinde sayısal tasarımlar kolay bir şekilde FPGA üzerine aktarılabilmektedir. Bu çalışmada tercih edilen dil ise VHDL olmuştur.

Yazılım geliştirme ve test işlemleri Xilinx ISE Design Suite 14.4 üzerinde gerçekleştirilmiş olup, algoritma testleri Matlab üzerinde yapılmıştır. Piyasada var olan FPGA modellerinin hepsine uyumluluk sağlamak adına tüm tasarımlarda mecbur

kalınmadıkça üreticiye özel kütüphaneler kullanılmamıştır. Bu sayede tek bir FPGA modeline bağlı kalmadan kullanılabilen bir yapı geliştirilmiştir.

2.2 Sayı Sistemleri

Sayısal sistemler üzerinde işlem yapılırken sıklıkla kullanılan iki sayı sistemi mevcuttur. Gerçel sayıları elektronik ortamda ifade ederken bu sistemlerden faydalanırız.

2.2.1 Kayan Noktalı Sayılar

Kayan noktalı sayı formatı, gerçek sayıların bilgisayar ortamında gösterim şeklidir. Kayan noktalı sayı gösterimi sabit noktalı sayılara göre daha geniş sayı aralığında işlem yapmakta fakat buna karşılık ihtiyaç duyduğu donanım kaynağı miktarı yüksek olmaktadır. Bu sayı formatı IEEE-754 standardı olarak da kabul edilmektedir. Bu standartlar çerçevesinde iki kullanım biçimi mevcuttur.

- Tek Duyarlı Gösterim (32-Bit)
- Çift Duyarlı Gösterim (64-Bit)

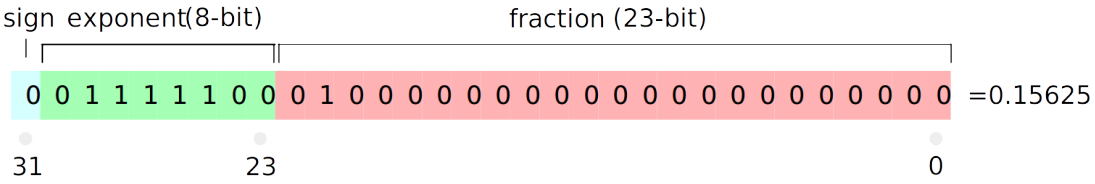
Kayan Noktalı Sayıların ifade biçimi ise Eşitlik 2.1 ve 2.2 ile verilmiştir.

$$Sayı = (-1)^{s}(1.\mathrm{f})2^{e-bias} \tag{2.1}$$

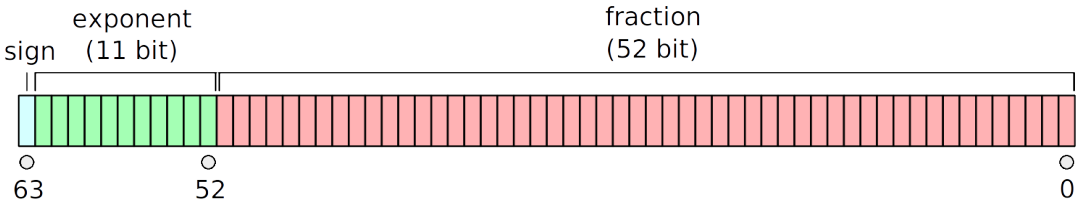
$$e-bias = floor\left(\log_2^{sayı}\right) \tag{2.2}$$

- "**s**" ifadesi işaret bitini göstermektedir. İfadenin değeri 0 ise sayı pozitif, 1 ise negatiftir.
- "**bias**" ifadesi IEEE754 standardında tanımlanmış olup, üs sayısından çıkarılan değeri ifade etmektedir. 32 Bit gösterim için bu değer 127 olarak ifade edilmiştir.
- "**f**" ifadesi ise çarpan olarak verilmiştir.

Özetlemek gerekirse kayan noktalı sayı gösteriminde en anlamlı bit işaret biti olarak atanmış, ondan sonra gelen sekiz bit üssel kısım için atanmış ve kalan 23 bit ise ondalık kısmın gösterimi için ayrılmıştır. Şekil 2.2 ve 2.3'de tek hassasiyetli ve çift hassasiyetli kayan noktalı sayılara ait gösterim verilmiştir.



Şekil 2.2 Tek duyarlılıklı kayan noktalı sayı gösterimi.



Şekil 2.3 Çift duyarlılıklı kayan noktalı sayı gösterimi.

2.2.2 Sabit Noktalı Sayılar

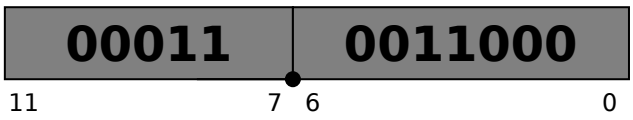
Sabit noktalı sayı formatı sayının tam kısmı ve ondalık kısmının beraberce gösterimi ile elde edilir. Bu sayı sisteminde tasarlanan yapının ihtiyacına göre azami ve asgari sayı büyüklüğü, istenen hassasiyet gibi bileşenler göz önüne alınarak bit uzunlukları belirlenir. Kayan noktalı sayılardaki gibi karmaşık bir gösterim biçimine sahip olmadığı için son derece kolay bir şekilde sayısal devrelerde uyarlanabilir. Sabit noktalı sayı gösterimi Eşitlik 2.3 ile verilmiştir.

$$Sayı = n * 2^{-m} \tag{2.3}$$

- "**n**" n bitlik sayının tam sayı değerini ifade etmektedir.
- "**m**" ise sayının ondalık kısmının bit uzunluğudur.

Sabit noktalı sayılarda, gösterilen sayı pozitif ise tam kısım ikilik tabanda ifade edilir. Ondalık kısmın gösterimi ise ikilik tabanda tam olarak ifade edilemeyeceği için kullanılan bit sayısına bağlı olarak hata meydana gelir. Tasarlanan yapıya bağlı olarak istenilen hassasiyet oranına göre bu bit uzunluğu belirlenmelidir.

Sabit noktalı sayılarda sayının işareti en anlamlı bite bakılarak belirlenir. Eğer bu değer 0 ise sayı pozitif, 1 ise negatiftir. Negatif sayıların gösteriminde ise birden fazla yöntem kullanılabilir. Bunlar normal gösterim, bire tümleyen ve ikiye tümleyen şeklindedir. Örnek bir sabit noktalı sayı gösterimi Şekil 2.4'te verilmiştir.



Şekil 2.4 Sabit noktalı sayı gösterimi.

Daha öncede belirtildiği üzere sabit noktalı sayılarda ondalık kısımların ifadesinde kullanılan bit sayısının, ifade edilmeye çalışılan değerin doğruluğunda doğrudan etkisi olacağından sabit noktalı sayılarla çalışılırken bit uzunluğu seçimine ayrıca bir önem gösterilmelidir.

Bu çalışmada mümkün mertebe genelleştirilmiş bir yapı elde edilmeye çalışıldığı için sabit noktalı sayılar yerine kayan noktalı sayılarla çalışılmıştır. Bu tercih donanım kaynak tüketiminin artmasına neden olmuşsa da, sabit noktalı sayılara dayalı bir yapının genel olarak uyarlanabilmesi için gereken ön çalışmalardan kaçınarak kolay uyarlanabilirlik adına fayda sağlamıştır.

BÖLÜM III

TASARIM

3.1 Yazılım Kütüphanesi

Bu çalışmada gerçekleştirilen yapının tasarımında hiyerarşik bir tasarım oluşturulmuştur. İşlemler için gereken birimler ayrı ayrı tasarlanarak bir üst katmanda birleştirilmiştir. Bu tasarımlar yapılırken VHDL dili kullanılmış ve bir önceki bölümde anlatılan kayan noktalı sayılarla çalışılmıştır. Tasarımın farklı üreticilerin ürettiği FPGA modelleri üzerinde de çalışabilmesi adına kayan noktalı sayı işlemleri için dahili donanımsal çözümler kullanılmamış bunun yerine uyumluluğu arttırmak adına IEEE STD 1076-2008 standardınca belirlenmiş VHDL-2008 kütüphaneleri tercih edilmiştir. Fakat bazı tasarım araçlarının bu standardı desteklemediği de göz önüne alınarak VHDL 2008 kütüphanelerinin VHDL-93 standardına göre uyarlanmış sürümleri kullanılmıştır.

Kalman süzgeci matematiksel işlemleri için matrisler kullanmakta olduğu için kayan noktalı sayılar kullanan ve matrisler üzerinde işlemler yapan bileşenler FPGA üzerinde gerçeklenmiştir.

3.2 Matris Toplama İşlemi

Matrisler üzerinde toplama işlemi her bir elemanla karşılıklı olarak yapılır. İki matrisin toplanabilmesi için satır ve sütün sayılarının eşit olması gerekmektedir. Eşitlik 3.1'de bu işlemin matematiksel ifadesi verilmiştir.

$$A=[a_{ij}]_{mxn},\ B=[b_{ij}]_{mxn}\ olmak\ üzere\ A+B=[a_{ij}+b_{ij}]_{mxn}=C \tag{3.1}$$

İşlemi gerçekleştirmek için **m_topla** adında bir yapı oluşturulmuştur. Yapıya ait tanımlama bilgileri aşağıda verilmiştir.

```
entity m_topla is
  generic (
    secim : mimari  := yp;
    satir : integer := 2;
    sutun : integer := 2);
  port (
    clk   : in  std_logic;                              -- Saat kaynagi.
    rst   : in  std_logic;                              -- Reset ucu
    ena   : in  std_logic;                              -- Etkinlestirme Ucu
    mat_a : in  matris(0 to satir-1, 0 to sutun-1);     -- MAT_A girisi.
    mat_b : in  matris(0 to satir-1, 0 to sutun-1);     -- MAT_B
    sonuc : out matris(0 to satir-1, 0 to sutun-1);     -- Sonuc cikisi.
    hazir : out std_logic);                             -- Islem tamam cikisi.
end m_topla;
```

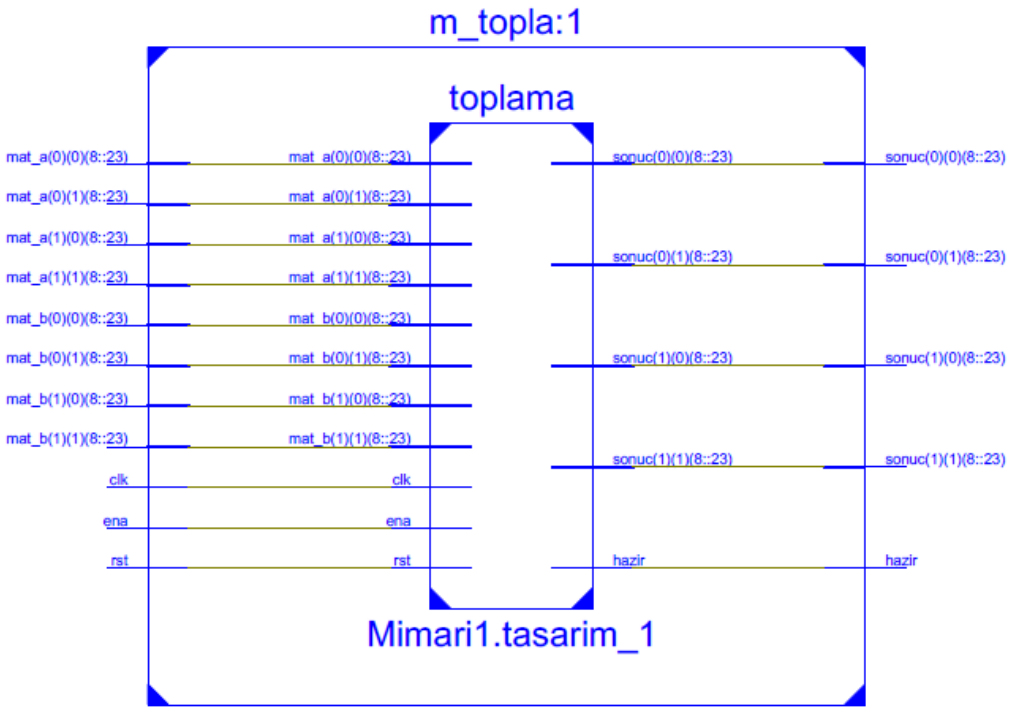
Tasarım yapılırken ölçeklenebilirlik adına 3 genel ifade tanımlanmıştır. “**satir**” ve “**sutun**” değerleri işlem yapılacak matrisin boyutlarını belirlerken “**secim**” değeri oluşturulacak yapının işlem mimarisini belirlemektedir. 3 farklı işlem mimarisi tasarlanmıştır.

Bunlar sıralı işlem, yarı paralel ve tam paralel mimariler olarak belirlenmiştir. Ayrıca kullanılan kütüphanelerin sağladığı esneklik sayesinde IEEE754 standardının yanında başka bit uzunluklarındaki kayan noktalı sayılarla da işlem yapılabileceği için tasarımda bu durum da göz önüne alınmıştır.

Sıralı mimariye ait kaynak tüketim bilgileri Tablo 3.1'de , tasarımın genel görünümü ise Şekil 3.1'de verilmiştir. Aynı şekilde yarı paralel tasarımın kaynak kullanım bilgileri Tablo 3.2'de ve genel mimari görünümü Şekil 3.2'de, tam paralel tasarımın kaynak kullanım bilgileri Tablo 3.3'de ve genel mimari görünümü Şekil 3.3'de verilmiştir.

Tablo 3.1 Matris Toplama İşlemi Sıralı Mimari Donanım Kaynak Tüketimi

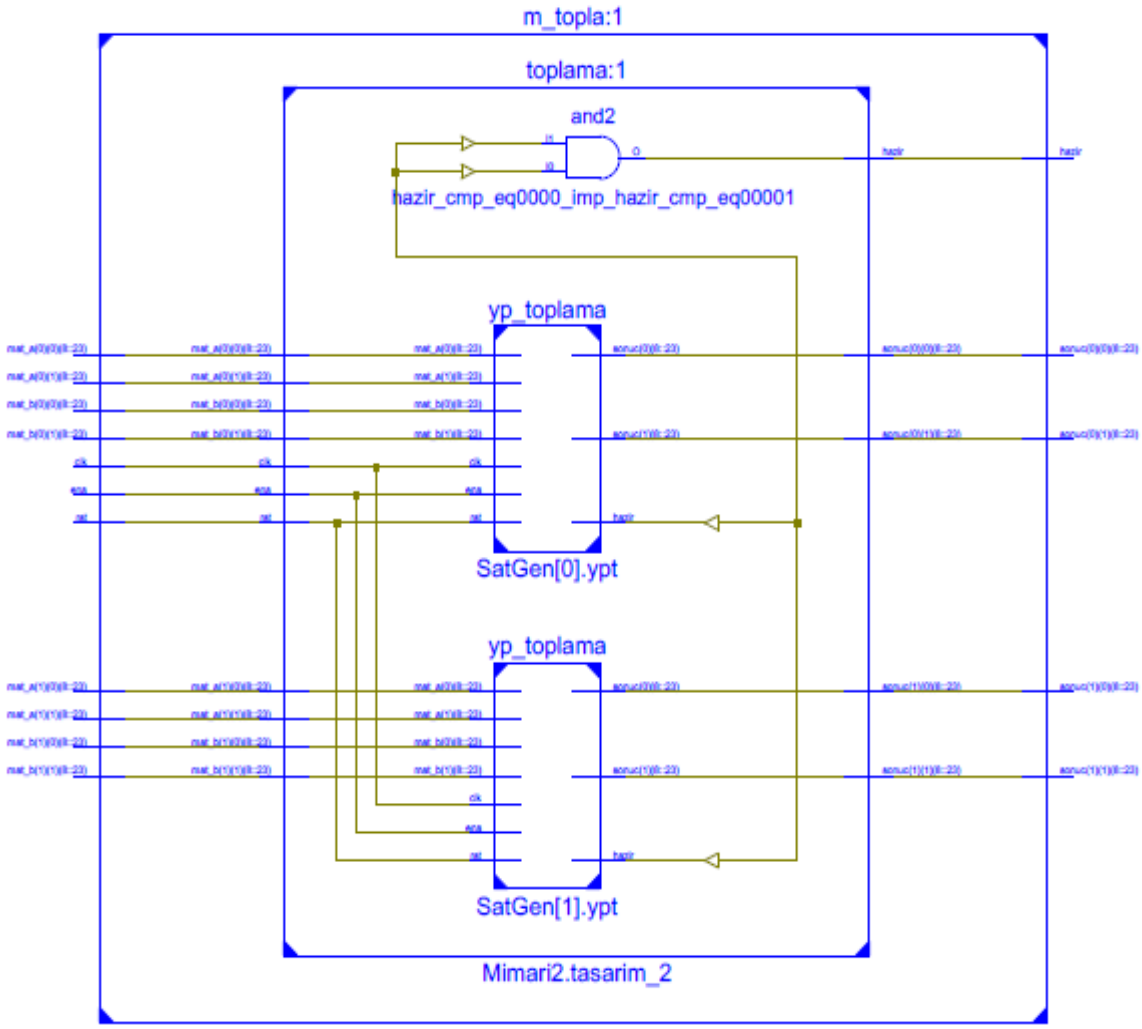
Matris Boyutu	Sıralı																	
	20 Bit (8+12)						24 Bit (8+16)						32 Bit (8+24)					
	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)
4x4	21	328	1	783	2	71,11	21	392	1	986	2	69,349	21	519	1	1900	3	62,384
5x5	31	507	1	1059	2	65,979	31	607	1	1211	2	67,516	31	807	2	2220	4	60,33
6x6	43	730	2	1159	2	67,629	43	872	2	1426	3	64,787	43	1170	2	2604	4	60,826



Şekil 3.1 Matris Toplama İşlemi Sıralı Mimari Görünümü

Tablo 3.2 Matris Toplama İşlemi Yarı Paralel Mimari Donanım Kaynak Tüketimi

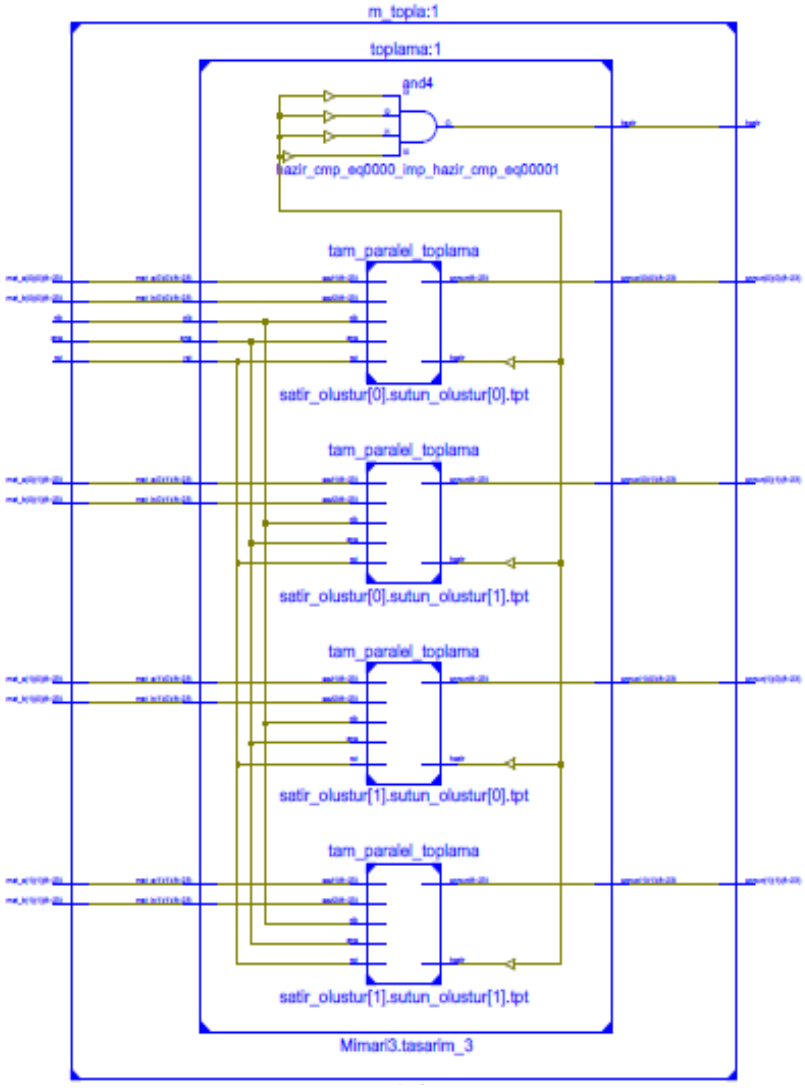
Matris Boyutu	Yarı Paralel																	
	20 Bit (8+12)						24 Bit (8+16)						32 Bit (8+24)					
	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)
4x4	5	340	1	1885	3	76,351	5	404	1	2393	4	77,018	5	528	1	5333	8	65,13
5x5	6	520	1	2316	4	74,062	6	620	1	3046	5	75,387	6	820	2	6207	9	62,805
6x6	7	744	2	2905	5	79,54	7	888	2	3793	6	73,887	7	1176	2	7395	11	63,211



Şekil 3.2 Matris Toplama İşlemi Yarı Paralel Mimari Görünümü

Tablo 3.3 Matris Toplama İşlemi Tam Paralel Mimari Donanım Kaynak Tüketimi

Matris Boyutu	Tam Paralel																	
	20 Bit (8+12)						24 Bit (8+16)						32 Bit (8+24)					
	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)
4x4	1			4512	7		1			7392	11		1			15775	23	
5x5	1			7050	11		1			11550	17		1			24649	36	
6x6	1			10152	15		1			16663	25		1			35494	52	



Şekil 3.3 Matris Toplama İşlemi Tam Paralel Mimari Görünümü

3.3 Matris Çıkarma İşlemi

Matrisler üzerinde çıkarma işlemi her bir elemanla karşılıklı olarak yapılır. Bir matrisin diğerinden çıkarılabilmesi için satır ve sütün sayılarının eşit olması gerekmektedir. Eşitlik 3.2'de bu işlemin matematiksel ifadesi verilmiştir.

$$A=[a_{ij}]_{mxn}, B=[b_{ij}]_{mxn} olmak\ üzere\ A-B=[a_{ij}-b_{ij}]_{mxn}=C \quad (3.2)$$

İşlemi gerçekleştirmek için **m_cikar** adında bir yapı oluşturulmuştur. Yapıya ait tanımlama bilgileri aşağıda verilmiştir.

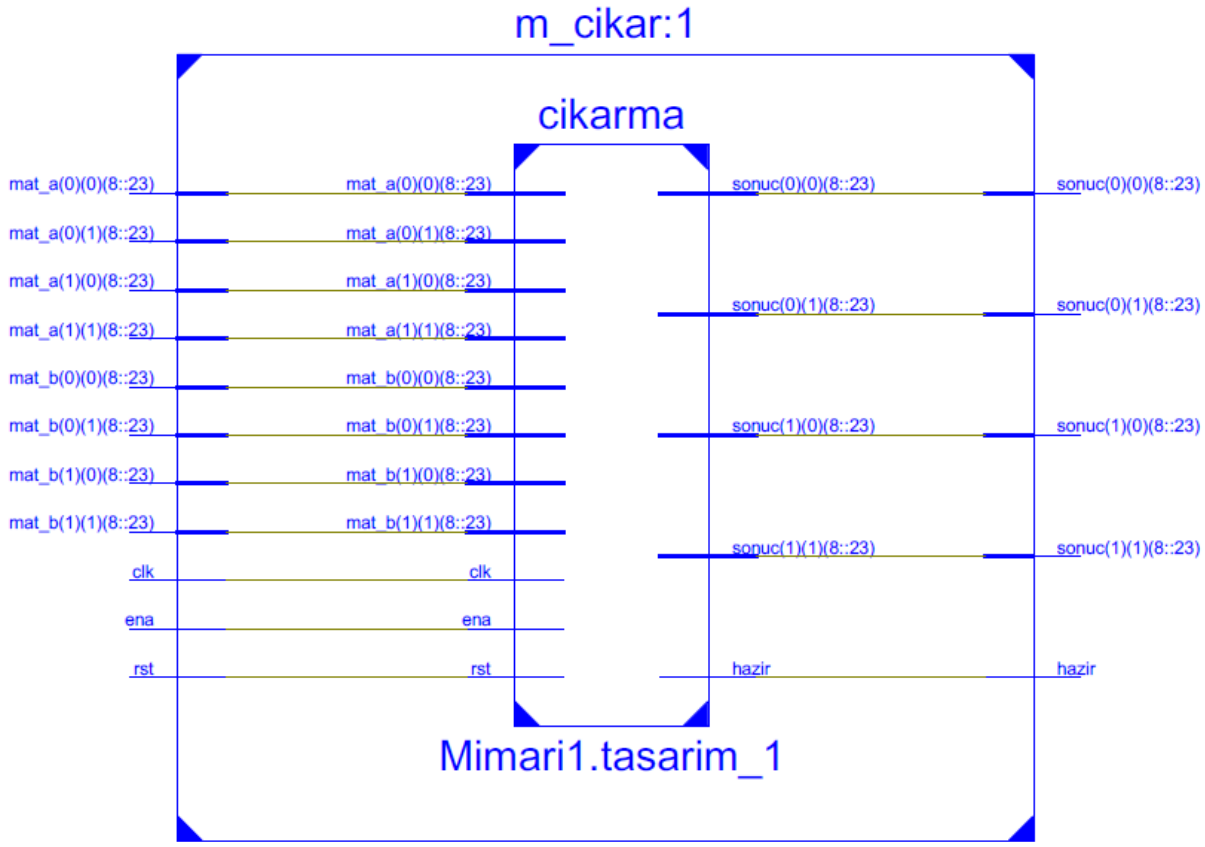
```
entity m_cikar is
  generic (
    secim : mimari  := tpt;
    satir : integer := 2;
    sutun : integer := 2);
  port (
    clk   : in  std_logic;                              -- Saat kaynagi.
    rst   : in  std_logic;                              -- Reset ucu
    ena   : in  std_logic;                              -- Etkinlestirme Ucu
    mat_a : in  matris(0 to satir-1, 0 to sutun-1);     -- MAT_A girisi.
    mat_b : in  matris(0 to satir-1, 0 to sutun-1);     -- MAT_B
    sonuc : out matris(0 to satir-1, 0 to sutun-1);     -- Sonuc cikisi.
    hazir : out std_logic);                             -- Islem tamam cikisi.
end m_cikar;
```

Toplama işleminde olduğu gibi bu işlem biriminde de benzer bir tasarım yapılmıştır. Ölçeklenebilirlik adına boyut bilgisi ve işlemin sıralı mı, yarı paralel mi yoksa tam paralel mi yapılacağı seçilebilecek bir şekilde tasarlanmıştır.

Kullanılan kütüphane sayesinde istenilen bit uzunluğunda çalışan bir yapı oluşturulabilmektedir. Şekil 3.4, Şekil 3.5 ve Şekil 3.6'da sırsı ile tüm mimarilere ait şemalar verilmiştir. Her mimariye ait kaynak tüketim bilgileri de sırası ile Tablo 3.4, Tablo 3.5 ve Tablo 3.6'da verilmiştir.

Tablo 3.4 Matris Çıkarma İşlemi Sıralı Mimari Donanım Kaynak Tüketimi

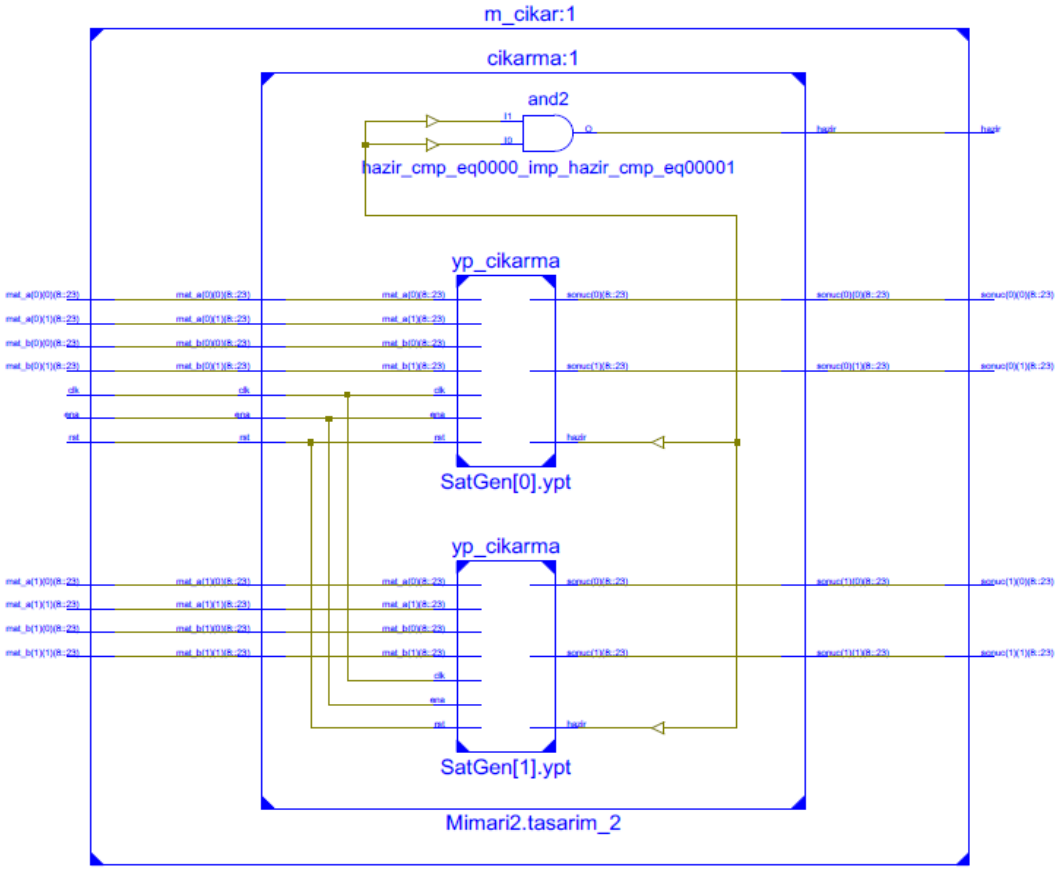
Matris Boyutu	Sıralı																	
	20 Bit (8+12)						24 Bit (8+16)						32 Bit (8+24)					
	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)
4x4	21	328	1	784	2	73,8	21	392	1	990	2	70,904	21	519	1	1899	3	63,527
5x5	31	508	1	956	2	71,25	31	608	1	1212	2	67,197	31	807	2	2178	4	59,269
6x6	43	731	2	1187	2	70,582	43	878	2	1455	3	65,71	43	1165	2	2663	4	61,273



Şekil 3.4 Matris Çıkarma İşlemi Sıralı Mimari Görünümü

Tablo 3.5 Matris Çıkarma İşlemi Yarı Paralel Mimari Donanım Kaynak Tüketimi

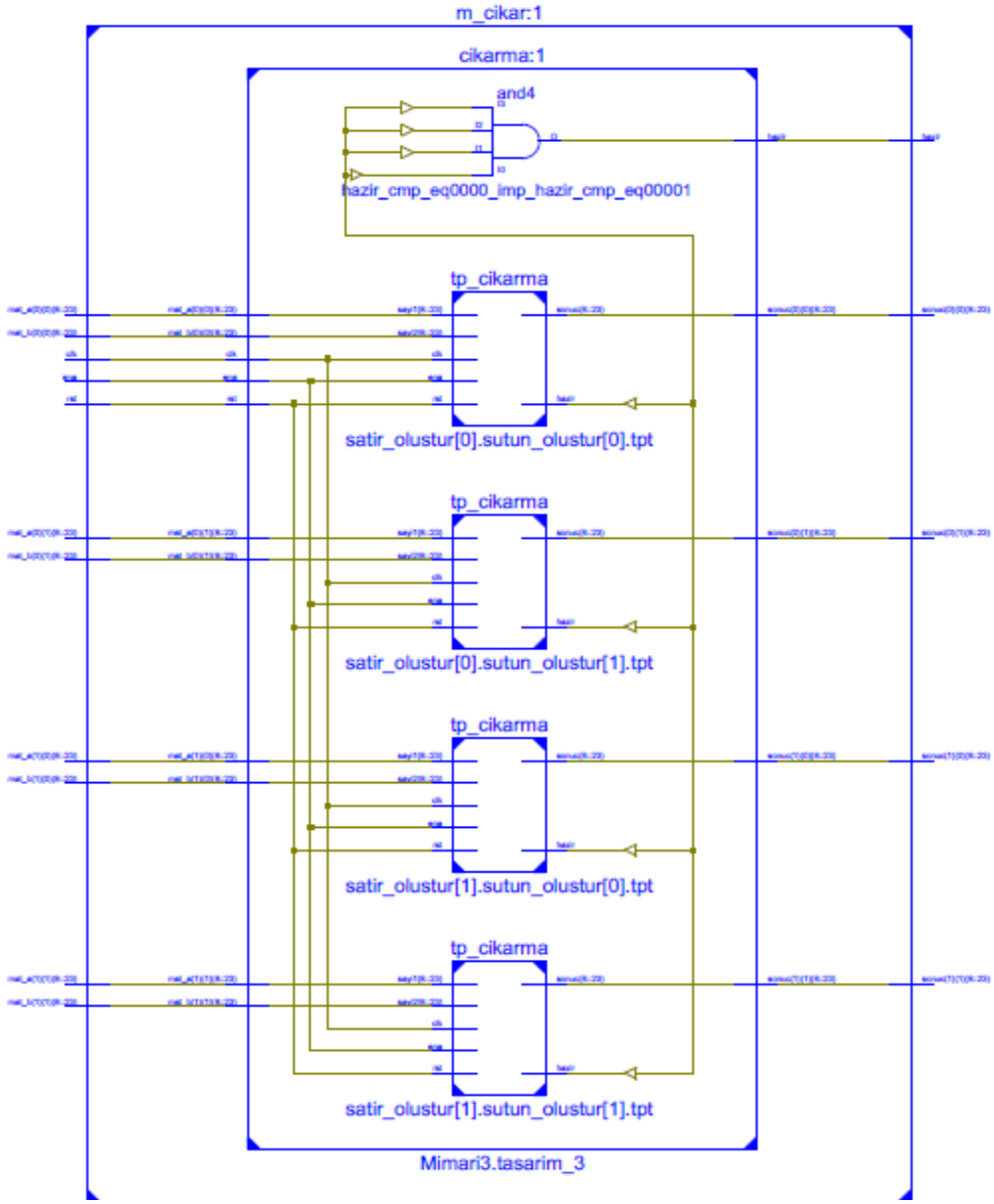
Matris Boyutu	Yarı Paralel																	
	20 Bit (8+12)						24 Bit (8+16)						32 Bit (8+24)					
	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)
4x4	5	336	1	1896	3	76,892	5	400	1	3197	5	74,367	5	528	1	5213	8	66,559
5x5	6	520	1	2456	4	77,355	6	620	1	4169	7	66,844	6	820	2	6059	9	62,171
6x6	7	744	2	3148	5	76,758	7	888	2	4845	8	67,156	7	1176	2	7261	11	59,788



Şekil 3.5 Matris Toplama İşlemi Yarı Paralel Mimari Görünümü

Tablo 3.6 Matris Çıkarma İşlemi Tam Paralel Mimari Donanım Kaynak Tüketimi

Matris Boyutu	Tam Paralel																	
	20 Bit (8+12)						24 Bit (8+16)						32 Bit (8+24)					
	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)
4x4	1			4960	8		1			7284	11		1			15614	23	
5x5	1			7725	12		1			11401	17		1			24397	36	
6x6	1			11160	17		1			16417	24		1			35132	51	



Şekil 3.6 Matris Toplama İşlemi Tam Paralel Mimari Görünümü

3.4 Matris Çarpma İşlemi

Matris çarpma işlemi toplama ve çıkarma işlemine göre daha karmaşık bir algoritmaya sahiptir. İki matrisin çarpılabilmesi için ilk matrisin sutun sayısı ikinci matrisin satır sayısına eşit olmadır. Buna göre A_{mxn} boyutlu B_{nxk} boyutlu matrisler olmak üzere;

$$A_{mxn} \times B_{nxk} = C_{mxk} \tag{3.3}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{3.4}$$

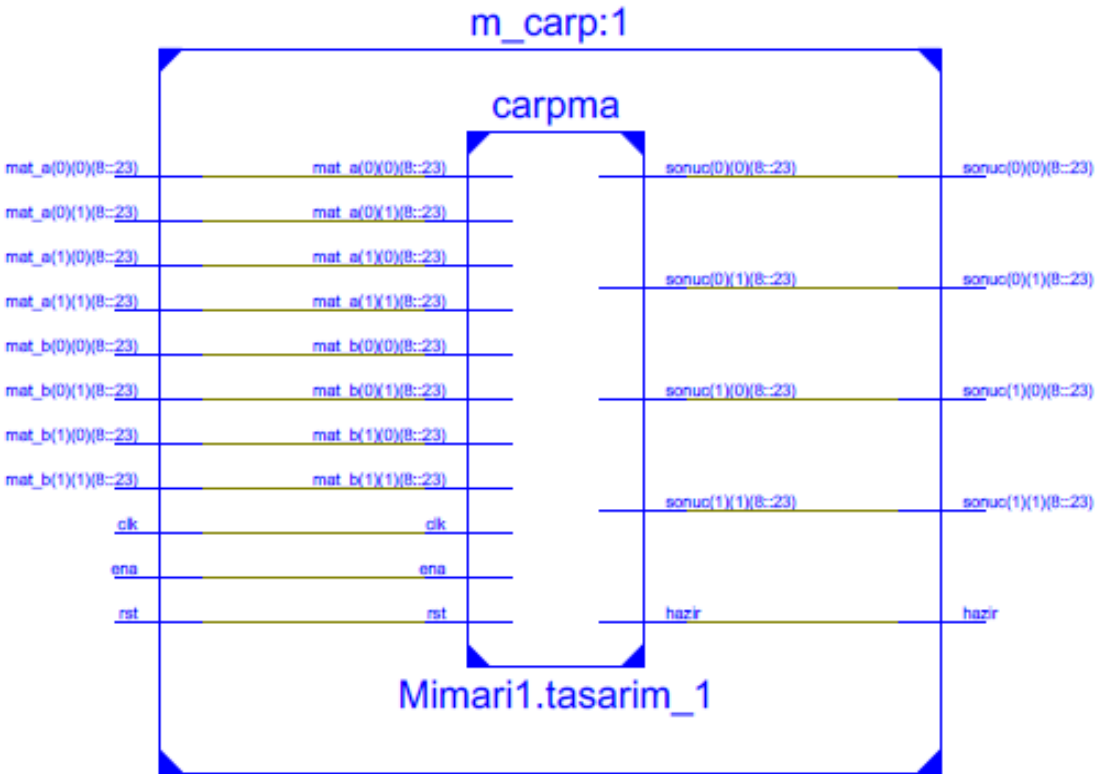
Bu işlemleri gerçekleştirmek için gereken yapı **m_carp** adıyla tasarlanmıştır. Yapıya ait tanımlama bilgileri aşağıda verilmiştir.

```
entity m_carp is
  generic (
    secim : mimari  := yp;                                  -- Mimari Secimi
    sat_a : integer := 4;                                   -- MAT_A Satir Boyutu.
    sut_a : integer := 4;                                   -- MAT_A Sutun Boyutu.
    sat_b : integer := 4;                                   -- MAT_B Satir Boyutu.
    sut_b : integer := 4);                                  -- MAT_B Sutun Boyutu.
  port (
    clk   : in  std_logic;                                  -- Saat kaynagi.
    rst   : in  std_logic;                                  -- Reset ucu.
    ena   : in  std_logic;                                  -- Etkinlestirme girisi.
    mat_a : in  matris(0 to sat_a-1, 0 to sut_a-1);         -- MAT_A Girisi.
    mat_b : in  matris(0 to sat_b-1, 0 to sut_b-1);         -- MAT_B Girisi
    sonuc : out matris(0 to sat_a-1, 0 to sut_b-1);         -- Sonuc cikisi.
    hazir : out std_logic
    );                                                      -- Islem tamam cikisi.
end m_carp;
```

Toplama ve çıkarma işleminden farklı olarak bu işlem biriminde fazladan girişler eklenmiştir. Çarpılan matrislerin boyutları farklı olabileceği için matrislere ait boyut bilgileri ayrı bir şekilde tanımlanmıştır. Bu işlem birimi kullanılırken matris çarpımının en temel kuralı olan birinci matrisin sütun boyutu ile ikinci matrisin satır boyutlarının aynı olması gerektiği kuralına uyulması gerekmektedir. Tasarlanan birime ait mimari şemaları Şekil 3.7 ve 3.8'de verilmiş olup ilgili birime ait ölçüm değerler Tablo 3.7 ve 3.8'de bulunmaktadır.

Tablo 3.7 Matris Çarpma İşlemi Sıralı Mimari Donanım Kaynak Tüketimi

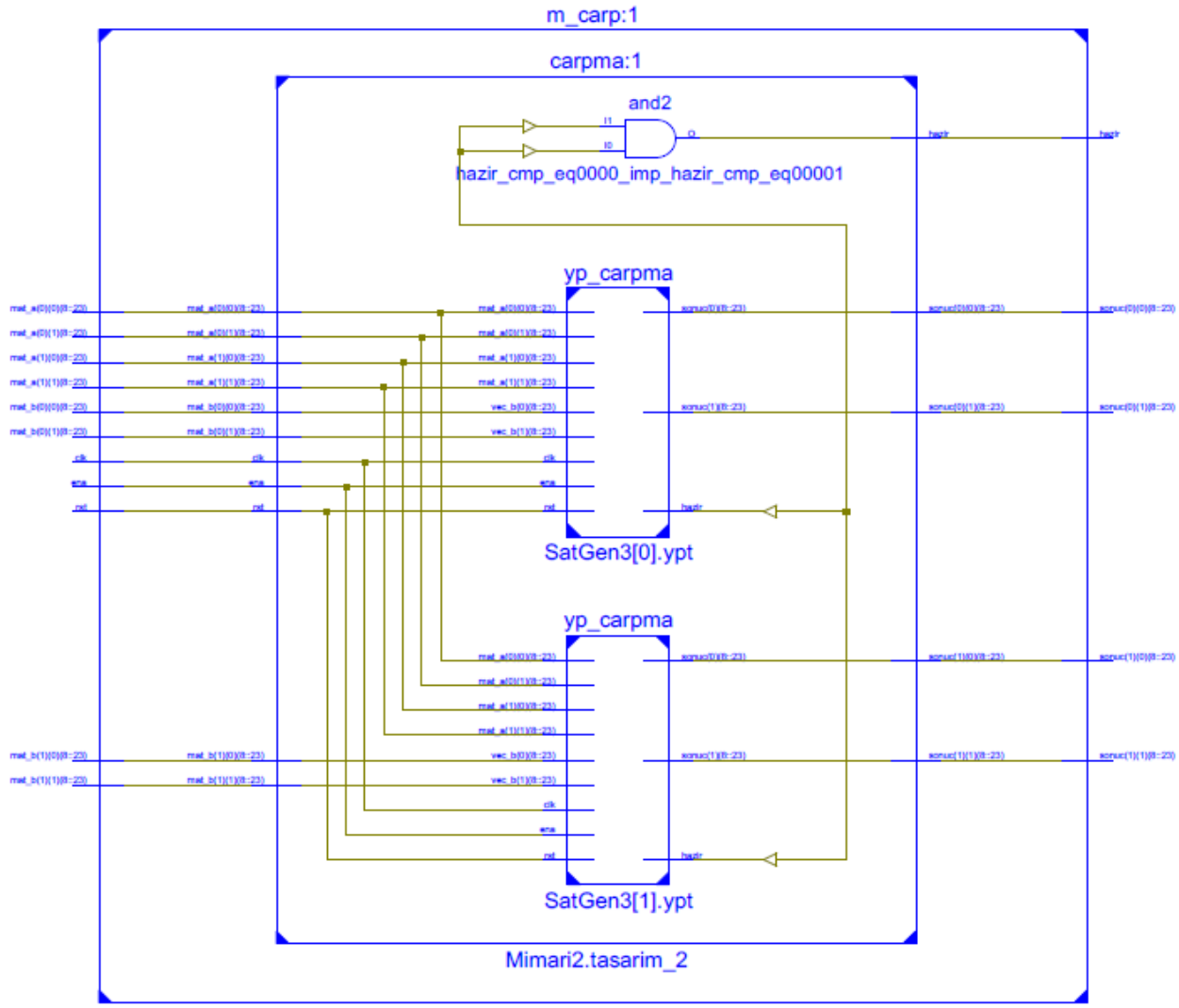
Matris Boyutu	Sıralı																	
	20 Bit (8+12)						24 Bit (8+16)						32 Bit (8+24)					
	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)
4x4	214	435	1	784	2	82,387	214	519	1	1128	2	80.181	214	687	1	2113	4	73,924
5x5	407	615	1	1142	2	77,721	407	735	2	1368	2	80,314	407	975	2	2352	4	73,627
6x6	692	835	2	1187	2	80,37	692	999	2	1526	3	81,591	692	1327	2	2624	4	74,055



Şekil 3.7 Matris Çarpma İşlemi Sıralı Mimari Görünümü

Tablo 3.8 Matris Çarpma İşlemi Yarı Paralel Mimari Donanım Kaynak Tüketimi

Matris Boyutu	Yarı Paralel																	
	20 Bit (8+12)						24 Bit (8+16)						32 Bit (8+24)					
	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)
4x4	54	764	2	2707	4	84,471	54	908	2	3773	6	80,351	54	1204	2	6658	10	66,438
5x5	82	1055	2	4243	7	75,186	82	1259	2	6166	9	75,961	82	1655	3	9079	14	71,046
6x6	116	1386	3	5253	8	81,153	116	1650	3	7501	11	74,841	116	2227	4	12139	18	68,914



Şekil 3.8 Matris Çarpma İşlemi Yarı Paralel Mimari Görünümü

3.5 Matris Tersi İşlemi

Matris işlem birimlerinden belki de en önemli ve karmaşık olan birim matris tersi alma işlemini yapan birimdir. Matris tersi alma işlemi başlı başına bir sorun olup konu ile ilgili pek çok çalışma mevcuttur. Yüksek oranda ölçeklenebilir ve hızlı bir şekilde üçgen matrislerin tersinin alınması için tasarlanan bir yapı Edman ve Owall tarafından önerilmiştir (Edman ve Owall 2003). Daha sonra yayınlanan bir diğer çalışmalarında Edman ve Owall qr ayrıştırmasına dayalı, iş hattı tekniği kullanan ölçeklenebilir bir mimari tasarlamışlardır (Echman ve Owall 2005). Başka bir çalışmada Laroche ve Roy çoklu giriş çıkışa sahip sistemlerde kullanılacak bir yapıyı VirtexII FPGA'sı üzerinde gerçeklemişlerdir (LaRoche ve Roy 2006). Benzer bir çalışmada çoklu giriş çıkışa sahip yazılım tabanlı radyo uygulamalarında kompleks matris tersi alma işlemi için tasarlanan yapı Eilert ve diğerleri tarafından Virtex4 FPGA'sı üzerinde gerçekleştirilmiştir (Eilert, Di Wu, ve Liu 2007). Büyük boyutlu matrislerin tersinin alınması için geliştirilen iş hattı yaklaşımına dayalı bir tasarım Zhou ve diğerleri tarafından fpga üzerinde tasarlanmış ve masaüstü işlemciler ile bir kıyaslaması yapılmıştır (Zhou ve diğerleri 2009). Arias-Garcia ve diğerlerinin yaptığı çalışmada ise kayan noktalı sayılardan oluşmuş matrislerin tersinin alınması için Gauss-Jordan Eliminasyonuna dayalı bir yöntem FPGA üzerinde gerçeklenmiştir (Arias-Garcia ve diğerleri 2011). Aynı ekibin bir diğer çalışmasında ise değişik bit uzunluklarındaki kayan noktalı sayılarla işlem yapan bir mimari oluşturulmuştur (Arias-Garcia ve diğerleri 2012).

Bu çalışmada ise matris tersi işlemi için Gauss-Jordan Eliminasyonuna dayalı bir mimari tasarlanmıştır. Tasarlanan mimari donanım kaynakları gözetilerek herhangi boyuttaki tekil olmayan bir matrise uygulanabilir şekilde oluşturulmuştur. Bu yapıya ait tanımlama bilgileri aşağıda verilmiştir.

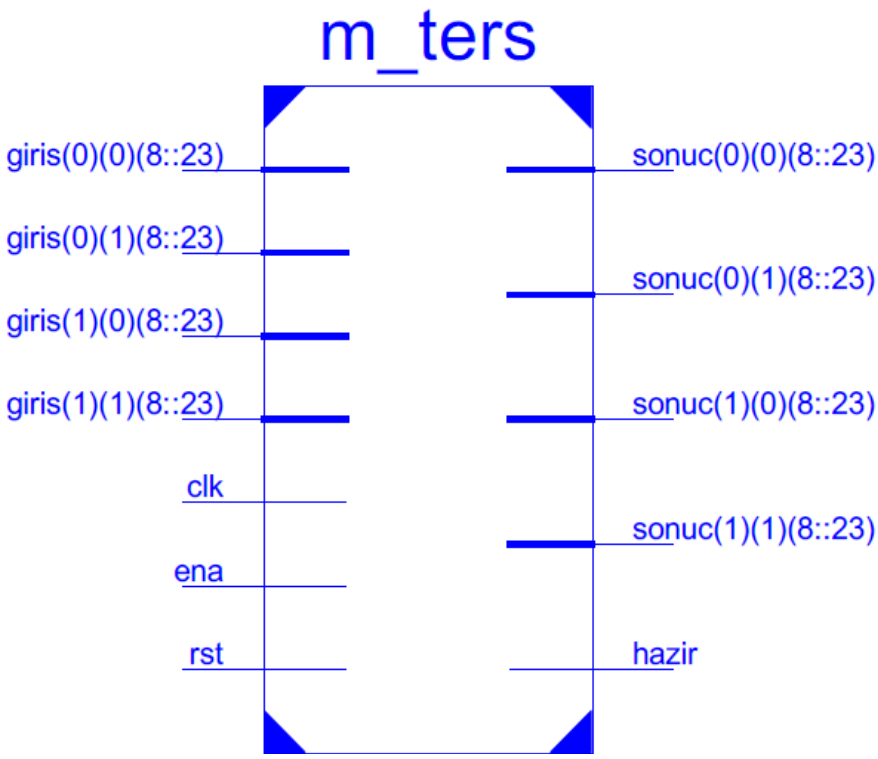
```
entity m_ters is
  generic (
    boyut : integer := 4);                              -- Matrisin boyutu.
  port (
    clk   : in  std_logic;                              -- Saat kaynagi.
    rst   : in  std_logic;                              -- Reset Ucu.
    ena   : in  std_logic;                              -- Etkinlestirme ucu.
    giris : in  matris(0 to boyut-1, 0 to boyut-1);     -- Matris Girisi
```

```
    sonuc : out matris(0 to boyut-1, 0 to boyut-1);  -- Islem sonucu cikisi.
    hazir : out std_logic
    );                                               -- Islem tamam cikisi.
end m_ters;
```

Bu tasarıma ait değişik bit uzunlukları ve değişik boyutlardaki matrislere ait kaynak tüketim bilgileri Tablo 3.9'da verilmiştir.

Tablo 3.9 Matris Tersi İşlemi Donanım Kaynak Kullanımı

Matris Boyutu	Matris Tersi																	
	20 Bit (8+12)						24 Bit (8+16)						32 Bit (8+24)					
	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)	Döngü	Reg.	%	LUT	%	Hız(MHz)
4x4	249	1198	2	3515	6	41,372	249	1426	3	4561	7	31,346	249	1882	3	6738	10	20,185
5x5	461	1741	3	4493	7	42,492	461	2077	4	5671	9	31,976	461	2751	4	8480	13	20,43
6x6	769	2401	4	5891	9	42,688	769	2869	5	7333	11	32,106	769	3807	6	###	16	20,497



Şekil 3.9 Matris Tersi İşlemi Mimari Görünümü

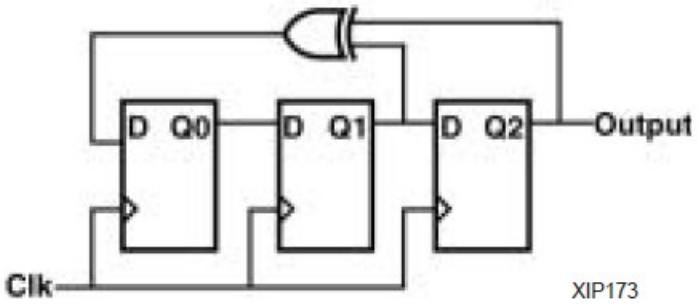
İç yapının oldukça karmaşık olması sebebiyle, iç yapının tamamının gösterimi yapılamamış olup sadece en üst seviyedeki genel gösterim yapılmıştır.

3.6 Rastgele Sayı Üreteci

Donanım üzerinde denemeler yapılırken ihtiyaç duyulacak gürültü bilgisini oluşturmak için rastgele sayı üretecine ihtiyaç duyulmuştur. Bu ihtiyacı karşılamak için donanım üzerinde çalışsan sözde rastlantısal sayı üreteci gerçekleştirilmiştir.

İlk olarak sayısal sistemlerde sıklıkla kullanılan LFSR (Linear Feedback Shift Register) yapısı kullanılarak sözde rastgele sayı üreteci tasarlanmıştır. Şekil 3.10'da 3 bitlik bir LFSR mimarisine ait gösterim verilmiştir (Xilinx 2003). Kullanılacak uygulamaya göre uygun uzunlukta seçilecek bir LFSR yapısının kendini tekrar etme döngüsü uzun olacağından üretilen sayılar rastgele gibi görünecektir.

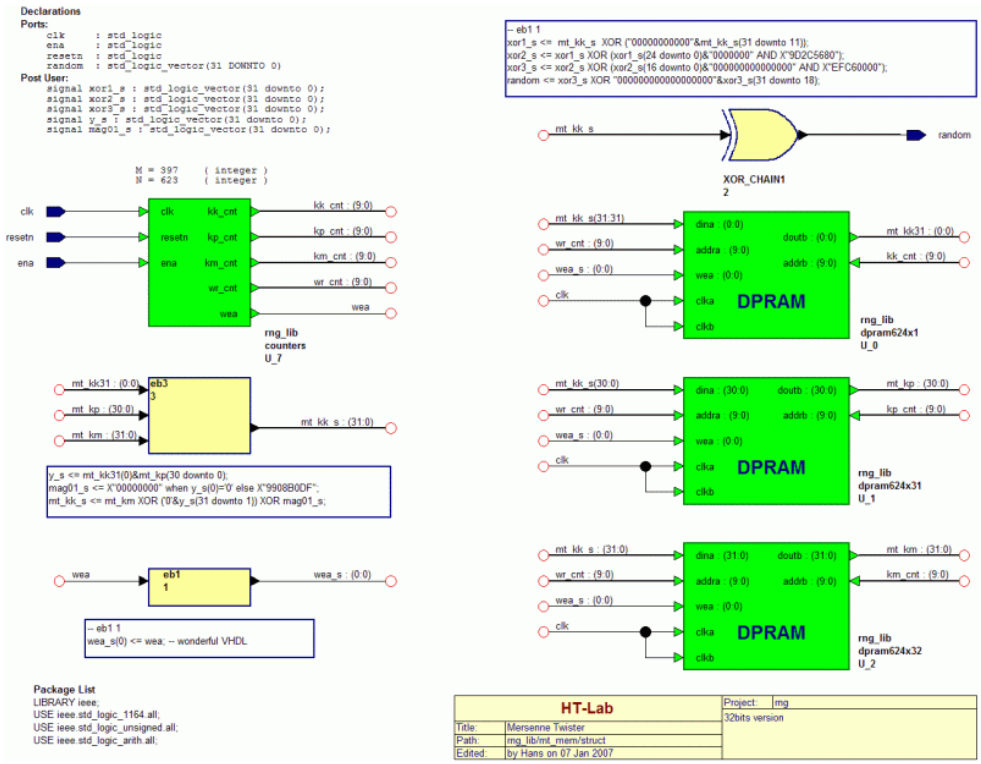
Bir LFSR tasarımındaki olası durumların sayısı **n** bit sayısı olmak üzere $2^n - 1$ ifadesi ile belirtilir. Kullanılacak uygulamaya göre seçilecek bit sayısı LFSR mimarisinin kendini tekrar etme süresini doğrudan etkileyecektir.



Şekil 3.10 LFSR Mimarisi (Xilinx 2003)

Bu çalışmada istenilen uzunlukta çalışabilecek bir LFSR tasarlanmış olup, LFSR başlangıç değeri kullanıcının isteğine göre belirlenebilecek bir parametre olarak belirlenmiştir. Başlangıç değeri sistemin her başladığında fiziksel dünyadan alınabilecek ve değişen bir büyüklükle ilişkilendirilirse, LFSR mimarisi her çalıştırıldığında farklı bir düzende çıkış üretecektir. Fakat sabit bir başlangıç değeri belirlenirse üretilen sayıların düzeni her seferinde aynı olacaktır.

Sayısal sistemlerde kullanılan bir diğer yöntem ise Mersenne Twister denilen yöntemdir. İlk olarak 1998 yılında Matsumoto ve Nishimura tarafından önerilen yöntem günümüzde pek çok alanda sıklıkla kullanılmaktadır (Matsumoto ve Nishimura 1998). Kendini tekrar döngüsü 2^{19937}-1 olan ve 32 bit uzunluk için 623 düzlemde eş dağılım gösterdiği ispatlanmış bir yöntemdir. Bu özellikleri sebebiyle özellikle benzetim ortamları için sıklıkla tercih edilmektedir. Bu yapıya ait Xilinx için tasarlanmış mimarinin görünümü Şekil 3.11'de verilmiştir.

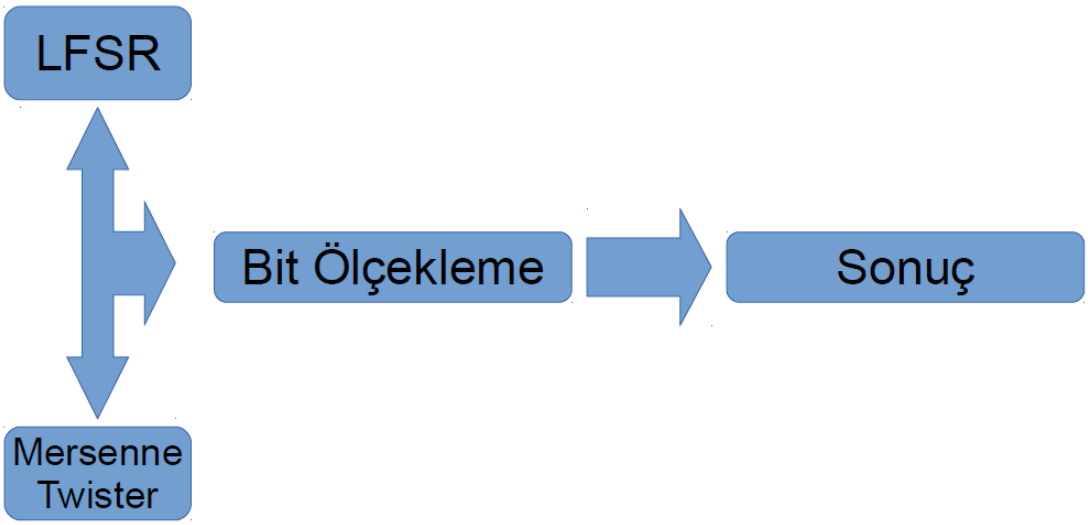


Şekil 3.11 Mersenne Twister Mimari Yapısı

(http://www.ht-lab.com/freecores/mt32/mersenne.html)

Tez çalışmasında kullanılan Mersenne Twister yapısı ise Heidelberg Üniversitesi Fizik bölümü tarafından ALICE (A Large Ion Collider Experiment) projesi kapsamında yapılan alt tasarımlarında kullanılan yapı esas alınmış olup tez çalışmasının ölçeklenebilirlik amacına uygun olarak bit uzunluğunun ve üretilen rastgele sayıların aralığının ayarlanabildiği bir yapı haline getirilmiştir. Bu sayede üretilen rastgele sayıların uygulamaya göre istenilen uzunlukta elde edilmesi sağlanmıştır.

Şekil 3.12'de girş değeri olarak LFSR veya Mersenne Twister tarafından üretilen sözde rastgele sayıları alıp, istenilen bit uzunluğunda ve istenilen aralıkta ölçekleyen işlem modülüne ait blok gösterim verilmiştir.



Şekil 3.12 Sözde Rastgele Sayı Üretimi Blok Gösterimi

Şekil 3.12'de verilen blok gösterime ait VHDL kodu parçası aşağıda verilmiştir. Verilen kod parçasında genelleştirilmiş parametrelerden **width** toplam bit uzunluğunu ifade etmektedir. **Gnrt** parametresi ile sayı kaynağı seçimi yapılmaktadır. Bu ayar ile sayı kaynağının LFSR ya da Mersenne Twister olup olmayacağı seçilmektedir. Geri ye kalan iki parametre olan **e** ve **f** parametresi ise kayan noktalı sayı için yapılacak ölçekleme ayarını belirtmektedir. **"e"** parametresi kayan noktalı sayının üst kısmını belirlerken **f** paramatresi ise ondalıklı kısmı belirtmektedir.

```
entity randomizer is
        generic(
                width   :       integer := 32;
                gnrtr   :       rnd_gen_type;
                e       :       integer := 3;
                f       :       integer := -28
                 );
        port(
                clk     :       in      std_logic;
                rst     :       in      std_logic;
                rnd     :       out     std_logic_vector(width-1 downto 0);
                rdy     :       out     std_logic
                 );
end randomizer;
```

BÖLÜM IV

TAKİP ALGORİTMALARI

4.1 Kalman Süzgeci

R.E Kalman tarafından 1960 yılında önerilmesinden bu yana sıklıkla kullanılan bir yöntemdir (Kalman 1960). Basitçe daha önceki ölçüm değerlerine dayanarak bir sonraki adımı tahmin etmeye dayalı bir yöntemdir. İlk yayınlandığı dönemde sadece doğrusal sistemlere uygulanmış olup daha sonra lineer olmayan sistemlere uygulanabilen hali olan Genişletilmiş Kalman Süzgeci geliştirilmiştir. Kalman filtresi temel olarak Eşitlik 4.1 ve 4.2'de gösterildiği şekilde ifade edilmiştir.

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{4.1}$$

$$z_k = Hx_k + v_k \tag{4.2}$$

Bu ifadeye göre herhangi bir x_k değerini hesaplamak için bir önceki adımdaki değerinden, buna eklenmiş olan kontrol sinyalinden u_k ve son olarak da bir önceki adımdaki işlem gürültüsünden w_{k-1} faydalanabiliriz. Ölçüm bilgisinin elde edilmesinde ise yine x_k değerinden ve ona eklenmiş olan ölçüm gürültüsünden v_k faydalanabiliriz. Denklemlerde bahsedilen gürültü işaretinin Gaussian olduğu unutulmamalıdır.

Denklem setini düzenlersek iki başlık altında toplanabilir. Tahmin aşaması ve güncelleme aşaması olarak adlandırabileceğimiz bu başlıklara ait genel gösterim Eşitlik 4.2 ve 4.3'de verilmiştir.

$$\hat{x}(k|k-1) = F\,\hat{x}(k-1|k-1) + Bu(k) \tag{4.2}$$

$$P(k|k-1) = FP(k-1|k-1)F^T + GQG^T \tag{4.3}$$

Eşitlik 4.2 ve 4.3'te verilen ifadeler tahmin aşamasının denklemleridir. Eşitlik 4.4 ve 4.8 arasındaki ifadelerde ise güncelleme aşamasına ait denklemler verilmiştir.

$$\hat{x}(k|k-1)=\hat{x}(k|k-1)+W(k)v(k) \tag{4.4}$$

$$P(k|k)=P(k|k-1)-W(k)SW(k)^T \tag{4.5}$$

$$v(k)=z(k)-H\hat{x}(k|k-1) \tag{4.6}$$

$$W(k)=P(k|k-1)H^T S^{-1} \tag{4.7}$$

$$S=HP(k|k-1)H^T+R \tag{4.8}$$

4.2 Genişletilmiş Kalman Süzgeci

Genişletilmiş Kalman Süzgeci neredeyse Kalman Süzgeci ile aynıdır. Gerçek hayatta çoğunlukla doğrusal olmayan sistemler olduğu için Kalman Süzgeci'nin doğrusal olmayan sistemlere uyarlanmış halidir. Bu uyarlama yapılırken doğrusal olmayan modeller en iyi tahminin olduğu noktada doğrusallaştırılır. Genişletilmiş Kalman Süzgeci denklemleri sırasıyla Eşitlik 4.9- 4.17'de verilmiştir.

Tahmin Denklemleri :

$$\hat{x}(k|k-1)=f(\hat{x}(k-1|k-1),u(k),k) \tag{4.9}$$

$$P(k|k-1)=\nabla F_x P(k-1|k-1)\nabla F_x^T+\nabla F_v Q \nabla F_v^T \tag{4.10}$$

$$z(k|k-1)=H(\hat{x}(k|k-1)) \tag{4.11}$$

Güncelleme Denklemleri :

$$\hat{x}(k|k)=\hat{x}(k|k-1)+W(k)v(k) \tag{4.12}$$

$$P(k|k)=P(k|k-1)-W(k)S(k)W(k)^T \tag{4.13}$$

$$v(k)=z(k)-z(k|k-1) \tag{4.14}$$

$$W=P(k|k-1)\nabla H_x^T S^{-1} \tag{4.15}$$

$$S=\nabla H_x P(k|k-1)\nabla H_x^T+R \tag{4.16}$$

$$\nabla F_x=\frac{(\partial f)}{(\partial x)}=\begin{bmatrix} \frac{(\partial f_1)}{(\partial x_1)} & \cdots & \frac{(\partial f_1)}{(\partial x_m)} \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \frac{(\partial f_n)}{(\partial x_1)} & \cdots & \frac{(\partial f_n)}{(\partial x_m)} \end{bmatrix}$$

$$\nabla F_u=\frac{(\partial f)}{(\partial u)}=\begin{bmatrix} \frac{(\partial f_1)}{(\partial u_1)} & \cdots & \frac{(\partial f_1)}{(\partial u_m)} \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \frac{(\partial f_n)}{(\partial u_1)} & \cdots & \frac{(\partial f_n)}{(\partial u_m)} \end{bmatrix} \tag{4.17}$$

$$\nabla H_x=\frac{(\partial h)}{(\partial x)}=\begin{bmatrix} \frac{(\partial h_1)}{(\partial x_1)} & \cdots & \frac{(\partial h_1)}{(\partial x_m)} \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \frac{(\partial h_n)}{(\partial x_1)} & \cdots & \frac{(\partial h_n)}{(\partial x_m)} \end{bmatrix}$$

Denklem setlerinden de görüleceği üzere Genişletilmiş Kalman Süzgeci'nin Kalman Süzgecin'den farkı sistem ölçüm ve model denklemlerinde Jacobian matrislerinin olmasıdır. Bu sayede yapılan en iyi tahmin noktasında sistem modeli doğrusallaştırılarak işlemlere devam edilir.

BÖLÜM V

DONANIM UYGULAMALARI

5.1 Kalman Süzgeci Örnek Uygulama

Bu çalışmada gerçekleştirilen mimarinin uygulanması için Kalman Süzgeci ile takip yapılan bir eğik atış problemi ele alınmıştır. Problem olarak basit olmasına rağmen mimarinin uyarlanabilirliği adına anlaması kolay bir örnek olduğu için bu uygulamada karar kılınmıştır. Sistemin sabit ivmeye sahip olduğu kabul edilmiş olup model denklemleri sırasıyla Eşitlik 4.18'de verilmiştir.

$$x_{t+1}=\begin{bmatrix} 1 & \Delta t & \frac{(\Delta t^2)}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \cdot x_t = \begin{bmatrix} x_t+\Delta t \cdot v_t+\frac{(\Delta t^2 a)}{2} \\ v_t+\Delta t \cdot a \\ a \end{bmatrix} \tag{5.1}$$

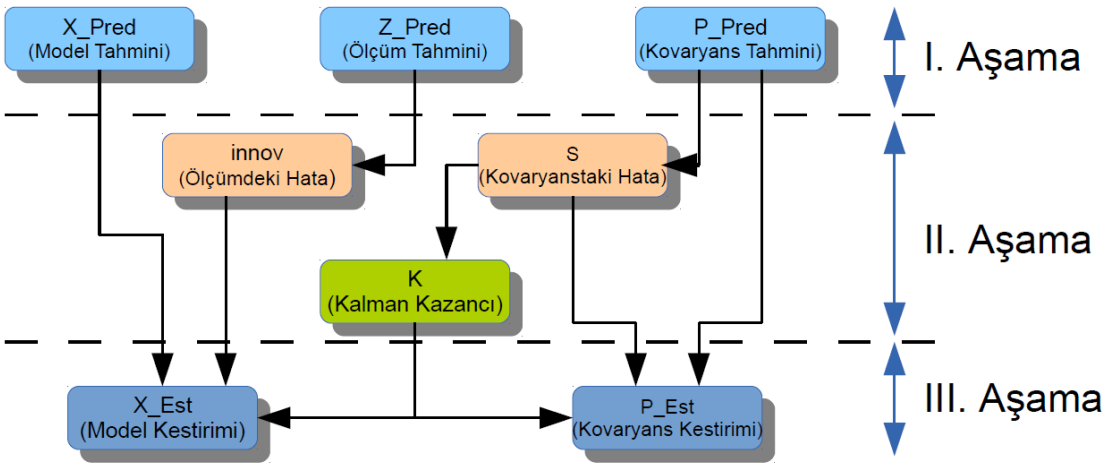
Eşitlik 4.18'de x yönündeki harekete ait denklemler, Eşitlik 4.19'da ise y yönündeki bileşenine ait denklemler verilmiştir.

$$y_{t+1}=\begin{bmatrix} 1 & \Delta t & \frac{(\Delta t^2)}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \cdot x_t = \begin{bmatrix} y_t+\Delta t \cdot v_t+\frac{(\Delta t^2 a)}{2} \\ v_t+\Delta t \cdot a \\ a \end{bmatrix} \tag{5.2}$$

Her iki yöndeki harekete ait denklemleri birleştirip sisteme ait modeli oluşturursak Eşitlik 4.20'de verilen denklem setini elde etmiş oluruz.

$$A=\begin{bmatrix} 1 & \Delta t & \frac{(\Delta t^2)}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{(\Delta t^2)}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x_t+\Delta t \cdot v_t+\frac{(\Delta t^2 \cdot a_x)}{2} \\ v_t+\Delta t \cdot a_x \\ a_x \\ y_t+\Delta t \cdot v_t+\frac{(\Delta t^2 \cdot a)}{2} \\ v_t+\Delta t \cdot a_y \\ a_y \end{bmatrix} \quad (5.3)$$

İşlem kolaylığı açısından konum bilgilerinin doğrudan ölçülebildiği kabul edilmiştir. Bu kabullerden sonra daha önceki bölümlerde anlatılan matematiksel işlem bileşenlerini kullanarak gerekli işlemler gerçekleştirilmiştir. Oluşturulan yapıya ait mimarinin gösterimi Şekil 5.1'de verilmiştir.



Şekil 5.1 Kalman Süzgeci FPGA Mimarisi

Bileşenler arasındaki ilişki gözetilerek mimari tasarımında mümkün mertebe paralellik sağlanmaya çalışılmıştır. İşlem birimleri arasında birbirine bağlı olmayan kısımlar eş zamanlı çalışacak şekilde tasarlanmıştır. İşlem basamaklarını gösteren denklem seti sırasıyla Eşitlik 5.4-5.11'de verilmiştir.

$$X_{pred} = A \cdot X_{est} \tag{5.4}$$

$$P_{pred} = APA^T + Q \tag{5.5}$$

$$Z_{pred} = H \cdot X_{est} \tag{5.6}$$

$$innov = Z - Z_{pred} \tag{5.7}$$

$$S = HP_{pred} H^T + R \tag{5.8}$$

$$K = P_{pred} H^T S^{-1} \tag{5.9}$$

$$X_{est} = X_{pred} + K \cdot innov \tag{5.10}$$

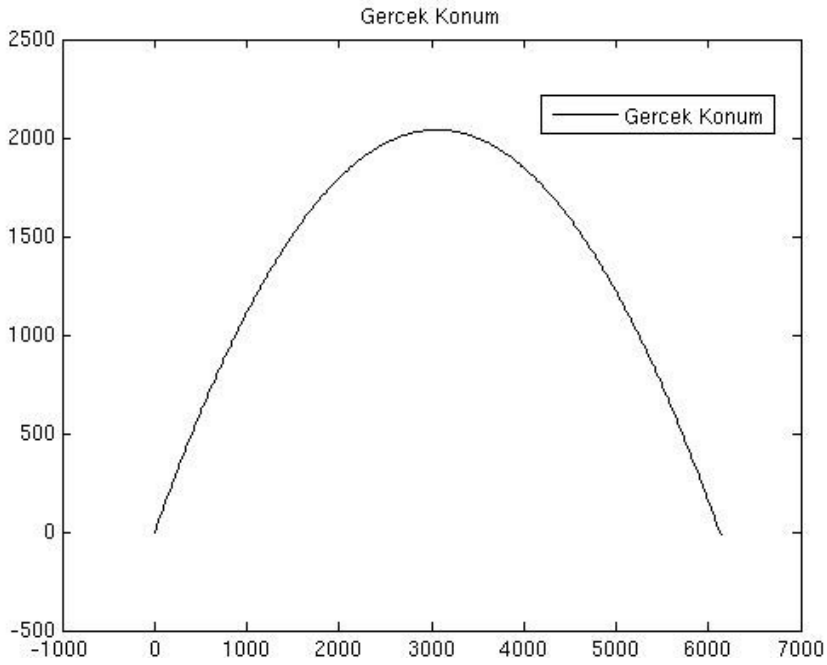
$$P_{est} = P_{pred} - KSK^T \tag{5.11}$$

Verilen formüllere uygun olarak tasarlanan ve paralel çalışma için birimler arasındaki ilişkilere dikkat edilen yapı FPGA üzerinde IEEE754 standartlarına uygun olarak sentezlenmiş ve çalıştırılmıştır. Yapının esnek tasarımı sayesinde farklı bit uzunluklarında sentezlenmiş ve elde edilen kaynak kullanım sonuçları Tablo 5.1'de verilmiştir.

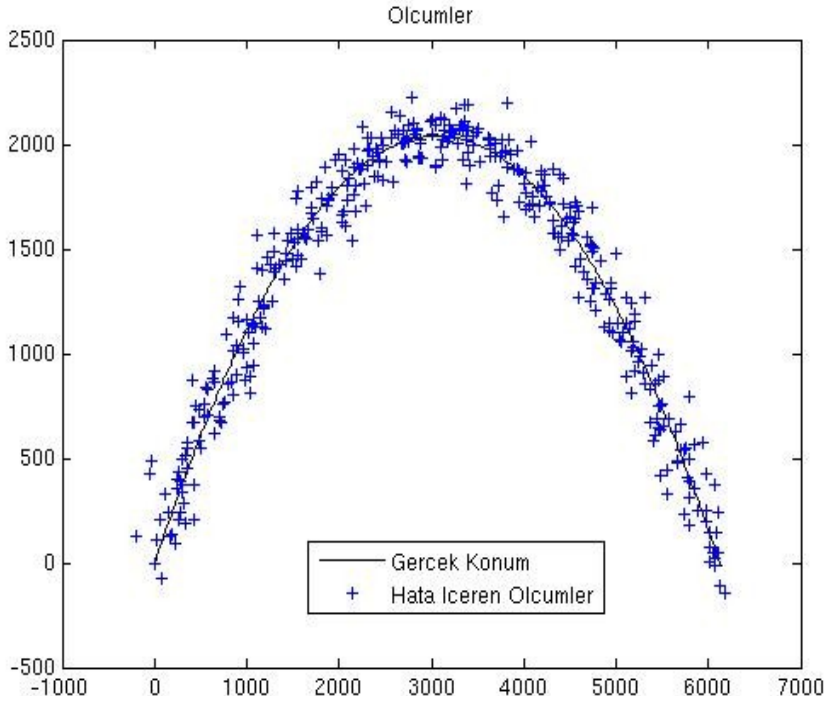
Tablo 5.1 Kalman Süzgeci Donanım Kaynak Tüketimi

BİT UZUNLUĞU	Döngü	Register	%	LUT	%	DSP48
20 Bit	2268	7614	12	16269	24	12
24 Bit	2268	9070	14	19098	28	12
32 Bit	2268	11979	18	35057	51	12

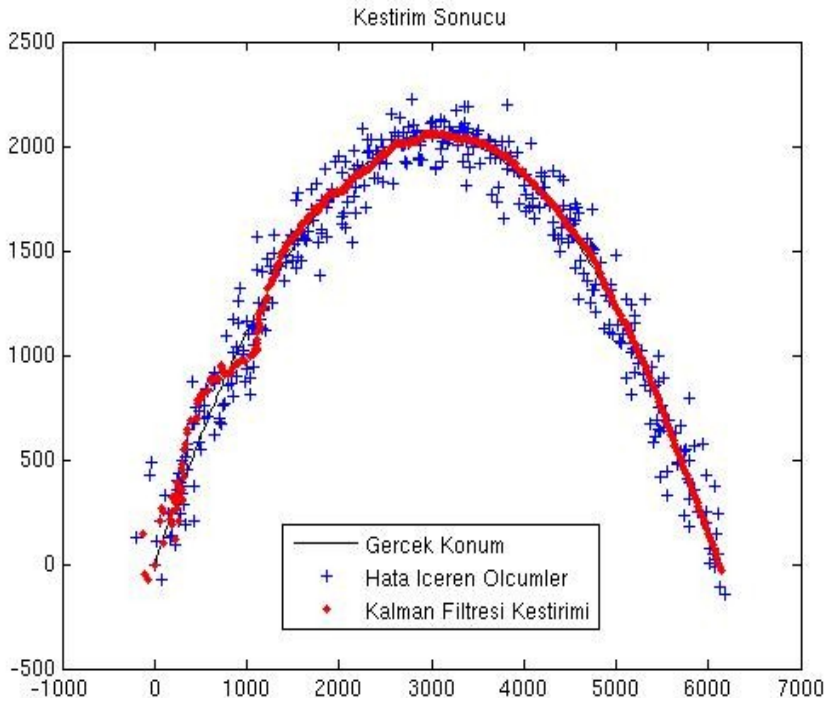
Tasarlanan yapının sonuçlarını kıyaslamak için aynı yapı MATLAB üzerinde uyarlanıp çalıştırılmıştır. Sırasıyla Şekil 5.2, Şekil 5.3 ve Şekil 5.4'te MATLAB sonuçlarına ait çıktılar verilmiştir.



Şekil 5.2 Matlab Gerçek Konum Sonuçları



Şekil 5.3 Matlab Ölçüm Sonuçları



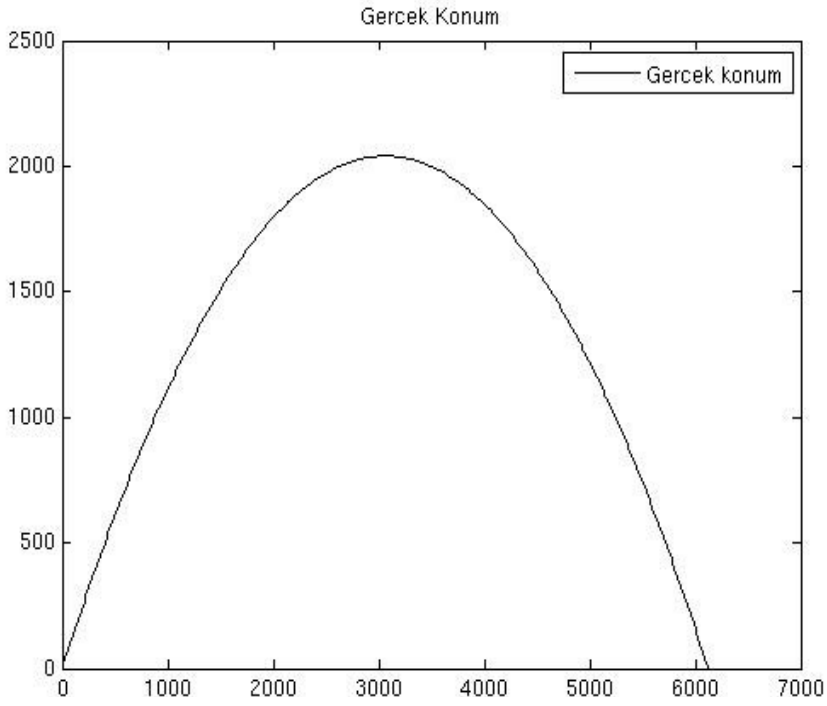
Şekil 5.4 Matlab Kestirim Sonuçları

Aşağıda verilen VHDL kod parçasında ise mimarinin çalışma sıralamasına ait bir kısım gösterilmiştir.

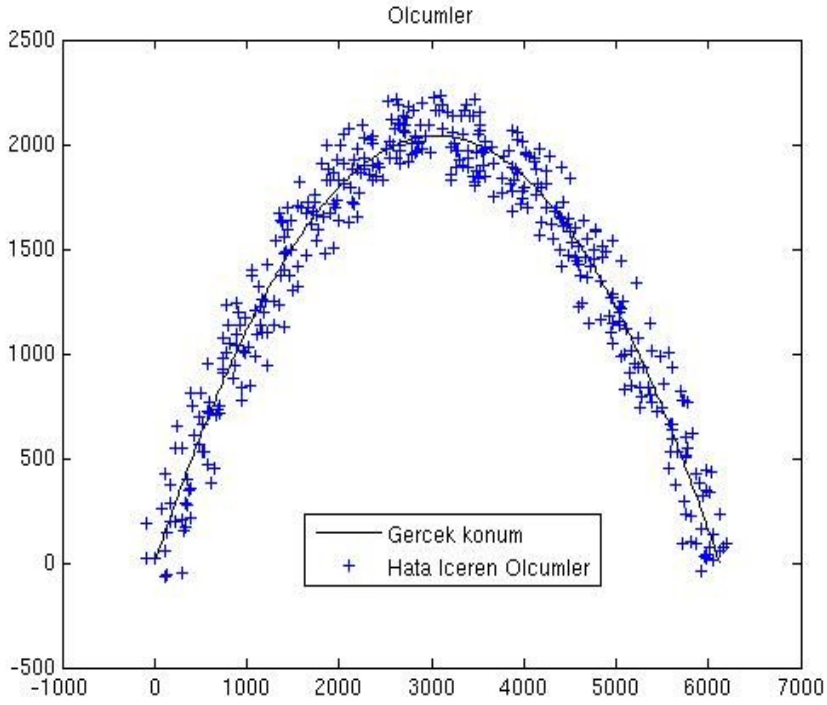
```
case state is
  when 0 => hazir <= '0';
            if(ena = '1') then
              p_rst   <= '0';
              prd_ena <= '1';
              obs_ena <= '1';
              upd_ena <= '1';
              state   := state + 1;
            end if;

  when 1 => if(upd_rdy_3 = '1') then
              est_x_val <= tmp_sig_8;
              p_matrix  <= tmp_sig_9;
              p_rst     <= '1';
              prd_ena   <= '0';
              obs_ena   <= '0';
              upd_ena   <= '0';
              hazir     <= '1';
              state     := 0;
            end if;
end case;
```

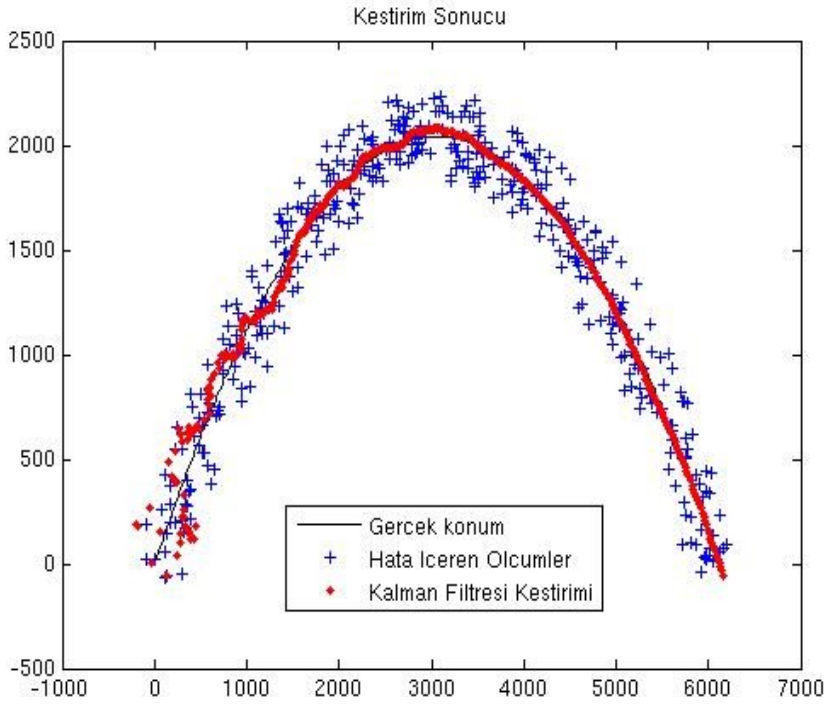
Aynı mimari FPGA üzerinde gerçekleştirildiğinde elde edilen sonuçlar sırasıyla Şekil 5.5, Şekil 5.6 ve Şekil 5.7'de verilmiştir.



Şekil 5.5 FPGA Gerçek Konum Sonucu



Şekil 5.6 FPGA Ölçüm Sonuçları



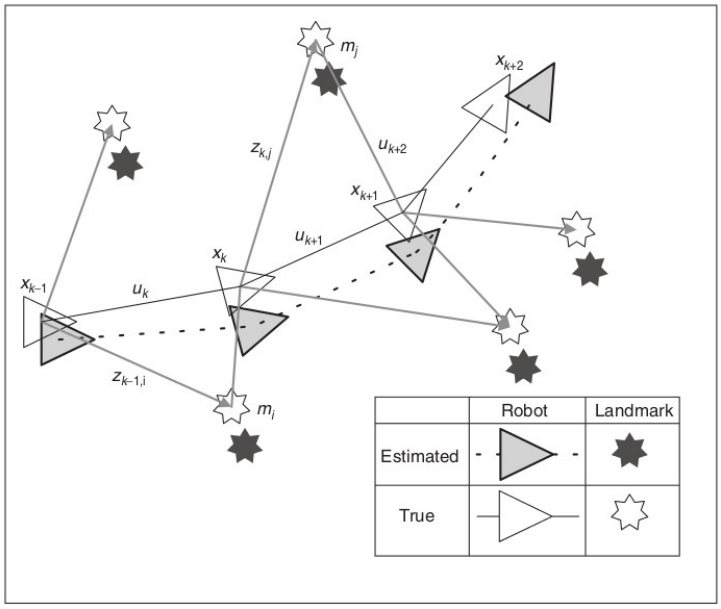
Şekil 5.7 FPGA Kestirim Sonuçları

FPGA üzerinde elde edilen sonuçlar RS232 üzerinden bilgisayara aktarılmış ve MATLAB üzerinde işlenerek grafikler çizdirilmiştir.

5.2 Genişletilmiş Kalman Süzgeci Örnek Uygulama

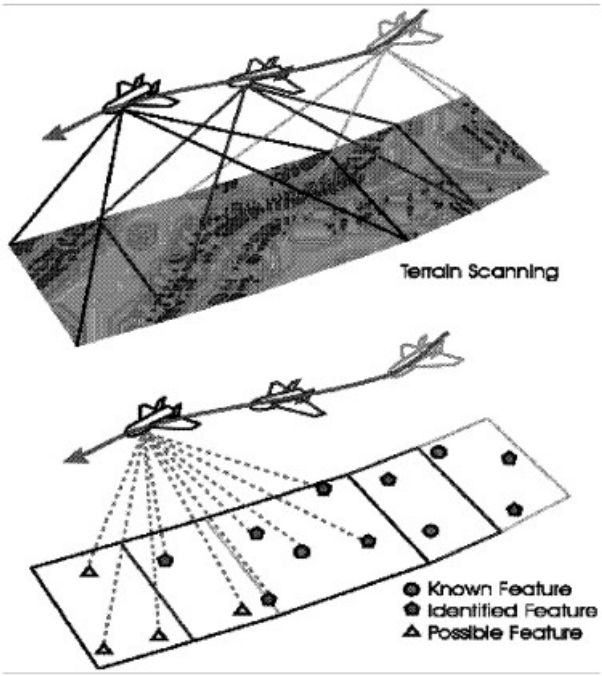
Bir önceki bölümde Kalman Süzgeci FPGA üzerine uyarlandıktan sonra, işlem altyapısı Genişletilmiş Kalman Süzgeci için değiştirilerek benzer ama farklı bir mimari tasarlanmıştır. Bu tasarım için seçilen örnek uygulama ise SLAM tekniğinde, kestirim amacıyla kullanılan Genişletilmiş Kalman Süzgeci'dir. Çalışma kapsamında seçilen bu uygulamadaki asıl amaç aynı altyapıyı kullanarak alt seviyede tasarlanmış işlem birimlerinin kolayca probleme göre tekrar organize edebilmektir.

SLAM problemi otonom araçlar için son dönemde sıklıkla çalışılan bir konudur. Bilinmeyen bir ortamda hareket eden otonom aracın hem kendi konumunu bilmesi hem de bulunduğu alanın haritasını çıkarması ve bu haritaya göre kendini konumlandırması gerekmektedir. Literatürde aynı zamanda tavuk - yumurta problemi olarak da adlandırılmaktadır. Konum belirlemek için kullanılan noktaların gerçek konumları bilinmemekte veya doğrudan doğruya ölçülememektedir. Kesin doğruluğu bilinmeyen bu noktalara göre otonom aracın konumunu kestirmesi beklenmektedir. Şekil 5.8'de gösterilen çizim problemi göstermektedir (Durrant-Whyte ve Bailey 2006).



Şekil 5.8 SLAM Problemi (Durrant-Whyte ve Bailey 2006)

Bu yöntemin yakın dönemdeki uygulamalarından biri de insansız hava araçların konumlandırmasıdır. Şekil 5.11'de insansız hava araçları ve SLAM arasındaki ilişkiyi gösteren bir çizim verilmiştir (Kim ve Sukkarieh 2004).



Şekil 5.9 (Kim ve Sukkarieh 2004)

SLAM problemini FPGA üzerine uygulamadan önce ilk olarak bilgisayar ortamında benzetimini yapmak için MATLAB üzerinde çalışılarak bir benzetim ortamı hazırlanmıştır. Bu ortamda kullanılacak yer işaretlerinin belirlenmesi için gerekli görüntü Google Earth programı yardımıyla Niğde Üniversitesi Merkez Yerleşkesinden 1600x900 boyutunda elde dilmiştir. İlgili görüntü Resim 5.1'de verilmiştir.

Resim 5.1 Niğde Üniversitesi Merkez Yerleşke Uydu Görüntüsü

Alınan görüntüden uygun yer işaretçilerinin çıkarılması için görüntü işleme yapılarak uygun noktaların tespit edilmesi gerekmektedir. Bu amaçla kullanılabilecek köşe tespiti, kenar tespiti, damla tespiti, SIFT yöntemi, SURF yöntemi, SUSAN yöntemi gibi çoğaltılabilecek pek çok yöntem mevcuttur(Canny 1986; Harris ve Stephens 1988; S. Smith 1992; Lindeberg 1996; Carson ve diğerleri 2002; Lowe 2004; Bay ve diğerleri 2008).

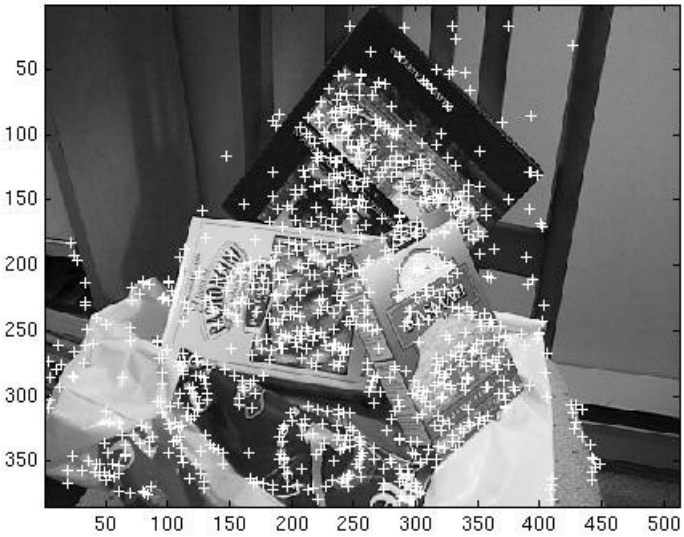
Literatür incelendiğinde SLAM uygulamalarında SIFT yönteminin sıklıkla tercih edildiği görülmektedir (Davison ve diğerleri 2007; Newman, Cole, ve Ho 2006; Elinas, Sim, ve Little 2006; Se, Lowe, ve Little 2005). Bu çalışmada da elde edilen görüntüden uygun yer noktalarının bulunması için SIFT yöntemi kullanılmıştır. Bu yöntemin en önemli faydası bulunan yer işaretçilerini tanıttıktan sonra, ölçekten ve yönden bağımsız olarak bu işaretçileri görüntüde ayırt edebilmesidir. Bu sayede SLAM uygulaması için önemli bir nokta olan bulunan işaretçilerin daha önceden bilinen bir işaretçi olup olmadığı bilgisi elde edilebilmektedir.

SIFT yöntemi şu adımlardan oluşmaktadır:

1. Ölçek uzayını oluştur.
2. Anahtar noktaları belirle.

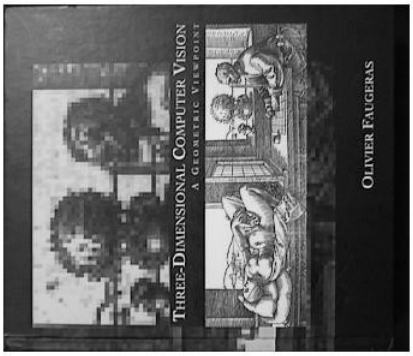
3. Kenar ve az kontrast içeren yerler gibi kötü noktaları ele.
4. Anahtar noktaların yönelimlerini bul.
5. SIFT özniteliklerini oluştur.

Elde edilen çok sayıdaki eşsiz, ölçekten bağımsız öznitelikler sayesinde görüntü takibi, nesne bulma, nesne tanıma gibi görüntü işleme açısından önemli işlemler gerçekleştirilebilmektedir. Resim 5.2'de örnek bir resim üzerinde SIFT uygulanarak bulunan anahtar noktalar gösterilmiştir.

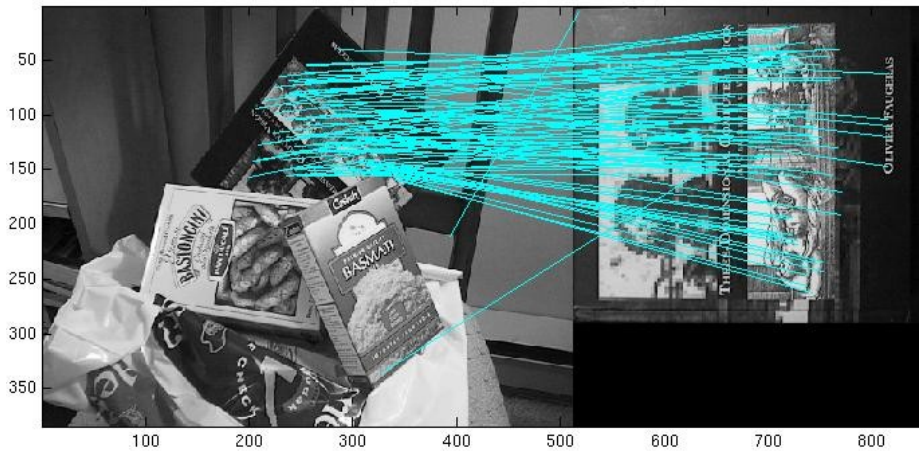


Resim 5.2 SIFT ile bulunan anahtar noktalar.

Anahtar noktalar bulunduktan sonra görüntü üzerinde Resim 5.3'te verilen örnek bir nesne ile eşleştirme yapılarak elde edilen sonuçlar Resim 5.4'te verilmiştir.



Resim 5.3 Eşlenecek Nesne



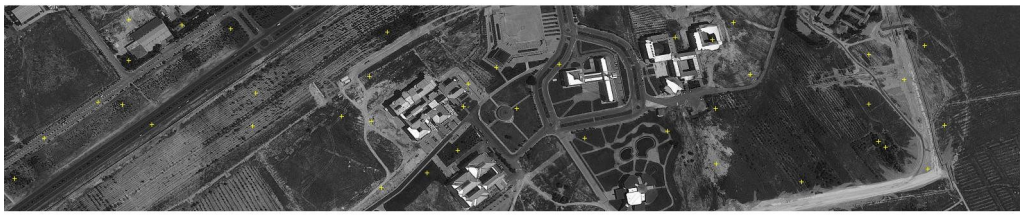
Resim 5.4 Eşlenmiş resim.

SIFT yöntemi bu çalışmada görüntü üzerindeki anahtar noktaları belirlemek amacıyla kullanılmıştır. Bulunan yer işaretçilerinin aynı olup olmadığının belirlenmesi bu çalışma kapsamında hedeflenmemiştir.

Yöntemin herhangi bir eşik değer kullanılmadan uygulanması halinde örnek görüntümüzde yaklaşık 10000 aday nokta belirlenmektedir. Resim 5.5'de belirlenen aday noktaları gösterilmiştir.

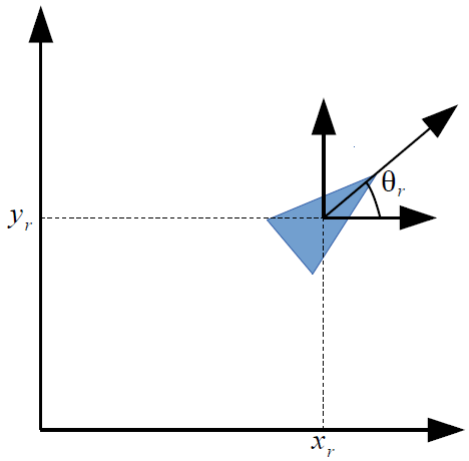
Resim 5.5 Kampüs Bölgesi Aday Noktaları

Çalışma alanını küçültmek için örnek görüntüden 1600x320 boyutunda bir kesit alınarak işlemler bu bölgede gerçekleştirilmiştir. SIFT ile bulunan aday noktaları SIFT öznitelikleri incelenerek ölçek uzayı değerleri en büyükten en küçüğe olacak şekilde sıralanmış ve en yüksek değere sahip noktalar seçilmiştir. Resim 5.6'da kesit alınmış görüntü üzerinde seçilen yer işaretçileri gösterilmiştir.



Resim 5.6 Kesit Görüntü ve Aday Noktalar

Belirlenen yer işaretleri SLAM uygulaması için Genişletilmiş Kalman Süzgecine giriş olarak kullanılan bilgilerden bir kısmını oluşturmaktadır. Genişletilmiş Kalman Süzgecine giriş olarak kullanılacak bir diğer bilgi ise SLAM uygulamasında kullanıldığı kabul edilen İnsansız Hava Aracı modelidir. Uygulama kolaylığı açısından insansız hava aracı modelinin serbestlik derecesinin 3 olması tercih edilmiş ve sabit yükseklikte uçtuğu kabul edilmiştir. Şekil 5.10'da insansız hava aracının gösterimi verilmiştir. Buna göre insansız hava aracının işlem modeli Eşitlik 5.12, 5.13, 5.14 ve 5.15'te gösterilmiştir.



Şekil 5.10 İnsansız hava aracının gösterimi.

x_r ve y_r kuş bakışı olarak insansız hava aracının konumunu belirmekte olup θ_r ise insansız hava aracının yönelim açısını göstermektedir.

$$x_{iha}(k)=[x_r(k);y_r(k);\theta_r(k)] \tag{5.12}$$

$$x_r(k)=x_r(k-1)+\Delta t*v(k-1)*\cos\theta_r(k-1) \tag{5.13}$$

$$y_r(k)=y_r(k-1)+\Delta t*v(k-1)*\sin\theta_r(k-1) \tag{5.14}$$

$$\theta_r(k)=\theta_r(k-1)+\Delta t*v(k-1)*\tan(\theta_r(k-1)) \tag{5.15}$$

Elde edilen denklem setine yer işaretçileri de dahil edilerek Eşitlik 5.16 ve 5.17'de bulunan denklemler oluşturulmuştur.

$$x_{hrt}(k)=[x_1(k);y_1(k);x_2(k);y_2(k);x_3(k);y_3(k);.....x_n(k);y_n(k)] \tag{5.16}$$

$$x(k)=\begin{bmatrix}x_{iha}\\x_{hrt}\end{bmatrix}=\begin{bmatrix}x_r(k)\\y_r(k)\\\theta_r(k)\\x_1(k)\\y_1(k)\\x_2(k)\\y_2(k)\\.\\.\\.\\x_n(k)\\y_n(k)\end{bmatrix}+W(k) \tag{5.17}$$

Bir önceki denklem seti ile sistemimiz için gerekli olan durum denklemleri elde edilmiştir. Modelin tamamlanması için gereken bir diğer denklem seti ise ölçüm denklemleridir. Eşitlik 5.18 ve 5.19'da ise ilgili denklem setleri verilmiştir. **v(k)** ölçüm gürültüsü olmak üzere;

$$z(k)=h(x(k))+v(k) \tag{5.18}$$

$$h(k)=\begin{bmatrix} h_1(x_{1,}y_1) \\ h_2(x_{2,}y_2) \\ . \\ . \\ . \\ h_n(x_n,y_n) \end{bmatrix} \quad ve \quad h_i(x_i,y_i)=\begin{bmatrix} h_{i1} \\ h_{i2} \end{bmatrix}=\begin{bmatrix} \sqrt{(\Delta x_i^2+\Delta y_i^2)} \\ atan2(\Delta y_i,\Delta x_i)-\theta_r(k) \end{bmatrix} \tag{5.19}$$

Δx ve Δy hareket halinde ki aracımızla yer işaretçisi arasındaki mesafeyi belirtmektedir. Bölüm IV'te anlatılan Genişletilmiş Kalman Süzgecine göre sistem modeli uygulanır ve Jacobian matrisleri hesaplanırsa Eşitlik 5.20'de verilen denklemler elde edilmektedir.

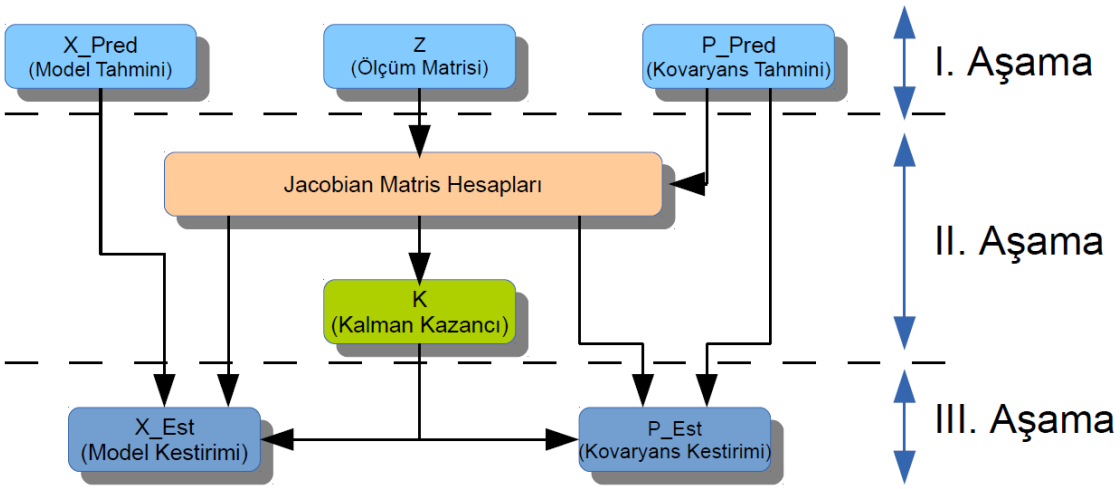
$$F(k)=\nabla F=\begin{bmatrix} 1 & 0 & -\sin\theta_r(k)*\Delta t*v & 0 \\ 0 & 1 & \cos\theta_r(k)*\Delta t*v & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & I_{nxn} \end{bmatrix} \tag{5.20}$$

Aynı işlemlerin ölçüm matrisleri içinde gerçekleştirilmesi durumunda Eşitlik 5.21'de bulunan matematiksel ifade elde edilmektedir. Bu matrislerin boyutları bulunan yer işaretçi sayısına göre değişmektedir.

$$H_i(k)=\nabla h_i=\frac{(\partial h_i(x_i,y_i))}{(\partial x(k))}=\begin{bmatrix} \frac{-(\Delta x_i)}{r_i} & \frac{-(\Delta y_i)}{r_i} & 0 & . & . & . & 0 & \frac{(\Delta x_i)}{r_i} & \frac{(\Delta y_i)}{r_i} & 0 & . & . & 0 \\ \frac{(\Delta y_i)}{r_i^2} & \frac{-(\Delta x_i)}{r_i^2} & -1 & . & . & . & 0 & \frac{-(\Delta y_i)}{r_i^2} & \frac{(\Delta x_i)}{r_i^2} & 0 & . & . & 0 \end{bmatrix} \tag{5.21}$$

$$r_i=\sqrt{(\Delta x_i^2+\Delta y_i^2)}$$

Elde edilen denklemler Bölüm IV'te anlatılan Genişletilmiş Kalman Süzgeci denklemlerinde yerine koyularak sistem modeli tamamlanmış olur. Tamamlanan sistem modeli Matlab ve FPGA üzerinde uyarlanarak ölçümler yapılmıştır. FPGA üzerinde tasarlanan sistemin blok gösterimi Şekil 5.11'de verilmiştir.



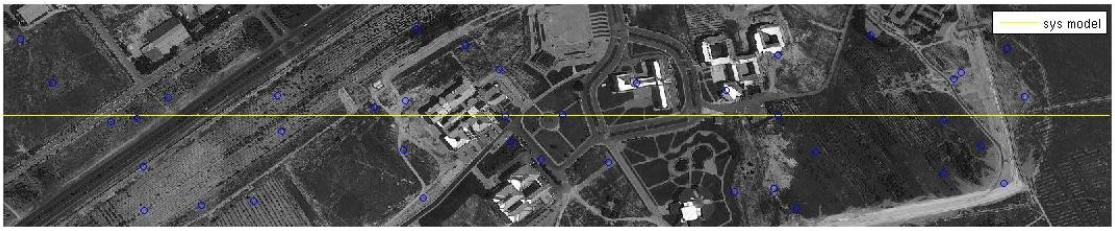
Şekil 5.11 FPGA Mimarisi

Kalman Süzgeci FPGA mimarisinden farklı olarak bu tasarımda Jacobian Matrislerinin hesaplandığı bir birim eklenmiştir. Bu birim seçilen yer işaretçi sayısına bağlı olarak döngüsel bir işlem yapmaktadır. Döngünün her bir adımında tek bir yer işaretçisi için hesap yapılıp matrisler güncellenmektedir. Bu sebeple yer işaretçi sayısının doğrudan olarak işlem süresi üzerinde etkisi mevcuttur.

Tamamlanan sistem modeli için ilk olarak MATLAB ortamında benzetimler yapılarak 3 farklı rota türü için modelin davranışı incelenmiştir. Bu rotalar aşağıda sıralanmıştır:

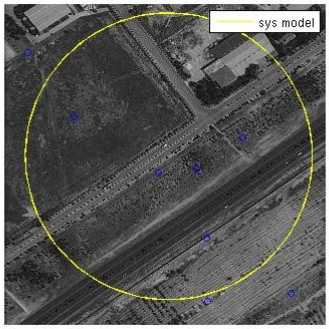
1. Doğrusal Rota
2. Dairesel Rota
3. Kesişen Rota (Sekiz Rota)

Doğrusal rota sistemin iki nokta arasında düz bir hattı takip etmesi gerektiğinde oluşan davranışı incelemek için seçilmiştir. Şekil 5.12'de ideal durumdaki doğrusal rota bilgisi verilmiştir.



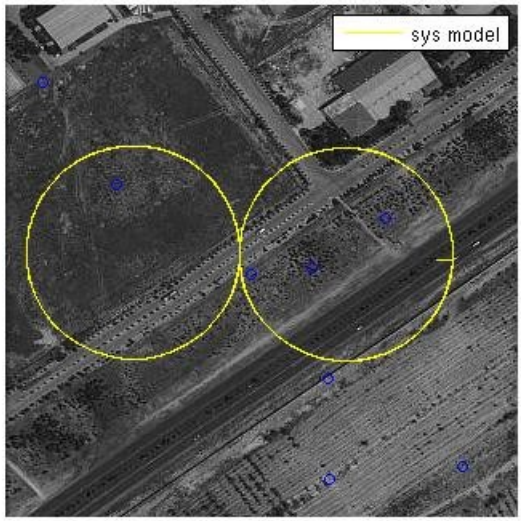
Şekil 5.12 Doğrusal Rota

Dairesel rota ise istenen bir nokta etrafında belirlenerek sistemin bu nokta etrafında turlarken oluşan davranışının incelenmesi için seçilmiştir. Şekil 5.13'te bu davranışa ait gösterim verilmiştir.



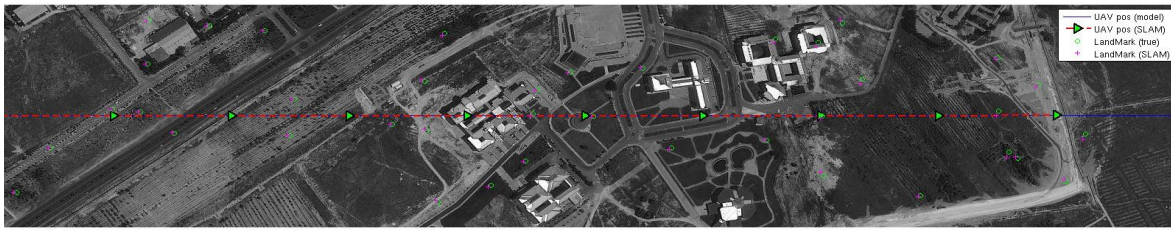
Şekil 5.13 Dairesel Rota

Çakışan rota (Sekiz Rota) ise rotaların birbiri ile kesiştiği durumda sistemin davranışını gözlemlemek için seçilmiştir. Şekil 5.14'te bu rotaya ait gösterim verilmiştir.



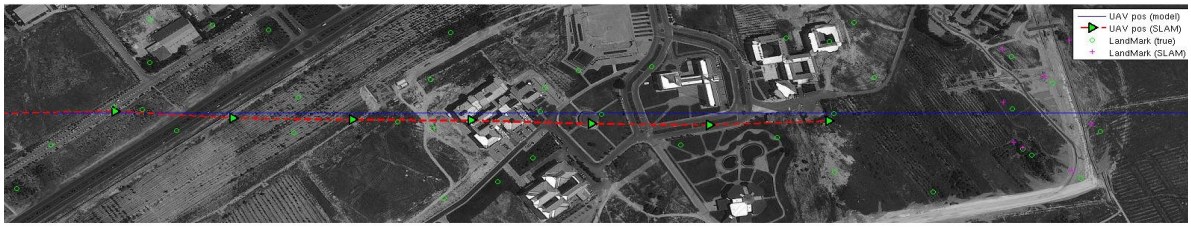
Şekil 5.14 Kesişen rota

Resim 5.7 'de ideal durumda hareket etmesi beklenen insansız hava arcına ati rota bilgisi gösterilmiştir.



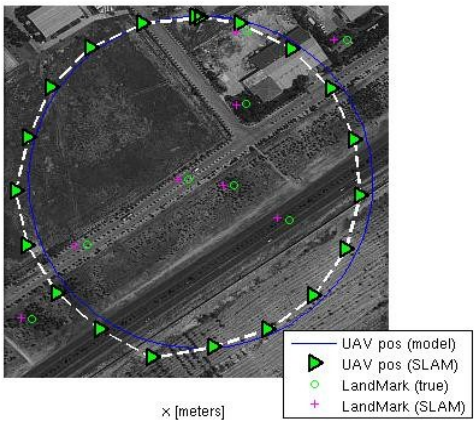
Resim 5.7 İdeal durumda İnsansız Hava Aracının Hareketi

Resim 5.8' de ise FPGA üzerinde tasarlanmış sisteme ait rota davranışı gösterilmiştir.



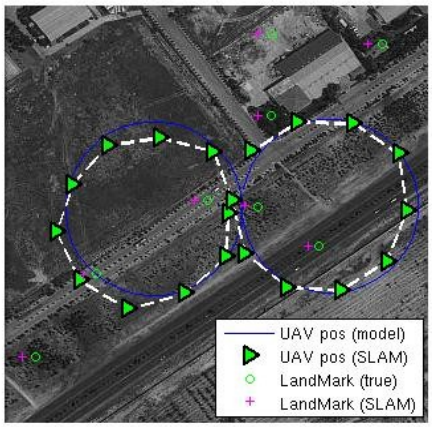
Resim 5.8 FPGA üzerinde çalışan sistemin davranışı (Doğrusal Rota)

Aynı arazi üzerinde İnsansız Hava Aracı dairesel uçuş yaparak rota takibi istenmiş ve gerçek durumla FPGA üzerinde çalışan sistemin durumunun görüntüsü Resim 5.9'da verilmiştir.



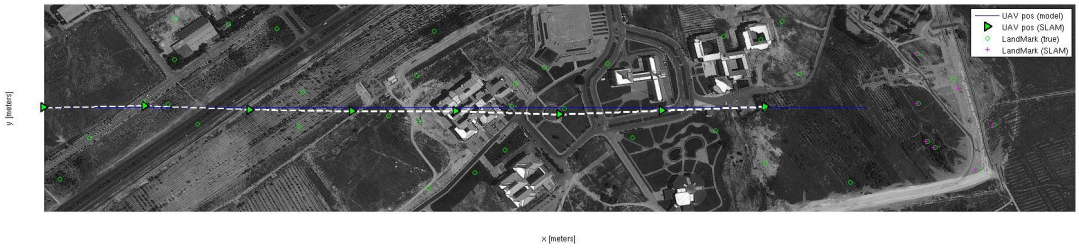
Resim 5.9 FPGA üzerinde çalışan sistemin davranışı (Dairesel Rota).

Resim 5.9'da bulunan görüntü üzerinde izlenen bir diğer rota'ya ait çizim Resim 5.10'da verilmiştir.

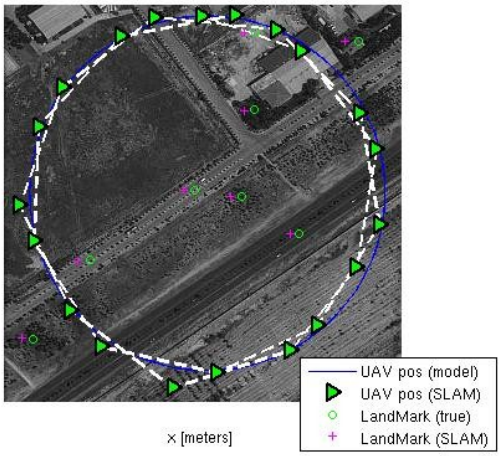


Resim 5.10 FPGA üzerinde çalışan sistemin davranışı (Kesişen Rota).

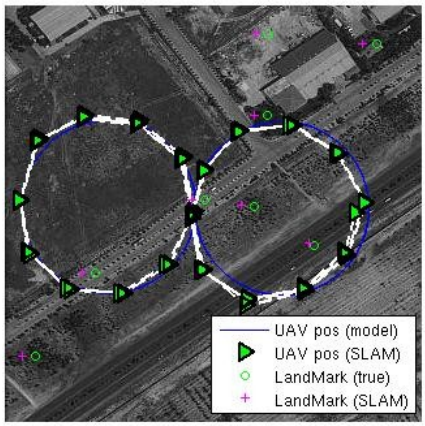
Yapılan çalışmada kıyaslama yapabilmek adına farklı bir senaryo daha izlenerek konum bilgilerinin belirli zaman aralıklarında hatasız alınabileceği kabul edilmiş ve bu durumdaki sistemin davranışı incelenmiştir. Aynı uçuş senaryoları tekrar edilmiş ve elde edilen sonuçlar sırası ile Resim 5.11, 5.12 ve 5.13'de verilmiştir.



Resim 5.11 Periyodik Zamanlı Veri-Rota davranışı (Doğrusal Rota)

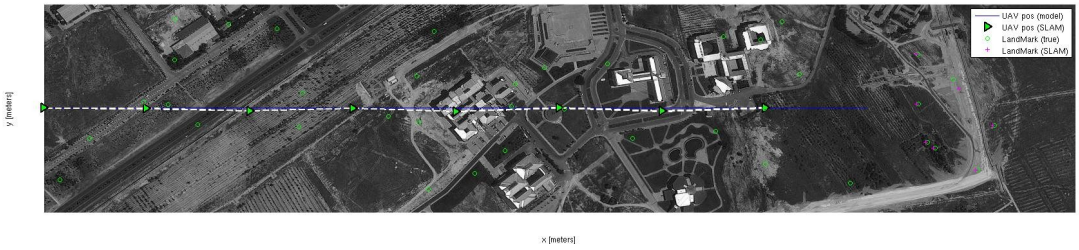


Resim 5.12 Periyodik Zamanlı Veri-Rota davranışı (Dairesel Rota)

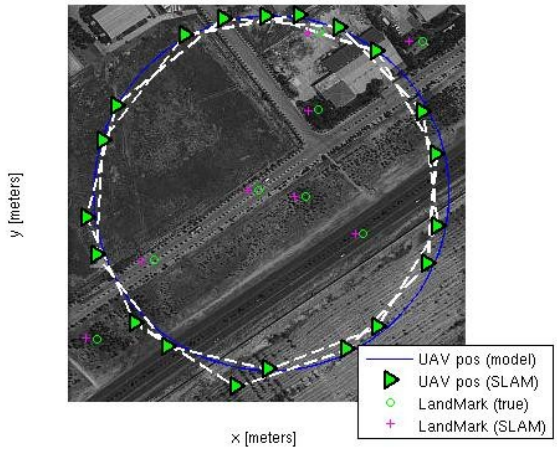


Resim 5.13 Periyodik Zamanlı Veri-Rota davranışı (Dairesel Rota)

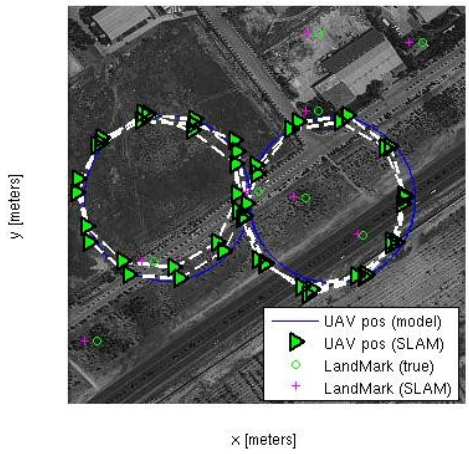
Uygulanan son senaryoda ise konum bilgisinin rastlantısal aralıklarla hatasız alındığı kabul edilmiş ve sistem davranışı incelenmiştir. Resim 5.14, 5.15 ve 5.16'da bu senaryoya ait görüntüler verilmiştir.



Resim 5.14 Rastgele Zamanlı Veri-Rota davranışı (Doğrusal Rota)



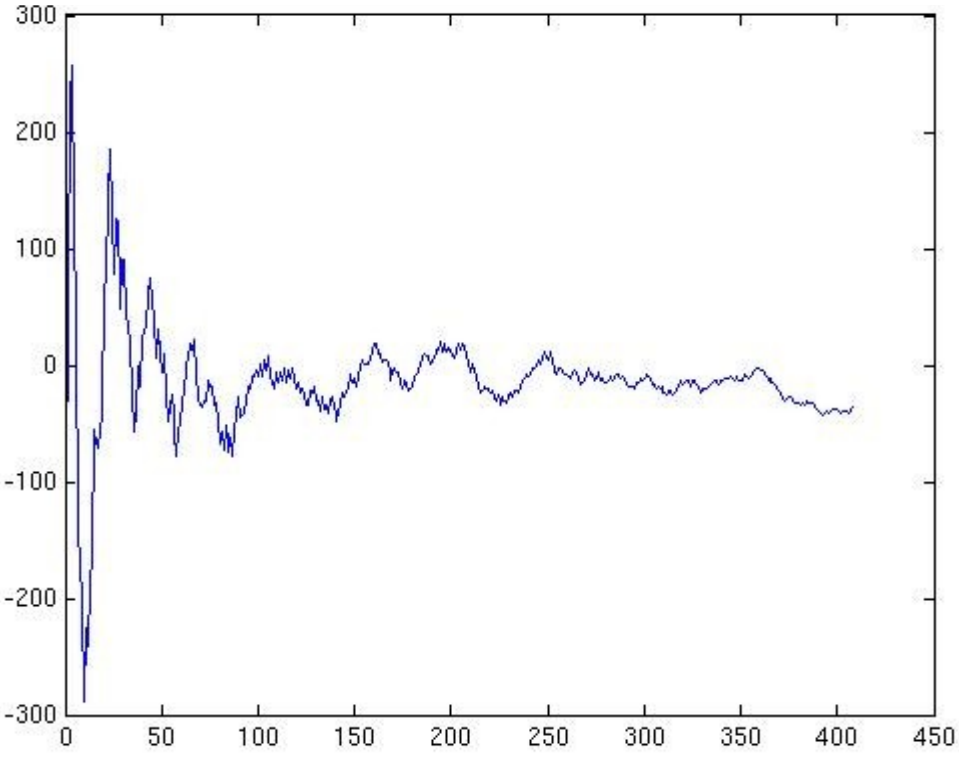
Resim 5.15 Rastgele Zamanlı Veri-Rota davranışı (Dairesel Rota)



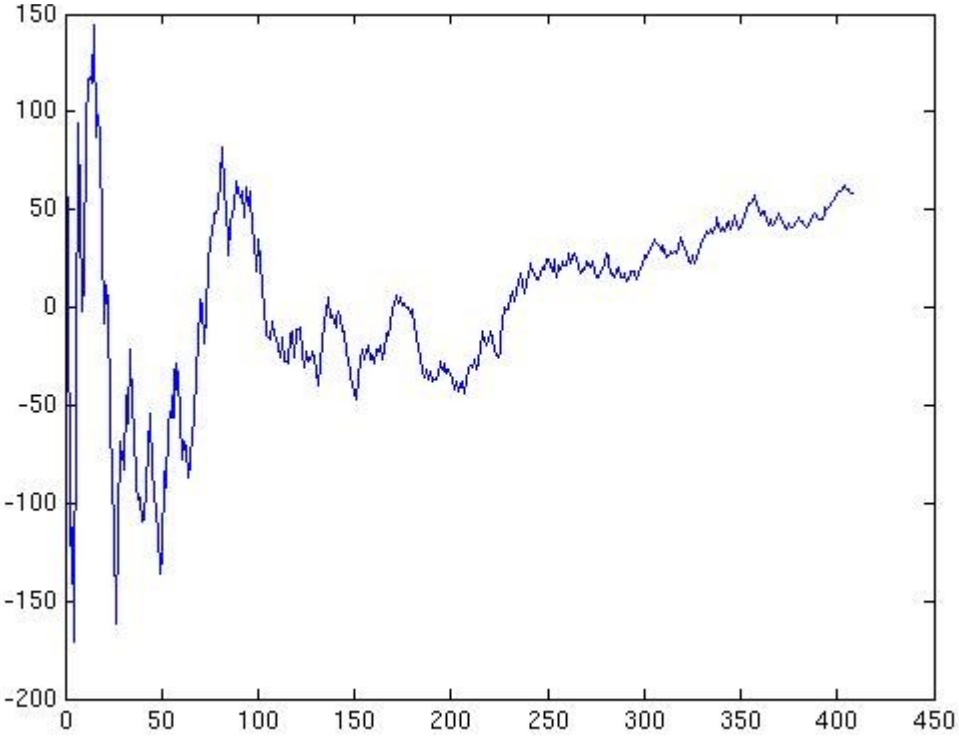
Resim 5.16 Rastgele Zamanlı Veri-Rota davranışı (Kesişen Rota)

Yapılan denemeler sonucunda rastgele zamanlarda ve periyodik aralıklarla alınan hatasız konum ölçümlerinin olduğu durumlarda sistemin davranışının daha iyiye gittiği görülmüştür. Diğer durumda konum bilgisinin alınan görüntüden çıkarılarak işlendiği durumda beklendiği şekilde hatanın arttığı gözlenmiştir.

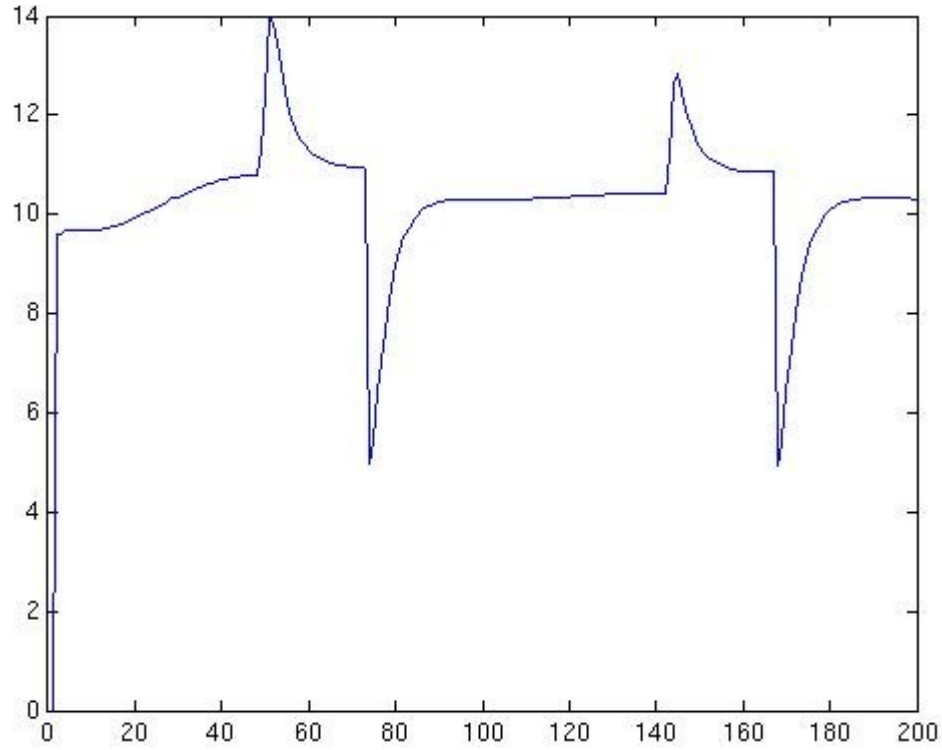
Sistemin davranışında temel olarak iki tür hatanın olduğu görülmüştür. İlki daha en başta kabul edilen ölçüm yöntemlerinden ve çevresel faktörlerden kaynaklandığı kabul edilen hatalardır. Bu hatalara ilişkin grafikler sırasıyla Şekil 5.15 - Şekil 5.20 aralığında verilmiştir.



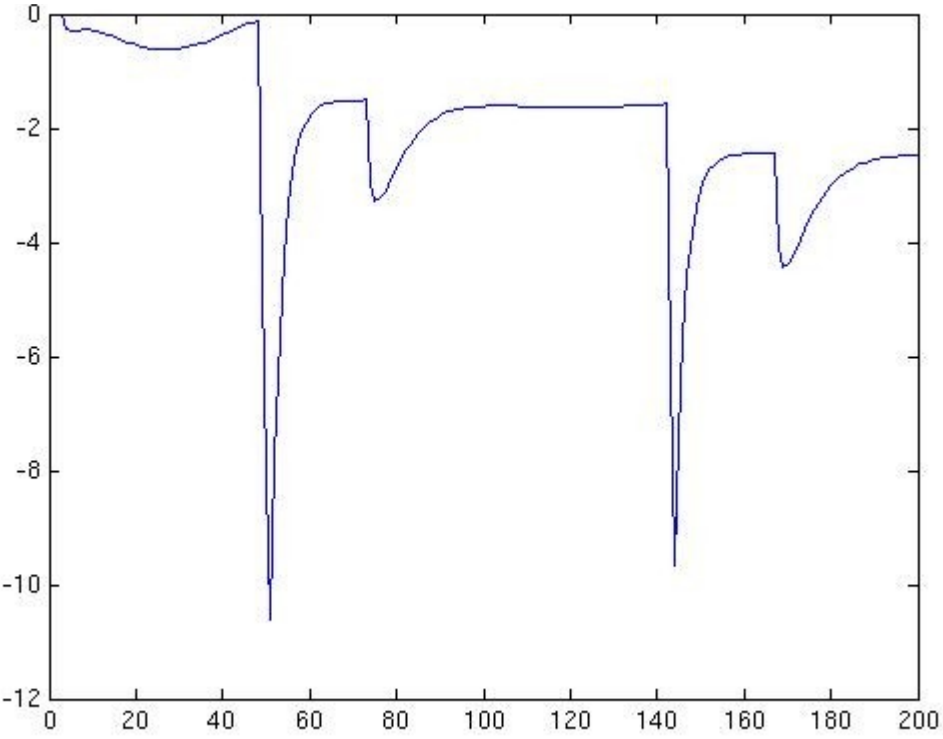
Şekil 5.15 Doğrusal rota X yönündeki hata.



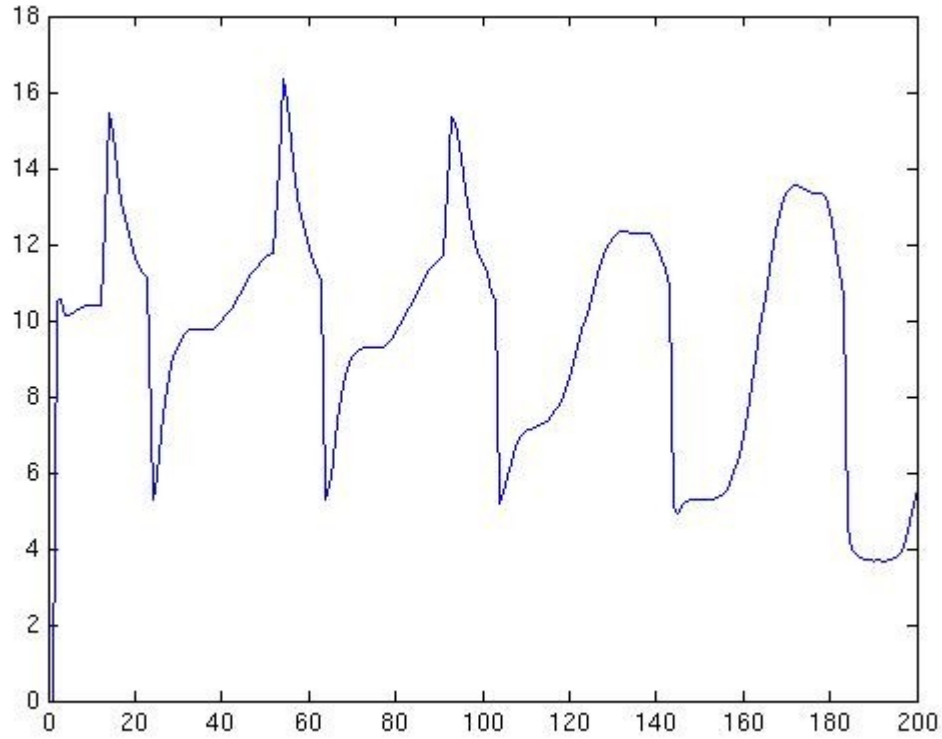
Şekil 5.16 Doğrusal rota Y yönündeki hata.



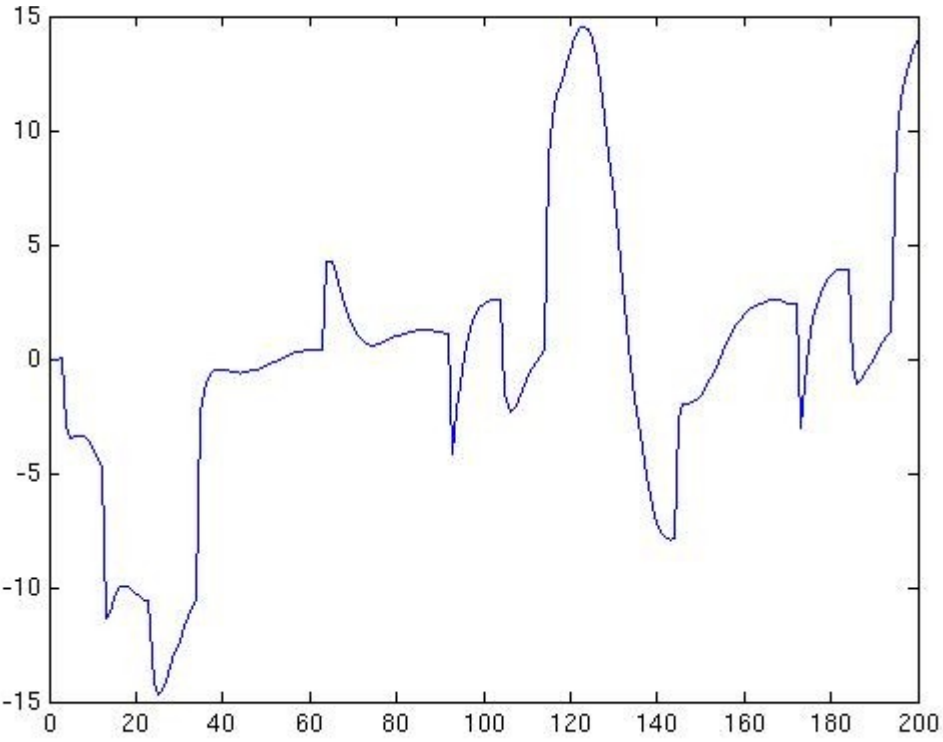
Şekil 5.17 Dairesel rota X Yönündeki hata.



Şekil 5.18 Dairesel rota Y Yönündeki hata.

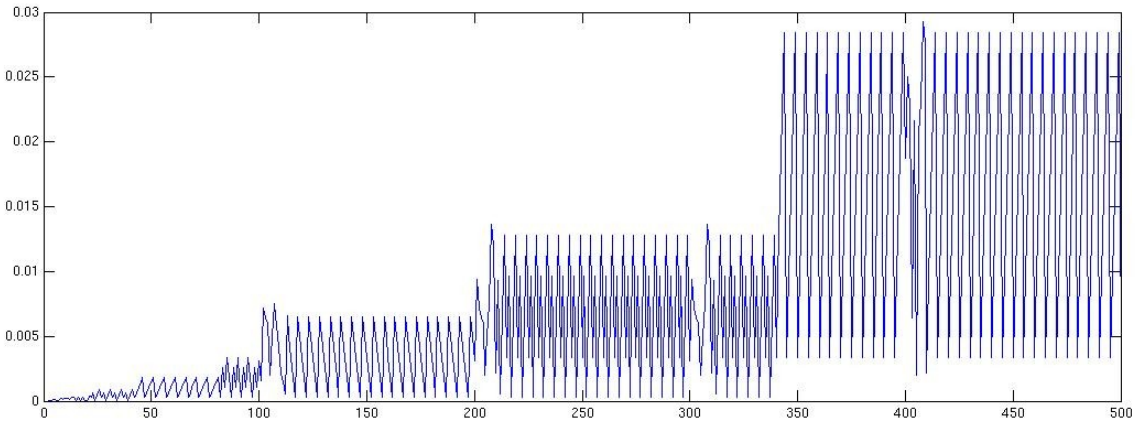


Şekil 5.19 Kesişen rota X yönündeki Hata

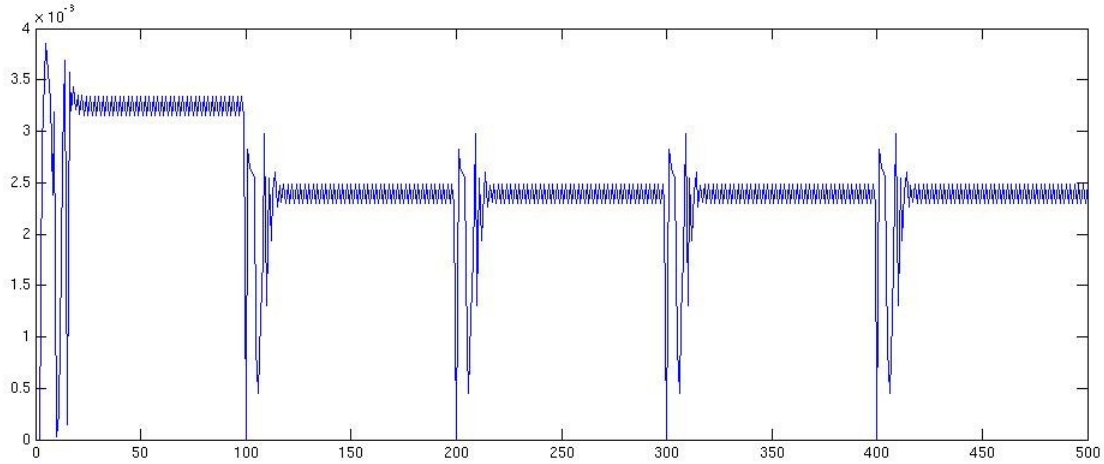


Şekil 5.20 Kesişen rota Y yönündeki hata.

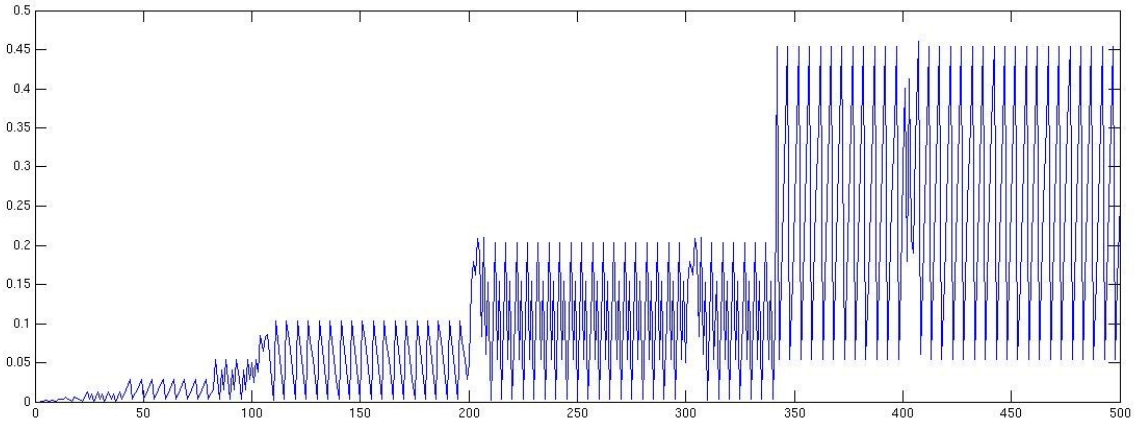
Dikkate alınan bir diğer hata türü ise sayı sisteminde kullanılan bit uzunluğuna bağlı olarak oluşan hatalardır. Her rota için farklı bit uzunluklarındaki grafikler sırasıyla Şekil 5.21- Şekil 5.32 aralığında verilmiştir. Hata ölçümü için 32 bitlik uzunluk referans kabul edilerek diğer bit uzunlukları bu referans ile karşılaştırılmıştır.



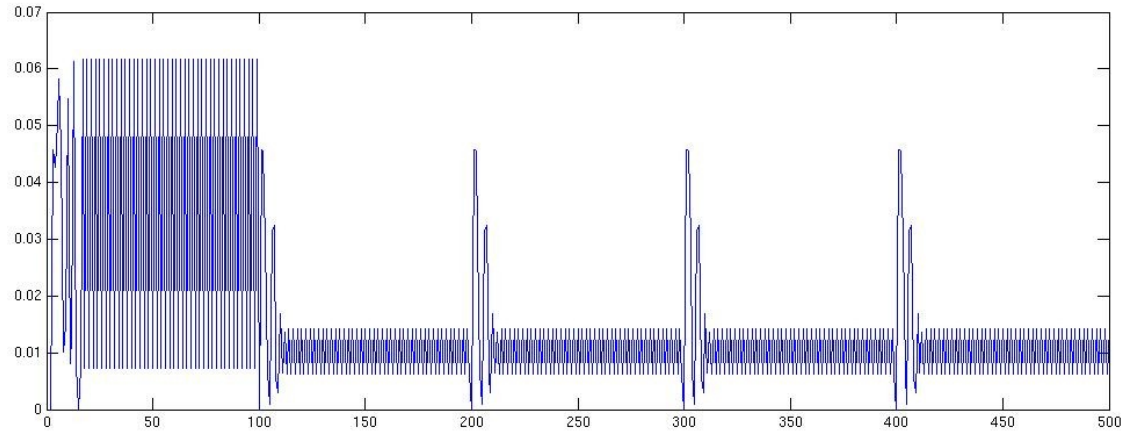
Şekil 5.21 Doğrusal rota X yönündeki hata (24 Bit Uzunluk İçin)



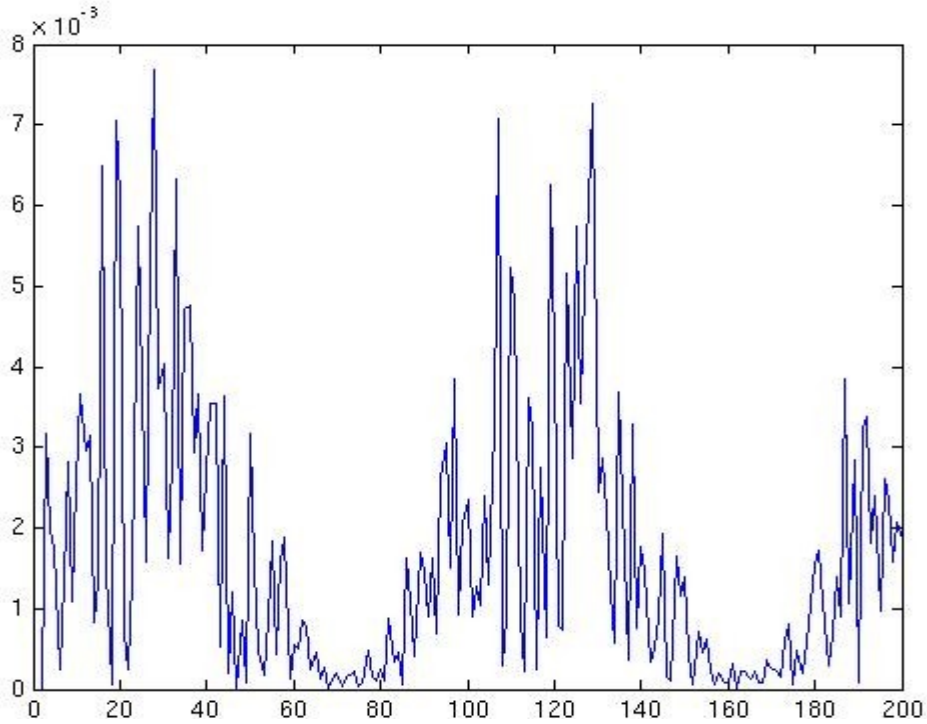
Şekil 5.22 Doğrusal rota Y yönündeki hata (24 Bit Uzunluk İçin)



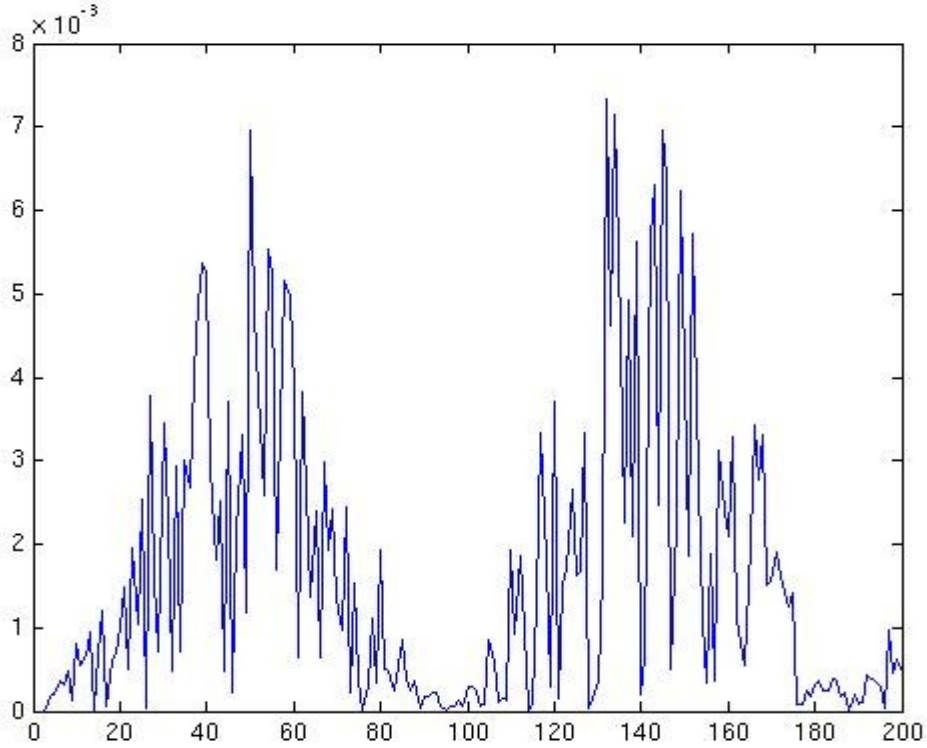
Şekil 5.23 Doğrusal rota X yönündeki hata (20 Bit Uzunluk İçin)



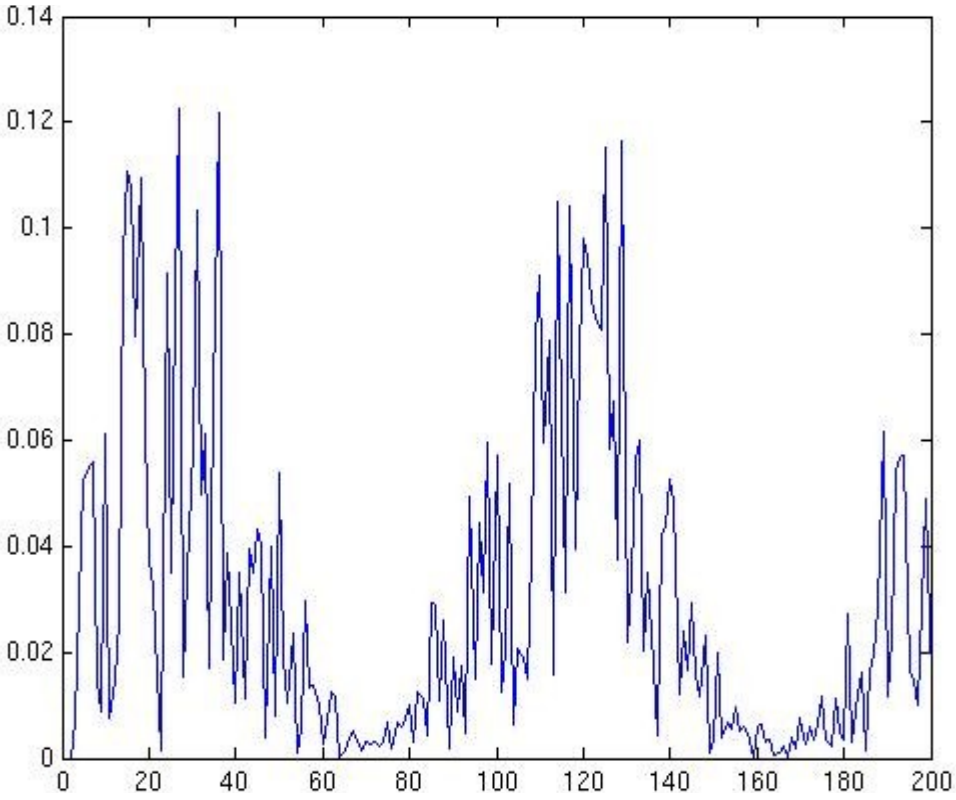
Şekil 5.24 Doğrusal rota Y yönündeki hata (20 Bit Uzunluk İçin)



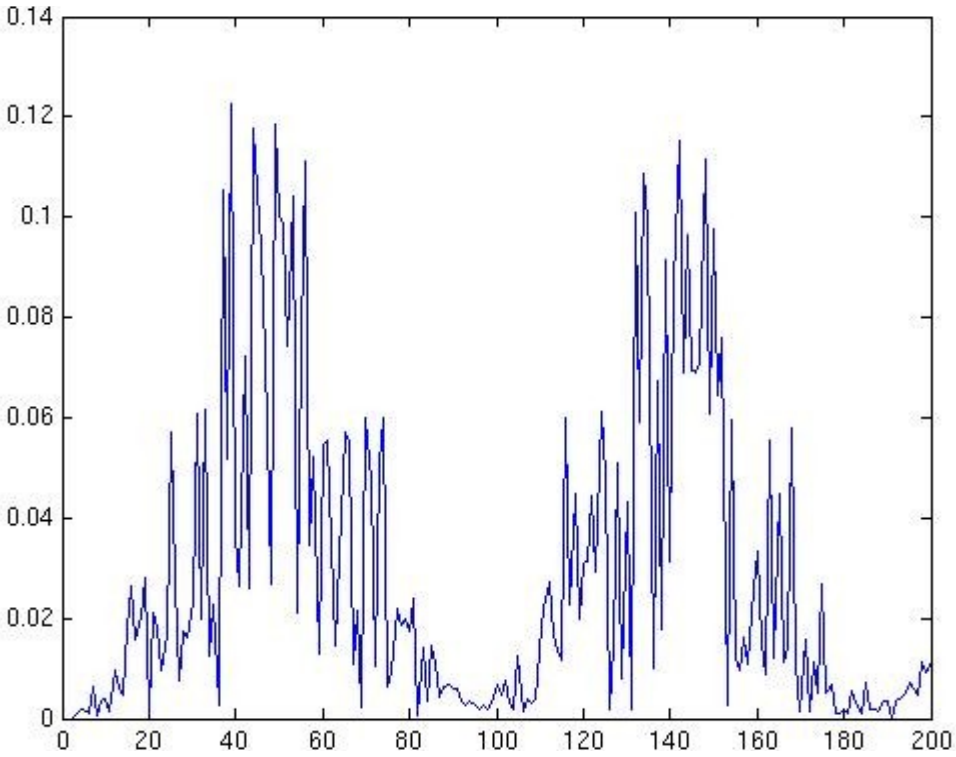
Şekil 5.25 Dairesel rota X yönündeki hata (24 Bit Uzunluk İçin)



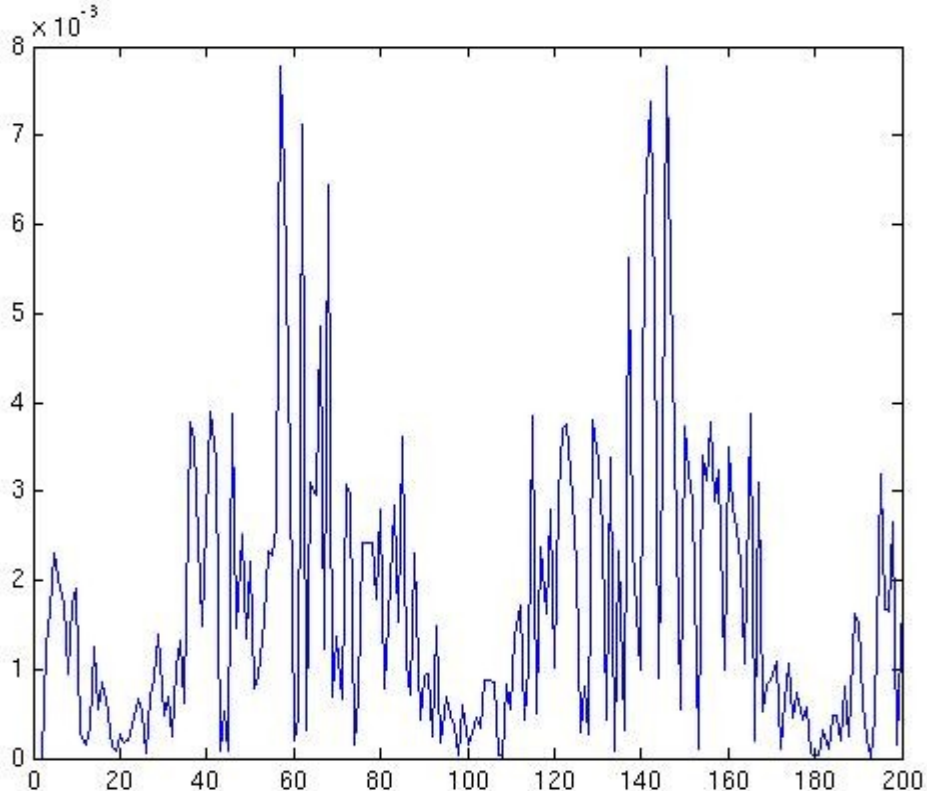
Şekil 5.26 Dairesel rota Y yönündeki hata (24 Bit Uzunluk İçin)



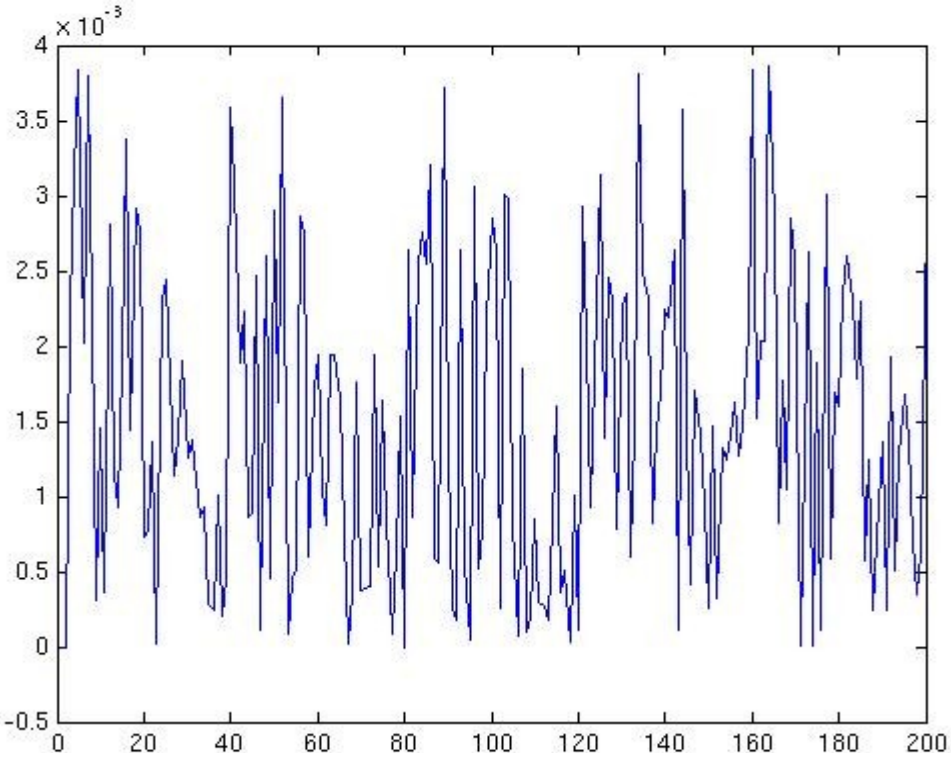
Şekil 5.27 Dairesel rota X yönündeki hata (20 Bit Uzunluk İçin)



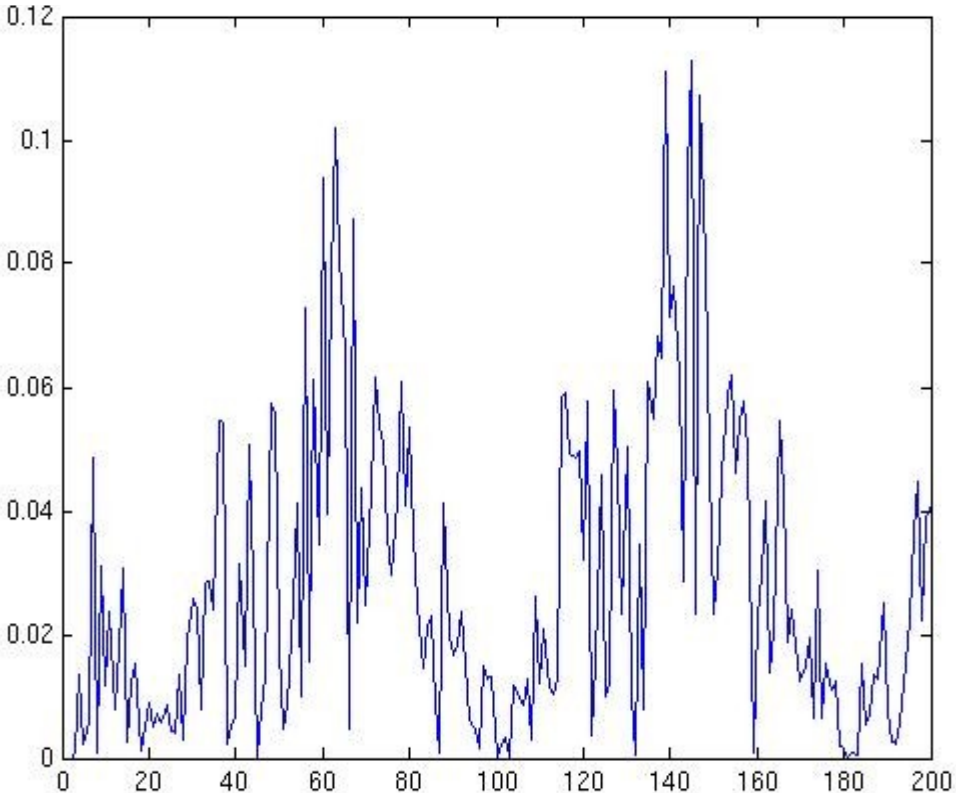
Şekil 5.28 Dairesel rota Y yönündeki hata (20 Bit Uzunluk İçin)



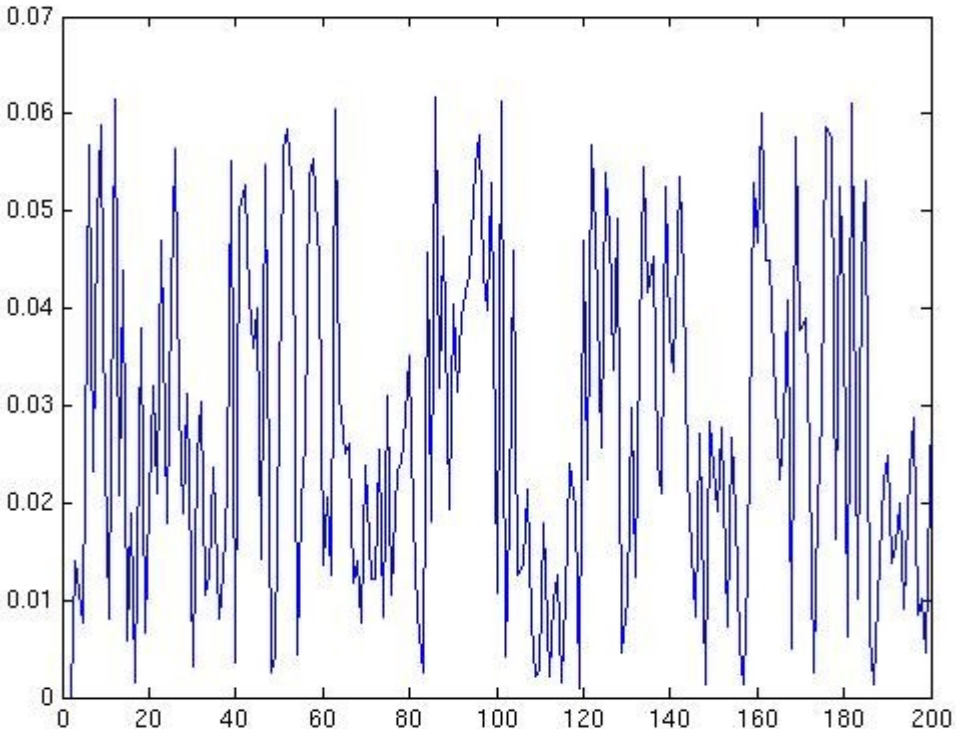
Şekil 5.29 Kesişen rota X yönündeki hata (24 Bit Uzunluk İçin)



Şekil 5.30 Kesişen rota Y yönündeki hata (24 Bit Uzunluk İçin)



Şekil 5.31 Kesişen rota X yönündeki hata (20 Bit Uzunluk İçin)



Şekil 5.32 Kesişen rota X yönündeki hata (20 Bit Uzunluk İçin)

Tasarlanan sisteme ait kaynak kullanım bilgileri ise Tablo 5.2'de verilmiştir.

Tablo 5.2 Genişletilmiş Kalman Süzgeci Donanım Kaynak Tüketimi

BİT UZUNLUĞU	Döngü	Register	%	LUT	%	DSP48
20 Bit	16542	14524	22	31660	46	12
24 Bit	16542	17070	25	37142	54	12
32 Bit	16542	21899	32	51972	76	12

BÖLÜM VI

SONUÇLAR ve TARTIŞMA

Tez çalışması sonucunda gerçek zamanlı uygulamalarda kullanılabilecek bir yapı oluşturulmuş olmakla beraber tasarlanan yapı iyi derecede optimize edilmemiştir. Bunun temel nedeni FPGA üzerindeki tasarımlarda kaynak kullanımı, hız ve tasarım esnekliği konularında bir açmaz olmasıdır. Tasarım hedefi olarak farklı problemlere kolayca uyarlanabilecek bir yapı oluşturulmak istendiği için bu açmaz kabul edilerek sistem tasarlanmıştır.

Donanımsal mimari tasarlanırken tekrar kullanılabilirlik adına genelleştirilmiş bir mimari tasarlanmış ve bu problemler dışında matris işlemleri gerektiren diğer problemlere de kolayca uyarlanabilecek bir yapı elde edilmiştir.

Çalışma sonucunda görülen bir diğer sorun ise Eş Zamanlı Haritalandırma ve Konumlama problemi gibi uygulamalarda sadece donanım tabanlı bir çözümün tasarımının oldukça zor olmasıdır. İhtiyaç duyulan işlevlerin yerine getirilmesi amacıyla içerisinde işlemci barındıran melez bir çözümün daha verimli olacağı düşünülmektedir.

Kayan noktalı sayı kullanılarak tasarlanan sistemin 32 bit uzunlukta kullanıldığında oldukça donanım kaynağı tükettiği görülmüş olup gerçek bir uygulama için 24 bitlik bir uzunluğun yeterli olacağı anlaşılmıştır.

Sistemin doğrudan olarak hız kıyaslaması yapılamamış olup ele alınan problemin modeline göre değişecek olan işlem yükü sebebiyle her tasarımın kendine has bir hız davranışı gözlenmiştir. Bir kıyaslama yapabilmek adına yapılan tasarımların toplam çalışma döngüleri hakkında bilgiler verilmiştir. Problem uyarlamasından sonra Xilinx ISE programındaki zamanlama analizlerine bakılarak sistemin toplam hızı hakkında bir kanıya varılabileceği görülmüştür.

SLAM problemi ele alındığında uzun süreli konum kestiriminde fazla miktarda yer

işaretçisinin işlenmesi gerektiği için işlem yükünün artacağı, yer işaretçilerinin sayısının azaltılması ile bu açmazın aşılabileceği fakat hata oranın artması gibi bir yan etkisi olduğu bununla beraber azaltılmış işaretçi sayısının donanımsal uyarlama adına yeterli başarımı sunabildiği görülmüştür.

Yapılan insansız hava aracı rota tayini uygulamalarında sistem davranışını gözlemlemek için 3 farklı senaryo 3 farklı rota davranışı için teker teker çalıştırılmıştır. İlk senaryoda konum bilgisinin hiç alınamadığı varsayılmış ve 3 farklı rota türü için rota kestirilmiştir. İkinci senaryoda ise konum bilgisinin periyodik aralıklarla alınabildiği kabul edilmiş ve 3 farklı rota üzerinde denenmiştir. Son uçuş senaryosunda ise konum bilgisinin rastgele zaman aralıklarında doğru bir şekilde alınabildiği kabul edilmiş ve 3 farklı rota için rota kestirimi buna göre yapılmıştır.

Yapılan çalışma sonucunda denenen uçuş senaryoları ele alındığında bu sistemin uzun süreli kullanımdan ziyade GPS v.b konum belirleme sistemlerinden gelen sinyalin koptuğu ya da belli aralıklarla ancak erişilebilen durumlarda kullanılmasının daha yerinde bir seçim olacağı anlaşılmıştır.

KAYNAKLAR

Agarwal, V., H. Arya, ve S. Bhaktavatsala. 2009. «Design and Development of a Real-Time DSP and FPGA-Based Integrated GPS-INS System for Compact and Low Power Applications». ***IEEE Transactions on Aerospace and Electronic Systems*** 45 (2) (Nisan): 443–454. doi:10.1109/TAES.2009.5089533.

Al-Dhaher, A. H.G, E. A Farsi, ve D. Mackesy. 2005. «Data Fusion Architecture - An FPGA Implementation». ***Proceedings of the IEEE Instrumentation and Measurement Technology Conference****, 2005. IMTC 2005*, 3:1985–1990. IEEE. doi:10.1109/IMTC.2005.1604519.

Arias-Garcia, J., R.P. Jacobi, C.H. Llanos, ve M. Ayala-Rincon. 2011. «A suitable FPGA implementation of floating-point matrix inversion based on Gauss-Jordan elimination». ***2011 VII Southern Conference on Programmable Logic (SPL)***, 263–268. doi:10.1109/SPL.2011.5782659.

Arias-Garcia, J., C.H. Llanos, M. Ayala-Rincon, ve R.P. Jacobi. 2012. «FPGA implementation of large-scale matrix inversion using single, double and custom floating-point precision». ***2012 VIII Southern Conference on Programmable Logic (SPL)***, 1–6. doi:10.1109/SPL.2012.6211787.

Bay, Herbert, Andreas Ess, Tinne Tuytelaars, ve Luc Van Gool. 2008. «Speeded-Up Robust Features (SURF)». ***Computer Vision and Image Understanding*** 110 (3) (Haziran): 346–359. doi:10.1016/j.cviu.2007.09.014.

Bonato, V., E. Marques, ve G. A Constantinides. 2007. «A Floating-Point Extended Kalman Filter Implementation for Autonomous Mobile Robots».***International Conference on Field Programmable Logic and Applications****, 2007. FPL 2007*, 576–579. IEEE. doi:10.1109/FPL.2007.4380720.

Bonato, V., R. Peron, D. F Wolf, J. A.M de Holanda, E. Marques, ve J. M.P Cardoso.

2007. «An FPGA Implementation for a Kalman Filter with Application to Mobile Robotics». ***International Symposium on Industrial Embedded Systems****, 2007. SIES '07*, 148–155. IEEE. doi:10.1109/SIES.2007.4297329.

Canny, John. 1986. «A Computational Approach to Edge Detection». ***IEEE Transactions on Pattern Analysis and Machine Intelligence*** **PAMI-8** (6): 679–698. doi:10.1109/TPAMI.1986.4767851.

Carson, C., S. Belongie, H. Greenspan, ve J. Malik. 2002. «Blobworld: image segmentation using expectation-maximization and its application to image querying». ***IEEE Transactions on Pattern Analysis and Machine Intelligence*** 24 (8): 1026–1038. doi:10.1109/TPAMI.2002.1023800.

Chappell, S., A. Macarthur, D. Preston, D. Olmstead, B. Flint, ve C. Sullivan. 2005. «Exploiting Real-time FPGA Based Adaptive Systems Technology for Real-time Sensor Fusion in Next Generation Automotive Safety Systems». ***Design, Automation and Test in Europe****, 2005. Proceedings*, 180– 185 Vol. 3. IEEE. doi:10.1109/DATE.2005.147.

Charoensak, C., ve S. S Abeysekera. 2004. «FPGA Implementation of Efficient Kalman Band-pass Sigma-delta Filter for Application in FM Demodulation». ***SOC Conference****, 2004. Proceedings. IEEE International*, 137– 138. IEEE. doi:10.1109/SOCC.2004.1362379.

———. 2005. «System on Chip FPGA Design of an FM Demodulator Using a Kalman Band-pass Sigma-delta Architecture». ***IEEE International Symposium on Circuits and Systems****, 2005. ISCAS 2005*, 33– 36 Vol. 1. IEEE. doi:10.1109/ISCAS.2005.1464517.

Davison, A.J., I.D. Reid, N.D. Molton, ve O. Stasse. 2007. «MonoSLAM: Real-Time Single Camera SLAM». ***IEEE Transactions on Pattern Analysis and Machine Intelligence*** 29 (6): 1052–1067. doi:10.1109/TPAMI.2007.1049.

Durrant-Whyte, H., ve Tim Bailey. 2006. «Simultaneous localization and mapping: part I». ***IEEE Robotics Automation Magazine*** 13 (2): 99–110. doi:10.1109/MRA.2006.1638022.

Echman, F., ve V. Owall. 2005. «A Scalable Pipelined Complex Valued Matrix Inversion Architecture». ***IEEE International Symposium on Circuits and Systems****, 2005. ISCAS 2005*, 4489– 4492 Vol. 5. IEEE. doi:10.1109/ISCAS.2005.1465629.

Edman, F., ve V. Owall. 2003. «Implementation of a Highly Scalable Architecture for Fast Inversion of Triangular Matrices». ***Proceedings of the 2003 10th IEEE International Conference on Electronics, Circuits and Systems****, 2003. ICECS 2003*, 3:1137– 1140 Vol.3. IEEE. doi:10.1109/ICECS.2003.1301712.

Eilert, J., Di Wu, ve D. Liu. 2007. «Efficient Complex Matrix Inversion for MIMO Software Defined Radio». ***IEEE International Symposium on Circuits and Systems****, 2007. ISCAS 2007*, 2610–2613. IEEE. doi:10.1109/ISCAS.2007.377850.

Elinas, P., R. Sim, ve J.J. Little. 2006. «/spl sigma/SLAM: stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution». ***Proceedings 2006 IEEE International Conference on Robotics and Automation****, 2006. ICRA 2006*, 1564–1570. doi:10.1109/ROBOT.2006.1641930.

Garbergs, B., ve B. Sohlberg. 1996. «Specialised Hardware for State Space Control of a Dynamic Process». *TENCON '96. Proceedings. 1996 IEEE TENCON.* ***Digital Signal Processing Applications***, 2:895–899 vol.2. IEEE. doi:10.1109/TENCON.1996.608466.

Gonzalez, A., J. Codol, ve M. Devy. 2011. «A C-embedded algorithm for real-time monocular SLAM». ***2011 18th IEEE International Conference on Electronics, Circuits and Systems (ICECS)***, 665–668. doi:10.1109/ICECS.2011.6122362.

Harris, Chris, ve Mike Stephens. 1988. «A combined corner and edge detector». ***In Proc. of Fourth Alvey Vision Conference***, 147–151.

Idris, Mohd. Yamani Idna, Hamzah Arof, Noorzaily Mohamed Noor, Emran Mohd. Tamil, ve Zaidi Razak. 2012. «A co-processor design to accelerate sequential monocular SLAM EKF process». ***Measurement*** 45 (8) (Ekim): 2141–2152. doi:10.1016/j.measurement.2012.05.018.

Idris, Mohd Yamani Idna, N.M. Noor, E.M. Tamil, Zaidi Razak, ve H. Arof. 2010. «Parallel Matrix Multiplication Design for Monocular SLAM». ***2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation (AMS)***, 492–497. doi:10.1109/AMS.2010.100.

Islam, A., J. M.P Langlois, ve A. Noureldin. 2009. «A Design Methodology for the Implementation of Embedded Vehicle Navigation Systems». ***IEEE International Conference on Electro/Information Technology****, 2009. Eit '09*, 297–300. IEEE. doi:10.1109/EIT.2009.5189630.

Kalman, R.E. 1960. «A new approach to linear filtering and prediction problems». ***Journal of basic Engineering*** 82 (Series D): 35–45.

Kim, J., ve S. Sukkarieh. 2004. «Autonomous airborne navigation in unknown terrain environments». ***IEEE Transactions on Aerospace and Electronic Systems*** 40 (3): 1031–1045. doi:10.1109/TAES.2004.1337472.

LaRoche, I., ve S. Roy. 2006. «An Efficient Regular Matrix Inversion Circuit Architecture for MIMO Processing». ***2006 IEEE International Symposium on Circuits and Systems****, 2006. ISCAS 2006. Proceedings*. IEEE. doi:10.1109/ISCAS.2006.1693709.

Lee, C.R., ve Z. Salcic. 1997. «A fully-hardware-type maximum-parallel architecture for Kalman tracking filter in FPGAs». ***Information, Communications and Signal Processing****, 1997. ICICS., Proceedings of 1997 International Conference on*, 1243–1247 vol.2. doi:10.1109/ICICS.1997.652183. 10.1109/ICICS.1997.652183.

Leonard, J.J., ve H.F. Durrant-Whyte. 1991. «Simultaneous map building and localization for an autonomous mobile robot». ***IEEE/RSJ International Workshop on Intelligent Robots and Systems*** *'91. 'Intelligence for Mechanical Systems, Proceedings IROS '91*, 1442–1447 vol.3. doi:10.1109/IROS.1991.174711.

Lindeberg, T. 1996. «Edge detection and ridge detection with automatic scale selection». ***Proceedings CVPR '96, 1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition****, 1996*, 465–470. doi:10.1109/CVPR.1996.517113.

Lowe, David G. 2004. «Distinctive Image Features from Scale-Invariant Keypoints». ***International Journal of Computer Vision*** 60: 91–110.

Matsumoto, Makoto, ve Takuji Nishimura. 1998. «Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator». ***ACM Trans. Model. Comput. Simul***. 8 (1) (Ocak): 3–30. doi:10.1145/272991.272995.

McElhoe, Bruce A. 1966. «An Assessment of the Navigation and Course Corrections for a Manned Flyby of Mars or Venus». ***IEEE Transactions on Aerospace and Electronic Systems*** AES-2 (4): 613–623. doi:10.1109/TAES.1966.4501892.

Mc Gee, L. A., S. F. Schmidt, ve G. L. Smith. 1962. *APPLICATION OF STATISTICAL FILTER THEORY TO THE OPTIMAL ESTIMATION OF POSITION AND VELOCITY ON BOARD A CIRCUMLUNAR VEHICLE*. http://archive.org/details/nasa_techdoc_19620006857.

Newman, P., D. Cole, ve K. Ho. 2006. «Outdoor SLAM using visual appearance and laser ranging». ***Proceedings 2006 IEEE International Conference on Robotics and Automation****, 2006. ICRA 2006*, 1180–1187. doi:10.1109/ROBOT.2006.1641869.

Rosa, Leandro de Souza, ve Vanderlei Bonato. 2012. «A Method to Convert Floating to Fixed-point EKF-SLAM for Embedded Robotics». ***Journal of the Brazilian Computer***

Society (Kasım): 1–12. doi:10.1007/s13173-012-0092-4.

Salcic, Z., ve C. -R Lee. 2001. «FPGA-based Adaptive Tracking Estimation Computer». ***IEEE Transactions on Aerospace and Electronic Systems*** 37 (2) (Nisan): 699–706. doi:10.1109/7.937481.

Se, S., D.G. Lowe, ve J.J. Little. 2005. «Vision-based global localization and mapping for mobile robots». ***IEEE Transactions on Robotics*** 21 (3): 364–375. doi:10.1109/TRO.2004.839228.

Smith, Randall C., ve Peter Cheeseman. 1986. «On the Representation and Estimation of Spatial Uncertainty». ***The International Journal of Robotics Research*** 5 (4) (Ocak 12): 56–68. doi:10.1177/027836498600500404.

Smith, R., M. Self, ve P. Cheeseman. 1987. «Estimating uncertain spatial relationships in robotics». ***1987 IEEE International Conference on Robotics and Automation. Proceedings***, 4:850–850. doi:10.1109/ROBOT.1987.1087846.

Smith, Stephen. 1992. «A New Class of Corner Finder». *BMVC92*, editör David Hogg Bsc DPhil MSc ve Roger Boyle BA, 139–148. Springer London. http://link.springer.com/chapter/10.1007/978-1-4471-3201-1_15.

Steux, B., ve O. El Hamzaoui. 2010. «tinySLAM: A SLAM algorithm in less than 200 lines C-language program».***2010 11th International Conference on Control Automation Robotics Vision (ICARCV)***, 1975–1979 . doi:10.1109/ICARCV.2010.5707402.

Turney, R. D, A. M Reza, ve J. G.R Delva. 1999. «FPGA Implementation of Adaptive Temporal Kalman Filter for Real Time Video Filtering». ***1999 IEEE International Conference on Acoustics, Speech, and Signal Processing****, 1999. Proceedings*, 4:2231–2234 vol.4. IEEE. doi:10.1109/ICASSP.1999.758380.

Veera Ragavan, S., V. Ganapathy, ve E. Xian. 2009. «A Reconfigurable FPGA Framework for Data Fusion in UAV's.» ***World Congress on Nature & Biologically Inspired Computing****, 2009. NaBIC 2009*, 1626–1631. IEEE. doi:10.1109/NABIC.2009.5393662.

Vincke, Bastien, Abdelhafid Elouardi, ve Alain Lambert. 2012. «Real Time Simultaneous Localization and Mapping: Towards Low-cost Multiprocessor Embedded Systems». ***EURASIP Journal on Embedded Systems*** 2012 (1) (Aralık 1): 1–14. doi:10.1186/1687-3963-2012-5.

Vincke, B., A. Elouardi, ve A. Lambert. 2011. «Multiprocessing improvements on a low-cost system based Simultaneous Localization and Mapping». ***2011 International Conference on Multimedia Computing and Systems (ICMCS)***, 1–5. doi:10.1109/ICMCS.2011.5945612.

Wei-Tsen Lin, ve Dah-Chung. 2007. «Adaptive Carrier Synchronization Using Decision-Aided Kalman Filtering Algorithms». ***IEEE Transactions on Consumer Electronics*** 53 (4) (Kasım): 1260–1267. doi:10.1109/TCE.2007.4429210.

Wei-Tsen Lin, ve Dah-Chung Chang. 2005. «Design of a QAM Receiver with the Kalman Algorithm for Adaptive Carrier Synchronization». ***12th IEEE International Conference on Electronics, Circuits and Systems****, 2005. ICECS 2005*, 1–4. IEEE. doi:10.1109/ICECS.2005.4633603.

———. 2006. «The Extended Kalman Filtering Algorithm for Carrier Synchronization and the Implementation». ***2006 IEEE International Symposium on Circuits and Systems****, 2006. ISCAS 2006. Proceedings*. IEEE. doi:10.1109/ISCAS.2006.1693514.

XILINX. 2003. «Linear Feedback Shift Register v3.0». XILINX.

Yang Liu, C. -S Bouganis, ve P. Y.K Cheung. 2007. «Efficient Mapping of a Kalman Filter into an FPGA Using Taylor Expansion». ***International Conference on Field***

***Programmable Logic and Applications**, 2007. FPL 2007*, 345–350. IEEE. doi:10.1109/FPL.2007.4380670.

Zhi-Jian Sun, ve Xue-Mei Liu. 2008. «Application of Floating Point DSP and FPGA in Integration Navigation System». ***2008 International Conference on Computer Science and Software Engineering***, 4:58–61. IEEE. doi:10.1109/CSSE.2008.1526.

Zhou, Jie, Yong Dou, Jianxun Zhao, Fei Xia, Yuanwu Lei, ve Yuxing Tang. 2009. «A Fine-Grained Pipelined Implementation for Large-Scale Matrix Inversion on FPGA». ***Proceedings of the 8th International Symposium on Advanced Parallel Processing Technologies***, 110–122. APPT '09. Berlin, Heidelberg: Springer-Verlag. doi:10.1007/978-3-642-03644-6_9. http://dx.doi.org/10.1007/978-3-642-03644-6_9.

ÖZGEÇMİŞ

Mehmet Muzaffer KÖSTEN 24.05.1986 tarihinde Bursa ili Yenişehir ilçesinde doğdu. İlk okul eğitimini Bilecik Osmaneli Atatürk İlkokulu'nda tamamladı. 1997 yılında Osmaneli 75. Yıl Anadolu Lisesi'ni kazandı. Hazırlık sınıfını ve ilk yılını burada tamamladıktan sonra geri kalan eğitimini Bursa Yenişehir Ertuğrul Gazi Anadolu Lisesi'nde tamamladı. 2004 Yılında Niğde Üniversitesi Elektrik Elektronik Mühendisliği Bölümünü kazandı. 2008 yılında mezun oldu ve aynı yıl yüksek lisans eğitimine başladı. 2011 yılında Erasmus Programı ile yüksek lisans eğitiminin 6 aylık kısmını Polonya'da Uniwersytet Technologiczno-Przyrodniczy üniversitesinde tamamladı. 2011-2012 döneminde askerlik görevini yerine getirdi. Ocak 2013'te Niğde Üniversitesi Bilgisayar Mühendisliği Donanım Anabilim Dalına araştırma görevlisi olarak atandı ve halen bu bölümde çalışmaktadır. Bilim dalındaki ilgi alanı ise gömülü sistem tasarımıdır.