TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DÖRT ROTORLU BİR İNSANSIZ HAVA ARACINA FIELD PROGRAMMABLE GATE ARRAY TABANLI İRTİFA KONTROLCÜSÜ TASARIMI

YÜKSEK LİSANS TEZİ

Bekircan KEÇEOĞLU

Elektrik ve Elektronik Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Coşku KASNAKOĞLU

Ağustos 2021

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Bekircan Keçeoğlu

ÖZET

Yüksek Lisans

DÖRT ROTORLU BİR İNSANSIZ HAVA ARACINA FIELD PROGRAMMABLE GATE ARRAY TABANLI İRTİFA KONTROLCÜSÜ TASARIMI

Bekircan Keçeoğlu

TOBB Ekonomi ve Teknoloji Üniveritesi
Fen Bilimleri Enstitüsü
Elektrik ve Elektronik Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Coşku Kasnakoğlu

Tarih: Ağustos 2021

Günümüz modern dünyasında, insansız hava araçları otonom hareket eden veya uzaktan bir insan tarafından kontrol edilebilen araçlar anlamına gelmektedir. İnsansız hava araçları farklı alanlarda kullanılmakta ve bu alanlara göre farklı tasarımları bulunmaktadır. İnsansız hava araçlarının başlıca kullanıldığı yerler; askeri amaçlı keşif, sınır gözetleme ve saldırıdır. Askeri amaçlar dışında İHA'lar; tarımda gözetleme ve ilaçlama faaliyetlerinde, taşımacılıkta, orman yangını tespiti ve söndürme çalışmalarında, doğal afet bölgelerinde arama faaliyetlerinde ve bölgeyi haritalamada kullanılmaktadır. İnsansız hava araçlarının otopilotlarında genel olarak mikroişlemciler tercih edilmektedir. Ancak yüksek hız gerektiren uygulamalarda Field Programmable Gate Array'ler (FPGA) işlemciler yerine tercih edilebilir. Bu işlemciler gelen sensör ve komuta kontrol verilerini okuyarak, gerekli olan karar mekanizmasını gerçeklerler. İnsansız hava araçları uçuş eksenleri için oransal, integral, diferansiyel kapalı çevrim kontrolcüleri bulundurmaktadır. Bu üç yapı kazanç olarak nitelendirilmektedir ve testler ile beraber değerleri ayarlanmalıdır. İnsansız hava araçlarından biri olan dört rotorlu insansız hava araçları dikey iniş – kalkış yapabilmeleri ve kapalı alanlarda da kullanılabilir olmaları sebebiyle farklı sektörler tarafından yaygın olarak kullanılmaktadır. Bu tez çalışması kapsamında bir Quadrotorun doğrusal olmayan matematiksel modeli oluşturulmuştur. İrtifa modeli Simulink üzerinde gerçeklenmiştir. Model üzerine PID kontrolcü eklenmiş ve simülasyonları yapılmıştır. Sonraki adımda irtifa kontrolcünün FPGA üzerinde

gerçeklendiği adıma geçilmiştir. İrtifa kontrolcü FPGA üzerinde farklı yöntemler ve yazılımlar ile gerçeklenebilmektedir. İlk uygulamada FPGA tabanlı kontolcü, Xilinx FPGA tasarım blokları ile Xilinx System Generator üzerinde oluşturulmuştur. Farklı irtifa değerleri ile test edilmiştir ve referans takibi doğrulanmıştır. İkinci uygulamada ise HDL coder ile sentezlenebilir kodları üretmek için PID irtifa kontrolcü Matlab üzerinde gerçeklenmiştir. Referans takibi doğrulandıktan sonra sentezlenebilir VHDL kodları oluşturulmuştur. Üçüncü yöntemde, popülaritesi artmakta olan high level synthesis ile irtifa kontrolcü oluşturulmuş ve HLS üzerinde test edilmiştir. Dördüncü ve beşinci yöntemler FPGA üzerinde gerçeklenmiştir. Dördüncü yöntemde PID irtifa kontrolcü Xilinx FPGA üzerine yerleştirilen softcore işlemci "MicroBlaze" üzerinde gerçeklenmiştir. Beşinci uygulamada sayısal irtifa kontrolcü IP'si VHDL ile oluşturulmuş ve Zynq ile entegre edilerek testleri gerçekleştirilmiştir. İki yöntemde de referans takibi "Hardware in Loop" yöntemi ile Matlab Intrumentation Toolbox kullanılarak test edilmiştir. Test sistemine parametre belirsizliği ve gürültü eklenerek irtifa kontrolcüsünün işlevselliği doğrulanmıştır. Sonuç bölümünde ise FPGA tabanlı irtifa kontrolcü tasarımlarında kullanılan farklı yöntemlerin performans, kaynak kullanımı ve güç tüketimi yönünden karşılaştırması yapılmıştır.

Anahtar Kelimeler: İnsansız hava aracı, Quadrotor, Matematiksel model, Oransal integral türevsel kontrolcü, Field Programmable Gate Array, Xilinx System Generator, HDL Coder, HLS, MicroBlaze, Zynq, Hardware in Loop

ABSTRACT

Master of Science

FIELD PROGRAMMABLE GATE ARRAY BASED ALTITUDE CONTROLLER DESIGN FOR A FOUR ROTOR UNMANNED AERIAL VEHICLE

Bekircan Keceoglu

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Electrical and Electronics Engineering Science Programme

Supervisor: Prof. Dr. Cosku Kasnakoglu

Date: August 2021

In today's modern world, unmanned aerial vehicles mean vehicles that can move autonomously or can be controlled remotely by a human. Unmanned aerial vehicles are used in different areas and have different designs according to these areas. The fields where unmanned aerial vehicles are mainly used; military reconnaissance, border surveillance and attack. Other than military purposes UAV's are used for; in monitoring and spraying activities in agriculture, in transportation by commercial companies, in forest fire detection and extinguishing works, exploration activities and mapping in natural disaster areas. Microprocessors are generally preferred for autopilots of unmanned aerial vehicles. However, in applications requiring high speed, Field Programmable Gate Arrays (FPGA) can be preferred over processors. These processors read the incoming sensor and command control data and make the necessary decision mechanism. Unmanned aerial vehicles have proportional, integral and differential closed loop controllers for vehicle flight axes. These three elements are considered gains and their values must be tunned with the experiments. Four – rotor unmanned aerial vehicles, which are one of the unmanned aerial vehicles, are frequently preferred by researchers because they can take off and land vertically. They can also be used in indoor environments. Within the scope of this thesis non-linear mathematical model of a quadrotor was created. Quadrotor altitude model was implemented on Simulink. PID altitude controller was added to the realized model and simulations were made. In the next step, FPGA based altitude controller is presented.

Altitude controller can be implemeted with different approaches and software design tools. At first method, FPGA based altitude controller was designed with Xilinx FPGA design blocks on Xilinx System Generator. Controller was tested with different referance altitudes and reference tracking also verified. At second method, altitude controller was implemeted on Matlab and reference tracking was verified. After reference tracking was verified, HDL codes were generated using HDL coder. At third method, altitude controller was implemented and tested with HLS which is growing in popularity recent years. Fourth and fifth methods were implemented on FPGA. At fourth method, PID altitude controller was implemented on FPGA using softcore microprocessor called “Microblaze”. At fifth method, digital altitude controller IP was implemeted using VHDL and integrated with Zynq before testing functionality. At fourth and fifth methods reference tracking was verified using “Hardware in Loop” technique with Matlab Instrumentation Toolbox. Parameter uncertainty and noise parameters were added to design and alttitude controller functionality was tested. At final section of thesis, different approaches that were used at FPGA based alttitude controller, were compared to their performance, resource utility and power consumption.

Keywords: Unmanned Aerial Vehicle, Quadrotor, Mathematical model, Proportional integral derivative controller, Field Programmable Gate Array, Xilinx System Generator, HDL Coder, HLS, MicroBlaze, Zynq, Hardware in Loop

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Prof. Dr. Coşku Kasnakoğlu'na, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Elektrik Elektronik Mühendisliği Bölümü öğretim üyelerine ve destekleriyle her zaman yanımda olan annem Nermin Keçeoğlu, babam Hasan Keçeoğlu ve değerli iş arkadaşlarıma çok teşekkür ederim.

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

KISALTMALAR

İHA **:** İnsansız Hava Aracı
DOF **:** Degree of Freedom
PID **:** Proportional Integral Derivative
FPGA **:** Field Programmable Gate Array
HLS **:** High level synthesis
HIL **:** Hardware in loop

SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
m	Quadrotor ağırlığı
F	Kuvvet
g	Yer çekimi
I_x	X ekseni için eylemsizlik momenti
I_y	Y ekseni için eylemsizlik momenti
I_z	Z ekseni için eylemsizlik momenti
U	İrtifa kontrolcü girdisi

RESİM LİSTESİ

Sayfa

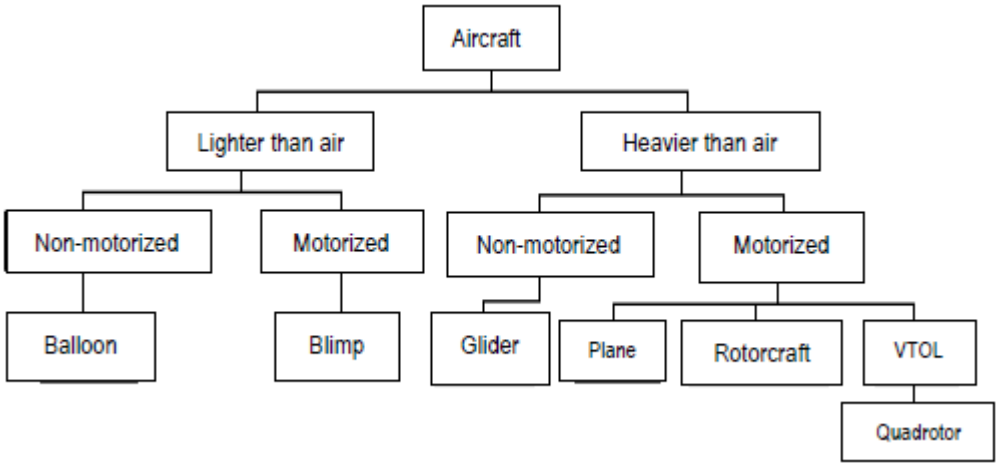
1. GİRİŞ

Modern insansız hava araçları (İHA) günümüzde kullanım alanları gelişen teknoloji ile birlikte giderek yaygınlaşmaktadır (Resim 1.1). Bu gelişmelere örnek olarak gelişen ve ucuzlayan sensör teknolojisi (gyro, GPS, ivmeölçerler, kablosuz haberleşme sensörleri, kameralar), daha hızlı mikroişlemciler – gelişen Field Programmable Gate Array (FPGA) teknolojisi, minyatür motor teknolojileri, yüksek kapasiteli güç kaynakları verilebilir. Bu gelişmeler ile İHA'ların kullanım amaçları da çeşitlenmektedir. Başta sadece askeri gözetleme ve saldırı amaçları için kullanılan İHA'lar artık sivil amaçlar ile de kullanılmaktadır. Bu amaçlara; afet anında arama ve kurtarma, çoğrafi haritalama, tarımda keşif ve ilaçlama, yangın tespit ve söndürme çalışmaları, meteorolojik veri toplama, kargo taşıma, sivil fotoğrafçılık örnek gösterilebilir. Ancak İHA'lara en büyük yatırımlar hala askeri amaçlar için yapılmaktadır [1].

İnsansız hava araçları bir kullanıcı tarafından uzaktan kontrol edilebilirler veya kullanıcıdan bağımsız olarak otonom uçuşlar gerçekleştirebilirler. Bazı faaliyetleri kullanıcı denetiminde olan, yarı otonom İHA sistemleri de bulunmaktadır. İnsan faktörünü ve kısıtlarını ortadan kaldıran, sert koşullarda çalışabilen insansız hava araçları günümüzde popülerliği giderek artmaktadır [2].

İnsansız hava araçlarının tasarım sürelerini ve maliyetlerini azaltmak için hazır ürünler de bulunmaktadır. Hazır İHA ürünü satan şirketlere örnek olarak MicroPilot gösterilebilir. MicroPilot kullanıcılara hazır otopilot kartları, çoklu otopilotlar ve hazır sensör kartları (haberleşme ve analog dijital dönüştürücüler) sağlamaktadır. Bu şirketler İHA tasarım süresini ve maliyeti düşürebilirler ancak kullanıcıya özel gereksinimleri tam olarak karşılamamaktadırlar [3].

Dört rotorlu İHA'lar dikey iniş ve kalkış özelliğine sahiptirler. Quadrotor sistemi giriş olarak yalpa (roll), yunuslama (pitch), sapma (yaw), pervane kuvveti (throttle) alır ve altı çıkış üretir. İHA'ların diğer pilotlu hava araçlarına göre olan avantajı ise uçuş sürelerinin doğrudan güç kaynağı kapasitesine bağlı olmasıdır. Pilotlu hava araçlarında ise bir insanın günde ne kadar aktif olarak pilotluk yapabileceği de güç kaynağı kadar etkilidir. Bu sebeple İHA'lar insan enerjisinin dayanamayacağı uzun süreli çalışmalarda tercih sebebi olmaktadır [4].



Resim 1.1 : İnsansız hava araçlarının sınıflandırılması [4]

1.1 Tezin Amacı

FPGA teknolojisinin gelişmesi ve yaygınlaşması ile mikroişlemci temelli sistemlere yeni bir alternatif oluşmuştur. FPGA temelli sistemler yüksek hız, düşük güç tüketimi ve değiştirilebilir donanım imkânı sunmaktadır. Aynı zamanda mikroişlemcilerden farklı olarak gerçek zamanlı paralel mimariler FPGA'ler üzerinde gerçeklenebilmektedir [5]. Bu sebepler ile FPGA'lerin havacılık elektroniğinde popülerliği artmaktadır (Resim 1.2). Bu tez çalışmasında bir dört rotorlu bir insansız hava aracının matematiksel modellemesi yapılacaktır. Matematiksel irtifa modeli Simulink ortamında gerçeklenecektir. Model üzerine irtifa kontrolcüsü yapılacak ve referans takibi gerçekleştirilecek benzetimler ile doğrulanacaktır. Katsayılar ile Simulink üzerinde gerçeklenen model çıkarılarak yerine "PID Tuning" bloğu kullanılacak ve referans takibi benzetimler ile doğrulanacaktır. FPGA tasarımı bölümünde ise farklı yöntem ve yazılımlar kullanılarak irtifa kontrolcü oluşturulacaktır. İlk tasarım yöntemi olarak Simulink üzerinde doğrulanan model System Generator üzerinde FPGA tasarım blokları ile tasarlanacak ve gerçekleştirilecek benzetimler ile irtifa referansı takibi doğrulanacaktır. İkinci yöntemde HDL coder yardımı ile Matlab üzerinde yazılan PID irtifa kontrolcü sentezlenebilir HDL'e dönüştürülecektir. Üçüncü yöntemde high level synthesis "HLS" kullanılarak irtifa kontrolcü oluşturulacak ve Vivado HLS üzerinde referans takibi test edilecektir. Dördüncü yöntemde, softcore bir işlemci olan "MicroBlaze" üzerinde irtifa kontrolcü çalıştırılacak ve referans takibi döngüde donanım "HIL" ile doğrulanacaktır. Beşinci tasarım yönteminde ise VHDL ile hazırlanan PID irtifa

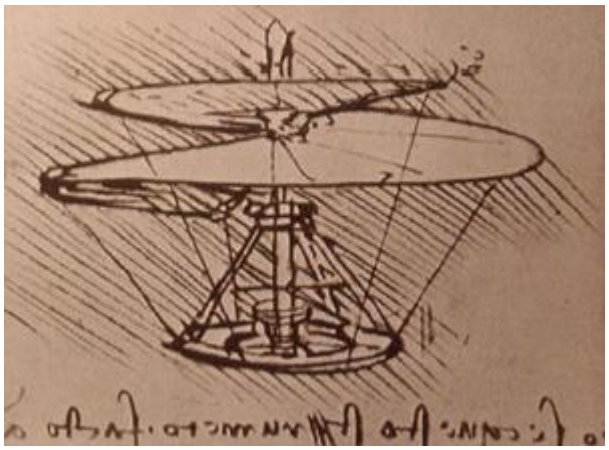
kontrolcü Zynq ile entegre edilecek ve referans takibi doğrulanacaktır. Son bölümde ise tasarımda kullanılan farklı yöntemler referans takibi, kaynak kullanımı ve güç tüketimi açılarından karşılaştırılacaktır.

Resim 1.2 : Havacılıkta kullanılabilir, sertifikalı FPGA'ler (Intelligent Aerospace Staff)

1.2 İnsansız Hava Araçlarının Tarihçesi

İnsansız hava araçları (İHA) günümüzde hala gelişmekte olan teknolojiler olmasına rağmen, İHA'ların varlığı daha eskiye dayanmaktadır. Ancak İHA'ların özellikle de keşif amaçlı kullanılan dronların esas gelişimi soğuk savaş yıllarında olmuştur. İHA'lar en yaygın olarak askeri ve gözetleme amaçları ile kullanılmaktadır [6].

Leonardo Da Vinci, 1452- 1519 yıllarında yaşamış ve kuşların uçuşlarından yola çıkarak uçuş manevralarını içeren ilk dokümanı ortaya koymuştur. Aynı zamanda Da Vinci şu anda Hiller Müzesinde bulunan dikey iniş ve kalkış yapabilen hava aracını geliştirmiştir (Resim 1.3) [7].



Resim 1.3 : Leonardo Da Vinci'nin helikopter tasarımı [7]

1917 yılında, Amerika Birleşik Devletleri donanması uzun menzilli bir silah olan ve aynı zamanda bir İHA olan "Hava Torpidosu"'nu geliştirmiştir. Bu bomba pilotsuz şekilde uzun menzil gidebilmektedir ve tahtadan yapılmıştır. Ağırlığı yaklaşık 270 kilo olmaktadır ve Ford tarafından geliştirilen 40 beygir gücündeki bir motor ile hareket etmektedir. Motor devri, gücü kesmekte ve bombanın hedefi vurmasını sağlamakta kullanılmıştır (Resim 1.4) [8].

Resim 1.4 : Amerika Birleşik Devletleri tarafından geliştirilen "Hava Torpidosu" [8]

1918 yılında, Amerika Birleşik Devletleri ordusu “Kettering Bug” isminde 82 kiloluk bomba taşıyabilen İHA’yı geliştirmiştir (Resim 1.5). Tasarım Charles F. Kettering tarafından yapılmıştır. Güdümlü füzelerin atası olarak kabul edilmektedir. Kettering Bug saatte 80 kilometre gidebilmekte ve 60 kilometre menzildeki bir hedefi vurabilmektedir. Ancak Kettering Bug testlerde yüksek başarı sağlayamamıştır. Gerçekleştirilen 36 test denemesinin sadece sekizi başarılı olmuştur. Verilere göre stabilitesinin iyileştirilmesi ve kullanılan motor gücünün artırılması gerektiği ortaya çıkmıştır. 1. Dünya Savaşında tasarlanan İHA projelerinde ortaya çıkan başlıca problemler; aracı havaya fırlatmada karşılaşılan sorunlar, pilotsuz stabiliteyi yeterince sağlayamama, aerodinamik bilginin yeterli olmaması, İHA’larda kullanılan malzemelerin dayanıksız olması ve atış sonrası hatayı giderebilecek uçuş verilerinin toplanamamasıdır [8][9].

Resim 1.5 : Kettering Bug [8]

1. Dünya Savaşından sonra hava taşımacılığının yaygınlaşmasıyla İHA’ların gelişimi hız kazanmıştır. Radyo sinyalleri ile araçların kontrol edilmesi yaygınlaşmaya başlamıştır. Testlerde başarısızlığa uğrayan ve savaş bittikten sonra geliştirme çalışmaları bırakılan Bug üzerine radyo kontrolcü eklenerek tekrar test çalışmalarına başlanmıştır. 1923 yılında radyo sinyalleri ile F-5L uçağının 10 mill alan içerisinde güvenli uçuşu test edilmiştir (Resim 1.6). 1924 yılına kadar süren testlerde uçağın

kalkışı, manevraları ve inişi radyo sinyalleri ile uzaktan başarılı şekilde kontrol edilmiştir. [12]

Resim 1.6 : Curtiss F-5L, radyo sinyalleri ile uçuşu

Uçakların ve İHA'ların gelişmesi donanmaların önemini azaltmıştır (Resim 1.7). Televizyon ekipmanlarının da İHA teknolojisi ile birleşmesiyle suikast dronları ve güdümlü füzelerin gelişmesi hız kazanmıştır. Amerika Birleşik Devletleri'nde doğrudan radyo sinyalleri ile kontrol edilen İHA'ları geliştirmek için "Radioplane" adında bir şirket kurulmuştur. Bu şirket 2. Dünya Savaşı yıllarında 3000 üzerinde suikast dronu üretmiştir ve bu şirket daha sonra uzay – savunma alanlarında faaliyet gösteren "Northrop Grumman" tarafından satın alınmıştır [10].

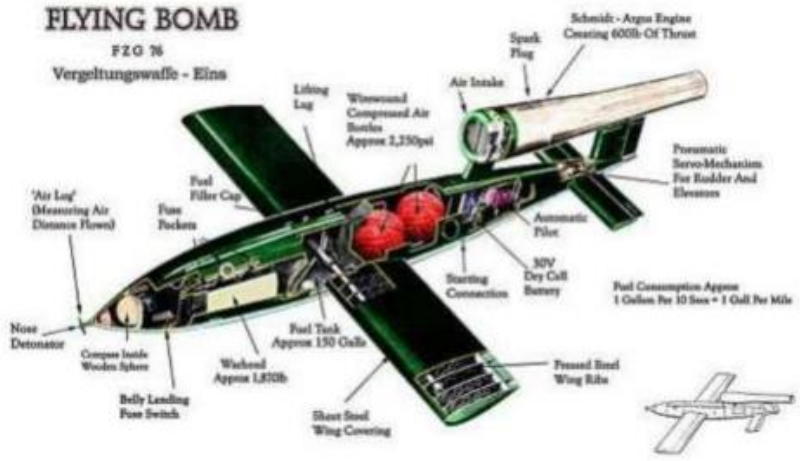
Resim 1.7 : USS Alabama'nın hava saldırısı ile batırılması [10]

2.Dünya Savaşında İHA'ları daha güçlü kılmak adına uçaklara silahların entegre edilmesi hızlanmıştır. İHA pilotları küçük bir monitör aracılığı ile uçağı sürerken, gelen görüntü verileri ile hedeflere torpido atabilmişlerdir. Avrupada ise Almanya'nın geliştirdiği V-1 ve V-2 roketleri İHA örnekleridir ve Londra'da ciddi tahribata yol açmıştır (Resim 1.8). [11].

Resim 1.8 : Alman V1 roketi, Imperial War Müzesi [12]

Alman V-1 ve V-2 roketleri 2. Dünya Savaşında kullanılmıştır (Resim 1.9). İsim gereği roket veya bomba olarak adlandırılmalarına rağmen İHA olarak da adlandırılabilirler. Uzunluğu yaklaşık 8 metre ve ağırlığı 850 kilogram olan bu İHA'lar günümüzde güdümlü füzelere eşittirler. Fransız sahillerinden Londrayı

vurmak için kullanılmıştırlar. Yapılan atışlarda İngiltere minimum hasar aldığını açıklamıştır ancak bunun daha sonradan yanlış bir açıklama olduğu ortaya çıkmıştır. 244 roket Fransa kıyılarında İngiltere'ye atılmıştır ve bu roketlerden 155'i İngiltere'ye ulaşmıştır. 73 roket ise Londra'ya düşmüştür. Londra'ya yapılan saldırıdan sonra müttefikler tarafından yapılan bombalamada roket fırlatma rampaları yokedilmiştir. V-1 roketlerinin ilk tasarımları 1930 yıllarında yapılmış ve hedef isabet oranlarının düşük olması sebebiyle çalışmalara gerekli yatırım daha sonra yapılmamıştır. Ancak tasarımcılar projeyi gizlice ilerletmişlerdir. V-1 roketlerinde "Kettering Bug"'da olduğu gibi motor yakıtı bittiği anda aşağı düşerek bombanın patlaması prensibi kullanılmıştır. Ancak bazı V-1 roketlerinin Almanlar tarafından radyo sinyalleri ile yönlendirildiği düşünülmektedir. [12]



Resim 1.9 : Alman V1 roketi içyapısı [12]

Vietnam savaşında Amerika Birleşik Devletleri İHA'ları yoğun olarak kullanmıştır. 1965-1975 yılları arasında 3000'den fazla İHA operasyonu bölgede düzenlenmiştir. En çok kullanılan İHA "Lightning Bug" olmuştur. DC-130 uçağı ile görev yerlerine salınan bu İHA'lar düşman bölgelerde uzaktan kontrol edilmiştir ve bilgi toplamada kullanılmıştır. Görev sonrası H-53 helikopterleri ile havadan asılı olarak alınmaktadırlar. Lightning Bug İHA'ları önceki İHA'lardan farklı olarak jet motorlarına sahiptirler ve supersonic hızlara çıkabilmektedirler. 50.000 feet yükseklikte uçabildikleri gibi alçak seviyelerde de operasyon gerçekleştirebilirler. DC-130 taşıma uçağı içerisine yerleştirilen monitörler aracılığı ile operasyonlar

kontrol edilmiştir. Yüksek çözünürlükte görüntü alabilirler. Dönemin gelişmiş İHA'larından olmasına rağmen Lighting Bug üzerinde bulunan navigasyon sistemleri problem yaratmıştır. Yapılan operasyonların yarısı navigasyon hataları sebebiyle başarısız olmuştur. Uydu tabanlı GPS sensörler gelişene kadar navigasyon problemi devam etmiştir. Operasyonlarda İHA kullanmanın bir diğer problemi ise bozulan ya da yakıtı biten İHA'ların kurtarılamamasıdır. İHA kurtarmada kullanılan MARS operasyonu ile %40 oranda bir başarı elde edilmiştir (Resim 1.10). Ancak yapılan iyileştirmeler ile bu oran %90'lara çıkarılmıştır. Vietnam savaşında İHA'lar sadece keşif amacıyla kullanılmamıştır. Özel donanım ile donatılan İHA'lar radar verilerini toplamak ve gönderilen roketleri yanıltmak amacıyla da görev yapmıştır. Sovyetler tarafından geliştirilen SA-2 radar destekli füze sistemi Vietnama konumlandırılmıştı. CIA tarafından geliştirilen sinyal izi kaydeden donanım İHA'lara yerleştirilmiş ve hava savunma sistemini jamleyecek bilgiler toplanmıştır.[10][13]

Resim 1.10 : MARS ile kurtarılan "Lightning Bug" [10]

1.3 İnsansız Hava Araçlarının Sınıflandırılması

İHA'lar sahip oldukları farklı özellliklere göre sınıflara ayrılabilir. Sınıflandırmada kullanılabilecek başlıca özellikler; ağırlık, dayanıklılık, menzil, uçuş irtifası, kanat yükü ve motor tipidir.

1.3.1 Ağırlığa Göre Sınıflandırma

İHA'lar kullanım alanlarına bağlı olarak birkaç gram ağırlığında olabildikleri gibi tonlarca ağırlıkta da olabilirler. Ancak İHA'lar genellikle görevleri sebebiyle hafif yapılı tasarımlardır. Başlıca İHA ağırlık sınıfları; mikro, hafif, orta, ağır ve aşırı ağırdır (Çizelge 1.1).

Çizelge 1.1 : Ağırlığa göre sınıflandırma [14]

Sınıf	Ağırlık Aralığı	Örnek
Mikro	< 5 kg	Dragon Eye
Hafif	5- 50 kg	RPO Midget
Orta	50 – 200 kg	NASA SIERRA
Ağır	200 – 2000 kg	A-160
Aşırı Ağır	>2000 kg	Darkstar

Farklı ağırlık sınıflarında İHA örnekleri:

- Darkstar ve Global Hawk 2 ton üzeri ağırlığa sahiptir ve aşırı ağır İHA sınıfına girerler [14].
- Predator ise boş ağırlıkta ~500 kg olarak ağır sınıf İHA'lar arasında yer almaktadır [14].
- NASA SIERRA, orta ağırlıklı İHA sınıfına girmektedir. Yaklaşık 50 kg ağırlıktadır [15].
- Dragon eye yaklaşık 5.9 pound ağırlıktadır. Gelişen teknoloji ile beraber küçük minyatür MEMS sensör ve küçük GPS alıcı içerir. Sahip olduğu ağırlık sebebiyle taşımaya uygundur (Resim 1.11)[16].

Resim 1.11 : Mikro ağırlık sınıfında yer alan Dragon Eye [16]

1.3.2 İrtifaya Göre Sınıflandırma

İHA'lar kullanım amaçlarına göre farklı irtifalarda çalışabilirler (Çizelge 1.2). Askeri uygulamalar tehditi azaltmak ve tespit edilmemek için yüksek irtifalarda uçmaktadır.

Çizelge 1.2 : Uçuş irtifasına göre sınıflandırma [14]

Sınıf	Maksimum İrtifa	Örnek
Alçak	< 1000 m	Pointer
Orta	1000 – 10000 m	Finder
Yüksek	> 10000 m	Darkstar

Farklı irtifa seviyelerinde çalışan İHA örnekleri [14]:

- 1000 m'den alçak irtifalarda görev yapan İHA'lar alçak irtifa sınıfı İHA'lara girmektedir. Dragon Eye ve Pointer alçak irtifa İHA'larına örnektir.
- 1000 m – 10000 m arası irtifalarda görev yapan İHA'lar orta irtifa sınıfı İHA'lara girmektedir.
- 10000 m üzeri irtifalarda görev yapan İHA'lar yüksek irtifa sınıfına girmektedir. Darkstar, Global Hawk ve Predator yüksek irtifa sınıfına girmektedir. Askeri amaçla kullanılan İHA'lar bu sınıfa girer (Resim 1.12).

Resim 1.12 : Yüksek irtifa sınıfında yer alan Darkstar [17]

1.3.3 Kanat Yüküne Göre Sınıflandırma

Göreve bağlı olarak İHA'lar kanatlarında sensör veya silah gibi yükler bulundurabilirler. Kanat yükü İHA'nın ağırlığının kanat alanına bölünmesi ile hesaplanmaktadır. İHA'lar kanat yüklerine göre 3 sınıfa ayrılırlar [14] (Çizelge 1.3).

Çizelge 1.3 : Kanat yüküne göre sınıflandırma [14]

Sınıf	Yük: kg/m^2	Örnek
Hafif	< 50	Seeker
Orta	50 – 100	X-45
Ağır	> 100	Global Hawk

- Kanat yükü 50 kg/m^2'den az olan İHA'lar hafif kanat yükü ağırlıklı İHA sınıfına girmektedir. Bu sınıfa örnek olarak Dragon Eye gösterilebilir.
- Kanat yükü 50 kg/m^2 ile 100 kg/m^2 arasında olan İHA'lar orta kanat yükü ağırlıklı İHA sınıfına girmektedir.

- Kanat yükü 100 kg/m^2'den fazla olan İHA'lar ağır kanat yükü ağırlıklı İHA sınıfına girmektedir. Bu sınıfa örnek olarak yüksek teknolojiye sahip bir İHA olan X-50 ve Global Hawk gösterilebilir (Resim 1.13).

Resim 1.13 : Yüksek irtifa ve yüksek dayanıklı Global Hawk [18]

1.3.4 Motor Tipine Göre Sınıflandırma

İHA'lar farklı görevlerde kullanılabildikleri için farklı tasarımlara sahiptirler. Bu sebeple İHA'larda kullanılan motor tipleri çeşitlilik gösterir (Çizelge 1.4). İHA'larda gövde ağırlığı arttıkça kullanılan motorun ağırlığı da artmaktadır. Hafif İHA'larda küçük yapılı elektrik motorları yeterli olurken, ağır gövdeli askeri amaçlı İHA'larda daha fazla itki sağlamaları sebebiyle pistonlu yapıya sahip motorlar kullanılmaktadır.

Çizelge 1.4 : Motor tipine göre sınıflandırma [14]

Turbofan	Piston	Elektrik	Pervane	Turbo Pervane	İtme-Çekme
Global Hawk	Predator	Dragon Eye	LEWK	Predator B	Hunter
Darkstar		Raven			

1.3.5 Görev ve Yeteneklere Göre Sınıflandırma

İHA'lar görev ve yeteneklerine göre sınıflandırılabilirler. Bu yetenek ve görevler sırasıyla:

- Keşif amaçlı İHA'lar
- Savaş İHA'ları
- Dikey iniş-kalkış yapabilen İHA'lar
- Kargo İHA'ları (Resim 1.14)
- Çok amaçlı kullanılabilir İHA'lar

Kargo İHA'ları özellikle çevrim içi satış yapan firmalar tarafından kullanılmaktadır. Amazon altı yıl içerisinde kargoların büyük bölümünü İHA'lar ile yapmayı planlamaktadır. Kargo İHA'ları Avustralya, Afrika ve Avrupada aktif olarak kullanılmaktadır. En büyük merak konusu ve tehlike ise mini dronların insan taşıyan uçaklara oluşturacağı negatif etkidir. DJI hava güvenliğini sağlamak için geliştireceği dronlar ile uçaklar arasında bir haberleşmenin olmasını planlanmaktadır. Böylece dronlar yakınlarında olan uçağı algılayacak ve uçağa tehlike oluşturmayacak şekilde hareket edecektir. [19]

Resim 1.14 : DJI kargo dronu [19]

Ticari amaçla en çok kullanılan dronlar dikey iniş ve kalkış yapabilen VTOL dronlardır. En büyük dezavantajları uçuş sürelerinin kısa olmasıdır. Sabit kanatlı alternatif İHA'lar daha uzun süreler havada kalabilirler. Ancak sabit kanatlı İHA'ların yere inişleri VTOL İHA'lara göre daha zordur.[20]

1.3.6 Türkiye'de İnsansız Hava Araçları

Dünya'da olduğu gibi Türkiye'de de yerli imkânlar ile İHA tasarımları yaygınlaşmaktadır. Farklı şirketler kendi İHA'larını geliştirmektedir. En çok askeri amaçlı İHA tasarımlarına yatırım yapılmaktadır.

STM'nin geliştirdiği akıllı mühimmat "KARGU" (Resim 1.15)[21];

- Döner kanatlı bir İHA türüdür.
- Gece ve gündüz çalışabilir.
- Görüntü işleme ile kontrol edilebilir.
- Hareketli hedefleri vurabilir.

Resim 1.15 : KARGU, döner kanatlı vurucu İHA [21]

STM'nin geliştirdiği keşif dronu "TOGAN" (Resim 1.16) [21];

- Döner kanatlı bir İHA türüdür.
- Gece ve gündüz çalışabilir.
- Otonom nesne takip yeteneği

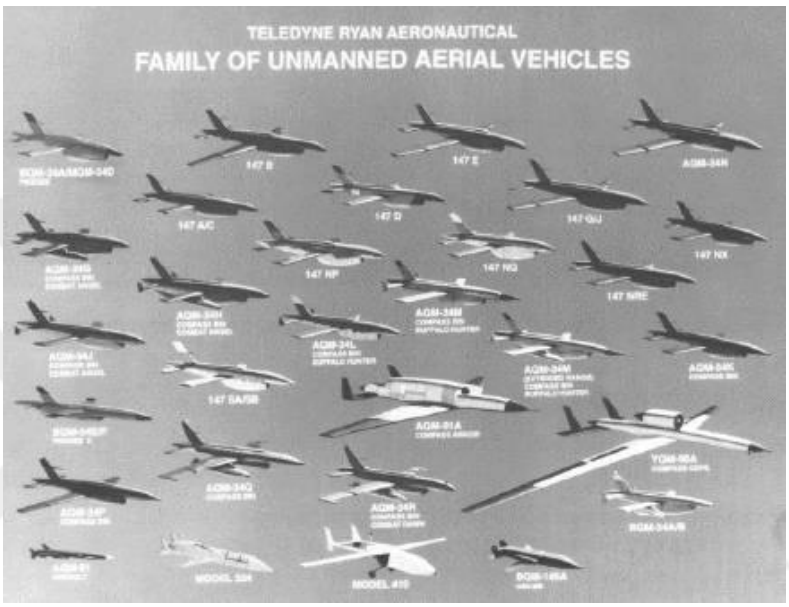
Resim 1.16 : TOGAN, gözcü İHA [21]

1.4 İnsansız Hava Araçlarının Avantajları

Soğuk savaş yıllarında ortaya çıkan gözetleme ve casusluk ihtiyacı dronların gelişmesini hızlandırmıştır. Özellikle insanlı uçuşlarda olabilecek kazalar gizli görevleri tehlikeye atmaktadır. Amerika'nın yaşadığı bir örnekte bir pilot tarafından kontrol edilen U-2 uçağı elde ettiği kayıt verileri ile Sovyetler birliği tarafından düşürülmüştür. Uçak düşmesine rağmen veriler ele geçirilmiştir ve politik anlamda Amerika Birleşik Devletleri'ne problem yaratmıştır. Bu gibi politik sorunları engellemesi sebebiyle dronlar önem kazanmıştır. [22]

Gelişen İHA teknolojisi ile ülkeler askeri görevleri minimum personel kaybı ile gerçekleştirmektedir. İHA'lar tehlikeli bölgelerde uçuş yaparken onları kontrol eden pilotlar yüzlerce kilometre uzakta güvenli bölgelerde olabilirler.[23]

İHA’lar otonom olarak insansız görev yapabilirler. Bu özellikleri sayesinde insana bağlı olan sınırlılıkları yoktur. Örneğin “Defense Advanced Research Projects Agency” ses hızından 20 kat hızda hareket eden İHA geliştirmiştir. Bu sayede İHA hızlı yol alabilirken daha az yakıt yakmaktadır. Hipersonik hızda hareket eden İHA’lar hareket halindeki hedefleri Dünya’nın farklı bir yerinde de olsa kolaylıkla imha edebilirler [23].



Resim 1.17 : Ryan Aeronautical tarafından geliştirilmiş olan İHA’lar [22]

Tam otonom olan İHA’lar insan bir pilota bağımlı olmadıkları için daha uzun süreler görevlerini yapabilirler. Görev yapma süreleri yakıt – şarj sürelerine bağlıdır.

1.5 İnsansız Hava Araçlarının Dezavantajları

İHA’ların yaygınlaşması ve toplum içinde kullanımının artması özel hayat ve veri güvenliğini tehdit etmektedir. Dronlar üzerlerinde kamera ve ses kaydedici gibi kayıt sistemleri bulundurabilirler. Gelişmiş algoritmalara sahip dronlar insan yüz tespiti dahi yapabilirler. İHA’lar kazalara ve yaralanmalara sebep olabilirler ancak henüz İHA’lar ile ilgili bir sigorta bulunmamaktadır. Olası kaza durumunda İHA kullanıcısının bulunması da başka bir problemdir. Sivil amaçla kullanılan İHA’lar eğer şifresiz kanallar kullanıyorsa radyo alıcıları, WIFI veya GPS üzerinden siber saldırıya uğrayabilirler. Askeri alanda şifreli mesajlar kullanıldığı için askeri İHA’ların siber saldırı zafiyetleri daha düşüktür. Bunlara ek olarak askeri alanda

İHA'ların yaygınlaşması sınır ötesi operasyonları kolaylaştırmakta ve savaş çıkma ihtimalini artırmaktadır [24].

1.6 Literatür Taraması

Quadrotorlar üzerine şirketler tarafından yapılan çalışmalar olduğu gibi üniversiteler ve araştırma enstitüleri tarafından da yapılan çalışmalar vardır.
2006 yılında, Ohio State Universitesi'nden Rongh Xu ve Ümit Özgüner'in yaptığı çalışmada kayan kipli kontrolcü ile quadrotoru (x,y,z) eksenlerinde robust ve stabil şekilde kontrol edildiği gözlenmiştir [25].

2007 yılında, İsviçre Federal Teknoloji Enstitüsü'nde yapılan çalışmada Samir Bouabdallah ve Roland Siegwart simülasyon modeli üzerinde integral geri adımlama yöntemi ile quadrotorun irtifa ve pozisyon kontrolü gösterilmiştir [26].

2012 yılında, Zhao Bo ve Xian Bin tarafından yapılan çalışmada TI-TMS320F dijital sinyal işlemcisi kullanılarak PD kontrolcüsü ile minyatür yapıda olan bir quadrotorun kontrolü sağlanmıştır [27].

2013 yılında, New South Wales Universitesi'nde yapılan çalışmada Christopher Lehner ve Peter Corke robot işletim sistemine (ROS) sahip mikro quadrotor tasarımını anlatmışlardır (Resim 1.18). Quadrotor kontrol işlemlerini 8-bitlik Atmega işlemci üzerinde gerçekleştirmişlerdir. Ayrıca daha gelişmiş işlemcilerden biri olan ARM cortex A-8 ile de kamera modülü ile haberleşme yapmışlardır [28].

Resim 1.18 : New South Wales Universitesinde tasarlanan mikro quadrotor[28]

2017 yılında, Ege Üniversitesi'nde Yusuf Atalay ve Aydoğan Savran tarafından yapılan çalışmada LQR ve PID kontrolcü yöntemleri ile Matlab/Simulink üzerinde benzetimi yapılmıştır. Daha sonra Ardupilot kontrolcü kartı ile İHA üzerinde testler gerçekleştirilmiştir [29].

2017 yılında, TOBB Ekonomi ve Teknoloji Üniversitesi'nde Ceren Cömert ve Coşku Kasnakoğlu tarafından yapılan çalışmada quadrotor kontrolü için kayan kipli kontrolcü ile PID kontrolcü karşılaştırılmıştır. İki kontrolcünün de referans değerlerini istenildiği gibi takip ettiği görülmüştür. Kayan kipli kontrolcü ile çatırtı probleminin önlendiği görülmüştür [30].

2018 yılında, Hacettepe Üniversitesi'nde Ahmet Demiryürek ve Hüseyin Demircioğlu tarafından yapılan çalışmada quadrotor modeli üzerinde PID, geri adımlamalı ve kayan kipli kontrolcülerin performansı incelenmiştir [31].

2019 yılında, Ortadoğu Teknik Üniversitesi'nde Şerafettin Tüzel tarafından yapılan çalışmada PID kontrolcü ile Rasberry Pi üzerinde kontrolcü oluşturulmuştur. Simulink ve Embedded Coder kullanılarak algoritmalar modellerden üretilmiştir [32].

2019 yılında, TOBB Ekonomi ve Teknoloji Üniversitesi'nde Mehmet Karahan ve Coşku Kasnakoğlu tarafından yapılan çalışmada quadrotor modellemesi yapılmıştır. Model Simulink ortamına aktarıldıktan sonra yükseklik, yunuslama, yalpa ve sapma açıları için PID kontrolcüler tasarlanmıştır. Ayrıca kontrolcülerin parametre belirsizliği ve gürültü altında fonksiyonel olduğu gösterilmiştir [33].

2020 yılında, Belgrad Savunma Üniversitesi'nde Taki Lechekhab ve Stojadin Manojlovic tarafından yapılan çalışmada ADRC ile robust quadrotor kontrolü üzerine çalışılmıştır. System Generator kullanılarak kontrolcü FPGA üzerinde gerçeklenmiş ve HIL yöntemi ile test edilmiştir [34].

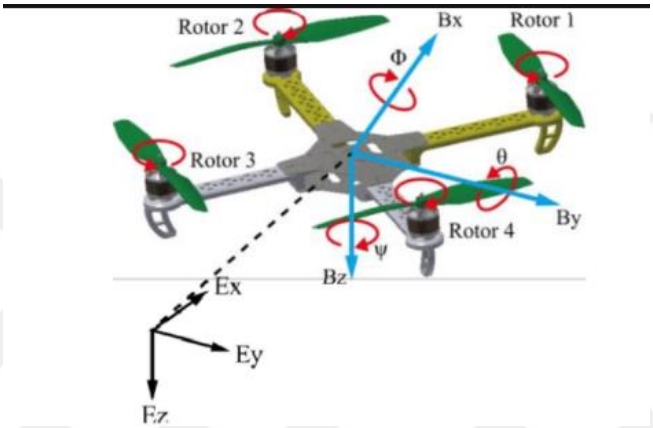
Quadrotor kontrolü üzerine yapılan çalışmalarda ağırlıklı olarak model üzerinde simülasyonlar ile çalışmaların yapıldığı görülmüştür. Gerçeklemelerde ise genellikle mikroişlemciler tercih edilmiştir. FPGA tabanlı uygulamalar yüksek performans verebilmeleri ve üzerlerinde paralel yapılar uygulanabilir olmaları sebebiyle mikroişlemci tabanlı uygulamalara göre daha üstündür. Bu tez çalışmasında PID irtifa kontrolcüsü farklı yöntemler ve yazılımlar ile FPGA üzerinde gerçeklenecektir. Bu yöntemler sırası ile System Generator ile irtifa kontrolcü tasarımı, HDL Coder ile

irtifa kontrolcü tasarımı, HLS ile irtifa kontrolcü tasarımı, sanal işlemci ile irtifa kontrolcü tasarımı ve sayısal irtifa kontrolcü IP tasarımıdır. FPGA üzerinde gerçeklenen sanal işlemci tabanlı irtafa kontrolcü ve irtifa kontrolcü IP'si HIL yöntemi ile doğrulanacaktır. Daha sonra referans takibi performansı, parametre belirsizliği ve gürültü etkisinde incelenecektir. Farklı yaklaşımların FPGA üzerinde alan kaplama, enerji harcama ve karmaşıklık yönünden kıyaslaması yapılacaktır. Sonuç olarak FPGA tabanlı kontrolcü ile mikroişlemci tabanlı kontrolcü uygulamalarından farklı olarak quadrotor kontrolünün paralel ve daha hızlı gerçekleştirilebileceği gösterilecektir.

2. QUADROTOR MATEMATİKSEL MODELİ

2.1 Quadrotora Etki Eden Aerodinamik Kuvvetler

Bu bölümde irtifa kontrolcüsünün Simulink üzerinde simülasyonları yapılmadan ve HDL kodları oluşturulmadan önce quadrotorun matematiksel modeli anlatılmıştır. Quadrotor sahip olduğu dört rotor ile hareket etmektedir. Örnek quadrotorun hareketleri Resim 2.1'de gösterilmiştir.



Resim 2.1 : Quadrotor hareketleri

Quadrotor sisteminin kütlesi m olarak belirtilmektedir ve üzerine etkiyen yer çekimi ivmesi g ile gösterilmiştir.

Quadrotorun poziyonunu (x, y, z) ile gösterilirken, (ϕ, θ, ψ) yunuslama (pitch), yalpa (roll), sapma (yaw) hareketlerini göstermektedir.

Rotor 1 ve rotor 4 hızlı dönerken, rotor 2 ve rotor 3 normal hızda dönerse Quadrotor ileri yönde hareket eder. Rotor 2 ve rotor 3 hızlı dönerken, rotor 1 ve rotor 4 normal hızda dönerse Quadrotor geri yönde hereket eder. (Yunuslama hareketi)

Rotor 1 ve rotor 2 hızlı dönerken, rotor 3 ve rotor 4 normal hızda dönerse Quadrotor sola eğilir. Rotor 3 ve rotor 4 hızlı, rotor 1 ve rotor 2 normal hızda dönerse Quadrotor sağa eğilir. (Yalpa hareketi)

Rotor 2 ve rotor 4 hızlı dönerken, rotor 1 ve rotor 3 normal hızda dönerse Quadrotor sola döner, rotor 1 ve rotor 3 hızlı dönerken, rotor 2 ve rotor 4 normal hızda dönerse Quadrotor sağa döner. (Sapma hareketi)

Dört rotorda düşük hızda dönerse Quadrotor aşağı yönde hareket eder, dört rotor hızlı dönerse Quadrotor yükselir.

Yunuslama, yalpa ve sapma açıları sınırları Eşitlik (2.1-2.2-2.3)'de gösterilmiştir.

$$-\frac{\pi}{2} < \phi < \frac{\pi}{2} \quad (2.1)$$

$$-\frac{\pi}{2} < \theta < \frac{\pi}{2} \quad (2.2)$$

$$-\pi < \psi < \pi \quad (2.3)$$

R dönüşüm matrisi Quadrotorun yönünü gösterir ve Euler açıları ile belirtilir. Euler açıları Eşitlik 2.4'de gösterilmiştir [35].

$$\mathrm{R}(\phi,\theta,\psi) = \begin{bmatrix} \cos(\psi)\cos(\theta) & \sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\theta) & \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\psi)\sin(\phi) \\ \sin(\psi)\cos(\theta) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\psi)\cos(\theta) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \quad (2.4)$$

Bir rotorun ürettiği itme kuvveti Eşitlik 2.5'te gösterilmiştir. Eşitlikte p hava yoğunluğu, r yarıçapı, Λ pervanenin bölümünü ve C_T de aerodinamik itme katsayısıdır [35].

$$\mathrm{F} = \frac{1}{2} p \Lambda C T \mathrm{r}^2 w^2 = b w^2 \quad (2.5)$$

Quadrotorun yunuslama (pitch), yalpa (roll) ve sapma (yaw) torkları Eşitlik 2.6, Eşilik 2.7 ve Eşitlik 2.8'de gösterilmiştir. C orantı kaysayısını, l ise rotorlar ile Quadrotorun merkezi arasındaki uzaklığı göstermektedir [35].

$$\tau_\theta = \mathrm{l}(F_3 - F_1) \quad (2.6)$$

$$\tau_\phi = \mathrm{l}(F_4 - F_2) \quad (2.7)$$

$$\tau_{\psi} = \mathrm{C}(F_1 - F_2 + F_3 - F_4) \tag{2.8}$$

Quadrotor farklı hızlarda çalışabilen dört rotor tarafından kontrol edilmektedir. Bu kontrol girişleri Eşitlik 2.9'da gösterilmiştir [35].

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \mathrm{F} \\ \tau_{\phi} \\ \tau_{\theta} \\ \tau_{\psi} \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ -lb & 0 & lb & 0 \\ d & -d & d & -d \end{bmatrix} \begin{bmatrix} {w_1}^2 \\ {w_2}^2 \\ {w_3}^2 \\ {w_4}^2 \end{bmatrix} \tag{2.9}$$

2.2 Quadrotor Dinamik Sistem Modeli

Newton kanunları ile quadrotor hareket denklemleri Eşitlik 2.10-2.11-2.12'de gösterildiği gibi yazılabilir [35]. Quadrotorun ivmelenmesi Eşitlik 2.11'de gösterildiği gibi yer çekimi ivmesi ve rotorların toplam itmesine bağlıdır.

$$m\ddot{\boldsymbol{\xi}} = F_{th} + F_d + F_g \tag{2.10}$$

$$m\ddot{\boldsymbol{\xi}} = \mathrm{G} + RT_B \tag{2.11}$$

$$\mathrm{J}\dot{\Omega} = M - M_{gp} - M_{gb} - M_a \tag{2.12}$$

Eşitlik 2.10'da yer alan $\mathrm{F_{th}}$, rotorların ürettiği toplam itki kuvvetini belirtmektedir. $\mathrm{F_d}$ quadrotorun hareketine karşı olan hava kuvvetidir. $\mathrm{F_g}$ yer çekimi kuvvetidir. M yunuslama, yalpa ve sapma torklarının toplamıdır. $\mathrm{M_{gp}}$ ve $\mathrm{M_{gb}}$ gyroskopik torklardır. $\mathrm{M_a}$ aerodinamik sürtünmeyi temsil etmektedir.

Konum vektörünü ve kuvvet ifadelerini Eşitlik 2.10-2.12'de yerlerine koyarak Eşitlik 2.13'te gösterilen quadrotorun dinamik modeli elde edilir [1].

$$\ddot{\boldsymbol{x}} = \frac{\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)}{\mathrm{m}} U_1$$

$$\ddot{\boldsymbol{y}} = \frac{\cos(\phi)\sin(\theta)\cos(\psi) - \sin(\phi)\sin(\psi)}{\mathrm{m}} U_1$$

$$\ddot{\boldsymbol{z}} = \frac{\cos(\phi)\cos(\theta)}{\mathrm{m}} U_1 - g \qquad (2.13)$$

$$\ddot{\phi} = \dot{\theta}\dot{\psi}\frac{(I_y - I_Z)}{I_X} - \frac{J_r}{I_x}\dot{\theta}\Omega + \frac{\dot{\mathrm{l}}}{I_x} U_2$$

$$\ddot{\theta} = \dot{\phi}\dot{\psi}\frac{(I_Z - I_X)}{I_Y} + \frac{J_r}{I_Y}\dot{\phi}\Omega + \frac{\dot{\mathrm{l}}}{I_Y} U_3$$

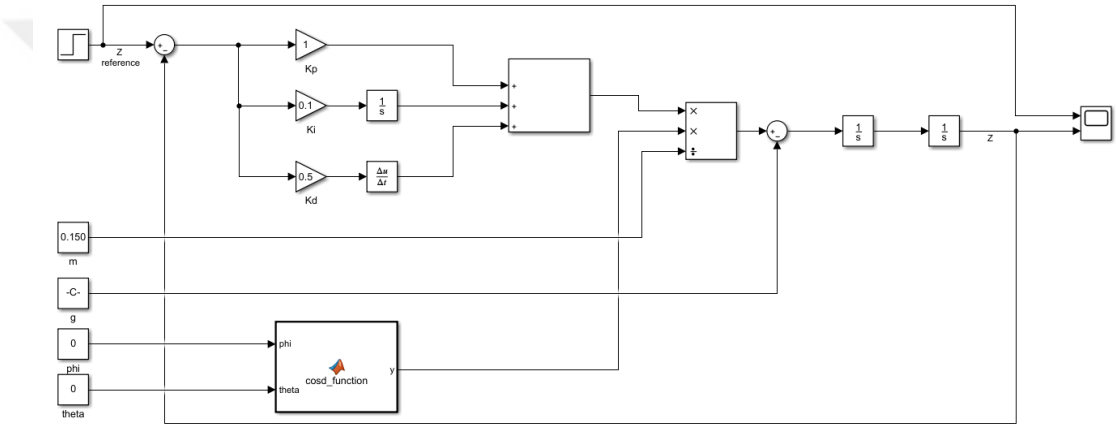
$$\ddot{\psi} = \dot{\phi}\dot{\theta}\frac{(I_X - I_Y)}{I_Z} + \frac{\dot{\mathrm{l}}}{I_Z} U_4$$

3. SİMULİNK İLE QUADROTOR İRTİFA KONTROLCÜ TASARIMI

Bu bölümde PID irtifa kontrolcü kazanç blokları ve PID tuning bloğu kullanılarak Simulink üzerinde oluşturulmuştur. Giriş referansına farklı irtifa değerleri girilerek testler gerçekleştirilmiştir.

3.1.Kazanç Blokları ile İrtifa Kontrolcü Tasarımı

Kazanç blokları ile PID kontrolcü Simulink üzerinde gerçeklenmiştir (Şekil 3.1). Referans olarak birim basamak fonksiyonu yükseklik değeri olarak verilmiş ve referans takibi gösterilmiştir.

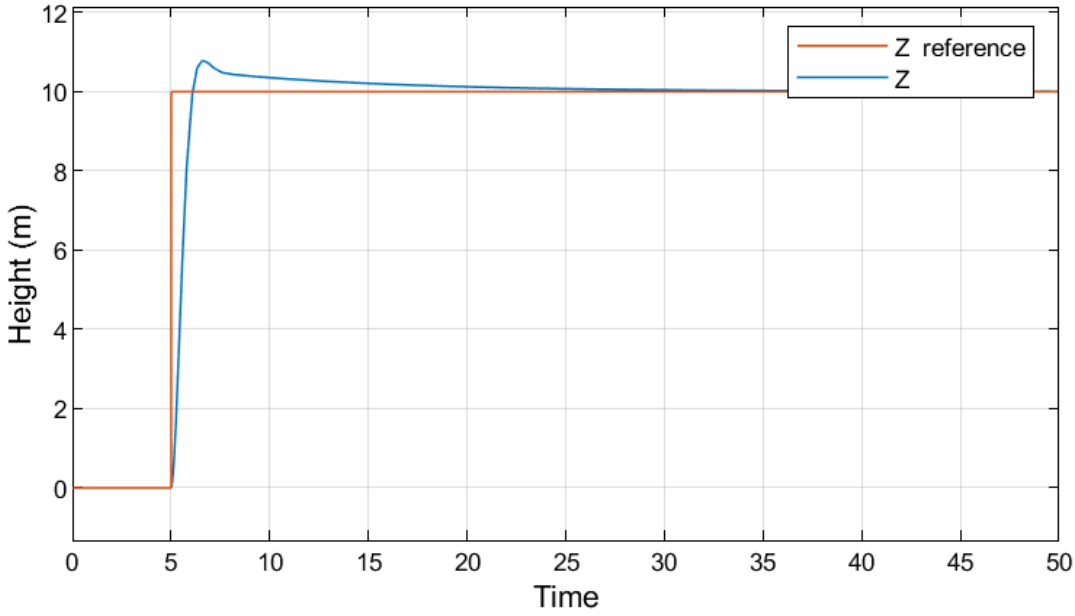


Şekil 3.1 : Kazanç blokları ile irtifa kontrolcü

Çizelge 3.1 : Yükseklik kontrolcü parametreleri

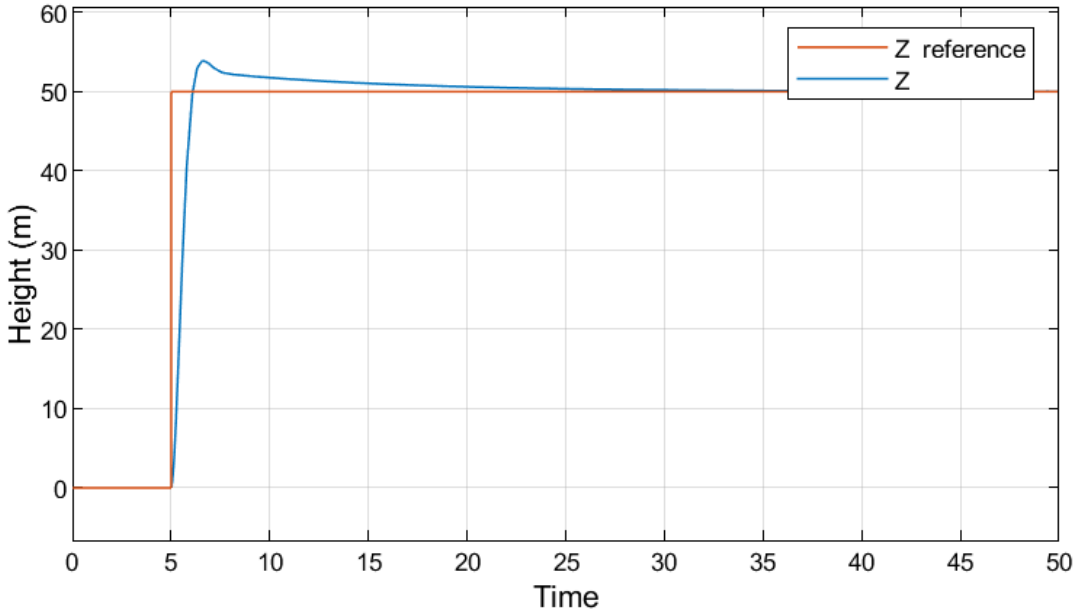
	Kp	Ki	Kd
Değerler	1	0.1	0.5

Kontrolcü kazanç bloklarında Çizelge 3.1'de gösterilen değerler kullanılmıştır. İrtifa kontrolcüsünü test etmek için ilk olarak 10 metrelik referans değeri sisteme verilmiştir. Sistemin referansı takibi Şekil 3.2'de gösterilmiştir. 10 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 3.2 : Referans 10 metre ile irtifa kontrolcü testi

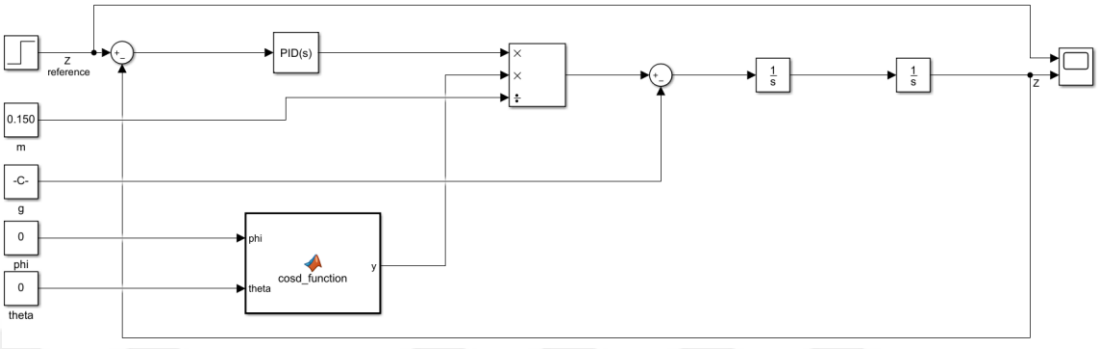
Giriş referansı 50 metreye çıkarılmış ve test tekrarlanmıştır. Referans takibi Şekil 3.3'de gösterilmiştir. 50 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 3.3 : Referans 50 metre ile irtifa kontrolcü testi

3.2.PID Tuner Bloğu ile İrtifa Kontrolcü Tasarımı

Bu bölümde kazanç blokları çıkarılmış ve yerine PID tuner bloğu kullanılmıştır (Şekil 3.4). Referans olarak basamak fonksiyonu yükseklik değeri olarak verilmiş ve referans takibi gösterilmiştir.

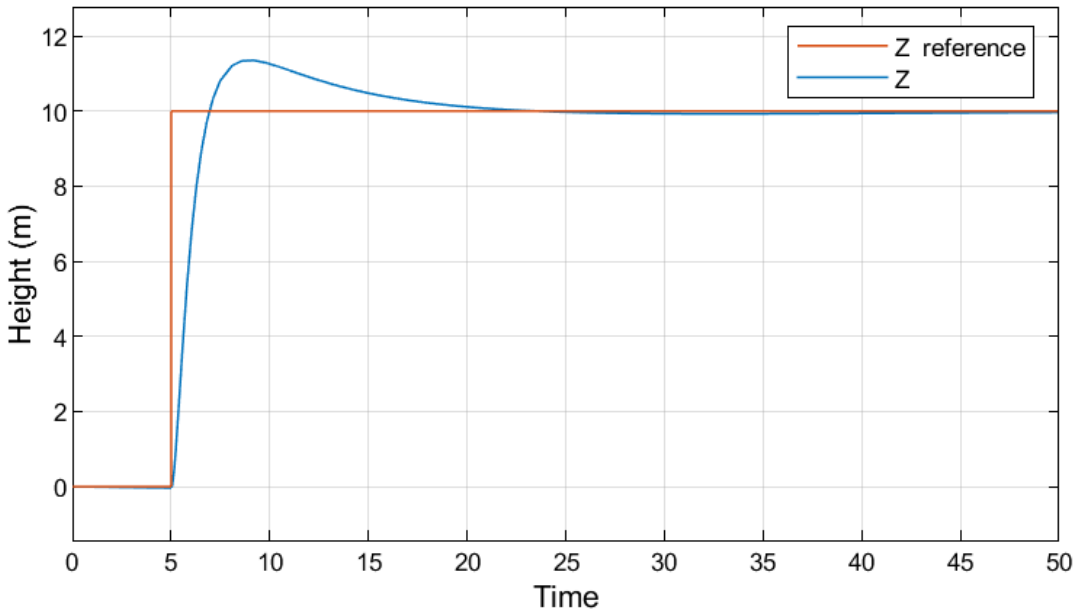


Şekil 3.4 : PID tuner ile irtifa kontrolcü

Çizelge 3.2 : PID tuner parametreleri

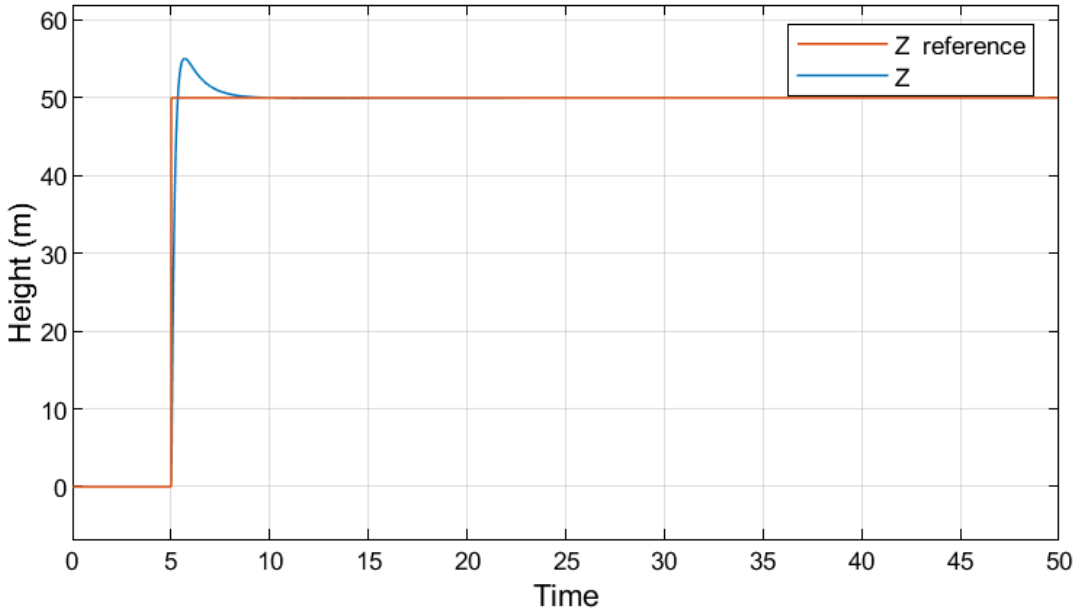
	Kp	Ki	Kd	Filtre Katsayısı
Değerler	1	0.1	1	100

PID tuner'da Çizelge 3.2'de gösterilen değerler kullanılmıştır. İrtifa kontrolcüsünü test etmek için ilk olarak 10 metrelik referans değeri sisteme verilmiştir. Sistemin referans takibi Şekil 3.5'te gösterilmiştir. 10 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 3.5 : Referans 10 metre ile irtifa kontrolcü testi

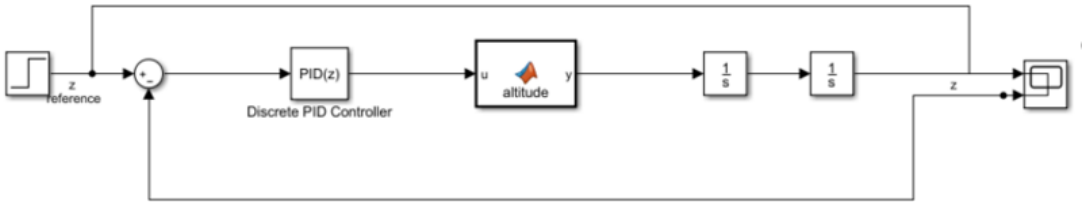
Giriş referansı 50 metreye çıkarılmış ve test tekrarlanmıştır. Referans takibi Şekil 3.6'da gösterilmiştir. 50 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 3.6 : Referans 50 metre ile irtifa kontrolcü testi

3.3.Ayrık PID Bloğu ile İrtifa Kontrolcü Tasarımı

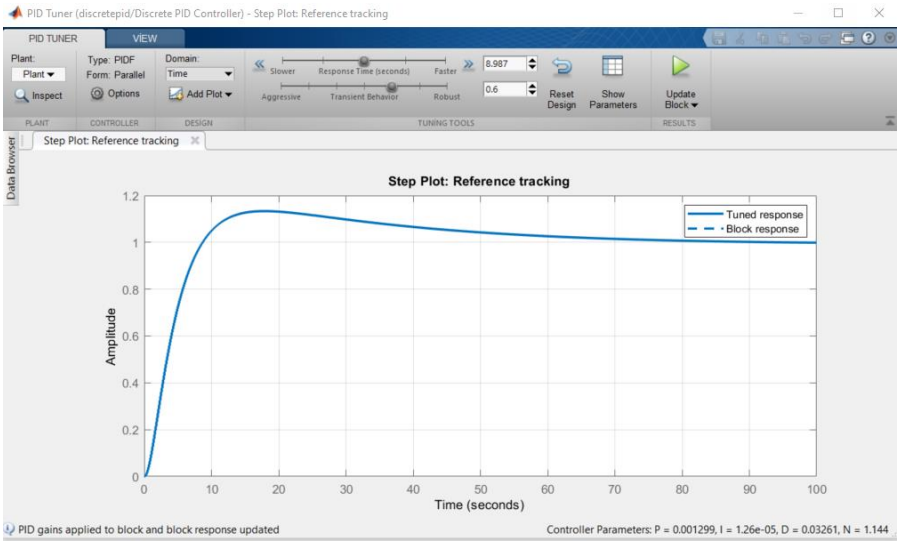
Bu bölümde sürekli zaman PID tuner bloğu yerine ayrık PID bloğu kullanılmıştır ve blok tasarımı Şekil 3.7'de gösterilmiştir. Ayrık PID katsayıları tuner kullanılarak bulunmuştur (Çizelge 3.3). PID tuner ayarları Şekil 3.8'de gösterilmiştir.



Şekil 3.7 : Ayrık PID tuner ile irtifa kontrolcü

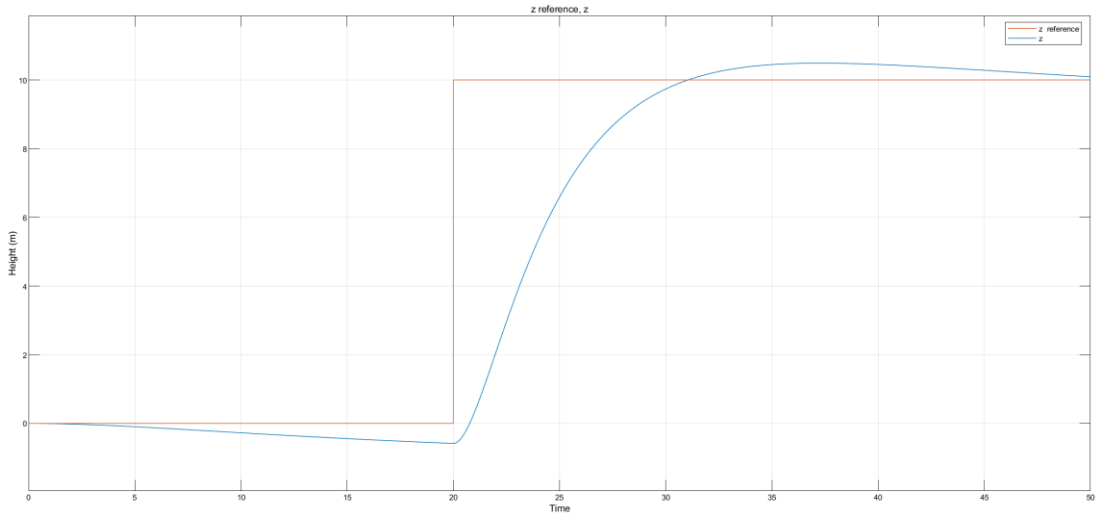
Çizelge 3.3 : PID tuner parametreleri

Parametreler	Değerler
Kp	0.00129892521277267
Ki	1.26047545827616e-05
Kd	0.0326105083323061
Filtre Katsayısı	1.14439525064094



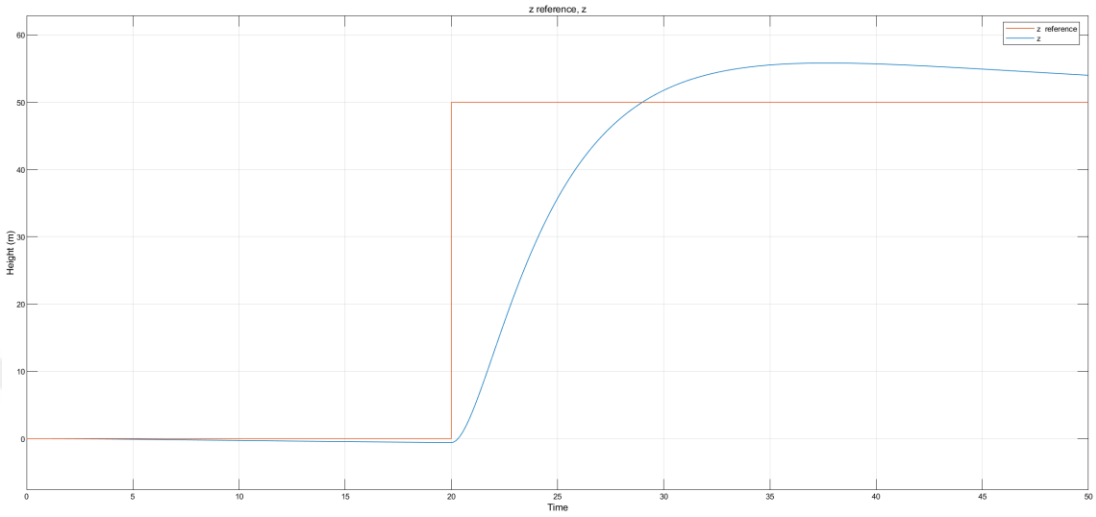
Şekil 3.8 : Ayrık PID için PID Tuner Kullanımı

Ayrık irtifa kontrolcüsünü test etmek için ilk olarak 10 metrelik referans değeri sisteme verilmiştir. Sistemin referansı takibi Şekil 3.9'da gösterilmiştir. 10 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 3.9 : Referans 10 metre ile ayrık PID irtifa kontrolcü testi

Giriş referansı 50 metreye çıkarılmış ve test tekrarlanmıştır. Referans takibi Şekil 3.10'da gösterilmiştir. 50 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



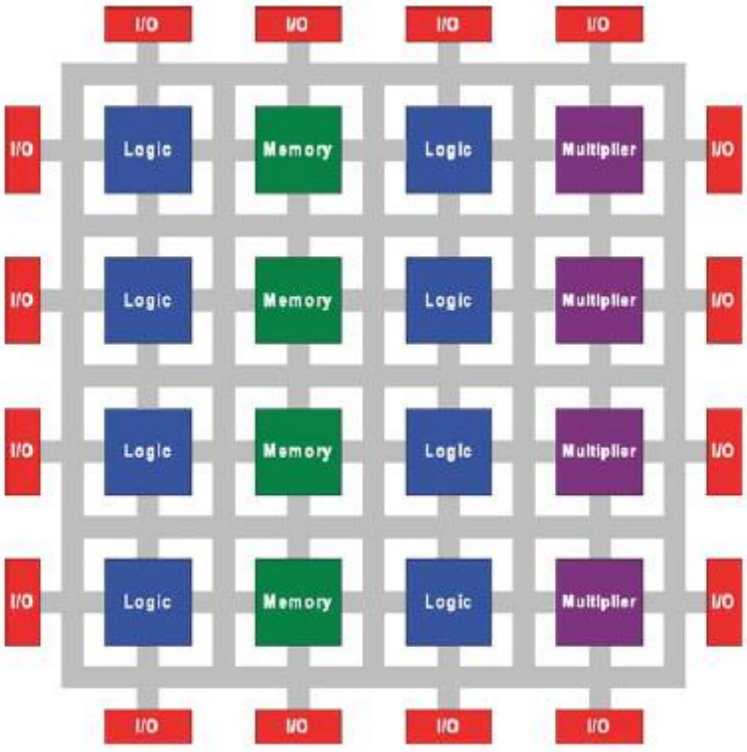
Şekil 3.10 : Referans 50 metre ile ayrık PID irtifa kontrolcü testi

4. FPGA ÜZERİNDE QUADROTOR İRTİFA KONTROLCÜ TASARIMI

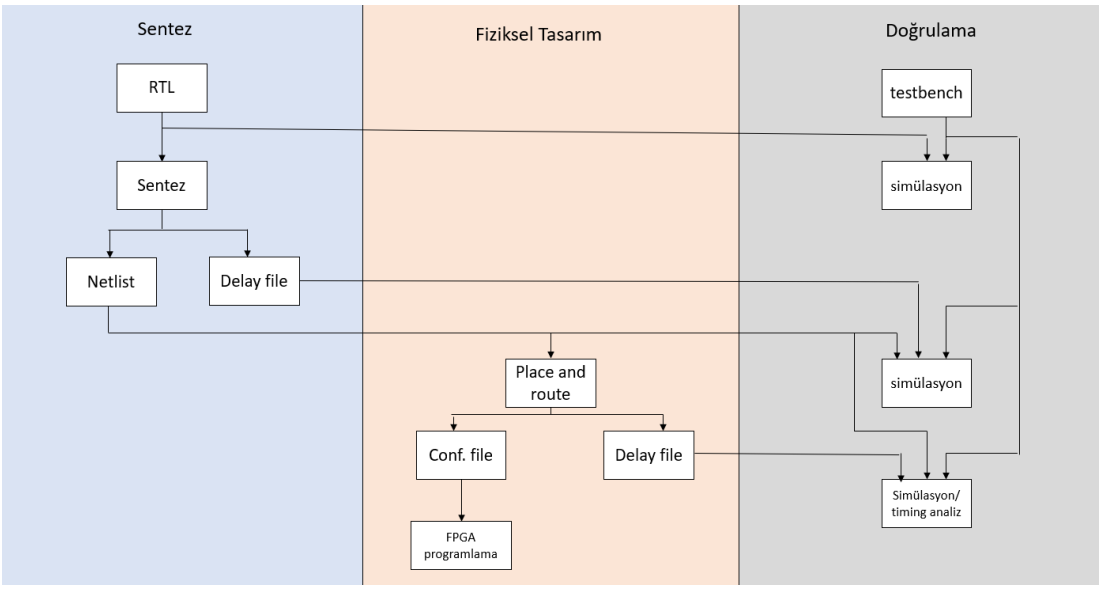
Bu bölümde FPGA tabanlı irtifa kontrolcü beş farklı yaklaşım ile tasarlanmış ve test edilmiştir. İlk olarak FPGA ile ilgili kısa bilgi verilmiştir. Daha sonra FPGA tasarım yöntemlerine geçilmiştir.

4.1.FPGA Hakkında Bilgiler

FPGA ilk olarak Xilinx tarafında 1984'te tanıtılmıştır. FPGA içerisinde pek çok programlanabilir mantık bloğu barındırmaktadır. Bu mantık blokları farklı tiplerdedir. Çarpıcılar, registerlar ve logic hücreler programlanabilir fabric ile çevrilmektedir. Bir logic hücre içerisinde dört ya da daha fazla LUT, çeşitli multiplexer ve bir tane flip flop bulundurmaktadır. Arraylerin etrafında programlanabilir I/O'lar bulunur. Bu I/O'lar fiziksel dünya ile olan arayüzdür. FPGA'ler ASIC'lerden farklı olarak fabrikasyon aşamasından sonra tekrar programlanabilirler. FPGA yapısı Resim 4.1'de gösterilmiştir. Örnek FPGA tasarım akışı Resim 4.2'de gösterilmiştir [36].



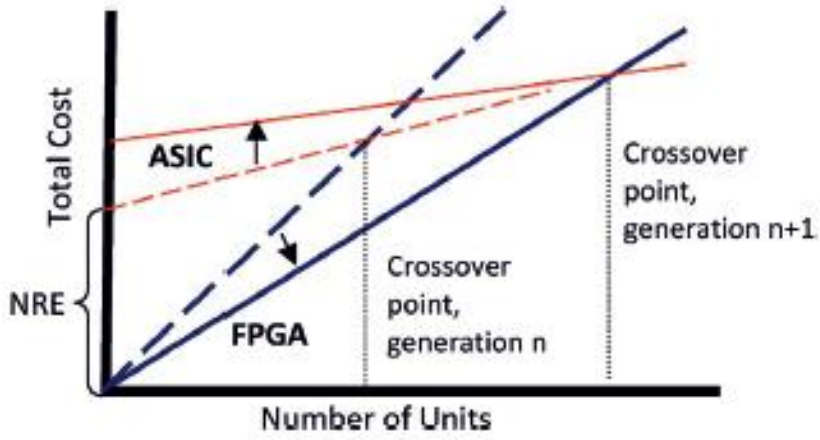
Resim 4.1 : FPGA yapısı [36]



Resim 4.2 : FPGA tasarım akışı

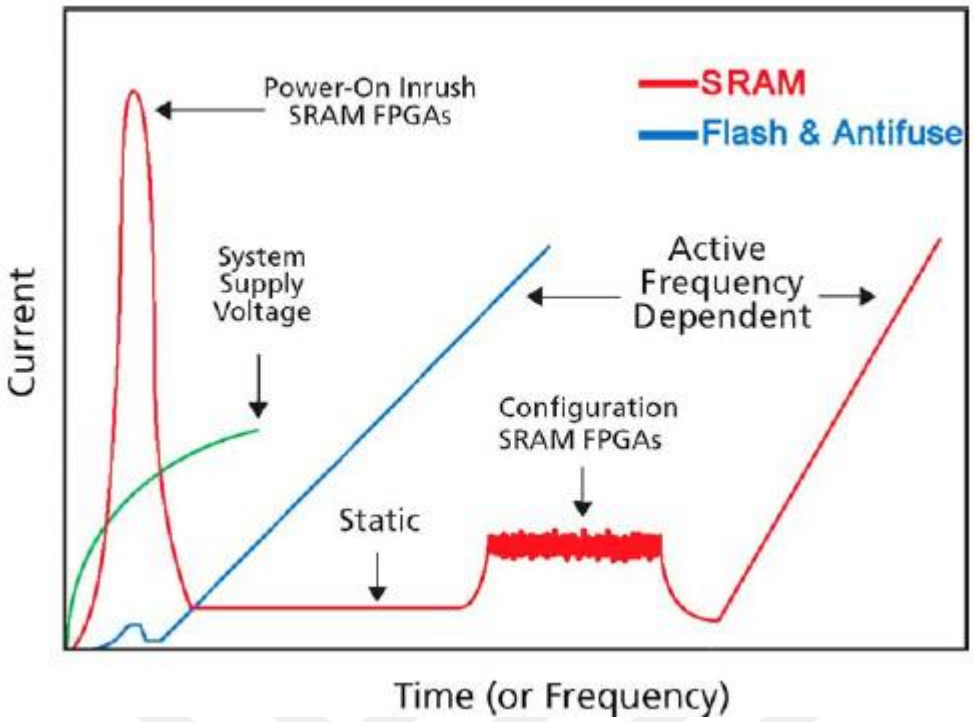
ASIC'ler yüksek hız ve kapasite sağlamaktadır ancak her uygulamaya özel farklı üretimlerinden kaynaklı maliyetleri fazla olmaktadır. FPGA'ler uygulamaya özel tasarımdan doğan maliyeti ortadan kaldırmıştır. Üretilen FPGA'ler yüzlerce farklı firma tarafından farklı amaçlarla kullanılabilir. Eğer çoklu miktarda üretim olacaksa ASIC'ler ve onun ortaya çıkardığı özel tasarım maliyeti düşük olmaktadır. Ancak az sayıda ASIC tasarım yerine FPGA kullanmak maliyeti düşürmektedir. Resim 4.3'te gösterildiği gibi eğer üretilecek ASIC sayısı belirli bir noktadan fazla ise ASIC kullanmak FPGA kullanmaya göre maliyet avantajı sağlamaktadır. Günümüzde FPGA – ASIC kıyaslamasında maliyet önceliği arka planda kalmıştır. Daha çok performans, tasarım hızı, giriş-çıkış sayısı, güç tüketimi gibi konular ön plana çıkmıştır. FPGA'ler sadece özel tasarım maliyetlerini düşürmekle kalmamış, kullanıcılara tasarım kolaylığı da sağlamıştır. Tasarımlarda FPGA kullanmanın kolaylaştırdığı bazı problemler; sinyal bütünlüğü, test, transistör seviyesi tasarım, giriş – çıkış tasarımları ve clock dağılımıdır. FPGA'ler tasarım kolaylığı sağlamakta ve markete açılmayı hızlandırmaktadır. FPGA'lerde hata yapma ihtimali azalırken, ASIC tasarımlarda ise ilk uygulamalarda hata ihtimali fazladır. Hataların düzeltilmesi ve tekrar fabrikasyonda geçen süre ayları bulmaktadır. Bu yüksek hata ihtimali de doğrulama çalışmalarını pahalı kılmaktadır. FPGA tasarımları günler içerisinde yapılabildiği gibi oluşan hataları düzeltmekte kısa sürmektedir. Aynı

şekilde tasarım doğrulama çalışmaları da kısa sürelerde olmaktadır. ASIC üretimleri risklidir. Üretim aşamasında gelen gereksinim değişiklikleri, tasarım hataları sebebiyle ASIC tasarımların üçte biri üretim aşamasına geçmektedir [37].



Resim 4.3 : FPGA ve ASIC karşılaştırması [37]

SRAM ve flash tabanlı FPGA'lerin güç tüketimleri farklılık göstermektedir. SRAM ve flash tabanlı FPGA'lerin zamana bağlı güç tüketimleri Resim 4.4'de gösterilmiştir. Bu çalışmada SRAM tabanlı Xilinx FPGA kullanıldığı için SRAM tabanlı bir FPGA'in güç tüketimi anlatılmıştır. SRAM tabanlı FPGA'lerin güç tüketim modelleri statik ve dinamik güç tüketimlerinden oluşur. Dinamik güç tüketimi; çalışma frekansına, kaynak voltajına, FPGA kaynak kullanımına, anahtarlama aktivitelerine ve efektif kapasitansa bağlıdır. FPGA'lerin ASIC tasarım ile güç tüketimi konusunda rekabet edebilmesi için yaklaşımlar vardır. Düşük güç tüketen uygulamalarda FPGA üzerinde özelleştirilmiş blokların kullanılması önerilmektedir [38].



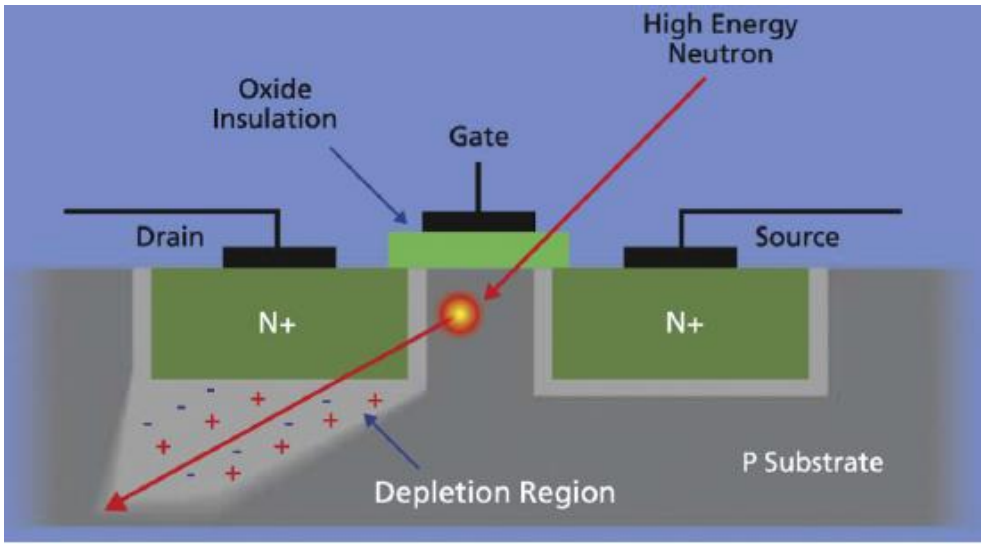
Resim 4.4 : SRAM ve flash tabanlı FPGA'lerin güç tüketimi [38]

FPGA'ler performans ve hızlı tasarım olanağı sağlamasıyla yüksek emniyet arz eden projelerde önemli yer edinmiştir. Modern FPGA'lerde siber güvenliği sağlamak adına şifrelenmiş bitstreamler kullanılarak yüklü olan FPGA'lerdeki tasarımların korunması sağlanır. Xilinx gibi birçok FPGA üreticisi bitstream şifreleme konusunda çözümler sunmaktadır [39].

FPGA'ler aviyonik projelerde yoğun olarak kullanılmaya başlamıştır. NASA hataya toleranslı ve esnek aviyonik sistemler ile yeni uzay araçlarını tasarlamak istemiştir. Geleneksel tasarımlarda hardcore işlemciler ve DSP'ler kullanılmıştır. İletişimlerde de standart bus yapıları tercih edilmiştir. Ancak bu tasarımların dezavantajları olduğu görülmüştür. Sistemin hataya toleranslı olması karmaşıklığı artırmakta ve performansı düşürmektedir. Ayrıca hardcore işlemciler ve DSP'ler kullanılarak yapılan tasarımlarda sonradan değişiklikler yapmak imkânsız veya pahalı olmaktadır. HDL ve hızla gelişen FPGA teknolojisi radyasyona dayanıklı tasarımlara imkân sağlamakta ve System on Chip teknolojisi ile beraber donanım – yazılım temelli tasarımlar gelişimi hızlandırmaktadır [40].

Resim 4.5 : Xilinx SRAM tabanlı FPGA içeren Mars Rover [40]

SRAM tabanlı FPGA'ler flashlı FPGA'lere göre daha iyi performans vermektedir. Kritik görevlerde SRAM tabanlı FPGA'ler kullanılmasına rağmen "Single Event Upset" (SEU) bu sistemler için tehlike oluşturabilir. SEU sistem üzerinde farklı hatalara sebep olabilir. Bu hatalar; durum değişimleri veya kalıcı yanmalar olabilir. Kalıcı yanma durumunda sistem tamamen kullanılamaz hale gelebilir. Resim 4.6'da bir nötron taneciğinin elektronik cihazdaki atomlar ile çarpışması gösterilmiştir. Bu çarpma iyonizasyona sebep olmaktadır. Setup ve hold zamanlarında oluşan SEU'lar sistemin hatalı davranmasına neden olur [41].

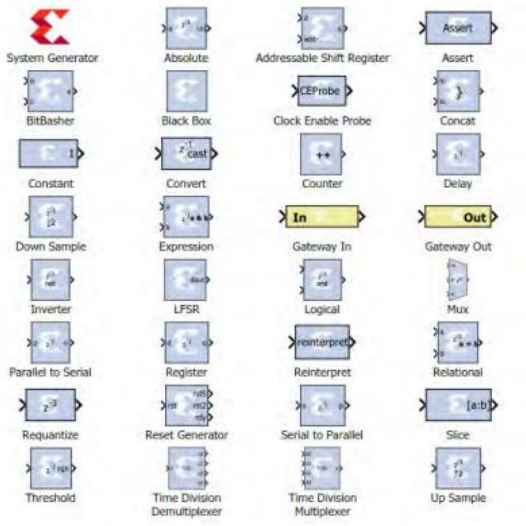


Resim 4.6 : Nötron parçacığının yarı iletkene çarpması [41]

SEU'dan korunmak için FPGA'ler kendilerini düzeltebilirler. SEU'yu tespit etmek için hata bulma ve düzeltme devreleri ile konfigürasyon hafızası denetlenebilir. Xilinx Virtex FPGA'lerinde konfigürasyon hafızasını okumak ve tekrar konfigüre etmek için ICAP ismi verilen portlar vardır. ICAP üzerinden yazma ve okuma komutları ile konfigürasyon hafızasına erişilir. ICAP'in radyasyona dayanıklı hardcore bir işlemci tarafından kontrol edildiği uygulamalar vardır. FPGA üzerine hem uygulama hem de hata bulma ve düzeltme devresi yerleştirilebilir. Buradaki önemli konu hata bulma ve düzeltme devresinin SEU'dan etkilenmemesidir. Hata bulma ve düzeltme devresinin SEU'dan etkilenme ihtimalini azaltmak için "Tripple Modular Redundancy" (TMR) tekniği kullanılır [42].

4.2.Xilinx Blokları ile PID İrtifa Kontrolcüsünün Oluşturulması

DSP için System Generator, Vivado Design Suite ile beraber gelen ve Simulink ortamında FPGA geliştirmek için kullanılan bir yazılımdır. RTL tasarım teknikleri ve Xilinx FPGA tecrübesi istemeden tasarım ve simülasyon yapma kolaylığı sağlar. Tasarımlar Xilinx blokları kullanılarak oluşturulur, RTL sentezi ve implementasyon yapılarak FPGA konfigürasyonunda kullanılan bitstream üretilir. Xilinx blokları ile yapılan tasarımlar, Simulink blokları ile beraber test edilebilir. System Generator, Simulink'e ek olarak 130'dan fazla hazır blok bulundurur (Resim 4.7). Basit Xilinx blokları; toplayıcı, çarpıcı, register'dır. Ek olarak System Generator karmaşık yapılı DSP blokları da içerisinde barındırır; FFT bloğu, filtereler, hafıza birimleri.[43] [44]



Resim 4.7 : System Generator'da bulunan Xilinx blokları [44]

Bu bölümde ayrık kontrolcü Xilinx System Generator blokları ile oluşturulmuş ve Simulink blokları ile beraber test edilmiştir.

Ayrık hale getirilecek PID kontrolcünün denklemi Eşitlik 4.1'de gösterilmiştir. Eşitlikte K_p oransal kazancı, T_i integral sabitini, T_d diferansiyel sabiti, e hatayı, u ise kontrol çıkışını belirtmektedir [45].

$$u(t) = K_p\{e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + \frac{T_d de(t)}{dt}\} \tag{4.1}$$

Eşitlik 4.1 T örnekleme zamanı ile Eşitlik 4.2'ye dönüştürülmüştür. İmplementasyon için hesaplanan hata değerleri kaydedilmelidir ve bu gerçek uygulamalar için oldukça zordur.

$$u(k) = K_p\{e(k) + \frac{T}{T_i}\sum_{i=0}^{k-1} e(i) + \frac{T_D}{T}[e(k) - e(k-1)]\} \tag{4.2}$$

Bütün hata değerlerini kaydetmek yerine recursive PID algoritması kullanılabilir [8]. Eşitlik 4.2'den yola çıkarak u(k-1) çıkışı Eşitlik 4.3'de gösterildiği gibi yazılabilir.

$$u(k-1) = K_p\{e(k-1) + \frac{T}{T_i}\sum_{i=0}^{k-1} \mathrm{e(i)} + \frac{T_D}{T}[e(k-1) - e(k-1)]\} \tag{4.3}$$

Recursive kontrol çıkışını hesaplamak için Eşitlik 4.2 ve 4.3 farkı alınır. Alınan farka göre recursive kontrol çıkışı Eşitlik 4.5'te gösterilmiştir.

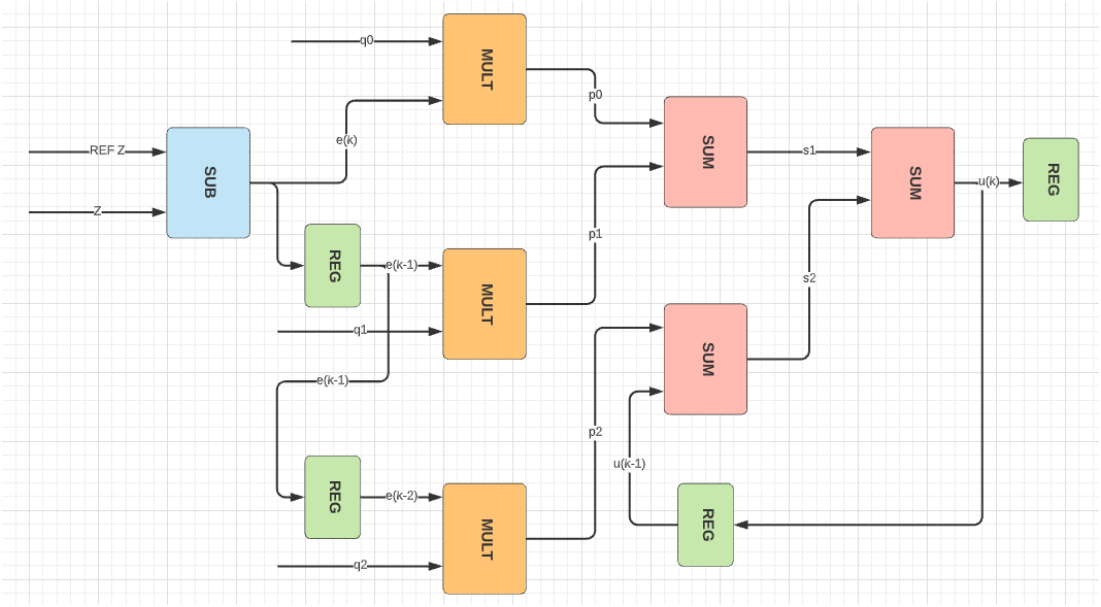
$$u(k) - u(k-1) \tag{4.4}$$
$$= P\{e(k) - e(k-1) + \frac{T}{T_i}e(k-1) + \frac{T_D}{T}[e(k) - 2e(k-1) - e(k-2)]\}$$

$$u(k) = u(k-1) + q_0 * e(k) + q_1 * e(k-1) + q_3 * e(k-2) \tag{4.5}$$

Eşitlik 4.5'te kullanılan q_0, q_1, q_2 değerleri Eşitlik 4.6'da gösterilmiştir.

$$q_0 = P$$
$$q_2 = -P(1 - \frac{T}{T_i} + \frac{2 * T_d}{T}) \tag{4.6}$$
$$q_3 = P(\frac{T_d}{T})$$

FPGA üzerinde geliştirilecek olan paralel ayrık PID mimarisi Şekil 4.1'de gösterilmiştir. Registerlar e(k-1), e(k-2) ve u(k-1) değerlerini saklamak için kullanılmıştır.



Şekil 4.1 : Ayrık paralel PID tasarımı

Şekil 4.1'de gösterilen giriş – çıkış değerleri Eşitlik 4.7'de gösterilmiştir.

$$\mathrm{e(k)} = \mathrm{Ref\,Z} - \mathrm{Z}$$

$$p_0 = q_0 * e(k)$$

$$p_1 = q_1 * e(k-1)$$

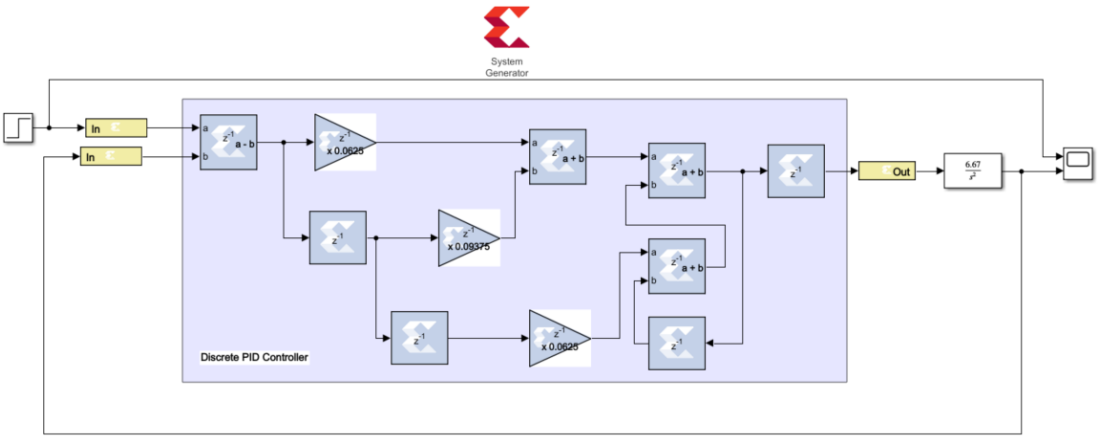
$$p_2 = q_2 * e(k-2) \qquad (4.7)$$

$$s_1 = p_0 + p_1$$

$$s_2 = p_2 + s_1$$

$$u(k) = \mathrm{s}_2 * u(k-1)$$

Ayrık paralel PID modeli Xilinx System Generator üzerinde Xilinx blokları kullanılarak oluşturulmuştur (Şekil 4.2). FPGA blokları ile Simulink blokları arasındaki bağlantı "Gateway In/Out" blokları ile yapılmıştır.

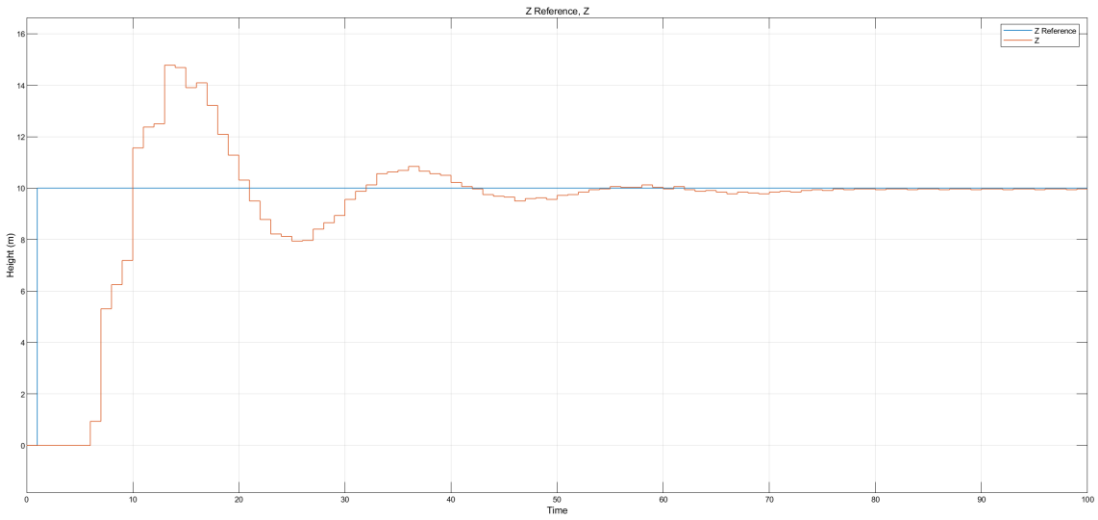


Şekil 4.2 : Ayrık paralel PID tasarımı

Çizelge 4.1 : Ayrık PID katsayıları

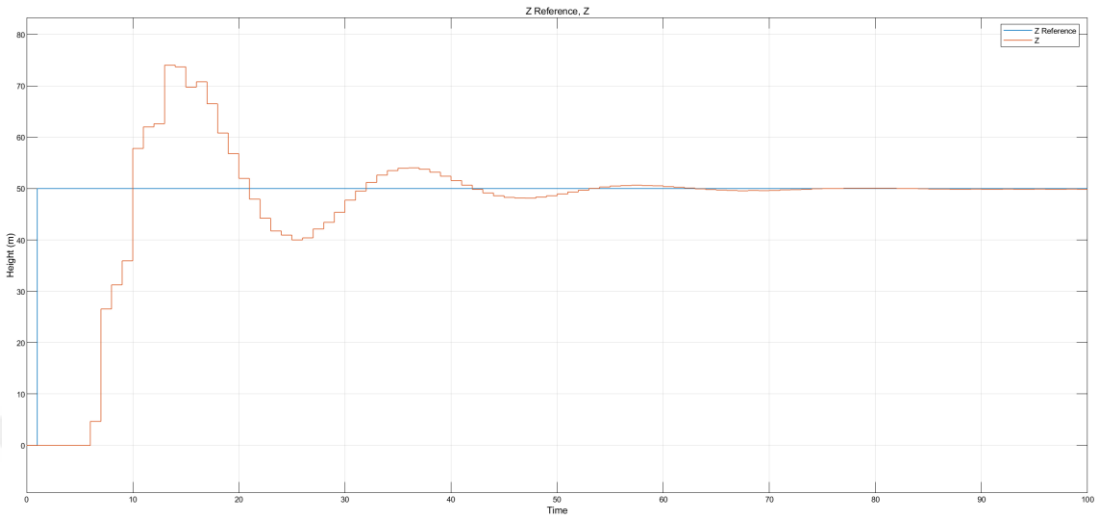
Parametreler	Değerler
Kp	0.093
Ki	0.437
Kd	0.093

Giriş referansı 10 metre olarak verilmiş ve referans takibi Şekil 4.3'te gösterilmiştir. 10 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 4.3 : Referans 10 metre ile ayrık PID irtifa kontrolcü testi

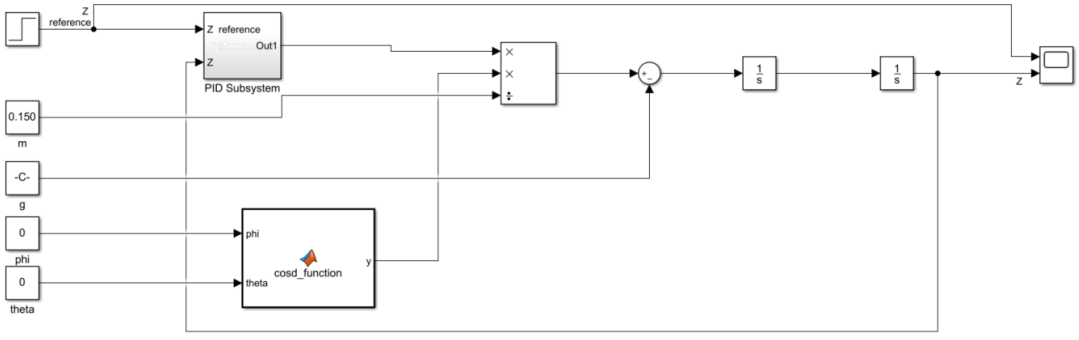
Giriş referansı 50 metreye çıkarılmış ve referans takibi Şekil 4.4'te gösterilmiştir. 50 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 4.4 : Referans 50 metre ile ayrık PID irtifa kontrolcü testi

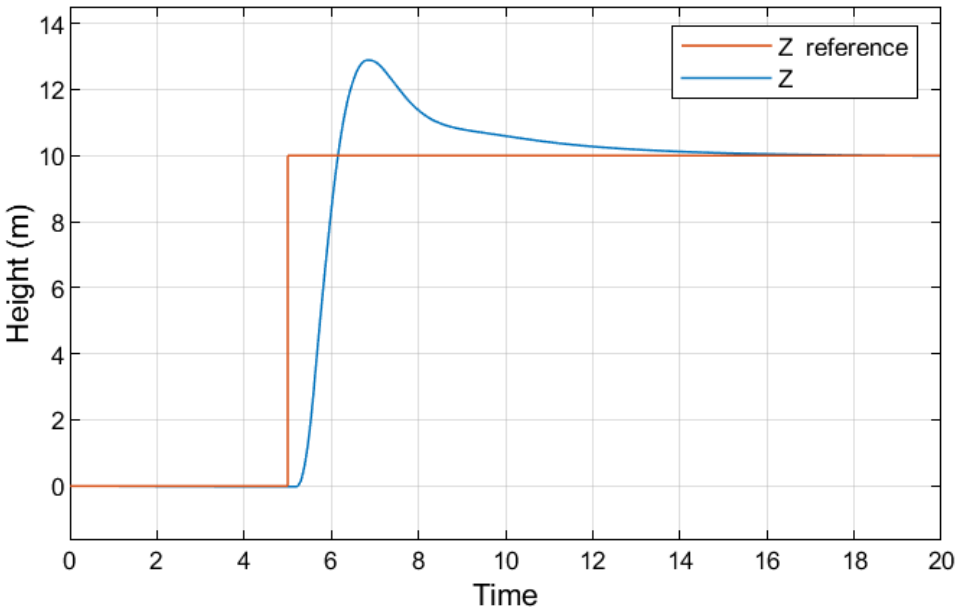
4.3.Matlab HDL Coder ile PID İrtifa Kontrolcüsünün Oluşturulması

Bu bölümde irtifa kontrolcünün sayısal tasarımı yapılmadan önce PID subsystem'inin referans takibini doğrulamak için Şekil 4.5'te gösterilen blok tasarım Simulink üzerinde gerçeklenmiştir.



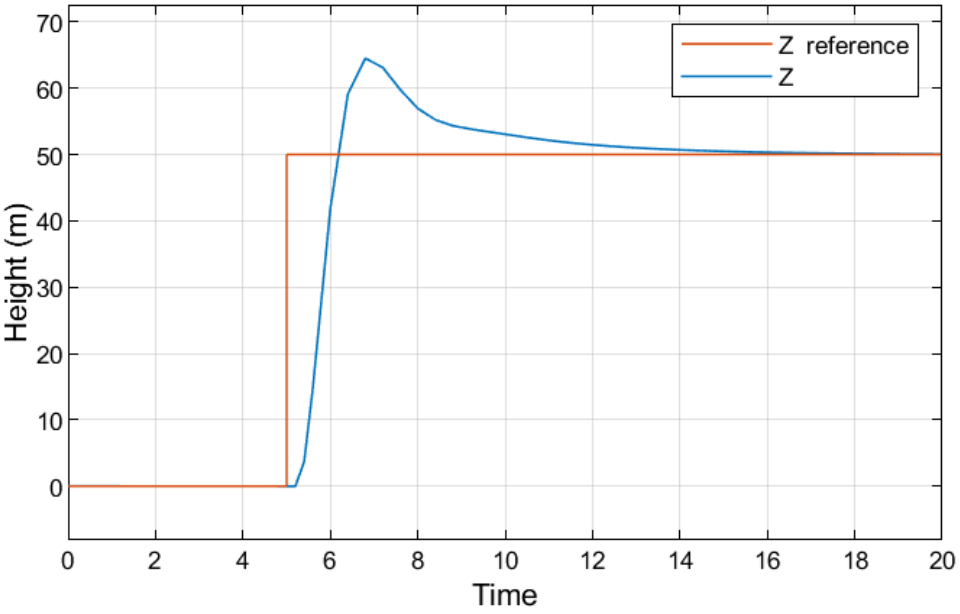
Şekil 4.5 : Sayısal tasarıma çevirilecek olan PID subsystem

İrtifa kontrolcüsünü test etmek için ilk olarak 10 metrelik referans değeri sisteme verilmiştir. Sistemin referans takibi Şekil 4.6'da gösterilmiştir. 10 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 4.6 : Referans 10 metre ile PID irtifa kontrolcü testi

Giriş referansı 50 metreye çıkarılmış ve test tekrarlanmıştır. Referans takibi Şekil 4.7'de gösterilmiştir. 50 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 4.7 : Referans 50 metre ile PID irtifa kontrolcü testi

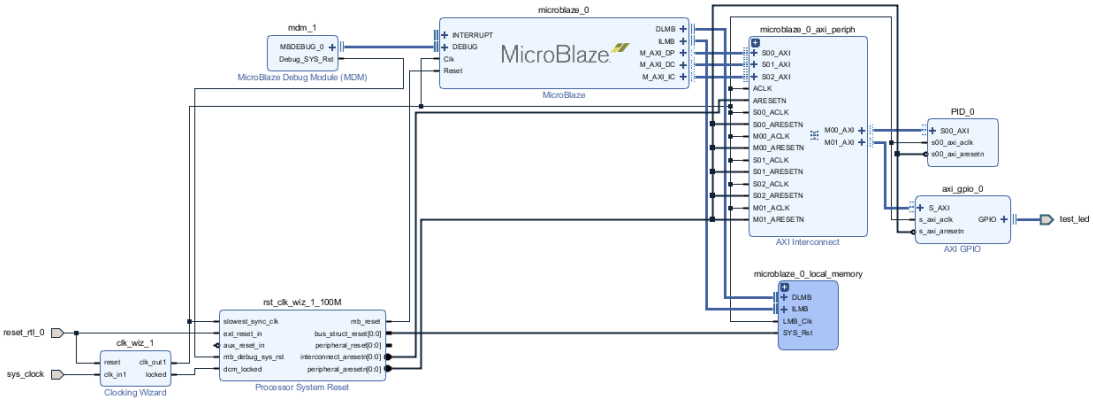
HDL coder ile sayısal devre haline getirelecek olan PID subsystem kodu Resim 4.8'de gösterilmiştir. Sentez sonucu kaynak kullanımı Çizelge 4.2'de verilmiştir. HDL'e dönüştürülen PID irtifa kontrolcünün IP'si oluşturulmuştur (Resim 4.9).

```
function out = pid_funct(refZ, Z)

    kp = 0.05;
    ki = 0.1;
    kd = 0.05;
    dt = 0.1;

    error = refZ - Z;

    Pout = kp* error;

    persistent integral_;
    persistent pre_error;

    if isempty(integral_)
        integral_=0;
    end
    if isempty(pre_error)
        pre_error=0;
    end

    integral_ = integral_ + error * dt;

    Iout = ki * integral_;

    derivative = (error - pre_error) / dt;
    Dout = kd * derivative;

    pre_error = double(error);
    out = Pout + Iout + Dout;

end
```

Resim 4.8 : HDL'e dönüştürülecek olan kod parçası

Çizelge 4.2 : FPGA kaynak kullanımı (Default sentez)

	LUT	FF	BRAM	URAM	DSP
Değerler	576	0	0	0	4



Resim 4.9 : PID irtifa kontrolcü IP'si

4.4.High Level Synthesis ile İrtifa Kontrolcü Oluşturulması

High level synthesis (HLS) ile FPGA tasarımı popülaritesi giderek artmaktadır. Özellikle Altera ve Xilinx gibi iki öncü FPGA firmasının Intel ve AMD tarafından satın alınması ilerleyen yıllarda HLS kullanımının giderek yaygınlaşacağını düşündürmektedir. HLS'in HDL'e göre tercih edilmesinin sebebi tasarım kolaylığı sağlamasıdır. HLS ile C, C++, SystemC gibi diller kullanılarak FPGA tasarımları geliştirilmektedir. Böylece register transfer level (RTL) tasarımında karşılaşılan problemler kısmen önlenmektedir. Ayrıca VHDL/Verilog gibi dillerin kullanımı ve hata oranı C/C++ dillerine göre daha fazladır [46].

Bu bölümde Vivado HLS üzerinde irtifa kontrolcüsü ve irtifa kontrolcü test benchi hazırlanmıştır. Test bench kullanılarak farklı irtifa değerlerinde kontrolcünün çalışma performansı incelenmiştir. C ile yazılan kontrolcü, VHDL dili seçilerek sentezlenmiş ve kaynak kullanımı sonuçlar bölümünde gösterilmiştir.

Vivado HLS üzerinde hazırlanan irtifa kontrolcü kodu Resim 4.10'te gösterilmiştir.

```
int pid(float *refZ, float *z, float *newZ)
{
    #pragma HLS INTERFACE ap_none port=*refZ
    #pragma HLS INTERFACE ap_none port=*z
    #pragma HLS INTERFACE ap_none port=*newZ

    float kp = 0.5;
    float ki = 1.5;
    float kd = 0.01;

    float Pout;
    float Iout;
    float Dout;
    float dt = 0.1;
    static float integral = 0;
    float derivative = 0;
    static float pre_error = 0;
    float out;
    float error;

    error = *refZ - *z;

    Pout = kp * error;

    integral += error * dt;
    Iout = ki * integral;

    derivative = (error - pre_error) / dt;
    Dout = kd * derivative;

    pre_error = error;

    out = Pout + Iout + Dout;
    *newZ = out;

    return out;
}
```

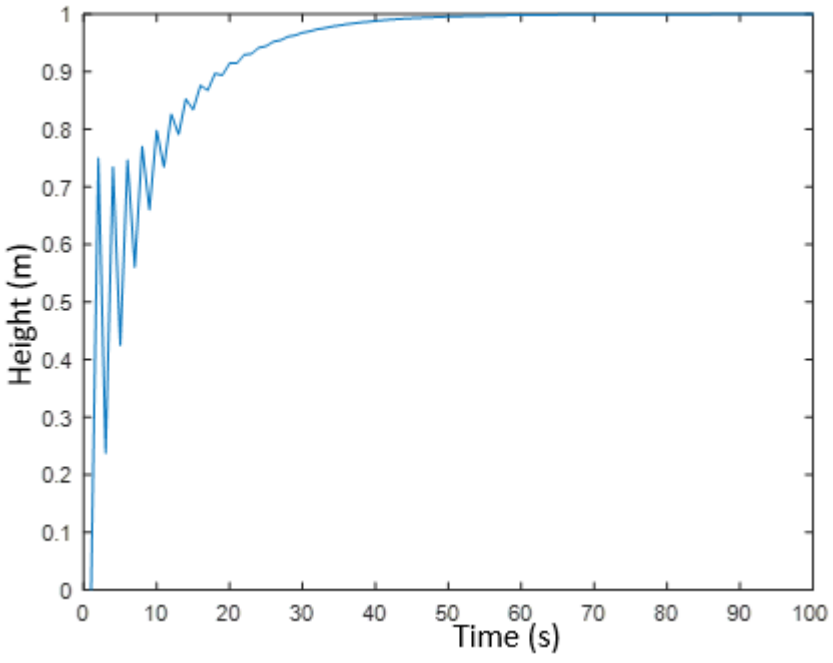
Resim 4.10 : Vivado HLS irtifa kontrolcü kod bloğu

İrtifa kontrolcü bloğunun testi Vivado HLS üzerinde test bench oluşturularak yapılmıştır. Test bench kodu Resim 4.11'de gösterilmiştir.

```c
#include <stdio.h>

int main()
{
    float refZ = 50;
    float newZ = 0;
    float z = 0;
    int i = 0;

    for(i = 0; i<100; i++)
    {
        pid(&refZ, &z, &newZ);
        printf("%03f \n", z);
        z = newZ;
    }

    return 0;
}
```

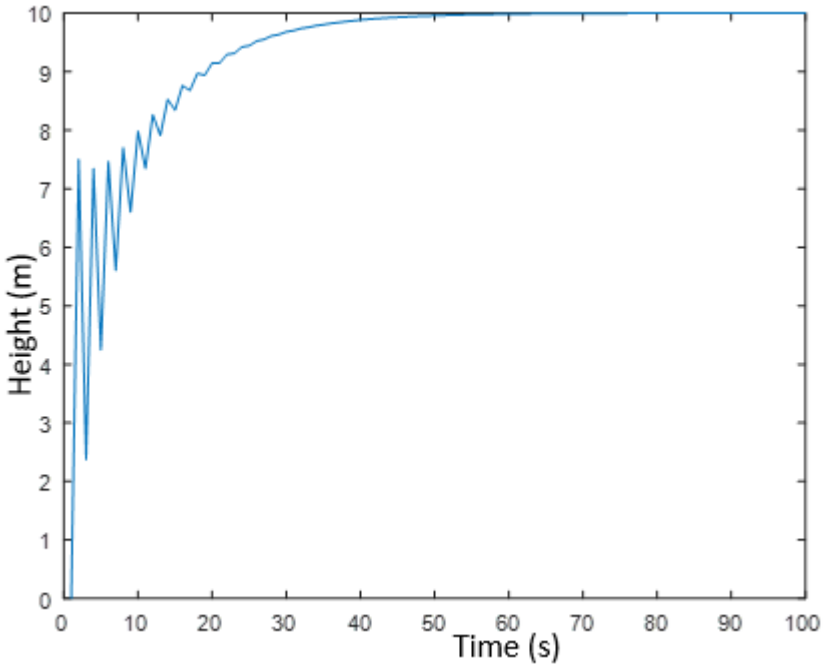
Resim 4.11 : Vivado HLS irtifa kontrolcü test bench kodu

HLS test bench üzerinde referans irtifa 1 metre olarak verilmiş ve simülasyon koşturulmuştur. Kontrolcü performansı Şekil 4.8'te gösterilmiştir.



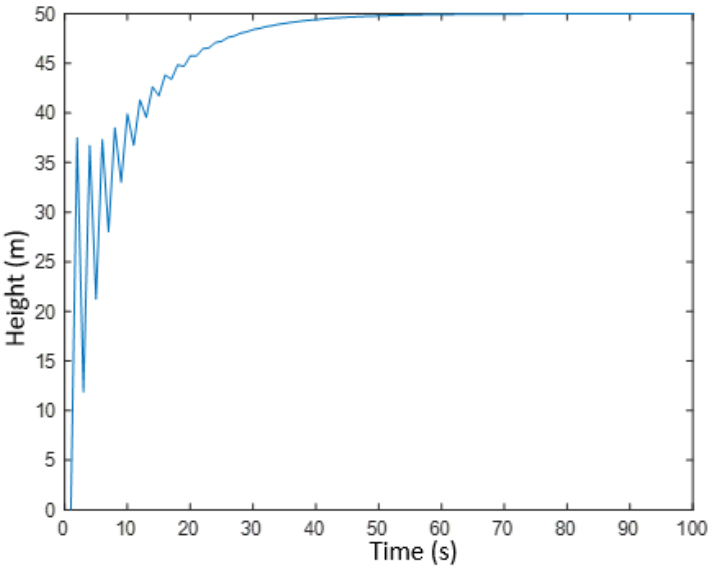
Şekil 4.8 : HLS test bench 1 metre referans irtifa

HLS test bench üzerinde referans irtifa 10 metreye çıkarılmış ve simülasyon koşturulmuştur. Kontrolcü performansı Şekil 4.9'te gösterilmiştir.



Şekil 4.9 : HLS test bench 10 metre referans irtifa

HLS test bench üzerinde referans irtifa 50 metreye çıkarılmış ve simülasyon koşturulmuştur. Kontrolcü performansı Şekil 4.10'da gösterilmiştir.



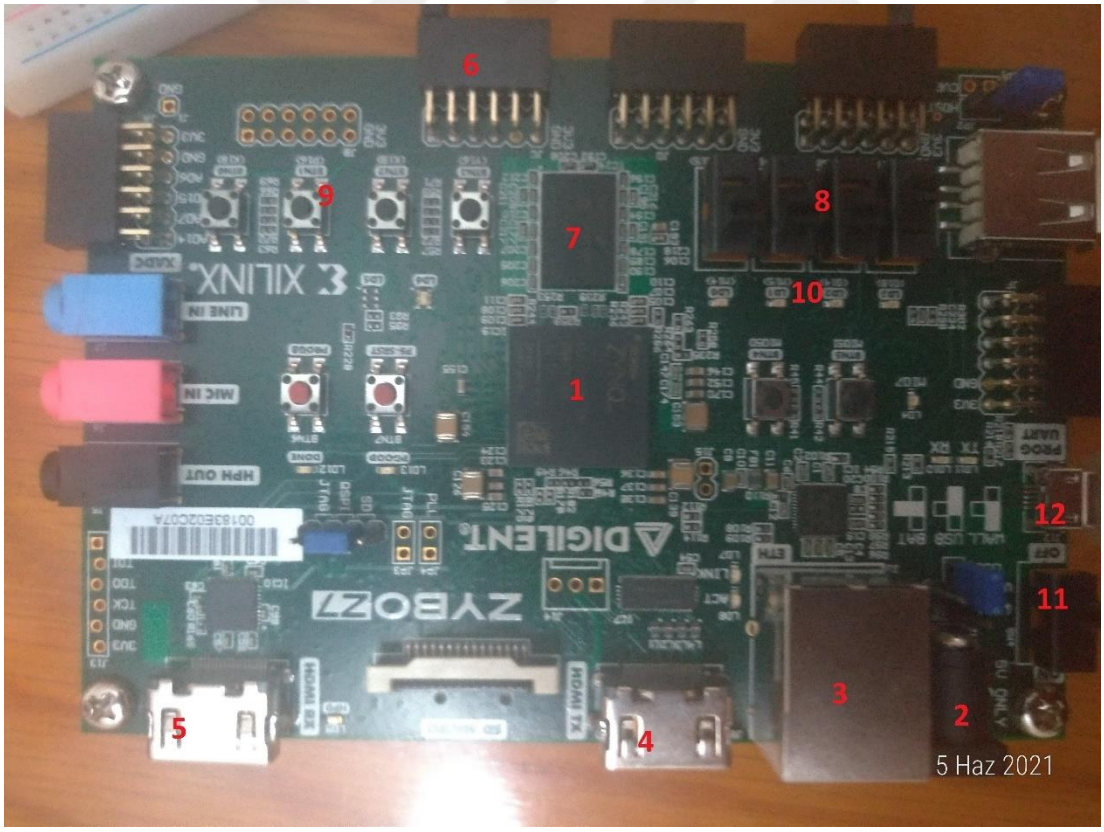
Şekil 4.10 : HLS test bench 50 metre referans irtifa

4.5.FPGA Üzerinde Softcore İşlemci ile İrtifa Kontrolcünün Oluşturulması

4.5.1. Zybo Z7-10 Kartı ve MicroBlaze ile İlgili Bilgiler

Bu bölümde irtifa kontrolcü, Microblaze soft işlemci ile Zybo Z7-10 ARM & FPGA SoC üzerinde gerçeklenmiştir. FPGA üzerinde implementasyonu yapılan kontrolcü ile Simulinkte yer alan irtifa modeli HIL testi ile referans takibi doğrulanmıştır. Geliştirme kartı üzerinde bulunan butonlar aracılığı ile sistemin çalışması esnasında irtifa değerleri değiştirilebilmektedir.

Zybo Z7 (Resim 4.12) içerisinde Xilinx Zynq-7000 bulundurmaktadır. Zynq Processing System (PS) ve Programmable Logic (PL) isimli iki birimden oluşmaktadır. PS çift çekirdekli ARM A9 işlemci bulundururken, PL'de Xilinx 7 serisi FPGA bulunur [47]. Çizelge 4.3'te Zybo Z7-10 kartı birimleri gösterilmiştir. Çizelge 4.4'te ise kartın sahip olduğu özellikler belirtilmiştir.



Resim 4.12 : Zybo Z7-10 kartı görünümü

Çizelge 4.3 : Zybo Z7-10 kartı birimleri

Numara	İsim	Numara	İsim
1	Zynq 7000	7	DDR3 hafıza
2	Harici güç girişi	8	Switchler
3	Ethernet port	9	Butonlar
4	HDMI çıkışı	10	Ledler
5	HDMI girişi	11	Açma – kapama
6	PMOD	12	JTAG ve UART arayüzü

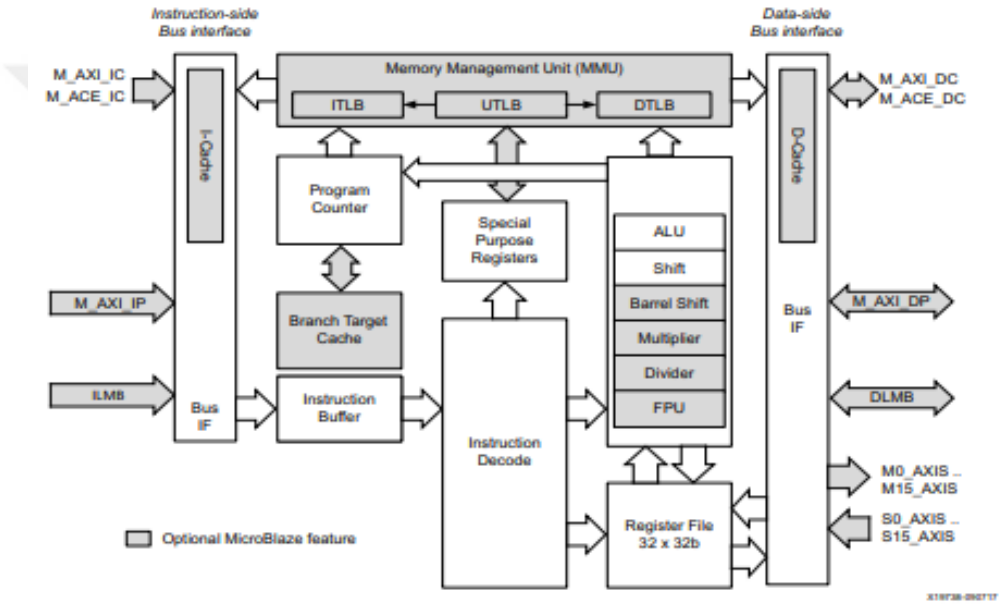
Çizelge 4.4 : Zybo Z7 özellikleri (37)

	Zybo Z7-10	Zybo Z7-20
Zynq Part	XC7Z010-1CLG400C	XC7Z020-1CLG400C
ADC	Evet	Evet
Look-up Tables (LUTs)	17600	53200
Flip Flops	35200	53200
Block RAM	270 KB	630 KB
Clock Management Tile	2	4
Number of PMOD	5	6
Fan connector	Yok	Var
Heat sink	Yok	Var

Microblaze soft işlemcisi RISC mimarisinde Xilinx FPGA'leri üzerinde kullanılmak üzere optimize edilmiştir. Microblaze soft işlemcisinin fonksiyonel yapısı Resim 4.13'de gösterilmiştir. [48]

Microblaze içeriği [48];

- 32 adet, 32 bit genel amaçlı register,
- 32 bit adres bus,
- 18 adet, 32 bit özel amaçlı register



Resim 4.13 : Microblaze çekirdeği blok yapısı [48]

Microblaze çekirdeğinin farklı FPGA ailelerinde çalışabildiği maksimum frekans değerleri Çizelge 4.5'te gösterilmiştir.

Çizelge 4.5 : Maksimum frekans değerleri

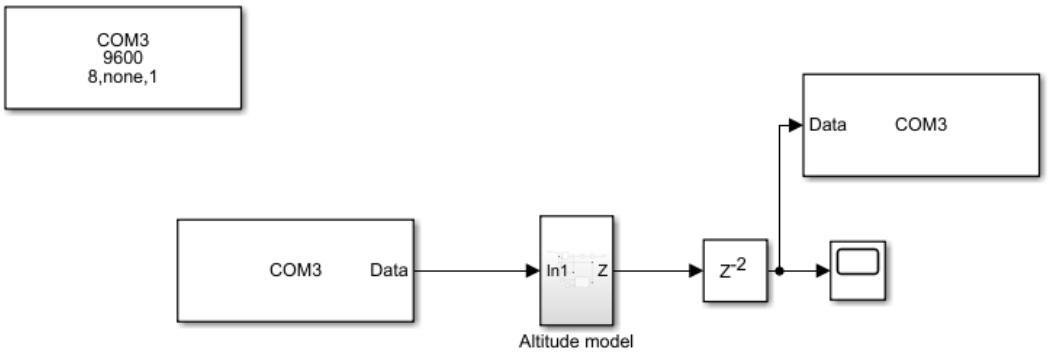
Aile	Frekans (Mhz)
Virtex-7	393
Kitnex-7	399

Artix-7	260
Zynq-7000	272
Spartan-7	232
Virtex UltraScale+	711

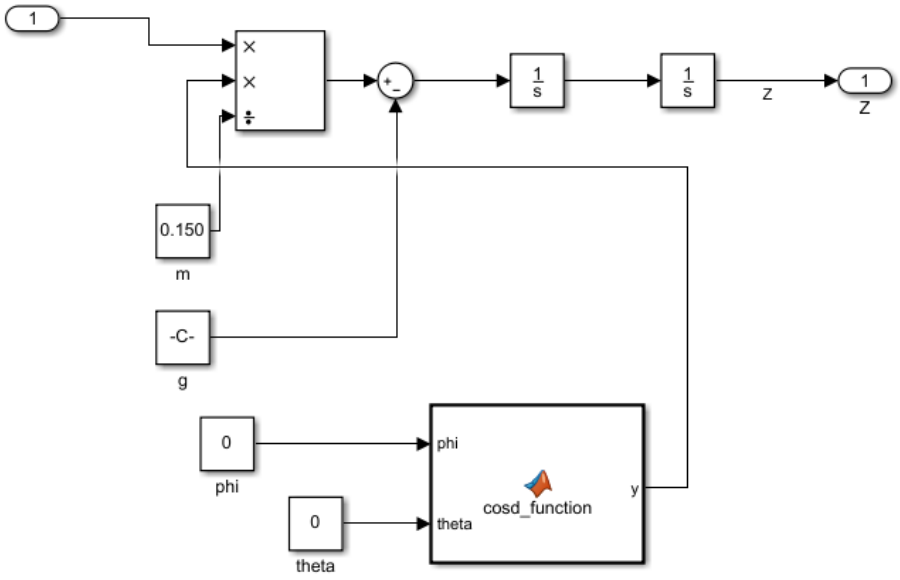
4.5.2. Simulink Üzerinde İrtifa Modelinin Gerçeklenmesi

İrtifa modeli Simulink üzerinde gerçeklenmiştir. Fiziksel dünyada yer alan FPGA'i Simulinke bağlamak için "Instrumentation Toolbox" kullanılmıştır (Şekil 4.11 - Şekil 4.12).

- Model – FPGA arası iletişimde UART kullanılmıştır ve baudrate değeri 9600 olarak ayarlanmış.
- İrtifa kontrolcüsünün çıktısı "Serial Receive" bloğu ile irtifa modeline verilmiştir.
- İrtifa modeli çıktısı "Serial Send" bloğu ile FPGA'e gönderilmiştir.



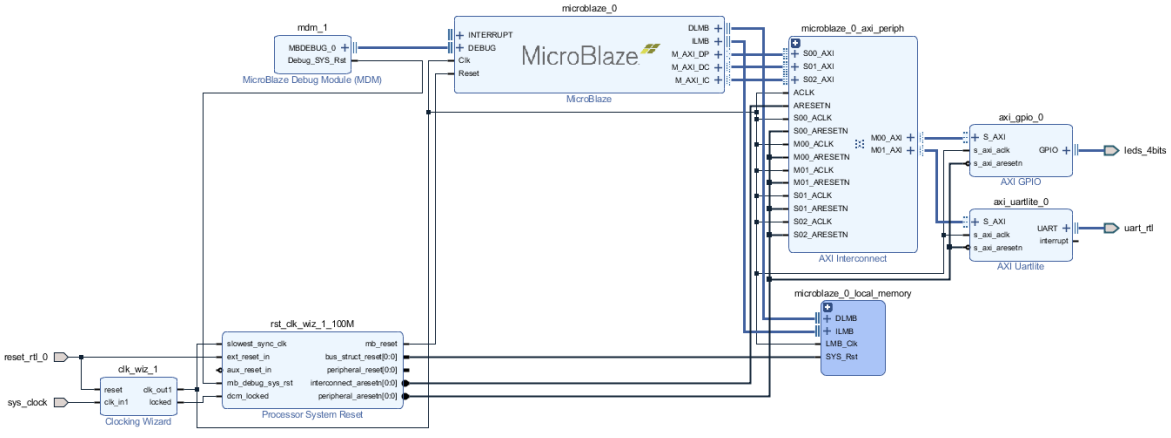
Şekil 4.11 : HIL modeli



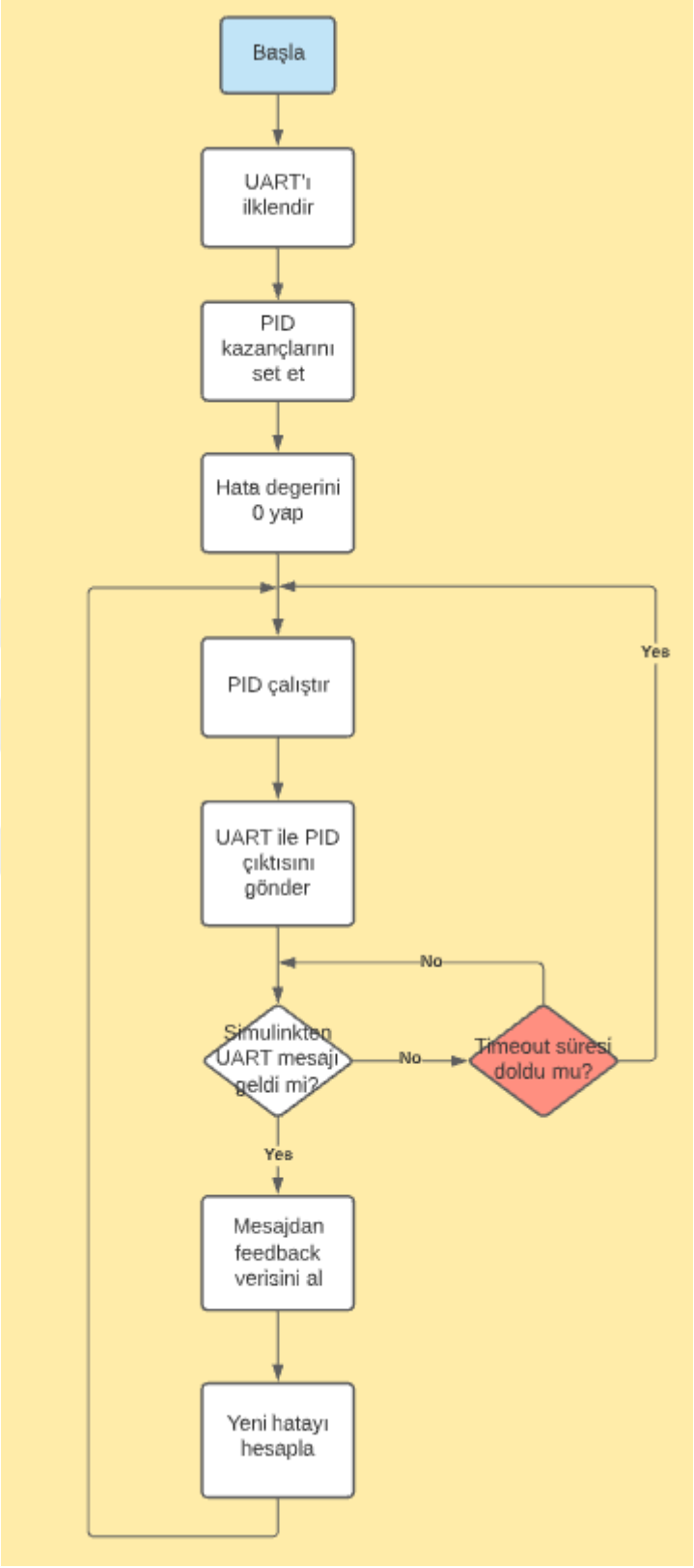
Şekil 4.12 : İrtifa submodeli

4.5.3. FPGA Üzerinde İrtifa Kontrolcüsünün Gerçeklenmesi

Kontrol algoritması FPGA tabanlı softcore bir işlemci olan MicroBlaze üzerinde gerçeklenmiştir. MicroBlaze'in Simulinkte koşan modele veri gönderebilmesi için tasarıma UART IP bloğu eklenmiştir ve Resim 4.14'de gösterilen diagram oluşturulmuştur. MicroBlaze üzerinde koşan yazılımın akışı Resim 4.15'de gösterilmiştir.

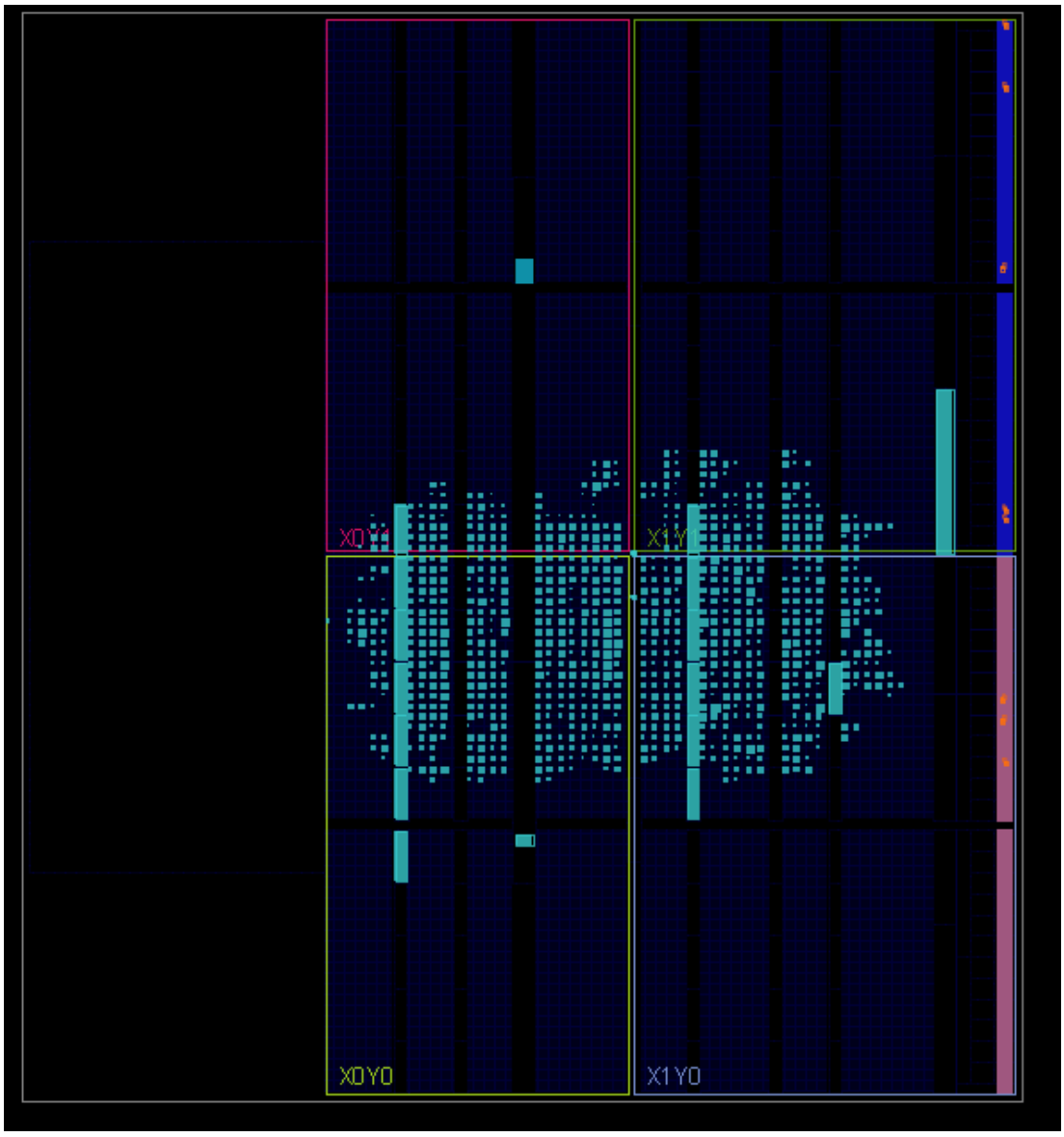


Resim 4.14 : Microblaze içeren Vivado tasarımı



Resim 4.15 : İrtifa kontrolcüsü yazılım akışı

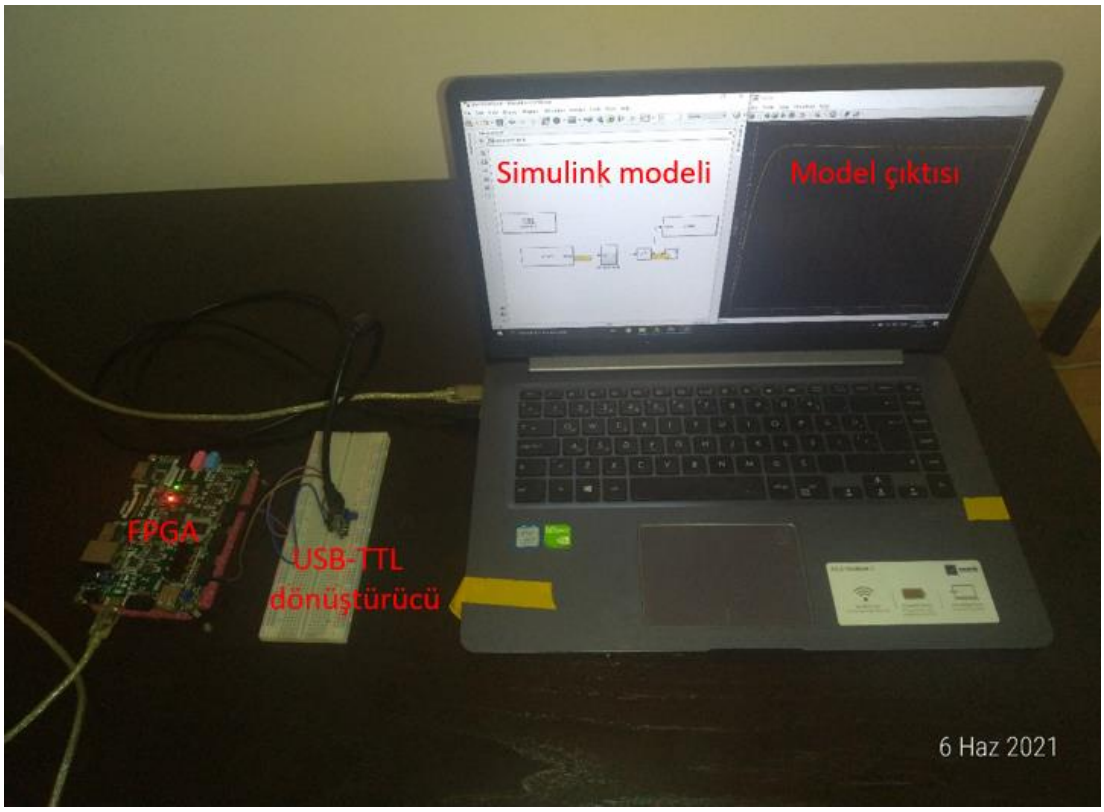
Vivado üzerinde gerçeklenen tasarım sentezlenmiş ve implementasyonu yapılmıştır (Resim 4.16).



Resim 4.16 : MicroBlaze içeren irtifa kontrolcünün FPGA implementasyonu

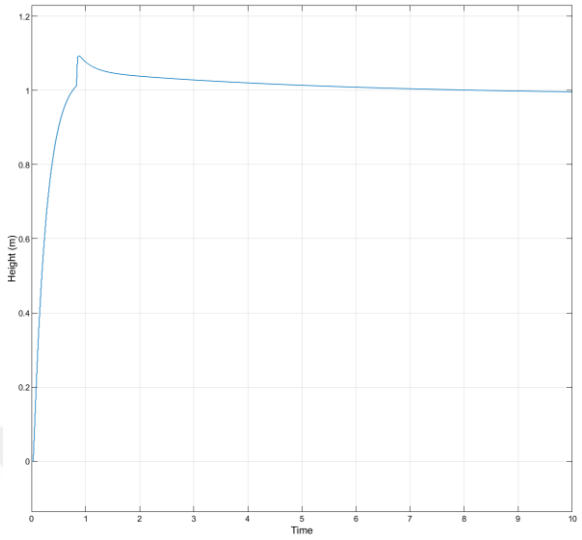
4.5.4. HIL Test Düzeneğinin Oluşturulması

FPGA üzerinde çalışan irtifa kontrolcüsü ile modeli test etmek için Resim 4.17'de gösterilen test düzeneği kurulmuştur. FPGA üzerinde MicroBlaze soft işlemci bulunmakadır ve PID irtifa kontrolcü algoritmasını çalıştırmaktadır. İrtifa modeli Simulink üzerinde gerçeklenmiştir. FPGA – Simulink arası veri transferi için UART protokolü kullanılmıştır. Kontrolcü çıkışlarını bilgisayara aktarabilmek için USB-TTL dönüştürücü kullanılmıştır.



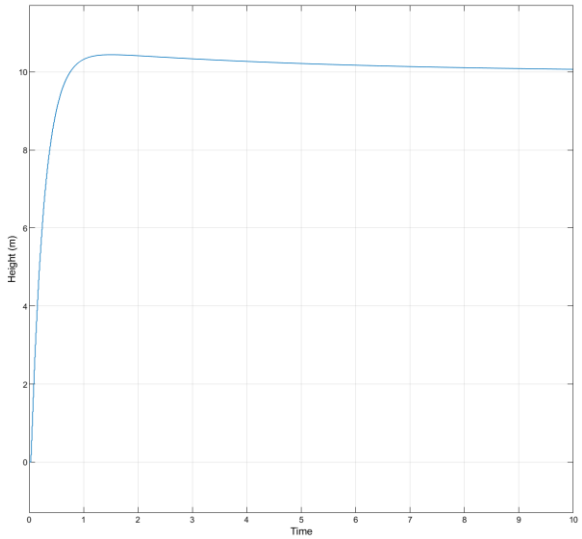
Resim 4.17 : HIL test düzeneği

FPGA’e irtifa referansı 1 metre olarak verilmiştir ve referans takibi Şekil 4.13’da gösterilmiştir. 1 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



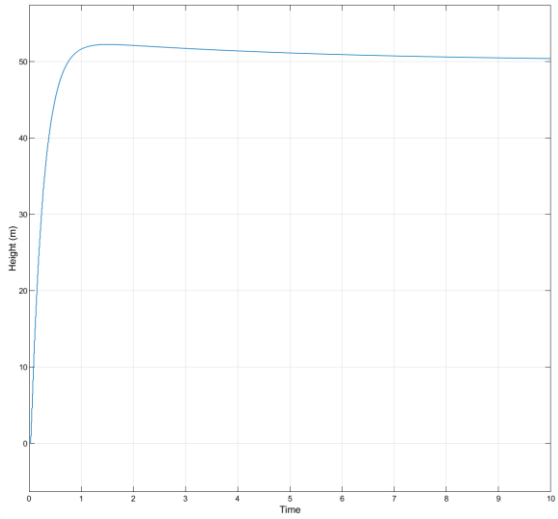
Şekil 4.13 : Referans takibi 1 metre

FPGA’e verilen irtifa referans değeri 10 metre olarak güncellenmiş ve test tekrarlanmıştır (Şekil 4.14). 10 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



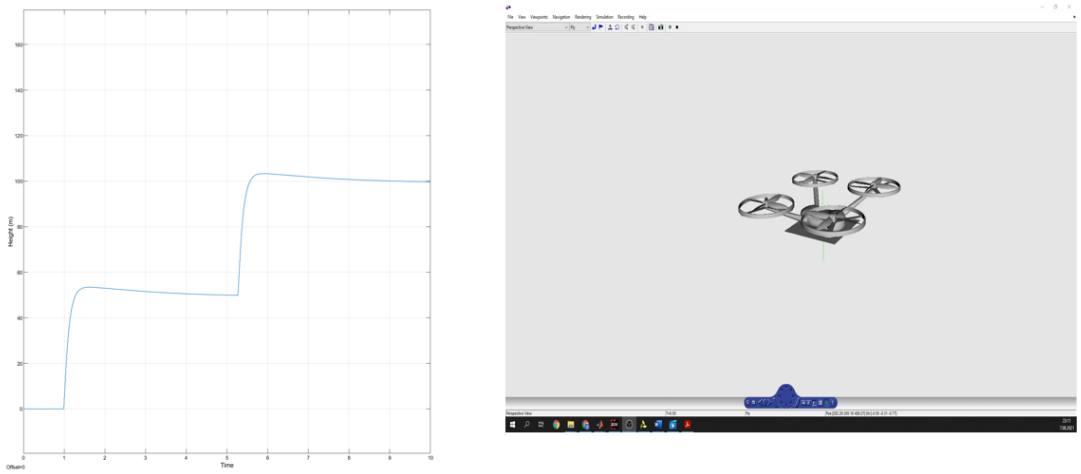
Şekil 4.14 : Referans takibi 10 metre

FPGA'e verilen irtifa referans değeri 50 metre olarak güncellenmiş ve test tekrarlanmıştır (Şekil 4.15). 50 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 4.15 : Referans takibi 50 metre

Butonlar aracılığı ile sistem çalışırken irtifa değerleri güncellenebilmektedir. Ayrıca VR sink aracılığı ile sistemin davranışı gösterilmektedir (Şekil 4.16).



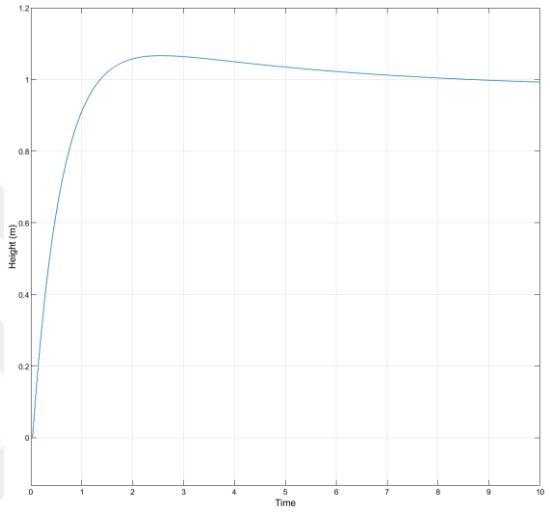
Şekil 4.16 : Buton değerleri ile sisteme irtifa değerlerinin girilmesi

4.5.5. Modele Parametre Belirsizliği Eklenmesi

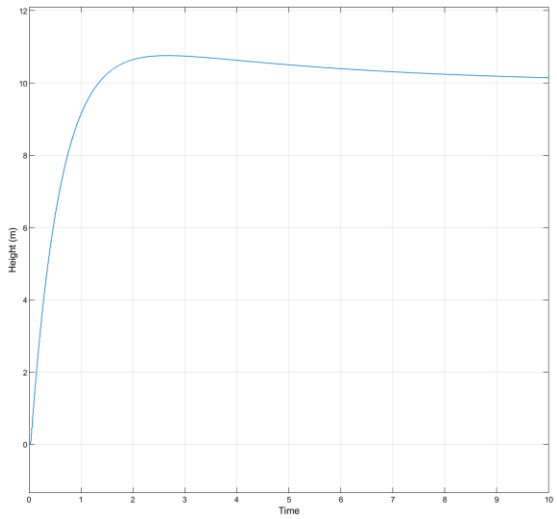
Quadrotorlar farklı görevlerde kullanılabilirler. Bu görevlere bağlı olarak donanımları ve yükleri değişmektedir (Örnek: kamera ve silah eklenmesi). Kontrolcünün değişen yüke karşı dayanıklı olması beklenir. Quadrotorun ağırlığına

sırasıyla %100, %150 ve %200 oranlarında parameter belirsizliği uygulanmış ve farklı irtifalarda referans takibi performansları incelenmiştir.

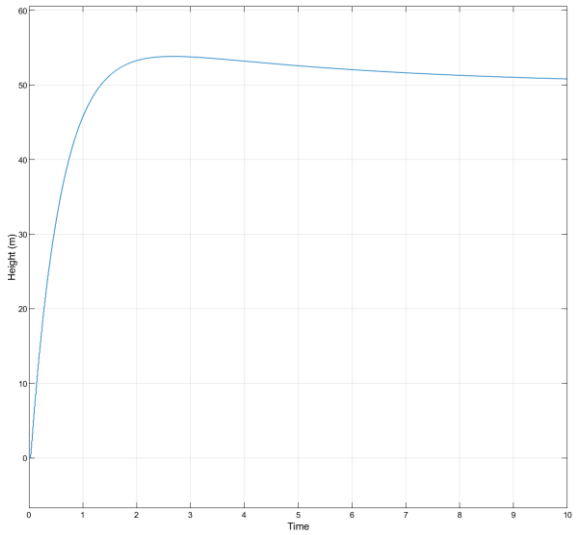
Parametre belirsizliği %100 olarak belirlenmiş ve Quadrotorun sırasıyla 1, 10 ve 50 metre irtifalarında referans takibi performansı Şekil 4.17, Şekil 4.18 ve Şekil 4.19 'te gösterilmiştir.



Şekil 4.17 : %100 parametre belirsizliği, 1 metre referans irtifa

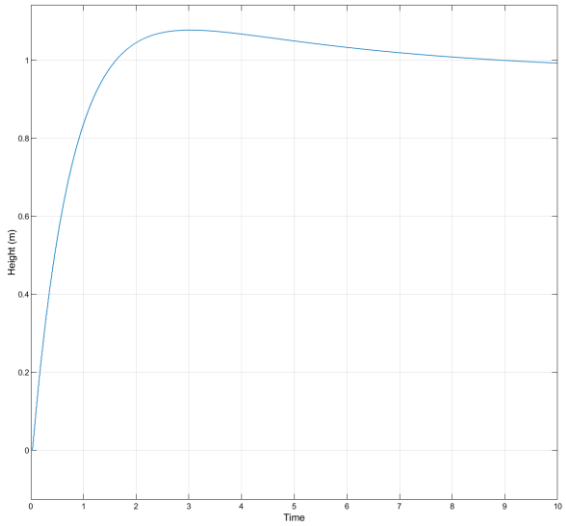


Şekil 4.18 : %100 parametre belirsizliği, 10 metre referans irtifa

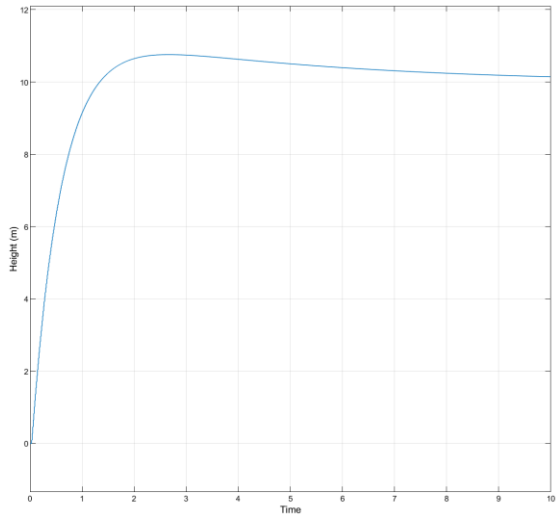


Şekil 4.19 : %100 parametre belirsizliği, 50 metre referans irtifa

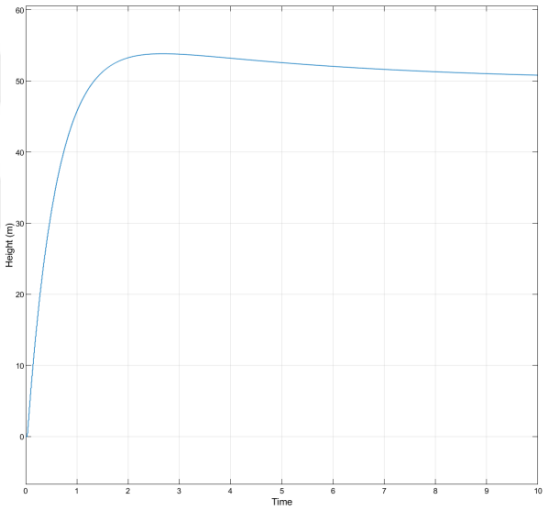
Parametre belirsizliği %150 olarak değiştirilmiştir ve Quadrotorun sırasıyla 1 – 10 ve 50 metre irtifalarında referans takibi performansı Şekil 4.20, Şekil 4.21 ve Şekil 4.22'de gösterilmiştir.



Şekil 4.20 : %150 parametre belirsizliği, 1 metre referans irtifa

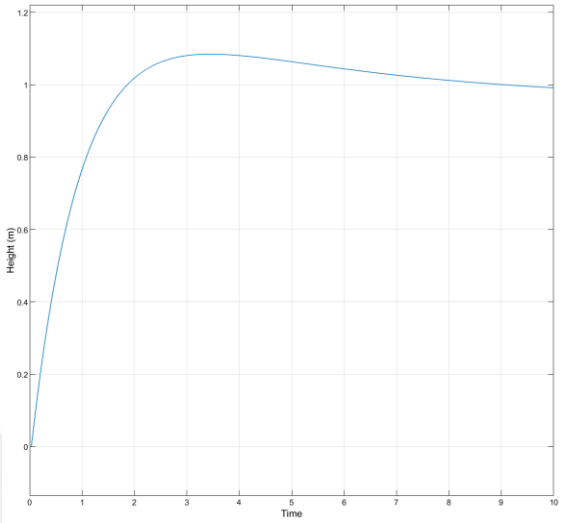


Şekil 4.21 : %150 parametre belirsizliği, 10 metre referans irtifa

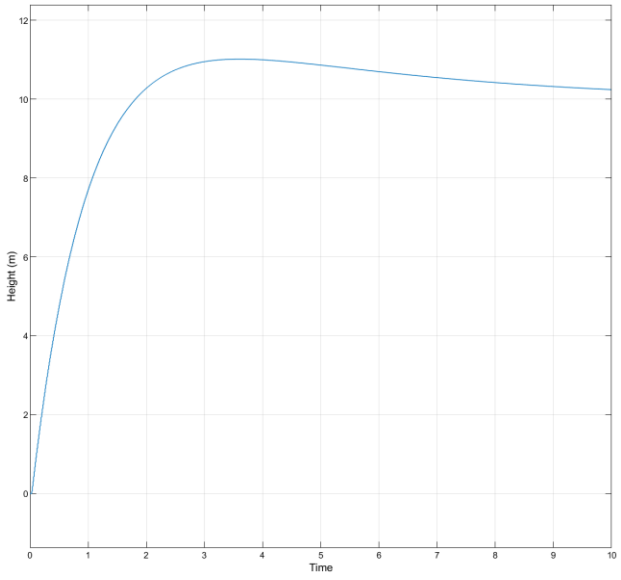


Şekil 4.22 : %150 parametre belirsizliği, 50 metre referans irtifa

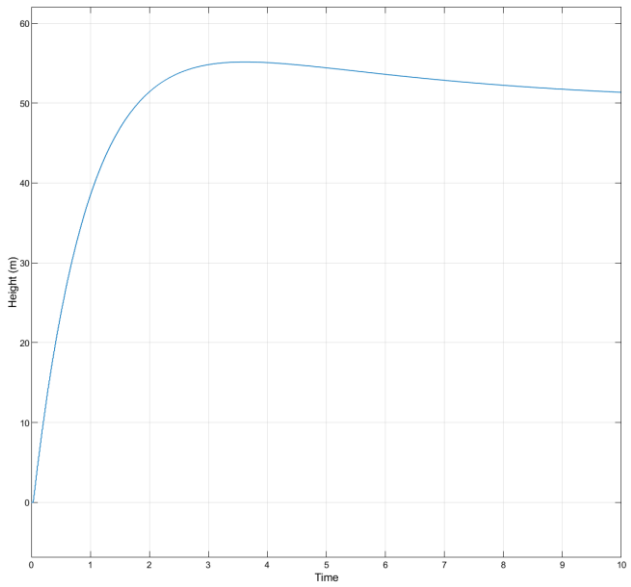
Parametre belirsizliği %200 olarak değiştirilmiştir ve Quadrotorun sırasıyla 1 – 10 ve 50 metre irtifalarında referans takibi performansı Şekil 4.23, Şekil 4.24 ve Şekil 4.25'de gösterilmiştir.



Şekil 4.23 : %200 parametre belirsizliği, 1 metre referans irtifa



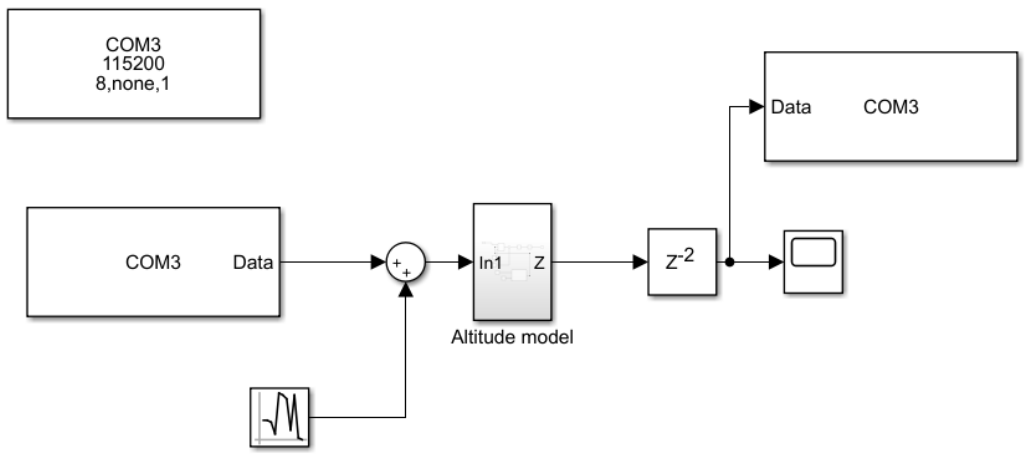
Şekil 4.24 : %200 parametre belirsizliği, 10 metre referans irtifa



Şekil 4.25 : %200 parametre belirsizliği, 50 metre referans irtifa

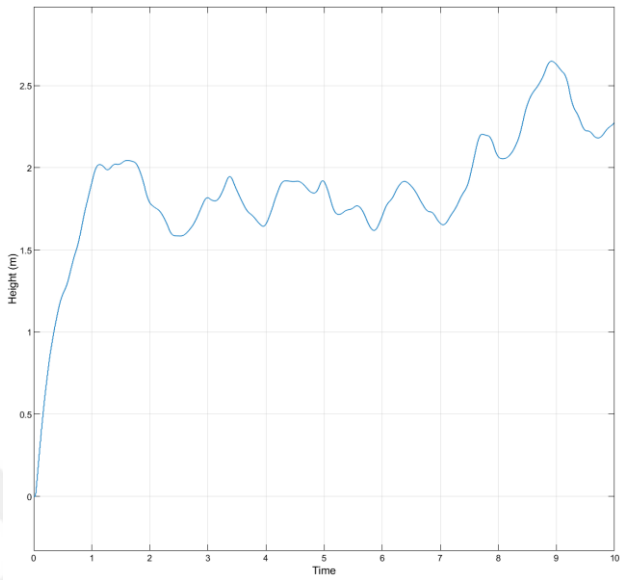
4.5.6. Modele Gürültü Eklenmesi

Kontrolcünün giriş gürültülü ortamda testini gerçekleştirmek için Şekil 4.26'da bulunan model oluşturulmuştur. Giriş gürültüsü olarak ortalaması 0.1 ve varyansı 1 olan Gaussian Noise kullanılmıştır.



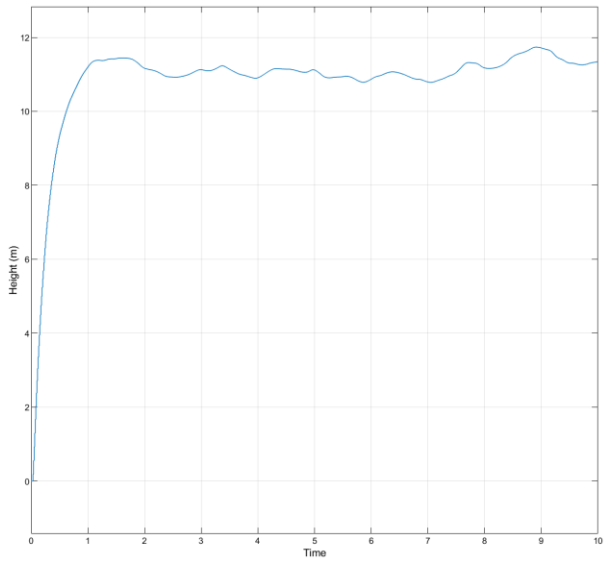
Şekil 4.26 : Modele giriş gürültüsü eklenmesi

FPGA’e irtifa referansı 1 metre olarak verilmiştir ve referans takibi Şekil 4.27’de gösterilmiştir. Giriş gürültüsüne rağmen 1 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



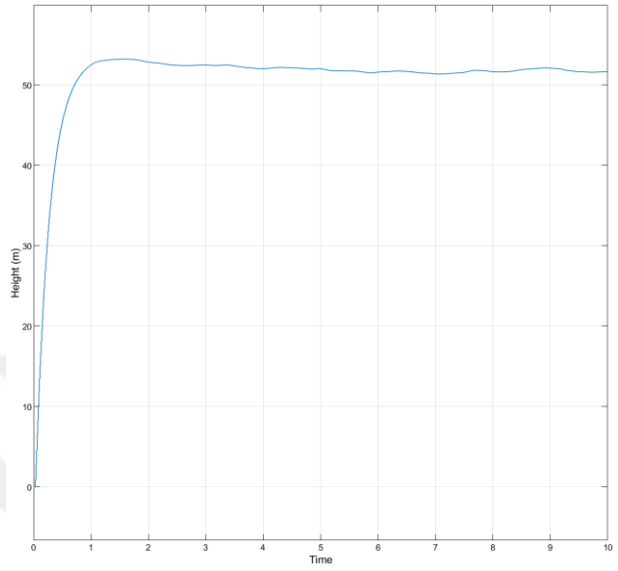
Şekil 4.27 : Giriş gürültüsü bulunan modelde referans takibi 1 metre

FPGA’e verilen irtifa referansı 10 metreye çıkarılmıştır ve referans takibi Şekil 4.28’de gösterilmiştir. Giriş gürültüsüne rağmen 10 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



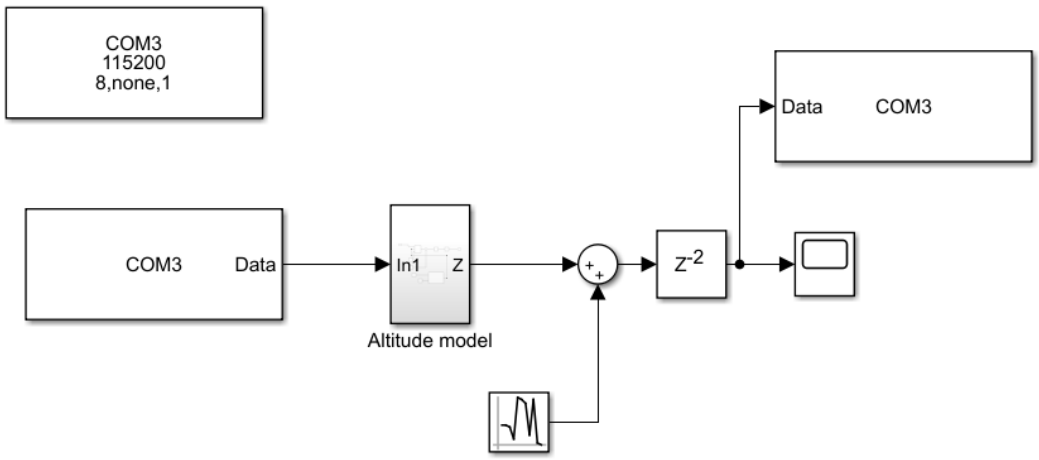
Şekil 4.28 : Giriş gürültüsü bulunan modelde referans takibi 10 metre

FPGA’e verilen irtifa referansı 50 metreye çıkarılmıştır ve referans takibi Şekil 4.29’da gösterilmiştir. Giriş gürültüsüne rağmen 50 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



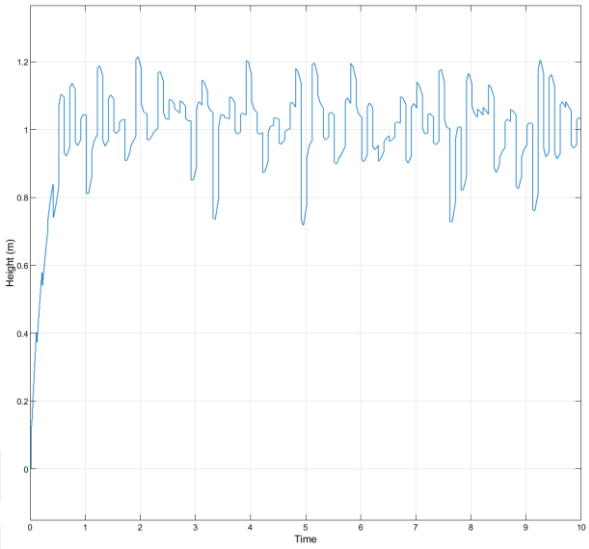
Şekil 4.29 : Giriş gürültüsü bulunan modelde referans takibi 50 metre

Kontrolcünün çıkış gürültülü ortamda testini gerçekleştirmek için Şekil 4.30‘da bulunan model oluşturulmuştur. Çıkış gürültüsü olarak ortalaması 0.01 ve varyansı 0.1 olan Gaussian Noise kullanılmıştır.



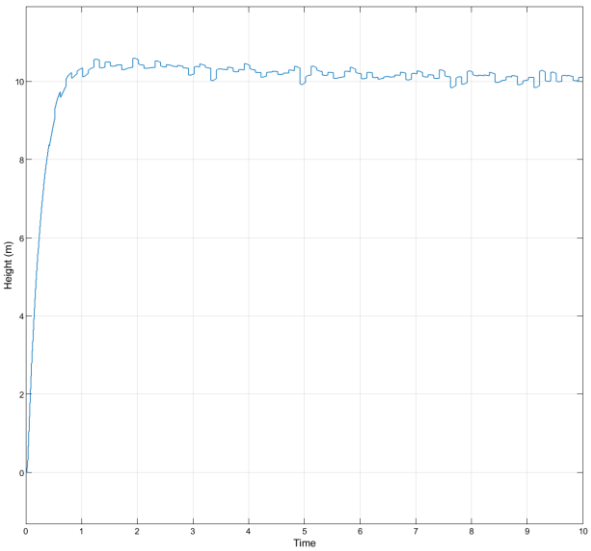
Şekil 4.30 : Modele çıkış gürültüsü eklenmesi

FPGA’e irtifa referansı 1 metre olarak verilmiştir ve referans takibi Şekil 4.31’de gösterilmiştir. Çıkış gürültüsünden sistem daha fazla etkilenmesine rağmen 1 metre irtifada referans takibinin belirli ölçüde sağlandığı görülmüştür.



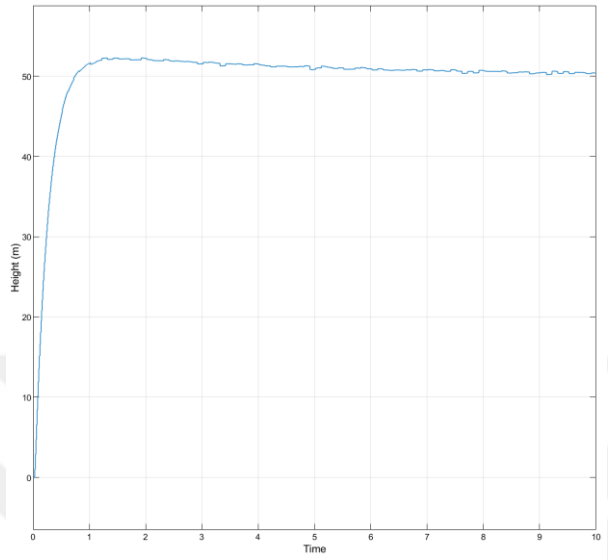
Şekil 4.31 : Çıkış gürültüsü bulunan modelde referans takibi 1 metre

FPGA’e verilen irtifa referansı 10 metreye çıkarılmıştır ve referans takibi Şekil 4.32’de gösterilmiştir. Çıkış gürültüsünden sistem daha fazla etkilenmesine rağmen 10 metre irtifada referans takibinin belirli ölçüde sağlandığı görülmüştür.



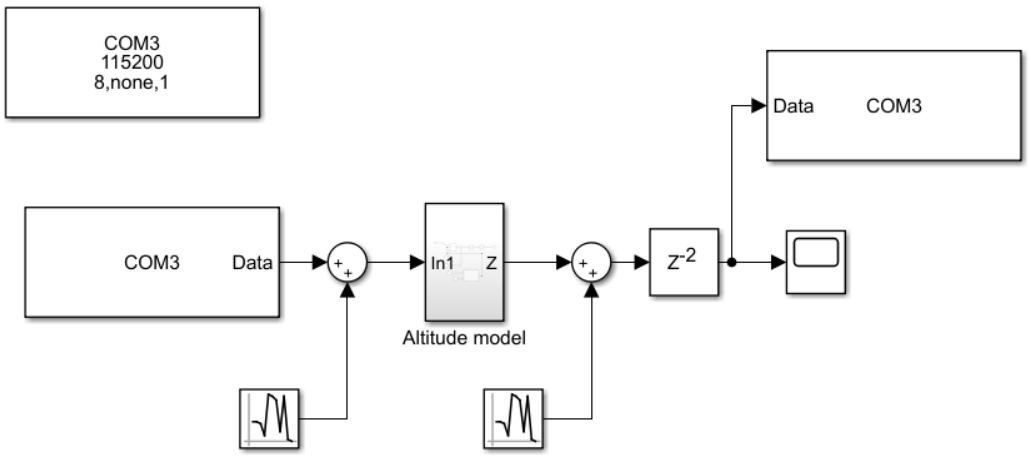
Şekil 4.32 : Çıkış gürültüsü bulunan modelde referans takibi 10 metre

FPGA'e verilen irtifa referansı 50 metreye çıkarılmıştır ve referans takibi Şekil 4.33'da gösterilmiştir. Çıkış gürültüsünden sistem daha fazla etkilenmesine rağmen 50 metre irtifada referans takibinin belirli ölçüde sağlandığı görülmüştür.



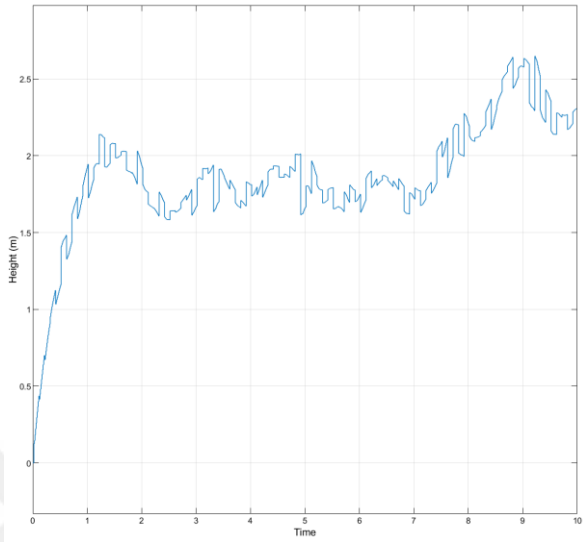
Şekil 4.33 : Çıkış gürültüsü bulunan modelde referans takibi 50 metre

Kontrolcünün gürültülü ortamda testini gerçekleştirmek için Şekil 4.34'da bulunan model oluşturulmuştur. Giriş gürültüsü olarak ortalaması 0.1 ve varyansı 1 olan Gaussian Noise kullanılmıştır. Çıkış gürültüsü olarak ortalaması 0.01 ve varyansı 0.1 olan Gaussian Noise kullanılmıştır.



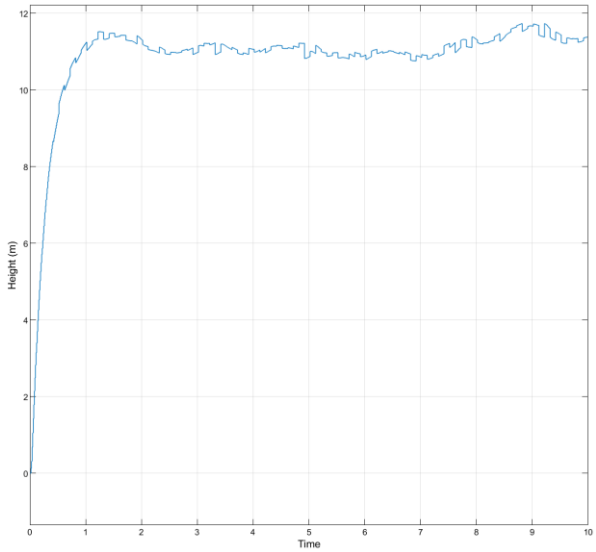
Şekil 4.34 : Modele giriş ve çıkış gürültüsü eklenmesi

FPGA’e irtifa referansı 1 metre olarak verilmiştir ve referans takibi Şekil 4.35’de gösterilmiştir. Giriş ve çıkış gürültüsü altında sistemin daha fazla etkilendiği ancak 1 metrede referans takibinin sağlandığı görülmüştür.



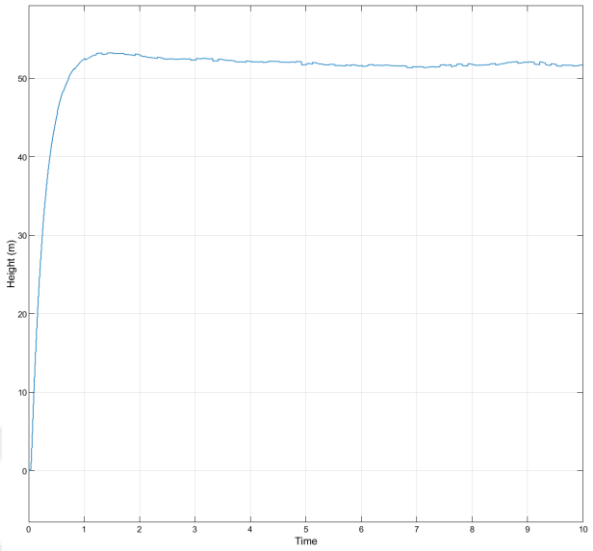
Şekil 4.35 : Giriş ve çıkış gürültüsü bulunan modelde referans takibi 1 metre

FPGA’e verilen irtifa referansı 50 metreye çıkarılmıştır ve referans takibi Şekil 4.36’de gösterilmiştir. Giriş ve çıkış gürültüsü altında sistemin daha fazla etkilendiği ancak 10 metrede referans takibinin sağlandığı görülmüştür.



Şekil 4.36 : Giriş ve çıkış gürültüsü bulunan modelde referans takibi 10 metre

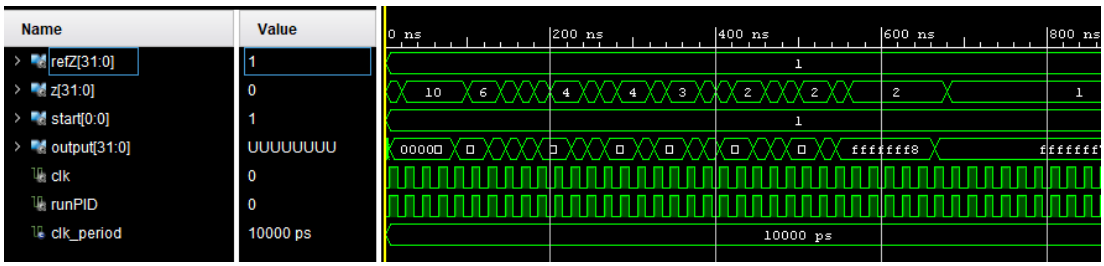
FPGA’e verilen irtifa referansı 50 metreye çıkarılmıştır ve referans takibi Şekil 4.37’te gösterilmiştir. Giriş ve çıkış gürültüsü altında sistemin daha fazla etkilendiği ancak 50 metrede referans takibinin sağlandığı görülmüştür.



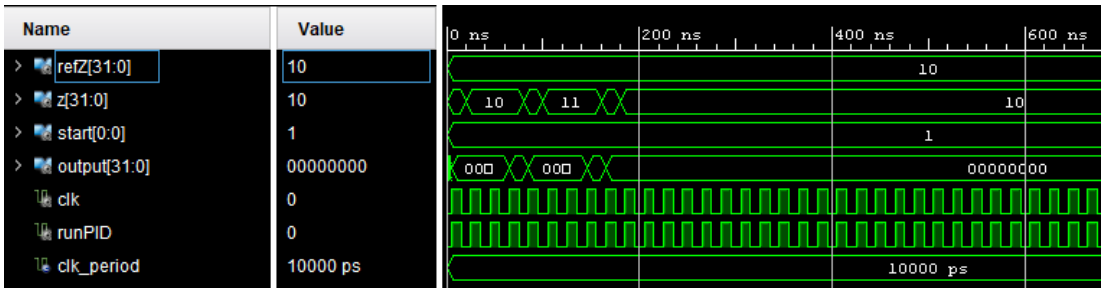
Şekil 4.37 : Giriş ve çıkış gürültüsü bulunan modelde referans takibi 50 metre

4.6. Sayısal İrtifa Kontrolcünün Oluşturulması

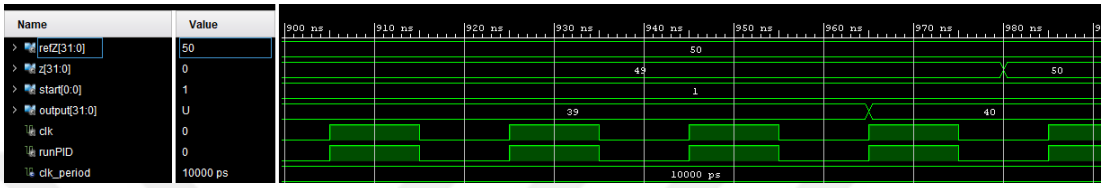
Bu yöntemde ise irtifa kontrolcü doğrudan VHDL ile oluşturulmuştur. Sayısal kontrolcünün performansı farklı referans irtifalar ile Vivado simülatör kullanılarak test edilmiştir. (Şekil 4.38, Şekil 4.39, Şekil 4.40)



Şekil 4.38 : Sayısal irtifa kontrolcü performansı 1 metre referans

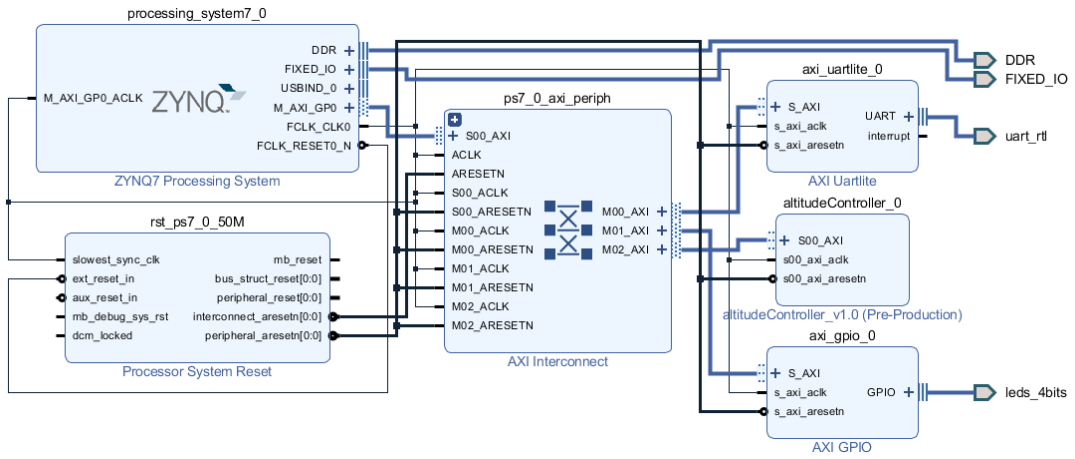


Şekil 4.39 : Sayısal irtifa kontrolcü performansı 10 metre referans



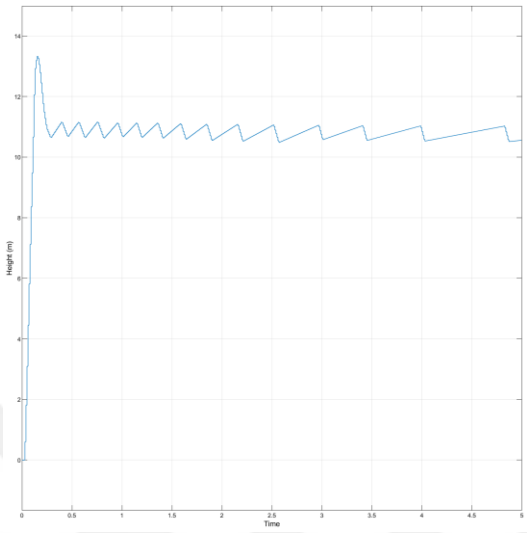
Şekil 4.40 : Sayısal irtifa kontrolcü performansı 50 metre referans

Vivado ortamında simülasyonları yapılan sayısal irtifa kontrolcü AXI arayüzü oluşturularak IP haline getirilmiştir. Daha sonra Zynq ile beraber entegrasyonu yapılarak nihai test düzeneği oluşturulmuştur. (Şekil 4.41)



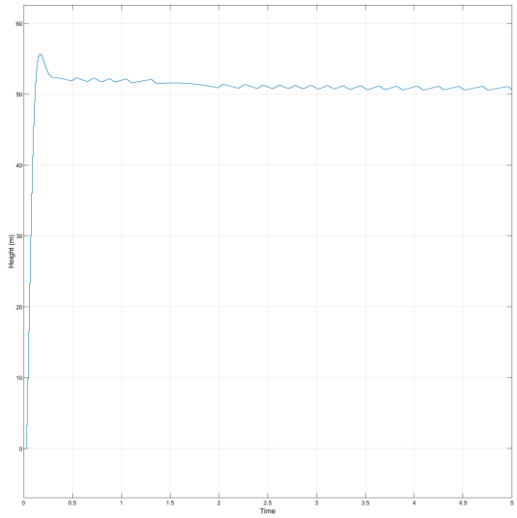
Şekil 4.41 : İrtifa kontrolcü – Zynq test düzeneği

Zynq üzerinden sayısal kontrolcüye irtifa referansı 10 metre olarak verilmiştir ve referans takibi Şekil 4.42‘de gösterilmiştir. 10 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



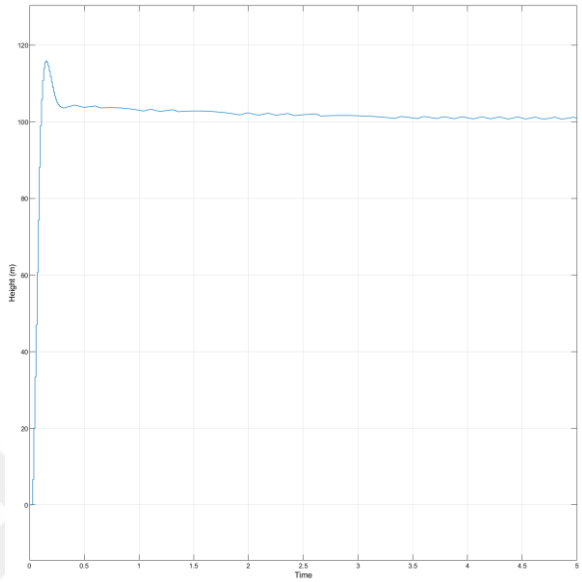
Şekil 4.42 : Sayısal irtifa kontrolcü 10 metre irtifa performansı

Zynq üzerinden sayısal kontrolcüye verilen irtifa referansı 50 metre olarak değiştirilmiş ve referans takibi Şekil 4.43‘de gösterilmiştir. 50 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



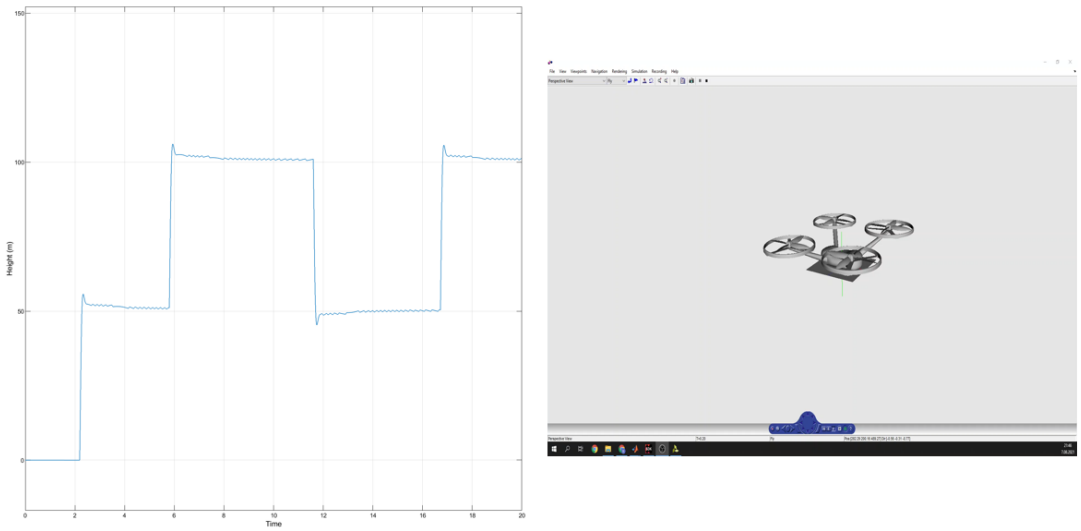
Şekil 4.43 : Sayısal irtifa kontrolcü 50 metre irtifa performansı

Zynq üzerinden sayısal kontrolcüye verilen irtifa referansı 100 metre olarak değiştirilmiş ve referans takibi Şekil 4.44‘de gösterilmiştir. 100 metre irtifada sistemin başarılı şekilde referans değerini takip ettiği görülmüştür.



Şekil 4.44 : Sayısal irtifa kontrolcü 100 metre irtifa performansı

Butonlar aracılığı ile sistem çalışırken irtifa değerleri güncellenebilmektedir. Ayrıca VR sink aracılığı ile sistemin davranışı gösterilmektedir (Şekil 4.45Şekil 4.16).

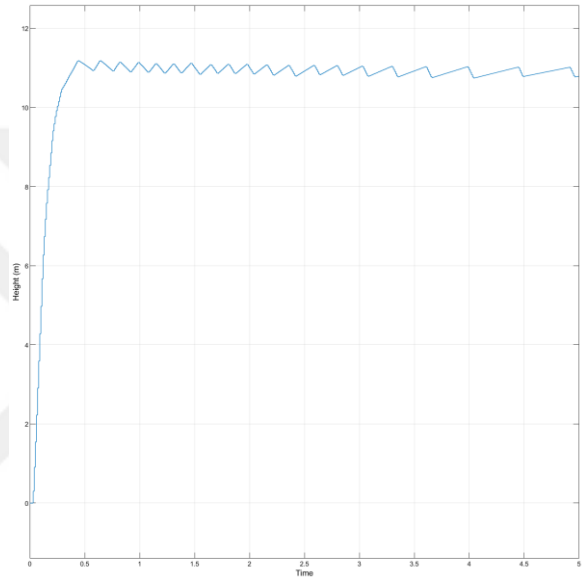


Şekil 4.45 : Buton değerleri ile sisteme irtifa değerlerinin girilmesi

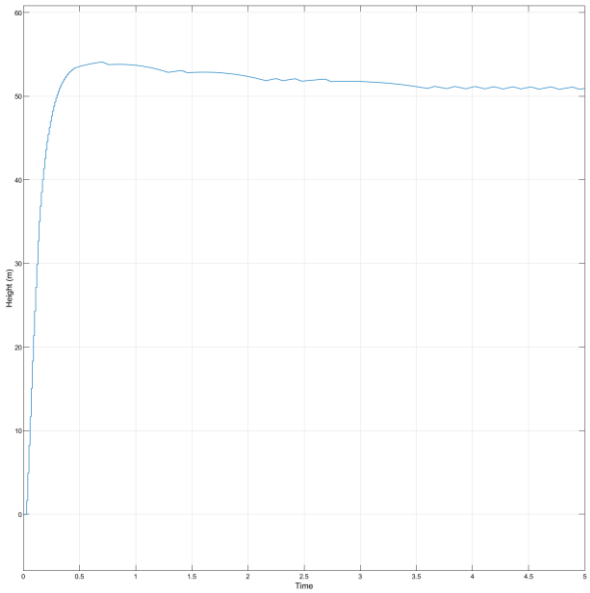
4.6.1. Modele Parametre Belirsizliği Eklenmesi

Kontrolcünün değişen yüke karşı dayanıklı olduğunu test etmek için. Quadrotorun ağırlığına sırasıyla %100, %150 ve %200 oranlarında parameter belirsizliği uygulanmış ve farklı irtifalarda referans takibi performansları incelenmiştir.

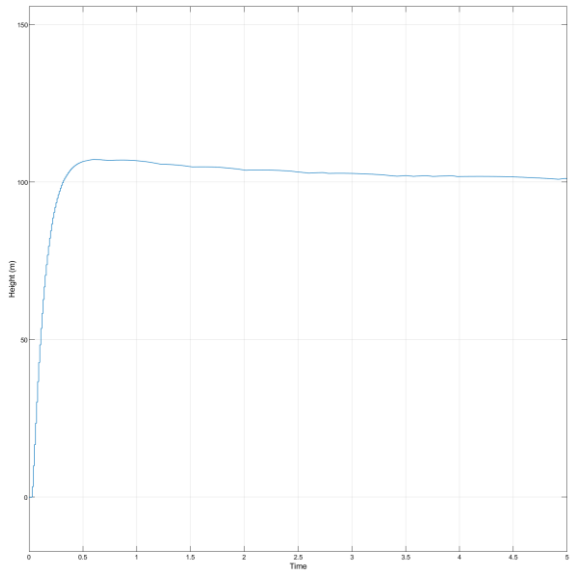
Parametre belirsizliği %100 olarak belirlenmiş ve Quadrotorun sırasıyla 10, 50 ve 100 metre irtifalarında referans takibi performansı Şekil 4.46, Şekil 4.47, Şekil 4.48'da gösterilmiştir.



Şekil 4.46 : %100 parametre belirsizliği, 10 metre referans irtifa

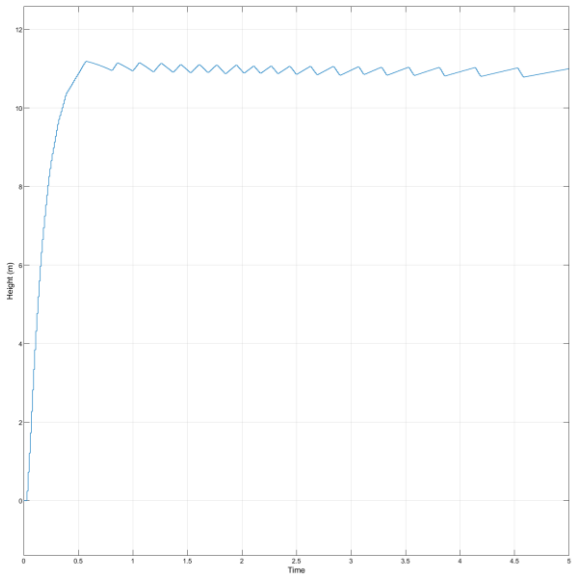


Şekil 4.47 : %100 parametre belirsizliği, 50 metre referans irtifa

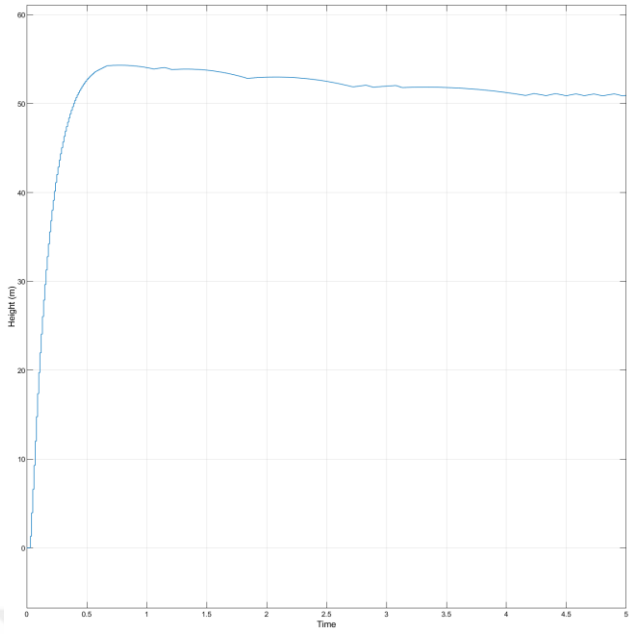


Şekil 4.48 : %100 parametre belirsizliği, 100 metre referans irtifa

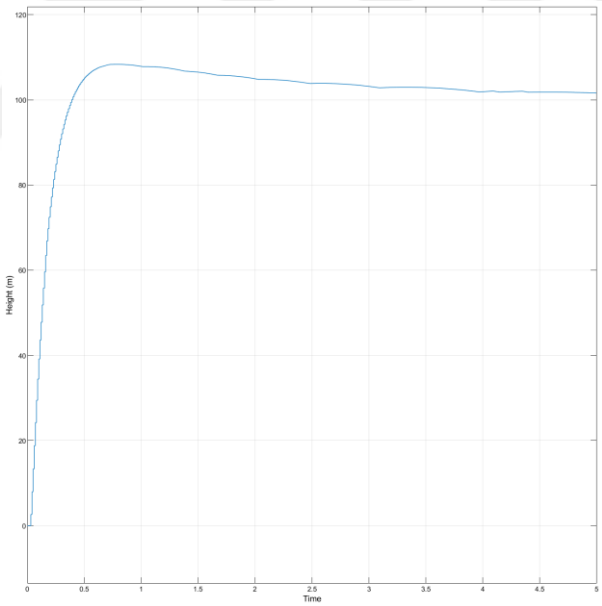
Parametre belirsizliği %150 olarak değiştirilmiş ve Quadrotorun sırasıyla 10, 50 ve 100 metre irtifalarında referans takibi performansı Şekil 4.49, Şekil 4.50, Şekil 4.51'da gösterilmiştir.



Şekil 4.49 : %150 parametre belirsizliği, 10 metre referans irtifa

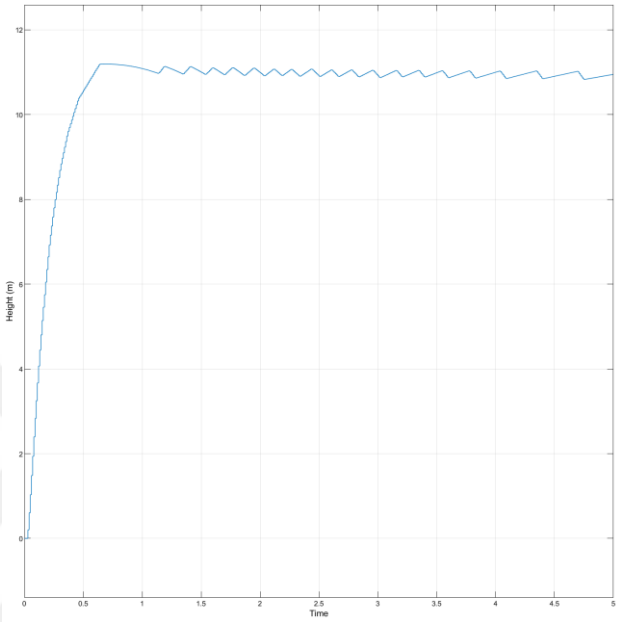


Şekil 4.50 : %150 parametre belirsizliği, 50 metre referans irtifa

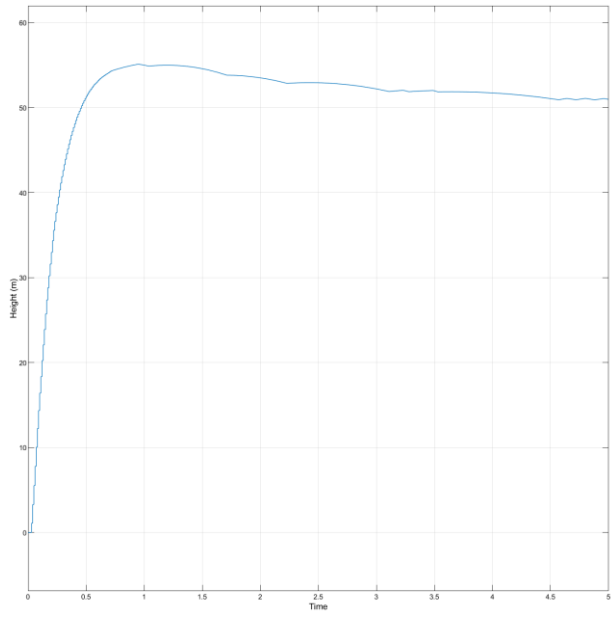


Şekil 4.51 : %150 parametre belirsizliği, 100 metre referans irtifa

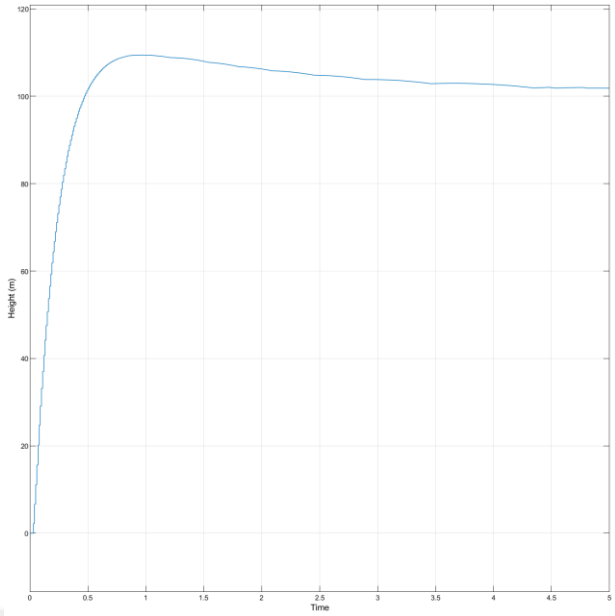
Parametre belirsizliği %200 olarak değiştirilmiş ve Quadrotorun sırasıyla 10, 50 ve 100 metre irtifalarında referans takibi performansı Şekil 4.52, Şekil 4.53, Şekil 4.54'da gösterilmiştir.



Şekil 4.52 : %200 parametre belirsizliği, 10 metre referans irtifa



Şekil 4.53 : %200 parametre belirsizliği, 50 metre referans irtifa

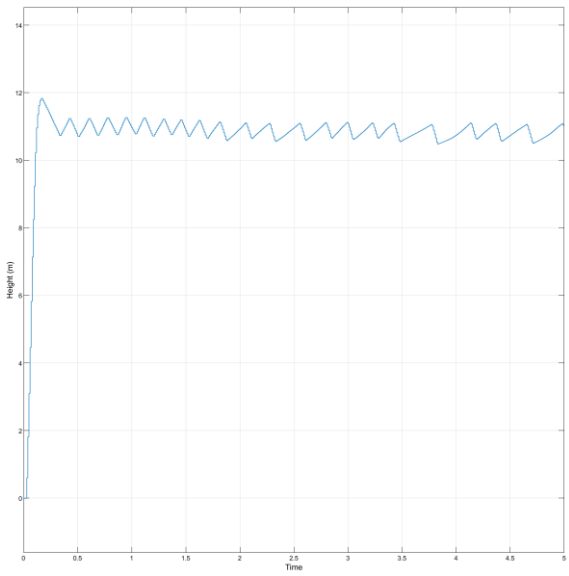


Şekil 4.54 : %200 parametre belirsizliği, 100 metre referans irtifa

4.6.2. Modele Gürültü Eklenmesi

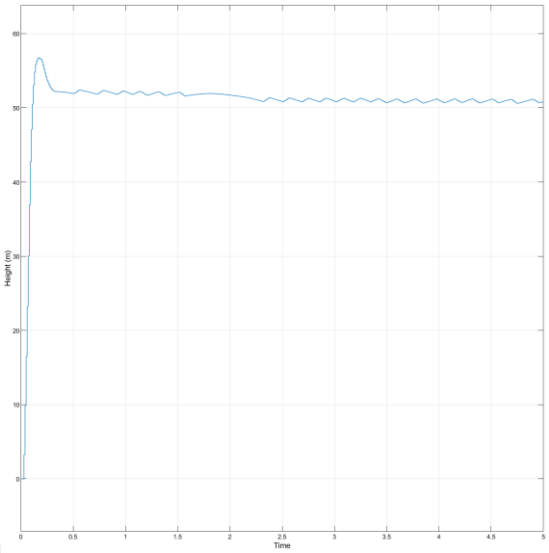
Kontrolcünün giriş gürültülü ortamda testini gerçekleştirmek için giriş gürültüsü olarak ortalaması 0.1 ve varyansı 1 olan Gaussian Noise kullanılmıştır.

FPGA'e verilen irtifa referansı 10 metreye çıkarılmıştır ve referans takibi Şekil 4.55'de gösterilmiştir. 10 metre irtifada referans takibinin sağlandığı görülmüştür.



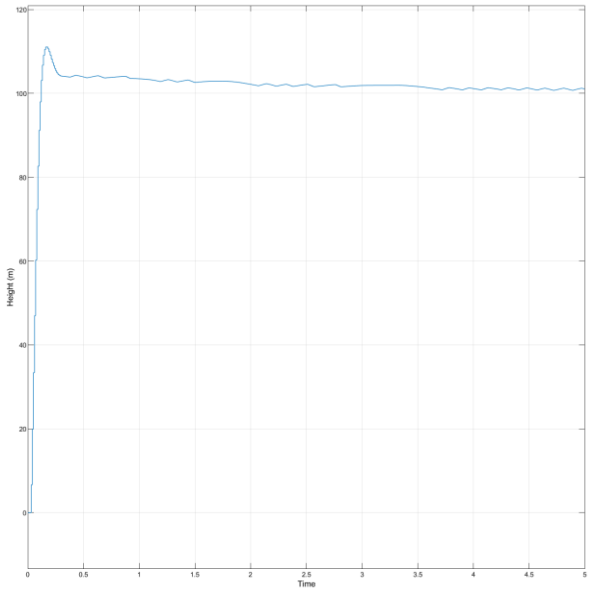
Şekil 4.55 : Giriş gürültüsü bulunan modelde referans takibi 10 metre

FPGA’e verilen irtifa referansı 50 metreye çıkarılmıştır ve referans takibi Şekil 4.56‘da gösterilmiştir. 50 metre irtifada referans takibinin sağlandığı görülmüştür.



Şekil 4.56 : Giriş gürültüsü bulunan modelde referans takibi 50 metre

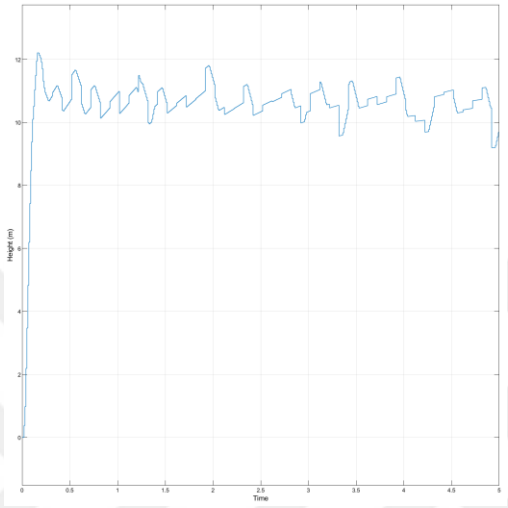
FPGA’e verilen irtifa referansı 100 metreye çıkarılmıştır ve referans takibi Şekil 4.57‘da gösterilmiştir. 100 metre irtifada referans takibinin sağlandığı görülmüştür.



Şekil 4.57 : Giriş gürültüsü bulunan modelde referans takibi 100 metre

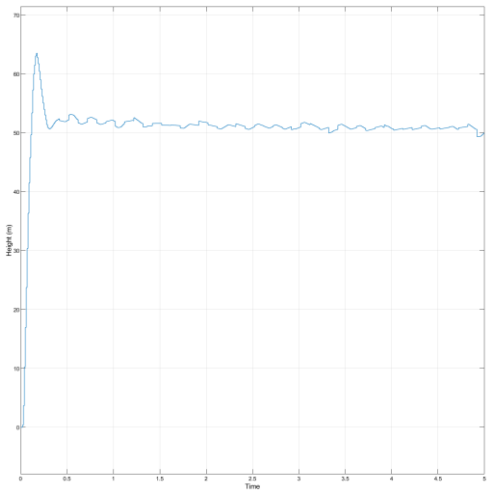
Kontrolcünün çıkış gürültülü ortamda testini gerçekleştirmek için çıkış gürültüsü olarak ortalaması 0.01 ve varyansı 0.1 olan Gaussian Noise kullanılmıştır.

FPGA'e irtifa referansı 10 metre olarak verilmiştir ve referans takibi Şekil 4.58'de gösterilmiştir. Çıkış gürültüsünden sistem daha fazla etkilenmesine rağmen 10 metre irtifada referans takibinin belirli ölçüde sağlandığı görülmüştür.



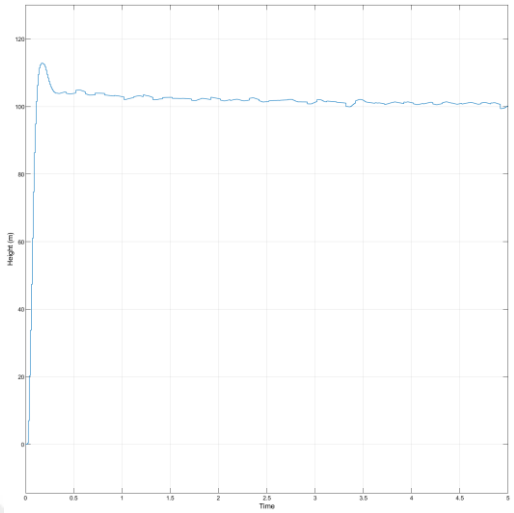
Şekil 4.58 : Çıkış gürültüsü bulunan modelde referans takibi 10 metre

FPGA'e irtifa referansı 50 metre olarak verilmiştir ve referans takibi Şekil 4.59'da gösterilmiştir. Çıkış gürültüsünden sistem daha fazla etkilenmesine rağmen 50 metre irtifada referans takibinin belirli ölçüde sağlandığı görülmüştür.



Şekil 4.59 : Çıkış gürültüsü bulunan modelde referans takibi 50 metre

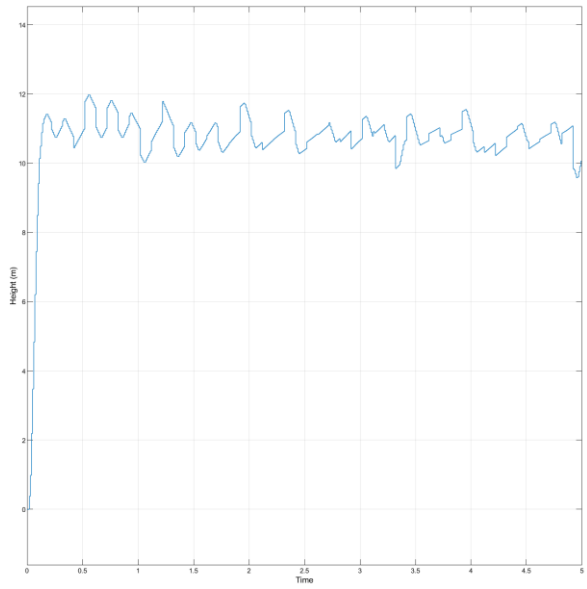
FPGA’e irtifa referansı 100 metre olarak verilmiştir ve referans takibi Şekil 4.60’da gösterilmiştir. Çıkış gürültüsünden sistem daha fazla etkilenmesine rağmen 100 metre irtifada referans takibinin belirli ölçüde sağlandığı görülmüştür.



Şekil 4.60 : Çıkış gürültüsü bulunan modelde referans takibi 100 metre

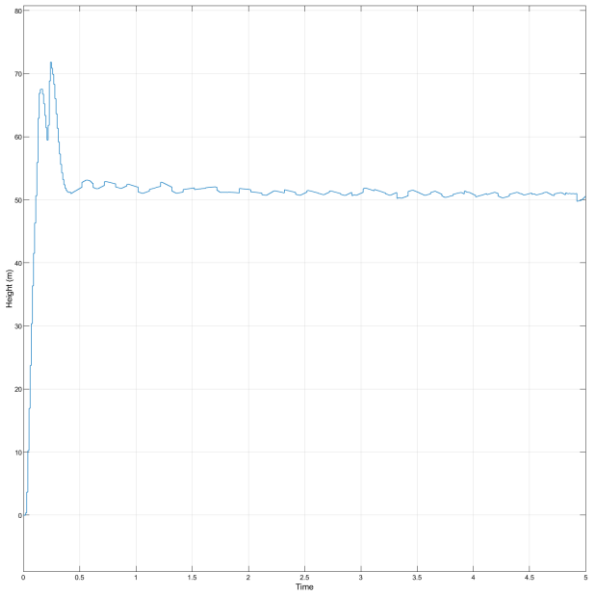
Kontrolcünün giriş ve çıkış gürültüsü bulunan modelde testini gerçekleştirmek için giriş gürültüsü olarak ortalaması 0.1 ve varyansı 1 olan Gaussian Noise modele eklenmiştir. Çıkış gürültüsü olarak ortalaması 0.01 ve varyansı 0.1 olan Gaussian Noise modele eklenmiştir.

FPGA’e irtifa referansı 10 metre olarak verilmiştir ve referans takibi Şekil 4.61’de gösterilmiştir. Giriş ve çıkış gürültüsü altında sistemin daha fazla etkilendiği ancak 10 metrede referans takibinin sağlandığı görülmüştür.



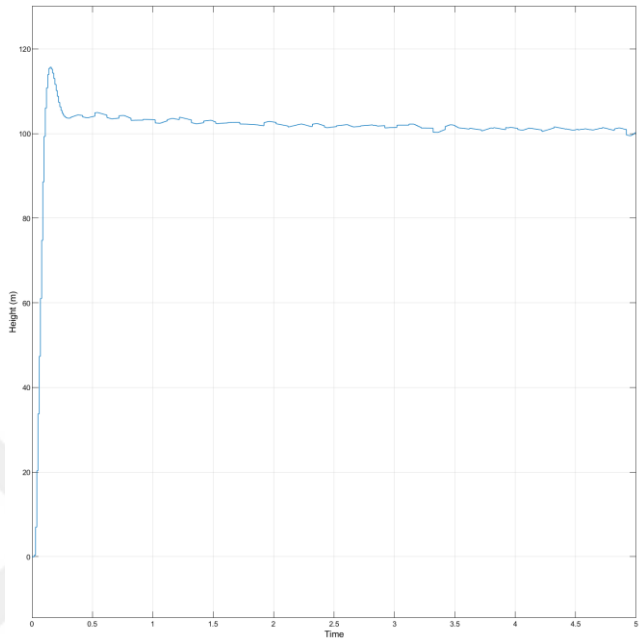
Şekil 4.61 : Giriş- çıkış gürültüsü bulunan modelde referans takibi 10 metre

FPGA'e irtifa referansı 50 metre olarak verilmiştir ve referans takibi Şekil 4.62'de gösterilmiştir. Giriş ve çıkış gürültüsü altında sistemin daha fazla etkilendiği ancak 50 metrede referans takibinin sağlandığı görülmüştür.



Şekil 4.62 : Giriş- çıkış gürültüsü bulunan modelde referans takibi 50 metre

FPGA’e irtifa referansı 100 metre olarak verilmiştir ve referans takibi Şekil 4.63’de gösterilmiştir. Giriş ve çıkış gürültüsü altında sistemin daha fazla etkilendiği ancak 100 metrede referans takibinin sağlandığı görülmüştür.



Şekil 4.63 : Giriş- çıkış gürültüsü bulunan modelde referans takibi 100 metre

5. SONUÇ VE ÖNERİLER

Tez kapsamında farklı yöntemler ve yazılımlar kullanılarak FPGA üzerinde çalışan irtifa kontrolcüler oluşturulmuştur. Bu yöntemlerin hepsinde referans takibi sağlanmasına rağmen FPGA kaynak kullanımı ve pratiklik açısından aralarında farklar olduğu görülmüştür.

Xilinx System Generator, hazır FPGA blokları ile FPGA tasarımını kolaylaştırmaktadır ancak karmaşık kodlar üretmektedir ve sentezleyicisi lisanslıdır.

HDL coder tasarım kolaylığı sağlamaktadır ancak karmaşık HDL kodları üretmektedir. Bu sebeple sentezlenen kodların üzerinde değişiklik yapmanın zor olduğu görülmüştür. HDL coder ile üretilmiş olan irtifa kontrolcü tasarımının farklı sentez stratejileri ile kaynak kullanımı Çizelge 5.1, Çizelge 5.2, Çizelge 5.3'te gösterilmiştir. Alan optimizasyonu ile en optimal kaynak kullanımı sağlandığı görülmüştür.

Çizelge 5.1 : FPGA kaynak kullanımı (Alan optimizasyonu)

	LUT	**FF**	**BRAM**	**URAM**	**DSP**
Değerler	576	0	0	0	4

Çizelge 5.2 : FPGA kaynak kullanımı (Performans optimizasyonu)

	LUT	**FF**	**BRAM**	**URAM**	**DSP**
Değerler	585	0	0	0	4

Çizelge 5.3 : FPGA kaynak kullanımı (Çalışma süresi optimizasyonu)

	LUT	**FF**	**BRAM**	**URAM**	**DSP**
Değerler	713	0	0	0	4

HDL coder'da olduğu gibi HLS kullanarak yapılan irtifa kontrolcü tasarımın ve testlerinin daha kolay olduğu görülmüştür. Ancak HLS, HDL coder'da da olduğu gibi karmaşık kodlar üretmektedir. Bu da kod üzerinde daha sonra yapılabilecek güncellemeleri zorlaştırmaktadır. Ayrıca kaynak kullanımının implementasyonu yapılan diğer yöntemlere göre fazla olduğu görülmüştür. Vivado HLS ile yapılan tasarımın kaynak kullanımı Çizelge 5.4'de gösterilmiştir.

Çizelge 5.4 : Vivado HLS ile FPGA kaynak kullanımı

	LUT	**FF**	**BRAM**	**URAM**	**DSP**
Değerler	2767	1611	0	0	8

FPGA üzerinde MicroBlaze kullanmak C/C++ dilleri ile tasarım imkânı sağlamaktadır ve bu sebeple tasarımı hızlandırdığı görülmüştür. Ayrıca MicroBlaze ile tasarıma farklı IP'ler rahatlıkla eklenebilmekte bu da tasarım akışını kolaylaştırmaktadır. Ancak sertifikasyon isteyen projelerde MicroBlaze kullanımı yüksek maliyetler getirebilmektedir. MicroBlaze kullanılarak gerçekleştirilen irtifa kontrolcünün ve HIL test düzeneğinin sentez sonrası zamanlama sonuçları, kaynak kullanımı ve güç tüketimi Çizelge 5.5, Çizelge 5.6, Çizelge 5.7'te gösterilmiştir.

Çizelge 5.5 : MicroBlaze tasarımı timing sonuçları

Setup		**Hold**		**Pulse Width**	
Worst Negative Slack	3.016 ns	**Worst Hold Slack**	0.007 ns	**Worst Pulse Width Slack**	2.00 ns
Total Negative Slack	0 ns	**Total Hold Slack**	0 ns	**Total Pulse Witdh Negative Slack**	0 ns

Number of Failing Endpoint	0	**Number of Failing Endpoint**	0	**Number of Failing Endpoint**	0
Total Number of Endpoints	8469	**Total Number of Endpoints**	8469	**Total Number of Endpoints**	3793

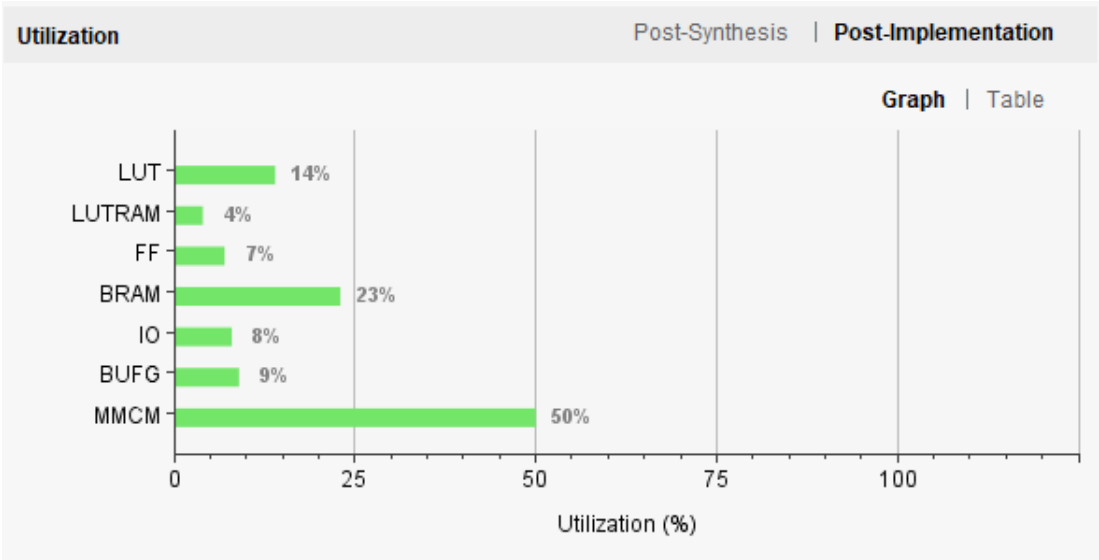
Çizelge 5.6 : MicroBlaze tasarımı güç tüketimi

	Static Güç Tüketimi	**Dinamik Güç Tüketimi**	**Toplam Güç Tüketimi**
Değerler	0.096 W	0.194 W	0.29 W

Çizelge 5.7 : FPGA kaynak kullanımı (Default sentez)

İsim	**Değerler**	**İsim**	**Değerler**
Slice LUTs	2491	**Bonded IOB**	8
Slice Registers	2290	**BUFGCTRL**	3
Mux	114		
BRAM	14		

MicroBlaze ve UART IP içeren tasarımın sentez ve implementasyon sonucu FPGA kaynak tüketim oranı Resim 5.1'de gösterilmiştir.



Resim 5.1 : MicroBlaze test sistemi FPGA kaynak tüketimi

Zynq ve sayısal irtifa kontrolcü IP'sinin kullanımı referans takibi ve performans açısından daha iyi sonuçlar vermektedir. Sayısal kontrolcünün kullanımı oturma zamanını düşürmektedir. İrtifa kontrolcü IP'sinin MicroBlaze içeren tasarıma göre güç tüketiminin 5 kattan fazla olduğu görülmüştür (Çizelge 5.9). İrtifa kontrolcü IP'si ile kaynak tüketimi MicroBlaze'e göre azalmıştır (Resim 5.2).

Çizelge 5.8 : ZYNQ – sayısal irtifa kontrolcü timing sonuçları

Setup		**Hold**		**Pulse Width**	
Worst Negative Slack	0.617 ns	**Worst Hold Slack**	0.007 ns	**Worst Pulse Width Slack**	9.02 ns
Total Negative Slack	0 ns	**Total Hold Slack**	0 ns	**Total Pulse Witdh Negative Slack**	0 ns
Number of Failing Endpoint	0	**Number of Failing Endpoint**	0	**Number of Failing Endpoint**	0
Total Number of Endpoints	3185	**Total Number of Endpoints**	3185	**Total Number of Endpoints**	1478

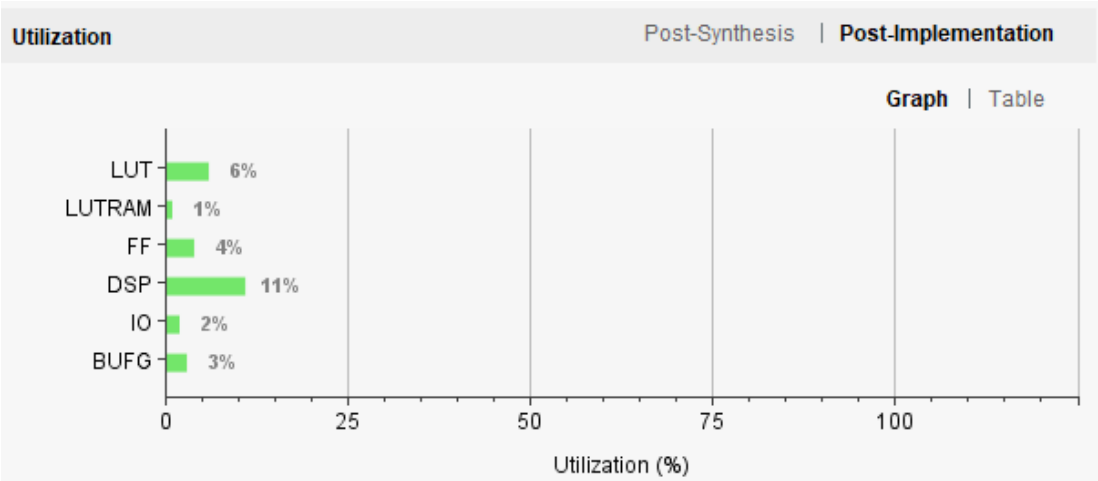
Çizelge 5.9 : ZYNQ – sayısal irtifa kontrolcü güç tüketimi

	Static Güç Tüketimi	**Dinamik Güç Tüketimi**	**Toplam Güç Tüketimi**
Değerler	0.120 W	1.408 W	1.528 W

Çizelge 5.10 : ZYNQ – sayısal irtifa kontrolcü kaynak kullanımı (Default sentez)

İsim	**Değerler**	**İsim**	**Değerler**
Slice LUTs	1064	**Bonded IOB**	130
Slice Registers	1350	**BUFGCTRL**	1

Zynq, sayısal irtifa kontrolcü IP'si, UART IP ve GPIO içeren tasarımın sentez ve zaman optimizasyonu ile implementasyonu sonucu FPGA kaynak tüketim oranı Resim 5.2'de gösterilmiştir.



Resim 5.2 : ZYNQ – sayısal irtifa kontrolcü kaynak tüketimi

Tasarımların FPGA üzerinde kapladığı alanlar sanal işlemci ve irtifa kontrolcü IP'si ile %14'ü geçmediği görülmüştür. Böylelikle FPGA üzerinde yalpa, yunuslama ve sapma kontrolcülerinin paralel bir şekilde gerçeklenmesi için gerekli kaynak kalmıştır.

Bu tez kapsamında gösterildiği gibi quadrotorların kontrol işlemleri FPGA üzerinde gerçekleştirilebilmektedir. FPGA'ler ayrıca görüntü verisi işlemede de yoğun olarak kullanılmaktadır. Hedef takip ve quadrotor kontrolü içeren gelecek uygulamalarda bu tezde gösterilen quadrotor kontrolcüler kullanılabilir [49].

KAYNAKLAR

[1] **Pierpaolo M., Samir B.** (2004). Design and Control of an Indoor Micro Quadrotor, IEEE International Conference on Robotics and Automation, New Orleans, USA, 26 April – 1 May

[2] **Övünç E.** (2013). Dört Rotorlu İnsansız Hava Aracı İçin Otopilot Tasarımı, Yüksek Lisans Tezi, TOBB Ekonomi ve Teknoloji Üniversitesi, Ankara, Türkiye

[3] **Ho-chung C., Ta-ming S.** (2009). Dört Rotorlu İnsansız Hava Aracı İçin Otopilot Tasarımı, IEEE International Symposium on Computational Intelligence in Robotics and Automation, Daejeon, South Korea, 15-18 December

[4] **Paul M., Shweta G.** (2012). A Surver of Quadrotor Unmanned Aerial Vehicles, Prooceeding of IEEE Southeastcon, Orlando, USA, 15-18 March

[5] **Kyungnam L., Youngmin K.** (2018). Design and Analysis of Digital PID Controller in MCU and FPGA, International SoC Design Conference, Daegu, South Korea, 12-15 November

[6] **George V., Kimon V.** (2015). Handbook of Unmanned Aerial Vehicles, Aviation History and Unmanned Flight, Sayfa 57

[7] **Richardson P.** (2018). Leonardo Da Vinci's Discovery of The Dynamic Soaring by Birds in Wind Shear, Royal Society

[8] **Carrillo L., Lopez A.** (2011). Quad-Rotor Control Vision-Based Hovering and Navigation, Brief History

[9] **Clark R.** (2000). Uninhabited Combat Aerial Vehicles; Airpower by the People, for the People But Not with the People, Cadre Papers

[10] **Keane J., Carr S.** (2013). A Brief History of Early Unmanned Aircraft, Johns Hopkins Apl Technical Digest, Volume 32

[11] **Blom J.** (2013). Unmanned Aerial Systems: A Historical Perspective, Combat Studies Institu Press

[12] **Davies T.** (2015). A note about the V1 and V2, King's London College, London

[13] **Ehrhard T.** (2010). Air Force UAVs The Secret History, A Mitchell Institute Study

[14] **Arjomandi A., Agostino S.** (2006). Classification of Unmanned Aerial Vehicles, The University of Adelaide, Australia

[15] **Anderson K., Kevin G.** (2013). Lightweight Unmanned Aerial Vehicles Will Revolutinize Spatial Ecolgy, Frontiers in Ecology and the Environment

[16] **Lyon D.** (2004). A Military Perspective on Small Unmanned Aerial Vehicles, IEEE Inrumentation & Measurement Magazine (Volume: 7 Issue 3)

[17] **Grier P.** (1996). Darkstar and Its Friends, Air Force Magazine

[18] **Kinzig B., MacAulay-Brown** (2010). Global Hawk Systems Engineering Case Study, Air Force Center for Systems Engineering

[19] **Schneider D.** (2020). The Delivery Drones Are Coming, IEEE Spectrum

[20] **Jo D., Kwon Y** (2017). Analysis of VTOL UAV Propellant Technology, Journal of Computer and Communications

[21] **STM** https://www.stm.com.tr/tr/cozumlerimiz/

[22] **McKinnon G.** (2010). The Birth of a Drone Nation: American Unmanned Aerial Vehicles Since 1917, Luisiana State University, Master's Thesis

[23] **Fields N.** (2012). Advantages and Challenges of Unmmaned Aerial Vehicle Autonomy in the Postheroic Age, James Madison University, Master's Thesis

[24] **Latici T.** (2019). Civil and Military Drones: Navigating a Drisruptive and Dynamic Technological Ecosystem, European Parliament

[25] **Xu R., Özgüner Ü.** (2006) Sliding Mode Control of a Quadrotor Helicopter, 45th IEEE Conference on Decision & Control, 13-15 November

[26] **Bouabdallah S., Siegwart R.** (2007) IEEE Full Control of a Quadrotor, International Conference on Intelligent Robots and Systems, October 29 – 2 November

[27] **Bo Z., Bin X.** (2012) Hardware-in-Loop Testbed for Quadrotor Aerial Vehicles, IEEE Chinese Control Conference, 25-27 July

[28] **Lehnert C., Corke P.** (2013) Design and Implementation of An Open Source Micro Quadrotor, Proceedings of Australasian Conference on Robotics and Automation, 2-4 December

[29] **Atalay Y., Savran A.** (2017) Quadrotor İnsansız Hava Araci İçin Kontrol Sistemi Tasarımı, Master of Science Thesis, Ege University

[30] **Cömert C., Kasnakoğlu C.** (2017) Comparing and Developing PID and Sliding Mode Controllers for Quadrotor, International Journal of Mechanical Engineering and Robotics Reseach, 3 May

[31] **Demiryürek A., Demircioğlu H.** (2018) Modeling and Control of a Quadrotor, Master of Science Thesis, Hacettepe University

[32] **Tüzel Ş.** (2019) Altitude and Position Control of a Quadrotor Using Onboard Vision System, Master of Science Thesis, Middle East Technical University

[33] **Karahan M. Kasnakoğlu C.** (2019) Dört Rotorlu Bir İnsansız Hava Aracının Modellenmesi ve PID Kontrolcü Tasarımı, Master of Science Thesis, TOBB Ekonomi ve Teknoloji Üniversitesi

[34] **Lechekhab T. Manojlovic S.** (2020) Robust Error Based Active Disturbance Rejection Control of a Quadrotor, Aircraft Engineering and Aerospace Technology, Volume 93

[35] **Sundarapandian V., Chang-Hua L.** (2017). Applications of Sliding Mode Control in Science and Engineering, Description and Dynamics Modeling of the Quadrotor UAV, Sayfa 64

[36] **Kuon I., Tessier R** (2007). FPGA Architecture: Survey and Challenges

[37] **Stephen M., Trimberger S.** (2015). Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology

[38] **Amara A., Amiel F.** (2006). FPGA vs ASIC for Low Power Applications, Microelectronic Journal 37, Elsevier

[39] **Huffmire T., Kastner R.** (2008). Managing Security in FPGA-Based Embedded Systems, IEEE

[40] **Tate F., Tate R.** (2005). Use of Field Programmable Gate Array Technology in Future Space Avionics, IEEE

[41] **Ciani L., Catelani M.** (2014). A Fault Tolerant Architecture to Avoid the Effects of Single Event Upset (SEU) in Avionics Applications, Elsevier

[42] **Lanuzza M., Zicari P.** (2010). Exploiting Self-Reconfigurable Capability to Improve SRAM-Based FPGA Rebustness in Space and Avionics Applications, ACM Transactions on Reconfiguraable Technology and Systems

[43] **Xilinx.** (2019). Vivado Design Suite Tutorial, Model-Based DSP Design Using System Generator, UG948

[44] **Xilinx.** (2019). Vivado Design Suite User Guide: Model-Based DSP Design Using System Generator, UG897

[45] **Kocur M., Kozak S.** (2014). Design and Implementation of FPGA – Digital Based PID Controller, International Carpathian Control Conference, Velke Karlovice, Czech Republic, 28-30 May

[46] **Georgopoulos K., Chrysos G.** (2016). An Evaluation of Vivado HLS for Efficient System Design, 58th International Symposium ELMAR, IEEE

[47] **Digilent.** (2018). Zybo Z7 Board Reference Manual

[48] **Xilinx.** (2018). MicroBlaze Processor Reference Guide, UG984

[49] **Karahan M., Kasnakoğlu C.** (2020). Autonomous Face Detection and Tracking Using Quadrotor UAV, 4th International Symposium on

Multidisciplinary Studies and Innovative Technologies (ISMSIT), 22 October