İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

FPGA TABANLI AES ŞİFRELEMESİ VE YAPAY SİNİR AĞI İÇEREN SENTETİK AÇIKLIKLI RADAR İŞARET İŞLEME UYGULAMASI

LİSANS BİTİRME TASARIM PROJESİ

Furkan CAN

Furkan AYDIN

Fatma GÖK

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

Uygundur
Berna Örs Yalçın
17.06.2021

HAZİRAN, 2021

İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

FPGA TABANLI AES ŞİFRELEMESİ VE YAPAY SİNİR AĞI İÇEREN SENTETİK AÇIKLIKLI RADAR İŞARET İŞLEME UYGULAMASI

LİSANS BİTİRME TASARIM PROJESİ

Furkan CAN
040160043

Furkan AYDIN
040160067

Fatma GÖK
040160060

Proje Danışmanı: Prof. Dr. Sıddıka Berna ÖRS YALÇIN
Doç. Dr. Özgür ÖZDEMİR

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

HAZİRAN, 2021

İTÜ, Elektronik ve Haberleşme Mühendisliği Bölümü'nün ilgili Bitirme Tasarım Projesi yönergesine uygun olarak tamamen kendi çalışmamız sonucu hazırladığımız "FPGA TABANLI AES ŞİFRELEMESİ VE YAPAY SİNİR AĞI İÇEREN SENTETİK AÇIKLIKLI RADAR İŞARET İŞLEME UYGULAMASI" başlıklı Bitirme Tasarım Projesi'ni sunmaktayız. Bu çalışmayı intihal olmaksızın hazırladığımızı taahhüt eder; intihal olması durumunda bitirme tasarım projesinin başarısız sayılacağını kabul ederiz.

Furkan CAN .....
040160043

Furkan AYDIN .....
040160067

Fatma GÖK .....
040160060

ÖNSÖZ

Bitirme tasarım proje çalışmamız boyunca her konuda anlayışlı olan ve yardımını esirgemeyen Prof. Dr. Berna Örs Yalçın hocamıza ve Doç. Dr. Özgür Özdemir hocamıza, projemizde bilgi ve tecrübeleriyle bize yardımcı olan Özgür Ozan Yılmaz'a sonsuz teşekkür eder, en içten saygılarımızı sunarız.

İstanbul Teknik Üniversitesi lisans hayatımızda eğitimimize katkı sağlayan, bakış açımızı genişleten ve bize vizyon katan hocalarımıza ve kıymetli arkadaşlarımıza , teşekkür ederiz.

Ayrıca hayatımızın her anında bize destek olan sevgili ailelerimize ve sevdiklerimize teşekkürlerimizi sunuyoruz.

Haziran 2021

Furkan CAN

Furkan AYDIN

Fatma GÖK

İÇİNDEKİLER

Sayfa

KISALTMALAR

SAR : Sentetik Açıklıklı Radar
GPU : Graphics Processing Unit
FPGA : Field Programmable Gate Array
RDA : Range-Doppler Algorithm
FFT : Fast Fourier Transform
IFFT : Inverse Fast Fourier Transform
RCMC : Range Cell Migration Correction
IP : Intellectual Property
RSA : Rivest Shamir Adleman
NN : Neural Network
SVM : Support Vector Machine
SOM : Self Organizing Map
CNN : Convolutional Neural Network
NASA : National Aeronautics and Space Administration
SIR : Shuttle Imaging Radar
ESA : European Space Agency
ERS : European Remote Sensing
ADC : Analog Digital Converter
SPECAN : Spectral Aanlysis
DTF : Darbe Tekrarlama Frekansı
PRF : Pulse Repetition Frequency
SEASAT : Sea Satellite
RAM : Random Access Memory
DTC : Data Type Conversion
LUT : Look Up Table
HDL : Hardware Description Language
AES : Advanced Encryption Standad
DES : Data Encryption Standard

SPN **:** Substution Permutaion Network

FIPS **:** Federal Information Processing Standards

NIST **:** National Institute of Standards and Technology

ISE **:** Integrated Software Environment

SDK **:** Software Development Kit

XOR **:** Exclusive OR

GF **:** Galois Field

RCON **:** Round Constant

AXI **:** Advanced Extensible Interface

SEMBOLLER

f_s : Örnekleme Frekansı

τ_p : Darbe Süresi

k : Cıvıltı Eğimi(Chirp Slope)

f : Frekans

λ : Dalgaboyu

f_{DC} : Doppler Merkez Frekansı

R_c : Görüntülenen alanın merkeze uzaklığı

f_r : Doppler frekans hızı

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

FPGA ÜZERİNDE AES ŞİFRELİ SENTETİK AÇIKLIKLI RADAR İŞARET İŞLEME UYGULAMASI

ÖZET

Sentetik açıklıklı radar(SAR)'lar günümüzde sıklıkla uzaktan algılama sistemlerinde görüntüleme amacıyla kullanılmaktadır. Bu sistemler genellikle uydu ve hava araçları olmak üzere iki platformda kullanılmaktadır. Bunlar dışında kara araçlarında mobil olarak kullanılarak çeşitli amaçlarla kullanılmaktadır. SAR sistemleri bu araçların uçuş rotasındaki güzergâhın yüksek çözünürlükte görüntülenmesini sağlamaktadır. Burada bulunduğu platformun hareket etmesiyle SAR anteni de bu güzergâhta ilerlemiş olur. SAR aktif bir radar olmakla birlikte gece-gündüz saatleri ve hava durumu fark etmeksizin zorlu koşullarda da görüntüleme yapabilmesi sebebiyle tercih edilmektedir.

SAR sisteminden elde edilen ham verilerin işlenmesi için genellikle GPU'lar ve FPGA'ler yüksek performansları sebebiyle tercih edilmektedir. GPU'da bu işlem yazılım tabanlı olmakla birlikte gerçekleme kolaydır ancak donanımın kullandığı yoğun kaynak sebebiyle güç tüketimi fazladır. FPGA'lerin tasarım mimarisi çok daha özgürdür ve daha az güç tüketir ancak tasarımı için sayısal tasarım alanında uzmanlık gerektirmesi sebebiyle tasarım süreçleri daha uzun sürmektedir ancak Xilinx firması ile Mathworks firmasının birlikte çalışarak ortaya çıkardığı Model Composer programı ile model tabanlı sayısal devre tasarımı yapılabilmektedir. Yeni geliştirilmiş olan bu program yardımıyla sinyal işleme, lojik işlemler ve bit manipülasyonları işlemlerini gerçekleştiren bloklar program tarafından sunulmaktadır ve Simulink ortamında bu bloklarla oluşturulan model tabanlı tasarımların benzetimi de yapılabilmektedir. Bunun yanında Model Composer kütüphanesin tarafından hazır olarak sunulan bloklar dışında C/C++ programlama dilleri ile oluşturulan fonksiyonlar Model Composer kütüphanesine eklenerek proje kütüphanesine eklenebilmektedir.

Proje kapsamında SAR işaret işleme uygulamasının FPGA'de gerçeklenmesi, aynı zamanda SAR görüntülerinden elde edilen görüntünün hava aracı veya uydu üzerinden bir yer istasyonuna gönderildiği düşünülerek oluşturulan görüntünün gönderilmesi bir karar mekanizmasına bağlanmıştır. Bunun için bir sinir ağı eğitilip sisteme eklenmiştir. Sinir ağının ardına da sistemin kullanım amacının güvenli haberleşmeye ihtiyaç duyacağı düşünülerek kripto modülü de projenin bir parçasını oluşturmuştur.

Projenin ilk kısmında Sentetik Açıklıklı Radar(SAR) verilerinin FPGA'de işlenmesi gerçekleştirilecektir. Bu işlemi yapmadan önce ilk olarak bir veri seti araştırılmıştır. En uygun veri seti ESA'nın akademik araştırmalar için paylaştığı ERS verileri kullanılması uygun görülmüştür. Bu veri setinin kullanılma nedeni görüntü oluşturulduktan sonra üzerinde sınıflandırma yapılabilmesi ve sistem parametrelerinin paylaşılmasıdır. Veri seti seçildikten sonra uygulanacak algoritma belirlenmiştir. Yaygın kullanılan ve uygulaması diğer algoritmalara göre daha kolay olan Menzil Doppler algoritması seçilmiştir. Ardından veri setinin parametreleri çıkartılıp algoritma MATLAB üzerinde gerçeklenmiş ve istenen netlikte görüntüler oluşturulmuştur. Bu aşamanın ardından algoritma Model Composer üzerinde tasarlanarak simülasyonu yapılmıştır. MATLAB'da elde edilen görüntü ile Model Composer simülasyonu sonucunda oluşturulan görüntü karşılaştırılıp sistem doğrulanmıştır. Bu aşamanın ardından oluşturulan model Model Composer ile paketlenip Vivado programına geçilmiştir ancak görüntü oluşturma aşamasında

kullanılan FFT/IFFT blokları'nın Vivado ile uyumsuzluğu sebebiyle oluşturulan paket istendiği gibi çalıştırılamamıştır. Bu problem için çeşitli çözümler geleceğe yönelik öneriler kısmında verilecektir.

İkinci kısımda ise üretilen görüntü için bir yapay sinir ağı eğitilerek elde edilecek karar görüntüsü SAR platformundan yer istasyonuna gönderilmeye hazır hale getirilmiştir. CNN algoritmasına göre giriş resimleri belli bir örnekleme sayısına göre ayrıştırılıp sıkıştırılarak çok katmanlı bir nöron yapısında eğitilir. Daha sonra elde edilen sinir ağındaki ağırlıklar kullanılarak sonuç almak istediğimiz görüntüde görülmek istenilen sınıfa ait bir bölgenin olup olmadığı, nerde olduğu gibi çıkarımlar yapılır. CNN algoritmasını kullanmak için eğitilecek olan görüntüleri etiketlenmiş olması gerekmektedir fakat radar görüntülerinin bir etiketi bulunmadığı için bir eğiticisiz öğrenme yöntemi kullanmamız gerekmektedir çünkü SAR verilerinden oluşturulan görüntülerde eğiticili öğrenme yapılırken etiketlenmiş bir veri seti gerekmektedir. Bu sebeple eğiticisiz öğrenme algoritması olan Öz Düzenlemeli Ağ (SOM-Self Organizing Map) yöntemi kullanılmıştır. Aynı zamanda SAR görüntülerinin etiketlendiği yaygın bir veri seti bulunmamasından dolayı burada kulanılacak yöntemin eğiticisiz olması kararı pekiştirilmiştir. Burada ağın eğitimi Python ve MATLAB ortamı kullanılarak tasarlanmıştır. Sistem ilk eğitim sonucunda teste sokulmuş ancak istenen seviyede doğruluk oranı yakalayamamıştır. Bunun üzerine ağ ikinci bir eğitime sokulmuş ve doğruluk oranı da arttırılmıştır. Sistem görüntülenen alanda %50 ve üzerinde su alanları tespit ettiğinde görüntüyü yer istasyonuna gönderilecek şekilde ayarlanmıştır. Bu sistem de aynı şekilde Model Composer'da modellenerek IP (Intellectual Property) paketi haline getirilmiş ve Vivado'da Microblaze açık kaynak kodlu işlemci ile kontrol edilmiştir.

Son kısım olan kripto modülüne gelirsek, yer istasyonu ile hava aracı arasındaki haberleşmenin güvenli olarak gerçekleşmesi için simetrik ve asimetrik anahtarlı şifrelemelerin yer aldığı bir sistem tasarlanmıştır. Bu sistem iki aşamadan oluşmaktadır. İlk aşama şifreleme için kullanılacak olan anahtarın elde edilmesidir. Şifrelemede kullanılan anahtarın belli aralıklarla değişmesi amaçlanmıştır. Bu anahtar değişim işlemi, Diffie Hellman anahtar değişimi protokolü ile gerçekleşecektir. İkinci aşamada, elde edilen anahtar ile hem hız hem de güvenlik açısından standart hale getirilen simetrik şifreleme algoritmasına sahip olan AES şifreleme algoritması kullanılacaktır. Burada da şifreleme ve şifre çözme olarak iki ayrı tasarım bulunmaktadır. Kripto modülü de diğer bloklardan bağımsız şekilde kendi içinde Microblaze işlemcisi ile haberleştirilip FPGA üzerinde simülasyonu yapılmıştır.

FPGA IMPLEMENTATION OF AES ENCRYPTED SYNTHETIC APERTURE RADAR SIGNAL PROCESSING APPLICATION

SUMMARY

Synthetic aperture radar (SAR) are frequently used for imaging in remote sensing systems today. These systems are generally used in two platforms as satellite and air vehicles. Apart from these, it is used as mobile in land vehicles and used for various purposes. SAR systems provide high resolution visualization of the route on the flight path of these vehicles. With the movement of the platform on which it is located here, the SAR antenna also moves on this routes. Although SAR is an active radar, it is preferred due to its ability to monitor in difficult conditions regardless of day and night hours and weather conditions.

GPUs and FPGAs are generally preferred for processing raw data obtained from the SAR system due to their high performance. In the GPU, this process is software-based, but implementation is easy and power consumption is high due to the intensive resource used by the hardware. The design architecture of FPGAs is much free and consumes less power, but the design processes take longer because it requires expertise in digital design for its design. With the help of this newly developed program, blocks that perform signal processing, logic operations and bit manipulations are presented by the program, and model-based designs created with these blocks in the Simulink environment can also be simulated. In addition, functions created with C / C ++ programming languages can be added to the project library, except for the blocks provided by the Model Composer library.

Within the scope of the project, the realization of the SAR signal processing application in FPGA, as well as the sending of the image created by thinking that the image obtained from the SAR images is sent to a ground station via an aircraft or satellite was connected to a decision mechanism. For this, a neural network is trained and added to the system. Considering that the purpose of use of the system will need secure communication behind the neural network, the crypto module has also formed a part of the project.

In the first part of the project, Synthetic Aperture Radar (SAR) data will be processed in FPGA. Before doing this, a data set was first investigated. The most appropriate data set was ERS data shared by ESA for academic research. The reason for using this data set is that after the image is created, classification can be made on it and system parameters are shared. After the data set was selected, the algorithm determination phase was started. The Range Doppler algorithm, which is widely used and easier to implement than other algorithms, has been chosen. Then, the parameters of the data set were extracted, the algorithm was implemented on MATLAB and images with the desired clarity were created. After this stage, the algorithm was designed and simulated on Model Composer. The system was verified by comparing the image obtained in MATLAB with the image created as a result of Model Composer simulation. After this stage, the model created was packaged with Model Composer and moved to the Vivado program, but the package could not be run as desired due to the incompatibility of the FFT / IFFT blocks used in the image creation phase with Vivado. Various solutions for this problem will be given in the future suggestions section.

In the second part, an artificial neural network is trained for the produced image and the decision image to be obtained is made ready to be sent from the SAR platform to the ground station. According to the CNN algorithm, the input images are decomposed

and compressed according to a certain sampling number and trained in a multi-layered neuron structure. Then, using the weights in the obtained neural network, inferences are made such as whether there is a region belonging to the class that we want to see in the image we want to get results from, and where. In order to use the CNN algorithm, the images to be trained must be tagged, but since the radar images do not have a tag, we need to use a unsupervised learning method because a tagged data set is required when performing guided learning on images created from SAR data. For this reason, the Self-Organizing Map (SOM) method, which is an unsupervised learning algorithm, was used. At the same time, due to the lack of a common data set in which SAR images are labeled, the decision to use the unsupervised learning method was reinforced. Here, the training of the network is designed using Python and MATLAB environment. The system was put to the test as a result of the first training, but it could not achieve the desired level of accuracy. Thereupon, the network was put into a second training and its accuracy rate was increased. When the system detects 50% or more water areas in the displayed area, the image is set to be sent to the ground station. This system was also modeled in Model Composer and turned into an IP package and controlled with Microblaze open source processor in Vivado.

As for the last part, the crypto module, a system with symmetric and asymmetric key encryptions has been designed for the secure realization of the communication between the ground station and the aircraft. This system consists of two stages. The first step is to obtain the key to be used for encryption. The key used in encryption is aimed to change periodically. This key exchange will be performed with Diffie Hellman Key Exchange Protocol. AES Encryption algorithm, which has a symmetric encryption algorithm standardized in terms of both speed and security, will be used with the key obtained in the second stage. Here, too, there are two different designs as encryption and decryption. The crypto module is also communicated with the Microblaze processor independently of other blocks and simulated on the FPGA.

1. GİRİŞ

Sentetik Açıklıklı Radar(Synthetic Aperture Radar- SAR)'lar uzaktan algılama sistemlerinde görüntüleme amacıyla kullanılmaktadır[1]. Bu sistemlerin genelde iki platformda kullanıldığı bilinmektedir. Bunlardan biri hava araçları iken diğeri ise uydulardır. SAR sistemleri bu araçların uçuş rotasındaki güzergâhın yüksek çözünürlükte görüntülenmesini sağlamaktadır. SAR aktif bir radar olmakla birlikte gece-gündüz saatleri ve hava durumu fark etmeksizin zorlu koşullarda da görüntüleme yapabilmesi sebebiyle tercih edilmektedir.

SAR sisteminden elde edilen ham verilerin işlenmesi için genellikle Grafik İşleme Birim'leri (Graphics Processing Unit- GPU)[2] ve Alanda Programlanabilir Kapı Dizi'leri(Field Programmable Gate Array- FPGA)[3] yüksek performansları sebebiyle tercih edilmektedir. GPU'da bu işlem yazılım tabanlı olmakla birlikte gerçekleme kolaydır ancak donanımın kullandığı yoğun kaynak sebebiyle güç tüketimi sistemlerin FPGA üzerinde gerçeklemesine göre fazladır. FPGA platformu özelleştirilmiş bir mimariye sahiptir ve daha düşük güç tüketimine izin verir ancak tasarımı için sayısal tasarım alanında uzmanlık gerektirmesi sebebiyle tasarım süreçleri daha uzun sürmektedir[4].

Yapılacak proje, tümü FPGA üzerinde olmak üzere sırasıyla görüntü işleme, yapay sinir ağı ve kriptografi bloğu içermektedir. Görüntü işleme bloğu, ham SAR görüntülerini işleyerek uygu görüntülerini elde etmektedir. Yapay sinir ağı bloğu, ortaya çıkan görüntünün iletim için uygun özellikleri barındırıp barındırmadığını belirlemektedir. Kriptografi bloğu, iletim için hazır olan görüntünün haberleşme güvenliğini sağlamaktadır.

SAR sistemlerinin işlenmesinin temeli, işlenmemiş SAR verileri üzerinde gerçekleştirilen iki boyutlu eşleştirme filtrelemesidir. SAR görüntüsü oluşturulurken kullanılan çok sayıda algoritmadan biri Menzil Doppler Algoritması (Range-Dopper Algorithm-RDA) [5]. Ancak gerçekleme gereksinimlerini karşılamak yüksek hesaplama yükü nedeniyle genellikle zordur.

Bu algoritmanın temel blokları eşleştirme filtreleri (range ve azimuthta), Hızlı Fourier Dönüşümü(Fast Fourier Transform- FFT)[6]/Ters Hızlı Fourier Dönüşümü(Inverse Fast Fourier Transform-IFFT)[6] (range ve azimuthta) ve Menzil Hücre Göçü Düzeltmesi (Range Cell Migration Correction- RCMC)[6]'dir. Burada FFT/IFFT uygulanmasının sebebi ise zaman domaininde konvolüsyon işleminin uygulanmasının zor olmasıdır. Frekans domainine geçilmesiyle birlikte basit çarpma işlemi uygulanır. Sinyal işleme temellerinden gelen özelliğe göre zamanda konvolüsyon frekansta çarpım, frekansta konvolüsyon zamanda çarpım işlemine denktir. FFT uygulanmasıyla birlikte sinyal oluşturulan menzil referans fonksiyonuyla çarpılır ve IFFT uygulanarak range ekseninde kısmı tamamlanır ve azimut ekseninde işlemeye geçilir. Range'de uygulanan işlemlerin aynısı uygulanır. Yalnızca IFFT işlemi öncesinde RCMC işlemi uygulanır[7].

Oluşturulan SAR görüntülerinin hava platformundan yer istasyonuna ne kadar sürede veya ne sıklıkta gönderileceğine karar vermek için bir sinir ağı kullanılmasına gerek duyulmuştur. Burada haberleşme kanalının bandgenişliğini verimli kullanma amacıyla bu yola başvurulmuştur. Yapay sinir ağları eğiticili öğrenme[8] ve eğiticisiz öğrenme[8] olarak ikiye ayrılır. Eğiticili öğrenme yapay sinir ağının etiketlenmiş veriler yardımı ile eğitilmesi, eğiticisiz öğrenme ise eğitimin etiket kullanılmadan yapılmasıdır [8]. SAR görüntüleri ile bir yapay sinir ağını eğitmek için birçok algoritma kullanılabilir. Çok Katmanlı Algılayıcı (Multi-Layer Perceptron-MLP)[9], Destek Vektör Makinesi (Support Vector Machine-SVM)[10], Evrişimsel Sinir Ağları (Convolutional Neural Network-CNN)[11] gibi algoritmalar arasından CNN algoritması SAR görüntüleri için yaygın kullanılan bir algoritmadır[12] [13]. Fakat CNN algoritmasını kullanmak için eğitilecek olan görüntüleri etiketlenmiş olması gerekmektedir. Bizim elimizdeki veri setleri etikenlenmemiş verilerden oluştuğu için bir eğiticisiz öğrenme yöntemi kullanmamız gerekmektedir. Eğiticisiz öğrenme algoritmaları arasında Öz Düzenlemeli Ağ (Self-Organizing Map-SOM)[14] sınıflandırma problemleri için oldukça kullanışlı ve Yinelemeli Öz Düzenlemeli Veri Analizi (Iterative Self Organizing Data Analysis-ISODATA) [15] gibi başka eğiticisiz öğrenme yöntemlerine göre daha tutarlı ve gerçeklemesi kolaydır[16]. Genel olarak SOM etiketlenmemiş bir eğitim setindeki farklılıkları bularak bu farklılıkları sınıf haline getirir bu sayede görüntüde istenilen sınıfları ayırt etmemize olanak sağlar. Bu sayede istenilen eğitilmiş ağı etiketlenmemiş verilerden elde edilecektir.

Güvenliğin sağlanması kısmında, yer istasyonu ile hava aracı arasındaki haberleşmenin güvenli olarak gerçekleşmesi için şifreleme sistemleri kullanılmıştır[17]. Kriptoloji [18] ayağı iki aşamadan oluşmaktadır. İlk aşama şifreleme için kullanılacak olan anahtarın elde edilmesidir. Yer istasyonu ile hava platformu, belli aralıklarda değiştirilecek anahtarı ortak kullanabilmeleri için Diffie Hellman anahtar değişimi protokolü[19] kullanılmıştır. İkinci aşamada elde edilen anahtar ile hem hız hem de güvenlik açısından standart hale getirilen simetrik şifreleme algoritmasına sahip olan Gelişmiş Şifreleme Standardı(Advanced Encryption Standard- AES)[20] kullanılacaktır.

Bu işlemleri FPGA'de gerçeklerken çeşitli araçlar kullanılmaktadır. Bu araçlardan biri de Xilinx firmasının MATLAB Simulink[21] altyapısı kullanılarak tasarlanan Model Composer[22] aracıdır. Model Composer'da tasarlanan modellerin Yüksek hızlı tümleşik devreler için Donanım Tanımlama Dili (Very high speed integrated circuit Hardware Description Language - VHDL) [23] kodları optimize şekilde üretilmesi ana amaçtır. Bu bağlamda bu firmanın ürettiği bu yazılım için hazırladığı dokümanları, araç hakkında yapılan incelemelerin temel kaynaklarını oluşturur[24]. Bu kaynaklar araç hakkında temel bilgilerin edinilmesini sağlayacaktır. Model Composer'da tasarlanan modellerin Vivado[25] ve Vitis[26] programları ile bir FPGA üzerinde gerçeklenmesi ve simulasyonunun yapılması sağlanır.

Sistemi özetlemek gerekirse ham veriden SAR görüntüsü oluşturan bir blok, ardından buradan elde edilen görüntüde sınıflandırma yapıp hangi görüntülerin gönderileceğine karar veren bir yapay sinir ağı ve gönderilecek verinin şifrelenmesini içeren bir bloktan oluşur. Böylelikle bu üç blok birleştirilerek bir sistem tasarımı anlatılacaktır.

2. SAR SİSTEMİ VE SAR HAM VERİSİNİN İŞLENMESİ

2.1 SAR Sistemlerin Tarihsel Gelişimi

SAR sistemleri üzerine ilk çalışma 1953 yılında Illinois Üniversitesi araştırmacıları tarafından yapılmıştır[27]. Bunun dışında ilk kapsamlı çalışma Amerikan ordusunun ihtiyaçları doğrultusunda Michigan Üniversitesinden bu konu üzerine (Volvorine Project) çalışmaları istenmiştir. Illinois Üniversitesi, General Electric, Philco, Varian ve Goodyear Aircraft Corporation şirketlerinin katılmasıyla oluşan bu topluluk SAR ile ilgili ilk detaylı çalışmayı ortaya koymuştur.

1957 yılında Michigan Üniversitesinin çalışmalarıyla ilk gerçekleştirilen SAR sistemi, Amerikan ordusunun savunma ihtiyaçları için yapılan X bantta çalışan SAR sistemi olduğu düşünülmektedir. Ulusal Havacılık ve Uzay Dairesi(National Aeronautics and Space Administration –NASA)'nin destekleriyle 1960'lı yılların sonlarında ilk X bant SAR sistemi Michigan Çevre Araştırmaları Enstitüsü(Enviromental Research Institute of Michigan- ERIM) tarafından yapılmış olup daha sonra NASA tarafından Lbant bir kanal daha eklenerek sistem yenilenmiştir [27].

1960' ların sonundan itibaren, NASA, SAR sistemleri için destekleyici olmaya başlamıştır. İlk X band sistemi ERIM tarafından yapılmıştır ve bu sistem 1973 yılında L bandında bir kanalın eklenmesiyle (NASA tarafından) güncellenmiştir. Bu sistemlerin tasarlanmasında Jet Tahrik Laboratuvarı(Jet Propulsion Laboratory- JPL) de destek vererek roket ve uçaklarda denenmiştir. Bu çalışmaların ardından NASA SEASAT-A adı verilen projeye başlamıştır. Bu projede okyanus, buzul ve yeryüzü incelemeleri yapılmıştır. Bu projenin sonuçlarının olumlu olmasının ardından NASA, Görüntüleme Radar Mekiği(Shuttle Imaging Radar- SIR) programını başlatmıştır. 1981 yılında programın ilk tasarımı olan SIR-A sistemi ile bilimsel araştırma, yeryüzü inceleme, Dünya kaynakları ve benzeri amaçlarla inceleme yapılmıştır. Ardından 1984 yılında SIR-B sistemi devreye girmiş, yenilenebilir kaynaklar, jeoloji ve okyanus coğrafyası gibi alanlarda kullanılarak bu alanlarda katkı sağlamıştır. Bu programın son etabı olan SIR-C sistemi 1994' te uygulanmıştır. Bu sistemde SIR-A ve SIR-B sistemlerinden farklı olarak L bandının dışında C bandda da çalışmıştır. Daha sonra sisteme X bandında da çalışması için SIR-C'ye eklemeler yapılmıştır. 90'lı yıllarda ise ESA(Europian Space Agency)[28]'nın atmosfer, yeryüzü ve okyanusları gözlemleme ve değişimleri inceleme amacıyla Avrupa Uzaktan Algılama Uydusu(European

Remote Sensing Satellite-1 - ERS-1) ve ERS-2 uydularında C bandda çalışan SAR sistemlerini kullanmıştır. 1991'de ve 1995'te başlatılan bu programlar sayesinde iklim değişikliği, çevresel araştırmalar, bilimsel araştırmalar ve hava tahminleri yapılmasında katkı sağlamışlardır. ERS-1 2000 yılında tutum kontrol sisteminin arızası sebebiyle işlevini kaybederken ERS-2 2011 yılında planlanan görev süresini tamamlamıştır. Bu iki sistem toplamda 18 yıl hizmet etmiştir. Günümüzde ise askeri, tarımsal izleme, coğrafi sistemler ve iklim çalışmaları gibi bir çok alanda yaygın şekilde uydu ve hava araçlarında kullanılmaktadır.

2.2 SAR Teorisi

SAR günümüzde uzaktan algılama sistemlerinde görüntüleme amacıyla çokça kullanılmaktadır. Bu sistemlerin genelde iki platformda kullanıldığı görülmektedir. Bunlardan biri hava araçları iken diğeri ise uydulardır. Bunlar dışında kara araçlarında mobil olarak kullanılarak çeşitli amaçlarla kullanılmaktadır. SAR sistemleri bu araçların uçuş rotasındaki güzergâhın yüksek çözünürlükte görüntülenmesini sağlamaktadır[29]. Burada bulunduğu platformun hareket etmesiyle SAR anteni de bu güzergâhta ilerlemiş olur. Bu sayede çok sayıda işaret gönderilerek hedef görüntüler oluşturulur. SAR aktif bir radar olmakla birlikte gece-gündüz saatleri ve hava durumu fark etmeksizin zorlu koşullarda da görüntüleme yapabilmesi sebebiyle tercih edilmektedir[30].

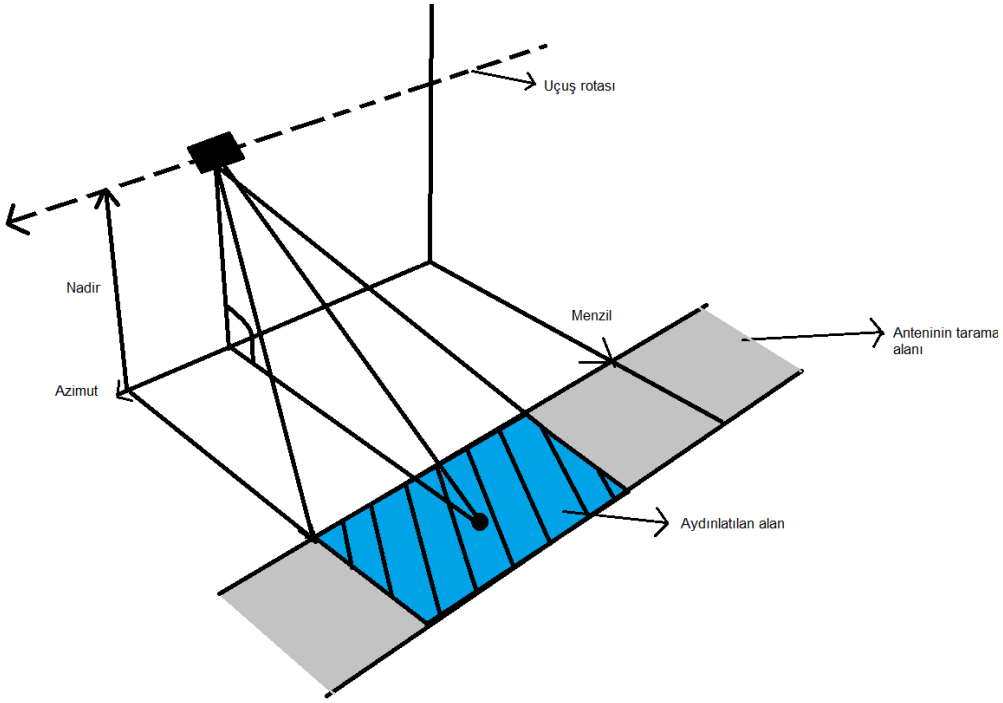
SAR sistemi aktif ve yan-bakışlı bir radardan oluşan donanım ve alınan veriyi görüntüye çeviren bir SAR işlemcisinden oluşmaktadır. Hareket halindeki bir uzay veya hava aracının üzerinde bulunan anten ve çevre donanımlarla cıvıltı(Chirp) sinyali gönderilir. Görüntülenen bölgedeki ortamdan yansıyıp gelen analog işaretler Analog-Dijital Dönüştürücü(Analog Digital Converter-ADC)'lerden ve diğer donanımlardan geçirilerek sayısallaştırılır ve işlemciye iletilir. SAR işlemcisine gelen bu veriler kaydedilir ve yer istasyonunda işlenmek üzere iletilir veya SAR işlemcisinde belli algoritmalardan geçirilerek işlenir ve görüntüye çevrilir. Oluşturulan görüntü yere indirilir.

2.2.1 SAR geometrisi

SAR geometrisi belirli bir rota veya yörüngede hareket eden bir platformun yine belirli bir hızda yan-bakışlı bir antenle bir bölgeyi taraması olarak söylenir. Antenin

hüzmesiyle ışıma yapılarak aydınlatılan bölgenin genişliği ve aydınlatılma süresi gönderilen işaretin süresine ve spektral özelliklerine bağlıdır. Platformun üzerindeki antenin hareket ekseni azimut (çapraz menzil) olarak adlandırılırken antenin hareket eksenine dik olan eksen ise menzil olarak adlandırılmaktadır. SAR görüntüleme geometrisi Şekil 2.1'de gösterilmektedir.

Anten ile hedef arasındaki doğrusal uzaklığa eğik menzil (slant range) denir. Antenin yerden yüksekliği ise nadir olarak adlandırılır. Aynı zamanda yatay eksende ilk menzil çizgisine yakın menzil denilirken yansıyan işaret alınan son menzil çizgisine ise uzak menzil adı verilir. Arada kalan alan kabaca antenin tarama alanıdır[31].



Şekil 2.1 : SAR görüntüleme geometrisi

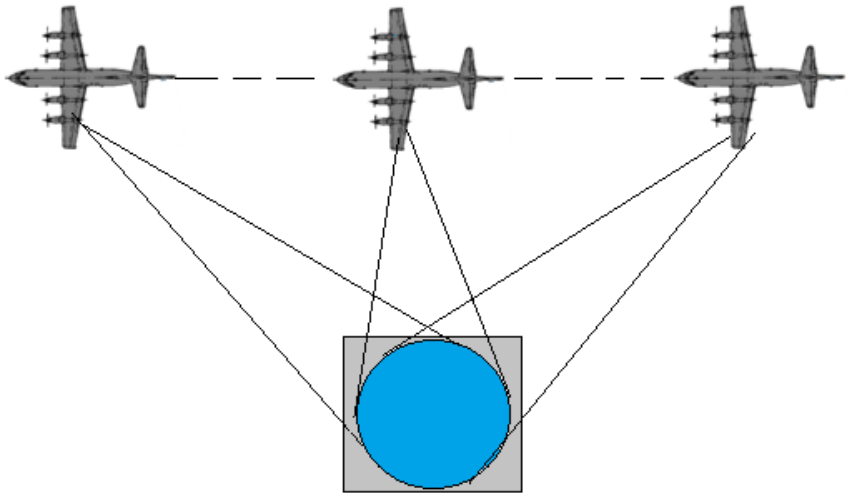
2.2.2 SAR modları

SAR sistemlerinde 3 farklı çalışma modu bulunmaktadır. Kullanılan bu modlar; spot aydınlatmalı(Spotlight) (bir diğer ismi ise nokta ışınlandırmadır), tarama (Scan) ve şerit tarama (Stripmap) çalışma modlarıdır. En sık kullanılan çalışma modu şerit tarama modu olarak bilinmektedir[32] [33].

2.2.2.1 Spot aydınlatmalı (Spotlight) SAR

Antenle veri toplanırken görüntülenen bölge hava veya uydu platformundan gönderilen ışın demetleri ile sürekli aydınlatılır. Hedeflenen bu bölgeye odaklanılarak

hedef alan görüntülenmiş olur. Veriler hedeflenen alanda bir referans nokta belirlenerek alınır ve bu nokta ile hesaplamalar yapılır. Spot aydınlatmalı modda yapılan taramalarda bölge farklı açılardan taranabilmesi sebebiyle yansıma katsayıları görüntüleme açılarıyla değişen durumlarda kullanılması önemli bir kazanç sağlar. Aynı zamanda bu modda elde edilen sentetik açıklık şerit taramalı (stripmap) moda göre daha fazla olarak elde edilebilir. Şekil 2.2'de spot aydınlatmalı modun elde edilmesi gösterilmiştir[34].



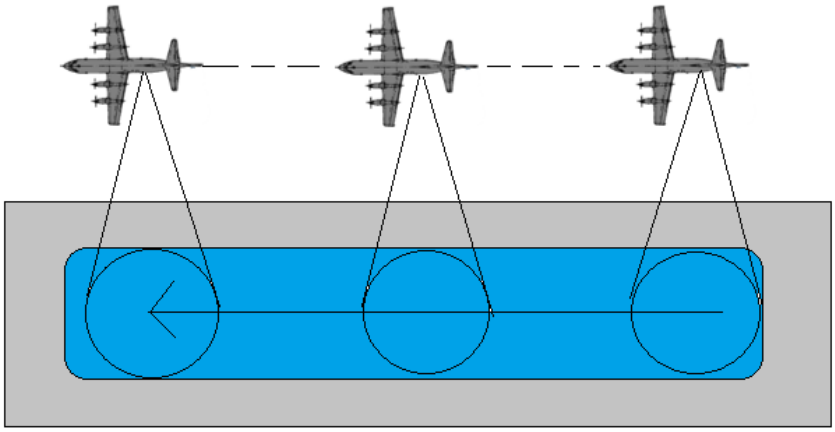
Şekil 2.2 : Spot aydınlatmalı mod görüntüleme geometrisi

2.2.2.2 Şerit taramalı (Stripmap) SAR

En çok kullanılan SAR görüntüleme modu olan şerit tarama modunda SAR anteni veya hareketli platform azimut yönünde hareket eder. Bu tarama sonucunda istenen azimut uzunluğunda bir bölgeden gönderilen ışınlar geri toplanarak görüntüleme işlemlerine geçilir. Spot aydınlatmada hedef bölgenin merkezinde bir referans seçilmişti. Şerit tarama modunda ise uçuş doğrultusu boyunca veya bir başka deyişle azimut yönü boyunca bir çizgi merkez olarak seçilir. SAR anteni bu görüntüleme merkezi çizgisine göre iki farklı şekilde seçilebilir. Bunlardan ilki dik-bakış(boresight) modelidir. Bu modelde antenin ışıma doğrultusu görüntüleme çizgisine dik konumlanmıştır. Diğer model ise açılı-bakış(squinted) modelidir. Bu modelde ise anten görüntüleme merkezi çizgisine ileri veya geri yönde belirli bir açıda olur ve bu açıya yan bakış açısı denir. Açılı-bakış modeli genellikle görüntülenecek bölgenin dağlık ve engebeli olduğu

durumlarda kullanılırken dik-bakış modeli düzlük alanların görüntülenmesinde tercih edilmektedir[27] [35].

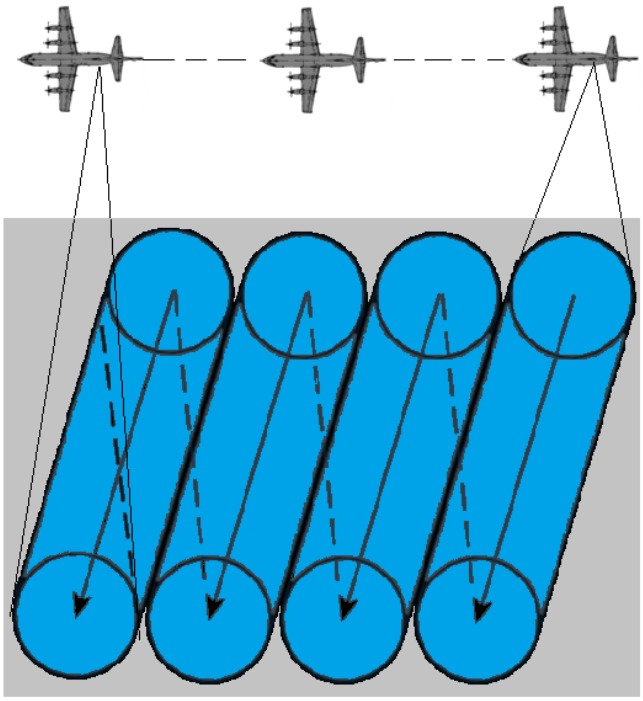
Bu modda görüntüleme yapılırken azimut yönündeki uzunluk SAR sisteminde gönderilen dalganın özelliklerine ek olarak hava veya uzay platformunun uçuş süresine de bağlıdır. Menzil yönünde ise yalnızca gönderilen dalganın biçimine bağlıdır. Minimum zamanda maksimum alan görüntüleyebilmesi sebebiyle sıklıkla kullanılır ancak diğer modlara göre çözünürlük daha az olmaktadır[34] [36]. Şerit tarama modu Şekil 2.3'te gösterilmektedir.



Şekil 2.3 : Şerit taramalı mod görüntüleme geometrisi

2.2.2.3 Taramalı (Scan) SAR

Bu modda belirli bir açıda hedeflenen bölge hareket yönü boyunca tarama yapılarak görüntüleme yapılır. Diğer bir deyişle şerit tarama ve spot aydınlatmalı modun hibrit modu olarak söylenebilir. Bu modun kullanılması ile menzil çözünürlüğü iyileştirilmesine karşın hedef bölgenin komşu bölgeleri için oluşturulan yapay açıklığın bir bölümü kullanılır. Bunun sonucunda ise azimuth çözünürlüğü azalır. Diğer yöntemlere göre gerçeklenmesi ve işaretlerin işlenmesi daha zor olduğu için çoğunlukla tercih edilmez [27]. Taramalı SAR Şekil 2.4'te gösterilmiştir.



Şekil 2.4 : Taramalı mod görüntüleme geometrisi

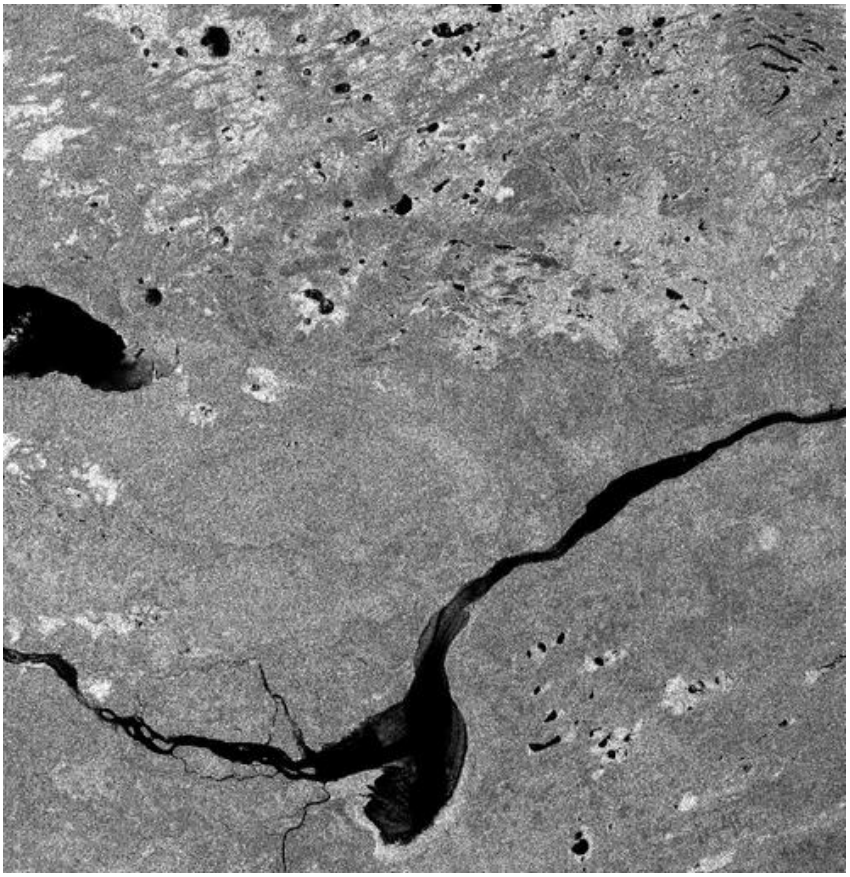
2.3 SAR Ham Verisinin İşlenmesi

Günümüzde SAR işaret işleme için literatürde bir çok algoritma bulunmaktadır. Bu algoritmalar çeşitli yönlerden fayda sağlarken getirdiği dezavantajları da bulunmaktadır. Literatürde menzil ve azimut yönü ayrı ayrı işlendiği görülmekle birlikte birlikte işlendiği algoritmalar da bulunmaktadır[37]. Bu sayede iki yönde de çözünürlükler iyileştirilebilmektedir. En yaygın kullanılan algoritmalardan en önemlileri Menzil Doppler Algoritması, Küresel Geri Yansıma(Global Back-Projection) [38], Spektrum Analizi(SPECAN) [39] ve Cıvıltı Ölçekleme (Chirp Scaling) [40] Algoritması algoritmaları olarak sıralanabilirler.

Bu projede ise gerçeklenme kolaylığı ve tek boyutta işlem yapılması sebebiyle Menzil-Doppler algoritması kullanılacaktır. İşleme kolaylığı ise birkaç sebeple oluşmaktadır. İlk olarak Menzil-Doppler algoritması frekans domeninde işlemler yapması sebebiyle gerçeklemesi zor olan konvolüsyon işlemi yerine çarpım işlemi kullanılır. Bu işlem sinyal işlemenin temel bilgilerinden olan zamanda konvolüsyon frekansta çarpıma eşittir kuralına dayanarak söylenmektedir. Aynı zamanda menzil ve azimut boyutlarında ayrı ayrı frekans domenine alınan referans fonksiyonları ile çarpılarak boyutlardaki işlem yükü azaltılır ve donanımda gerçeklenmesi kolaylaşır[27][35].

2.3.1 ERS-1/2 uyduları ve SAR parametreleri

ERS uyduları ESA tarafından uzaya yeryüzünü ve okyanusları gözlemlemesi amacıyla gönderilmişlerdir. Gönderilen ilk uydu olan ERS-1 1991 yılının Temmuz ayı iken programın ikinci uydusu ERS-2 Nisan 1995 yılında yörüngesine oturtulmuştur. ERS-1 uydusu görevini 1999 yılında tamamlarken ERS-2 2011 yılına kadar görevini sürdürmüştür. Dünya üzerinde gerçekleşen doğal afetleri ve buna benzer doğa olaylarını görüntüleyip kayıt altına alınmasını sağlamıştır. Üzerinde çok sayıda algılayıcı bulunan bu uydular farklı amaçlarla kullanılmaktaydı. SAR algılayıcısı da ERS uydularının temel görüntüleme aracı olmuştur. Projede kullanılan veriler ERS-1 verilerinden oluşmaktadır. Bu veriler açık kaynaklı olarak ESA'nın internet sitesinden akademik amaçla alınabilmektedir[28].



Şekil 2.5 : Örnek ERS-1 SAR Görüntüsü

Algoritmanın yazılım üzerinde gerçeklenmesi sırasında kullanılan SAR parametre değerleri Çizelge 2.1'de verilmiştir. Tablodaki veriler ESA tarafından paylaşılan dosyanın içinden okunabilmektedir. Aynı zamanda kullandığımız parametre değerlerini göstermektedir.

Çizelge 2.1 : ERS-1 SAR Parametre Değerleri

f_s	τ_p	k	f	λ	DTF	f_{DC}
18.962468 MHz	37.12 ms	4.1778e+11 Hz/s	5.3 GHz	0.057 m	1679.9 Hz	305 Hz

f_s: Menzil örnekleme hızı

τ_p:Menzil darbe süresi

k: Cıvıltı eğimi

f: Frekans

λ: Dalgaboyu

DTF: Darbe tekrarlama frekansı

f_{DC}: Doppler merkez frekansı

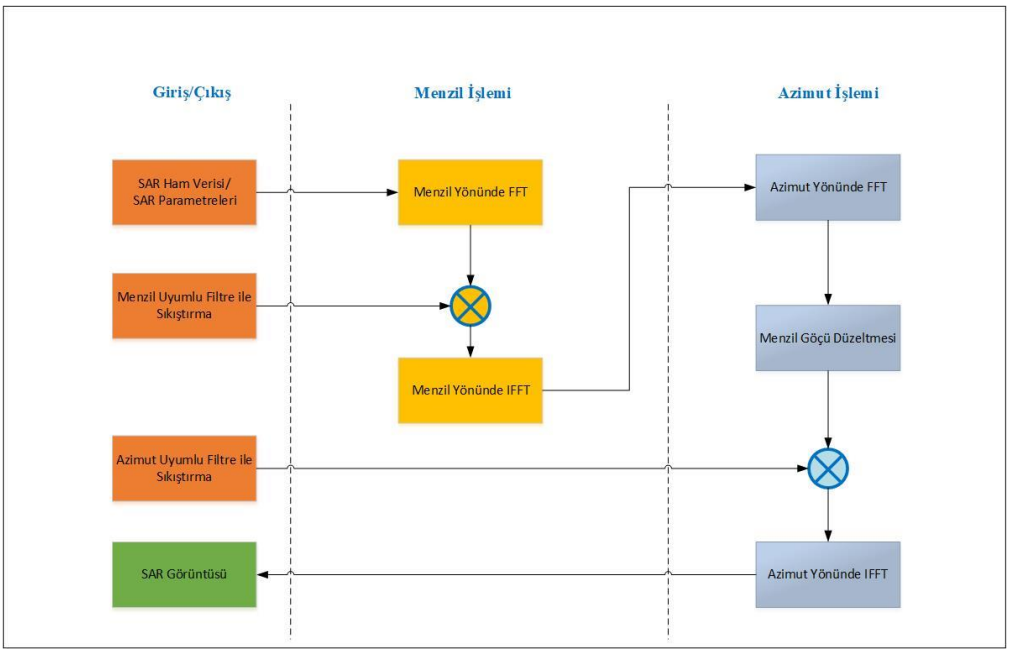
2.4 Menzil-Doppler Algoritması

İlk olarak SEASAT uydusu verilerinin kullanılması için kullanılan Menzil-Doppler algoritması daha sonraları, uydu görüntülerinin işlenmesinde bir çok sistemde kullanılmaya başlamıştır ve günümüzde de çok yaygın şekilde kullanılmaktadır[7].

SAR'da alınan işaretlerin enerjileri menzil ve azimut yönünde dağılmışlardır. Bu sebeple SAR işleme iki boyutlu bir problemi çözmesi gerekir. SAR Algoritmaları da her iki boyuttan alınan bu işaretleri belirli çözünürlük oranında piksellere ayırıp görüntü oluşturmayı sağlar. Menzil-Doppler Algoritması ise bu problemleri her iki boyutta ayrı ayrı işleyerek görüntüyü oluşturur. Sonraki bölümlerde çözünürlük oranlarının hesaplanması ve uygulanan işlemler açıklanacaktır.

Alınan işaretlerde menzil boyutunda enerji gönderilen işaretin(Chirp) süresince azimutta ise oluşturulan sentetik açıklık boyunca veya azimut ekseni boyunca dağılır. Azimut ekseni boyunca yaptığı bu hareket sonucu menzil hücre göçü problemi ile karşılaşılır. Bu iki problem Menzil-Doppler Algoritmasının çözdüğü problemlerdir [35].

Menzil-Doppler Algoritmasının akış diyagramı Şekil 2.6'da gösterilmektedir.



Şekil 2.6 : Menzil-Doppler akış diyagramı

Menzil-Doppler Algoritması, ilk olarak sistem parametreleriyle birlikte işlenmemiş verilerin okunmasıyla başlar. Okunan bu veriler bir sonraki adımdaki işlemlere uygun şekilde düzenlenerek menzil işleme adımına verilir. Menzil işleminde ilk olarak işlenmemiş(ham) verilerin FFT'si alınır ve frekans domenine geçilmiş olur. Önceki bölümlerde açıklandığı gibi donanımda gerçekleme kolaylığı açısından büyük fayda sağlayan bu işlemin ardından uyumlu filtre ile çarpım işlemi yapılır. Menzil uyumlu filtre oluşturulması ilk adımda hesaplanarak sisteme hazır olarak sunulabilir. Menzil uyumlu filtre oluşturulması ilerleyen bölümlerde gösterilecektir. Bu adımın ardından Ters Fourier dönüşümü uygulanarak zaman domenine tekrardan geçilmiş olur.

Bu adımın ardından azimut boyutunda tekrardan Fourier dönüşümü uygulanarak frekasn domenine geçilir ve azimut işleme kısmı başlatılır. Frekans domeninde olan veriye menzil hücre göçü düzeltmesi uygulanır. Bu adımda veriye faz çarpımı ve bazı yaklaşıklıklar yapılarak menzil göçü problemi aşılır. Menzil göçü düzeltildikten sonra, uyumlu filtre ile çarpma işlemi gerçekleştirilir. Menzil işleminde olduğu gibi azimut uyumlu filtre sistem parametreleri kullanılarak hazırlanır ve hücre göçü düzeltilmiş veriyle çarpılır. Ardından Ters Fourier Dönüşümü uygulanır ve zaman domenine dönülür ve algoritma tamamlanır. Bunun ardından çeşitli yöntemler ve araçlar kullanılarak görüntü oluşturulur [41] [42].

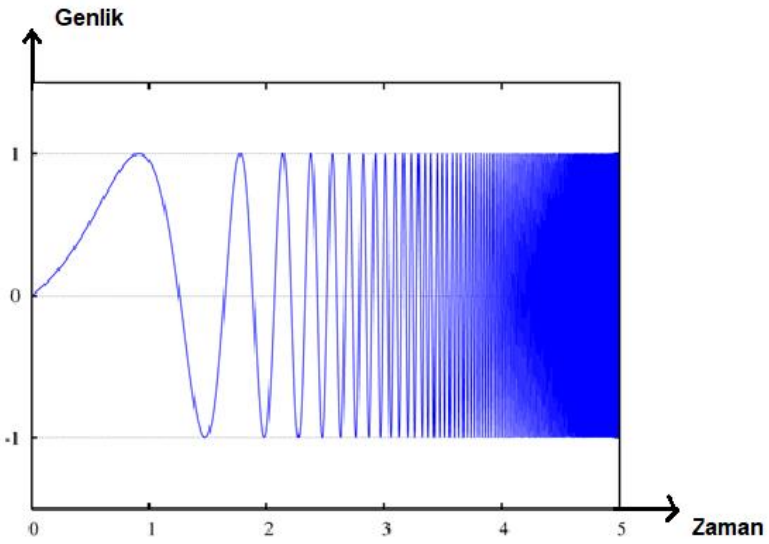
2.4.1 Menzil sıkıştırması

Menzil sıkıştırması; menzilde Fourier dönüşümü, frekans domeninde uyumlu filtre(matched filter) ile çarpma ve menzilde ters Fourier dönüşümü ile sonlanır.

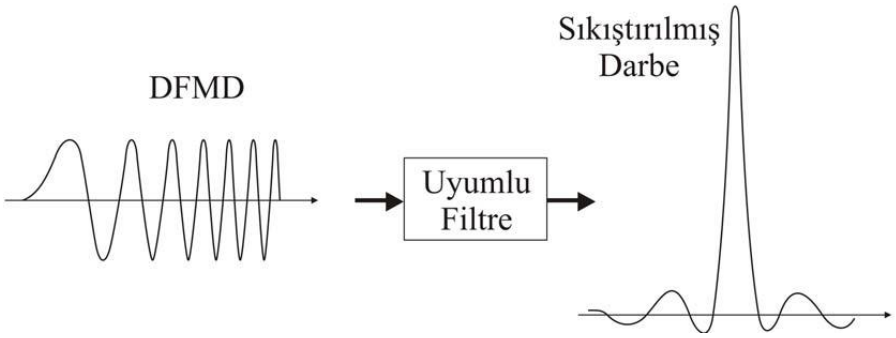
Kısa darbelerde tepe gücü azaltmak için uzun frekans modüleli bir cıvıltı sinyali gönderilir. Daha sonra bu sinyal gönderilen yüzeyden yansıyarak gönderilen cıvıltı işaretiyle konvolüsyon işlemiyle birlikte kompleks bir cevap oluşturur. Menzil sıkıştırma işleminde ise konvolv olan bu cıvıltı işaretini tekrar ters konvolüsyon işlemi ile geri döndürüp yansıyan yüzeyin cevabı alınmış olur. Menzi Doppler algoritmasında ise bu işlemler frekans domeninde yapılır. Yani konvolüsyon ve ters konvolüsyon işlemleri çarpma işlemine dönüşür. Anlatıldığı gibi bu işlemler Menzil Doppler algoritmasının ilk adımını oluşturur.

$$s(t) = e^{i\pi k t^2} \qquad | \, t \, | < \tau_p/2 \tag{2.1}$$

Denklem 2.1'de verilen ifade ERS uyduları için gönderilen cıvıltı darbesinin formülünü göstermektedir. Bu ifadenin kompleks eşleniği alınır ve yansıyan işaret ile eşlenik fade çarpılır ve gönderilen cıvıltı ifadesi yansıyan cevapta görülmez ve yansıyan yüzeyin darbe cevabı menzil boyutunda görülmüş olur. Bu işlemin ardından ters Hızlı Fourier işleminin ardından sinyal zaman domenine döner ve menzil sıkıştırma tamamlanır[43]. Aşağıda Şekil 2.7'de cıvıltı sinyali verilmiştir.



Şekil 2.7 : Cıvıltı sinyali[44]



Şekil 2.8 : Cıvıltı sinyaline uyumlu filtre uygulanmasıyla oluşan işaret [34]

2.4.2 Menzil hücre göçü düzeltmesi

Bir nokta hedefi, sentetik açıklıktan geçerken hiperbolik şekilli bir yansıma olarak görünecektir. Ek olarak, eliptik bir yörünge ve dünyanın dönüşüne bağlı olarak belirgin bir doğrusal kayma olabilir. Başka bir deyişle, hedef menzil hücresinde doğrusal bir eğilimle hiperbol olarak göç edecektir. Bundan dolayı, bir menzile ait noktasal yansıtıcıya ait veri komşu menzil hücrelerine de taşmaktadır. Doğru bir SAR işaret işleme yapabilmek için bu problemin giderilmesi gerekmektedir. SAR uygulamalarında Doppler merkez frekansının kestirimi ile birlikte RCMC işlemi en önemli adımlardandır. Bu göç yolunun şekli, kesin yörünge bilgisinden hesaplanır. 2.2'de verilen denklemdeki gibi genel bir ifadeyle hücre göçü ifadesi verilebilir. DR(s) düzeltilmiş mesafeyi ifade ederken $R(s)$ menzil uzaklığı R_c ise görüntülenen alanın merkezinin menzil uzaklığıdır[40].

$$DR(s) = R(s) - R_c \tag{2.2}$$

Menzil hücre göçünün düzeltilmesi işlemi, menzil işleme adımı gerçekleştirilip azimut doğrultusunda FFT'si alınmış veriye uygulanmaktadır. Menzil hücre göçünün düzeltilmesi, azimut boyunca Fourier Dönüşümü, faz çarpanı ve menzil boyunca Ters Fourier Dönüşümü ile gerçekleştirilebilmektedir[35].

2.4.3 Çapraz menzil sıkıştırması

Menzil sıkıştırması ve menzil hücre göçü düzeltmesi işlemleri ardından SAR işaretine çapraz menzil sıkıştırması işlemi uygulanır. Menzil hücre göçü düzeltmesinden sonra veriyi azimut yönünde odaklamak için uyumlu bir filtre uygulanır. Çapraz menzil

sıkıştırmasının amacı, SAR platformunun geometrisi nedeniyle çapraz menzil yönünde oluşan cıvıltı işaretini gönderilen işaretten ayırt etmektir. Bunula ilgili denklemler 2.3, 2.4 ve 2.5'te verilmiştir. 2.3'te verilen denklem RCMC işleminin ardından oluşan işareti ifade eder[27].

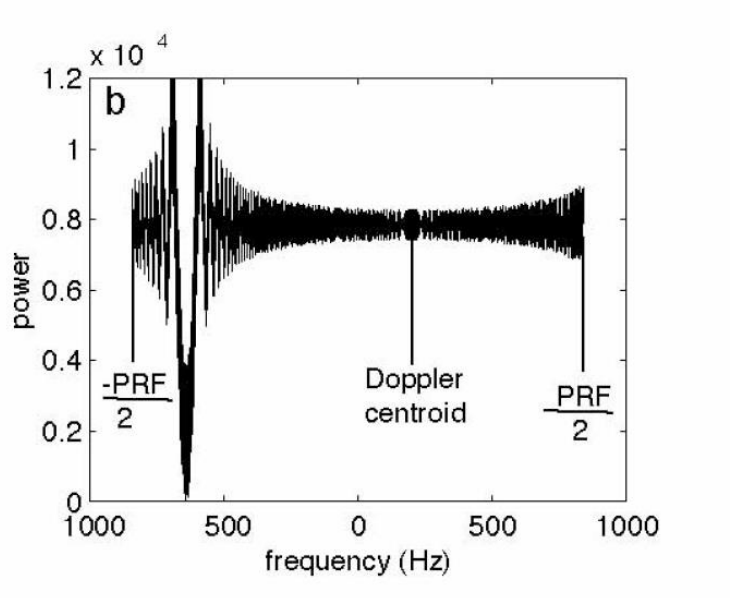
$$H(s) = exp\{4\pi R_0/\lambda\} exp\left\{i2\pi\left[-\frac{2V}{\lambda}.\frac{(x-s_0V)}{R_0}(s-s_0) + \frac{2V^2}{\lambda R_0}(s-s_0)^2/2\right]\right\} \quad (2.3)$$

Bu ifade belirli matematiksel yaklaşımlar kullanılarak sadeleştirilmiştir ve son olarak elde edilen karmaşık faz ifadesinde Doppler merkez frekansı ve Doppler frekans hızı da bulunmaktadır. H(s) fonksiyonunda 2.terimin $(s - s_0)$ elemanlarının katsayıları Doppler frekansları hakkında bilgi vermektedir.

$$f_{DC} = -\frac{2V}{\lambda}.\frac{(x-s_0V)}{R_0} \quad (2.4)$$

$$f_R = \frac{2V^2}{\lambda R_0} \quad (2.5)$$

İlk katsayı doppler merkez frekansını ifade eder ve platform ile hedef yansıtıcının aynı hizada olduğu zamandaki frekanstır. Bu frekans genellikle [-PRF/2, +PRF/2] aralığında yer almaktadır.2.4 denkleminde verilmiştir. İkinci katsayı ise doppler frekans hızı olarak tanımlanır. Bu değerler tasarlanan sistemin parametrelerini içerir. V ifadesi platformun hızı λ dalgaboyunu, R_0 görüntülenen alanın merkezinin platforma olan uzaklığını ifade eder. Doppler frekans hızı ise 2.5 denkleminde verilmiştir.



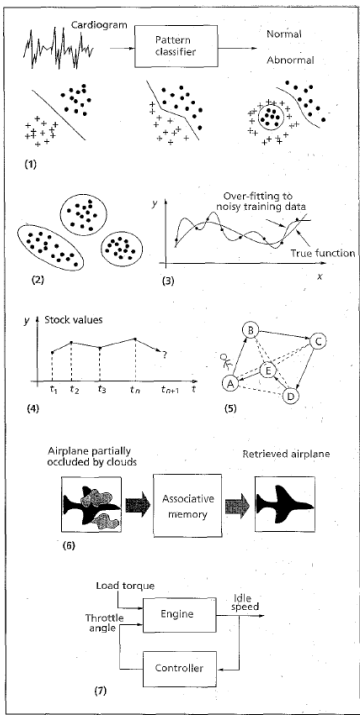
Şekil 2.9 : Doppler merkez frekansı gösterimi[42]

Çapraz menzil sıkıştırması işlemi de yine menzil sıkıştırması aşamasında yapıldığı gibi uyumlu filtreler ile gerçekleştirilir. Frekans domeninde olan veri oluşturulan bu filtre ile çarpıldığında işarette sadece darbe cevabı kalır. Bu işaret ters Fourier dönüşümü ile zaman domenine geçirilir ve algoritma tamamlanır. Dikkat edilmesi gereken bir nokta Doppler frekansının doğru belirlenmesi Menzil Doppler algoritması için büyük önem arz etmektedir. Görüntü kalitesini önemli ölçüde etkiler ve uygun şekilde hesaplanmazsa sonuçta da istenen tanılırlıkta SAR görüntüsü elde edilemez[27]. Çapraz menzil sıkıştırma işlemi ardından görüntülenen alanlar çizdirme yöntemleriyle görüntülenebilir. Bu aşamadan sonra görüntü iyileştirme teknikleri ile görüntü çıktı olarak alınabilir.

3. YAPAY SİNİR AĞI İLE KARAR MEKANİZMASI OLUŞTURULMASI

3.1 Yapay Sinir Ağları

Yapay sinir ağları, insan beyninde uzun evrimler sonucu oluşan öğrenme yeteneği, genelleme yeteneği, adaptasyon, doğal bilgi işleme, hata toleransı gibi yetilerin incelenmesi ve bu yetilerin bilgisayarlar tarafından taklit edilebilmesini sağlamaya çalışmaktadır. Bu bağlamda insan beynindeki nöronların yapısı ve işleyişi incelenerek yapay nöron ve yapay ağ tanımları ortaya konulmaktadır. Bu ağ yapıları genel olarak nöronlar arasındaki haberleşmenin nasıl olacağını belirten matematiksel kurallardan oluşmaktadır. Tanımlanmış bir yapay ağ yapısı kullanılarak var olan yaklaşımlarla çözüm bulamadığımız çeşitli problemlere çözüm getirilmeye çalışılmaktadır. Bu problemler desen sınıflandırma, kümeleme, fonksiyon yaklaşımı, tahmin, optimizasyon, içerik ilişkili hafıza ve kontrol problemleri olarak gruplanabilir[45].



Şekil 3.1 : Yapay sinir ağlarının çözebildiği problemler [30]

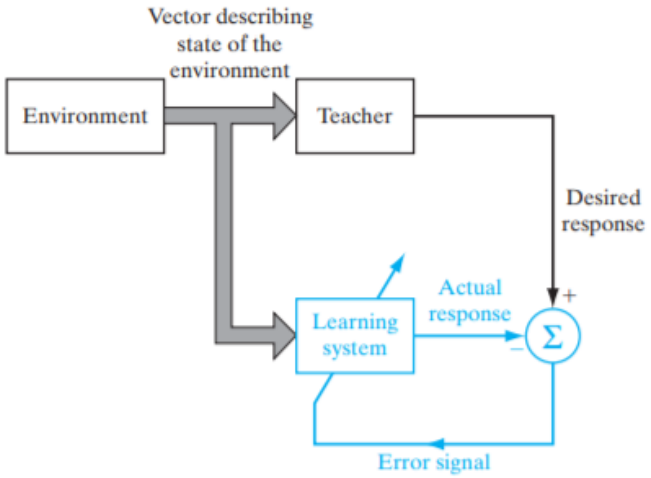
(1) desen sınıflandırma (2) kümeleme (3) fonksiyon yaklaşımı (4) tahmin

(5) optimizasyon (6) içerik ilişkili hafıza (7) kontrol

Bir yapay sinir ağı eğiticili ve eğiticisiz olmak üzere iki farklı şekilde eğitilebilir.

3.1.1 Eğiticili öğrenme

Eğiticili öğrenme[8], yapay sinir ağının eğitimini çıktısı bilinen girdiler ile yapmasıdır. Bu sayede bir öğretmen tarafından sonucun doğruluğu test edilip doğruluk oranı arttırılmaya çalışılır ve probleme en uygun nöron yapısı elde edilir. Eğiticili öğrenme, büyük bir problemi temsil edebilme yetisine sahip bilinen bir veri dizisi bulunan yapılarda kullanılır. Bu yapılara desen sınıflandırma, fonksiyon yaklaşımı ve tahmin yapıları örnek gösterilebilir[8].



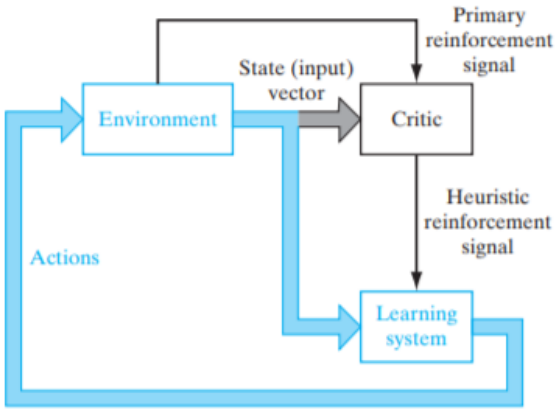
Şekil 3.2 : Eğiticili öğrenme [8]

3.1.2 Eğiticisiz öğrenme

Eğiticisiz öğrenme[8], herhangi bir öğretmen yapısı bulundurmaz. Ağ, girdileri, belli bir kural etrafında anlamlandırmaya çalışır. Örneğin bu kural yarıştırmalı öğrenmede kazanan herşeyi alır stratejisi ile girdilerin bir çıktıya ulaşmak için defalarca deneme yapması ve bu denemelerde çıktıya en yakın bitiren girdinin kazanarak sistemi şekillendirmesi olarak tanımlandırılır. Deneme sayısı, denemelerin hızı, kabul edilen hata oranı gibi ağın yapısını belirleyen sabit değerler her problem için farklı tanımlanır. Eğiticisiz öğrenme, bir öğretmenin olmadığı problemlerde kümeleme yapısı gibi yapılarda kullanılır[8].

3.1.3 Pekiştirmeli öğrenme

Pekiştirmeli öğrenme[8] temelinde bir eğiticisiz öğrenmedir. Bir öğretmen bulunmaz fakat ağ kendi kendisinin değerlendirmesini yaparak içinde bulunduğu ortamı tanımlar.

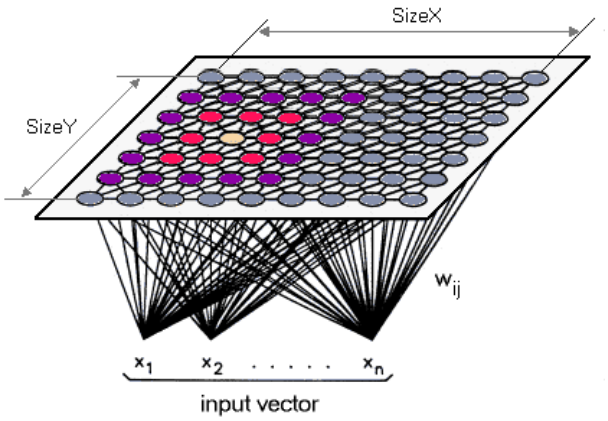


Şekil 3.3 : Pekiştirmeli öğrenme akış diyagramı [8]

3.2 Özdüzenlemeli Ağ

Öz düzenlemeli Ağ(Self Organizing Map- SOM)[14], 1982'de Finlandiyalı profesör ve araştırmacı Dr. Teuvo Kohonen tarafından öncülük edilen, girdi ve çıktı uzayları arasında bir topolojinin korunmasının önemli olduğu uygulamalara yönelik, kendi kendine haritalama yapan eğiticisiz bir öğrenme modelidir. SOM, giriş değerleri arasında birbirine en yakın olan verileri n sayıda ağırlıklarla (nöronlarla) temsil edebilmemizi sağlar. Bu ağırlıklar genelde 2 boyutlu bir ağ yapısıyla birbirlerine komşuluk oluşturur bu sayede her bir nöronun değişimi ona komşu olan diğer nöronları da etkileyerek oluşan ağ nöron yapısının düzenli bir şekilde dağılmasını sağlar.[14]

Öz düzenlemeli ağ, karmaşık veya büyük miktarlarda yüksek boyutlu veriyi görselleştirmek ve aralarındaki ilişkiyi, verilen etiketlenmemiş verinin herhangi bir yapısı olup olmadığını görmek için düşük boyutlu bir alanda temsil etmenin gerekli olduğu uygulamalarda tercih edilmektedir. Bir SOM yapısı giriş ve çıkış olmak üzere iki katmandan oluşur. Çıkış katmanı nöronlardan giriş katmanı ise eğitim kümesinden oluşur. Nöronların boyutları ile giriş boyutları birbirine eşittir ve her giriş değeri bütün nöronlara bağlıdır. Başlangıç nöron değerleri rasgele ve küçük seçilir.[46]



Şekil 3.4 : Öz düzenlemeli Ağ [47]

3.2.1 Özdüzenlemeli ağ algoritması

Algoritma ağın oluşturulmasıyla başlar. Ağ, giriş ile aynı boyutta olacak şekilde nöronlar içererek N adet nöronla oluşturulur. Eğitim kümesindeki her veri ağda kendisini en iyi temsil edecek nöronu bulmaya çalışır. Bunu için kendisine en yakın olan nöronu ve onun komşularını kendisine daha da yaklaştırır. Kendisine en yakın olan nöronu bulurken en çok kullanılan metodlardan biri minimum öklid uzaklığı[48] olarak bilinir[46]. Her iterasyonda bütün eğitim kümesi (X) için aşağıdaki adımlar tekrarlanır. Eğitimi sırasında ağın eğitim kümesini ezberlemesinden kaçınmak için her iterasyonda eğitim kümesinin sırası rasgele karıştırılabilir. Öğrenmenin tamamlandığını sorgulamak için son iterasyonlardaki girişlere karşılık gelen kazanan nöronların değişimine bakılır. Eğer K iterasyon boyunca değişim görülmemişse eğitim sonlandırılır. Minimum öklid uzaklığı aşağıdaki gibi tanımlanmıştır.

$$\boldsymbol{v^{[k]}} = \min_{\boldsymbol{n = 1, 2, 3, \ldots N;\ \forall\, i \in X}} \left(\boldsymbol{x_i} - \boldsymbol{W_n^{[k]}} \right)^2 \tag{3.1}$$

$\boldsymbol{x_i}$ i. giriş vektörü, $\boldsymbol{v^{[k]}}$ k. iterasyondaki i. giriş vektörüne göre kazanan nöron, $\boldsymbol{W_n^{[k]}}$ ise k. iterasyondaki n. nörondur. Her giriş vektörünün kazanan ağırlığı bulunduktan sonra nöronlar aşağıdaki formüle göre güncellenir.

$$\boldsymbol{W^{[k+1]}} = \boldsymbol{W^{[k]}} + \boldsymbol{\alpha^{[k]} H_v^{[k]}} \left(\boldsymbol{x_i} - \boldsymbol{W^{[k]}} \right) \tag{3.2}$$

$\boldsymbol{\alpha^{[k]}}$ k. iterasyondaki öğrenme hızıdır. $\boldsymbol{H_v^k}$ kazanan nöronun komşuları üzerindeki etkisini gösteren fonksiyondur. Genelde Gauss fonksiyonu[49] kullanılır.

$$H_v^{[k]} = \exp\left(-\frac{d^2(v^{[k]})}{2(\sigma^{[k]})^2}\right) \quad (3.3)$$

$d(v^{[k]})$ mevcut giriş için k. iterasyondaki kazanan nöronun diğer nöronlara olan uzaklığını (komşuluğunu) veren fonksiyondur. $\sigma^{[k]}$ k. iterasyondaki Gauss fonksiyonu için kullanılan sigma değeridir. Sigma değeri azaldıkça nöronların komşuluk bağları azalarak tek başlarına hareket etmesi sağlanır.

Öğrenme hızı ve sigma değeri yüksek bir değerden başlatılarak her iterasyonda azaltılır. Bu sayede ağın önce yapısının oluşması daha sonrasında ise nöronların detaylı bir şekilde temsil yerlerine yerleşmesi sağlanır. Her iterasyondaki azalma denklemleri aşağıdaki gibi tanımlanır.

$$\sigma^{[k]} = \sigma^{[0]} \exp\left(-\frac{k}{T_\sigma}\right) \quad (3.4)$$

$$\alpha^{[k]} = \alpha^{[0]} \exp\left(-\frac{k}{T_\alpha}\right) \quad (3.5)$$

T_σ ve T_α sabit değerleri azalma oranlarının büyüklüğünü verir.

3.2.2 Öz düzenlemeli ağın eğitimi

Bir özdüzenlemeli ağın eğitimini yapmak içi gereken adımlar asağıdaki gibidir:

1- Bütün sistemi temsil edebilen bir eğitim kümesi seçilir.

2- Ağın kaç boyutlu olacağı ve kaç nöron içereceği belirlenir.

3- Belirlenen ağda bulunan her nöronun diğer nöronlara olan komşuluğu (d fonksiyonu) hesaplanır.

4- Ağırlıkların başlangıç değerleri sistemin sınırları içersinde olacak şekilde rasgele ve birbirine yakın verilir.

5- Algoritmadaki sabitler öğrenme hızı, komuşuk katsayısı, öğrenme hızı değişimi, komşuluk katsayısı değişimi ($\alpha, \sigma, T_\alpha, T_\sigma$) her problem için farklı değerlerle belirlenir.

Bu adımlar gerçeklendikten sonra yerleştirme ve öğrenme aşaması olmak üzere iki aşamada algoritma gerçekleştirilir. Yerleştirme aşamasında nöronların sistemi temsil edebilecek şekilde dağılmaları sağlanmalıdır. Öğrenme aşaması ise nöronların komşularıyla olan bağının ve öğrenme hızının azaltılarak bulundukları konumda

detaylı bir yer saptamaları sağlanmalıdır. Yerleştirme aşamasından öğrenme aşamasına geçmek için bir koşul konulmalıdır.

Bu koşul K iterasyon arka arkaya kazanan ağırlık vektörünün değişmemesi olabilir. Öğrenme aşamasında ise zaten her iterasyonda azaltılmış olan öğrenme hızı ve komşuluk katsayısı azaltılmaya devam edilerek A iterasyon daha algoritma devam ettirilir.

Eğitim sonunda nöronların dağılımının uygun olup olmadığı kontrol edilerek eğitim aşaması bitirilir.

3.2.3 Öz düzenlemeli ağın testi

Öz düzenlemeli ağın testi, girişlerin kendisine en yakın nöronu bularak o nöronu çıkış olarak vermesidir. Bu sayede girişler kendisini en yakın olduğu nöron grubuna dahil ettiğini göstermiş olur. Test uygulanırken algoritmada kullanılan en yakını bulma yöntemi kullanılır. Yapılan eğitim algoritmasında Minimum Öklid Uzaklığı kullanıldığı için test algoritmasında da aynı yöntem kullanılacaktır. Minimum Öklid Uzaklığı aşağıdaki gibi tanımlanır.

$$\boldsymbol{v^{[k]} = \min(x - W_n)^2} \quad \boldsymbol{n = 1, 2, 3, \ldots N;} \tag{3.6}$$

Öz düzenlemeli ağın testi yapılırken seçilen yöntem her giriş için kullanılarak girişin kendisine en yakın nöronu bulması sağlanır. Bulunan nöron o girişi temsil eden nöron olarak çıkışa verilir.

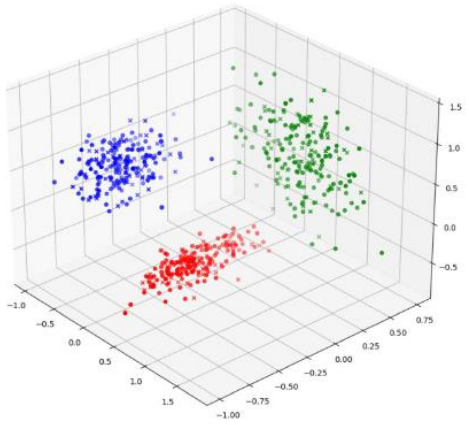
3.3 Öz düzenlemeli Ağ İle Sınıflandırma Örnekleri

Sınıflandırma, bir grup veriyi bazı özellikler bakımdan ayırarak birden çok sınıf oluşturmaktır. Özdüzenlemeli Ağ yapısı benzer verileri temsil eden bir nöron oluşturduğu için sınıflandırma problemlerinde oldukça uygun bir kullanıma sahiptir. Özdüzenlemeki Ağın hızlı, doğru ve görselleştirilmesi kolay olması sebebiyle sınıflandırma problemlerinde oldukça kullanışlıdır. [50]

3.3.1 Üç boyutlu düzlemde sınıflandırma

Üç boyutlu bir düzlemde Öz düzenlemeli ağı kullanarak sınıflandırma yapmak için 3 farklı sınıfa ait Gauss dağılımlı rasgele veri öbekleri oluşturuldu. Veriler oluşturulurken belirlenen Gauss dağılım parametreleri aşağıdaki gibidir :

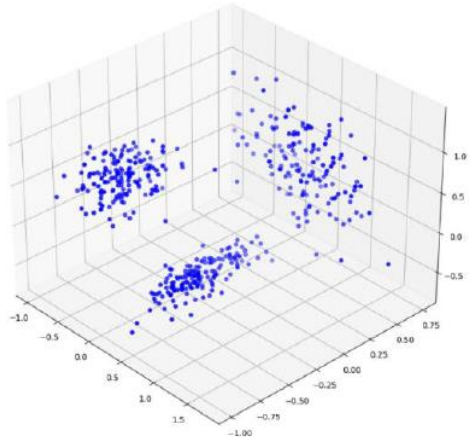
- A grubu için: $\sigma = (0.1,0.25,0.1)$, Ortalama = $(0.3,-0.3,-0.5)$
- B grubu için: $\sigma = (0.2,0.2,0.2)$, Ortalama = $(-0.5,-0.5,0.5)$
- C grubu için: $\sigma = (0.4,0.1,0.4)$, Ortalama = $(0.5,0.5,0.5)$



Şekil 3.5 : Üç boyutlu düzlemde gauss dağılımlı veriler

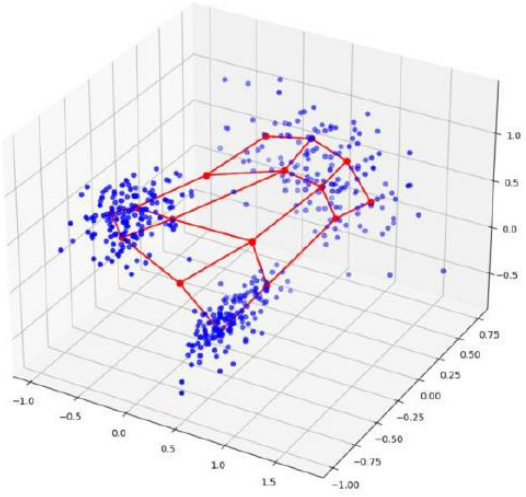
3.3.1.1 Eğitim

Her sınıfta 200, toplamda 600 adet oluşturulan verilerin %75'i eğitim için geri kalanı ise test için rasgele seçilerek ayrıldı. Eğitim seti Öz düzenlemeli ağı eğitebilmek için sınıf bilgisinden bağımsız hale getirildi.



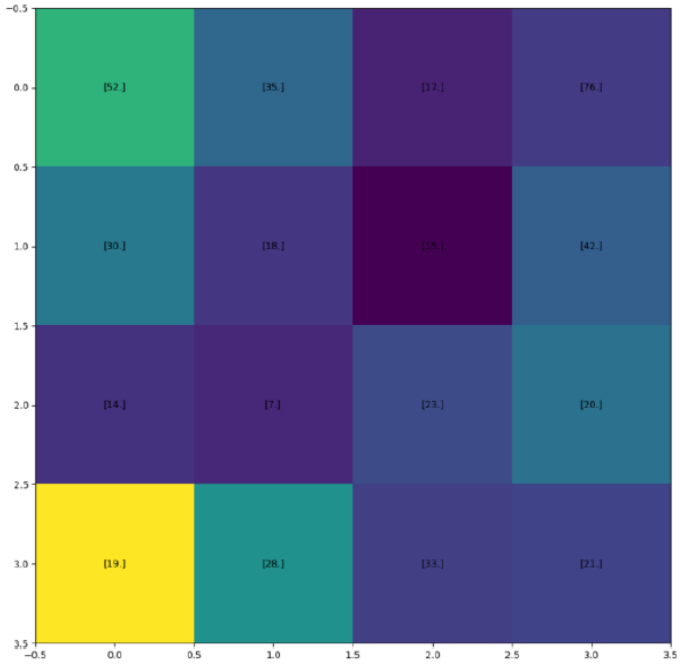
Şekil 3.6 : Üç boyutlu düzlemde eğitim seti

Eğitim için 2 boyutlu toplamda 16 nöron bulundurak 4x4 lük bir ağ yapısı belirlendi. Başlangıç ağırlık değerleri -0.25 ile 0.25 arasında rasgele olarak seçildi. Öğrenme hızı 0.1, komşuluk katsayı değeri 0.8, öğrenme hızı değişimi 10000, komşuluk katsayısı değişimi 5000 olarak seçilerek Öz düzenlemeli ağ ile eğitim yapıldı. Seçilmiş olan bu değişim katsayıları sayesinde 1000 iterasyonda komşuluk katsayısı yaklaşık %20, eğitim hızı ise yaklaşık %10 azalmaya uğramış oldu. Eğitim 2000 iterasyon sonucunda bitti.



Şekil 3.7 : Üç boyutlu düzlemde eğitim sonucu

Şekil 3.7'de görüldüğü gibi Öz düzenlemeli ağ nöronları en iyi temsil edebilecek şekilde bütün verilere dağılmış bir şekilde oluştu. Daha dağınık olan C grubunu 8 nöron temsil ederken diğer iki grubu 4'er nöron temsil etmiş oldu. Dağılımı daha iyi görebilmek adına eğitim sonucu nöron başına düşen kazanan ağırlık tablosu Şekil3.8'de verilmiştir. Tablo incelendiğinde sınıfların merkezine yakın olan nöronların, sınıf aralarında kalan nöronlara göre daha çok kazanan veriye sahip olduğu görülmektedir.



Şekil 3.8 : Üç boyutlu düzlemde eğitim sonucu kazanan ağırlık tablosu

3.3.1.2 Benzetim sonuçları

Yapılan eğitimin ardından test kümesi için ayrılan verileri ağa göre hangi sınıfa ait oldukları bulunup gerçek değerlerle karşılaştırıldı. Karşılaştırma sonucunda %98.6 oranında doğru sonuç bulundu.

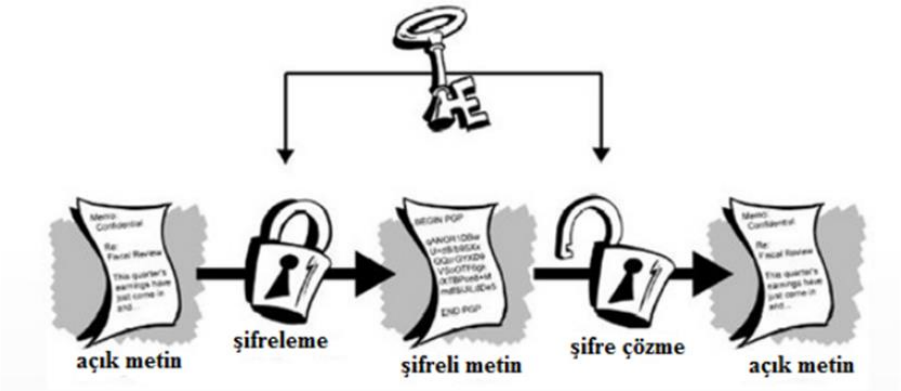
4. KRİPTOLOJİ

4.1 Kriptografi

İnsan tarihi boyunca güvenli veri iletimi ihtiyacı şifreleme algoritmaların tasarlanmasına neden olmuştur. Yunancada; “kriptos: gizli” ve ”graphi: yazı” anlamına gelmektedir[18]. Kriptografi bilgi güvenli amacıyla verinin şifrelenmesi ile ilgilenirken, Kriptografik bir sistem gizlilik, bütünlük, kimlik doğrulama ve inkâr edilemezlik ilkelerine sahip olmalıdır[51]. Bu ilkelerle verinin istenmeyen kişilerin eline geçmediğinden, bütünlüğünün bozulmadığından, göndericinin ve alıcının kimliklerinin doğrulanabileceğinden ve son olarak gönderen tarafça gönderdiği veriyi inkâr edememesinden emin olmak amaçlamaktadır. Günümüzde kullanılan şifreleme sistemleri, simetrik ve asimetrik anahtarlı şifreleme algoritmaları olarak ikiye ayrılmaktadırlar. Bu projede her iki şifreleme sistemi kullanılmıştır[52].

4.1.1 Simetrik anahtarlı şifreleme

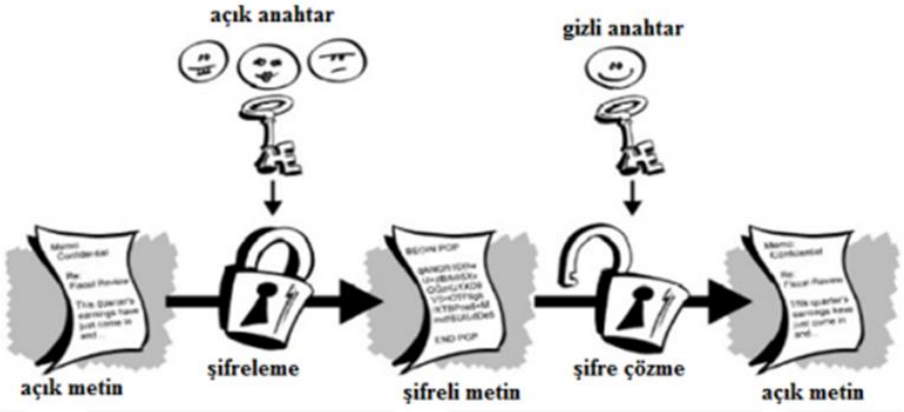
Kriptografik sistemlerde iki farklı anahtar kullanılmaktadır. Simetrik şifreleme[53] ve asimetrik şifreleme[54] olarak ayrılma nedeni kullanılan bu iki anahtar arasındaki ilişkiye dayanmaktadır. Verinin şifrelenmesinde kullanılacak olan anahtar ve şifrelenen verinin çözülmesi için kullanılan anahtar aynı ya da kolayca birbiri tarafından üretilebiliyor ise bu tür şifrelemeye simetrik anahtarlı şifreleme olarak adlandırılmaktadır. Şifrelemede kullanılacak olan anahtar sadece veriyi şifreleyen ve verinin şifresini çözecek taraflarda olmalıdır[55]. Veri Şifreleme Standardı (Data Encryption Standard – DES)[56] ve Gelişmiş Şifreleme Standardı (Advanced Encryption Standard – AES)[20] simetrik anahtarlı şifreleme algoritmalarına örnektir. Bu projede de AES kullanılmıştır.



Şekil 4.1 : Simetrik anahtarlı şifreleme şeması [52]

4.1.2 Asimetrik anahtarlı şifreleme

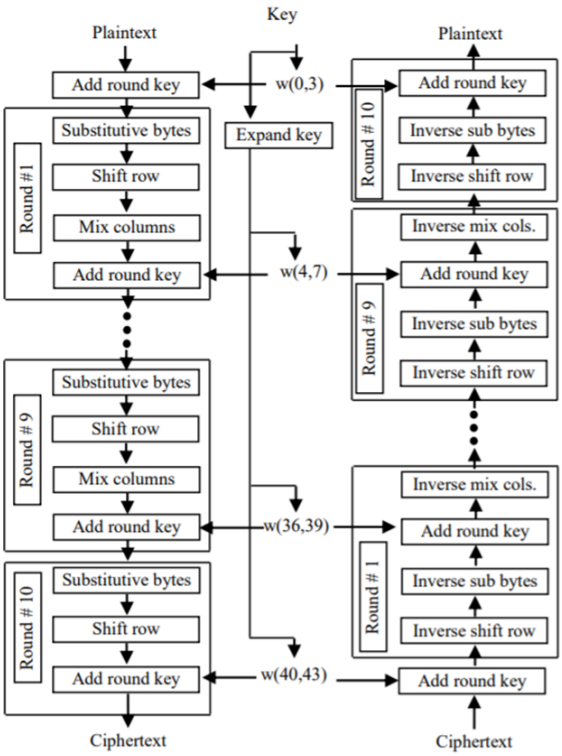
Asimetrik anahtarlı şifrelemede (Public Key Cryptography - PKC) verinin şifrelenmeside ve şifrelenmiş verinin çözülmesinde kullanılan anahtarlar birbirinden farklıdır. Bu anahtarlar gizli anahtar ve açık anahtar olarak adlandırılmaktadır[54]. Açık ve gizli anahtar üretimi arasında matematiksel bir ilişki vardır. Ama açık anahtar kullanılarak gizli anahtarın elde edilmesi pratikte mümkün olmamaktadır[55]. Asimetrik anahtarlı şifreleme algoritmaları matematiksel problemlere dayanmaktadır. Bu projede kullanılan Diffie-Hellman anahtar değişim protokolü[19] Ayrık Logaritma Problemine (Discrete Logarithm)[57] dayanmaktadır[58]. Asimetrik anahtarlı şifrelemenin güvenliği kullanılan matematiksel ifadelerin çözümlenmesine bağlıdır[59]. Bu nedenle algoritmalarda daha uzun veri boyutları kullanılmaktadır. Asimetrik şifreleme algoritmalara simetrik şifreleme algoritmalarına göre daha kısa veri boyutları ile çalıştıkları için daha yavaş çalışmaktadır[60].



Şekil 4.2 : Asimetrik anahtarlı şifreleme şeması [52]

4.2 Gelişmiş Şifreleme Standardı

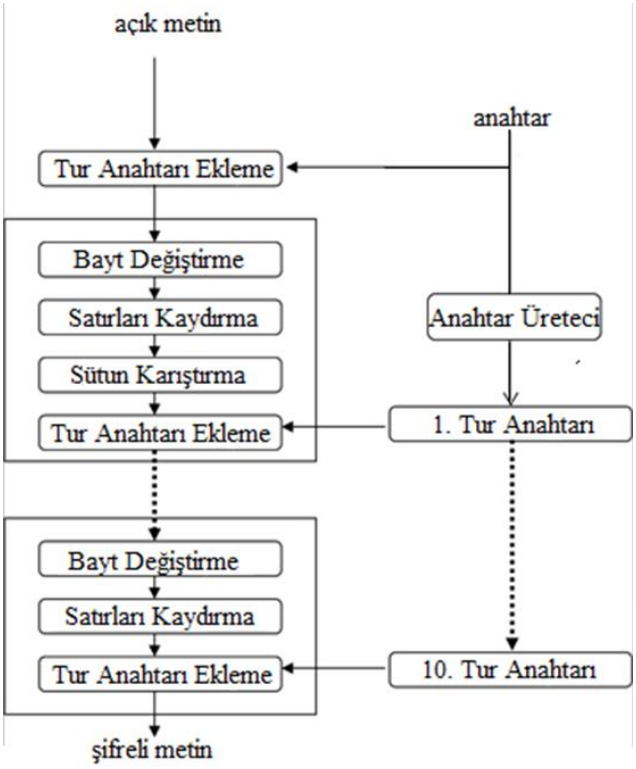
Gelişmiş Şifreleme Standardı (Advanced Encryption Standard – AES)[20] simetrik anahtarlı şifreleme algoritmasıdır. Teknolojinin gelişmesiyle blok uzunluğu 64 bit, anahtar uzunluğu sahip DES[56] algoritması ihtiyacı karşılayamaz olmuştur. Bu durumun üzerine Ulusal Standartlar ve Teknoloji Enstitüsü(National Institute of Standards and Technology-NIST)[61] 1997 yılında DES algoritmasının yerine geçmesi yeni bir blok şifreleme algoritması[62] tasarlama yarışması düzenledi. 2 Ekim 2000 yılında Rijndael algoritması[63], Gelişmiş Şifreleme Standardı adıyla NIST tarafından standartlaştırılmıştır[64]. AES, 128 bit mesaj bloğu boyutuna; 128,192 ve 256 bit olmak üzere üç farklı anahtar boyutuna sahiptir[20]. Anahtar boyutu arttıkça kullanılan döngü sayısı artmaktadır. Son tur hariç her bir turda sırasıyla; bayt değiştirme, satır kaydırma, sütün karıştırma ve son olarak tur anahtarıyla toplama işlemi gerçekleşmektedir. Son turdaki sütün karıştırma işleminin güvenliği arttırmadığı saptandığından dolayı bu turda sütün karıştırma işlemi yapılmamaktadır. Son turda yapılan işlemler bayt değiştirme, satır kaydırma ve tur anahtarıyla toplama şeklindedir[65]. Bu projede AES 128 bit mesaj blok uzunluğu ve 128 bit anahtar uzunluğuna sahiptir. Her bir 128 bit mesaj bloğu 10 turluk bir döngüye girecektir.



Şekil 4.3 : AES şifreleme akış diyagramı [66]

4.2.1 128-Bit Gelişmiş Şifreleme Standardı şifrelemesinin blokları

Bu bölümde AES şifreleme algoritmasının alt blokları adım adım anlatılacaktır.

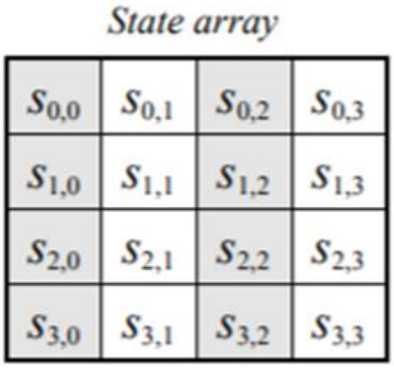


Şekil 4.4 : AES 128 şifreleme algoritmasının genel yapısı [52]

4.2.1.1 Bayt değiştirme

128 bitlik şifrelenecek metin, şifreleme alt modüllerine giriş olarak verilmeden önce 128 bitlik ana anahtar ile xor işlemine tabi tutulmaktadır. Xor işlemi 2 tabanında toplama işlemi olarak tarif edilebilir. Xor işlem çıktısı olan 128 bit uzunluğundaki veri, her bir değeri 8 bit olan 4x4 matrise en değerli bitten başlayarak sırasıyla sütunları dolduracak şekilde yerleştirilir. Bu matrise ‘durum matrisi’[20] ismi verilmektedir (Şekil 4.5). AES şifreleme algoritmasında uygulanacak işlemler bu matris üzerinden ilerleyecektir. Bayt değiştirme işlemi, durum matrisindeki baytların S- Kutusu matrisi yardımıyla farklı baytlara dönüştürülmesi işlemidir. Matematiksel işlemle üretilen S-Kutusu 16x16 boyutunda lineer olmayan sabit bir matristir (Şekil 4.6) [20]. Her bir değeri 8 bit ile ifade edilmektedir. S-Kutusu onaltılık sayı sisteminde 0-FF dâhil 8 bit ile ifade edilebilecek bütün değerlere sahiptir. Bu projede S-Kutusu matematiksel

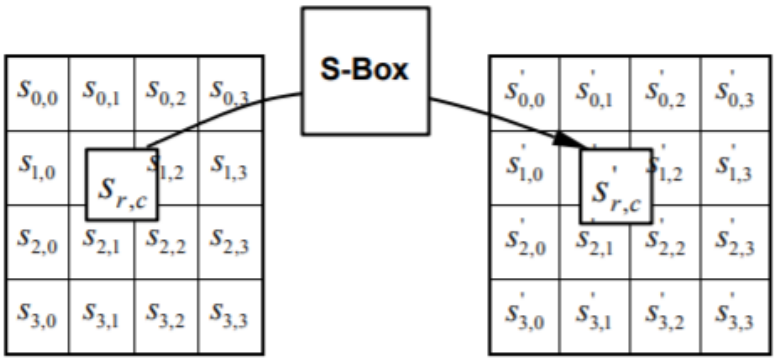
olarak üretilmeyecek sabit matris olarak doğrudan sisteme verilecektir. Bu durumda durum matrisindeki her bir 8 bit 4bit - 4bit olarak ayrı ayrı düşünülmektedir. İlk dört bit S-Kutusundaki satır sayısını, son dört bit ise sütun sayısını belirtmektedir. Kesişimde bulunan değer, durum matrisindeki ilgili elemanın yeni değeridir. Bu işlem her elaman için her döngüde tekrar edilmektedir [67].



Şekil 4.5 : Durum matrisi [20]

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

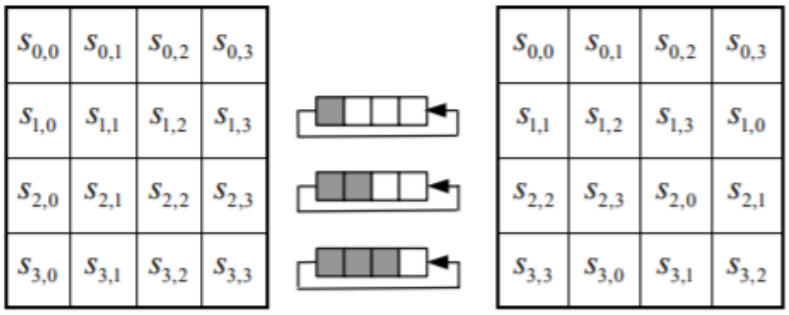
Şekil 4.6 : S-Kutusunun onaltılık sayı sistemindeki değerleri [20]



Şekil 4.7 : Bayt değiştirme işlemi [20]

4.2.1.2 Satır kaydırma

Satır kaydırma işleminde durum matrisinin satırları çevrimsel şekilde kaydırılmaktadır. İlk satırda kaydırma işlemi yapılmaz. İkinci satır sola 1 bayt, üçüncü satır sola 2 bayt ötelenir ve son satır sola 3 bayt ötelenir. Taşan baytlar satırın sonuna eklenmektedir. [20]



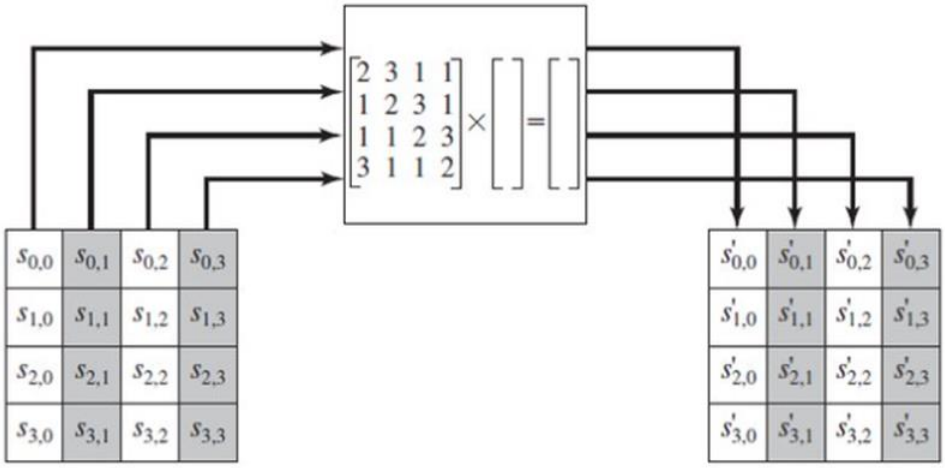
Şekil 4.8 : Satır kaydırma işlemi [20]

4.2.1.3 Sütun karıştırma

Sütun karıştırma alt bloğunda, durum matrisinin sütun elemanları kullanılarak yeni sütunlar elde edilmektedir. Algoritmanın bu adımında durum matrisinin sütunları sırayla matematiksel işleme tabi olmaktadır. A matrisi ile durum matrisi arasında matris çarpma işlemi yapılır. Matris çarpması yapılırken kullanılan ondalık sayılarda çarpma işlemi yerine Sonlu Alan (Galois Field – GF(2^8))'da çarpma işlemi kullanılır. Aynı şekilde ondalık sayılarda toplama işlemi yerine de xor işlemi kullanılır. A matrisi sabit bir matris olup polinomsal gösterimi ve matris gösterimi Şekil 4.9'de aşağıya eklenmiştir.

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \tag{4.1}$$

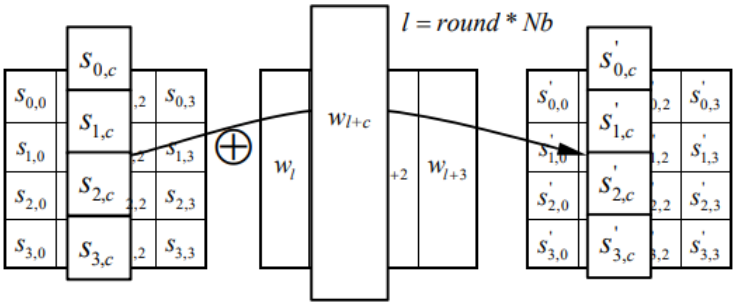
$$s'(x) = a(x) * s(x) \tag{4.2}$$



Şekil 4.9 : Sütun karıştırma işlemi [68]

4.2.1.4 Tur anahtarıyla toplama

Her turda sütün karıştırma işleminin sonucu ile o tur için hazırlanmış olan yeni anahtar xor işlemine tabi tutulur.

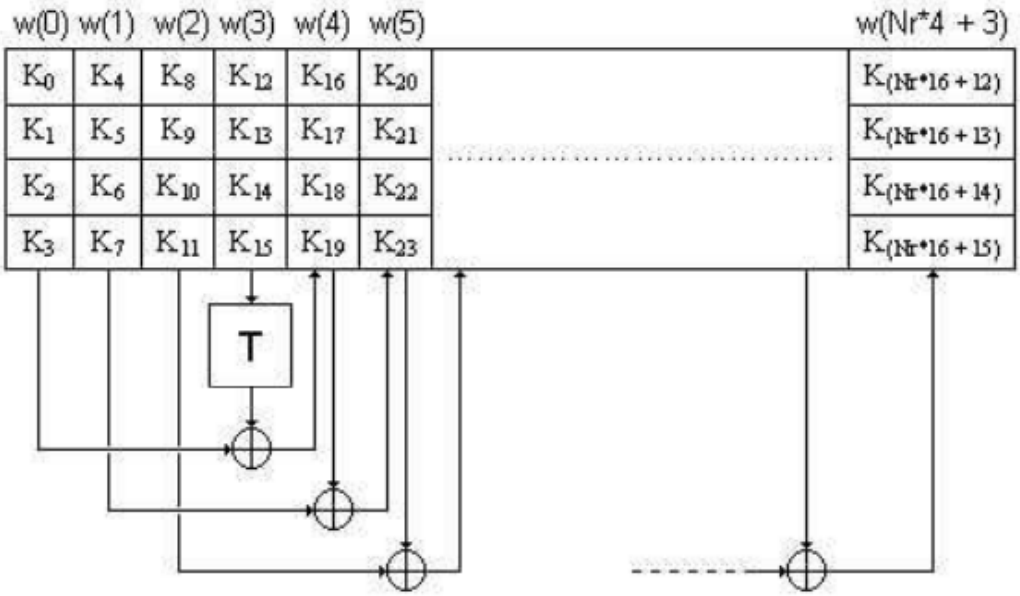


Şekil 4.10 : Tur anahtarıyla toplama işlemi [20]

4.2.2 Anahtar genişletilmesi

Bu projede 128 bit uzunluğunda anahtar kullanılmaktadır. Her tur için ana anahtardan oluşturulan yeni 128 bit uzunluğundaki anahtar kullanılır. Bu nedenle başlangıçtaki ana anahtar genişletilmektedir. Bu genişletme işlemi sonucu 128 bit uzunluğunda 10 adet yeni anahtar üretilmektedir[20]. Şekil 4.11'de gösterilen anahtar genişletilmesi işleminde "T" bloğu, $w(Nr * 4 + 3)$ sütun değerinin bazı işlemlere tabi tutulmuş halini ifade etmektedir. $Nr = 0,1,2 \ldots (Nb - 1)$ değerlerini alabilmektedir. Bu projede 128 bit uzunluğunda anahtar kullanıldığından tur sayısı 10 'dur. Bu tasarım için

$Nb = 10$'dur. $w(Nr * 4 + 3)$ sütunu, bayt kaydırma, bayte değiştirme ve son olarak Round Constant (Rcon) değeri ile xor işlemine girdikten sonraki sonuç T değerini oluşturmaktadır.



Şekil 4.11 : 128 bitlik giriş anahtarı için Anahtar Üreteci **[67]**

4.2.3 128-Bit Gelişmiş Şifreleme Standardı şifre çözme blokları

AES algoritmasında kullanılan bloklar tersi alınabilir işlemlerden oluşmaktadır. Şifre çözme işlemini gerçeklerken iki farklı algoritma kullanılabilir. Bu projede şifre çözme (Inverse Cipher) algoritması kullanılacaktır[20]. Algoritma gereği, anahtar üretme algoritmasıyla üretilen 10 tur anahtarı ters sıralamayla kullanılacaktır. Şifre çözme algoritmasına girmeden önce; şifrelenmiş metin ile xor işlemine girecek olan anahtar, şifrelemenin onuncu turunda kullanılan anahtardır. Şifre çözmede son turda kullanılacak olan anahtar, bütün anahtarların türetildiği ana anahtar olacaktır. Şifre çözme algoritmasında alt blokların kullanım sıralaması şifreleme algoritmasından farklıdır. AES şifre çözme algoritmasının alt blokları kullanma sıralaması; satır kaydırma işleminin tersi, bayte değiştirme işleminin tersi, tur anahtarıyla toplama işlemi ve son olarak da sütun karıştırma işleminin tersidir. Son turda, son aşama olan sütün karıştırma işleminin tersi yapılmamaktadır.

4.2.3.1 Bayt değiştirme işleminin tersi

Bayt değiştirme işleminin tersi olup bayt değiştirme işleminin sonuçlarını ortadan kaldırmak için kullanılır. Bu işlem için sabit olarak kullanılması tercih edilen S-

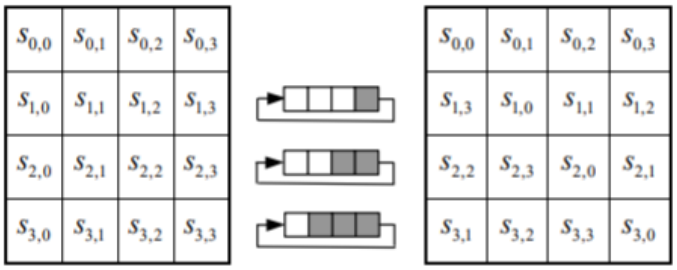
Kutusunun tersi kullanılmaktadır. Aşağıdaki Şekil 4.11'de S-Kutusunun tersi yer almaktadır[20]. Çalışma prensibi bayte değiştirme alt bloğundaki gibidir. Tek fark kullanılan 16x16 boyutundaki matrisinin değerlerinin değişmesidir.

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Şekil 4.12 : S-Kutusu tablosunun tersinin onaltılık sayı sistemi değerleri[20]

4.2.3.2 Satır kaydırma işleminin tersi

Satır kaydırma işleminin tam olarak tersidir. Amaç satır kaydırma işleminde uygulanan adımların etkisini kaldırmaktır. Satır kaydırma işleminde sırasıyla çevrimsel bir şekilde sola kaydırılan satırlar bu sefer sağa kaydırılacaktır. Aynı şekilde ilk satırda kaydırma işlemi yapılmayacaktır. İkinci satır sağa 1 bayt yani 8 bit ötelenir, üçüncü satır sağa 2 bayt ötelenir ve son satır sağa 3 bayt ötelenir. Taşan baytlar satırın başına eklenir. [20]



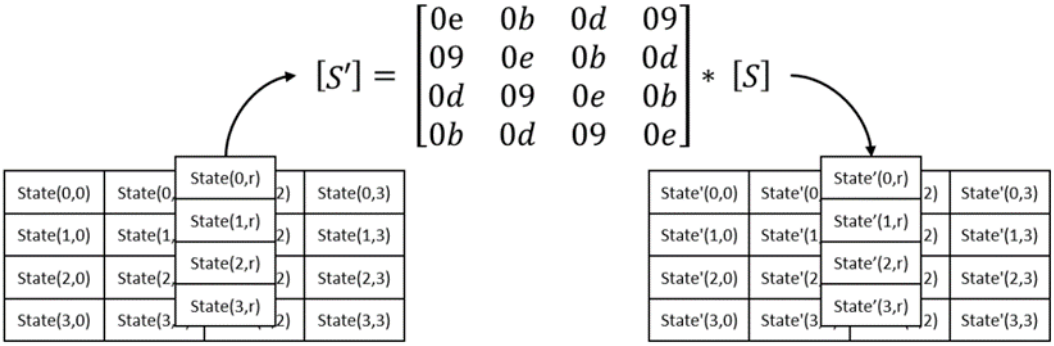
Şekil 4.13 : Satır kaydırma işleminin tersi [20]

4.2.3.3 Sütun karıştırma işleminin tersi

Sütün karışma işleminin sonuçlarını geri almak için kullanılır. Sütün karıştırma işleminde A matrisi ile durum matrisi çarpılarak yeni sütünlar elde edilmektedir. Sütün karıştırma işleminin tersi olan bu işlem için, sütün katıştırma işlemindeki adımlar burada da aynen uygulanmaktadır. Tek fark bu işlemi yaparken A matrisinin tersinin

kullanılmasıdır. A matrisinin tersi ile durum matrisi çarpılır. Matris çarpmasında Glois Field çarpma ve xor işlemi kullanılır. A matrisinin tersi aşağıda görünmektedir.[20]

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \quad (4.3)$$

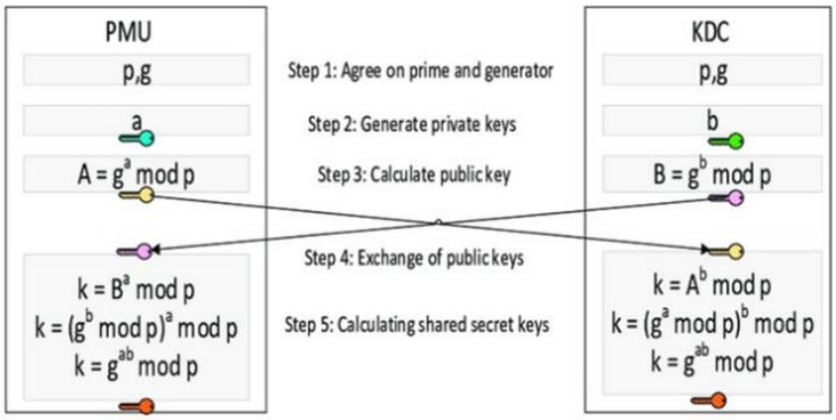


Şekil 4.14 : Sütun karıştırma işleminin tersi[69]

4.3 Diffie Hellman Anahtar Değişimi Algoritması

Diffie Hellman anahtar değişim protokolü[19], alıcı ve gönderici arasındaki ortak anahtar kullanımı sorununu çözmektedir. Ayrık logaritma problemine dayanmaktadır[58]. Algoritma üs alma işlemi ve mod alma işlemi içermektedir. Büyük sayılarla işlemlerin hızlı ve doğru yapılabilmesi amacıyla için Kare Alma ve Çarpma algoritması (Square and Multiply Algorithm) kullanılmaktadır[70]. Bu işleminin zaman açısından daha verimli bir şekilde yapılabilmesi için farklı algoritmalar tercih edilebilmektedir.

Bu proje tasarlanırken AES şifrelemede kullanılan ana anahtarın belli sürelerde değişmesi planlanmıştır. Belli bir kullanım süresi sonrası ortak anahtarın yenilenmesi için Diffie Hellman anahtar değişimi protokolü kullanılmıştır. Şekil 4.14 ile gösterilen "k" değeri, ortak olarak üretilen anahtar değeridir. Mod değeri olan "p" ve "g" alpha değeri herkes tarafından bilinmektedir. "a" değeri PMU kullanıcısı için, "b" değeri KDC kullanıcısı için gizli anahtardır ve kimseyle paylaşmamalıdır. PMU "A" değerini KDC'ye gönderir, "B" değerinin KDC'den alır. PMU, "$k = B^a modp$" işlemi yaparak gizli anahtara ulaşır. Aynı sayısal işlemler farklı değerler ile KDC kullanıcısı da k değerine ulaşır. KDC kullanıcısı "$k = A^b modp$" işlemini yapar.



Şekil 4.15 : Diffie-Hellman anahtar değişimi algoritması[71]

Diffie Hellman anahtar değişim algoritmasında modüler üs alma, tekrarlanan modüler çarpmalar gerektirir. İşleminin zaman açısından daha verimli bir şekilde yapılabilmesi için farklı algoritmalar kullanılmaktadır. Bu projede Kare Alma ve Çarpma algoritması, Montgomery modüler çarma işlemini [72] kullanmaktadır.

1985 Montgomery, modüler çarpma için yeni bir yöntem tanıttı [72]. Montgomery indirgemesi, klasik modüler indirgeme adımını açıkça gerçekleştirmeden modüler çarpmanın verimli bir şekilde uygulanmasına izin veren bir tekniktir [73]. Yönteminin çok verimli olduğu kanıtlanmıştır. [74]

Algorithm 2. Modular exponentiation

INPUT: integers $0 \leq M < N$, $0 < e < N$, $e = (e_{t-1}, e_{t-2} \cdots, e_0)_2$

$e_{t-1} = 1$ and N

OUTPUT: $M^e \bmod N$

1. $A \rightarrow M$
2. For i from $t-2$ to 0 do:
   - 2.1 $A \rightarrow AA \bmod N$
   - 2.2 If $e_i = 1$, then $A \rightarrow AM \bmod N$
3. Return (A)

Şekil 4.16 : Montgomery Modüler Çarpma Algoritması [74]

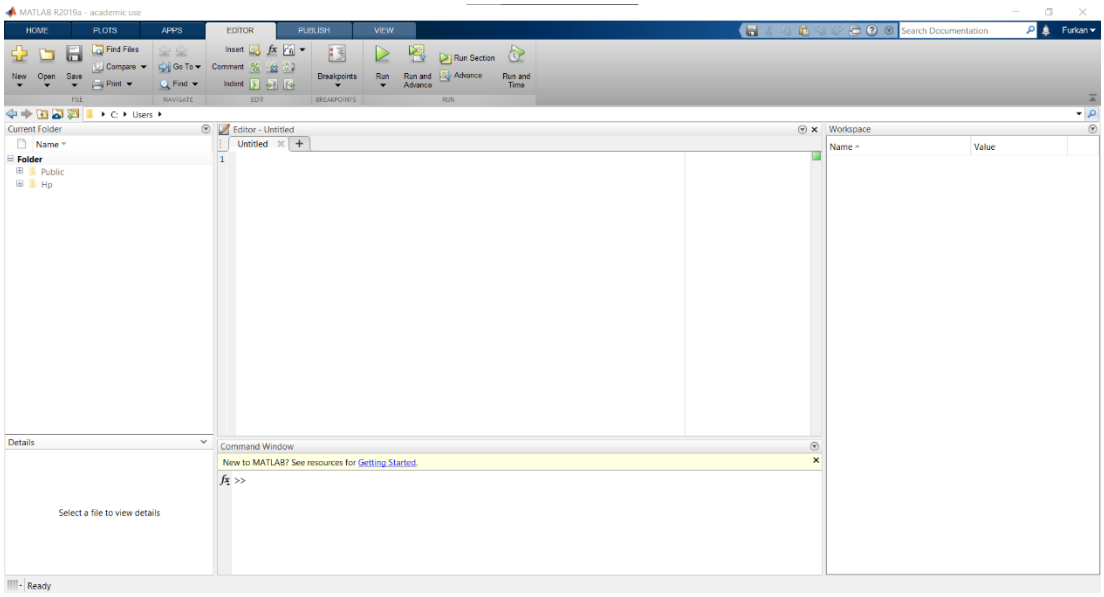
5. KULLANILAN ARAÇLAR

Projenin yapımında kullanılan yazılımlar MATLAB, Model Composer, Xilinx Vivado ve Vitis'tir. Bu yazılımlarla algoritma tasarımı, model tabanlı sayısal devre tasarımı ve IP tabanlı sayısal tasarım yapılmıştır. Aynı zamanda kullanılan Microblaze'in programlanması amacıyla Vitis programında C kodu yazılarak Microblaze'e gömülmüştür. Bu bölümde bu programlarla ilgili kısa bir bilgi ve proje tasarlamaya başlamak için gereken ilk adımlar hakkında bilgi verilecektir.

5.1 MATLAB Ortamı

Komut Penceresi(Command Window), Çalışma Alanı tarayıcısı(Workspace) ve Değişkenler düzenleyicisi(Variables Editor) dahil olmak üzere MATLAB[21] masaüstü ortamının öğelerini kullanarak verileri ortama dahil edilir, değişkenler tanımlanır ve hesaplama yapmayı sağlar. Tüm matematiksel işlemleri yapmaya uygun fonksiyonları bulunmaktadır. Günümüzde özellikle mühendislik alanlarında, endüstride ve akademide kullanılmaktadır. Önceden oluşturulmuş grafikleri kullanarak veriler görselleştirilebilir ve görselleştirmeler özelleştirilebilir. Hazır fonksiyon kütüphaneleri ve sözdizimi(syntax) ile ilgili kılavuzu ve fonksiyonların nasıl kullanılacağını gösteren kod örneklerini bulmak için belgeler de program tarafından sunulmuştur. Aynı zamanda, başkalarıyla eş zamanlı paylaşılabilen bir not defterinde kod, çıktı ve biçimlendirilmiş metni birleştiren komut dosyaları oluşturmak için Canlı Düzenleyici (Live Editor) kullanım imkanı da vermektedir.

Proje kapsamında Menzil Doppler algoritmasının gerçeklenmesi ve gerçeklenen algoritmanın görselleştirilmesi için kullanılmıştır. SAR uygulaması olması sebebiyle büyük bir veri yükü oluşmuştur. Bu sebeple bu tür yüksek kaynak kullanımı olan projelerde yüksek RAM ve işlemci gücü olan bilgisayarlarda işlem yapılması önerilmektedir. Projede Model Composer'la uyumlu şekilde çalışabilmesi amacıyla iki farklı MATLAB sürümü kullanılmıştır. R2019b versiyonu Model Composer[22] 2020.1 sürümü ile uyumlu olduğu için kullanılmıştır. Öncesinde algoritma tasarımı adımı yapıldığı için R2020a versiyonuyla başlanmış olduğundan bu versiyonla çalışılmıştır ancak algoritma tasarımı için her iki versiyon da kullanılabilir.

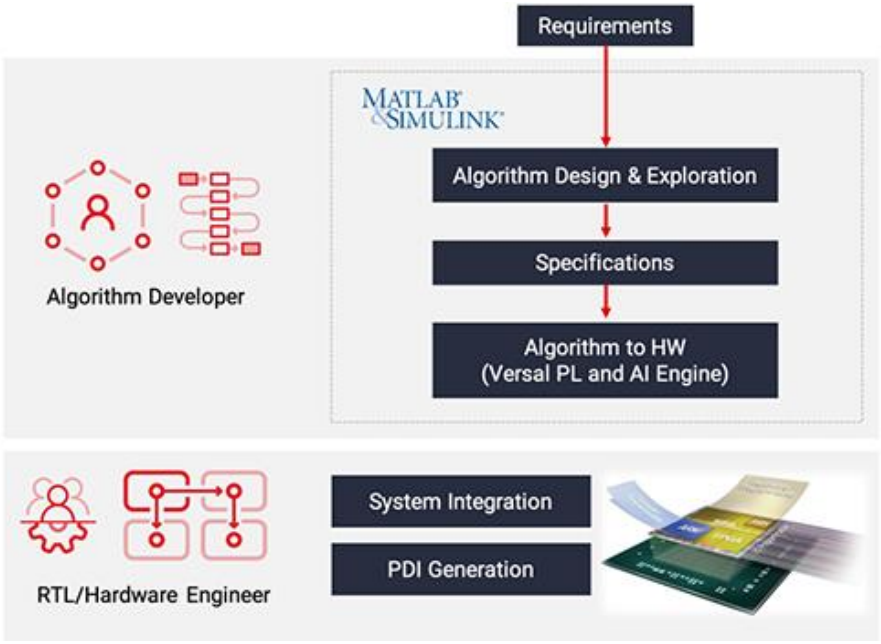


MATLAB programı arayüzü

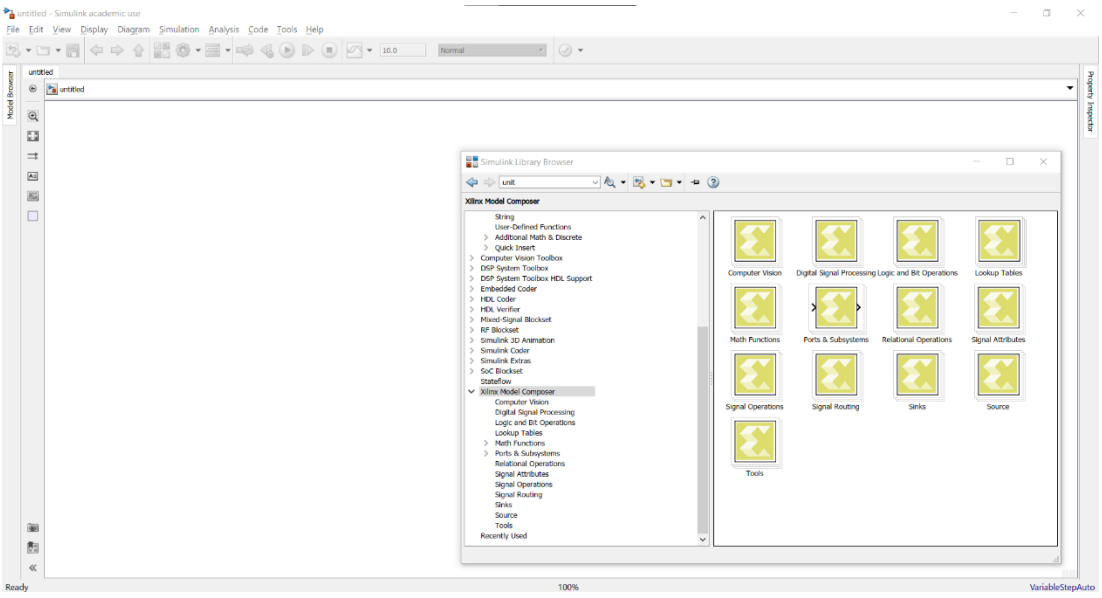
Program arayüzünde görüldüğü gibi yapılmak istenen işlemler editörde MATLAB sözdizimine(syntax) göre editörde yazılır. Kullanılan değişkenler sağ tarafta bulunan çalışma alanında listelenir. Kullanılan değişkenin türü ve büyüklüğü de burada gösterilir. Alt tarafta bulunan komut penceresinde de istenen işlemler hızlıca yapılıp yine aynı pencerede gösterilir ve sonuç çalışma alanına kaydedilir. Çalışma alanında bulunan değişkenler yine burada da kullanılabilir. Sol tarafta ise açılan projenin kayıtlı olduğu dizini gösterir. Buradan diğer oluşturulan projelere geçilebilir. Aynı zamanda başka bir dosyadan okuma yapılabilir. Bunu yaparken okunan dosya dizinde olmalıdır.

5.2 Model Composer Ortamı

Model Composer[22], Xilinx tarafından MathWorks Simulink ortamına uyarlanmış, model tabanlı tasarıma ve tasarımın simülasyonuna imkan sağlayan ayrıca tasarımı Xilinx FPGA'leri üzerinde gerçeklenebilir hale getiren bir araçtır. Model Composer'da işaret işleme, lojik işlemler ve bit manipülasyonları gibi işlemleri gerçekleştiren hazır bloklar bulunmaktadır ve aynı zamanda Simulink ortamında bu bloklarla oluşturulan model tabanlı tasarımların simülasyonu yapılabilmektedir. Ayrıca Model Composer kütüphanesinde hazır sunulan bloklar dışında kullanıcı tarafından C/C++ programlama dilleri ile tanımlanmış fonksiyonlar blok olarak Model Composer kütüphanesine eklenebilmektedir. Bu bloklar aynı zamanda kodları dinamik olarak yazıldığında Model üzerinde değiştirilebilme imkanı vermektedir.



Model Composer tasarım adımları ve sunduğu imkanlar[22]



Model Composer arayüzü ve sunulan hazır kütüphaneler

Model Composer arayüzü MATLAB simulink ile aynı arayüzü kullanmaktadır. Blok kütüphanesine Xilinx Model Composer adında ekstra bir kütüphane eklenir. Simulink'in diğer kütüphaneleri HDL olarak üretilmediği için bu elemanlar sistem tasarımında kullanılamamaktadır. Kullanıldığında simülasyon yapılabilir ancak kod veya IP üretimi yapılırken bu bloklar Model Composer tarafından uyarı verecektir.

Proje yapılırken karşılaşılan problemlerden biri de Simulink'in yüksek veri miktarı içeren uygulamalarda donmasıdır. Simülasyon esnasında program kilitleniyor ve

belirli bir yüzdede sabit kalıyor. Bu sebeple FFT denenirken uzunluğu maksimum 128 olarak seçilebilmiştir ve daha büyük sayılarda sistem tıkanmaktadır. Sistem 1024, 512 ve 256 uzunluklarında denenmiş ve başarısız olmuştur.

Sinir ağı tasarımı ve kripto bloklarının üretimi esnasında ise bir problemle karşılaşılmamış ve iki sistem de hatasız olarak üretilebilmiştir.

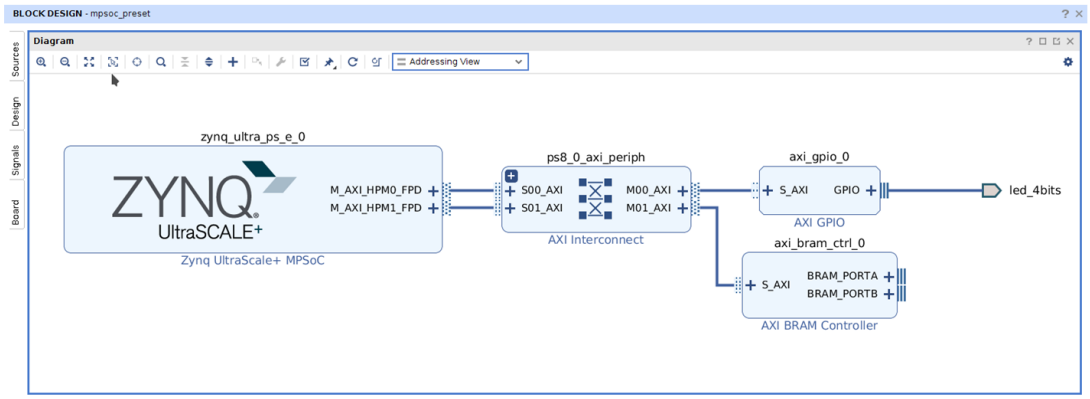
5.3 Xilinx Vivado Tasarım Ortamı

Xilinx Vivado, HDL tasarımların analiz edilip Xilinx FPGA'leri üzerinde sentezlenebilmesine olanak sağlayan bir araçtır. VHDL veya Verilog programlama dilleri kullanılarak tasarlanan sistemler Xilinx FGPA'leri üzerinde, istenilen optimasyon ayarları doğrultusunda setenlenebilmekte ve simülasyonları yapılabilmektedir.

Vivado Simulator ile yapılan tasarımlar test kodları yazılarak test ortamında simülasyonları yapılabilmektedir. Yapılan test sonuçları dalga görünümü veya konsol çıktısı olarak okunabilir.

Vivado Simulator dalga görünümü[75]

Vivado IP Integrator sayesinde hazır modeller ile dizaynlar yapılabilmekte ayrıca MathWork Simulink ortamında üretilmiş modeller de eklenerek kullanılabilmektedir. Eklenen modeller Vivado IP Integrator ortamında giriş çıkış portları birbirlerine bağlanarak haberleşmeleri yapılabilir. Vivado IP Integrator bağlanmış olan modellere oluşturulmuş dizaynların üst modül kodunu otomatik olarak üretir.



Vivado IP Integrator örneği[76]

Vivado Tcl Store, Vivadoya yeni eklentiler ekleyerek onun yapabildiklerini geliştirebilmemizi sağlar. Vivadonun temelinde olan bütün fonksiyonları kullandığı yerdir.

Proje yapılırken Model Composer'dan üretilen modeller Vivado IP Integrator ortamında hazır Microblaze ve hazır AXI modelleri ile birleştirilerek bir dizayn oluşturuldu. Bu dizayn sayesinde proje çıktılarımızı FGPA üzerinde sentezleyip, Vivado Simulator ile simüle ederek modellerimizin doğruluğunu gözlemledik.Xilinx Vivado, oluşturduğumuz modelleri FPGA üzerinde gerçeklerken ortaya çıkabilecek olası sorunları tespit edebilmemizi sağladı.

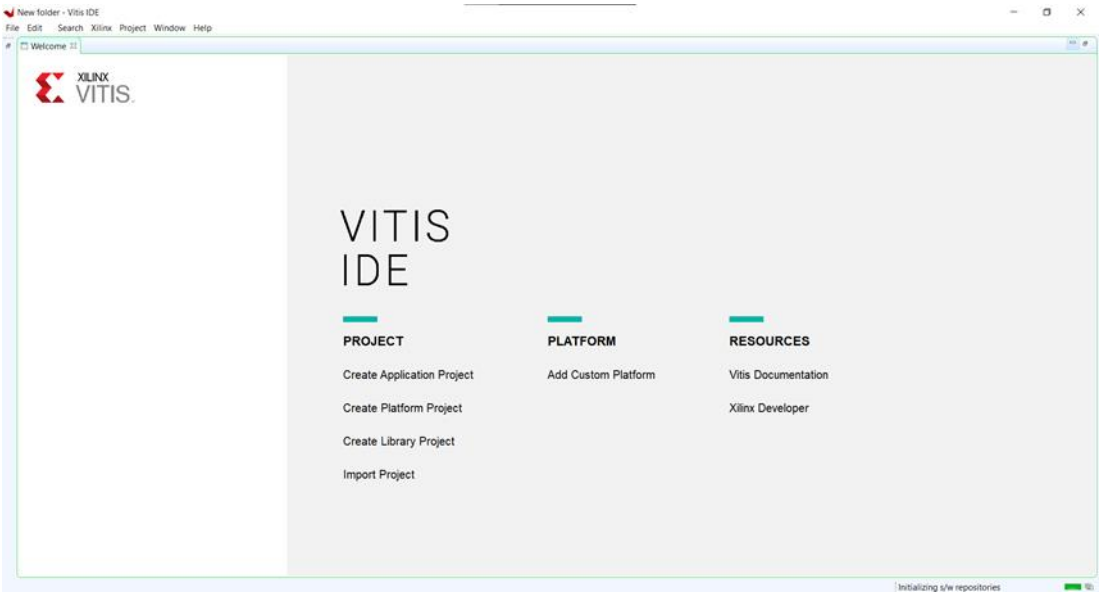
5.4 Vitis Ortamı

Vitis [26] birleşik yazılım platformu, Xilinx® yazılım geliştirmenin tüm yönlerini tek bir birleşik ortamda birleştiren yeni bir araçtır. Vitis yazılım platformu, hem yeni nesil teknolojiye geçmek isteyen Xilinx Yazılım Geliştirme Kiti (Software Development Kıt-SDK) kullanıcıları için Vitis yerleşik yazılım geliştirme akışını hem de Xilinx FPGA'daki en son teknolojileri kullanmak isteyen yazılım geliştiricileri için Vitis uygulama hızlandırma geliştirme akışını destekler.

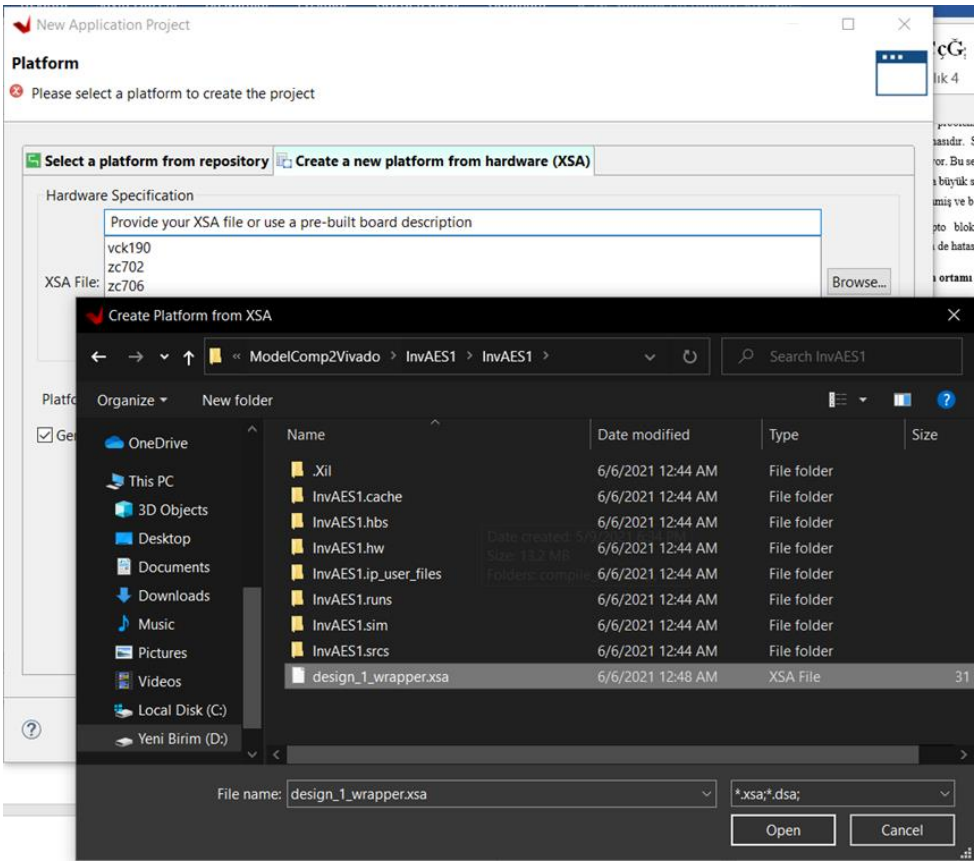
Vitis yazılım platformu, veri merkezi uygulamalarını gömülü platformlara taşımanıza olanak tanır. Vitis IDE, Xilinx gömülü işlemcilere yönelik gömülü yazılım uygulamalarının geliştirilmesi için kullanılmak üzere tasarlanmıştır. Vitis IDE, Vivado Design Suite ile oluşturulan donanım tasarımlarıyla çalışır.

Vitis ara yüzünde takip edilen adımlar sırayla anlatılmıştır. Xilinx Vivado ortamından başlayıp "elf" dosyası üretilmesilmesiyle bu aşama sonlanmaktadır. "Lunch Vitis

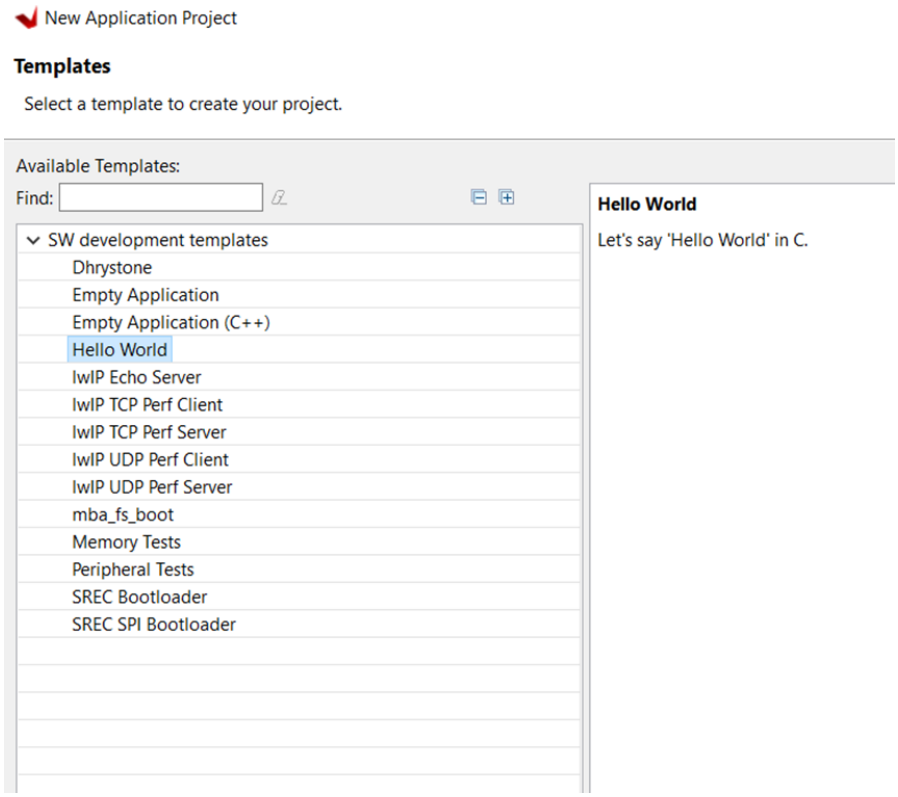
IDE" komutuyla programın çalışacağı adres yolu belirtilir. "Create Application Project" ile yeni proje oluşturulmaya başlanır. Vitis projesi, Vivado yardımıyla üretilen "xsa" dosyasıyla üretilmektedir. "Hello Word" şablonu seçilmiştir.



Vitis IDE proje oluşturma arayüzü

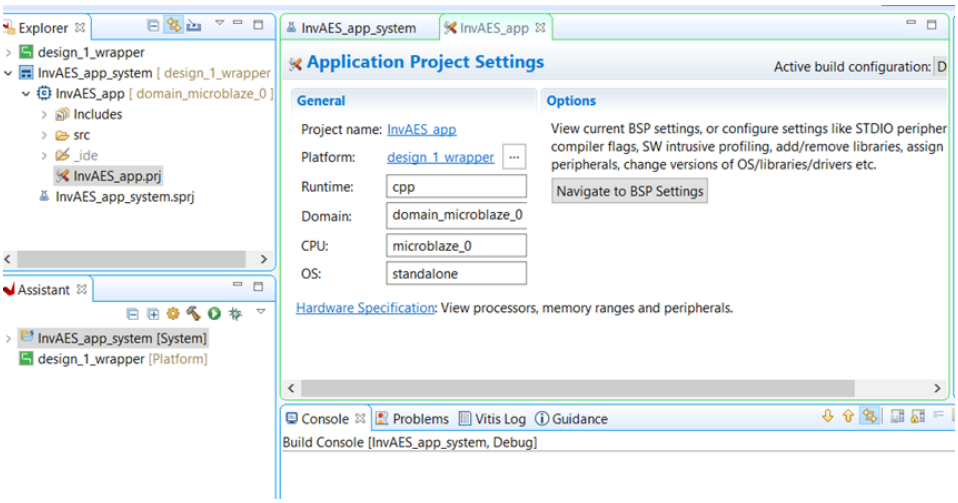


Xsa dosyası ile platform oluşturulması



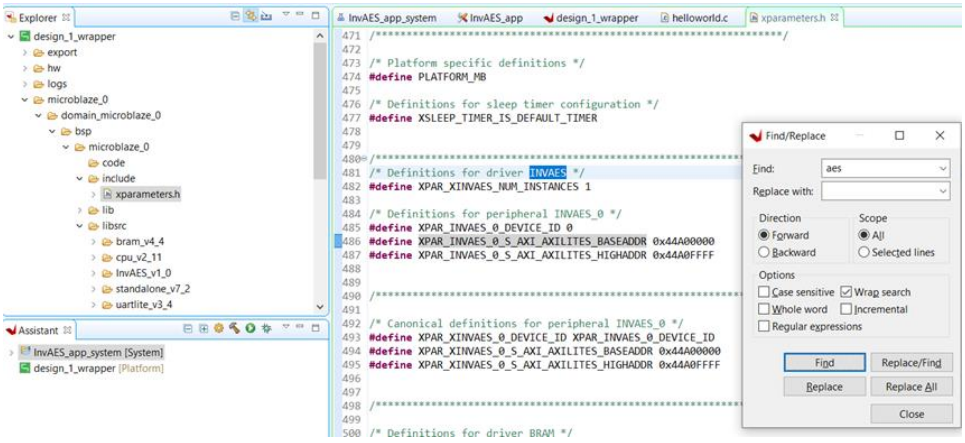
Şablon seçimi

Proje ayarlarında yer alan "Hardware Specification" bölümündei Vivado ile sentezlenen blok diyagramda yer alan IP'lerin kullandığı register adresleri yer almaktadır. Bu memory adreslerine göre ilgili girişler verilip, çıkışlar okunacaktır.



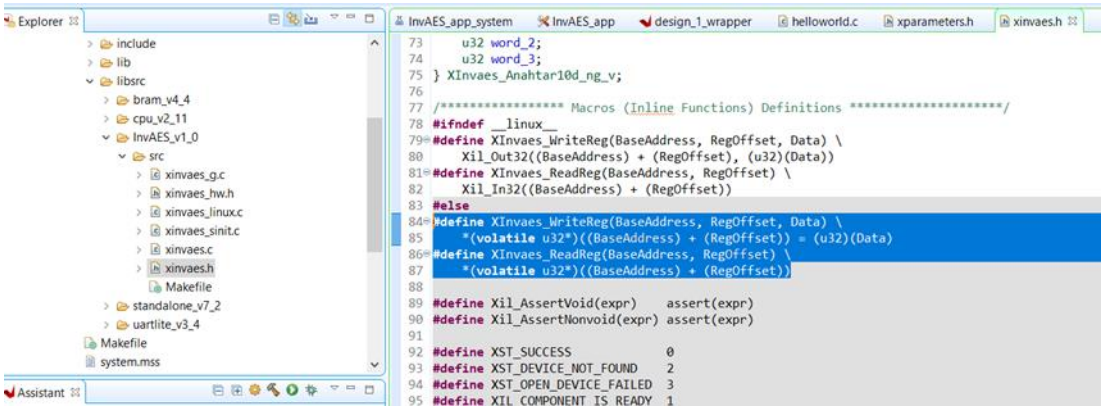
Proje ayarları menüsü

IP'lerin kullandığı adreslerin tesbit edilmesinden sonra "xparameters.h" dosyasında yer alan "XPAR_INVAES_0_S_AXI_AXILITES_BASEADDR" fonksiyonu ile belirlenen adreslere ulaşılması sağlanmaktadır.



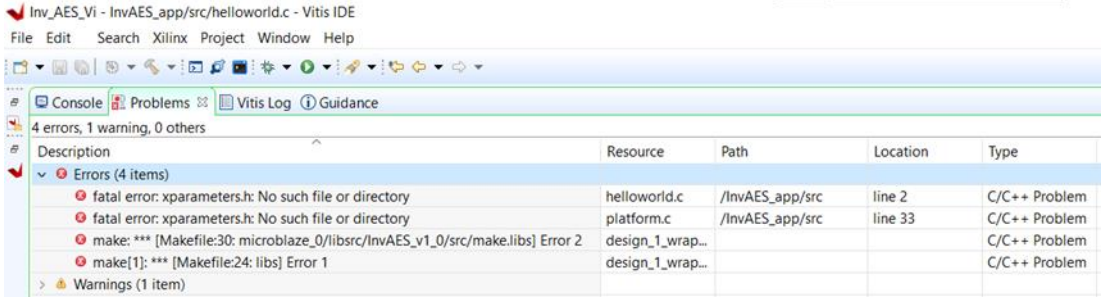
“Xparameters.h” dosyası

Ekran görüntülerinde yer alan Vitis projesi AES şifre çözme bloğu için oluşturulmuştur. Bu proje için “xinvaes.h” dosyasından, tespit edilen memory adreslerine okuma/yazma işlemlerinin belirtilmesinde kullanılacak olan fonksiyonlar alımıştır. Her IP için bu fonksiyonlar değişmektedir.



“xinaes.h” dosyası

Şekil 5.12 ile gösterilen hatalar, “Makefile” dosyasından kaynaklanmaktadır. Yapılan araştırmalar ile Şekil 5.13’de yer alan çözüm bulunmuştur. Burada verilen kodlar make file dosyasının içinden değiştirilmelidir.



Makefile hatası

```
INCLUDEFILES=*.h
LIBSOURCES=*.c
OUTS = *.o
OBJECTS = $(addsuffix .o, $(basename $(wildcard *.c)))
ASSEMBLY_OBJECTS = $(addsuffix .o, $(basename $(wildcard *.S)))

libs:
echo "Compiling myip"
$(COMPILER) $(COMPILER_FLAGS) $(EXTRA_COMPILER_FLAGS) $(INCLUDES) $(LIBSOURCES)
$(ARCHIVER) -r ${RELEASEDIR}/${LIB} ${OBJECTS} ${ASSEMBLY_OBJECTS}
make clean

include:
${CP} $(INCLUDEFILES) $(INCLUDEDIR)

clean:
rm -rf ${OBJECTS} ${ASSEMBLY_OBJECTS}
```

Makefile hatasının çözümü [77]

6. MENZİL DOPPLER ALGORİTMASININ GERÇEKLENMESİ

Menzil Doppler algoritması, projede kullanılan ESA'nın ERS-1/2 verilerini işleme amacıyla ilk olarak MATLAB'da gerçeklenecektir. Bunun için çeşitli makaleler araştırılmış ve referanslarda belirtilen makale ve kaynaklardan faydalanılmıştır. MATLAB'da tasarlanan algoritma daha sonra FPGA'de gerçeklenmesi için Model Composer programıyla model tabanlı olarak tasarlanıp bir IP paketi oluşturulacak, daha sonra bu IP paketi Vivado programına eklenerek burada devam edilecektir. Burada kullanılan modülün giriş çıkış ve AXI arayüzüne uyumlu kontrol sinyalleri İndirgenmiş Komut Takımlı Bilgisayar(Reduced Instruction Set Computer- RISC)[78] tabanlı açık kaynak kodlu Microblaze işlemcisiyle kontrol edip diğer bloklarla haberleştirilip SAR işleme sistemi son aşamaya geçirilmesi planlanmıştır ancak Model Composer tarafından üretilen IP'nin içerisinde kullanılan FFT bloğu Vivado'nun sunduğu IP'yi kullanması Microblaze tarafından kontrol edilmesini desteklememiştir. Model Composer tarafından tasarlanan sistem yine Model Composer üzerinde simülasyonu yapılıp çalışmasına rağmen IP paketlenip Vivado'ya geçilmesiyle giriş çıkış portlarına driver üretmemektedir. Aşağıdaki bölümlerde bu sorunla ilgili kapsamlı bir bilgi verilecektir.

Üretilen IP ile denenen sistem Vivado'da Microblaze ile kontrol edilmesi için gerekli sistemler ile birlikte sentezlenip daha sonra Vitis'te bu IP'yi kullanmak üzere .xsa dosyası oluşturulmuştur. Burada üretilen xsa dosyasıyla birlikte Vitis'te Microblaze'i kontrol eden C kodu yazılmıştır. Bu kod doğrulandıktan sonra Vitis tarafından bir .elf dosyası üretimiştir. Bu .elf dosyası Microblaze'e yüklenerek tasarlanan sistem kontrol edilebilmektedir ancak SAR görüntü oluşturma kapsamında üretilen IP'nin içinde FFT bloğu bu sistem için uyumsuzluk oluşturduğundan dolayı bu kısım projenin geliştirilmesi gereken yönünü ortaya çıkarmıştır. Aşağıdaki kısımlarda proje adımları program bazlı olarak açıklanacaktır. Ayrıca kaynak kodları bir github linki üzerinden verilecektir.

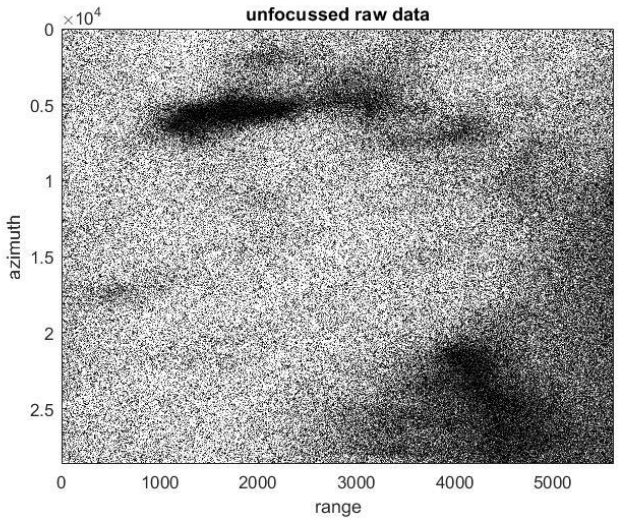
6.1 MATLAB

İlk olarak Menzil Doppler algoritması tasarlanarak çalışan bir algoritmaya sahip olmak istendi. Bunun için referanslarda belirtilen kaynaklardan faydalanarak bir algoritma oluşturuldu. Başlangıçta daha farklı algoritmalar araştırılmıştır ancak ERS veri seti ile yapılmış örnekler olması ve gerçeklemesinin diğer algoritmalara göre daha kolay ve sayısal lojiğe uygun olması nedeniyle Menzil Doppler algoritması seçilmiştir.

Projeye ilk olarak ERS verilerinin dosyadan okunmasıyla başlamaktadır. Bunun için ismi read_rawdata.m olan fonksiyon dosyası yazılmıştır ve process.m isimli temel script tarafından çağırılmaktadır. Bunların yanında sistem parametreleri birer değişkene atanarak çalışma alanına eklenmiştir. Burada dikkat edilmesi gereken nokta ERS verilerinin her birinin ortak değerli parametreleri olmasına ragmen Doppler frekansı ve sütun sayısı gibi farklı değerli olan verileri de bulunmaktadır. Bu değerler ERS sitesinden alınan dosyaların içinden okunarak kod güncellenebilir.

ERS verilerinin dosyadan okunması

ERS–2 uydusunun görüntüleme için oluşturduğu veriler internet sayfasından indirilerek .raw dosyasından okunmaktadır. Karmaşık sayılar içeren bu veri setinde MATLAB üzerinde dosya okuma fonksiyonu kullanılarak işlemeye hazır formata getirilmiştir. Format iki boyutlu bir matris gibi düşünürsek satırlar menzil doğrultusunu, sütunlar ise azimut doğrultusunu göstermektedir. Uydu 16.5 saniye çalışırken PRF(Pulse Repetition Fequency) değeri 1679.9 Hz'dir. Bu iki değerin çarpımından elde edilen sonuçla yaklaşık 28000 darbe(pulse) gönderilmektedir. Bu 28000 veri dosyadaki her bir satırı ifade etmektedir. Sütun sayısı ise 5616'dır. Elde edilen veri ise 28000 sütuna karşılık 5616 satırdan oluşan matris şeklindedir. SAR ham verisinin her satırı 11644 bayttan oluşurken ilk 412 bayt verinin alındığı zaman, konum ve benzeri parametreleri içermektedir ve görüntü oluşturma aşamasında kullanılmayacaktır. Kalan 11232 bayt ise ham verinin satırındaki karmaşık değerleri gösteren verilerdir. Bu 11232 baytın her bir baytının değerleri karmaşık sayının gerçek değerini gösterirken çift değerleri sanal kısmını göstermektedir. Bu ifadeden yola çıkılarak 11232 sayısının yarısı olan 5616 elde edilir. Bu da her bir satır verisinin 1 baytla ifade edildiğini göstererek ifadeyi doğrulamıştır. Ardışık gelen bu gerçek ve sanal ifadeler toplanarak karmaşık sayı elde edilir. İşlenmeye hazır hale gelen bu veriler menzil işleme kısmında işlenmeye başlayacaktır.



Şekil 6.1 : Dosyadan okunan işlenmemiş veri görüntüsü

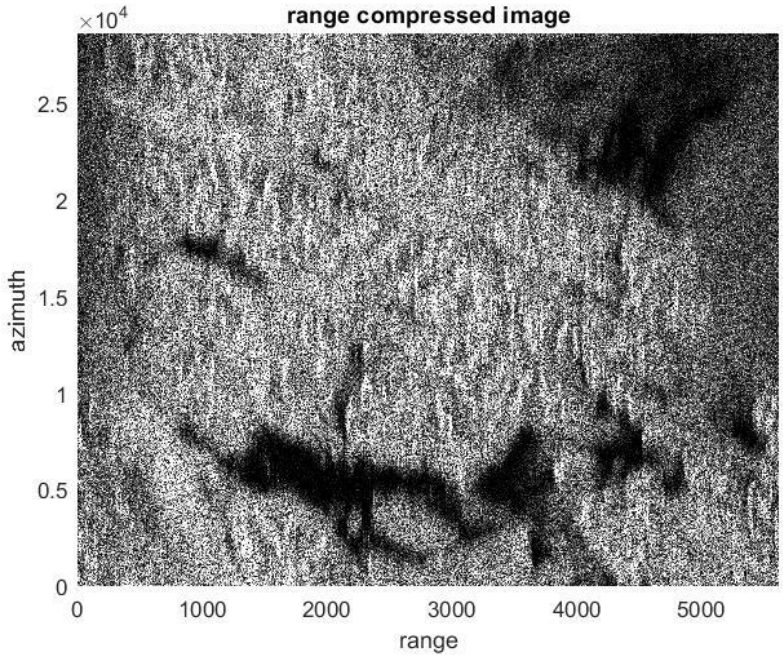
Editor - C:\Users\Hp\Desktop\Bitirme\MATLAB_files\read_rawdata.m

process.m | read_rawdata.m | range_ref.m | azi_ref.m | range_az_ref_code.m

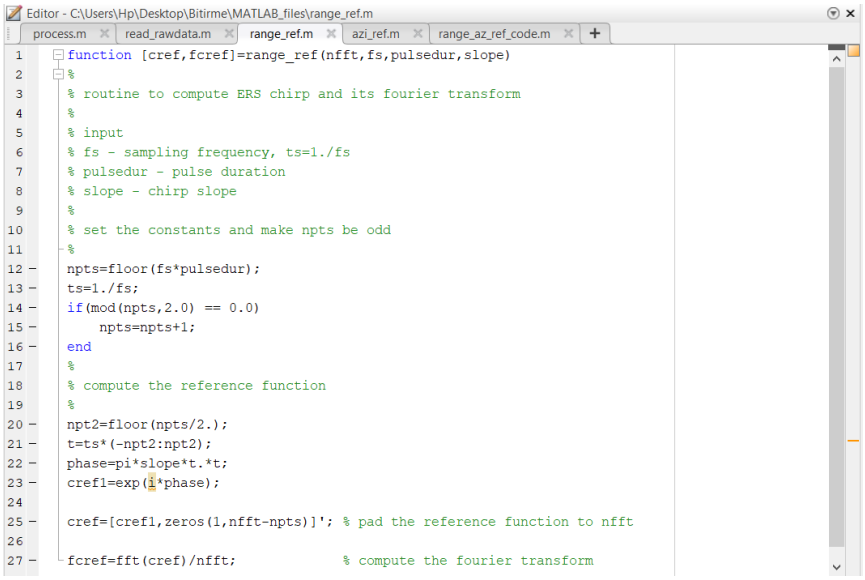
```
1  function [csar,nrow,ncol]=read_rawdata(sar_file)
2
3  fid=fopen(sar_file,'r');       % read and unpack ERS SAR data in DPAF format
4
5  [sar,nsar]=fread(fid,'uchar'); % read the bytes
6  sar=reshape(sar,11644,nsar/11644);
7  nrow=nsar/11644;               % shape of row
8  ncol=5616;                     % shape of column
9  st=fclose(fid);
10
11 % extracting real and imaginary part and subtract mean value
12 csar=complex(sar(413:2:11643,:)-15.5,sar(414:2:11644,:)-15.5);
13
```

Şekil 6.2 : Read_rawdata.m dosyadan okuma fonksiyonu

Daha sonra range işleme kısmı gelmektedir. İlk olarak dosyadan okunan verinin FFT'si MATLAB fft() fonksiyonu alınmaktadır. Burada fonksiyonun yalnızca sütun bazlı fft işlemi yaptığı göz önünde bulundurulmalıdır. Ardından sistem parametreleri kullanarak range_ref.m fonksiyonu ile menzil referans fonksiyonu oluşturulur ve FFT işleminin ardından veri ile tüm veri sütun bazlı olarak çarpılır. Daha sonra IFFT işlemine tabi tutulur ve azimut işleme adımına geçilir.

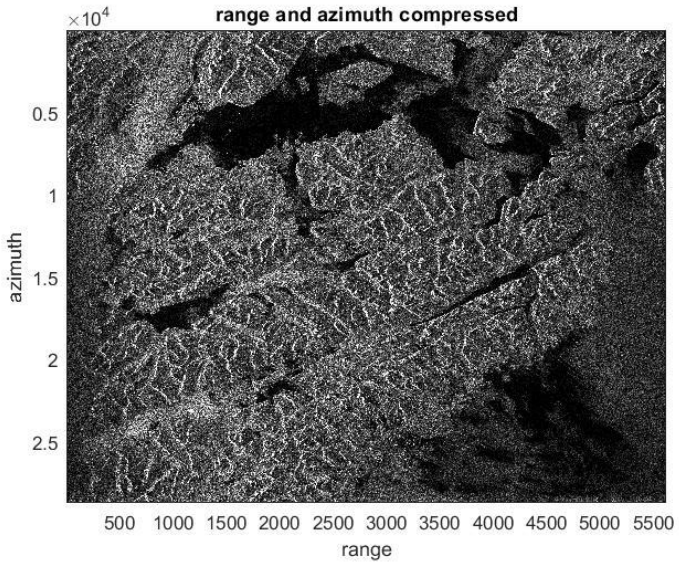


Şekil 6.3 : Menzil işlemi sonucunda oluşan SAR görüntüsü



Şekil 6.4 : Range_ref.m menzil referans fonksiyonu oluşturma

Bu adımların ardından azimut işleme adımına geçilmiştir. Menzil işleme adımında yapılan işlemler aynı şekilde azimut işleme adımında yapılacaktır. Burada yalnızca azimut referans fonksiyonu oluşturulup kullanılacaktır. Yalnızca MATLAB fft() komutunun yalnızca sütun bazlı FFT işlemi yapması nedeniyle menzil işleminden geçmiş verinin transpozu alınarak azimut işlemeye geçilir ve aynı işlemler tekrar edilir.



Şekil 6.5 : Azimut işlemi oluşan SAR görüntüsü

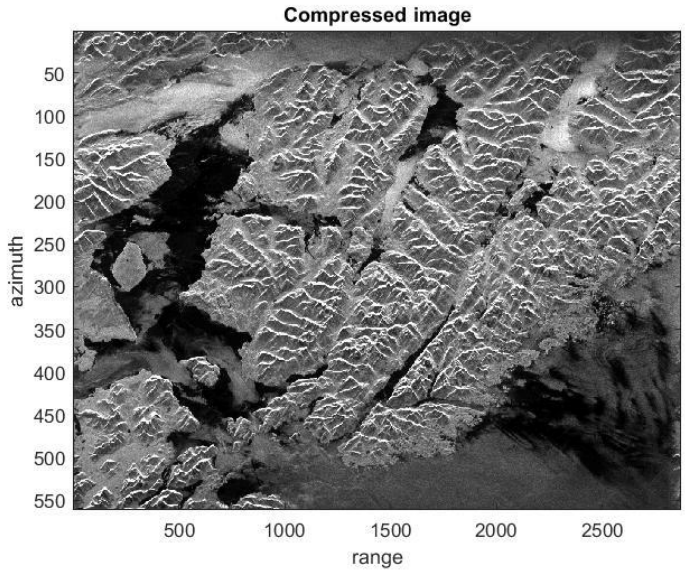
Editor - C:\Users\Hp\Desktop\Bitirme\MATLAB_files\azi_ref.m

process.m | read_rawdata.m | range_ref.m | azi_ref.m | range_az_ref_code.m

```
function [cazi,fcazi]=azi_ref(nazi,PRF,fdc,fr)
% routine to compute ERS azimuthal chirp and itsfourier transform
%
% input
% nazi - number of points in azimuth
% PRF - pulse repitition frequency, ts=1./fs
% fdc - doppler centriod frequency
% fr - doppler frequency rate
%
% set the constants and make npts be odd
%
npts=min(nazi-1,1296);
ts=1./PRF;
if(mod(npts,2.0) == 0.0)
npts=npts+1;
end
%
% compute the azimuth chirp function
%
npt2=floor(npts/2.);
t=ts*(-npt2:npt2);
phase=-2.*pi*fdc*t+pi*fr*t.*t;
cazi1=exp(i*phase);

cazi=[cazi1(npt2:npts),zeros(1,nazi-npts),cazi1(1:npt2-1)]'; % pad the function to nfft

fcazi=fft(cazi)/nazi;                                        % compute the fourier transform
```

Şekil 6.6 : Azi_ref.m azimut referans fonksiyonu oluşturma

Bu görüntülerin çizdirilmesi için de belirli işlemlerden geçmesi gerekmektedir. Çizdirme işlemi için ilk olarak verinin aralığı bulunur. Bu aralığın minimum ve maksimum değerleri arasında gri seviyede lineer şekilde değişen bir çizdirme işlemi uygulanır. Veriler yapay sinir ağına girmeden önce küçültülmesi ve daha net görüntüler oluşturmak istememiz nedeniyle oluşturlan görüntü istenen seviyede sıkıştırılıp yeniden çizdirilmiş ve bir sonraki adıma hazırlanmıştır.



Şekil 6.7 : Sıkıştırma sonucu oluşan SAR görüntüsü

Oluşturulan görüntülerde verilerin aralığına bakıldığında farklı verilerin birbirine yaklaşık eşit olduğu gözlenmiştir. Bu da tüm ERS verileri için ortak bir çizdirme algoritması oluşturulabileceğine işaret etmektedir.

Çizelge 5.1 : İşlenmiş ERS verilerinin minimum ve maksimum değerleri

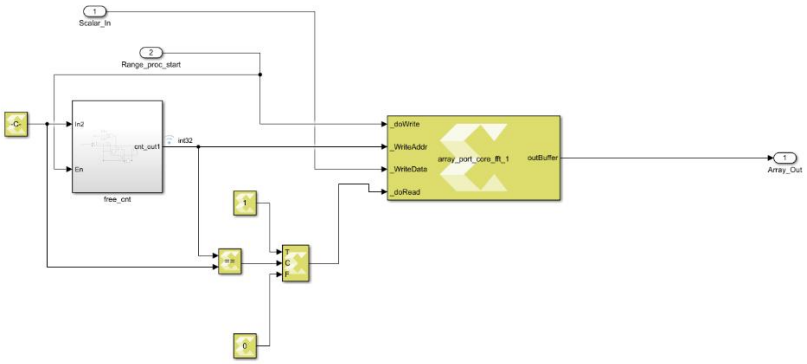
Dosya İsmi	Min Değer	Maks Değer
E1_32449_STD_L0_F477.000.raw	1.0198e-05	5.1240e-04
E1_32435_STD_L0_F434.000.raw	1.0272e-05	1.6851e-04
E1_32420_STD_L0_F491.000.raw	1.0513e-05	3.2593e-04
E1_32420_STD_L0_F451.000.raw	1.4717e-05	1.6653e-04
E1_27746_STD_L0_F249.000.raw	1.4018e-05	3.2358e-04
E1_25509_STD_L0_F329.000.raw	8.9588e-06	2.0336e-04
E1_21974_STD_L0_F305.000.raw	8.7243e-06	2.2579e-04

Bu işlemlerle MATLAB'da algoritma gerçekleme aşaması tamamlanmış oldu. Kullanılan temel tüm kodlar ve diğer yardımcı kodlar github hesabından paylaşılacaktır.

6.2 Model Composer

MATLAB'da doğrulanan algoritmayı Model Composer'da tasarlama aşamasına ilk olarak mezil işleme bloğunu tasarlamakla başlanılmıştır. Azimut işleme Menzil işleme ile benzer olması sebebiyle ayrı ayrı tasarlanmasına karar verilmiştir. Bunun bir sebebi de menzil işleme adımının ardından oluşan matris şeklindeki verinin satır bazlı işlem yapılmasına ihtiyaç duymasıdır. Bu sebeple iki ayrı blok kullanılmasına karar verilmiştir. Tek bir Microblaze ile birden fazla IP kontrol edilebilir olması ile tüm sistem birleştirilebilirdir.

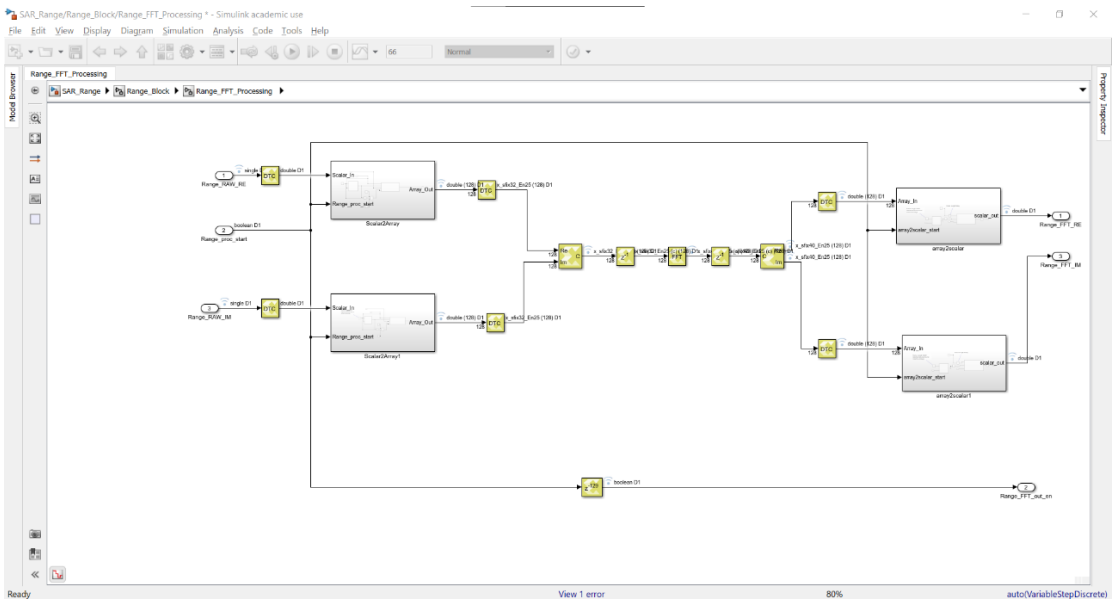
Menzil işleme adımı ilk olarak verinin FFT'sinin alınması bloğu ile başlamaktadır. Bu blokta kullanılan FFT IP'si tasarımın yapılmasının ardından Vivado ile uyumsuzluk çıkarttığından bhsedilmişti. Bunun sebebi Model Composer giriş olarak bir vektör istemesidir ancak Vivado'da buna karşılık düşen blok ise Xilinx'in sunduğu FFT bloğudur. Bu blok ise giriş olarak skaler olarak veri beklemektedir. FFT uzunluğu ise IP üzerinden seçilerek konfigüre edilebilmektedir. Giriş olarak vektör istemesinin ardından Model Composer'a vektör dönüşümü yapan bir buffer fonksiyonu yazılmıştır. Aynı şekilde çıkışında vektör verdiği için çıkışına da skaler dönüşümü yapan bir fonkiyon eklenmiştir.



Şekil 6.8 : Skalerden vektöre dönüşüm bloğu

Bu blok ile FFT işlemine veriler hazır edilmiştir. Daha sonra diğer bloklar yerleştirilerek sistem tamamlanmıştır. Sistemin girişleri reel ve imajiner veri kanalları ve dışardan alınan enable sinyalidir. FFT işlemi fixed point verilerle işlem yaptığı için DTC(Data Type Conversion) blokları kullanılmıştır. Aynı zamanda veriyi reel ve

imajiner kısımları birleştirilerek FFT'ye verilmesi için de kütüphaneden Real-Imag to Complex bloğu kullanılmıştır.



Şekil 6.9 : Menzil işleme bloğu FFT işleme bloğu

Burada kullanılan scalar2array ve array2scalar fonksiyonları da aşağıda verilmiştir. Fonksiyonlardaki 8 sayısı kullanılan bufferın uzunluğunu belirtmektedir. Bu sayı istenen FFT uzunluğu kadar uzatılabilmektedir. Ayrıca bu fonksiyyonların kütüphaneye eklenmesi için Model Composer komut penceresine girilmesi gereken fonksiyon da şekil 6.10'da verilmiştir. Bu fonksiyonun sonuna 'override' eklenerek fonksiyon güncellenebilir.

Editor - C:\Users\Hp\Desktop\Bitirme\Model_Composer\scalar2array.h

process.m | read_rawdata.m | range_ref.m | azi_ref.m | range_az_ref_code.m | scalar2array.h

```
1  #pragma XMC SUPPORTS_STREAMING
2  void scalar2array_1(bool _doWrite, int _WriteAddr, double _WriteData, bool _doRead, double outBuffer[8]);
3  #pragma XMC SUPPORTS_STREAMING
4  void scalar2array_2(bool _doWrite, int _WriteAddr, double _WriteData, bool _doRead, double outBuffer[8]);
5  #pragma XMC SUPPORTS_STREAMING
6  void scalar2array_3(bool _doWrite, int _WriteAddr, double _WriteData, bool _doRead, double outBuffer[8]);
7  #pragma XMC SUPPORTS_STREAMING
8  void scalar2array_4(bool _doWrite, int _WriteAddr, double _WriteData, bool _doRead, double outBuffer[8]);
9
```

Şekil 6.10 : Skalerden vektöre dönüşüm fonksiyonları header dosyası

Editor - C:\Users\Hp\Desktop\Bitirme\Model_Composer\scalar2array.cpp

process.m | read_rawdata.m | range_ref.m | azi_ref.m | range_az_ref_code.m | array2scalar.h | scalar2array.cpp

```
#pragma XMC SUPPORTS_STREAMING
void scalar2array_1(bool _doWrite, int _WriteAddr, double _WriteData, bool _doRead, double outBuffer[8])
{
   static const int BufferSize = 8;
   static double buffer[BufferSize];

   #pragma HLS dataflow

   if (_doWrite) {
        buffer[_WriteAddr % BufferSize] = _WriteData;
   }

   if (_doRead) {

       for (int j = 0; j<BufferSize; j++) {

           outBuffer[j] = buffer[j];
       }
   }
}

```

Şekil 6.11 : Skalerden vektöre dönüşüm fonksiyonu cpp dosyası

Editor - C:\Users\Hp\Desktop\Bitirme\Model_Composer\array2scalar.h

process.m | read_rawdata.m | range_ref.m | azi_ref.m | range_az_ref_code.m | array2scalar.h

```
#pragma XMC INPORT in
#pragma XMC OUTPORT out
#pragma XMC SUPPORTS_STREAMING
void array2scalar(bool _doRead, int _ReadAddr, double in[8], double &out) {
    #pragma HLS dataflow
    if (_doRead) {
        #pragma HLS dataflow
        out = in[_ReadAddr];
    }
}
```

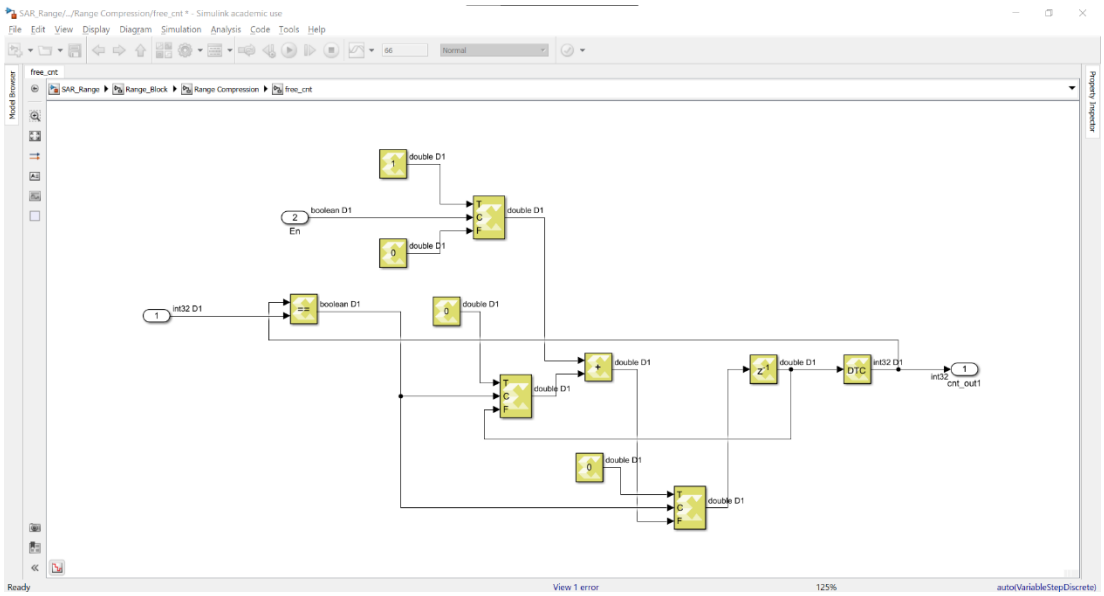
Şekil 6.12 : Vektörde skalere dönüşüm fonksiyonu header dosyası

Şekil 6.12'de verilen fonksiyonda yalnızca header dosyası bulunmaktadır. Bu şekilde de Model Composer'a fonksiyonlar eklenebilmektedir.

```
>> xmcImportFunction('SAR Lib',{'array2scalar fft 1'},'array2scalar fft 1.h',{},{},)
```

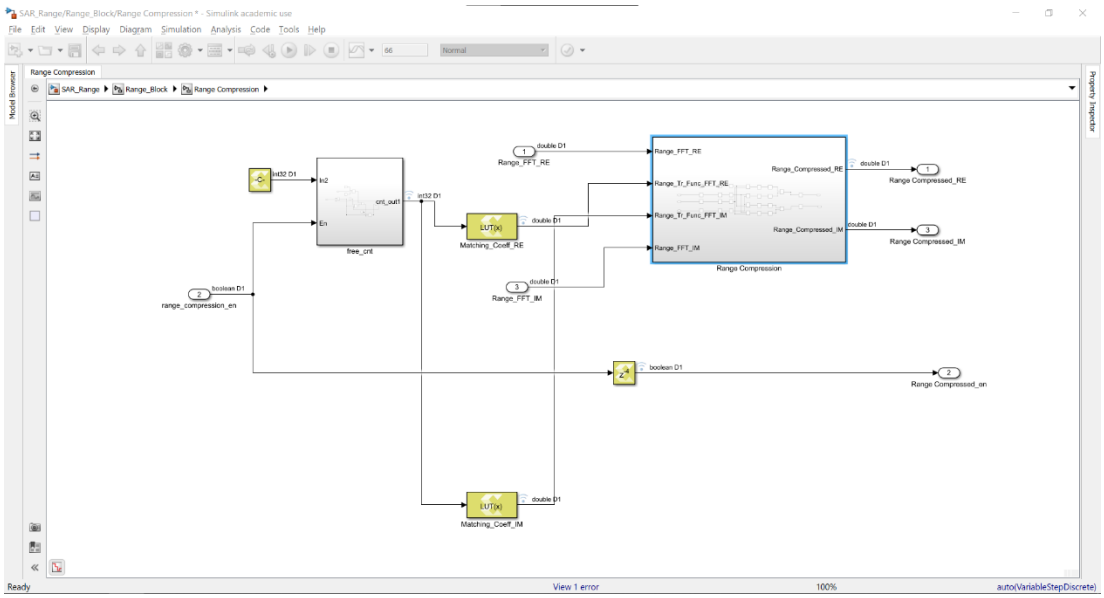
Şekil 6.13 : Model Composer'a fonksiyon dahil etme komutu örneği

Şekil 6.8'de verilen free_cnt bloğu ise bir sayıcıdır. Şekil 6.14'de bu bloğun içeriği verilmiştir. Sisteme girişte verilen değişkenle sistem o değişkene kadar sayıp dışarıya lojik 1 sinyali vermektedir.

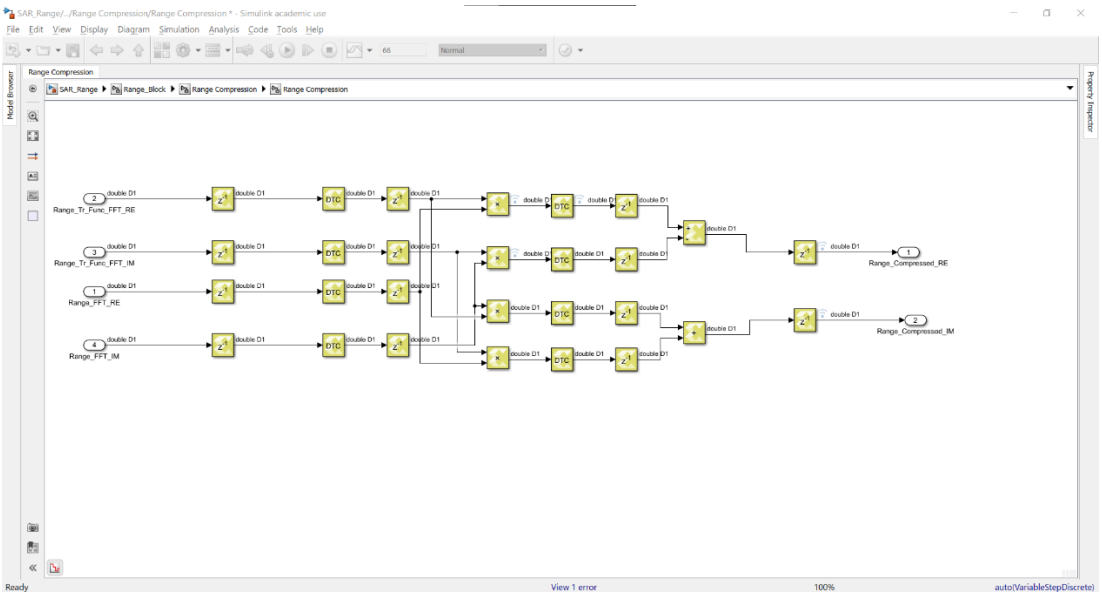


Şekil 6.14 : Model Composer sayıcı devresi

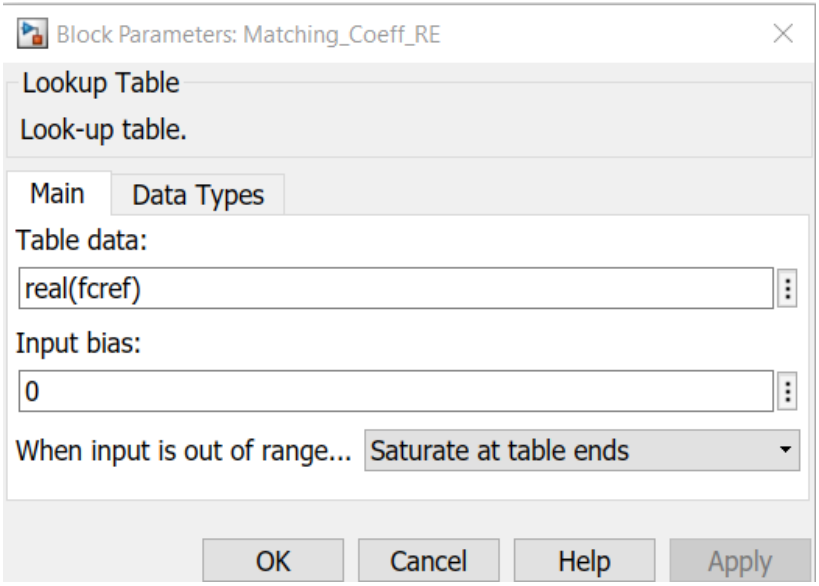
FFT'si alınmış verinin daha sonra referans fonksiyonu ile çarpılması işlemi gelmektedir. Referans fonksiyonu bir LUT(Look Up Table) ile her saat periyodunda bir elemanı çarpma işlemine vermektedir. Aynı zamanda referans fonksiyonu Model Composer çalışma alanından(Workspace) değişken olarak çekilmektedir. Şekil 6.15'te tasarım verilmiştir.



Şekil 6.15 : Referans fonksiyonları blok tasarımı

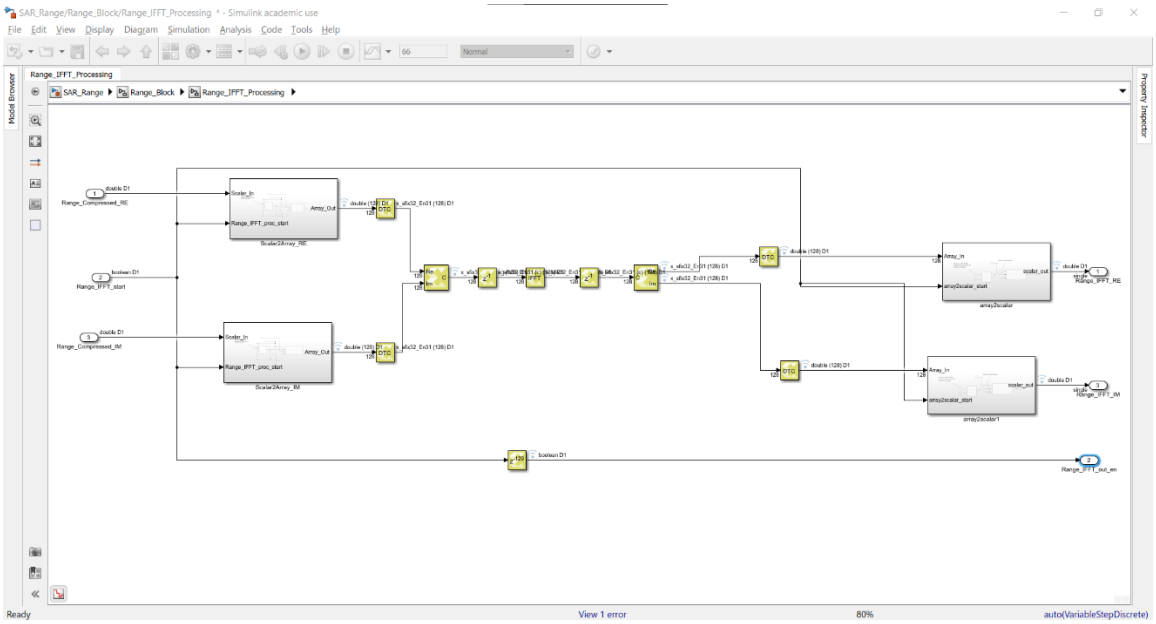


Şekil 6.16 : Kompleks çarpma işlemi bloğu



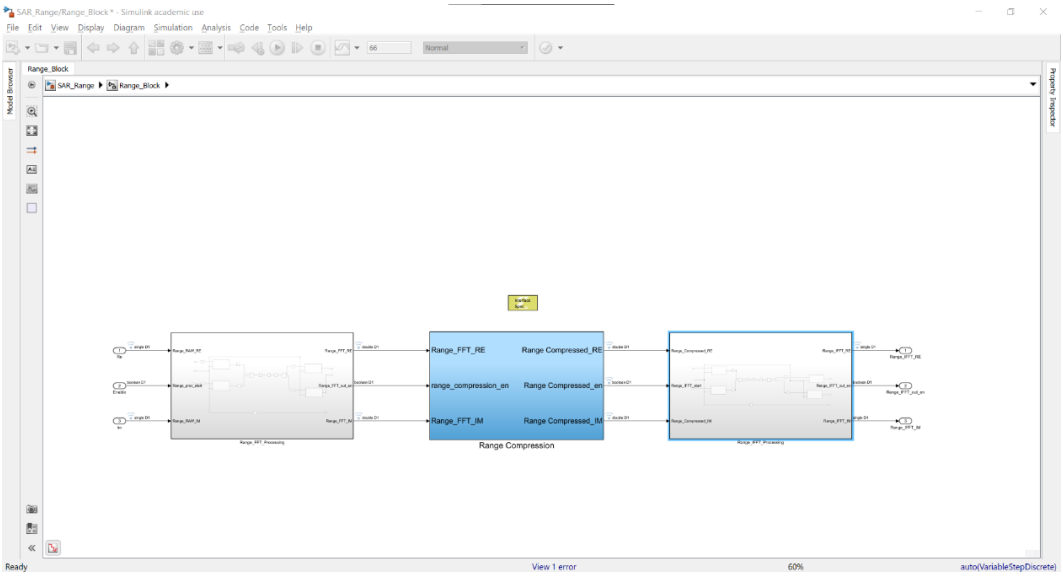
Şekil 6.17 : LUT içine yazılacak verinin workspaceden alınması

Kompleks çarpma işleminden sonra gelen IFFT bloğu şekil 6.18'de verilmiştir. Bu blok FFT bloğuyla aynıdır. Vivado'da aynı IP'ye denk düşmektedir.



Şekil 6.18 : Menzil işleme IFFT işlem bloğu

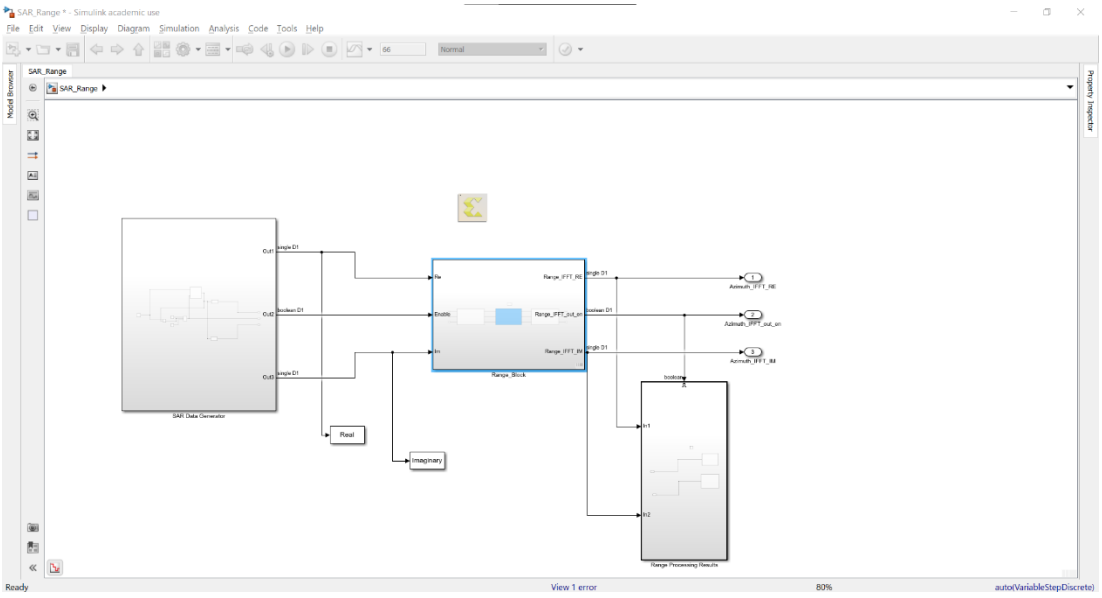
Tüm sistemin birleştirilmiş modülü ise şekil 6.19'da verilmiştir. Bu modül algoritmanın ana akışını vermektedir. Burada bir yere bağlanmayan Interface Spec bloğu üretilecek IP'nin diğer bloklarla arasındaki haberleşme arayüzünün nasıl olacağını belirler. Handshake, AXI-Stream, AXI-Lite Slave, FIFO ve AXI-Stream(Video) gibi olanak sağlamaktadır. Bu proje kapsamında AXI-Lite Slave protokolü kullanılmıştır.



Şekil 6.19 : Sistemin üst modülü

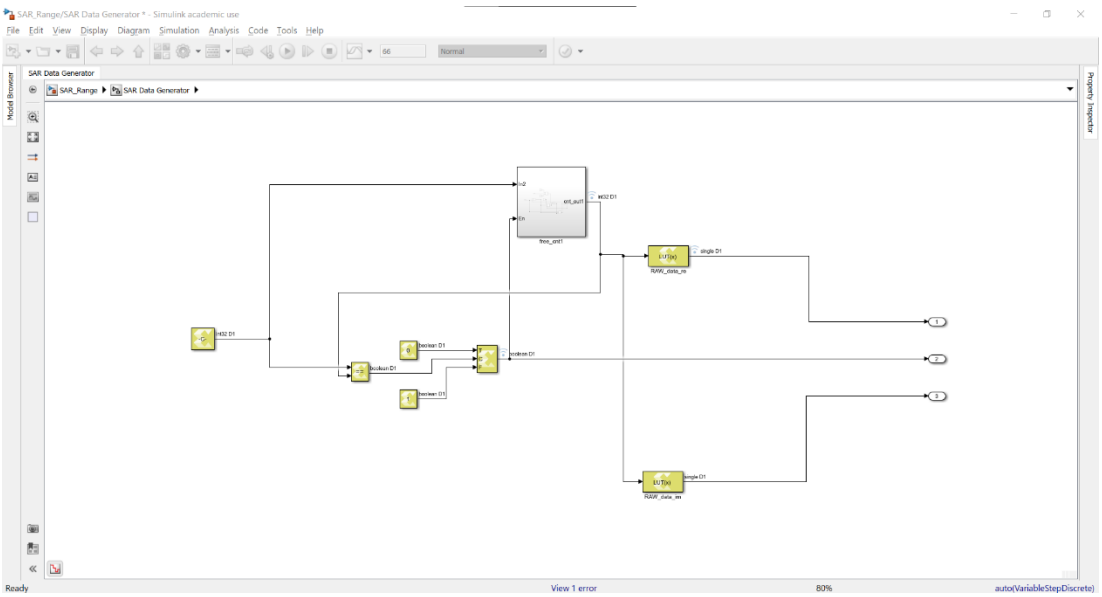
Tüm tasarım aşamalarının ardından elde edilen yapı şekil 6.20'de elde edilmiştir. Azimut işleme bloğu da menzil işleme bloğu ile aynı işlemleri yapmaktadır. Blok

yapısında farklı olan tek nokta menzil referans fonksiyonu yerine azimut referans fonksiyonu kullanılmasıdır. LUT'un içine azimut referans fonksiyonu değerleri işlenerek tüm sistem elde edilebilir.



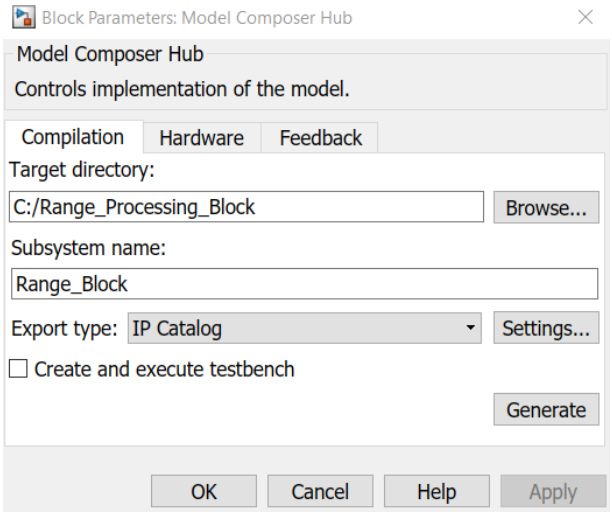
Şekil 6.20 : Sistem simülasyonu ve IP üretimi için en üst modül

Bu işlemin ardından sistem simülasyonu ve IP üretimi için uygulanan adımlar verilecektir. İlk olarak simülasyon ve IP üretimi için sisteme girişler verilmesi zorunludur. Bunun için ilk olarak bir veri üreteci oluşturulmuştur. Yapı şekil 6.21'de gösterilmiştir.



Şekil 6.21 : Menzil işleme veri üreteci yapısı

Sistem simülasyonu yapıldığında menzil işlemi sonucu ortalama piksel hatası %0.205 olarak bulunmaktadır. Azimut işleme sonucu bu hata oranı %0.2971 olarak çıkmıştır. Buradan hareketle Model Composer'da tasarlanan sistem Vivado'da kullanılmak üzere IP paketi olmaya hazır hale gelmiştir. Bunun için ilk olarak Model Composer Hub elemanı eklenir. Bu eleman ile saat frekansı, kullanılan FPGA ve hedef dizin seçilir ve IP üretimi başlatılır. Bu kısımdan sonra yapılan işlemler otomatik olarak sistem arka planında HLS kullanılarak yapılmaktadır ve seçilen dizinde IP ve proje dosyaları oluşturulur ve Vivado programında kullanıma hazır hale gelir.



Şekil 6.22 : Model Composer Hub blok parametreleri

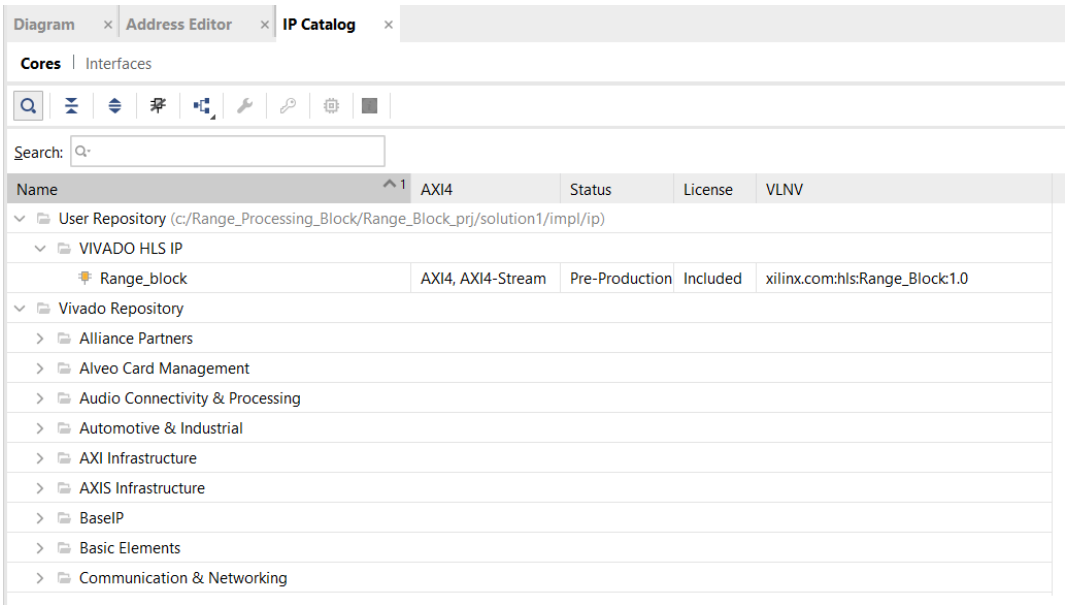
Sonuç olarak bu aşamada üretilen sistemin çalıştığı düşünülerek diğer adımlara devam edilmiştir. Bu işlemler sonucunda elde edilen tecrübelere göre sistemin geliştirilmesi ve güncellenmesi gereken noktalar geleceğe yönelik öneriler kısmında verilecektir.

6.3 Xilinx Vivado

Bu kısımda ilk olarak Model Composer'dan oluşturulan IP ile bir proje oluşturma anlatılacak. Projede seçilmesi gereken konfigürasyonlara ve elemanlar hakkında bilgi verilecektir. Daha sonra Vitis'te Microblaze kontrolü için oluşturulan .elf dosyasının projeye eklenip çalıştırılması gösterilecektir. Son olarak da Model Composer'ın ürettiği IP'de karşılaştığımız soruna değinilecektir.

İlk olarak üretilen IP'nin oluşturulduğu dizine gidilir ve dosyalar kontrol edilir. Burada proje oluşturmak için iki yöntem kullanılmıştır. İlk denenen ve normalde kullanılması gereken yöntem Vivado'da yeni bir proje oluşturulup IP catalog bölümüne girilir.

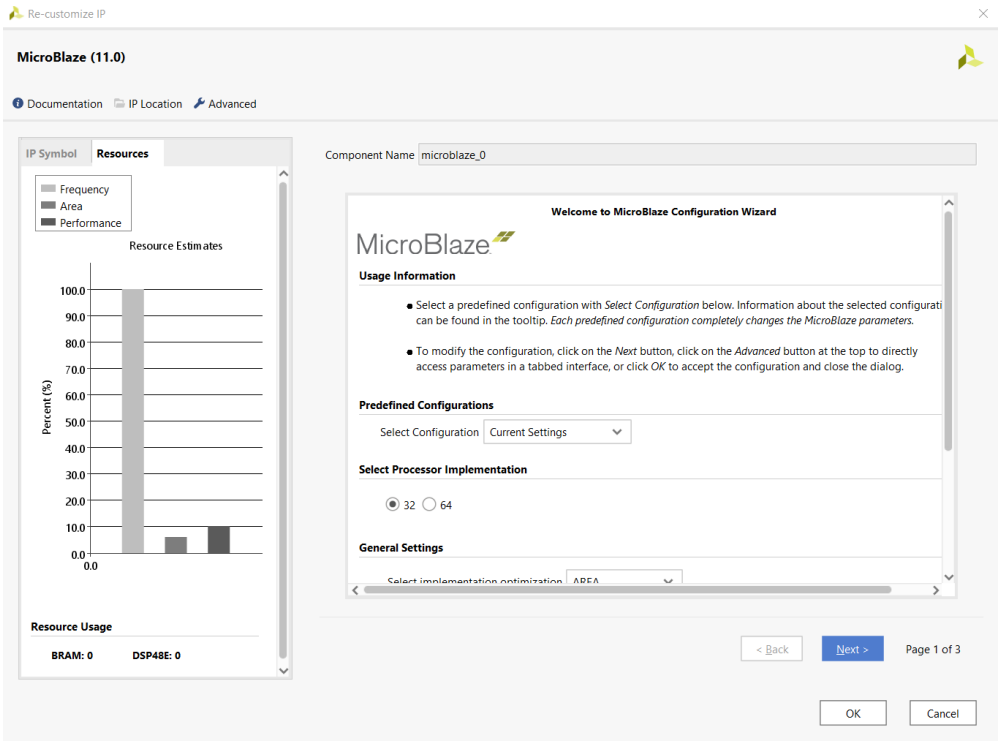
Burada bir alt başlığa sağ tıklanıp add repositorye tıklanır ve üretilen dizinden dosyalar seçilir. Daha sonra bu bloklar IP katalogdan seçilebilir hale gelmiş olur.



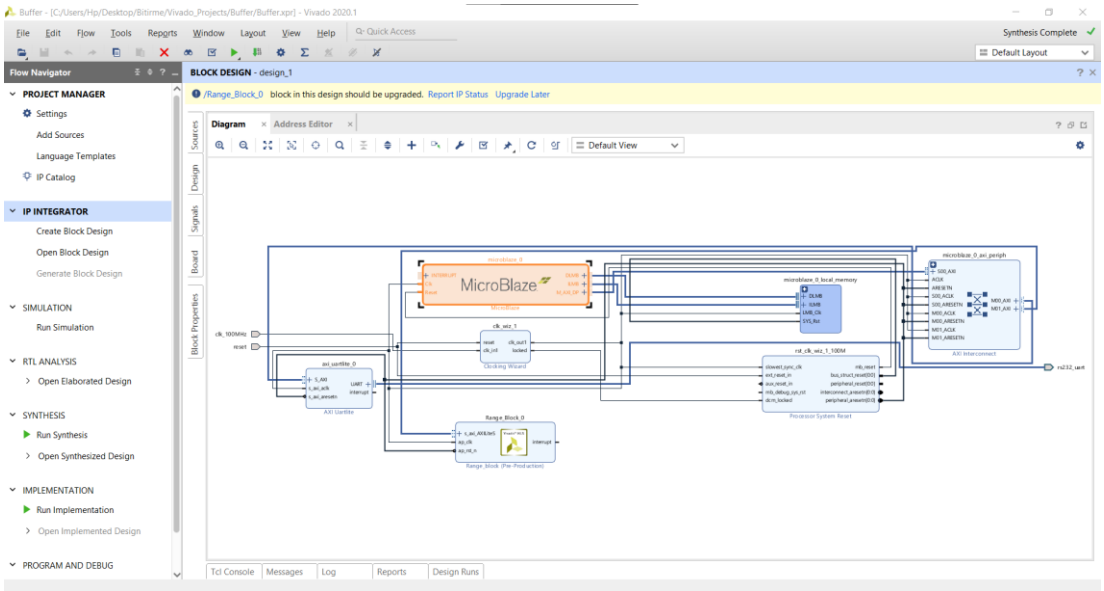
Şekil 6.23 : IP katalog'a yeni IP eklenmesi

Diğer yöntemde ise IP oluşturulurken program üretilen IP ile bir Vivado projesi oluşturmaktadır. Doğrudan bu projeye gidilerek blok tasarım üzerinde kendi istediğimiz değişimleri yaparak sıfırdan tüm bağlantıları yapılması zorunda kalınmaz. Bu yöntemin kullanılma nedeni menzil ve azimut işleme de kullanılan FFT IP'leri üretilen IP'den bağımsız şekilde iki farklı IP üretmektedir. Bu IP'lerden HLS sembolü olan IP'ye ekstra iki FFT IP'sini elle bağlayarak parametreleri belirlenmesi gerekiyor. Bunun yerine programın ürettiği proje üzerinden tasarım yapmak kolaylık sağlamaktadır.

Blok tasarım ekranına geçildikten sonra ilk olarak Microblaze elemanı seçilir. Ardından reset ve clk bağlantıları yapılır. Burada clk bağlantısı yapılırken FPGA'e uygun clk seçilmelidir. Diferansiyel girişli olursa giriş işareti diferansiyel bağlanmalıdır. Şekil 6.24'te Microblaze ayar ekranı görülmektedir. Buradaki ayarlar tamamlandıktan sonra hafıza blokları yerleştirilir ve bu aşama tamamlanır. Daha sonra üretilen IP tasarıma eklenir ve AXI bağlantıları Microblaze ile yapılır. Bu aşamanın ardından ilk olarak sol sekmede kaynaklar kısmında bulunan sarı işaretli dosyaya sağ tıklanır ve Create HDL Wrapper seçilir . Böylece sentez aşamasına geçilir ve tasarım sentezlenir. Şekil 6.25'te oluşan blok diyagram gösterilmektedir.

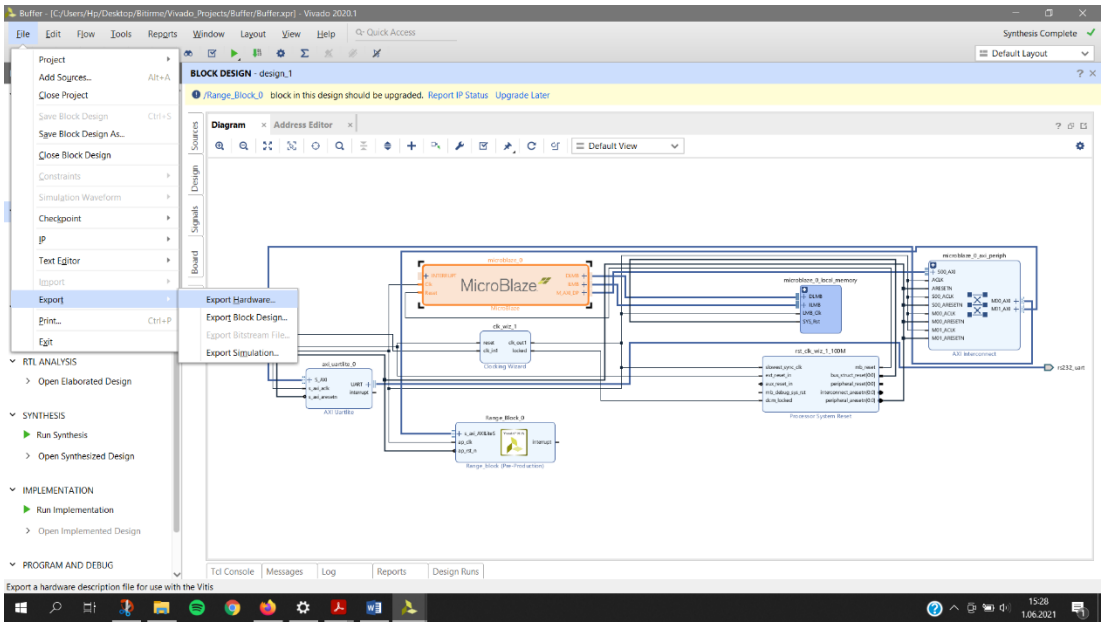


Şekil 6.24 : Microblaze bağlantı ve debug ayarları ekranı



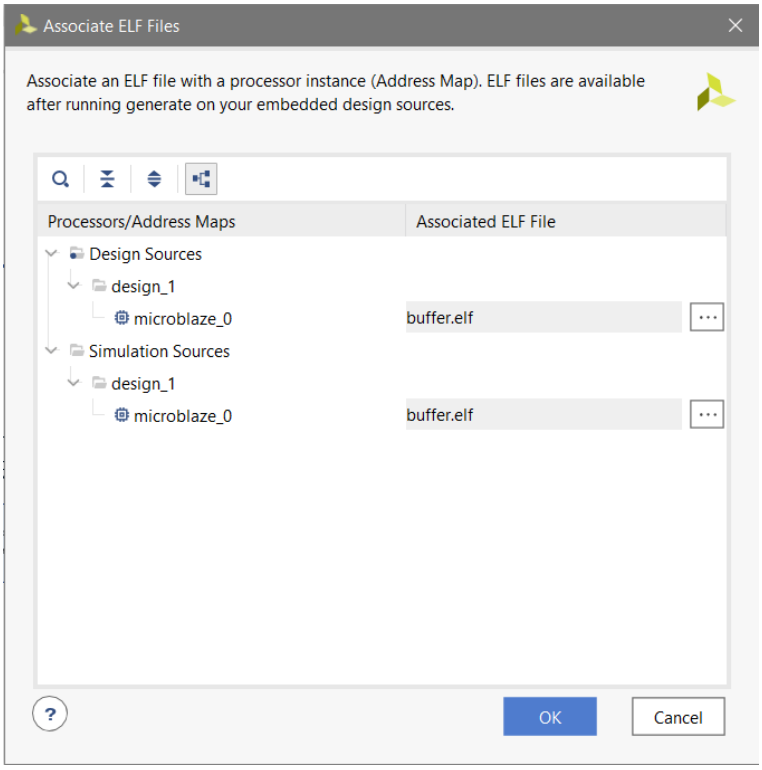
Şekil 6.25 : Sistem blok diyagramı

Daha sonra üst menüde bulunan file sekmesinden export'a oradan da export hardware seçeneğine tıklayarak bir .xsa dosyası oluşturulur. Oluşturulan bu dosya ile Vitis'e geçilir ve burada Microblaze ile kullanılan IP'nin giriş çıkışları ve tüm kontrollerini yapacak bir kod yazılır ve ardından Vitis tarafından bir .elf dosyası oluşturulur.



Şekil 6.26 : Donanımı dışarı aktarma seçme konumu

Oluşturulan .elf dosyası aşağıdaki şekildeki gibi bir konumdan seçilerek projenin hem simülasyon hem de tasarım da dosyalarına eklenmesi gerekmektedir. Aynı zamanda eklenen dosyalar üst sekmede bulunan Tools seçeneğinin altındaki Associate ELF Files sekmesinden eklenen dosyalar hem tasarım hem de simülasyon için ayrı ayrı eklenmelidir.



Şekil 6.27 : .elf dosyası ilişkilendirme seçme arayüzü

Bu aşamanın ardından tasarlanan sistemin simülasyonu yapılmaya hazır hale gelmiş olur. İlk olarak sistem için bir testbench yazılır. Bu testbench'te clk ve reset girişleri sisteme verilir ve üretilen IP waveformda gözlemlenir.

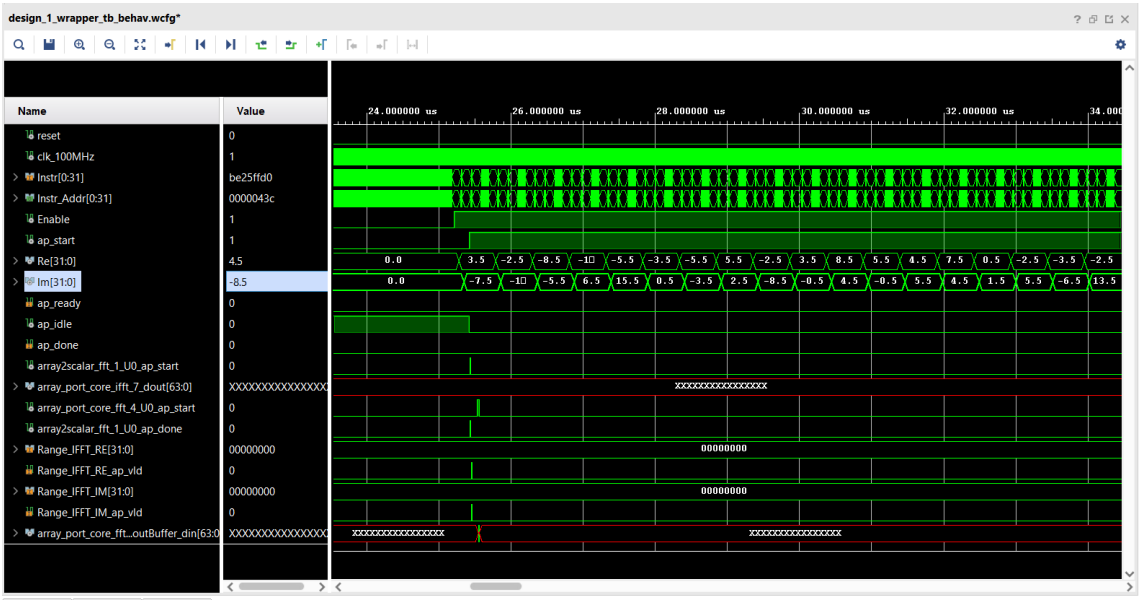
tb_buf.vhd

C:/Users/Hp/Desktop/Bitirme/Vivado_Projects/Buffer/Buffer.srcs/sim_1/new/tb_buf.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity design_1_wrapper_tb is
5  end design_1_wrapper_tb;
6
7  architecture tb of design_1_wrapper_tb is
8
9  component design_1_wrapper
10 port (    reset : in STD_LOGIC;
11           clk_100MHz : in STD_LOGIC;
12           rs232_uart_rxd : in STD_LOGIC;
13           rs232_uart_txd : out STD_LOGIC);
14 end component;
15
16     signal clk_100MHz     : std_logic := '0';
17     signal reset          : std_logic;
18     signal rs232_uart_rxd : std_logic;
19     signal rs232_uart_txd : std_logic;
20     constant TbPeriod : time := 10 ns; -- EDIT Put right period here
21 begin
22     dut : design_1_wrapper
23     port map (clk_100MHz     => clk_100MHz,
24               reset         => reset,
25               rs232_uart_rxd  => rs232_uart_rxd,
26               rs232_uart_txd  => rs232_uart_txd);
27     -- Clock generation
28     clk_100MHz <= not clk_100MHz after TbPeriod/2;
29
30     stimuli : process
31     begin
32        -- Reset generation
33         reset <= '1';
34          wait for 4*TbPeriod;
35
36         reset <= '0';
37         wait for 2*TbPeriod;
38
39         wait;
40     end process;
41 end tb;
```

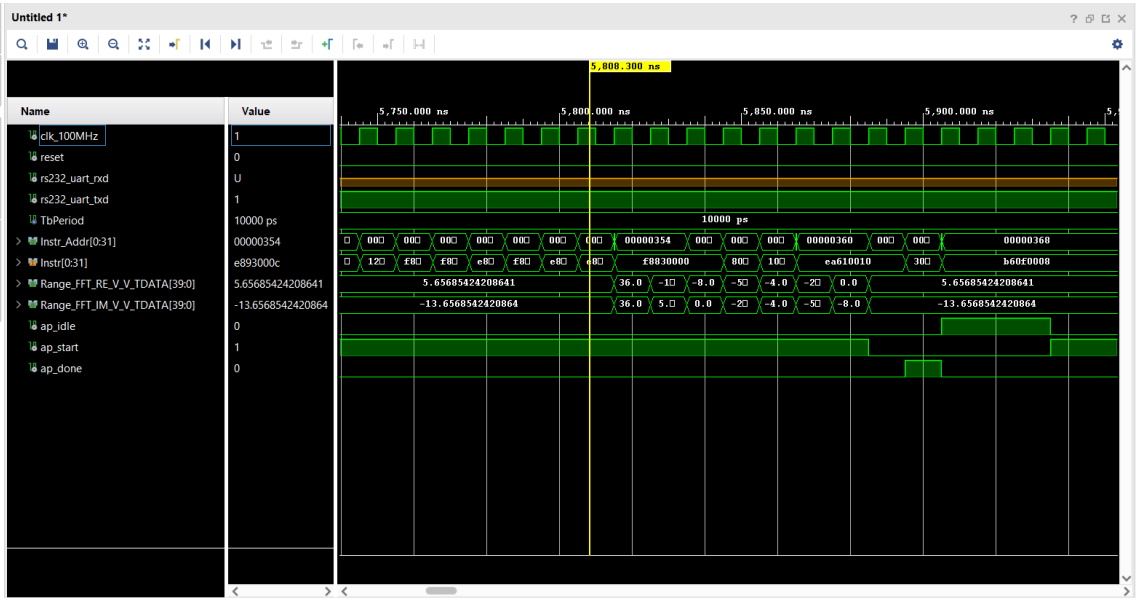
Şekil 6.28 : Testbench kodu

Şekil 6.29'da devrenin simülasyonu verilmiştir. Görüldüğü gibi ap_start sinyali yalnız bir kez geliyor. Bu da gelen ilk veriden sonra başka veri gelmesinin önüne geçiyor.



Şekil 6.29 : Sistem simülasyonu ile oluşan waveform

Şekil 6.29'da da görüldüğü gibi başlangıçta bir start işareti geliyor, daha sonra gelmiyor. Bu da sistemin akışını bozmaktadır. Dışardan gelen Re ve Im sinyalleri düzgünce gelmekte ancak içerde bulunan FFT ve diğer birimlerin akışı bozulmaktadır. Bu da FFT kullanılan IP'lerin uyumsuzluk çıkardığı anlaşılmaktadır. Buradaki sorunlardan biri Model Composer'ın vektör girişi alıp Vivado'nun skaler olarak girişi beklemesidir. Sistemde yalnızca FFT IP'si kullanıldığında sistem çalışmaktadır. Ayrıca denenen bu sistem 8 boyutlu bir FFT işlemini yapmaktadır. Sisteme verilen veriler çıkışta doğru sonuçlar vermiştir. Bu sistemin waveformu şekil 6.30'da verilmiştir.

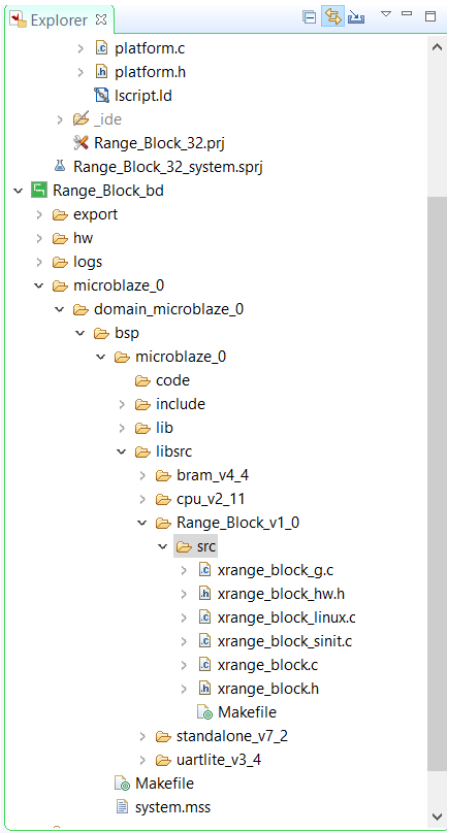


Şekil 6.30 : Tekil FFT IP projesi waveformu

Başlangıçta FPGA kaynak yetersizliği olduğu düşünülmüş ancak bunun da olmadığı farkedilmiş ve sorunun çözümü için denenecek yeni yollar geleceğe yönelik öneriler de verilecektir.

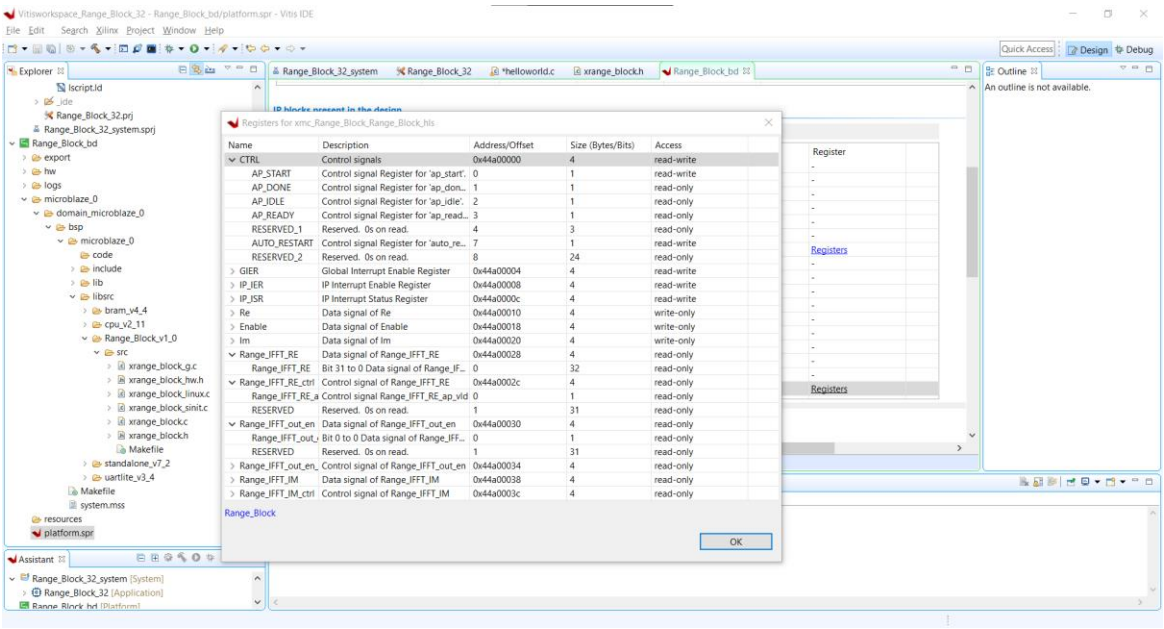
6.4 Vitis

Vitis programı Microblaze'i kontrol edecek C kodunun yazıldığı ve elf dosyasının bu kodla üretildiği programdır. Vivado'da dışa aktarmayla elde edilen xsa dosyasıyla yeni bir proje oluşturulur. Proje oluşturmanın ardından kaynaklara gidilir ve bir kontrolcü kodu yazılır. Kod yazmadan önce Üretilen IP'nin yazma ve okuma fonksiyonları şekil 6.31'de verilen dizindeki xrange_block.h dosyasından bulunur ve bu fonksiyonlar yardımıyla istenen adresteki registerlara giriş verileri ve start sinyalleri verilebilir.

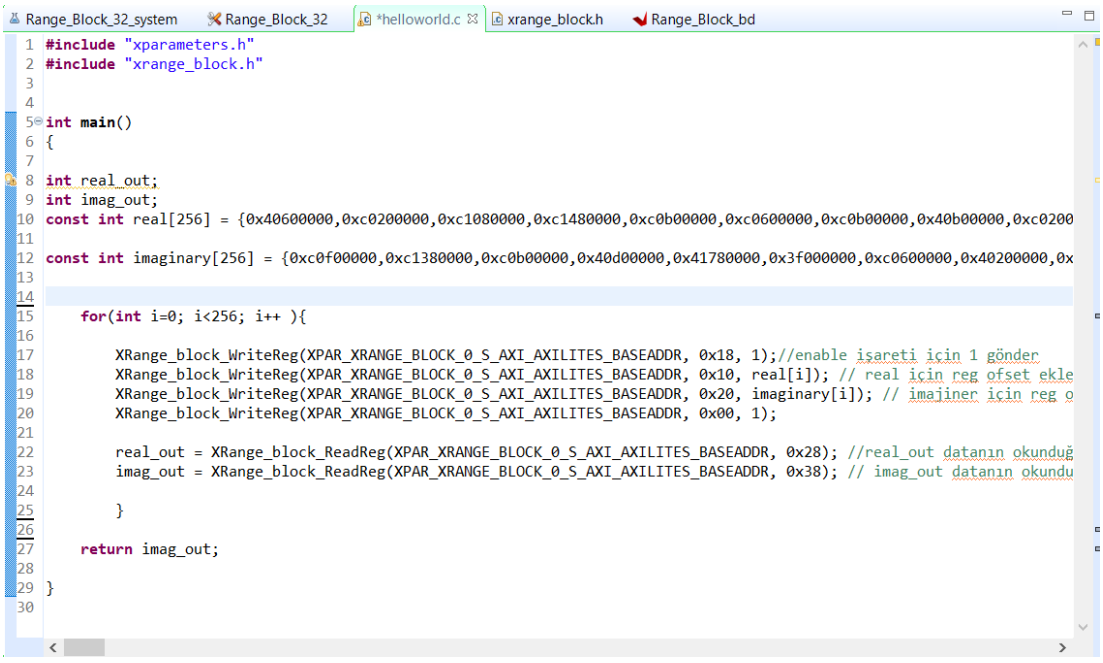


Şekil 6.31 : xrange_block.h dosyası konumu

Daha sonra sistem giriş çıkış adreslerini bulmak için şekil 6.32'deki konuma gidilir ve giriş çıkış adresleri bulunur. Address/Offset sekmesinde bulunan değerler bulundukları adresleri göstermektedir. Bu değerler okuma yazma fonksiyonlarına girilirken bir base adres ve üzerine eklenen ofset şeklinde girilir. Base adres değişken ismi xparameters.h dosyasının içinden çekilerek fonksiyonlara eklenir. Yanındaki değişkene de kullanılan giriş veya kontrol sinyalinin ofset değeri girilir.



Şekil 6.32 : Giriş, çıkış ve kontrollerin adresleri



Şekil 6.33 : Microblaze kontrolü için Vitis'te yazılan kod

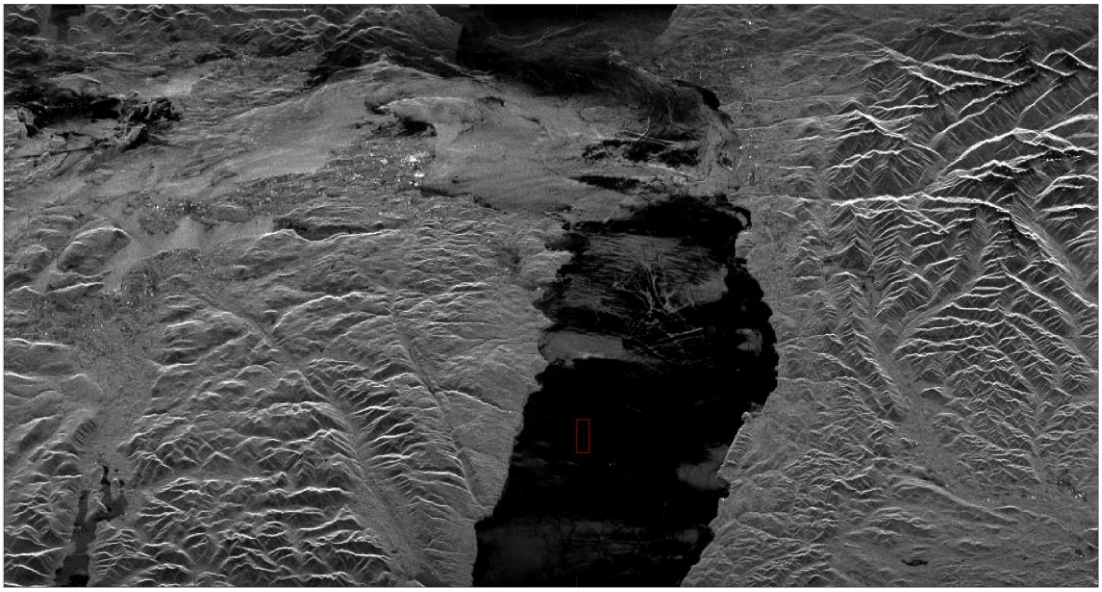
Sistemin çalışması için yazılması gereken şekilde kodlar yazılır ve üst sekmeden build run seçeneğine tıklanır ve aşağıdaki konsoldan elf dosyasının üretilip üretilmediği gözlemlenir. Daha sonra Vivado'da kullanılmak üzere üretildiği dizinden seçilerek kullanılır.

7. YAPAY SİNİR AĞININ GERÇEKLENMESİ

Yapay sinir ağı, Menzil Doppler algoritması ile işlediğimiz ham verinin çıktısını kullanarak görüntünün aktarılmaya uygun olduğuna karar vermesi istenen mekanizmadır. Bu yüzden yapay sinir ağının eğitiminde kullanacağımız görüntü, testinde kullanacağımız görüntü ile aynı özelliklere sahip olması için eğitimde Menzil Doppler algoritması ile işlenmiş görüntü kullanılmalıdır. Bu sebeple eğitim için kullanılacak görüntü öncelikle MATLAB ortamında ham veriden işlenerek elde edilmiştir. Daha sonra Python ortamında, 3. bölümde anlatılan 2 aşamalı eğitim yapılmış ve test için gerekli katsayılar bulmuştur. Sistemi Vivadoya taşımak için Model Composer ortamında Kohonen ağının test algoritması, Python ortamında bulunmuş olan katsayılar kullanılarak oluşturulmuş ve Model Composer üzerinde simulasyonu yapılmıştır. Simulasyon sonucunda istenilen veriler alındıktan sonra Vivado için IP oluşturulmuş ve Vivado ortamına geçiş yapılmıştır. Vivado ortamında oluşturulan IP, Microblaze ve Axi ile sentezlenip Vitis ortamı için xsa dosyası oluşturulmuştur. Vitis ortamında, oluşturulan xsa dosyası tanımlanıp Microblaze'in, oluşturulan IP'ye olan bağlantılarını test edebilmek adına C kodu yazılmıştır. Yazılma C kodunun sonuçlarını görüntülemek için Vitis ortamında elf dosyası oluşturulmuş ve IP, Microblaze, Axi ile sentezlediğimiz Vivado projesinde bu elf dosyası tanımlanarak bir test kodu yazılmıştır. Vivado ortamında yapılan test sonucu yapay sinir ağının istenildiği gibi çalıştığına karar verilmiştir. MATLAB ve Python ortamında kullanılan kodlara referanslarda bulunan github linkinden ulaşılabilir.

7.1 MATLAB - Python Ortamı

Kohonen ağının eğitimi için öncelikle ham verinin görüntüye dönüştürülmesi gerekmektedir. Bu sebeple ilk olarak MATLAB ortamında eğitimde kullanılması için seçilmiş görüntüler işlenmiştir. Görüntülerin piksel boyutları çok büyük olduğu için eğitimde yalnızca gerekli yerler kırpılarak kullanılmıştır.



Şekil 7.1 : Eğitim için kırpılan görüntü

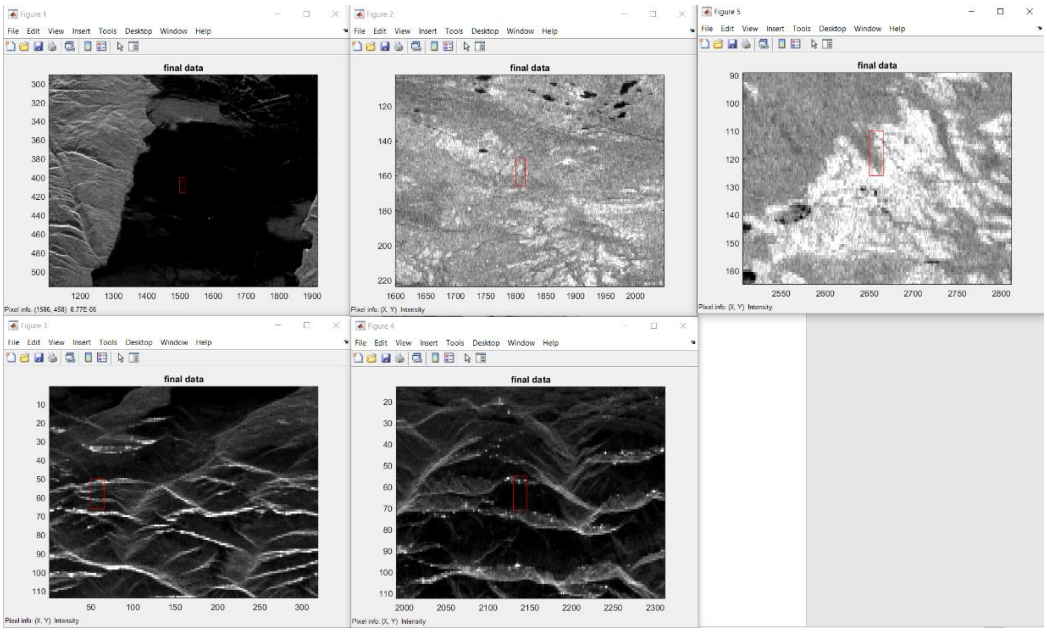
Şekil 7.1'de büyük bir görüntü içerisinden eğitimde kullanılmak üzere yalnızca su verilerini içeren kırmızı işaretli kısım kesilerek kullanılmıştır. 3. bölümde eğitim için kırpılmış bütün görüntüler mevcuttur. Bütün eğitim verileri alındıktan sonra Kohonen ağının eğitimi Python ortamında 2 aşamalı olarak yapılmıştır. Yapılan eğitimin yeterliliğini gözlemlemek için Python ortamında testler yapılmıştır. Eğitim ve test aşaması 3. bölümde detaylı olarak anlatılmıştır. Eğitim yeterli görüldükten sonra bulunan katsayılar Model Composer ortamında kullanılmak üzere .txt dosyalarına yazdırılmıştır.

7.1.1 Uydu görüntüsünde sınıflandırma

Kohonen ağının yapısı gözönüne alındığında uydu görüntüsünden yer şekili tespiti yapmak için Kohonen ağının kullanılması kararlaştırıldı. Bu işlem büyük sistemde bizim işlediğimiz verilerin sınıflandırması üzerine olacağı için yapılacak olan eğitim de bizim işlediğimiz görüntüler üzerinden yapılmalıdır. Bu sebeple eğitim veya test yapılacak olan görüntüler öncelikle işlendi. Uydu görüntüsünde yalnızca genlik bilgisi bulunduğu için 1 boyutlu veriler üzerinden eğitim yapıldı. Eğitim 1 boyutlu veriler üzerinden yapılacağı için mümkün olduğunca detaylı bir eğitim yapılması gerekmektedir. Eğitim sonucunda ise ağın bir uydu görüntüsünde su bulunan yerleri tespit edebilmesi beklenilmektedir.

7.1.1.1 Eğitim

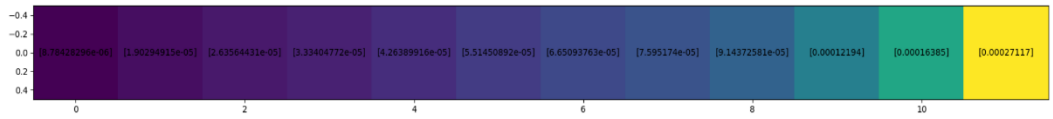
Sınıflandırma yapmak isteğimiz uydu görüntüleri çok fazla veri içerdiği için eğitim kümesini oluştururken her görüntüden belli kesitler seçildi. Seçilen kesitler her türlü yer şekilini içerecek şekilde ayarlandı.



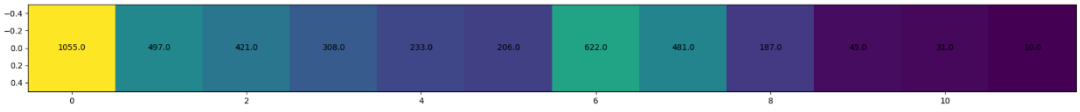
Şekil 7.2 : Uydu görüntüsü eğitim kümesi

Eğitim kümesi için büyük uydu resimlerinden 32x32 piksel boyutunda bölgeler kesildi. Şekil 7.2'de kesilen bu görüntüler görülmektedir. Kesilen bölgeler sırasıyla su, ova, buzul, tepe ve dağ şekillerini içermektedir. Öz düzenlemeli ağ 1 boyutlu ve 12 nöronlu olarak seçildi. Veri değerleri 10e-4 ile 10-7 arasında olduğu için başlangıç ağırlık değerleri 0 ile 10e-5 arasında rasgele olarak seçildi. Öğrenme hızı 0.1, komşuluk katsayı değeri 0.8, öğrenme hızı değişimi 1000, komşuluk katsayısı değişimi 500 olarak seçilerek öz düzenlemeli ağda eğitim yapıldı. Eğitim sonucu nöronların dağılımı verileri temsil edebilecek şekilde dağılarak Şekil 7.3'deki gibi oldu.

Şekil 7.3 : Uydu görüntüsü eğitim sonucu nöronların dağılımı

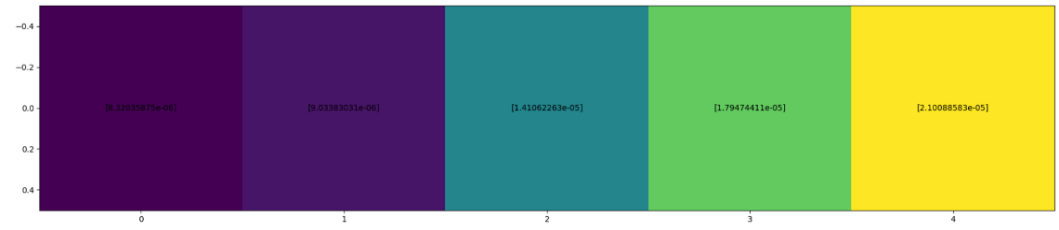


Şekil 7.4 : Uydu görüntüsü eğitim sonucu nöronların değeri

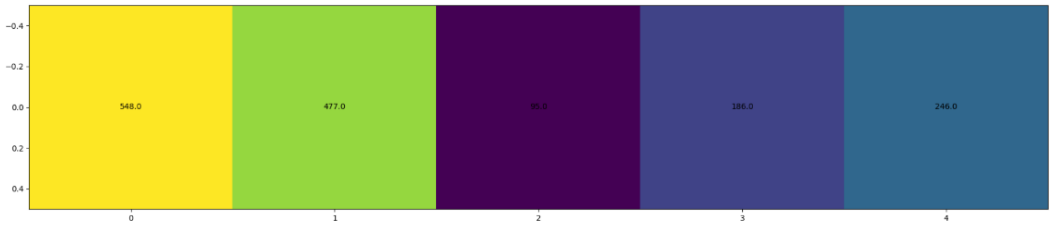


Şekil 7.5 : Uydu görüntüsü eğitim sonucu kazanan ağırlık tablosu

Eğitim sonucu nöronların değerleri Şekil 7.4'te, kazanan ağırlık tablosu ise Şekil 7.5'te görülmektedir. Nöronların dağılımı ve kazanan ağırlık tablosuna bakıldığı zaman genlik değeri en düşük olan su verilerinin ve su verilerine yakın genlikte olan dağ değerlerinin yalnızca iki nöronla temsil edildiği görülmektedir. Yapılan bu eğitim sonucunda testler yapıldığında gözlemlendiği gibi su ile dağ şekillerinin ayrımında istenilen hassaslık elde edilemediği görülmüştür. Bu görüntülerde elde etmek istediğimiz amacın uydu görüntülerindeki su olan yer şekillerini bulmak olduğu için su şekilinin hassaslığının arttırılması önemlidir. Bu sorunu çözmek için 0. ve 1. nöronun yani genlik değeri bakımından suya en yakın olan iki nöronun kazanan verileri arasında tekrar bir eğitim yapılması kararlaştırıldı. Bu eğitim için ilk yapılan eğitimden yalnızca 0. ve 1. nöronun kazanan verileri olan 1552 veri alındı. İkinci eğitim aynı sabit değerleri kullanarak 5 nöronla yapıldı.



Şekil 7.6 : Uydu görüntüsü 2. eğitim sonucu nöronların değeri

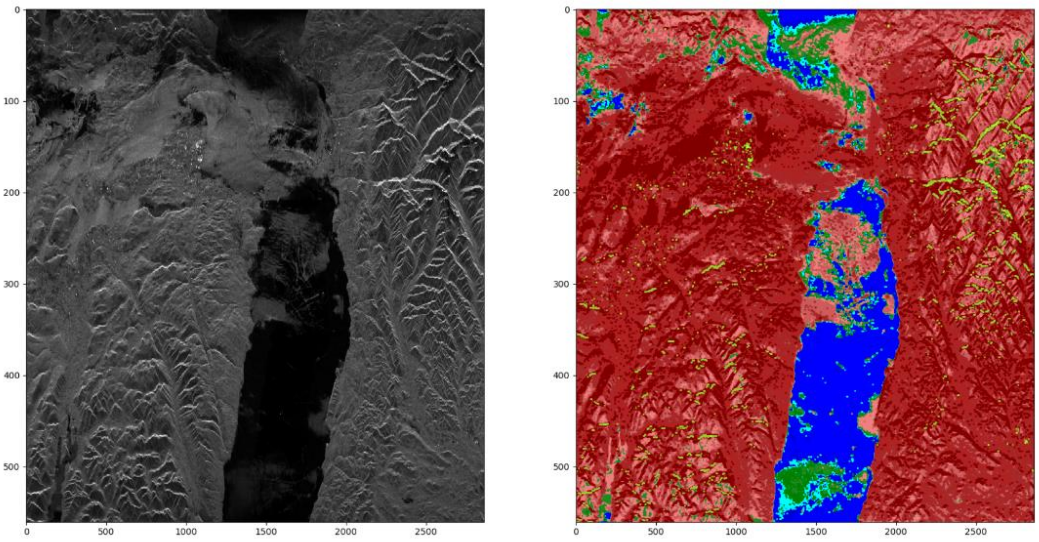


Şekil 7.7 : Uydu görüntüsü 2. eğitim sonucu kazanan ağırlık tablosu

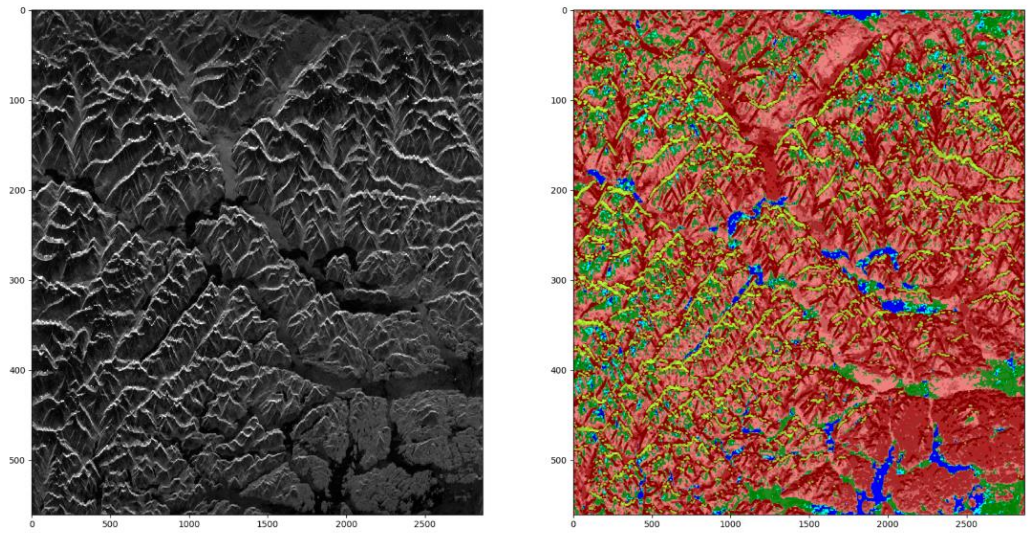
2. eğitim sonucu nöronların değerleri Şekil 7.6'da, kazanan ağırlık tablosu ise Şekil 7.7'de görülmektedir. Bu tablolarına bakılarak yapılan 2. eğitim sonucunda 1552 verinin daha detaylı bir nöron yapısıyla temsil edildiği gözlemlendi. İkinci eğitim sonucu bulunan 5 nöron ilk nörondaki 0. ve 1. nöronun yerine konularak toplamda 16 nöronlu bir ağ yapısı elde edildi.

7.1.1.2 Benzetim sonuçları

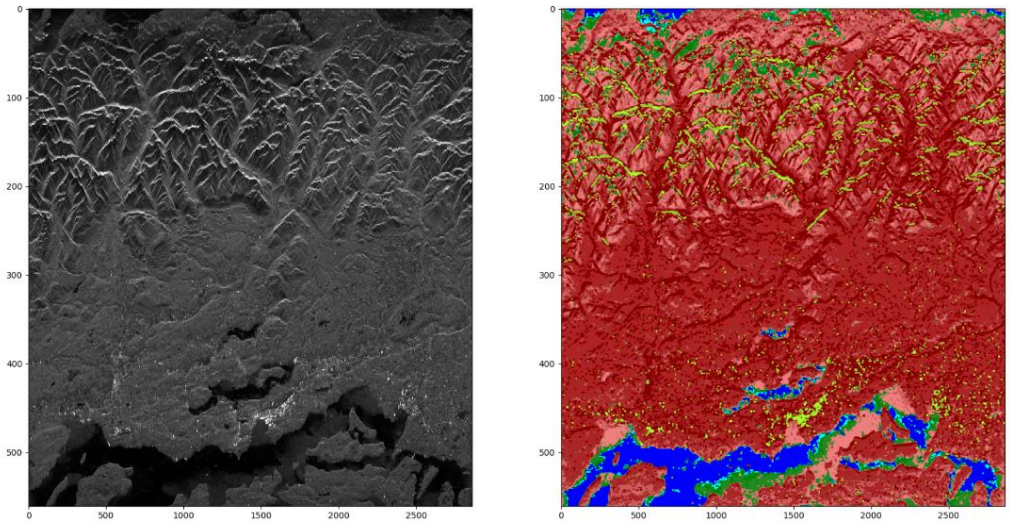
Elde edilen 16 nöronluk ağ yapısı test edebilmek için 3 farklı uydu görüntüsünün tamamı işlenerek ağa sunuldu ve Şekil 7.8'deki sonuçlar elde edildi.



Şekil 7.8 : Uydu görüntüsü test sonucu-1



Şekil 7.9 : Uydu görüntüsü test sonucu-2



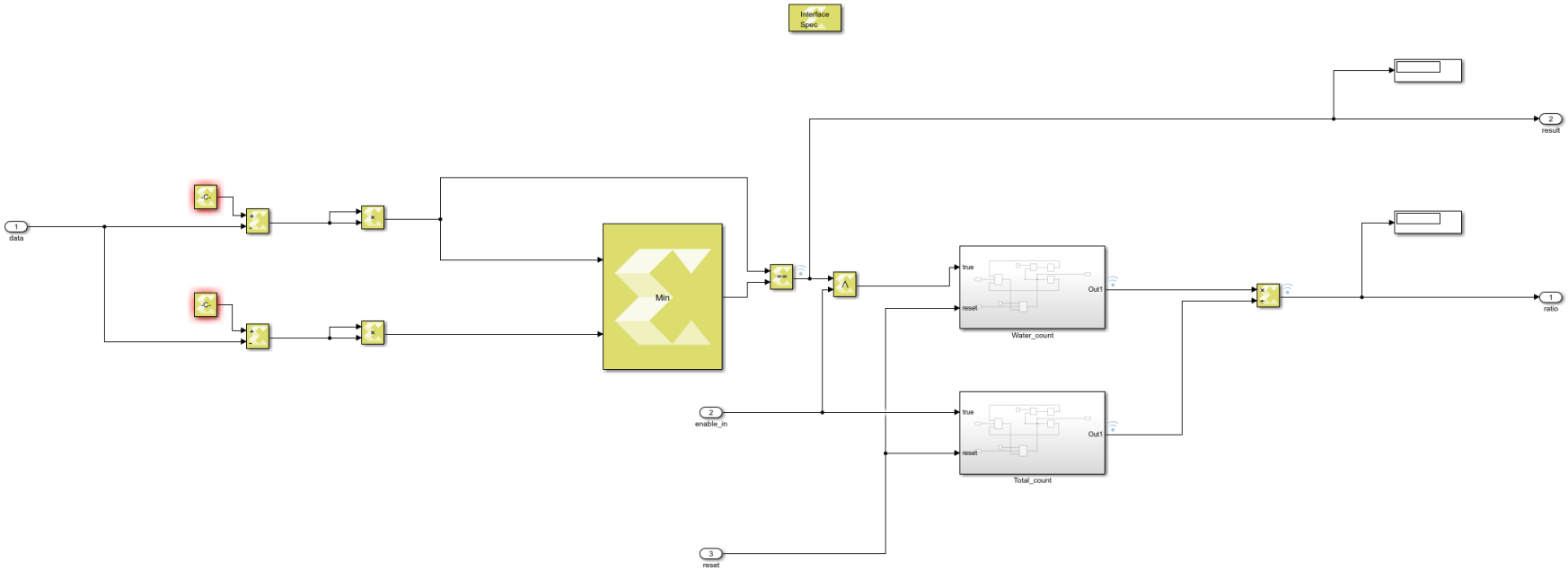
Şekil 7.10 : Uydu görüntüsü test sonucu-3

Şekil 7.10 görsel olarak incelendiğinde işlenmiş bir uydu görüntüsünde su olan yerlerin hassas bir şekilde bulunabildiği gözlemlendi. İlk eğitimde oluşan dağ olan yerleri de su olarak sınıflandırma sorunu 2. eğitimle birlikte büyük ölçüde iyileştirilmiş olsa da bazı dağlık arazilerde hala hata ortaya çıkmaktadır.

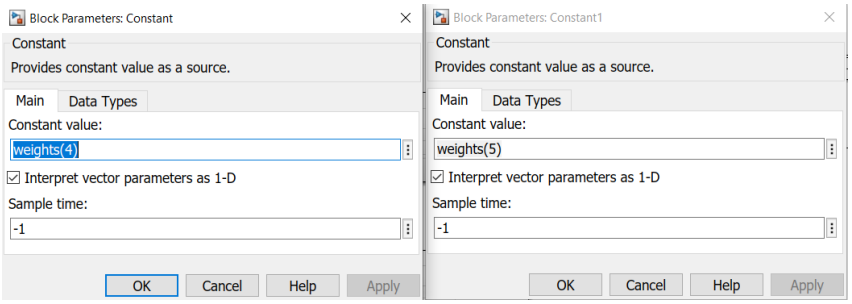
7.2 Model Composer

Model Composer ortamında test algoritması gerçeklenmiştir. Kohonen test algoritmasında bir girişin en yakın olduğu nöron bulunarak girişin ait olduğu sınıfa karar verilirken bize yalnızca girişin su grubunda olup olmadığı bilgisi yeterli olacağı

için oluşturulan yapıda verimliliğin arttırılması için yalnız su grubunu temsil eden nöronlar ve onun komşu nöronlarını içeren bir nöron yapısı yeterli görülmüştür. 3. bölümde yapılan görsel testler sonucu su grubunu temsil eden nöronların ilk 4 nöron olduğu gözlemlenmiştir. Bu sebeple Model Composer ortamında oluşturulan yapı yalnızca 4. ve ona komşu olan 5. nöronu içerecek şekilde oluşturulmuştur. Bu sayede ilk 4 nöronun grubunda olan her giriş 4. nöronun grubunda sayılacak, 5. ve daha büyük değerlere sahip nöronların grubunda olan her giriş 5. nöronun grubunda sayılacak.

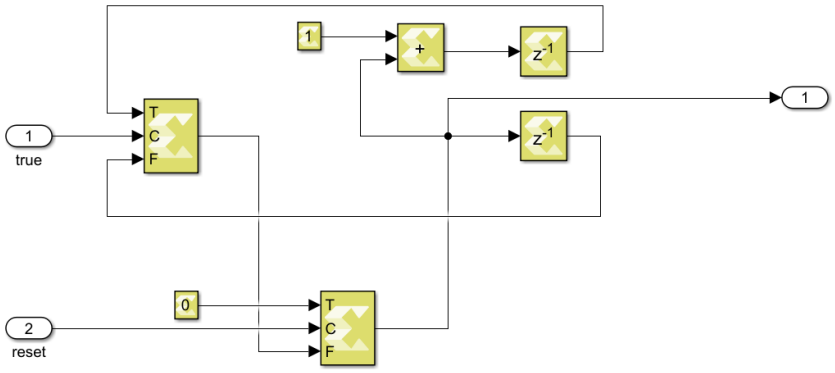


Şekil 7.11 : Yapay sinir ağı Model Composer yapısı



Şekil 7.12 : Yapay sinir ağı Model Composer nöron sabitleri

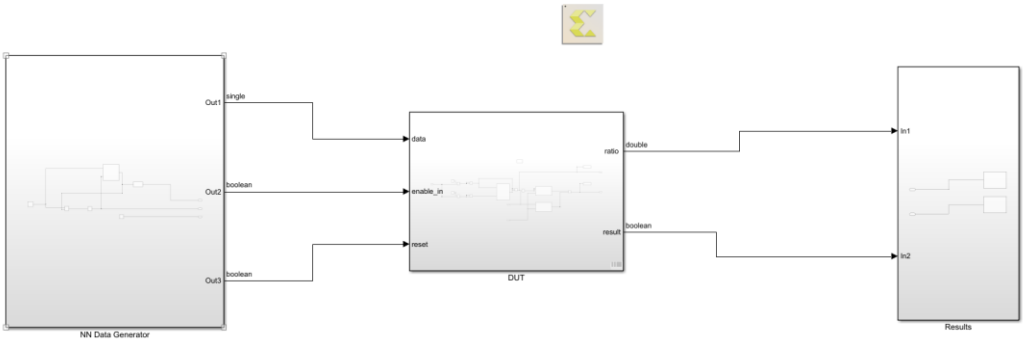
Şekil 7.11'de verilen yapıda verilen her giriş değeri için öz düzenlemeli ağ test algoritması uygulanmıştır. Algoritma sonucunda giriş değeri 4. nörona daha yakınsa result çıkış sinyaline 1, değilse 0 çıkışı verilmektedir. Burada 4. ve 5. nöron değerleri sabit olarak verilerek MATLAB Workspace üzerinden tanımlanmıştır.(Şekil 7.12) Her alınan girişin ardından toplam sayıcı alt bloğuna aktif sinyali verilmiştir. Her su olduğu tespit edilen giriş ardından ise su sayıcı alt bloğuna aktif girişi verilmiştir. Bir giriş dizisi sonucunda ise ratio çıkış sinyalinden giriş dizisinin yüzdesel olarak ne kadar su içerdiği çıkışı verilmektedir. Burada bir yere bağlanmayan Interface Spec bloğu üretilecek IP'nin diğer bloklarla arasındaki haberleşme arayüzünün nasıl olacağını belirler. Handshake, AXI-Stream, AXI-Lite Slave, FIFO ve AXI-Stream(Video) gibi olanak sağlamaktadır. Bu proje kapsamında AXI-Lite Slave protokolü kullanılmıştır.



Şekil 7.13 : Yapay sinir ağı sayıcı alt bloğu yapısı

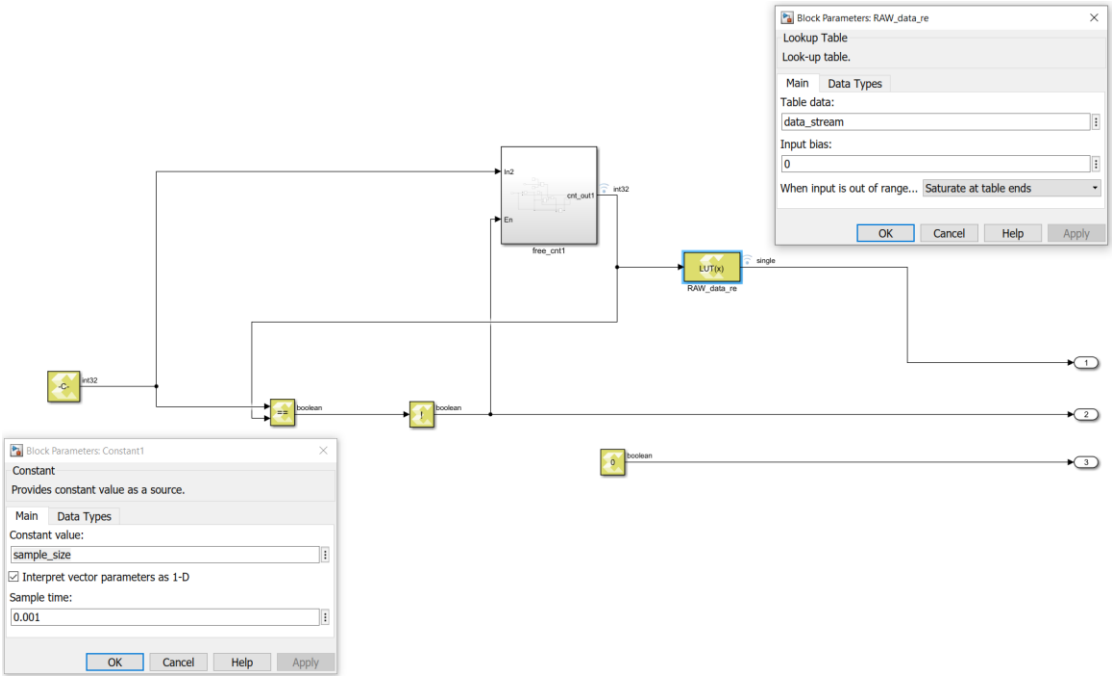
Toplam sayıcı altbloğu ve su sayıcı alt bloğu yapısı Şekil 7.13'deki gibidir. Aktif sinyali 1 iken artıp, 0 iken değişmeyen ve reset sinyali geldiğinde sıfırlanın bir sayaç içeren yapıdır.

Model yapısı tamamlandıktan sonra yapının simülasyonun yapılması için bütün yapı DUT adı altında bir alt bloğa alınarak oluşturulan yapıya giriş uygulanmış ve çıkış okunmuştur.

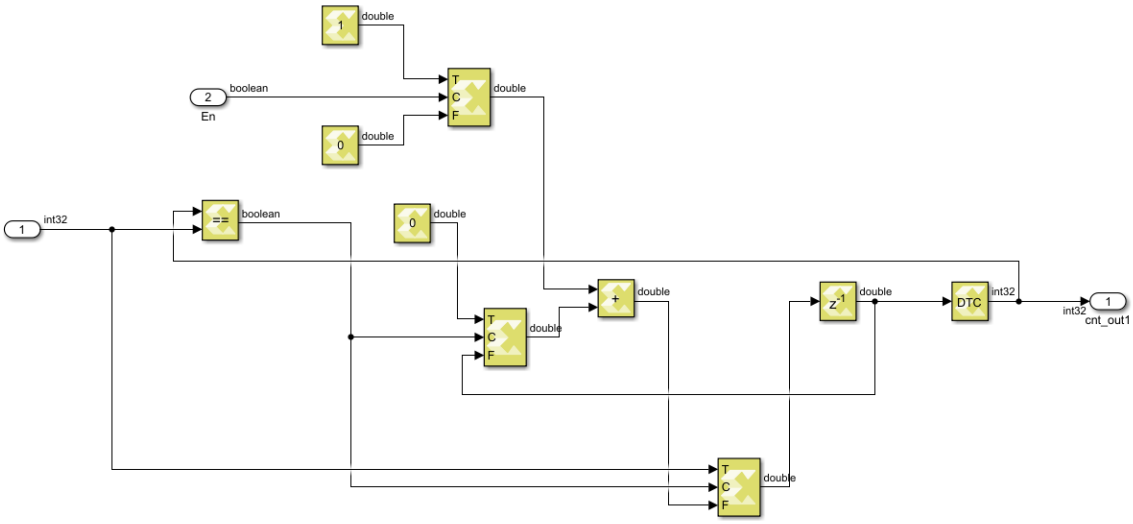


Şekil 7.14 : Yapay sinir ağı simülasyon modeli

Şekil 5.51'de uygulanacak olan giriş ve okunacak olan çıkış bloklarının yapısı büyük modelde gösterilmiştir. Model Composer'da tasarlanan sistem Vivado'da kullanılmak üzere IP paketine dönüştürülmek için Model Composer Hub elemanı eklenmiştir. Giriş uygulamak için kullanılan alt model (NN Data Generator) Şekil 7.15'de gösterilmiştir.



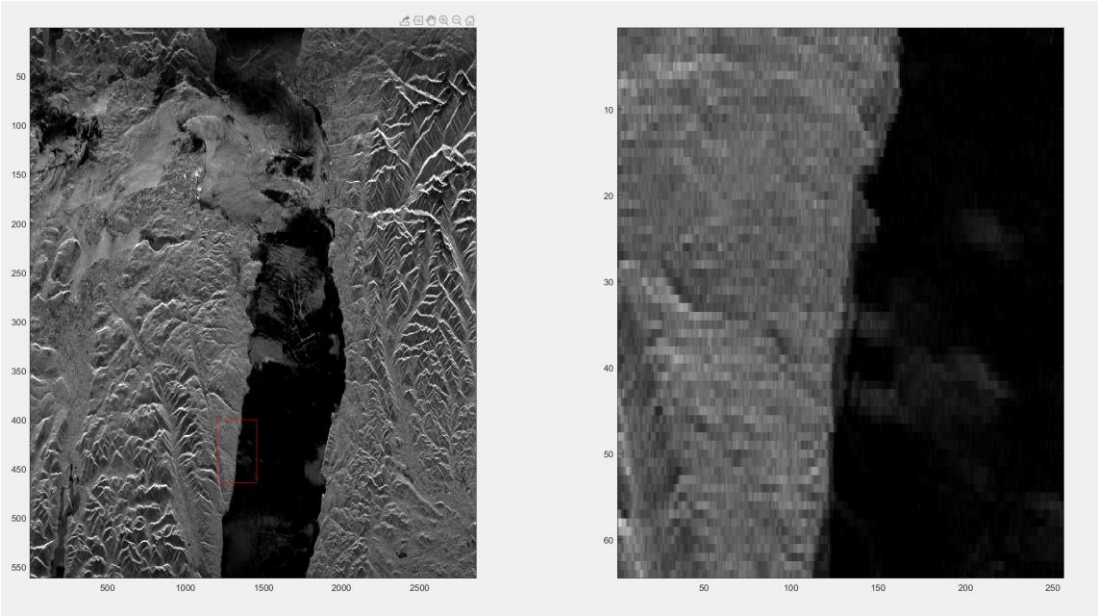
Şekil 7.15 : Yapay sinir ağı giriş modeli



Şekil 7.16 : Yapay sinir ağı giriş modeli sayıcısı

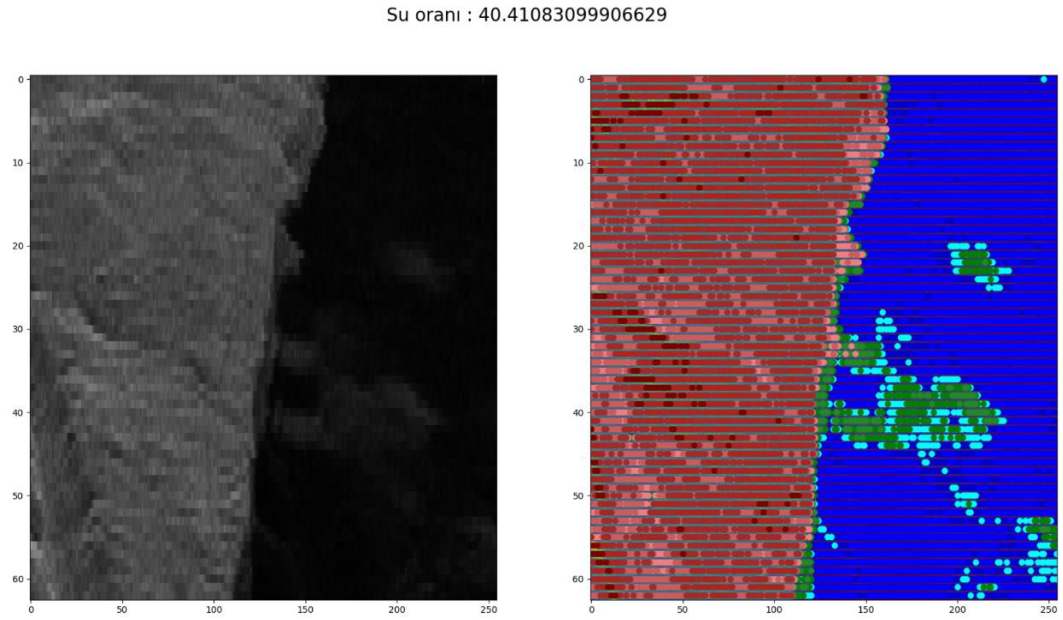
Şekil 7.15'deki yapı vektörel olarak verilen 'data_stream' değişkeninin içeriğini sırayla 1 çıkışından verir. Sayıcı çıkışı, 'sample_size' değişkenine eşit değilken 2 çıkışından 1 değeri, eşitken 2 çıkışından 0 değeri verir ve sayıcıyı durdurur. Model Composer aşamasında kullanılan değişkenler MATLAB Workspace üzerinden tanımlanmıştır.

Simülasyon için gerekli büyük model tamamlandıktan sonra simülasyonu yapılacak olan bölge belirlenmiştir.



Şekil 7.17 : Yapay sinir ağı giriş simülasyon girişi

Şekil 7.17'de belirlenen bölge vektörel bir giriş haline getirilip modelin simülasyonu yapılmıştır. Model Composerdaki modelin doğruluğunu ölçmek amacıyla aynı veriler ile Python ortamından da test yapılmıştır.



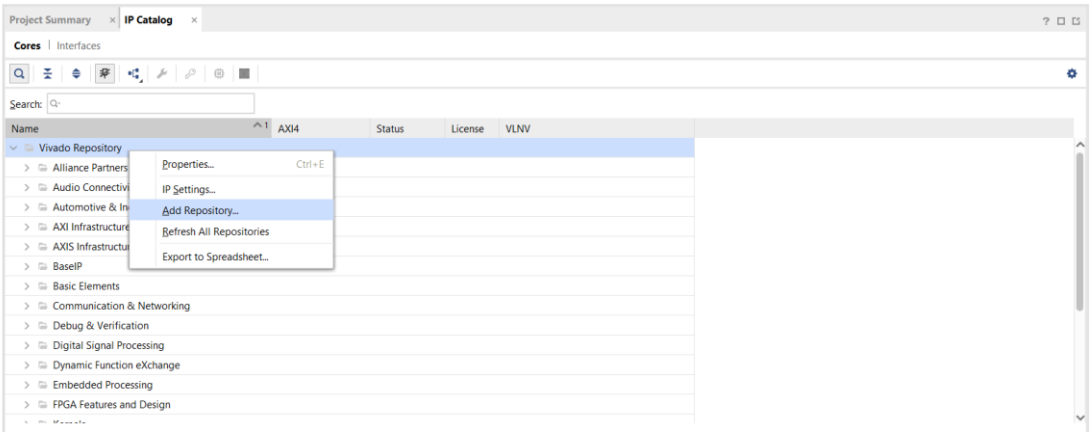
Şekil 7.18 : Yapay sinir ağı giriş simülasyon sonucu

Şekil 7.18'de Python ortamının çıkışı sonucu %40.41 su oranı bulunmuştur. Model Composer sonucu ise %40.20 olarak bulunmuştur. Aradaki hata oranı Model Composer'ın kullandığı virgül bitinde ki yuvarlamadan kaynaklanmaktadır. Hatanın kabul edilebilir düzeyde olduğuna karar verilerek oluşturulan modelin Vivado ortamına taşınması için IP üretimi gerçekleştirildi. IP üretimi için Model Composer Hub elemanından aygıt olarak xc7a200tfbg676-2 numaralı fpga ve 100 Mhz saat frekansı seçildi.

7.3 Xilinx Vivado

Bu kısımda ilk olarak Model Composer'dan oluşturulan IP ile bir proje oluşturma anlatılacak. Projede seçilmesi gereken konfigürasyonlara ve elemanlar hakkında bilgi verilecektir. Daha sonra Vitis'te Microblaze kontrolü için oluşturulan .elf dosyasının projeye eklenip çalıştırılması gösterilecektir.

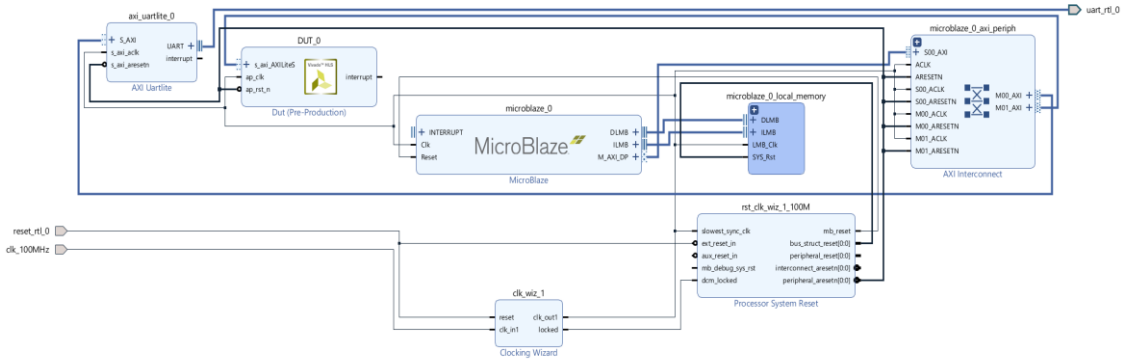
İlk olarak üretilen IP'nin oluşturulduğu dizine gidilir ve dosyalar kontrol edilir. Vivado'da Model Composer IP'sini üretirken seçtiğimiz FPGA seçilerek yeni bir proje oluşturulup IP katalog bölümüne girilir.



Şekil 7.19 : : IP katalog

IP katalog bölümünden Vivado Repository'e sağ tıklanarak Add Repository denir ve Model Composer IP'sinin üretildiği dizin seçilir.

Oluşturulan blok eklendikten sonra Blok tasarım ekranına girilir. İlk olarak Microblaze elemanı seçilir. Ardından reset ve clk bağlantıları yapılır. Burada clk bağlantısı yapılırken FPGA'e uygun clk seçilmelidir. Diferansiyel girişli olursa giriş işareti diferansiyel bağlanmalıdır. Buradaki ayarlar tamamlandıktan sonra hafıza blokları yerleştirilir ve bu aşama tamamlanır. Daha sonra üretilen IP tasarıma eklenir ve AXI bağlantıları Microblaze ile yapılır. Bu aşamanın ardından ilk olarak sol sekmede kaynaklar kısmında bulunan sarı işaretli dosyaya sağ tıklanır ve Create HDL Wrapper seçilir . Böylece sentez aşamasına geçilir ve tasarım sentezlenir. Şekil 7.20'de oluşan blok diyagram gösterilmektedir.



Şekil 7.20 : Yapay sinir ağı blok diyagramı

Daha sonra üst menüde bulunan file sekmesinden export'a oradan da export hardware seçeneğine tıklayarak bir .xsa dosyası oluşturulur. Oluşturulan bu dosya ile Vitis'e geçilir ve burada Microblaze ile kullanılan IP'nin giriş çıkışları ve tüm kontrollerini yapacak bir kod yazılır ve ardından Vitis tarafından bir .elf dosyası oluşturulur.

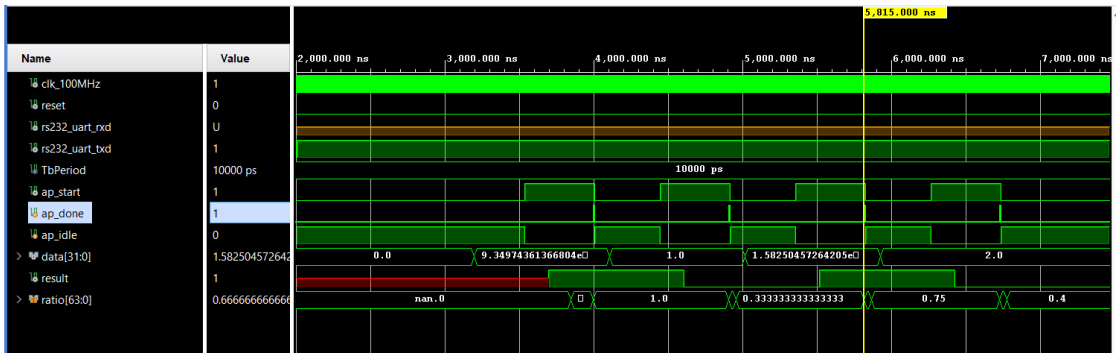
Oluşturulan .elf dosyası projeye hem simülasyon hem de tasarım dosyalarına eklenir. Aynı zamanda eklenen dosyalar üst sekmede bulunan Tools seçeneğinin altındaki Associate ELF Files sekmesinden hem tasarım hem de simülasyon için ayrı ayrı seçilmelidir.

Bu aşamanın ardından tasarlanan sistemin simülasyonu yapılmaya hazır hale gelmiş olur. İlk olarak sistem için bir testbench yazılır. Bu testbench'te clk ve reset girişleri sisteme verilir ve üretilen IP waveformda gözlemlenir.

```
library ieee;
use ieee.std_logic_1164.all;

entity design_1_wrapper_tb is
end design_1_wrapper_tb;

architecture tb of design_1_wrapper_tb is

component design_1_wrapper
port (    reset_rtl_0 : in STD_LOGIC;
          clk_100MHz : in STD_LOGIC;
          uart_rtl_0_rxd : in STD_LOGIC;
          uart_rtl_0_txd : out STD_LOGIC);
end component;

    signal clk_100MHz     : std_logic := '0';
    signal reset          : std_logic;
    signal rs232_uart_rxd : std_logic;
    signal rs232_uart_txd : std_logic;

    constant TbPeriod : time := 10 ns; -- EDIT Put right period here

begin

    dut : design_1_wrapper
    port map (clk_100MHz    => clk_100MHz,
              reset_rtl_0   => reset,
              uart_rtl_0_rxd  => rs232_uart_rxd,
              uart_rtl_0_txd  => rs232_uart_txd);

    -- Clock generation
    clk_100MHz <= not clk_100MHz after TbPeriod/2;

    stimuli : process
    begin
       -- Reset generation
        reset <= '1';
         wait for 4*TbPeriod;

        reset <= '0';
        wait for 2*TbPeriod;

        wait;
    end process;

end tb;
```

Şekil 7.21 : Yapay sinir ağı vivado test kodu

Simulasyon çalıştırıldığında verilen farklı değerlerin hepsi için doğru sonuçların verildiği görülmüştür.

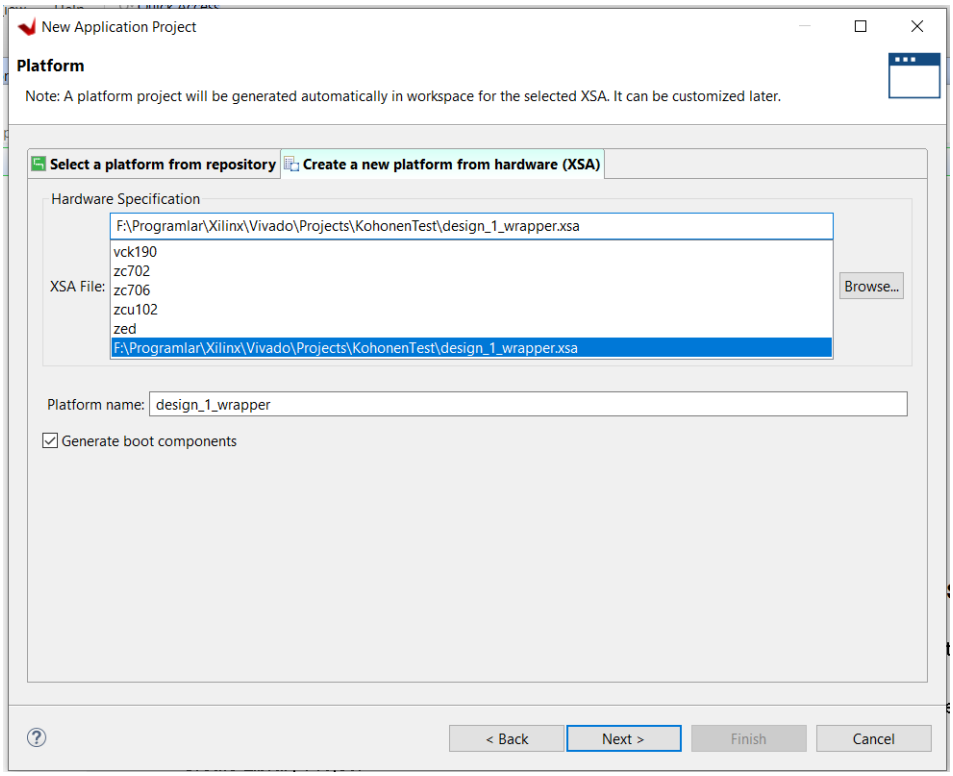


Şekil 7.22 : Yapay sinir ağı vivado simulasyon sonucu

Bütün bu adımların sonucunda eğitilen Kohonen ağının testinin FPGA üzerindeki uygulaması başarılı bir şekilde yapılmış oldu. Oluşturulan IP bütün sistemi içerecek olan bir projede kullanılabilecek düzeydedir.

7.4 Vitis

Vitis programı Microblaze'i kontrol edecek C kodunun yazıldığı ve elf dosyasının bu kodla üretildiği programdır. Vivado'da dışa aktarmayla elde edilen xsa dosyasıyla yeni bir proje oluşturulur.(Şekil 7.23) Proje adı KohonenTest olarak belirlendi.



Şekil 7.23 : Yapay sinir ağı vitis proje oluşturma

Proje oluşturmanın ardından kaynaklara gidilir ve bir kontrolcü kodu yazılır. Kod yazmadan önce Üretilen IP'nin yazma ve okuma fonksiyonları xdut.h dosyasından bulunur ve bu fonksiyonlar yardımıyla istenen adresteki registerlara giriş verileri ve start sinyalleri verilebilir.

Daha sonra sistem giriş çıkış adreslerini bulmak için "KohonenTest.prj/Hardware Specification" konuma gidilir ve orada bulunan DUT registerlarından giriş çıkış adresleri bulunur. Address/Offset sekmesinde bulunan değerler bulundukları adresleri göstermektedir. Bu değerler okuma yazma fonksiyonlarına girilirken bir base adres ve üzerine eklenen ofset şeklinde girilir. Base adres değişken ismi xparameters.h dosyasının içinden çekilerek fonksiyonlara eklenir. Yanındaki değişkene de kullanılan giriş veya kontrol sinyalinin ofset değeri girilir.

```c
#include <stdio.h>
#include "xparameters.h"
#include "xdut.h"


int main()
{
    int ratio1, ratio2, result;
    const int data[] = {0x40000000,0,0x40000000,0};

    XDut_WriteReg(XPAR_XDUT_0_S_AXI_AXILITES_BASEADDR, 24, 1); // enable
    XDut_WriteReg(XPAR_XDUT_0_S_AXI_AXILITES_BASEADDR, 32, 0); // reset

    for(int i=0;i<4;i++){
        XDut_WriteReg(XPAR_XDUT_0_S_AXI_AXILITES_BASEADDR, 16, data[i]); // input
        XDut_WriteReg(XPAR_XDUT_0_S_AXI_AXILITES_BASEADDR, 0, 1); // start
    }

    ratio1 = XDut_ReadReg(XPAR_XDUT_0_S_AXI_AXILITES_BASEADDR, 40);
    ratio2 = XDut_ReadReg(XPAR_XDUT_0_S_AXI_AXILITES_BASEADDR, 44);
    result = XDut_ReadReg(XPAR_XDUT_0_S_AXI_AXILITES_BASEADDR, 52);

    XDut_WriteReg(XPAR_XDUT_0_S_AXI_AXILITES_BASEADDR, 24, 0); // enable

    return result;
}

```

Şekil 7.24 : Yapay sinir ağı vitis kodu

Microblaze'i kontrol edecek olan test kodu yazıldıktan sonra "Project/Build Project" yoluna basılarak elf dosyasının oluşturulması sağlanır. Elde edilen elf dosyası vivadonun testinde kullanılarak bütün sistemin simülasyonu yapılmış olur.

8. KRİPTO ALGORİTMALARININ GERÇEKLENMESİ

Bu kısımda kullanılan kripto algoritmalarının donanımda gerçeklenmesi Model Composer, Vivado ve Vitis kullanımıyla birlikte anlatılacaktır.

8.1 Gelişmiş şifreleme standardı(AES)

Bu projede kripto algoritması olarak AES ve Diffie Helman anahtar değişim protokolü kullanılmıştır. Bu kısımda AES şifreleme ve şifre çözme tasarımlarını sırasıyla Model Composer, Vivado ve Vitis aracının kullanımı anlatılacaktır. AES algoritması bu projede anahtar uzunluğu 128 bittir. Bu nedenle gerçeklenecek tur sayısı 10'dur.

8.1.1 Model Composer

Model Composer ile tasarlanan Aes, şifreleme ve şifre çözme olarak iki bölümden oluşmaktadır. Sırayla anlatılmıştır.

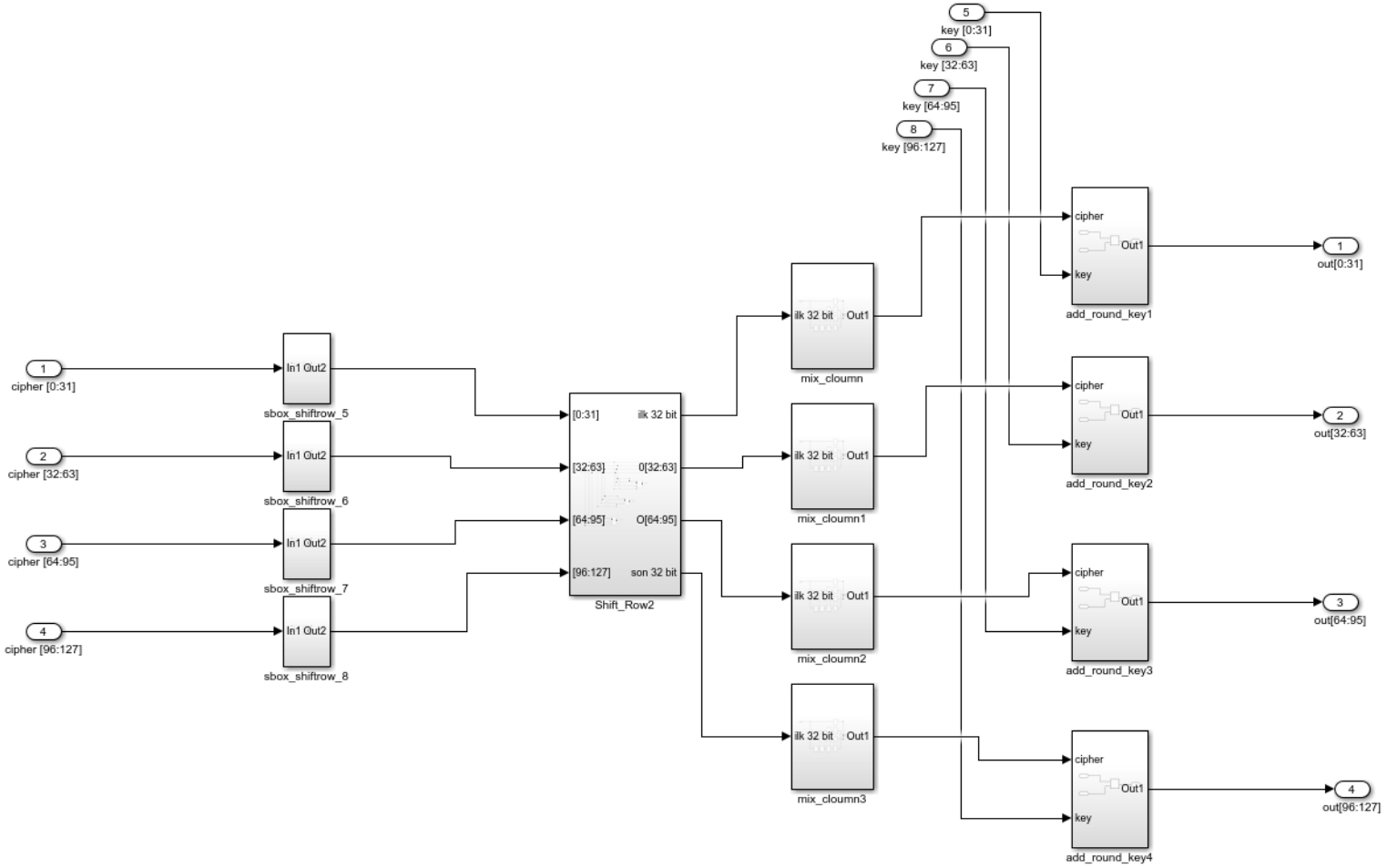
8.1.1.1 128-Bit Gelişmiş Şifreleme Standardı şifrelemesinin blokları

AES 128 bit şifreleme algortiması için 3 farklı tasarım yapılmıştır. Bunlar kombinezonsal tasarım, ardışıl tasarım-1 ve ardışıl tasarım-2 olarak alt başlıklar halinde verilmiştir.

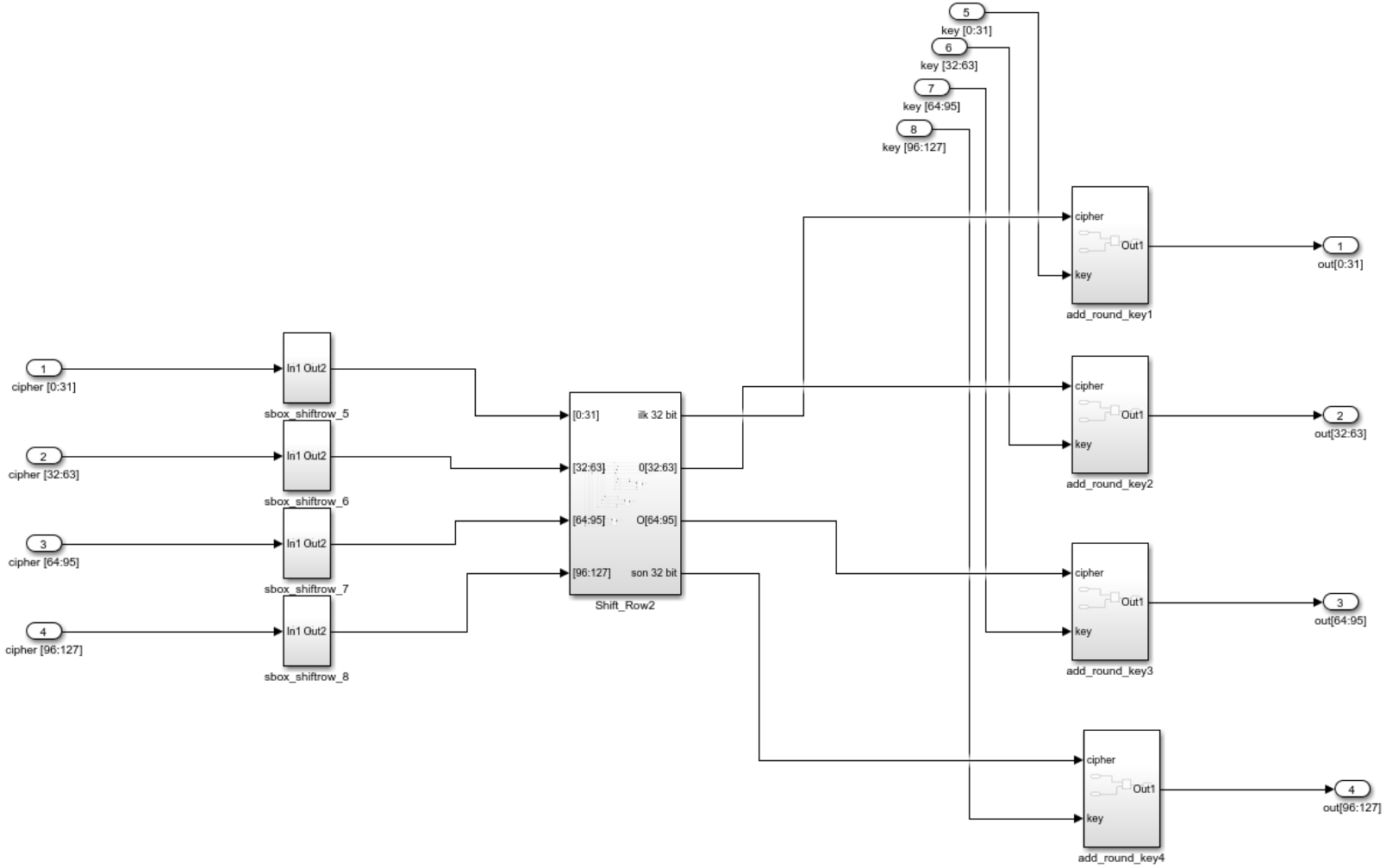
Kombinezonsal Tasarım

Model Composer daha çok yeni bir araç olduğundan bu konuda örnek bulmakta zorlanılmıştır. Model Composer piyasaya sürülmeden önce kullanılmakta olan System Generator[79] ile tasarlanan sistemlerde nasıl bir yaklaşım kullanıldığı gözlemlenmiştir[80] [81].

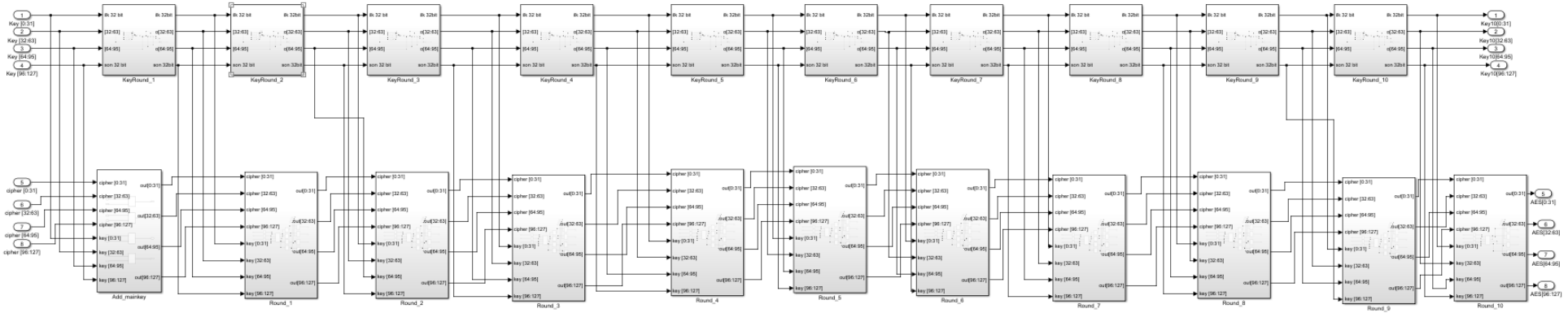
128 bit anahtar kullanılarak Model Composer ile tasarlanan AES şifreleme algoritmasın bir tur bloğunun içeriği Şekil 8.1 ile gösterilmiştir. AES algoritması tasarlanırken ilk önce tam kombinezonsal devresi tasarlanmıştır. Bu tasarım içerisinde kombinezonsal olarak tasarlanan, anahtar genişletme blokları ve şifreleme blokları yer almaktadır. Son blok hariç bütün şifreleme blokları Şekil 8.1'deki gibidir. Son şifreleme bloğunda sütun karıştırma işlemi yoktur. Şekil 8.2 ile gösterilmiştir**.** Şekil 8.3'de ise kombinezonsal AES tasarımında yer alan 10 şifreleme bloğu, 10 anahtar genişletme bloğu ve bir ana anahtar ile şifrelenecek metinin xor işleminin yapıldığı blok bulunmaktadır.



Şekil 8.1 : AES algoritmasının bir tur modülünün tasarımı



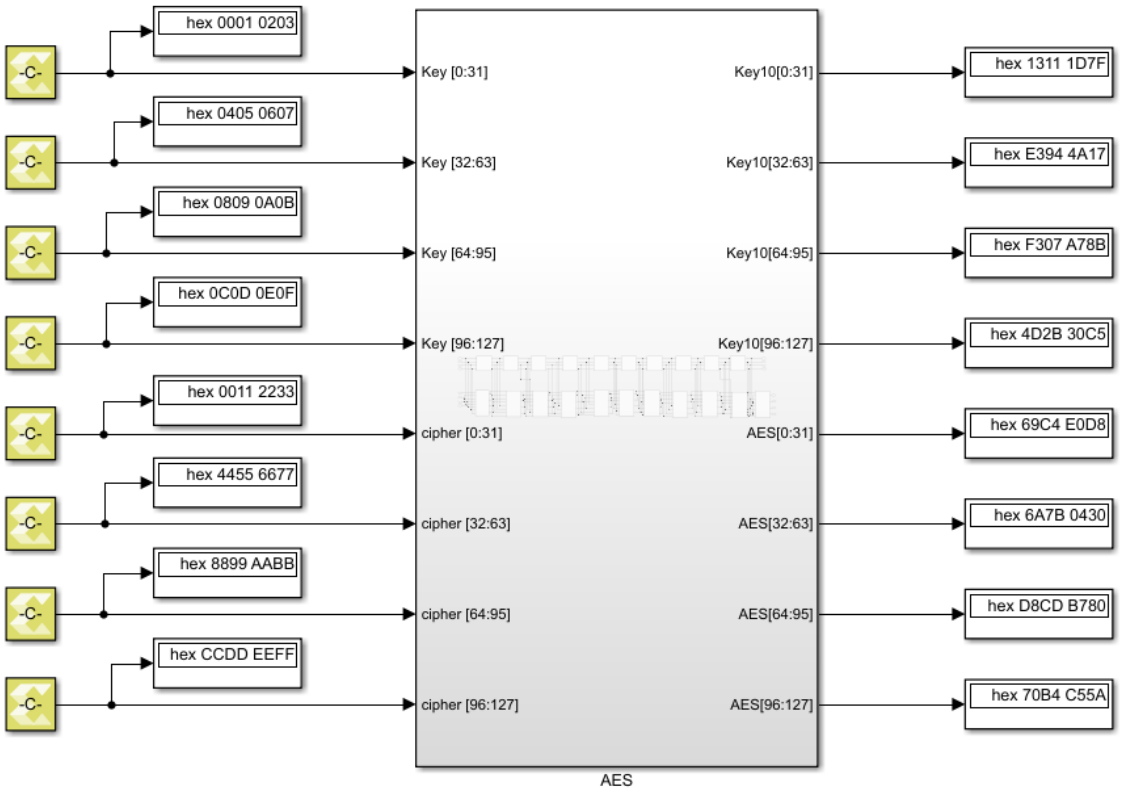
Şekil 8.2 : AES algoritmasının 10.tur modülünün tasarımı



Şekil 8.3 : Kombinezonsal AES şifreleme modülü tasarımı

Şekil 8.4 kombinezonsal olarak tasarlanan AES şifreleme bloğuna verilen sabit girişler ile şifrelenmiş metin ve on tur sonucunda oluşan anahtar çıktı olarak verilmiştir. Federal Bilgi İşleme Standartları Yayınları(Federal Information Processing Standards Publications, FIPS PUBS) 1996 tarihli Bilgi Teknolojisi Yönetim Reformu Yasası'nın 5131. Bölümü uyarınca Ticaret Bakanı tarafından onaylandıktan sonra Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) tarafından yayınlanır. NIST onaylı AES algoritması FIPS 197 dökümanında anlatılmaktadır.

FIPS 197[20] dökümanından sonucun doğru olduğu kontrol edilmiştir. Sistem doğru bir şekilde çalışmaktadır.

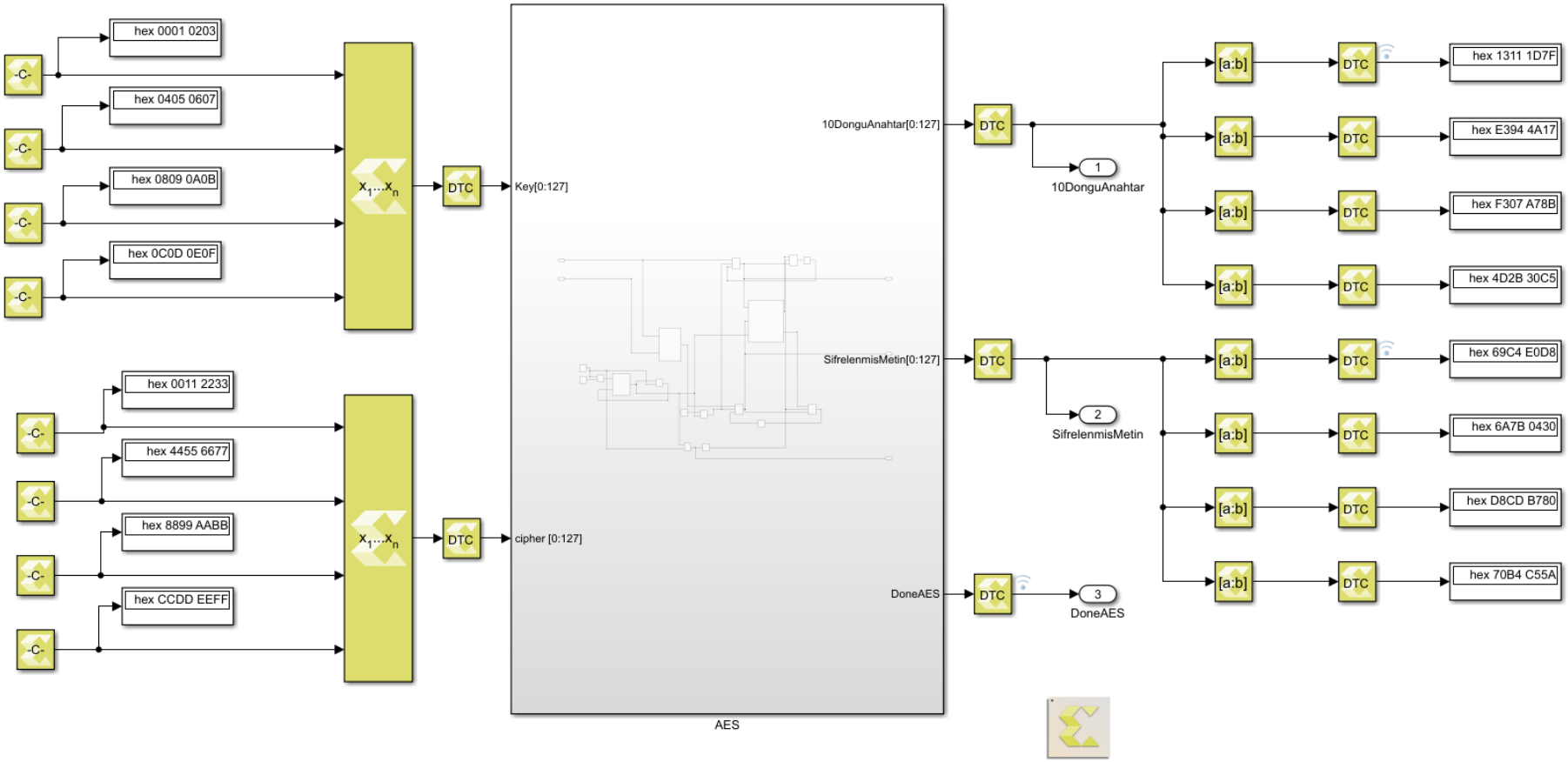


Şekil 8.4 : Kombinezonsal AES şifreleme algoritmasının testi

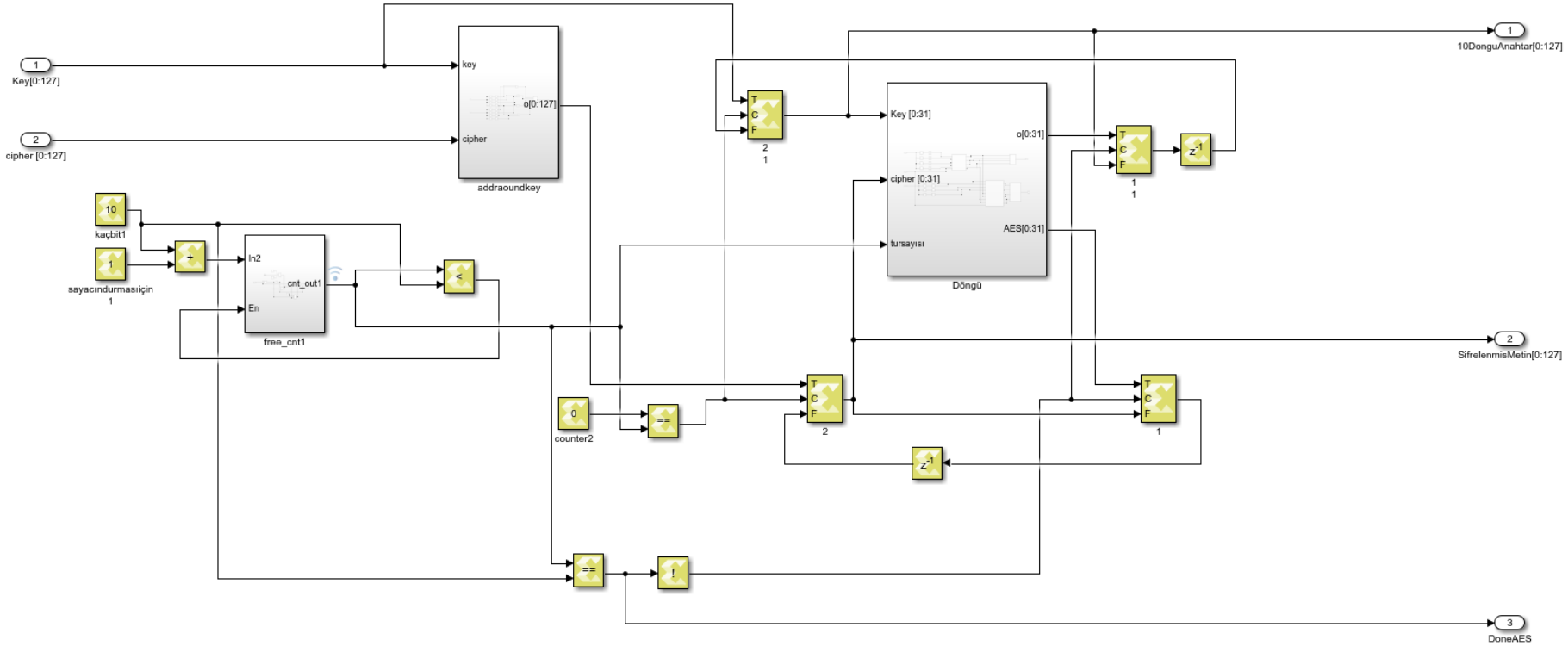
Ardışıl Tasarım-1

Alan ve performans ihtiyaçları doğrultusunda AES şifreleme algoritmasının, ardışıl devre tasarımı da gerçeklenmiştir. Şekil 8.1'deki blok yapısı aynen kullanılmıştır. Tek fark son blokta kullanılmayan sütun karıştırma işlemi için "Conditional" bloğu eklenmiştir. Şekil 8.1 ve Şekil 8.2 işlemleri Conditional bloğu ile koşula bağlanmıştır. Bu sorun üzerine yapılan tasarım Şekil 8.9'da gösterilmiştir**.** AES şifreleme

algoritmasının iki farklı ardışıl devre tasarımı gerçeklenmiştir. İlk tasarlanan AES algoritmasının ardışıl tasarımının içerisinde, anahtar genişletilmesi algoritmasının da ardışıl tasarımı yer almıştır. Oluşturulan döngüde hem o tur için anahtar türetilmiş hem de şifreleme yapılmıştır.

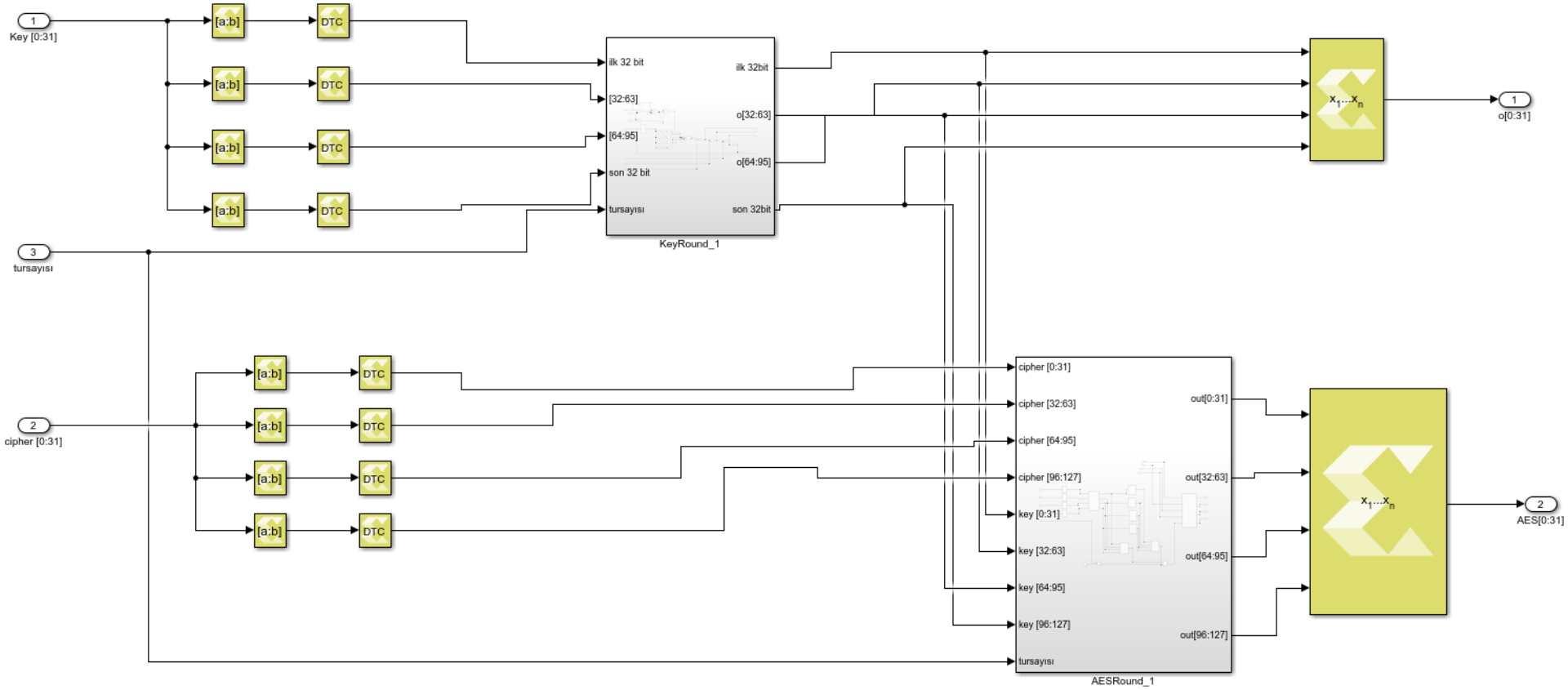


Şekil 8.5 : Ardışıl AES şifreleme modülü tasarım

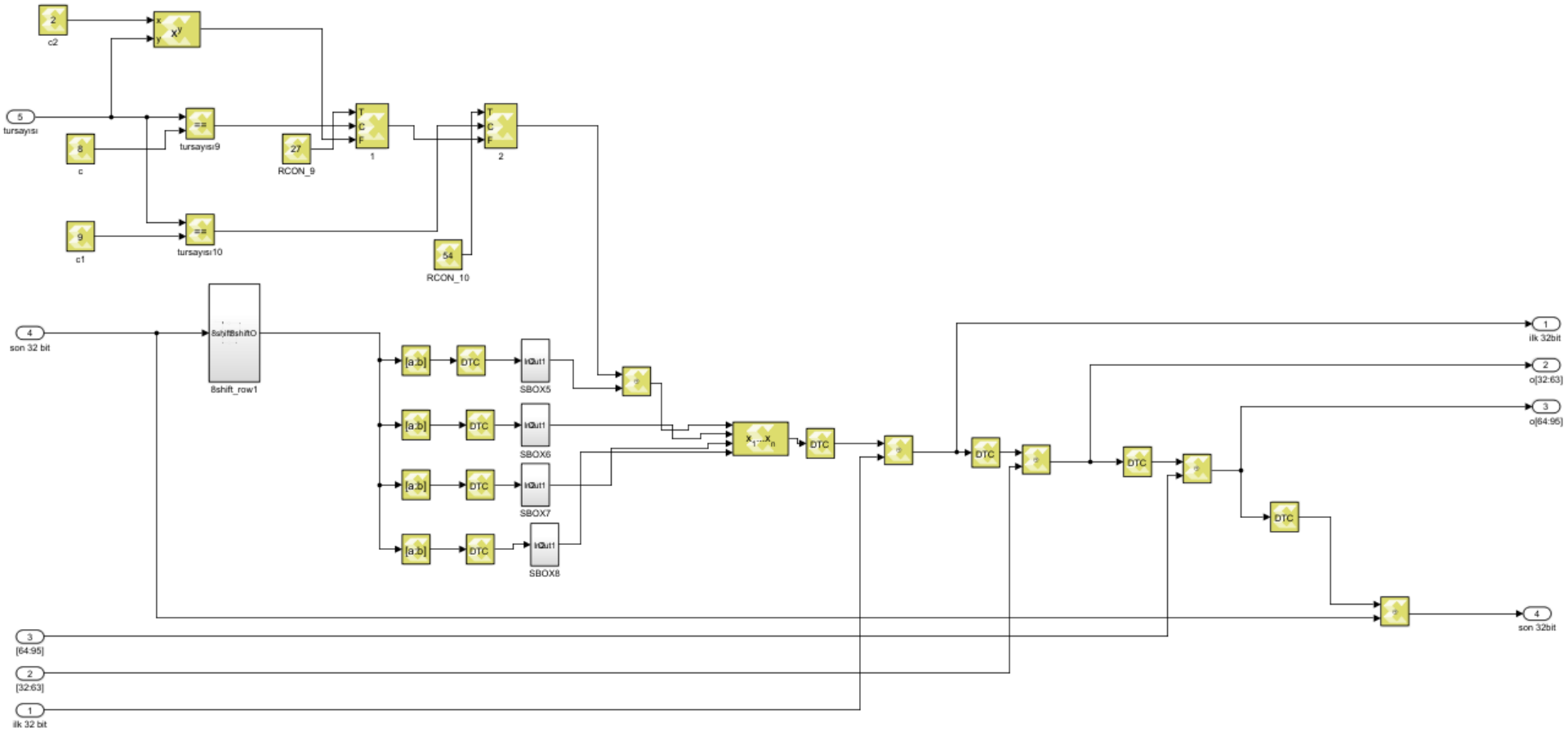


Şekil 8.6 : Ardışıl AES tasarım-1 modülünün iç tasarımı

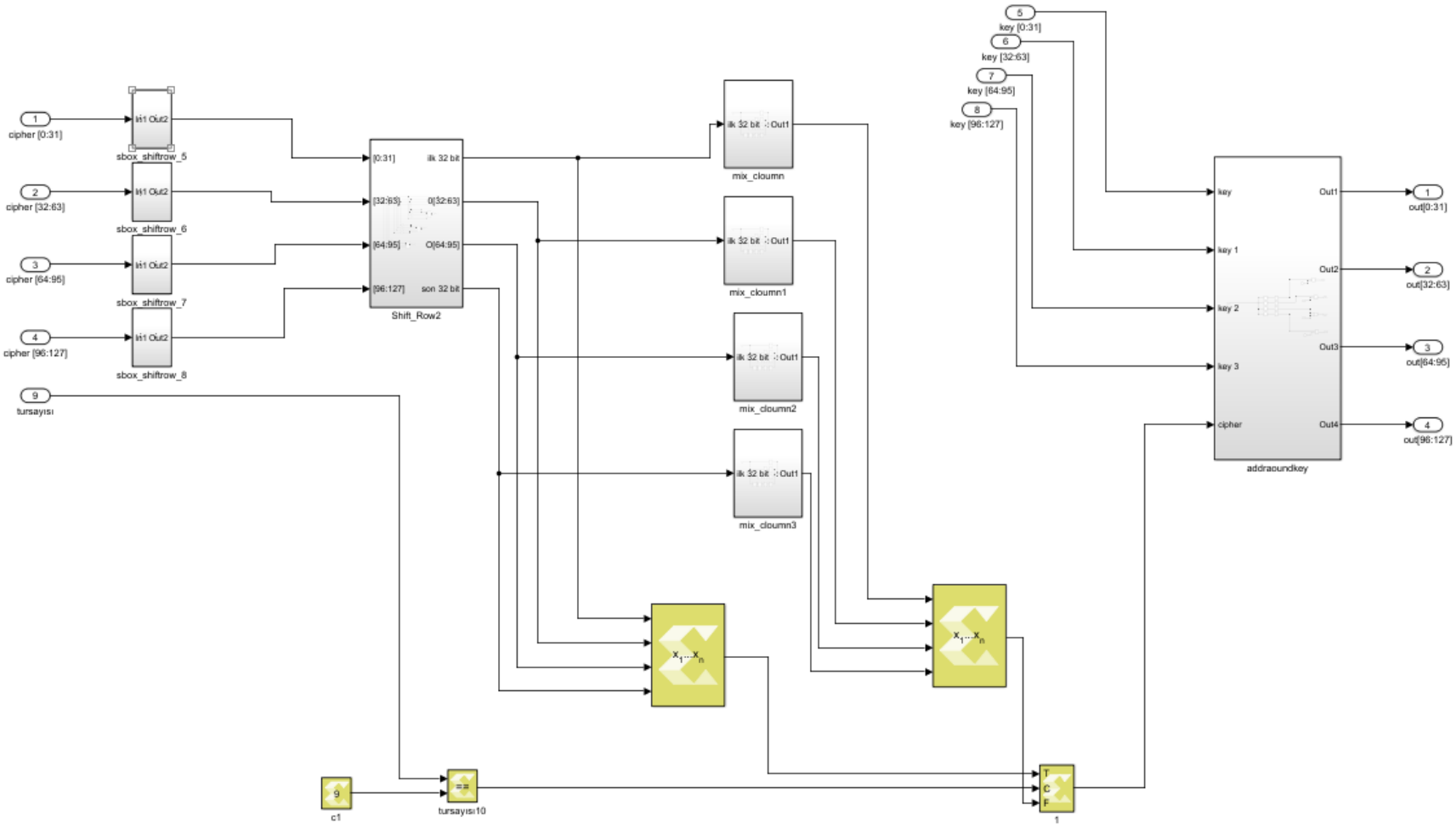
Şekil 8.7 ile “Döngü” bloğunun algoritması gösterilmektedir. “Key_Round1” alt bloğunun ürettiği anahtar, “AES_Round1” alt bloğununun tur anahtarı girişine verilmektedir. Şekil 8.8’de yer alan Key_Round1 bloğu, kombinezonsal anahtar genişletme tasarımında kullanılan bloktan farklı “RCON” tasarımı bulunmaktadır. Her anahtar üretim turunda farklı değere sahip olan RCON girişi için “tursayısı” girişine bağlı Conditional blok içeren bir yapı tasarlanmıştır. Tur sayısı ile RCON değeri arasındaki ilişkiden yararlan amacıyla üs işlemi kullanılmıştır. Ama dokuzuncu ve onuncu tur için RCON değerleri 27 ve 54’tür. Tur sayısı kontrolü kullanarak bu değerler sabit sayı olarak sisteme verilmiştir. Daha sonra kullanılan ardışıl anahtar genişletilmesi algoritmasında bu yapı yerine LUT bloğu kullanılmıştır.



Şekil 8.7 : Ardışıl AES tasarımda kullanılan "Döngü" alt modülünün iç tasarımı



Şekil 8.8 : Döngü bloğunda kullanılan “Key_Round1” alt modülünün iç tasarımı

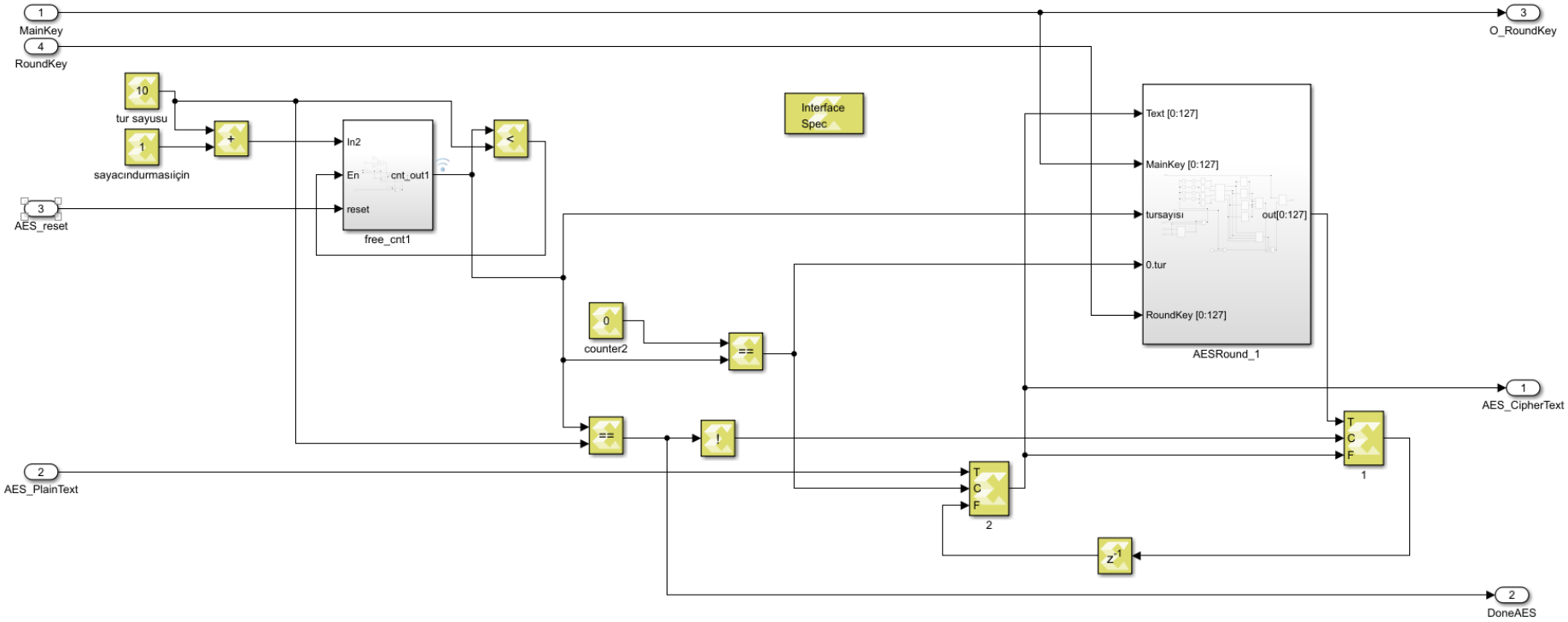


Şekil 8.9 : Döngü bloğunda kulanılan "AES_Round1" alt modülünün iç tasarımı

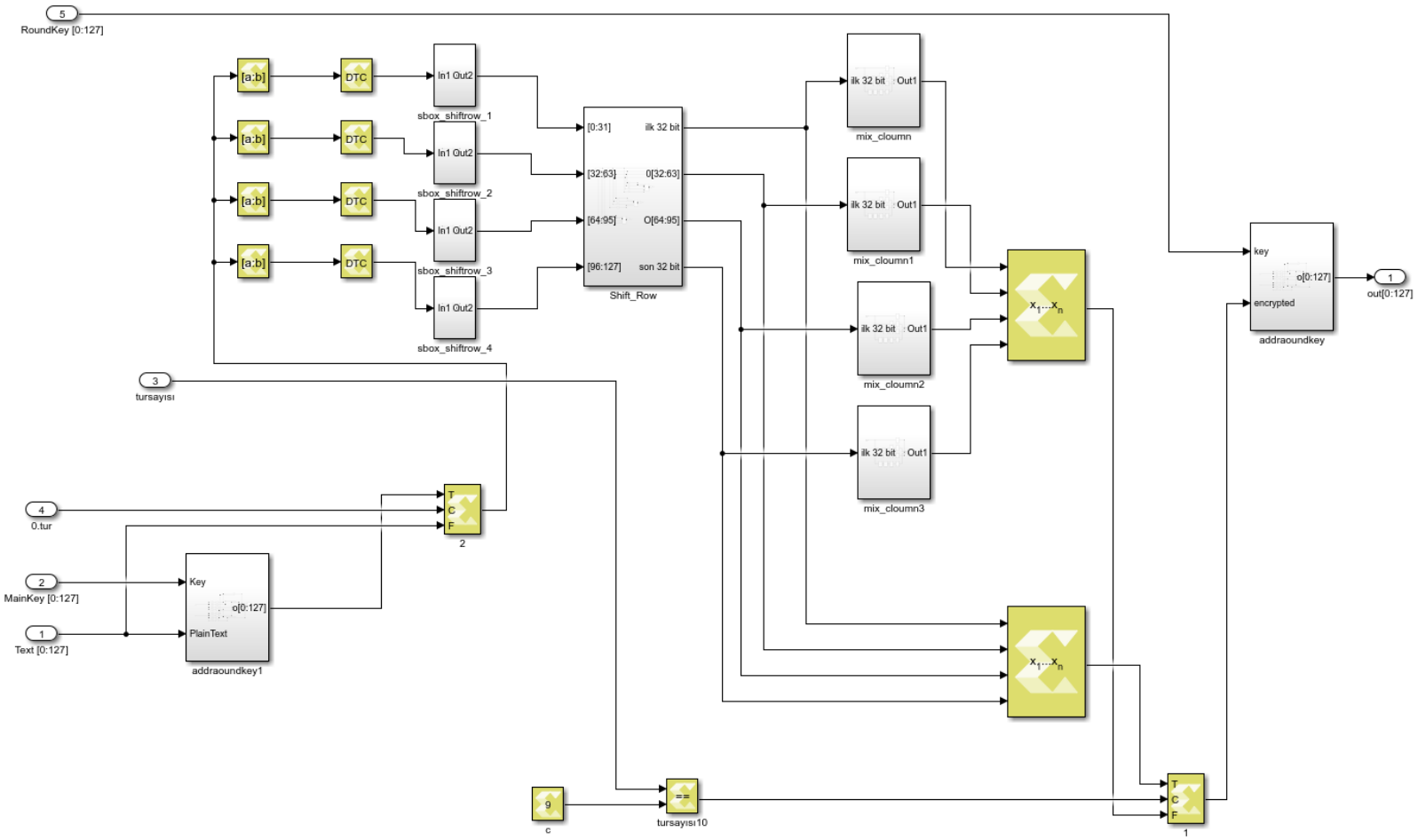
Ardışıl Tasarım-2

Bu projede tasarlanan yapılar hava platformu üzerinde kullanılması planlanmaktadır. Platform üzerinde de AES şifre çözme algoritmasını ihtiyaç duyulabileceği düşünülerek şifre çözme algoritmasının da ardışıl devre tasarımı yapılması gerekmektedir. AES şifre çözme algoritması, üretilen tur anahtarlarını farklı bir sıralama ile kullandığı için şifreleme algoritması için tasarlanan Ardışıl Tasarım-1 yapısı formatında şifre çözme algoritması tasarlanamadı. Bu nedenle AES şifreleme, AES şifre çözme ve anahtar genişletilmesi algoritmaları ayrı ayrı üç farklı blok olarak tasarlanmasına karar verildi. Bu doğrultuda AES şifreleme algoritmasında bazı değişimler yapıldı.

Şekil 8.11'de gözüktüğü gibi tasarlanan iki ardışıl devrede de kullanılan AES_Round1 alt bloğunun iç tasarımı değişmemiştir. Ayrı blok olarak tasarlanan anahtar genişletme bloğu ilerleyen kısımda anlatılmıştır. AES_Round1 bloğunda kullanılan alt modüller bütün şifreleme algoritmaları için aynıdır.

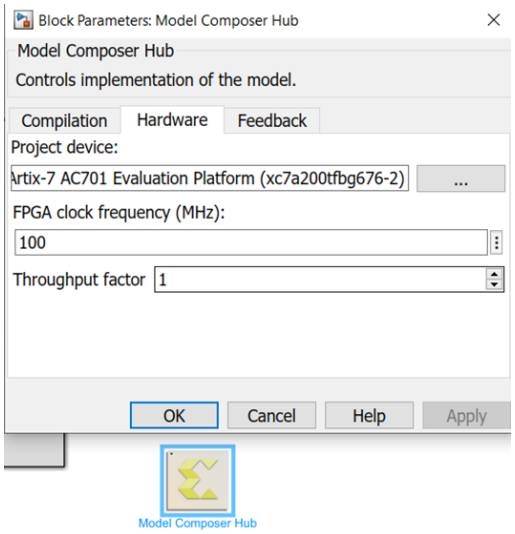


Şekil 8.10 : Ardışıl AES şifreleme tasarımı-2 modülü



Şekil 8.11 : “AES_Round1” alt modülünün iç tasarımı

Model Composer ile tasarlanan sistem, doğru sonuçlara ulaştığında “Model Composer Hub” bloğu ile HDL koda dönüştürme işlemi gerçekleştirilmiştir. IP olarak üretilirken kullanılacak olan fpga belirtilmelidir. Burada hedef olarak seçilen fpga ile Xilinx Vivado’da kullanılan fpga yanı olmalıdır. Tasarlanan kriptoloji müdüllerinde “xc7a200tfbg676-2” fpga kullanılmıştır.

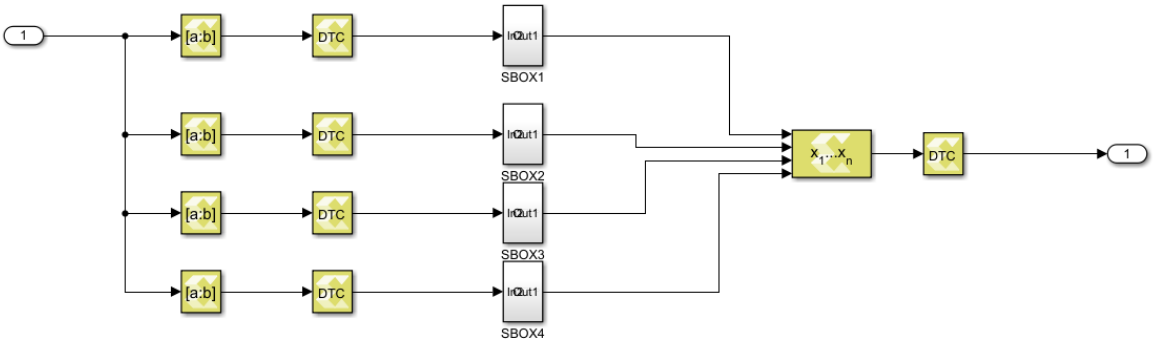


Şekil 8.12 : “Model Composer Hub” bloğu kullanımı

8.1.1.2 Şifreleme Alt Modülleri Tasarımı

Bayt değiştirme

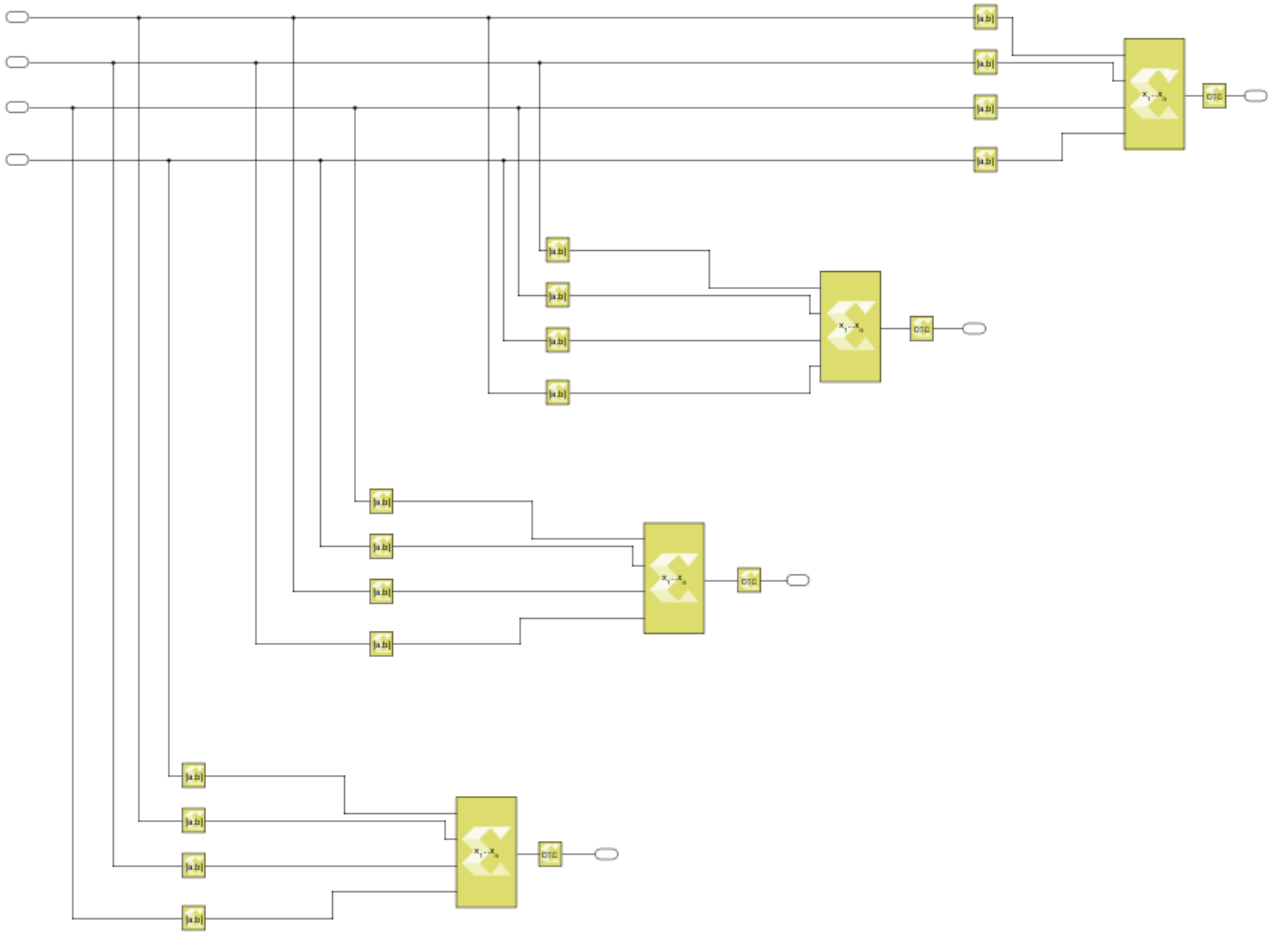
AES şifreleme algoritması Model Composer ile tasarımı gerçeklenirken durum matrisi yapısından uzaklaşılmıştır. Model Composerda işlem kolaylığı açısından 128 bit şifrelenmesi istenen metni “Bit Slice” bloğu yardımıyla 32 bit uzuluğunda dört girişe ayrılmış ve işlemler buna uygun olarak gerçekleştirilmiştir. Model Composer ile S-Kutusu matrisinin tasarlanması aşamasında, S-Kutusunun aslında matrise yerleştirilmiş 256 boyutunda bir vektör olma özelliği kullanılmıştır. Matris oluştururken bu vektör satır satır dolduracak şekilde yerleştirilmiştir. Xilinx Model Composer kütüphanesinde yer alan LUT bloğu kullanılmıştır. LUT bloğunun çalışma prensibi verilen giriş değerine göre içine yazılan vectörün ilgili sıradaki değerini çıktı olarak vermektir. LUT’un içine; S-Kutusunun onaltılık sayı sisteminde kullanılan değerlerini, ondalık sayı değerlerine dönüştürülerek yazılmıştır. Durum matrisinden gelen 8 bit çıktı ondalık sayıya çevrilerek LUT’un girişine bağlanmıştır. Durum matrisinden gelmesi gereken bitler Bit Slice blokları yardımı ile ayrılmıştır.



Şekil 8.13 : Bayt değiştirme işlemi tasarımı

Satır kaydırma

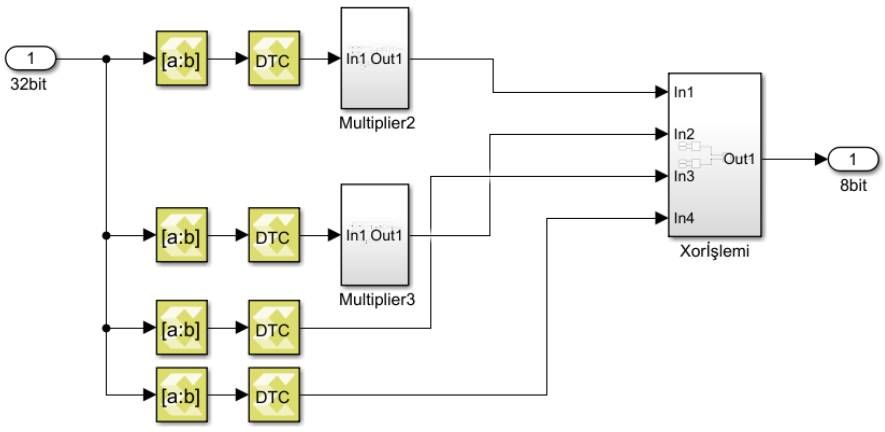
Satır kaydırma işlemi için Model Composer ile tasarlanırken Xilinx kütüphanesinde yer alan "Shift Left" bloğu kullanılamadı. Çünkü bu blok sola kaydırma yaptığında en anlamsız bit yerine otomatik olarak "0" biti eklemektedir. Satır kaydırma işlemini yaparken döngüsel sola kaydırma işlemi yapılmaktadır. Bayt değiştirme işleminden çıkan 32 bit uzunluğundaki bitler durum matrisin satırlarıdır. Satır kaydırma işlemine giren bu bitlerin ilk 32 biti [0:31] durum matrisinin ilk satırını, [32:63] bitler ikinci satırı, [64:95] üçüncü satırı ve [96:127] dördüncü satırı temsil etmektedir. Bu nedenle Satır kaydırma algoritması gereği "Bit Concat" bloğunun girişlerinin sıralaması fark edildiğini gibi her bir çıkış için farklıdır. Algoritma gereği ilgili bitlerin "Bit Slice" bloğu ile seçilerip "Bit Concat" bloğu ile bitlerin ard arda eklenmiştir. Bit Slice blokları dörtlü dörtlü kullanılmıştır. Her bir dörtlü aynı şekilde tasarlanmıştır. Yukarıdan aşağı doğru sırayla en değerli 8 biti[0:7],[8:15] ve son olarak en değersiz olan son 8 biti [24:31] ayırmaktadır. Satır kaydırma işleminden çıkan bitler durum matrisinin satırlarını oluşturmaktadır.



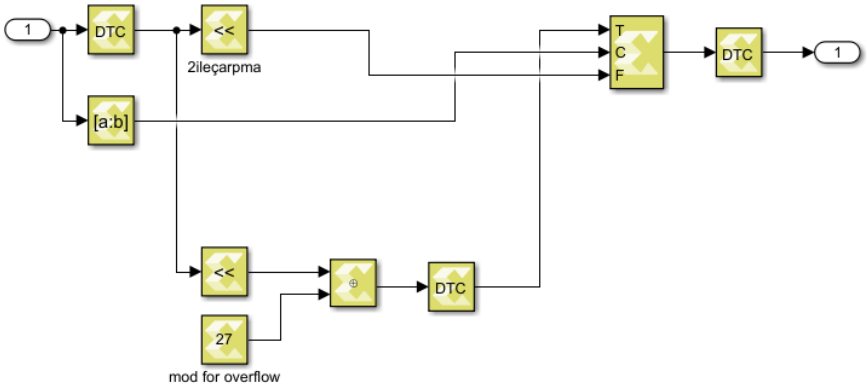
Şekil 8.14 : Satır kaydırma işlemi tasarımı

Sütun karıştırma

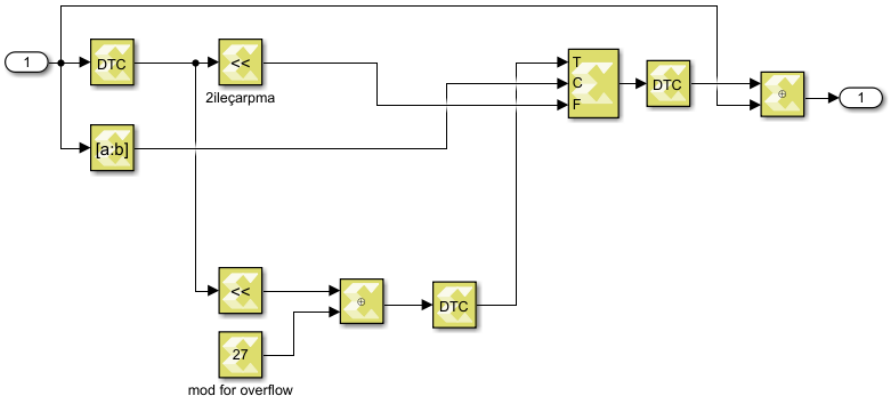
Model Composer ile sütun karıştırma işlemi tasarımı aşamasında, bu adımda kullanılan A matrisi yakından incelendi. A matrisi, birinci satırın bir alt satıra bir sağa kaydırılmış halinin yerleştirildiği bir matristir. Sütun karıştırma işlemi tasarlanırken bu özellikten yola çıkılmıştır. Aynı sütun değeri ile çarpma yapabilmek amacıyla her bir A matrisi satırı için, önce gerekli olan sola kaydırma işlemi yapılmıştır. Böylece durum matrisinin her bir sütünu A matrisinin birinci satırıyla matris çarpmasına girebilcek duruma gelmiştir. A matrisinin ilk satırı ile durum matrisi, Galois Field'de çarpma ve xor toplama işlemine tabi tutulmuşlardır. GF(2^8) için polinom gösterimi " $m(x) = (x8 + x4 + x3 + x + 1)$ " şeklindedir. A matrisinin ilk satır değerler 2,3 ve 1'dir. 2 ile çarpma ve 3 ile çarpma için ayrı ayrı bloklar tasarlanmıştır. Galois Field'de 2 ile çarpma işlemi bitleri sola kaydırma anlamına gelmektedir. Ama yedinci bit değeri "1" olma durumu kontrol edilmelidir. GF(2^8) ile çarpma yapılırken yedinci bit "x8" ile ifade edilmektedir. Elde edilen sonuca, yedinci bit değeri için $x8$ değeri (x8 = $x4 + x3 + x + 1$) eklenmektedir. Model Composer ile tasarım yapılırken ilk önce yedinci bitin değeri kontrol edilmektedir. Eğer değeri "1" ise önce sola kaydırma işlemi yapılıp daha sonra "x4+ x3+x+1 = 0001 1011" değerinin ondalık değeri ile xor işlemi yapılmaktadır. Yedinci bitin "0" olma durumunda sadece sola kaydırma işlemi uygulanmaktadır. 3 ile Galois Field çarpma işlemi ise 2 ile çarpma sonucu ile kendisinin xorlanması anlamına gelmektedir. 2 ile çarpma için tasarlanan alt-sistemin sonuna girişle xor işlemine girecek şekilde blok eklenmiştir.



Şekil 8.15 : Sütun karıştırma işlemi tasarımı



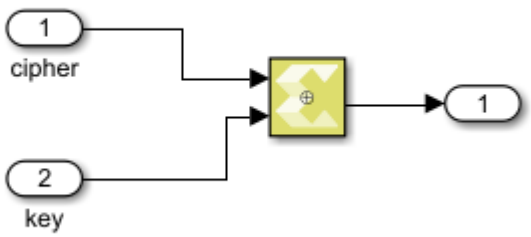
Şekil 8.16 : Multiplier2" alt modülünün iç tasarımı



Şekil 8.17 : "Multiplier3" alt modülünün iç tasarımı

Tur anahtarıyla toplama

Model Composer tasarımda da xor bloğu kullanılmıştır.

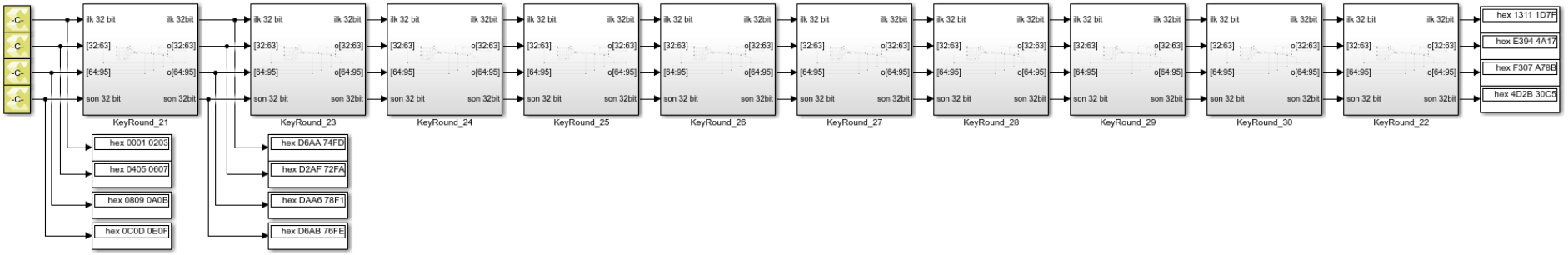


Şekil 8.18 : Tur anahtarıyla toplama işlemi tasarımı

Anahtar genişletilmesi

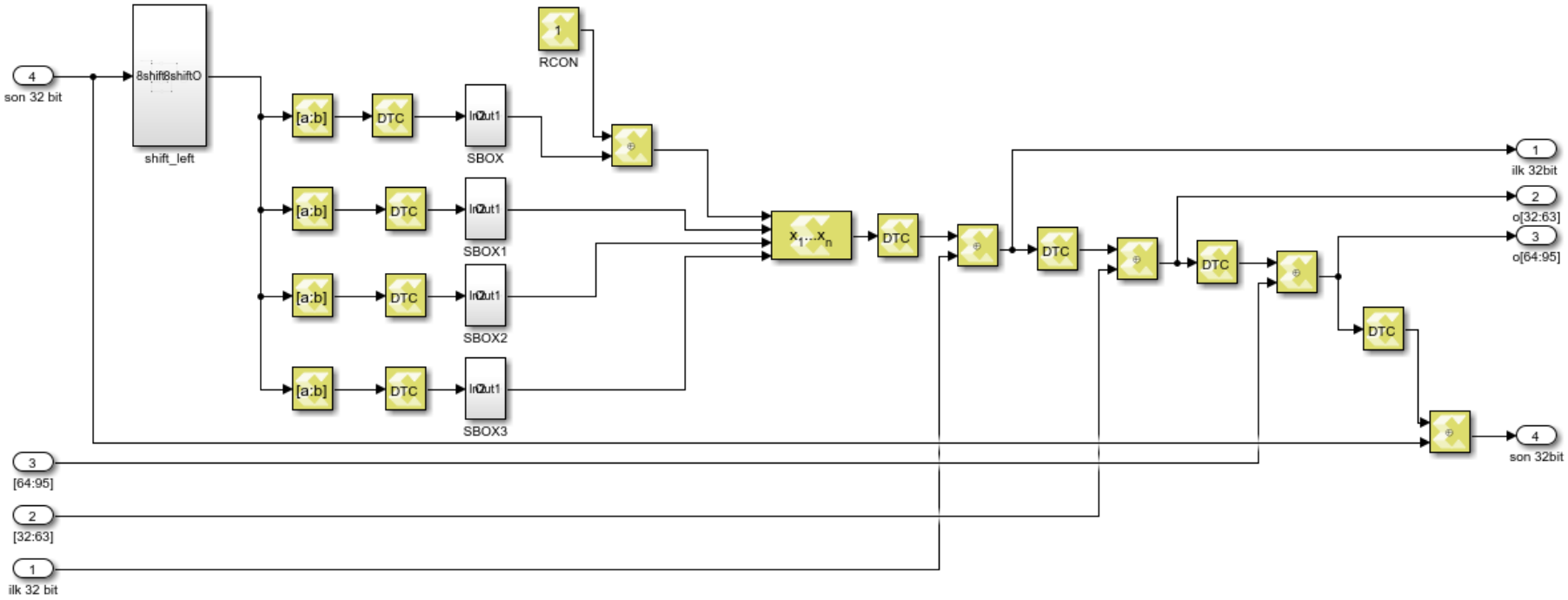
Anahtar genişletilmesi algoritması Model Composer ile tasarlanırken ilk önce tam kombinezonsal tasarımı gerçeklenmiştir. Kombinezonsal devrelerde çıkış sadece o anki girişe bağlıdır. Şekil 8.19'da görüldüğü gibi tasarlanan anahtar genişletilmesi alt bloğu 10 kere kullanılmıştır. Her bir bloğun çıkışı AES şifreleme algoritmasının ilgili turuna, tur anaharı olarak giriş yapmaktadır. Verilen şekilde "Display" ve "Constant" bloğu kullanılmıştır. Sabit sayı olarak verilen değerler için FIPS 197[20] dokümantasyonunda örnek olarak verilen vektör kullanılmıştır. İşlemlerin doğruluğu bu şekilde kontrol edilmiştir.

Anahtar genişletilmesi - Kombinezonsal tasarım



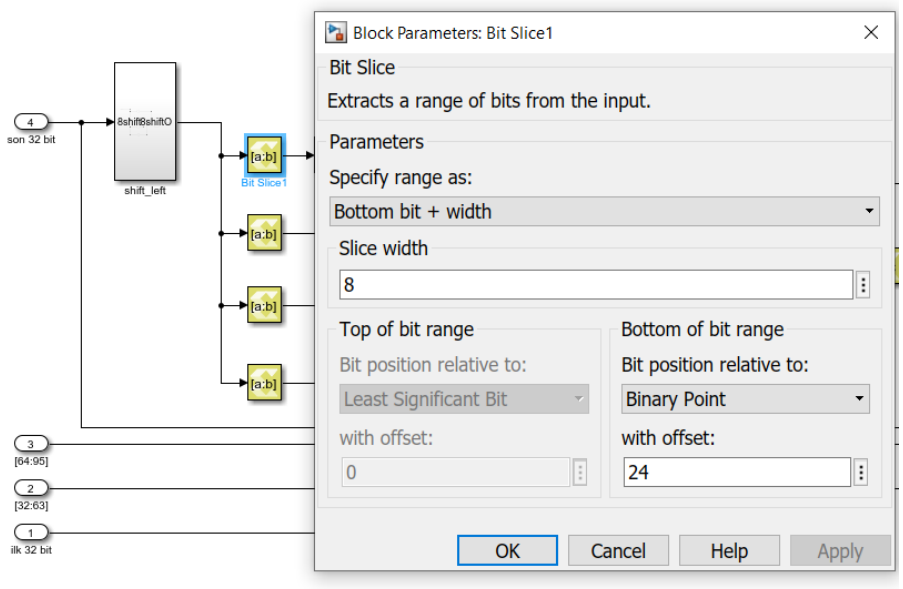
Şekil 8.19 : Kombinezonsal anahtar genişletme modülü tasarımı

Anahtar genişletilmesi algoritması gereğince giriş olarak belirtilen “son 32 bit (Şekil 8.20’de 4 numaralı giriş)” daha önce anahtar genişletme algoritmasında anlatılan $w(3)$ değerine eşittir. 4 numaralı giriş ilk önce “shift_left” bloğuna girer. Bu işlem ile 32 bit, 1 bayt sola kaydırılmaktadır. “Bit Slice” bloklarıyla en değerli biti “SBox” bloğuna girecek şekilde bayt bayt ayrılmaktadır. Bitlerin bayt bayt ayrılma nedeni, S Kutusuna giriş olarak 8 bit verilmesinden kaynaklanır. Algoritmanın bu aşamasında 4 numaralı girişteki en değerli bit ile başlayan 8 bit, RCON değeri ile xor işlemine girer. Kombinezonsal bir devre tasarlandığından RCON değeri her bir turda değişmektedir ve her bir turda sabit sayı olarak verilir. İşlemlerden çıkan bitler birleştirilerek tekrar 32 bit elde edilir. Algoritma gereği ilgili girişler, xor işlemlerinin girişlerine verilir. Böylece bir turun anahtarı, önceki tur anahtarından (birinci tur anahtarı için bu anahtar ana anahtar) yararlanarak üretilmiştir.



Şekil 8.20 : Kombinezonsal "KeyRound" alt modülünün iç tasarımı

Kullanılan “Bit Slice1” bloğuna girilen değerler ile giriş olarak verilen 32 bitin en değerli biti ile başlayan ilk 8 bitinin ayrılması sağlanır. Verilen parametreler Şekil 8.21 ile gösterilmiştir.

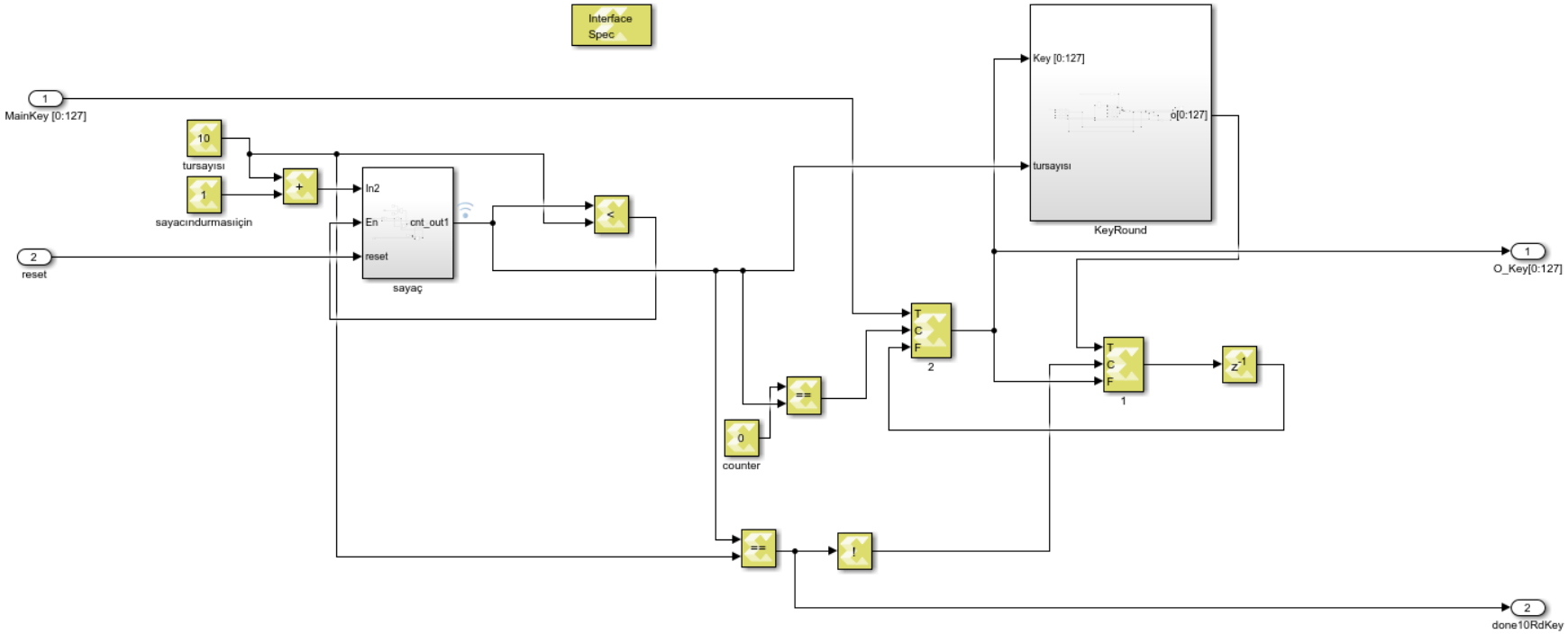


Şekil 8.21 : “Bit Slice1” bloğu parametreleri kullanımı

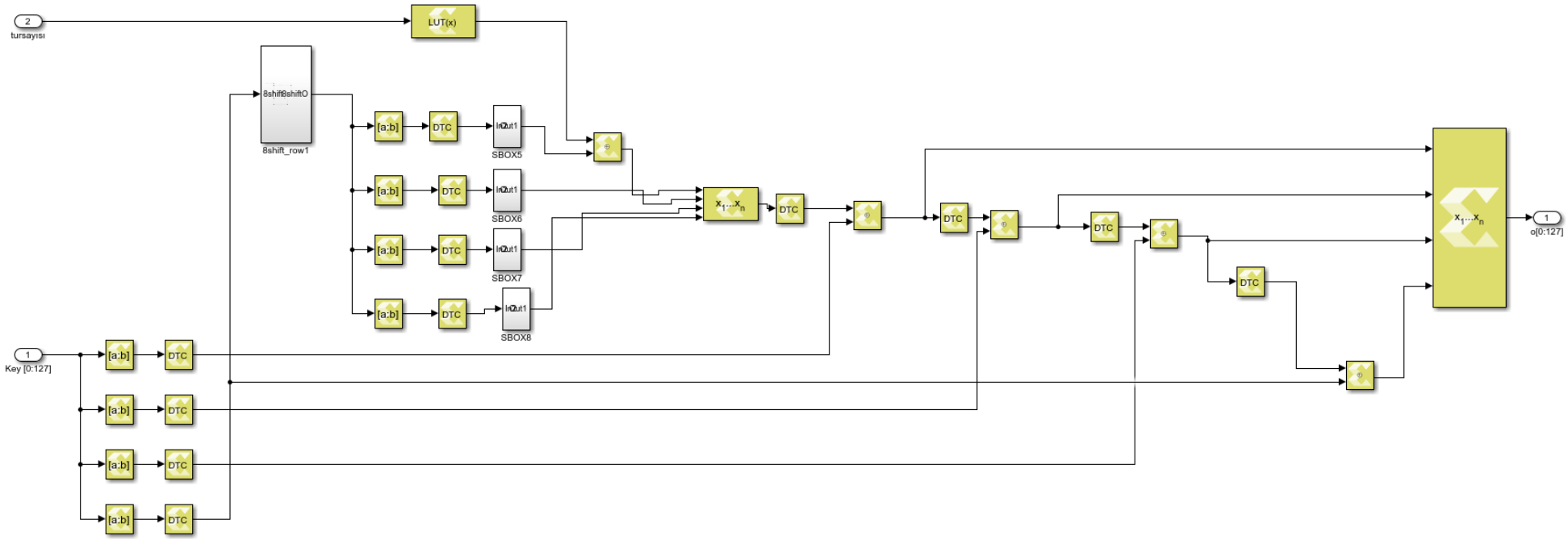
Anahtar genişletilmesi - Ardışıl Tasarım

Anahtar genişletilmesi algoritmasının kombinezonsal tasarımının doğru şekilde çalıştığı Model Composer ile test edilmiştir. Daha sonra HDL koda çevrilerek simülasyon sonucu tasarımın Vivado ortamında da doğru çalıştığı gözlemlenmiştir. Bu durum üzerine alan iyileştirmesi yapabilmek amacıyla anahtar genişletilmesi algoritmasının ardışıl tasarımı gerçeklenmiştir.

Ardışıl devre tasarlanırken bir tur anahtarının üretildiği blokta, sadece RCON değerleriyle yapılan işlem tasarımı değiştirilmiştir. Kombinezonsal tasarımda her turda değişen RCON değerleri sabit sayı olarak verilmiştir. Ardışıl tasarımda ise “LUT” bloğu kullanılmıştır. Her tur için çıkış değeri vektör olarak verilmiştir. Şekil 8.23’de görülen “KeyRound” alt bloğu bir tur anahtarının üretildiği alt sistemdir. Bu bloğa giriş olarak tur sayısını belirten sayacın çıkışı verilmiştir. “Tur sayısı” girişi LUT’un girişidir. Böylece tur için doğru RCON değeri çıkışı verilmektedir.

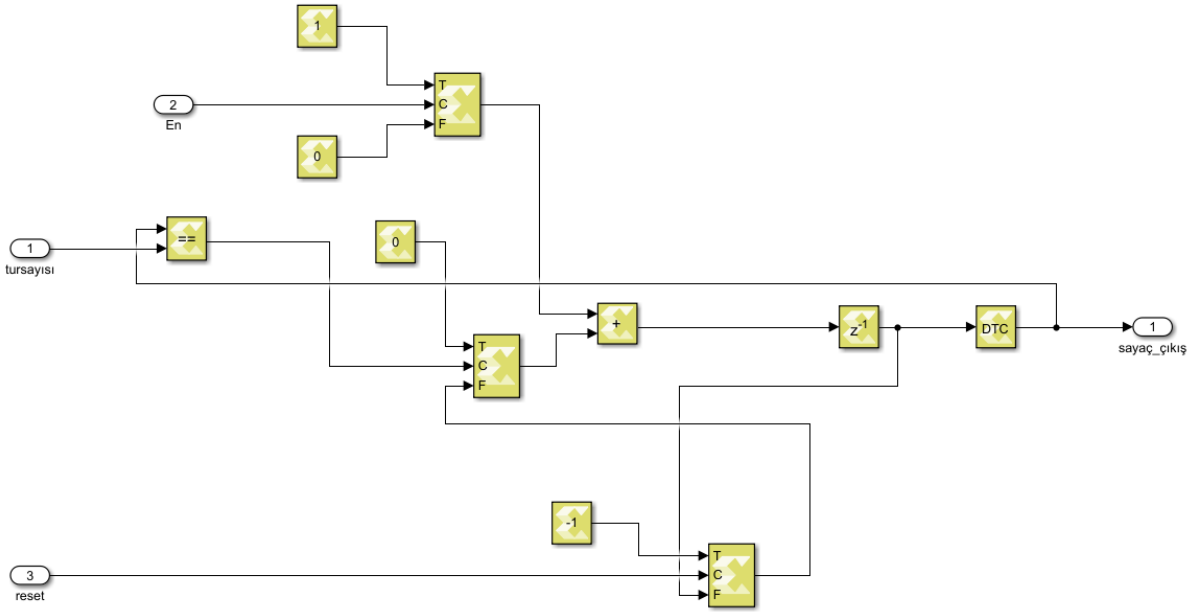


Şekil 8.22 : Ardışıl anahtar genişletme modülü tasarımı



Şekil 8.23 : Ardışıl "KeyRound" alt modülünün iç tasarımı

Daha önce anlatılan modüllerde de kullanılan sayaç modülünde bazı değişiklikler yapılmıştır. Sayaç modülüne "reset" girişi eklenmiştir. Reset girişi bir olduğu durumlarda sayaç değeri sıfırlanayacaktır. Tur anahtarını üretirken kullanılan sayaç değeri, her anahtar değişim algoritması kullanılmadan önce resetlenmelidir. Sayaç, modüle verilen sabit "tursayısı" girişi değerine geldiğinde değişmemektedir. O nedenle reset girişine ihtiyaç duyulmuştur. Aynı işlevi sisteme anahtar değişim algoritmasından önce reset sinyali verilmesiyle de gerçeklenir.



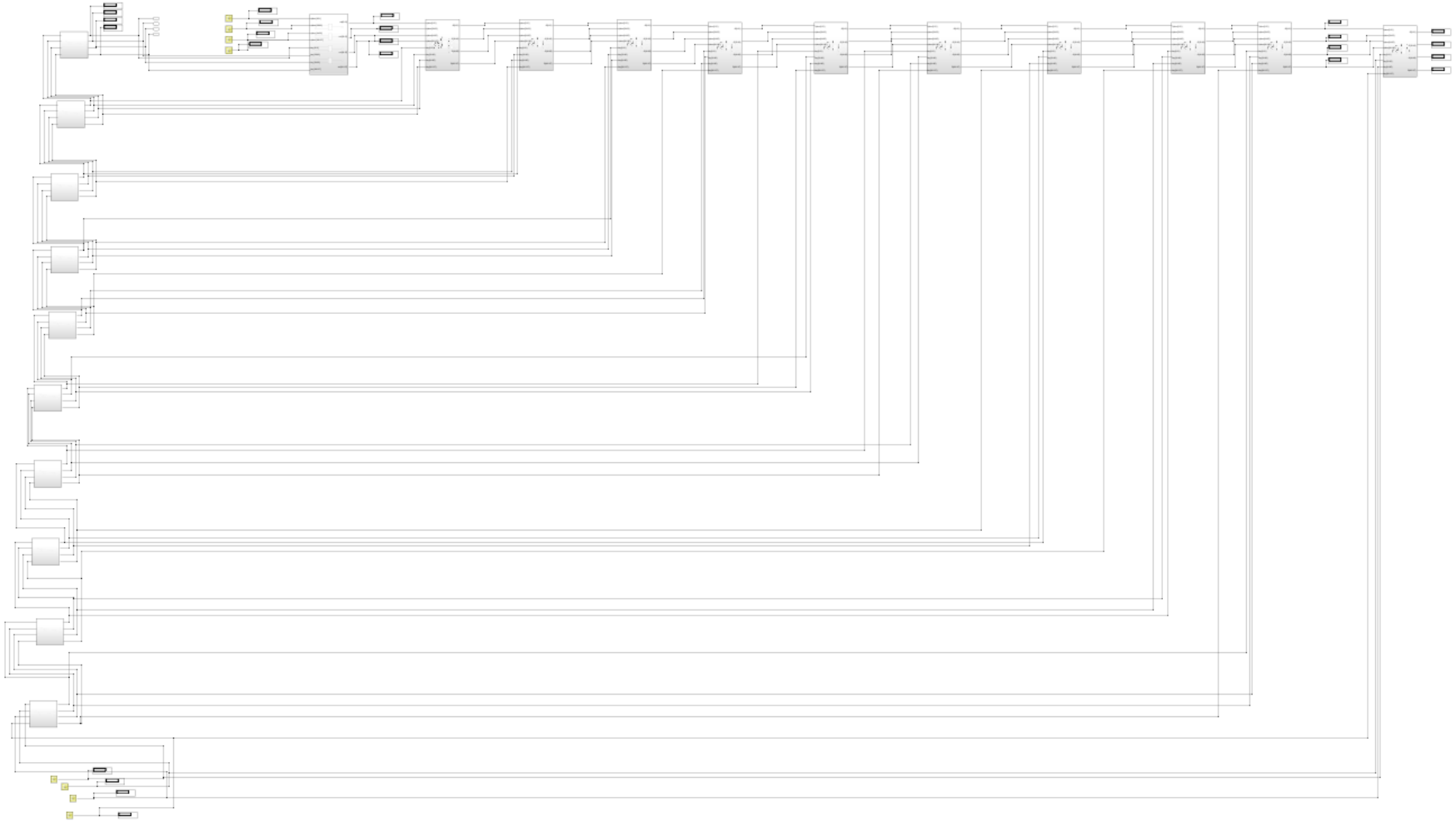
Şekil 8.24 : "Sayaç" alt modülünün iç tasarımı

8.1.1.3 128-Bit Gelişmiş Şifreleme Standardı şifre çözme blokları

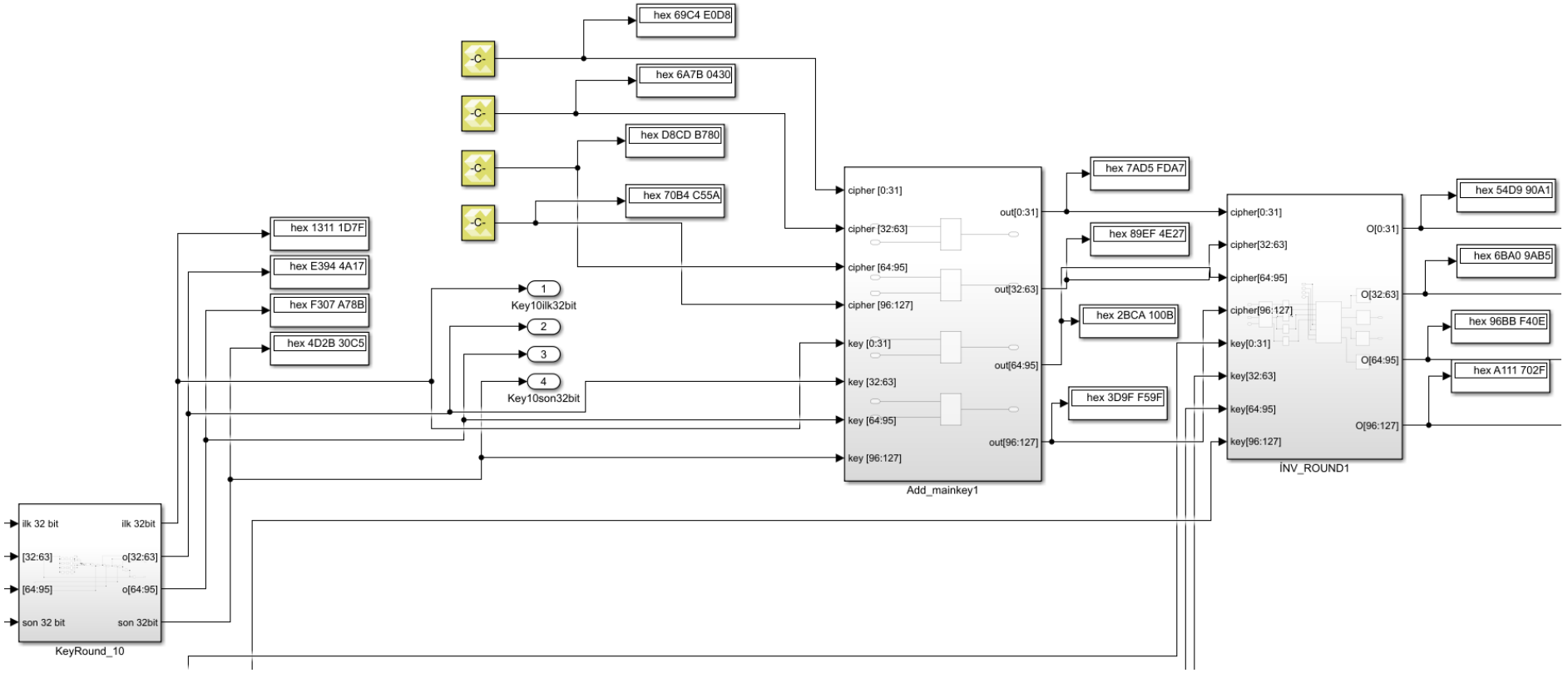
Kombinezonsal Tasarım

Şekil 5.86'da tamamen kombinezonsal olarak tasarlanan AES algoritmasının şifre çözme bloğu yer almaktadır. Ekran görüntüsünün sol tarafında aşağıdan yukarıya doğru 10 tur için anahtar üretenen anahtar genişletme bloğu yer almaktadır. Şifre çözme işlemi gereği anahtar genişletme işlemine giriş olarak verilen ana anahtar diğer on tur için anahtar üretmektedir. Şifreleme için de aynı ana anahtarla aynı tur anahtarları türetilir. Ama bu sefer üretilen anahtarların kullanım sırası değişmiştir. Bu nedenle Şekil 8.25'da karışık bir görütünsü yer almaktadır. Soldan sağa doğru yer alan bloklarda ise şifre çözme algoritmasının her bir tur işlemi gerçekleşmektedir. Şekil 8.26'da, FIPS 197 dökümanında[20] yer alan örnek AES ile şifrelenmiş vektörü şifre çözme işlemine giriş olarak verilmiştir. Anahtar olarak da yine aynı anahtar verilmiştir.

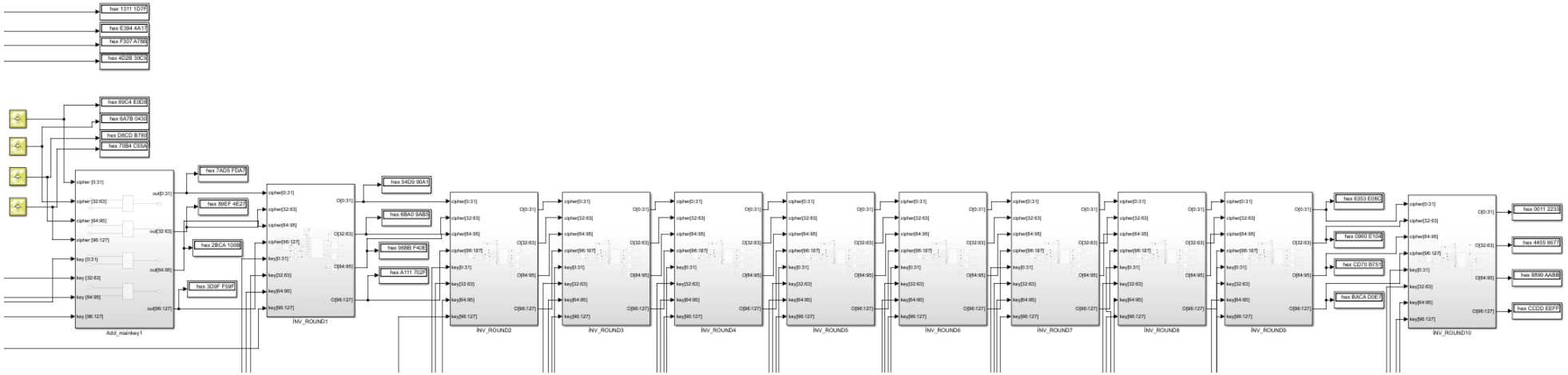
Bu şekilde şifrelenen mesajı, şifre çözme algoritması kullanarak şifrelenmeden önceki haline geri döndürülebildiği kontrol edilmiştir. AES şifreleme algoritmasına şifrelenmek üzere verilen vektör ile şifre çözme algoritmasının çıkışı birebir aynısıdır. Tasarlanan şifreleme ve şifre çözme algoritmalarının doğru çalıştığı doğrulanmıştır. İşlemler gerçekleşirken uygulanan adımlar arasındaki sonuçlar FIPS 197[20] dökümanından kontrol edilebilmektedir.



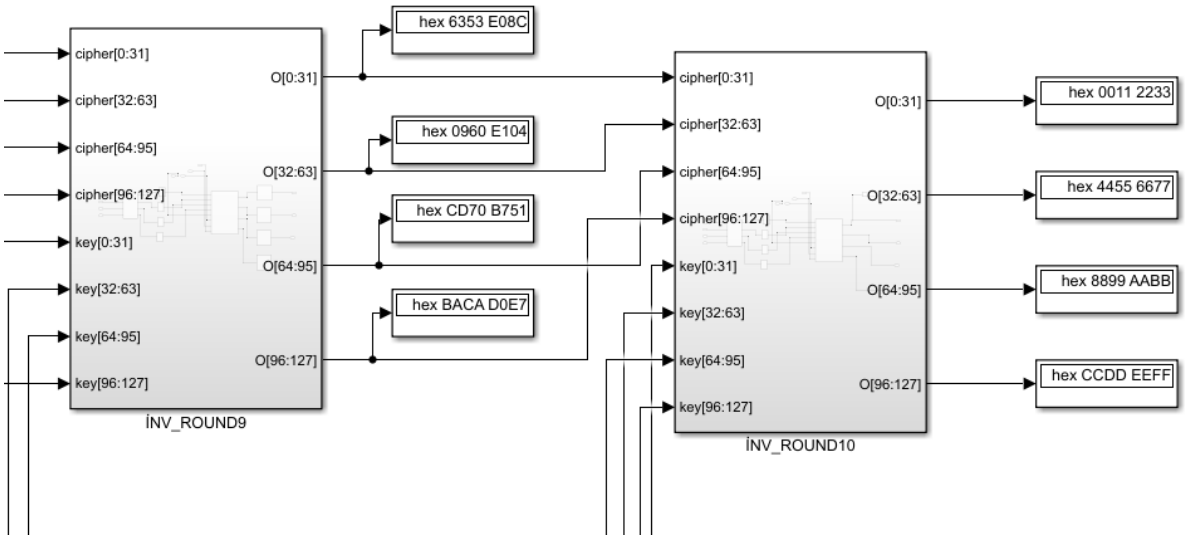
Şekil 8.25 : Kombinezonsal AES şifre çözme modülü tasarımı



Şekil 8.26 : AES şifre çözme algoritmasının testi-1

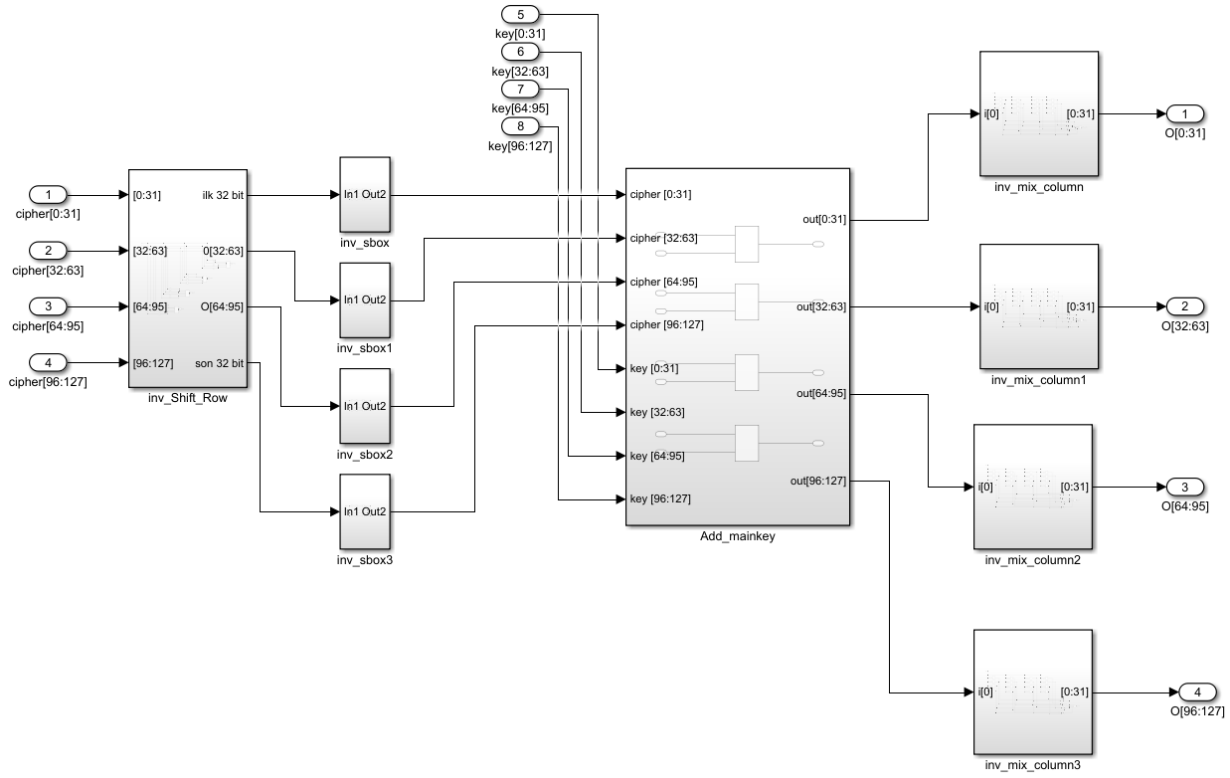


Şekil 8.27 : AES şifre çözme algoritmasının testi-2



Şekil 8.28 : AES şifre çözme algoritmasının testi-3

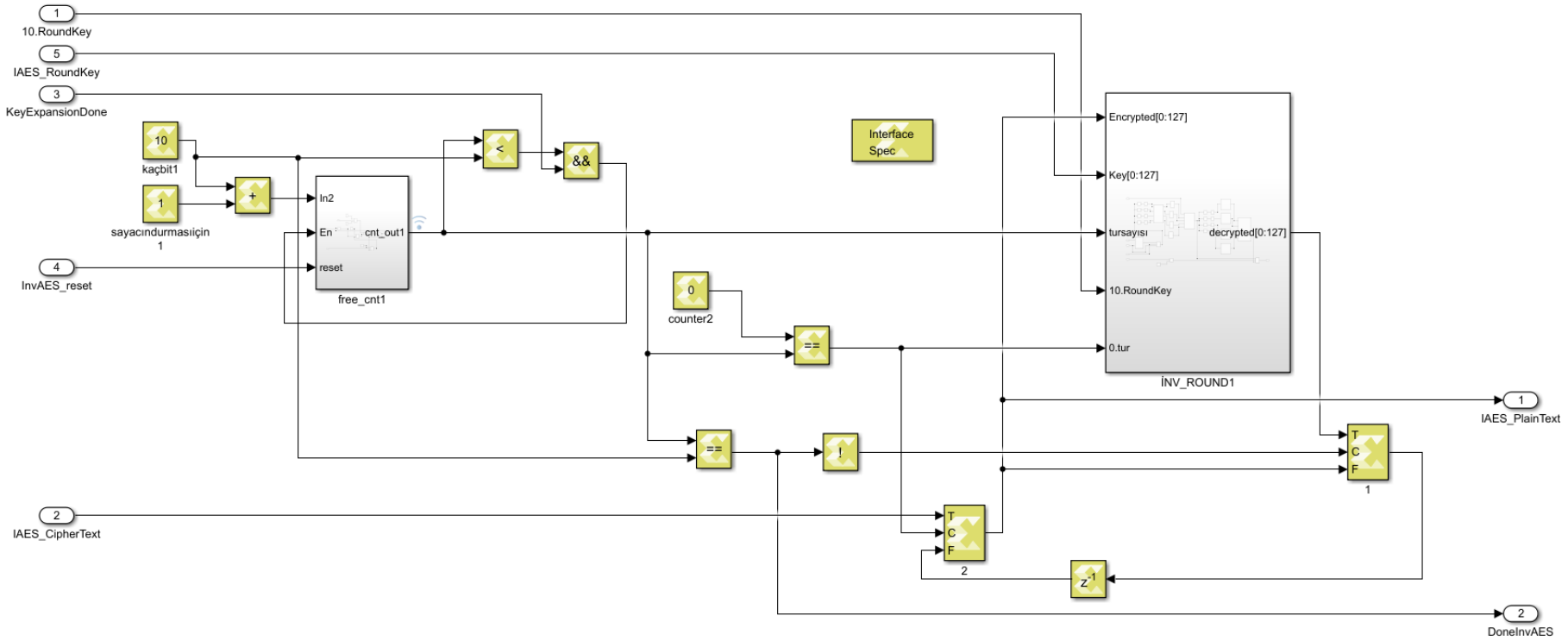
Şekil 8.29'da şifre çözme bloğunda bir tur için yapılan işlemler ve sıralaması gösterilmiştir. Yapılan işlemlerin detayları ileride daha detaylı anlatılacaktır.



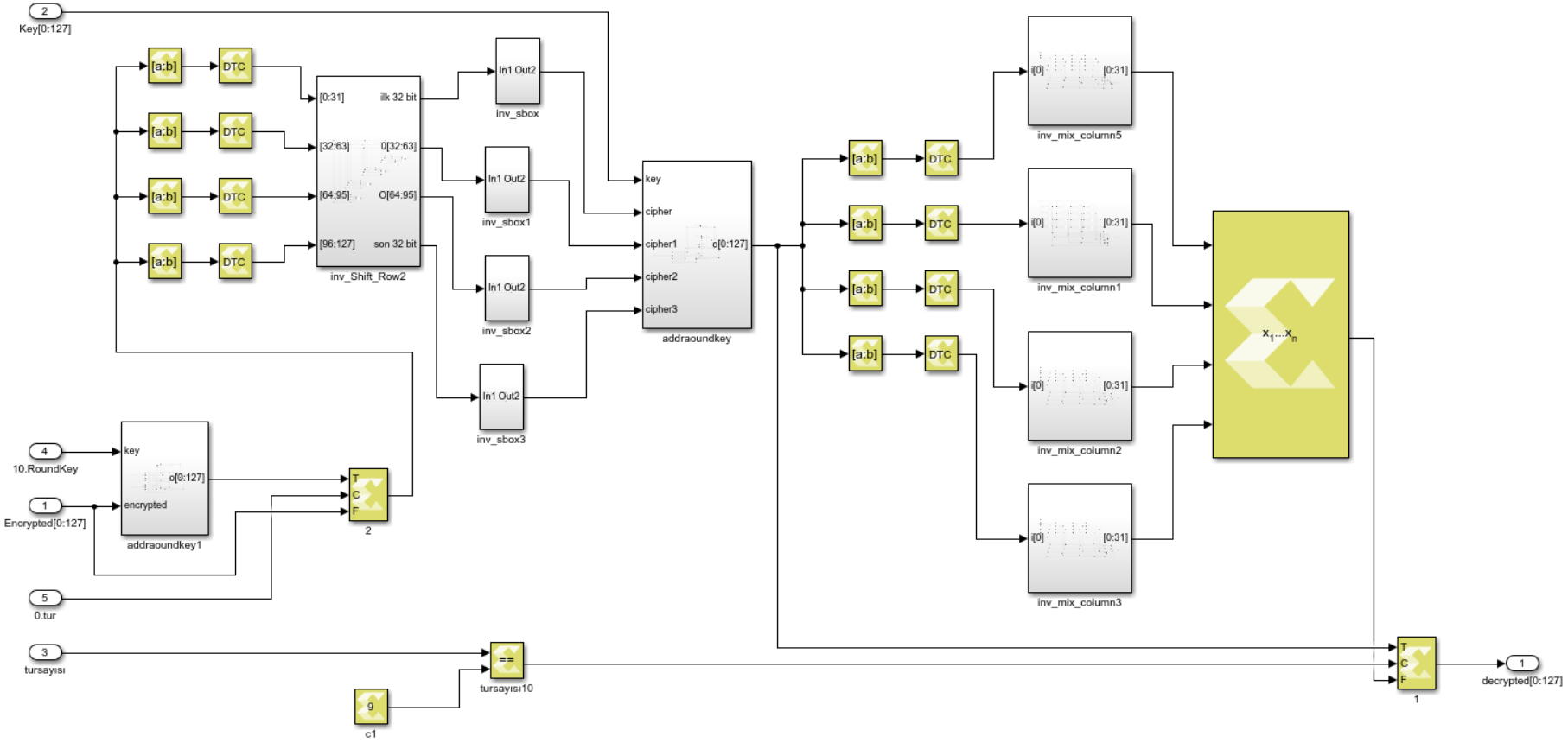
Şekil 8.29 : Kombinezonsal "INV_Round" alt modülünün iç tasarımı

Ardışıl Tasarım

AES şifreleme algoritmasının ardışıl tasarım-2 (Şekil 8.10) gerçeklemesinde AES_Round1 bloğunun yerine INV_Round1 bloğu konmuştur. Şekil 8.30'de ise INV_Round1 bloğunun iç yapısı gösterilmiştir. Kombinezonsal olarak tasarlanan şifre çözme algoritmasında bir turda kullanılan alt bloğun aynısıdır. Tek fark onuncu turda sütun karıştırma işlemi kullanılmadığı için ekstra koşul bloğu kullanılmıştır.



Şekil 8.30 : Ardışıl AES şifre çözme modülünün tasarımı

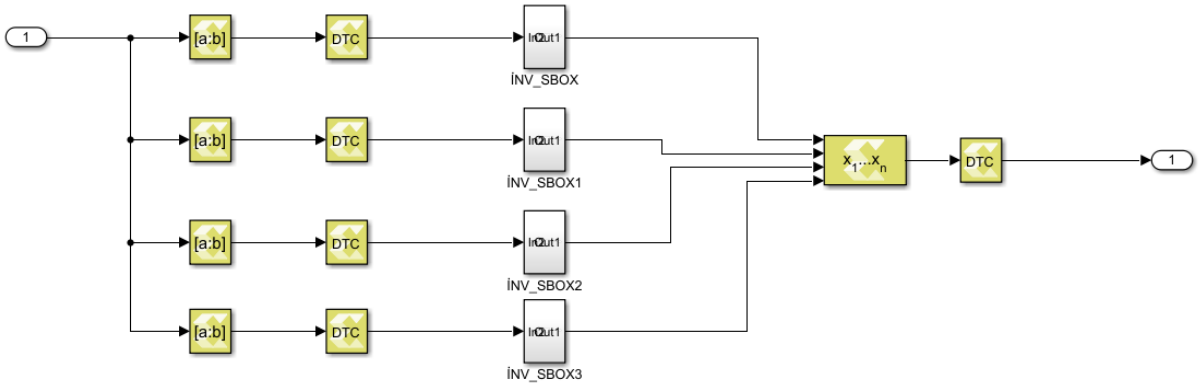


Şekil 8.31 : Ardışıl "INV_Round1" alt modülünün iç tasarımı

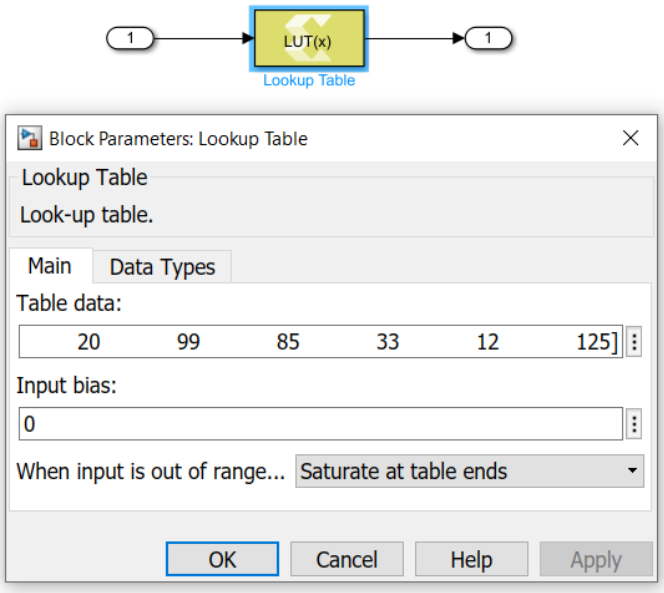
8.1.1.4 Şifre Çözme Alt Modülleri

Bayt değiştirme işleminin tersi

Model Composer ile S Kutusunun tersi kullanılmıştır. Bu işlem için "LUT" bloğu kullanılmıştır. "LUT" bloğuna girilen vektörün sonu Şekil 8.33 ile gösterilmiştir. S-Kutusu tersi matrisi de S-Kutusu matrisi gibi satır satır yerleştirilmiştirildiği fark edilmiştir. Bu özellikten yola çıkılarak 1x256 botundaki vektör değerleri LUT bloğuna ondalık sayı değerleri ile yazılmıştır. Matris değerleri dışında yapılan bütün işlemler bayt değiştirme işlemi ile aynıdır.



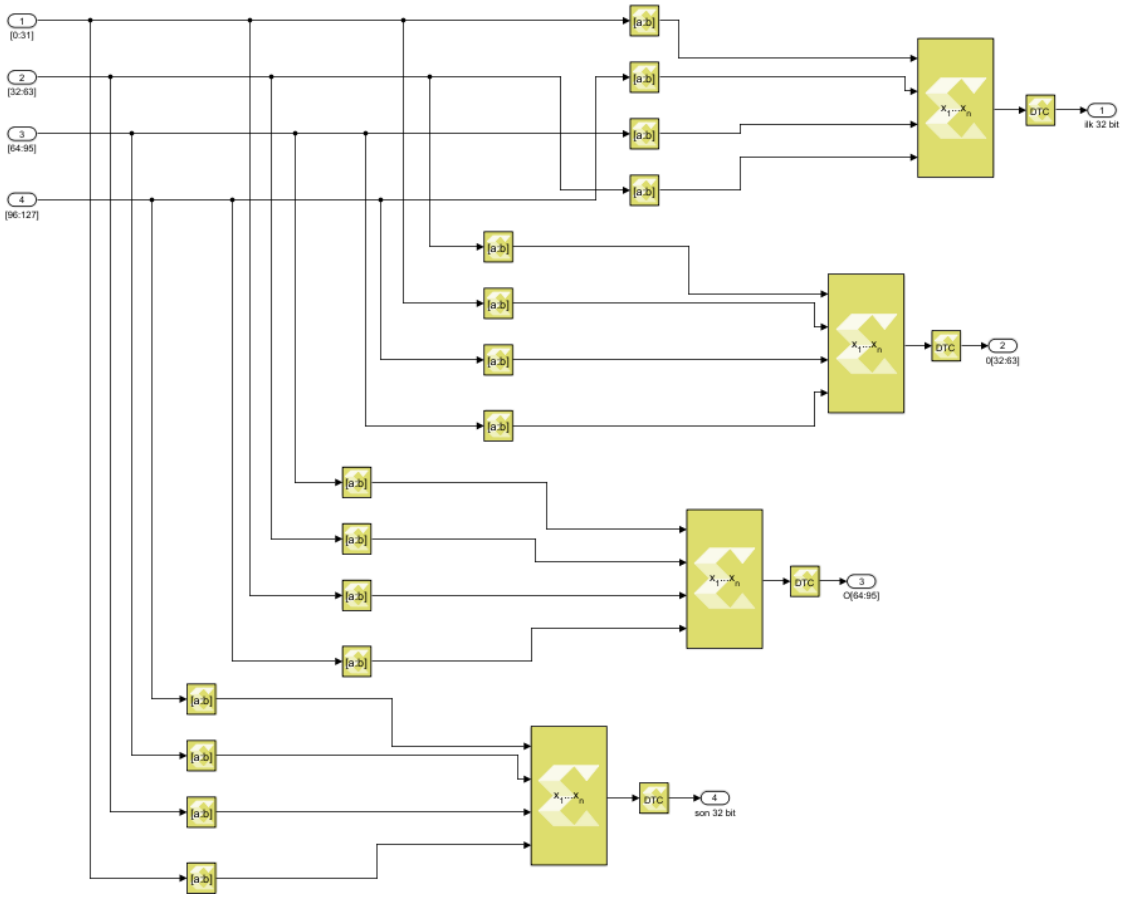
Şekil 8.32 : Bayt değiştirme işlemi tersinin tasarımı



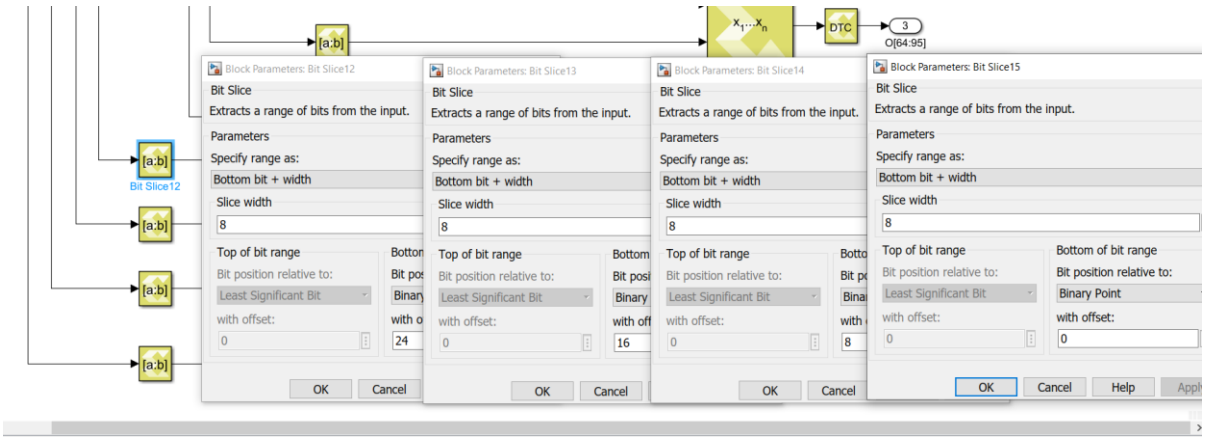
Şekil 8.33 : "LUT" bloğu parametre girişi

Satır kaydırma işleminin tersi

Model Composer ortamında tasarlanan satır kaydırma işleminin tersi ilgili bitleri Şekil 8.34'deki ekran görüntüsündeki gibi ayrılmıştır. Her bir 32 bit giriş için aynı sırada aynı bit ayırma işlemi uygulanmıştır. Tek fark girişlerin sıralamasıdır.



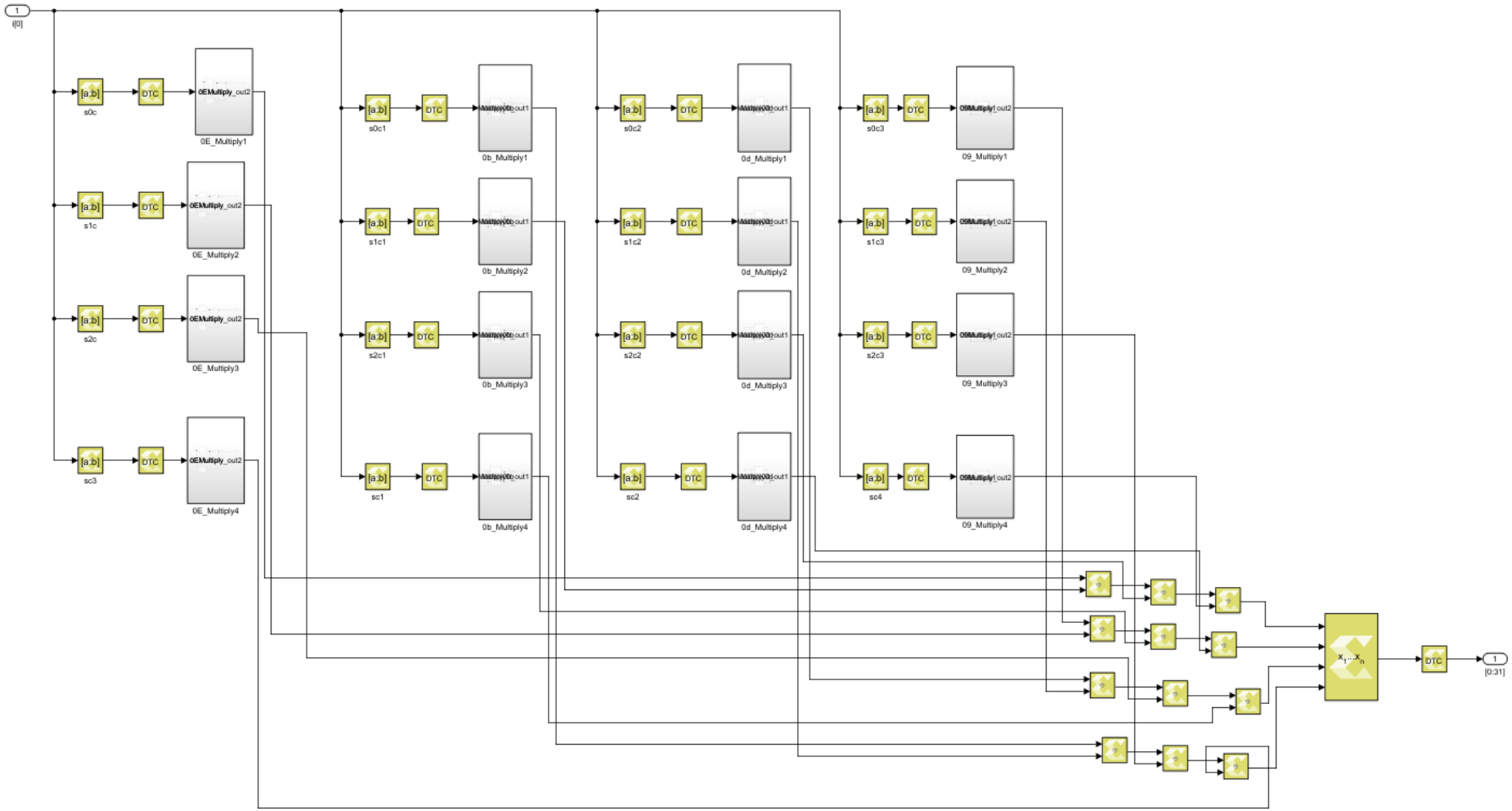
Şekil 8.34 : Satır kaydırma işlemi tersinin Model Composer tasarımı



Şekil 8.35 : "Bit Slice" bloklarının parametreleri

Sütun karıştırma işleminin tersi

Sütun Karıştırma işleminin tersi Model Composer ile tasarlanırken farklı bir yöntem izlenmiştir. Bütün tasarım sadece bit ayırma, xor ve bit birleştirme adımlarından oluşmaktadır.[82] Sütun karıştırma işleminin tersi tasarlanırken bu iyileştirme kullanılmıştır. Daha sonra FIPS 197 dökümanında anlatıldığı tasarım ile karşılaştırılması planlanıyordu ancak bu kısım projenin geliştirilmesi gereken yönlerine bırakılmıştır.



Şekil 8.36 : Sütun karıştırma işleminin tersinin tasarım

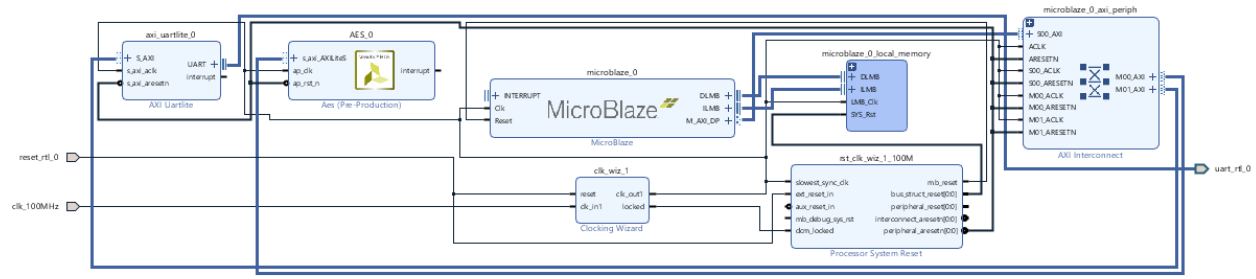
8.1.2 Xilinx Vivado

Proje tasarlama aşamasında Vivado projesi için xc7a200tfbg676-2 FPGA modeli seçilmiştir.

8.1.2.1 AES Şifreleme Algoritması Kombinezonsal Tasarımı

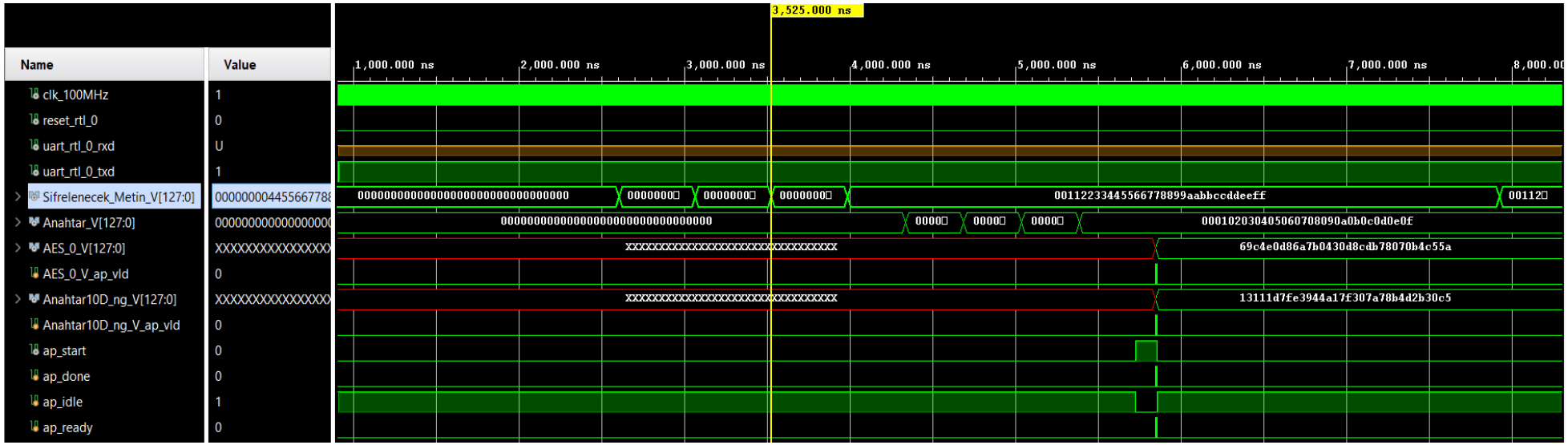
Menzil Doppler algoritmasının gerçeklenmesi sırasında Vivado için yapılması gereken adımlar anlatılmıştır. Bu kısımda sadece blok diyagram ve simülasyon sonuçları paylaşılacaktır.

AES şifreleme algoritmasının kombinezonsal tasımının test edilmesi amaçlanmıştır. Sadece AES şifreleme bloğunun IP'si kullanılmıştır. Bu projede bütün IP testleri için aynı testbench kullanılmıştır.

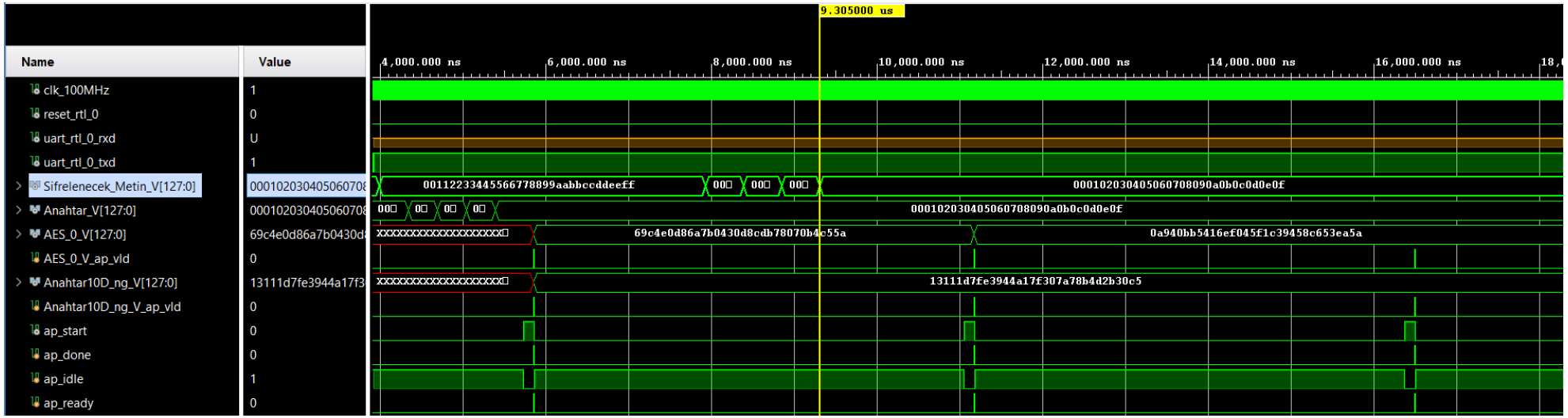


Şekil 8.37 : Kombinezonsal AES şifreleme modülünün blok tasarımı

AES şifreleme modülünün bir kere çalışmasıyla 128 bit şifrelenmektedir. Şifrelenecek metin boyutu çok daha büyük olacaktır. Tasarlanan algoritmanın buna uygun çalıştığından emin olmak amacıyla 256 bit giriş olarak verilmiştir. Aşağıda yer alan simülasyonda, ilk 128 bit için sonuçların FIPS 197 dökümanından[20] kontrol edildiğinde tasarımın doğru çalıştığı gözükmektedir. Model tabanlı tasarımın HDL koda çevrilen modülün Vivado ortamında da doğru sonuçlar verdiği test edilmiştir. İkinci simülasyon, geriye kalan 128 bitin sisteme giriş olarak doğru olarak alındığını göstermektedir. Bu durum uzun metinlerde AES şifreleme algoritmasının doğru çalışacağını göstermektedir.



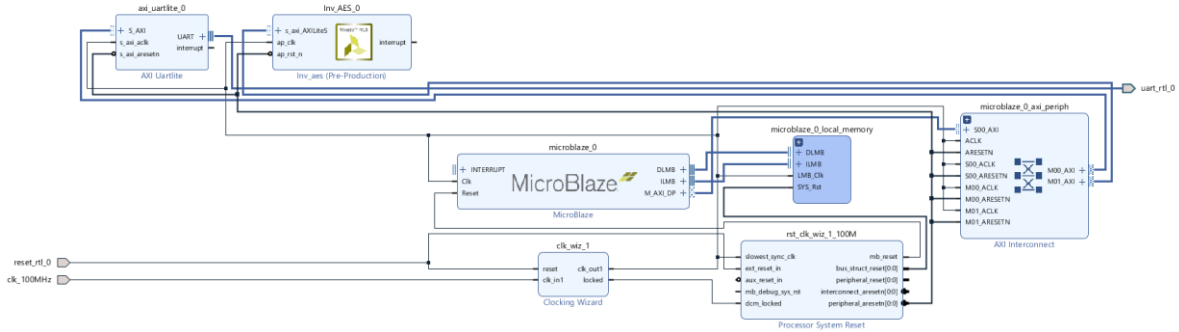
Şekil 8.38 : Kombinezonsal AES şifreleme modülü simülasyonu-1



Şekil 8.39 : Kombinezonsal AES şifreleme modülü simülasyonu-2

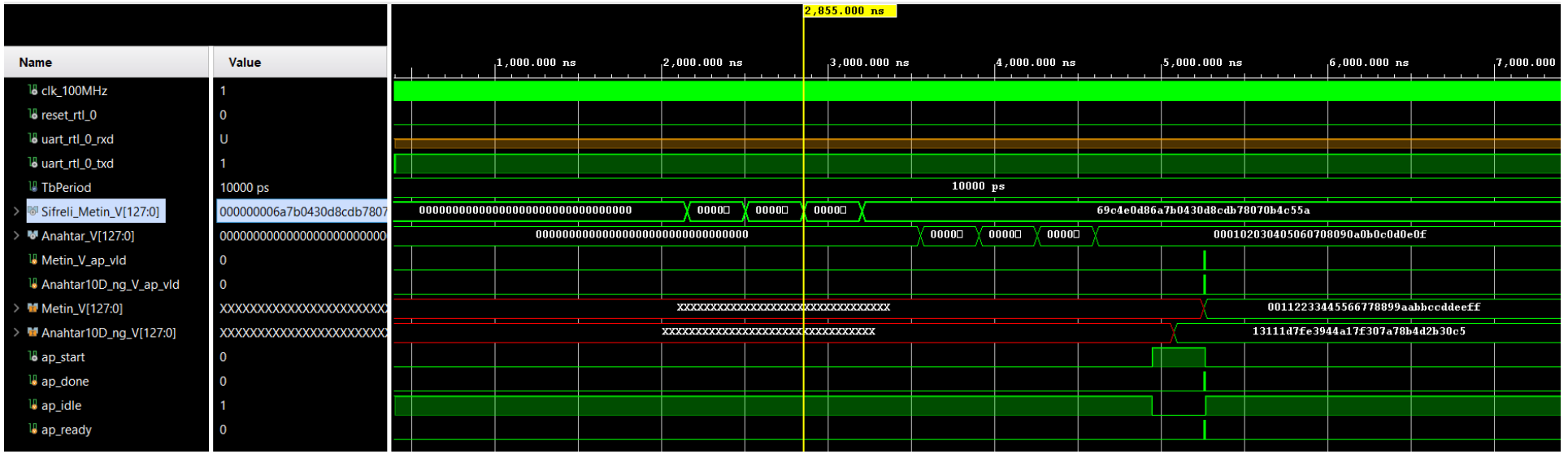
8.1.2.2 AES Şifre Çözme Algoritması Kombinezonsal Tasarımı

AES şifre çözme algoritmasının kombinezonsal tasarımının IP'si oluşturulmuştur. Bu IP ile aşağıda yer alan blok diyagram oluşturulmuş ve test edilmiştir.



Şekil 8.40 : Kombinezonsal AES şifre çözme modülünün blok tasarımı

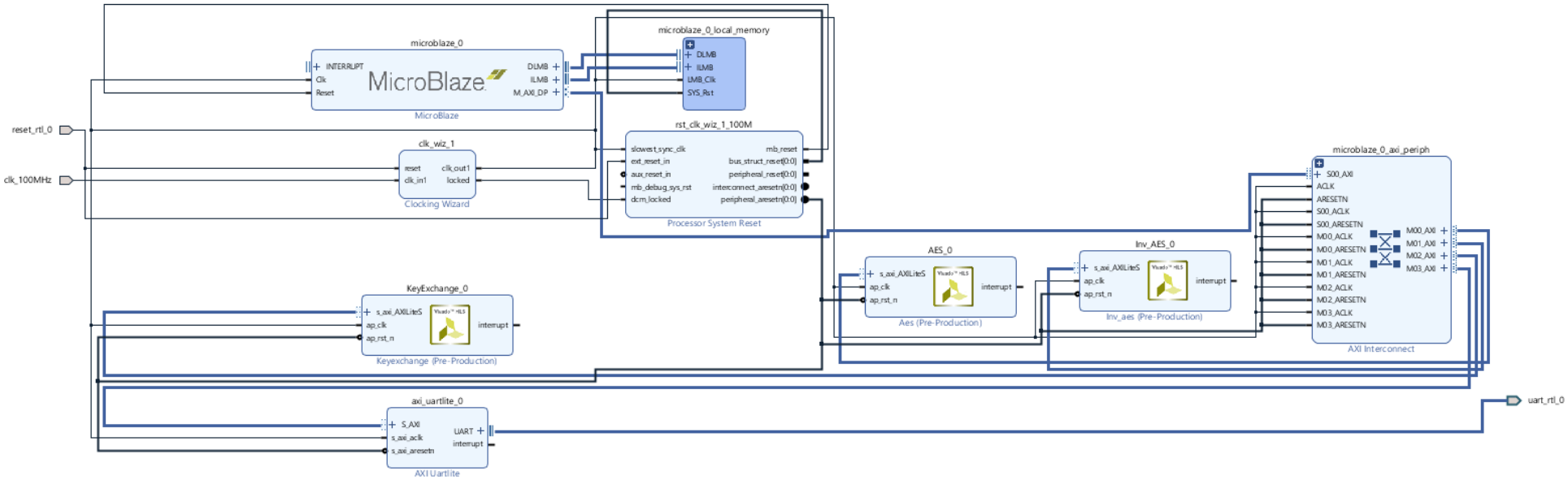
FIPS 197 dökümanında[20] yer alan şifrelenmiş veri ve kullanılan anahtar değerleri sabit sayı olarak şifre çözme algoritmasına giriş olarak verilmiştir. AES şifreleme algoritmasına verilen giriş ile şifre çözme algoritmanın sonucu eşittir. Bu şekilde şifre çözme modülünün Vivado ortamında da doğru çalıştığı ispatlanmıştır.



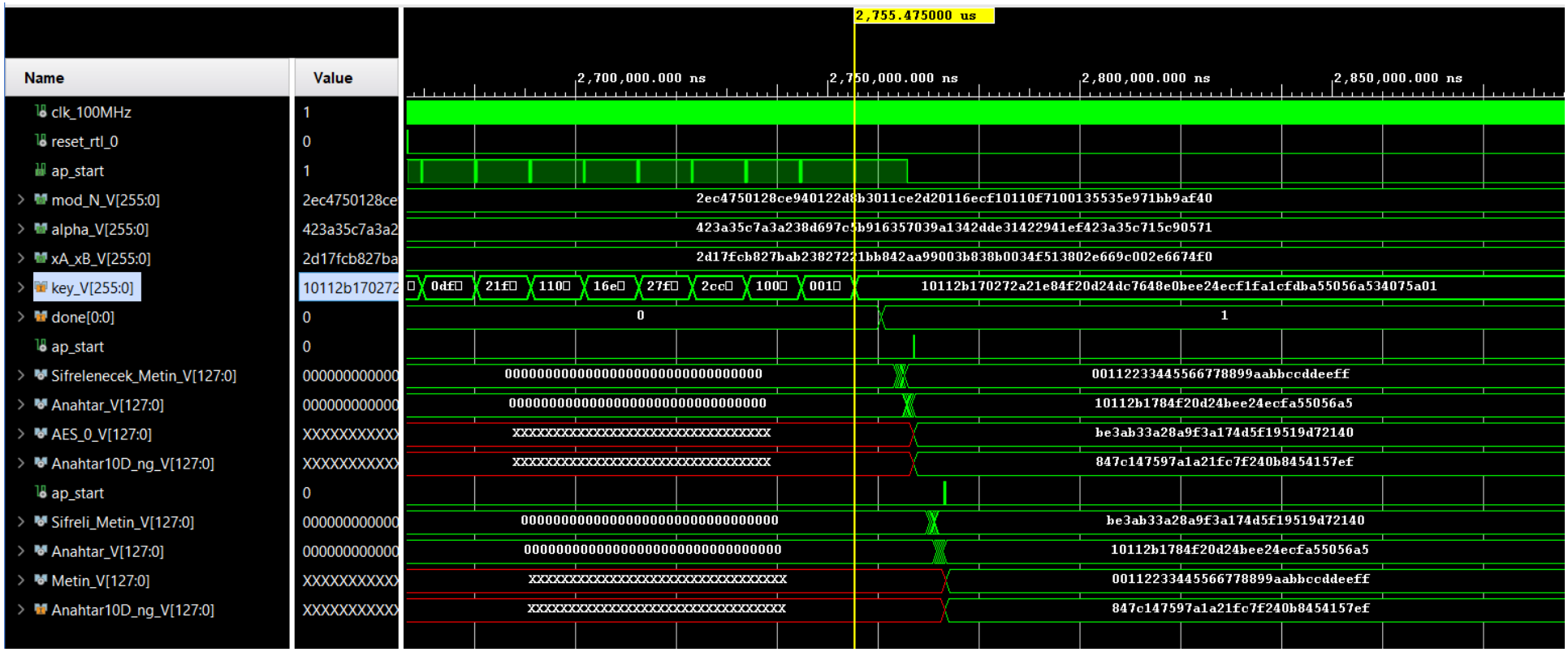
Şekil 8.41 : Kombinezonsal AES şifre çözme modülünün simülasyonu

8.1.2.3 Kombinezonsal AES şifreleme algoritması ve anahtar değişimi protokolü tasarımı

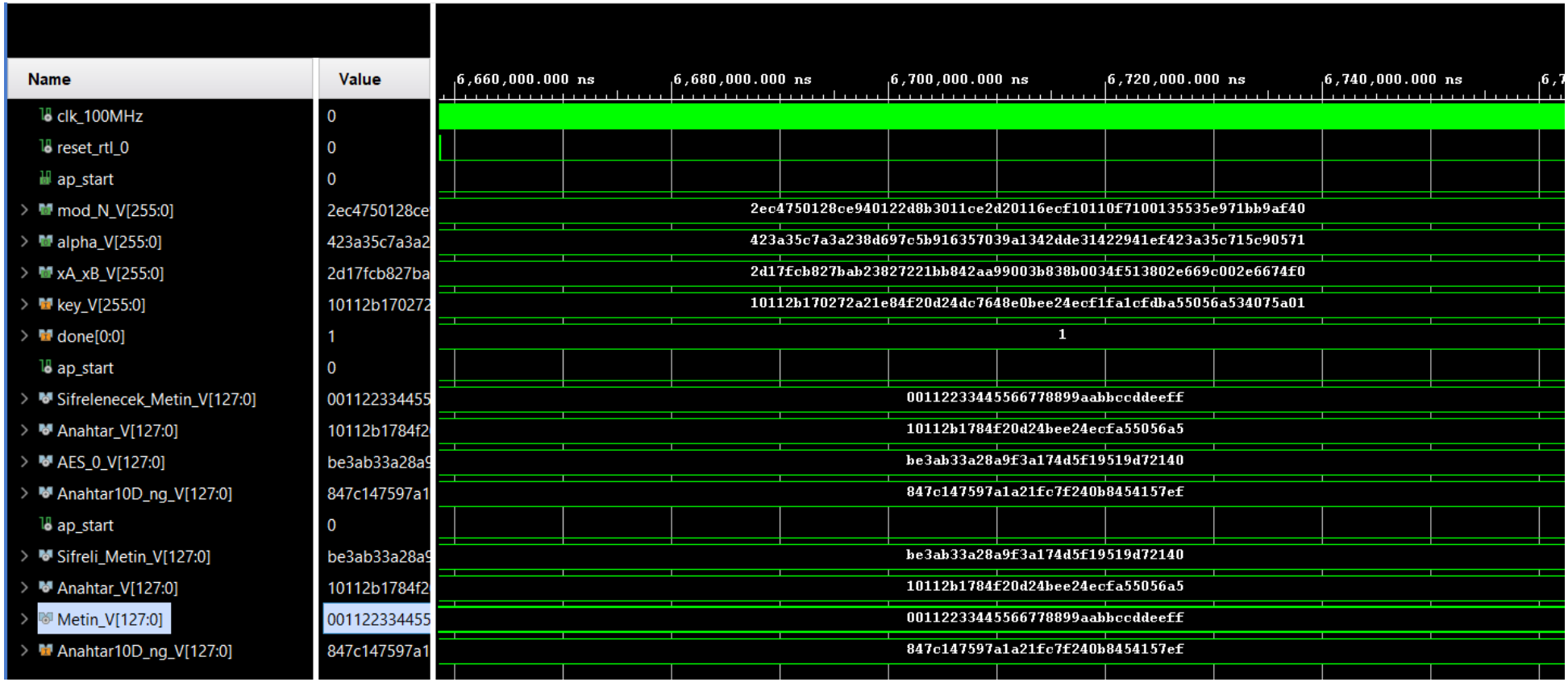
Kombinezonsal tasarımı gerçeklenen ve Vivado ortamında test edilen AES modülleri bu aşamada aynı blok diyagram birleştirilmiştir. Kombinezonsal AES bloklarına ek, anahtar paylaşımı için tasarlanan modül de kullanılmıştır. Diffie Helman anahtar değişim protokolü ileriki bölümlerde daha detaylı anlatılmıştır. Blok diyagrama eklenen IP'lerle, önce anahtar paylaşma algoritmasıyla 256 bit boyutunda anahtar üretmektir. Üretilen bu anahtarın içerisinden AES algoritmasında kullanılacak 128 bit ana anahtar seçilecektir. Bu seçim işlemi belirlenen kurallara göre belli sırada olacaktır. Aşağıda yer alan simülasyonda ana anahtar, anahtar paylaşımı algoritmasıyla üretilen 256 bitten sırayla bir 32 biti alıp bir 32 biti almayarak oluşturulmuştur. AES şifreleme modülüne şifrelenecek metin sabit değer olarak verilmiştir. Simülasyonda bu verinin şifrelendiğini, daha sonra şifre çözme bloğu ile şifresinin çözüldüğü görülmektedir. Şifre çözme modülünün çıkışı olan "Metin_v" sinyali ile şifreleme modülüne sabit olarak verilen ve simülasyonda "Sifrelenecek_Metin_V" sinyali olan verinin eşit olduğu gözlemlenmektedir.



Şekil 8.42 : Kriptoloji modülü blok diyagramı



Şekil 8.43 : Kriptoloji modülü simülasyonu

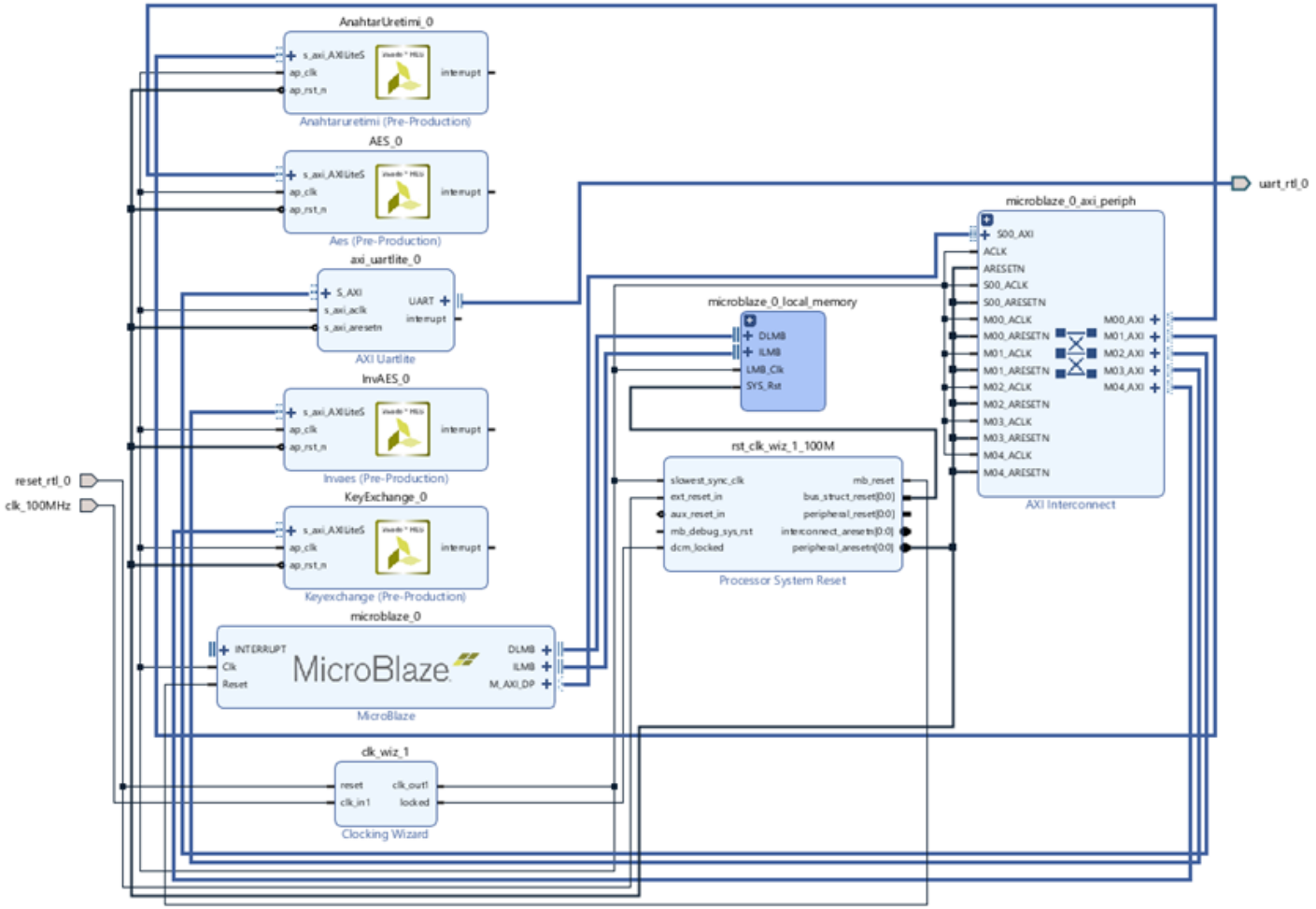


Şekil 8.44 : Kriptoloji modülü sonuç simülasyonu

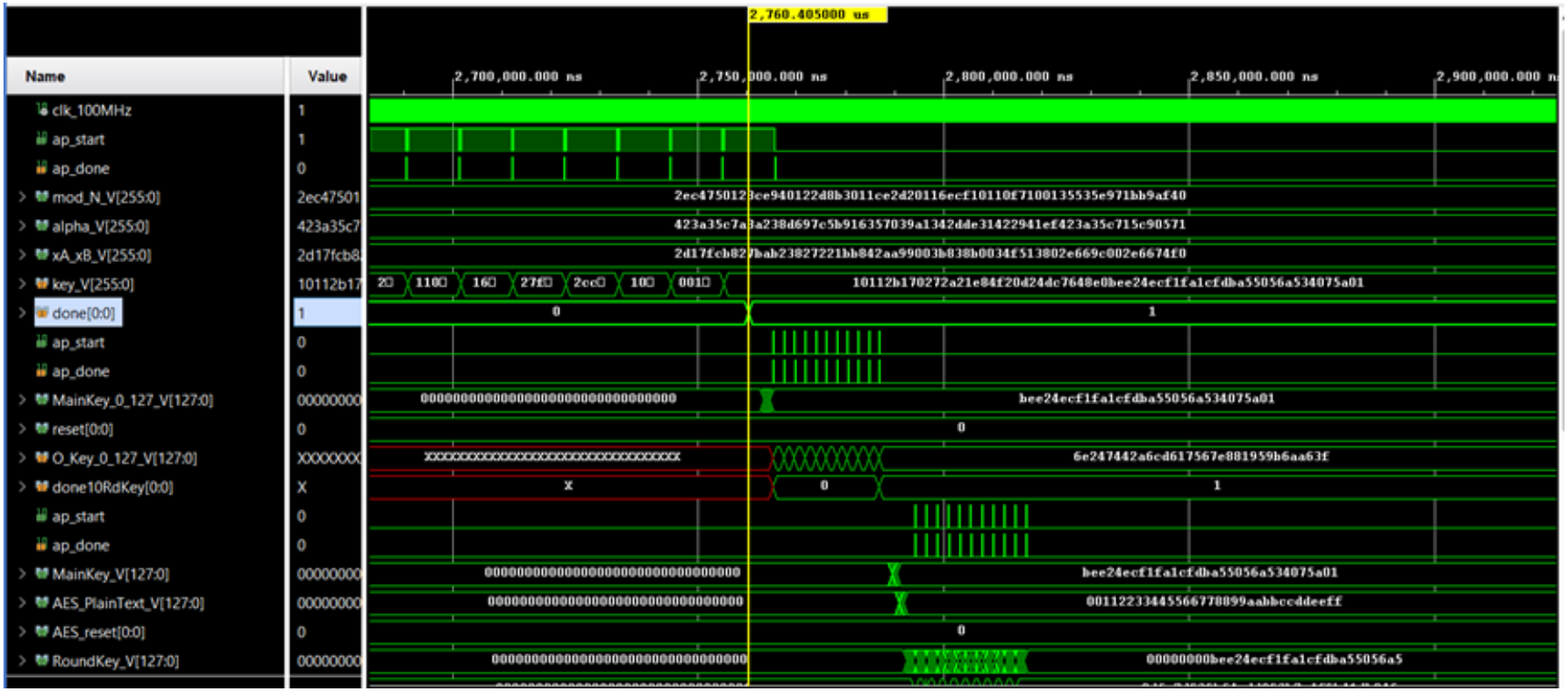
8.1.2.4 Ardışıl şifreleme algoritmaları ve anahtar değişim protokol tasarımı

Bütün sistem ardışıl olarak tasarlanmıştır. Sistemde sırasıyla anahtar değişim algoritması, ardışıl olarak tasarlanan AES şifreleme ve şifre çözme algoritmaları ve son olarak yine ardışıl olarak tasarlanan anahtar genişletilmesi algoritması yer almaktadır. Şekil 8.45'de tasarımın blok diyagramı verilmiştir.

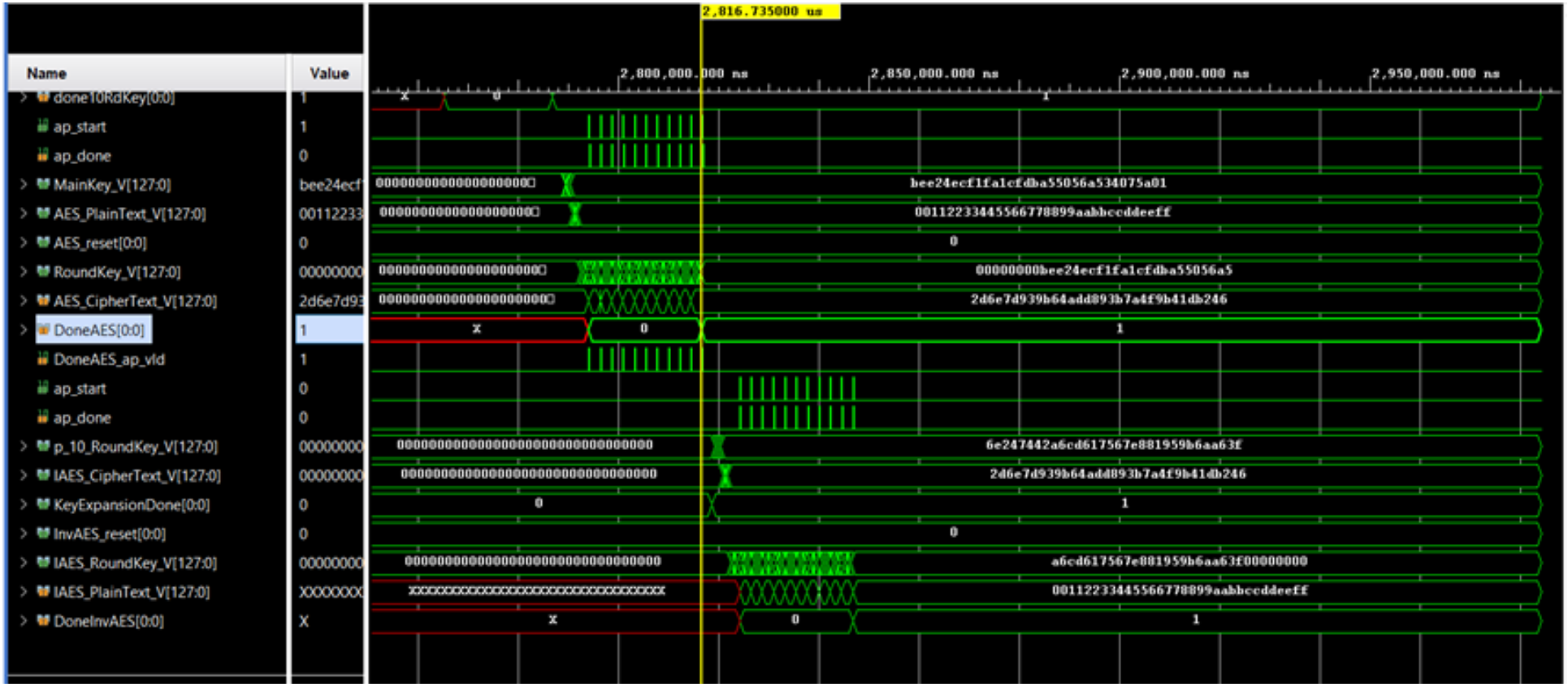
AES_PlainText sinyaline sabit olarak verilen mesaj AES şifreleme bloğu ile şifrelenmiştir. Şifrelenmiş metni AES_CipherText sinyalinden görebiliriz. Şifrelenmiş metin AES şifre çözme bloğuna doğrudan bağlanmıştır. AES şifre çözme bloğunun sonucu da IAES_PlainText çıkışından okunmaktadır. Şifrelenmek üzere verilen metin ile şifre çözme bloğunun çıkışının eşit olduğu gözükmektedir. Buradan blokların doğru şekilde çalıştığı anlaşılabilmektedir.



Şekil 8.45 : Ardışıl kriptoloji modülün blok diyagramı



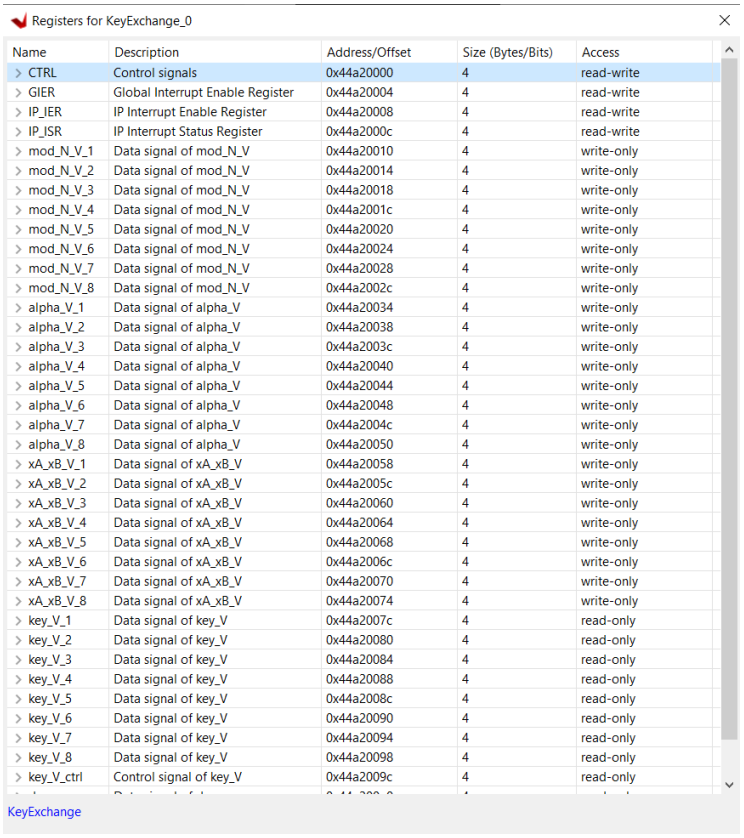
Şekil 8.46 : Ardışıl kriptoloji modülü simülasyonu



Şekil 8.47 : Ardışıl kriptoloji modülü simülasyonu-2

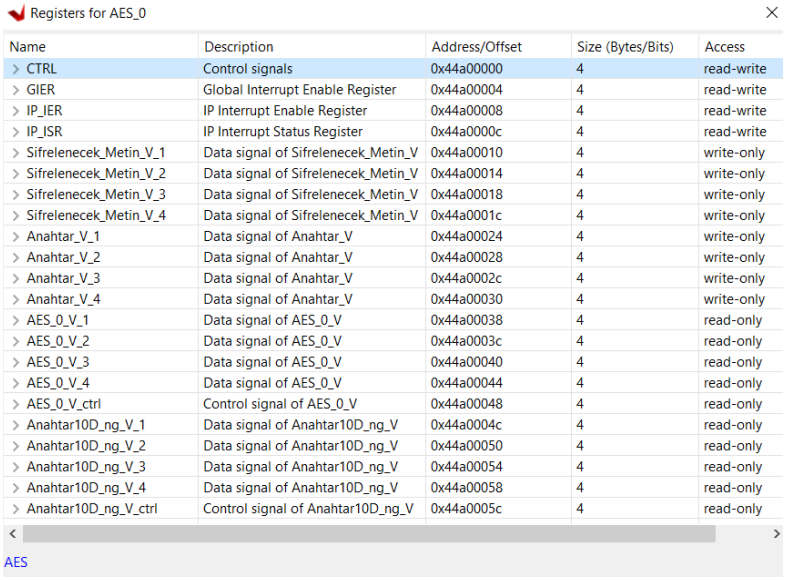
8.1.3 Vitis

Microblaze işlemcisinin kontrolü diğer blokların tasarımında olduğu gibi Vitis programı üzerinden yapılmıştır. Vivado'da oluşturulan xsa dosyası ile proje açılmış ve ardından boş bir C dosyası açılmıştır. Bu dosyaya AES şifreleme, şifre çözme ve anahtar değişimi bloklarının giriş çıkışları verilmiştir.(Şekil 8.48 - Şekil 8.49 – Şekil 8.50)



Name	Description	Address/Offset	Size (Bytes/Bits)	Access
CTRL	Control signals	0x44a20000	4	read-write
GIER	Global Interrupt Enable Register	0x44a20004	4	read-write
IP_IER	IP Interrupt Enable Register	0x44a20008	4	read-write
IP_ISR	IP Interrupt Status Register	0x44a2000c	4	read-write
mod_N_V_1	Data signal of mod_N_V	0x44a20010	4	write-only
mod_N_V_2	Data signal of mod_N_V	0x44a20014	4	write-only
mod_N_V_3	Data signal of mod_N_V	0x44a20018	4	write-only
mod_N_V_4	Data signal of mod_N_V	0x44a2001c	4	write-only
mod_N_V_5	Data signal of mod_N_V	0x44a20020	4	write-only
mod_N_V_6	Data signal of mod_N_V	0x44a20024	4	write-only
mod_N_V_7	Data signal of mod_N_V	0x44a20028	4	write-only
mod_N_V_8	Data signal of mod_N_V	0x44a2002c	4	write-only
alpha_V_1	Data signal of alpha_V	0x44a20034	4	write-only
alpha_V_2	Data signal of alpha_V	0x44a20038	4	write-only
alpha_V_3	Data signal of alpha_V	0x44a2003c	4	write-only
alpha_V_4	Data signal of alpha_V	0x44a20040	4	write-only
alpha_V_5	Data signal of alpha_V	0x44a20044	4	write-only
alpha_V_6	Data signal of alpha_V	0x44a20048	4	write-only
alpha_V_7	Data signal of alpha_V	0x44a2004c	4	write-only
alpha_V_8	Data signal of alpha_V	0x44a20050	4	write-only
xA_xB_V_1	Data signal of xA_xB_V	0x44a20058	4	write-only
xA_xB_V_2	Data signal of xA_xB_V	0x44a2005c	4	write-only
xA_xB_V_3	Data signal of xA_xB_V	0x44a20060	4	write-only
xA_xB_V_4	Data signal of xA_xB_V	0x44a20064	4	write-only
xA_xB_V_5	Data signal of xA_xB_V	0x44a20068	4	write-only
xA_xB_V_6	Data signal of xA_xB_V	0x44a2006c	4	write-only
xA_xB_V_7	Data signal of xA_xB_V	0x44a20070	4	write-only
xA_xB_V_8	Data signal of xA_xB_V	0x44a20074	4	write-only
key_V_1	Data signal of key_V	0x44a2007c	4	read-only
key_V_2	Data signal of key_V	0x44a20080	4	read-only
key_V_3	Data signal of key_V	0x44a20084	4	read-only
key_V_4	Data signal of key_V	0x44a20088	4	read-only
key_V_5	Data signal of key_V	0x44a2008c	4	read-only
key_V_6	Data signal of key_V	0x44a20090	4	read-only
key_V_7	Data signal of key_V	0x44a20094	4	read-only
key_V_8	Data signal of key_V	0x44a20098	4	read-only
key_V_ctrl	Control signal of key_V	0x44a2009c	4	read-only

Şekil 8.48 : Anahtar değişimi giriş çıkış memory adresleri

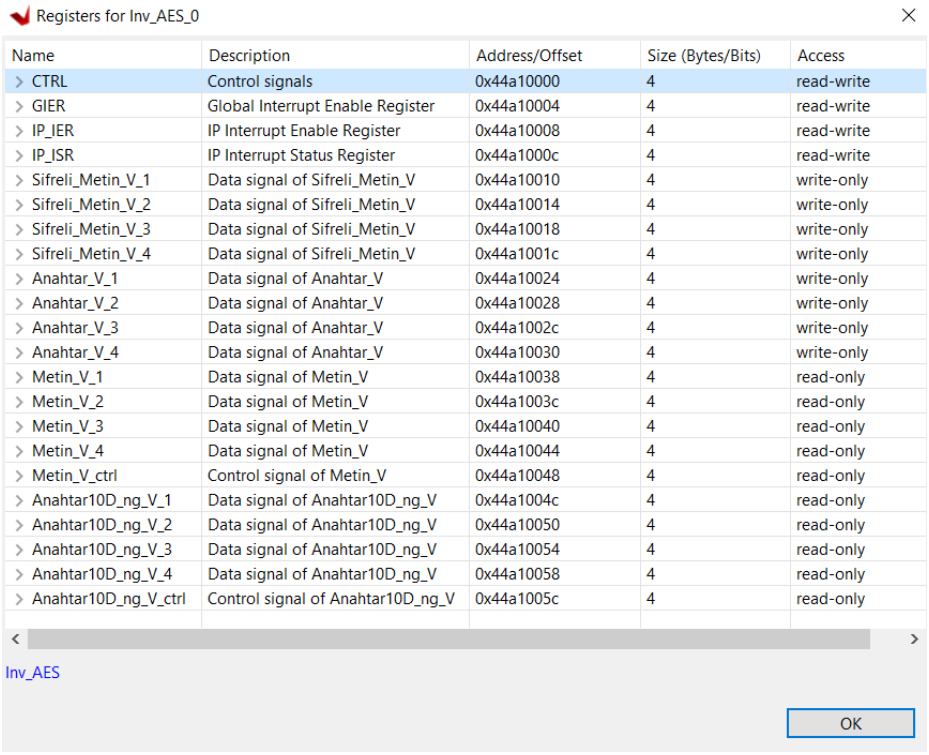


Registers for AES_0

Name	Description	Address/Offset	Size (Bytes/Bits)	Access
CTRL	Control signals	0x44a00000	4	read-write
GIER	Global Interrupt Enable Register	0x44a00004	4	read-write
IP_IER	IP Interrupt Enable Register	0x44a00008	4	read-write
IP_ISR	IP Interrupt Status Register	0x44a0000c	4	read-write
Sifrelenecek_Metin_V_1	Data signal of Sifrelenecek_Metin_V	0x44a00010	4	write-only
Sifrelenecek_Metin_V_2	Data signal of Sifrelenecek_Metin_V	0x44a00014	4	write-only
Sifrelenecek_Metin_V_3	Data signal of Sifrelenecek_Metin_V	0x44a00018	4	write-only
Sifrelenecek_Metin_V_4	Data signal of Sifrelenecek_Metin_V	0x44a0001c	4	write-only
Anahtar_V_1	Data signal of Anahtar_V	0x44a00024	4	write-only
Anahtar_V_2	Data signal of Anahtar_V	0x44a00028	4	write-only
Anahtar_V_3	Data signal of Anahtar_V	0x44a0002c	4	write-only
Anahtar_V_4	Data signal of Anahtar_V	0x44a00030	4	write-only
AES_0_V_1	Data signal of AES_0_V	0x44a00038	4	read-only
AES_0_V_2	Data signal of AES_0_V	0x44a0003c	4	read-only
AES_0_V_3	Data signal of AES_0_V	0x44a00040	4	read-only
AES_0_V_4	Data signal of AES_0_V	0x44a00044	4	read-only
AES_0_V_ctrl	Control signal of AES_0_V	0x44a00048	4	read-only
Anahtar10D_ng_V_1	Data signal of Anahtar10D_ng_V	0x44a0004c	4	read-only
Anahtar10D_ng_V_2	Data signal of Anahtar10D_ng_V	0x44a00050	4	read-only
Anahtar10D_ng_V_3	Data signal of Anahtar10D_ng_V	0x44a00054	4	read-only
Anahtar10D_ng_V_4	Data signal of Anahtar10D_ng_V	0x44a00058	4	read-only
Anahtar10D_ng_V_ctrl	Control signal of Anahtar10D_ng_V	0x44a0005c	4	read-only

AES

Şekil 8.49 : AES şifreleme giriş çıkış memory adresleri



Registers for Inv_AES_0

Name	Description	Address/Offset	Size (Bytes/Bits)	Access
CTRL	Control signals	0x44a10000	4	read-write
GIER	Global Interrupt Enable Register	0x44a10004	4	read-write
IP_IER	IP Interrupt Enable Register	0x44a10008	4	read-write
IP_ISR	IP Interrupt Status Register	0x44a1000c	4	read-write
Sifreli_Metin_V_1	Data signal of Sifreli_Metin_V	0x44a10010	4	write-only
Sifreli_Metin_V_2	Data signal of Sifreli_Metin_V	0x44a10014	4	write-only
Sifreli_Metin_V_3	Data signal of Sifreli_Metin_V	0x44a10018	4	write-only
Sifreli_Metin_V_4	Data signal of Sifreli_Metin_V	0x44a1001c	4	write-only
Anahtar_V_1	Data signal of Anahtar_V	0x44a10024	4	write-only
Anahtar_V_2	Data signal of Anahtar_V	0x44a10028	4	write-only
Anahtar_V_3	Data signal of Anahtar_V	0x44a1002c	4	write-only
Anahtar_V_4	Data signal of Anahtar_V	0x44a10030	4	write-only
Metin_V_1	Data signal of Metin_V	0x44a10038	4	read-only
Metin_V_2	Data signal of Metin_V	0x44a1003c	4	read-only
Metin_V_3	Data signal of Metin_V	0x44a10040	4	read-only
Metin_V_4	Data signal of Metin_V	0x44a10044	4	read-only
Metin_V_ctrl	Control signal of Metin_V	0x44a10048	4	read-only
Anahtar10D_ng_V_1	Data signal of Anahtar10D_ng_V	0x44a1004c	4	read-only
Anahtar10D_ng_V_2	Data signal of Anahtar10D_ng_V	0x44a10050	4	read-only
Anahtar10D_ng_V_3	Data signal of Anahtar10D_ng_V	0x44a10054	4	read-only
Anahtar10D_ng_V_4	Data signal of Anahtar10D_ng_V	0x44a10058	4	read-only
Anahtar10D_ng_V_ctrl	Control signal of Anahtar10D_ng_V	0x44a1005c	4	read-only

Inv_AES

OK

Şekil 8.50 : AES şifre çözme giriş çıkış memory adresleri

Sistemin kontrolü için ilk olarak girişler veriliyor daha sonra start sinyali verilerek donanım çalıştırılıyor ve işlem sonunda sonuçlar Vitis komutlarıyla ilgili adreslerden okunuyor. Bu kodlara referans kısmında github linkinden ulaşılabilir.

Ardışıl devre tasarımında kontrol için yapılan tek değişiklik start sinyali bir döngü içinde kullanılır. Böylelikle her bir döngüde start sinyaline 1 değeri verilmiş olur.

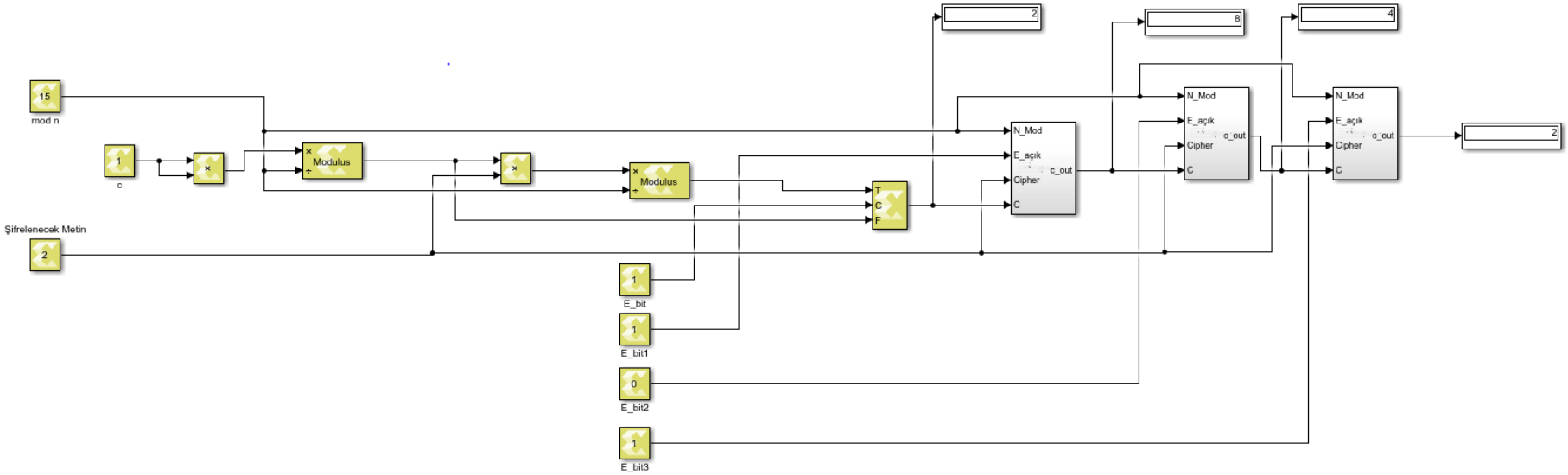
8.2 Diffie-Hellman anahtar değişim protokolü

8.2.1 Model Composer

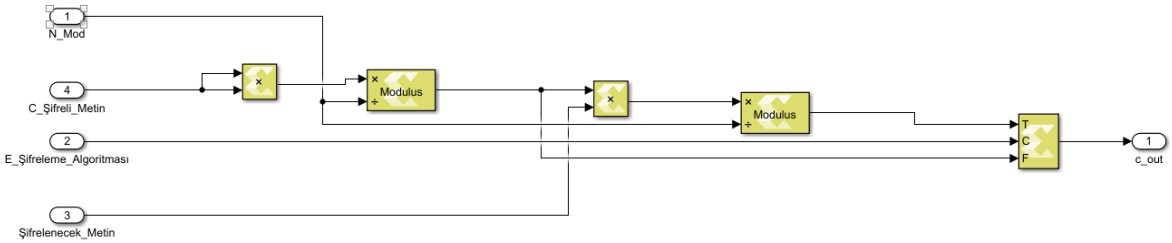
Diffie Hellman anahtar değişimi protokolünde 1024 bit uzunluğunda alpha, mod ve gizli anahtarlar kullanılır. Alpha değeri ve asal bir değer olan mod değeri herkes tarafından bilinmektedir. Biri alıcının biri göndericinin olmak üzere iki gizli anahtarlar kullanılmaktadır. Bu projede Model Composer tasarımda kullanılan bazı blokların maksimum değer olarak 1024 bit alabildiği için algoritmada kullanılan değişkenler 256 bit olarak tanımlanmıştır. Model Composer'da 2 farklı tasarım yapılmıştır. Tasarım-1 ve Tasarım-2 olarak adlandırılmıştır.

8.2.1.1 Tasarım-1

Anahtar değişim algoritmasının temel yapı taşı olan Montgomery modüler çarpma işlemi Model Composer ile gerçeklenmiştir.

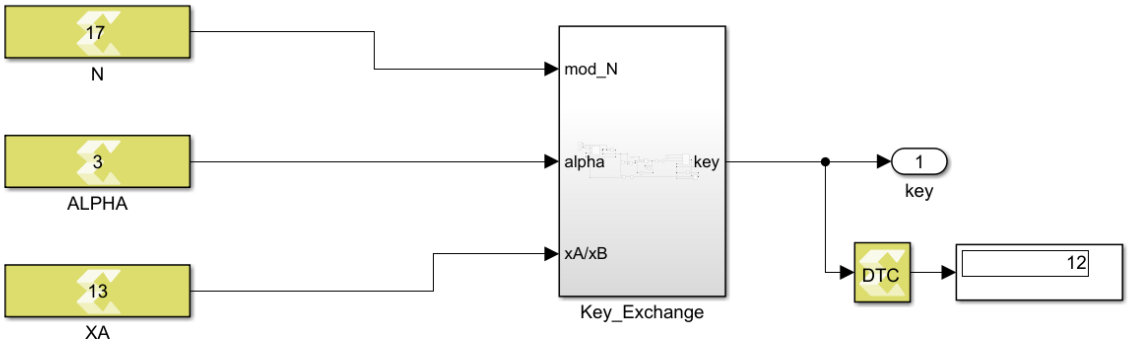


Şekil 8.51 : Montgomery modüler çarpma algoritması tasarımı-1



Şekil 8.52 : Montgomery modüler çarpma algoritması tasarımı-2

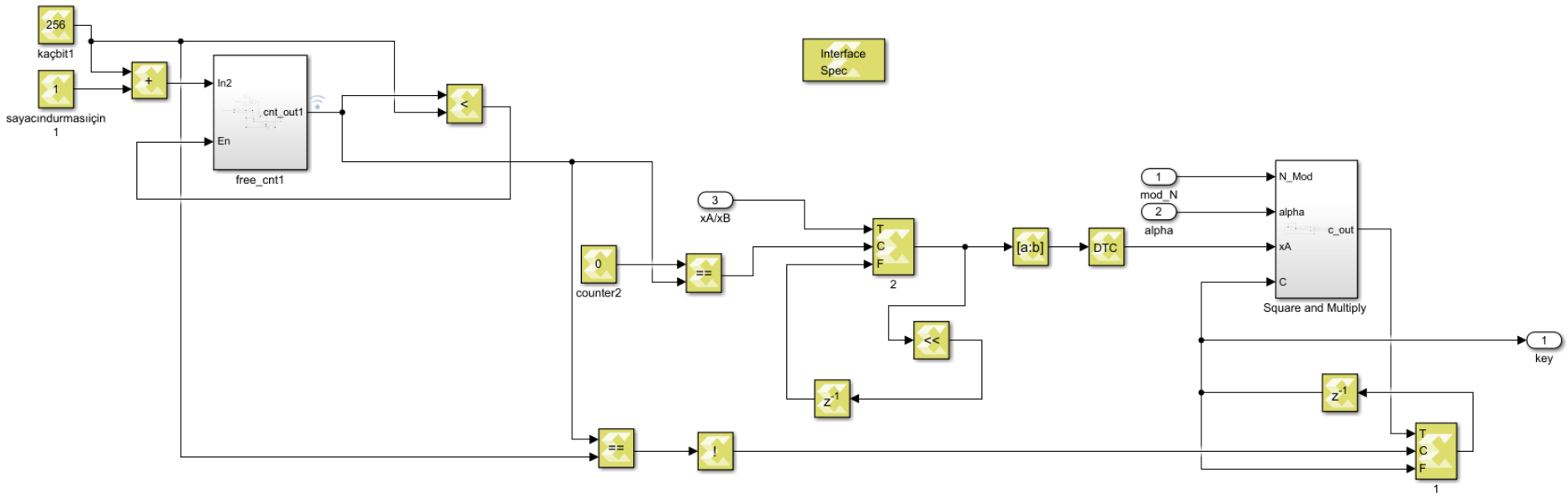
Yer istasyonu ile hava platformu için ortak anahtar kullanılması ve belli bir kullanım süresi sonrası yenilenmesi için Diffie Hellman anahtar değişimi protokolü kullanılacaktır. Bu algoritma gereğince herkes tarafından bilinen "Alpha" sayısı 1024 bit, asal bir değer olan ve yine herkes tarafından biline mod değeri "N" 1024 bit, hava platformu ve yer istasyonun 1024 bit uzunluğunda gizli "xA" ve "xB" sayıları kullanılıyor. Şekil 5.114'de 12 =(3^13)mod 17 hesaplanmıştır. Şekil-5.114'de ise algoritmasının matematiksel ifadesinin Model Composer ile gerçeklenmesi gösterilmiştir.



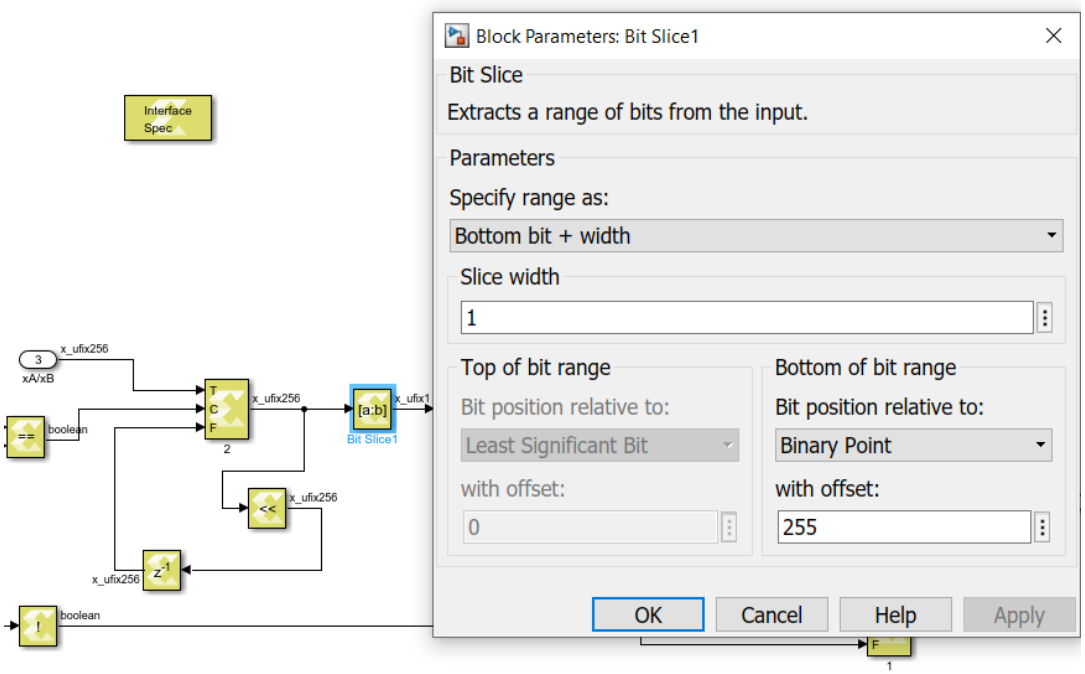
Şekil 8.53 : Diffie Hellman anahtar değişimi algoritması testi

Şekil 8.54'te gösterilen Diffie Hellman Anahtar Değişimi Algoritmasının üç numaralı girişi olan "xA/xB" diğer girişler gibi 256 bit uzunluğunluğundadır. Algoritma gereği en değerli bitinden en değersiz bitine doğru her bir döngü için bir biti, "xA" girişi olarak "Square and Multiply" bloğuna girmesi gerekir. Bu tasarım kombizonal devre olarak tasarlanamadığı için ardışıl devre olarak tasarlanmıştır. Sıralı şekilde ilgili biti giriş olarak verebilmek amacıyla "Shift Left", "Unit Delay" ve "Bit Slice" blokları kullanılmıştır. Tasarlanan algoritma her bir döngüde bir sola kaydırıp en değerli bitin

Bit Slice bloğu ile ayrılarak sisteme giriş olarak verilmesi planlanmıştır. “xA/xB”nın en değerli bitini bu şekilde sisteme alınamadığı ilk önce sola kaydırılıp daha sonra bit ayırma işlemi yapıldığı için “Conditional” bloğu kullanılmıştır. Sayaç ile sıfırıncı döngüde olma durumu kontrol edilmiş ve Conditional bloğunun “C” girişi doğru olduğu durumda direkt “xA/xB” girişi verilmiştir. “Shift Left” bloğu her bir kaydırma mikarı kadar sağa sıfır eklediği için bu döngünün 256. turda durdurulması gerekmiştir. Bu gereklilik için “free_cnt1” sayacının “En” girişine “Lesser” bloğu eklenmiştir. Sayaç bu şekilde istenilen sayı kadar saymakta daha sonrasında hep aynı çıkışı vermektedir. “Square and Multiply” bloğunun ardışıl olarak tasarlanırken “Unit Delay” kullanılmıştır. İşlem sonucu doğru sonucun çıkışa verilmesi için tasarımıa 1 numaralı Conditional bloğu eklenmiştir. Sabit değer olarak girilen değer ile sayacın sonucunun eşit olduğu durumda Conditional bloğunun “F” girişine önceki çıkış değeri bağlanmıştır.



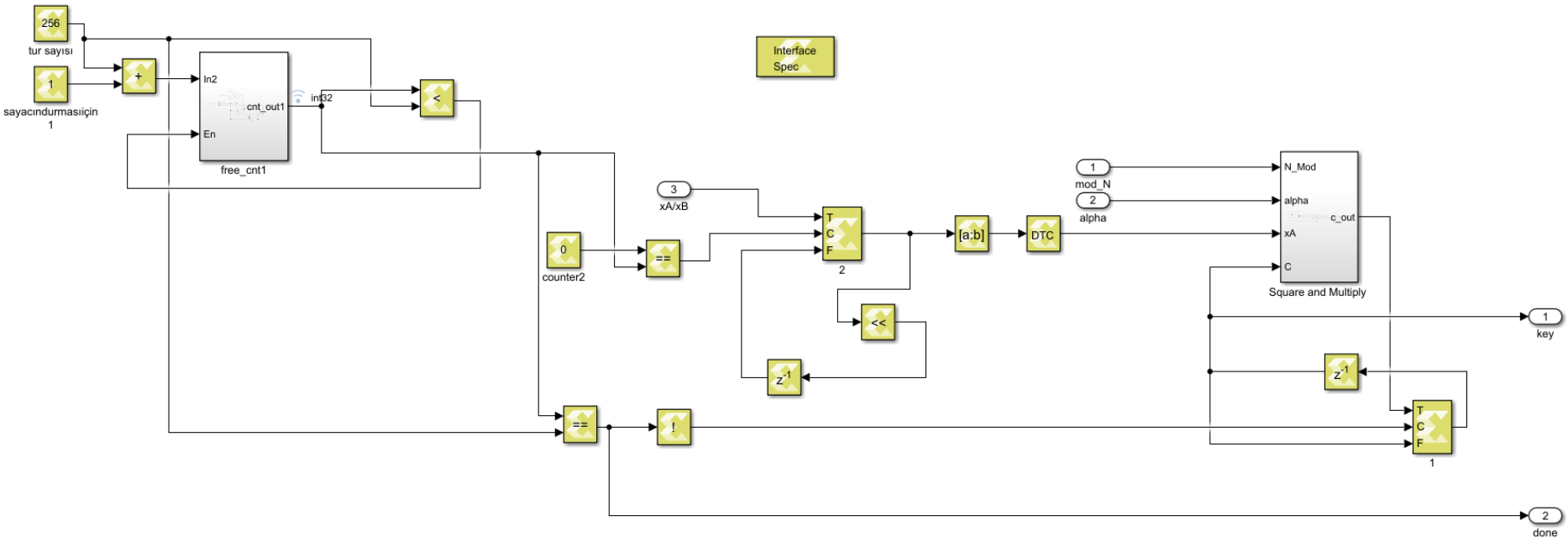
Şekil 8.54 : Diffie Hellman anahtar değişimi modül tasarımı-1



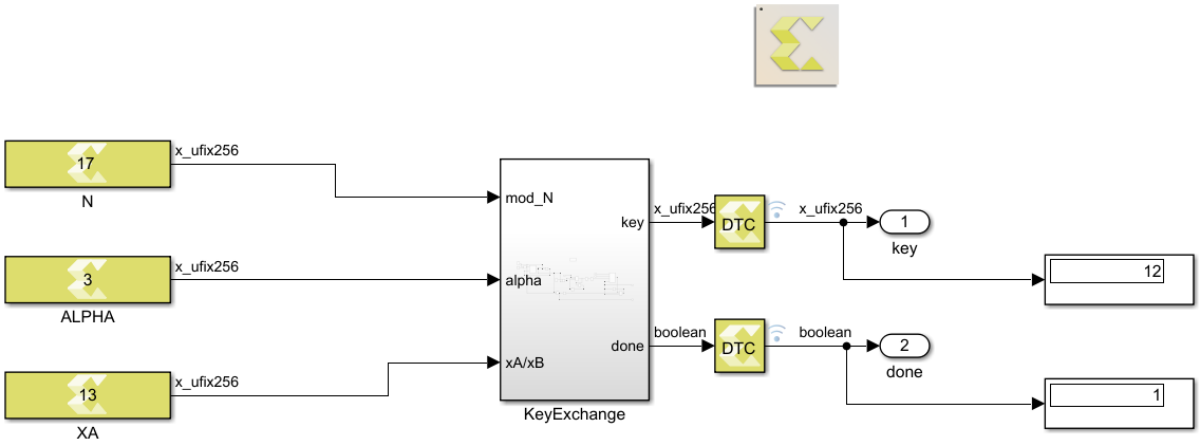
Şekil 8.55 : "Bit Slice1" bloğu kullanımı

8.2.1.2 Tasarım-2

Anahtar değişim algoritmasının bittiğinin anlaşılması için “done” sinyali çıkış olarak verilmiştir. Vitis ortamında yazılan kod ile Anahtar değişim algoritmasının bitmesi durumunda AES şifreleme bloğuna “start” sinyali ile başla komutu verilmektedir.

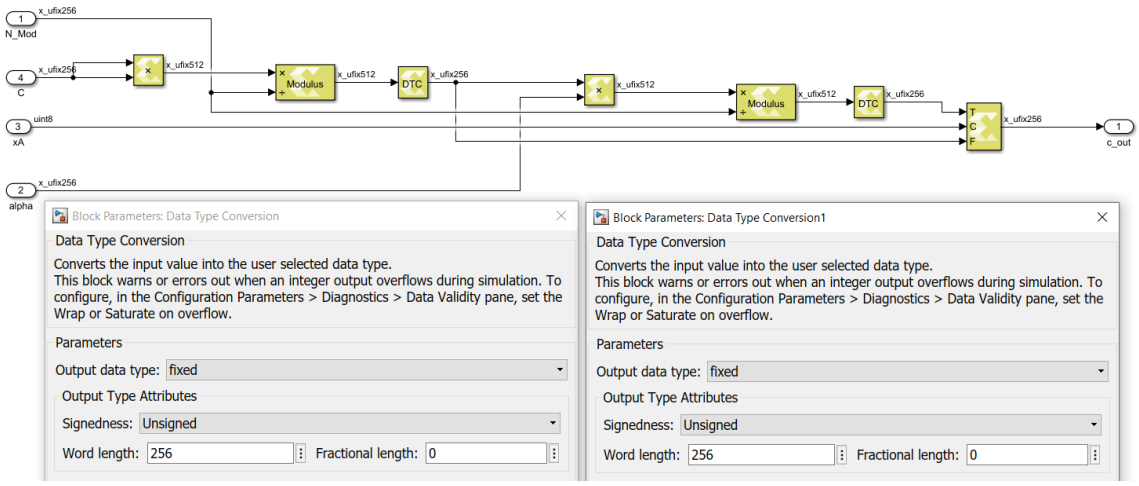


Şekil 8.56 : Diffie Hellman anahtar değişimi modül tasarımı-2



Şekil 8.57 : "KeyExchange" algoritmasının testi

"Square and Multiply" bloğunun içinde bazı değişiklikler yapılmıştır. Eklenen "Data Type Conversion" blokları ile" Modulus" bloğunun çıkışın boyutunu, girişlerden büyük olanın boyutu olarak vermesinden ortaya çıkan boyut farklılıkları sorununu ortadan kaldırmaktadır.

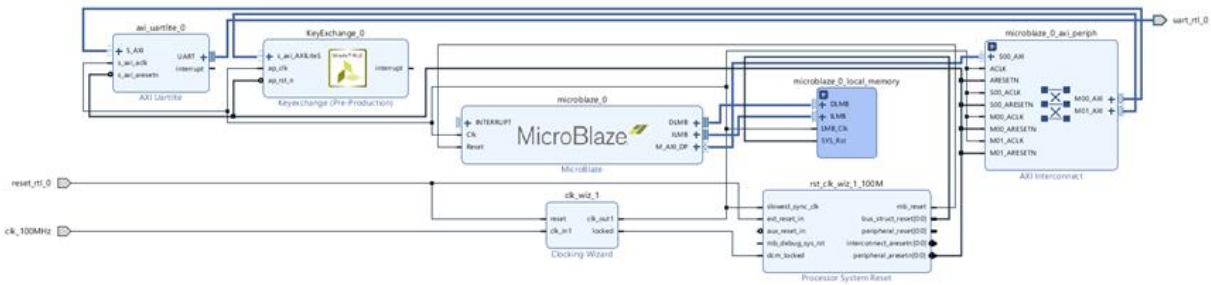


Şekil 8.58 : "Data Type Conversion" bloğu kullanımı

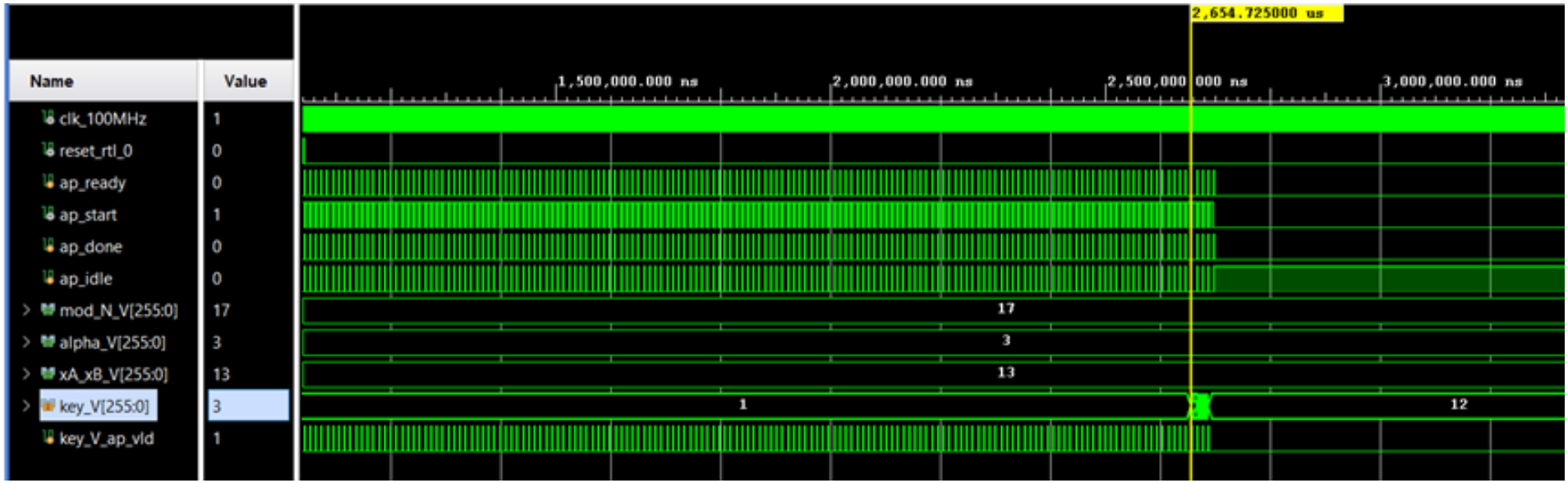
Tasarımı daha ileri taşımak amacıyla "free_cnt1" sayacına Anahtar Genişletilmesi algoritmasında olduğu gibi "reset" girişi eklenecektir. Ya da her anahtar değişim işlemi gerçekleşmeden önce Diffie-Hellman Anahtar Değişim algoritmasına reset atılması gerekmektedir. Çünkü kullanılan sayıcın çıkışının sıfırlanması gerekmektedir.

8.2.2 Xilinx Vivado

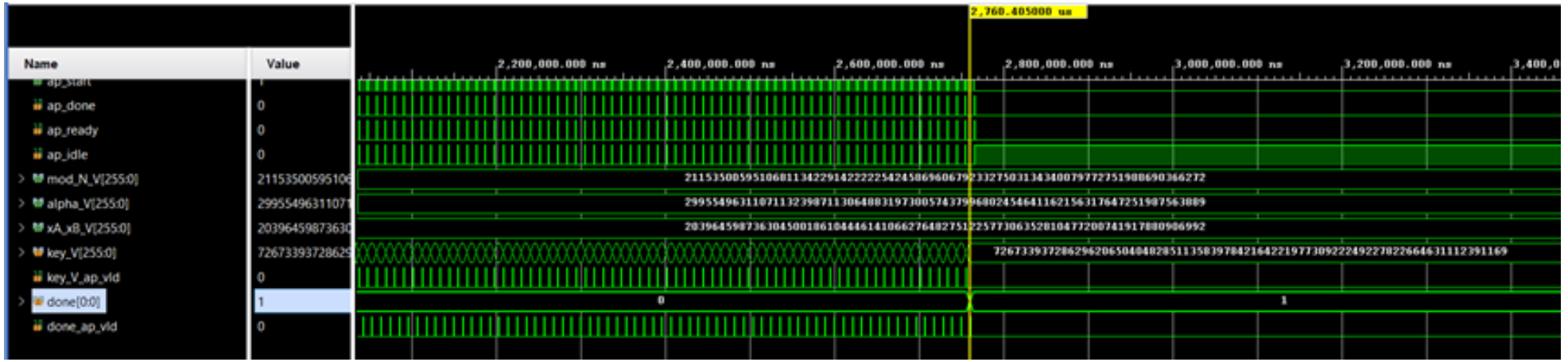
Menzil Doppler ve sinir ağı bloklarında olduğu gibi Model Composer'da üretilen IP blok diyagrama eklenmiştir. Sistemin simülasyonu da diğer modüllerde yapıldığı şekilde aynen yapılmıştır. Şekil 5.120'de blok diyagram verilmiştir. 8.60'da Tasarım-1, 8.61'de ise Tasarım-2 simülasyon sonucu verilmiştir. Key_V sinyali okunarak sonuç kontrol edilebilir. Girişlere "Alpha" sayısı olarak 3 değeri, "mod_N" mod değeri 17, gizli "xa_xB" değerine 13 verilmiştir. Şekil 5.114'de 12 =(3^13)mod 17 hesaplanmıştır.



Şekil 8.59 : Diffie Hellman anahtar değişimi modülünün blok diyagramı



Şekil 8.60 : Diffie Hellman anahtar değişimi modül tasarımı-1 simülasyonu



Şekil 8.61 : Diffie Hellman anahtar değişimi modül tasarımı-2 simülasyonu

8.2.3 Vitis

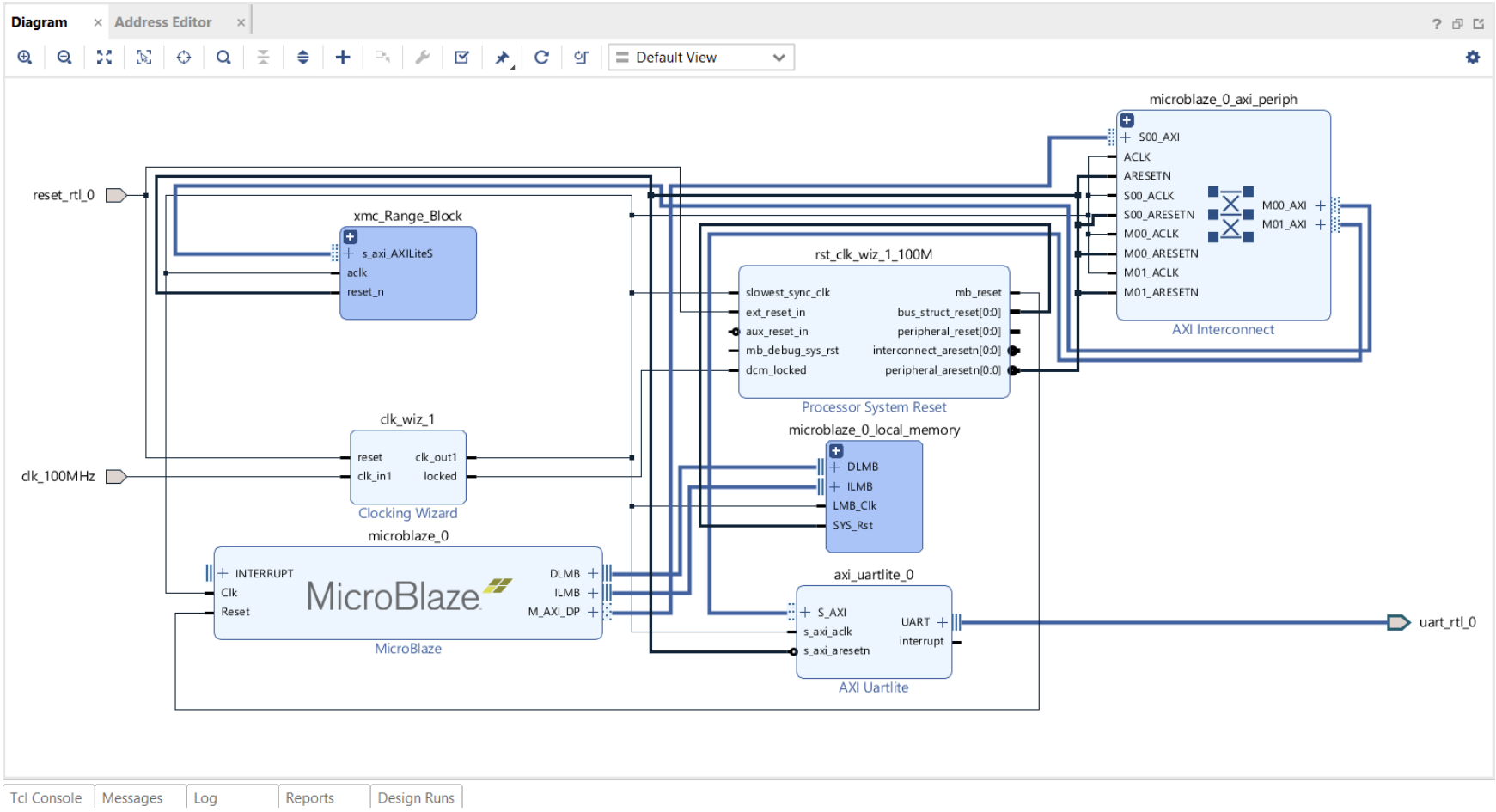
Microblaze işlemcisinin kontrolü diğer blokların tasarımında olduğu gibi Vitis programı üzerinden yapılmıştır. Vivado'da oluşturulan xsa dosyası ile proje açılmış ve ardından boş bir C dosyası açılmıştır. Bu dosyaya anahtar değişimi bloklarının giriş çıkışları verilmiştir.(Şekil 8.48)

Sistemin kontrolü için ilk olarak girişler veriliyor daha sonra start sinyali verilerek donanım çalıştırılıyor ve işlem sonunda sonuçlar Vitis komutlarıyla ilgili adreslerden okunuyor. Bu kodlara referans kısmında github linkinden ulaşılabilir.

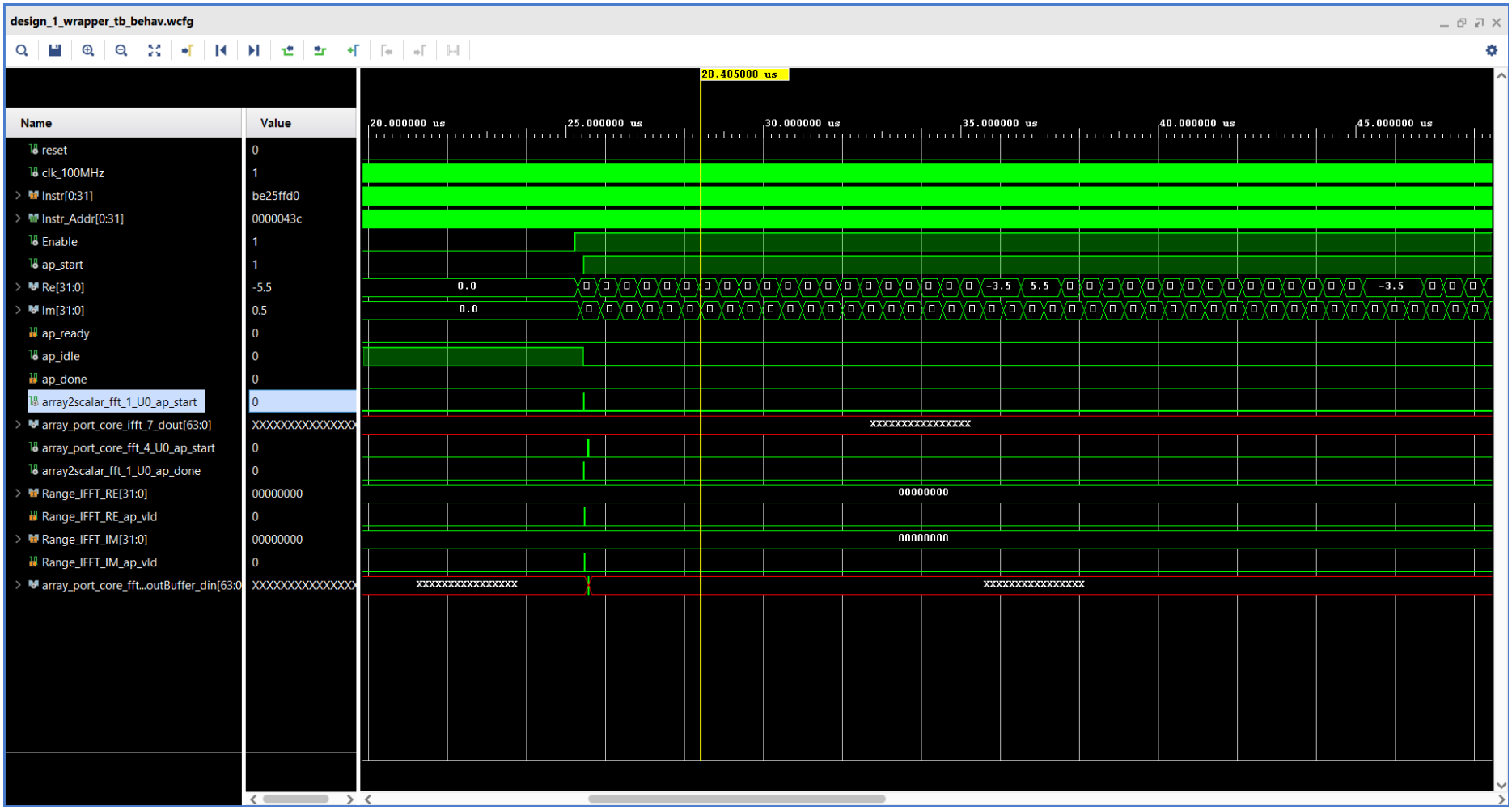
Ardışıl devre tasarımında kontrol için yapılan tek değişiklik start sinyali bir döngü içinde kullanılır. Böylelikle her bir döngüde start sinyaline 1 değeri verilmiş olur.

8.3 Sistem Benzetimi

Menzil Doppler algoritması Menzil ve Azimut işleme blokları ayrı ayrı tasarlanmıştır. Bu bloklar arasındaki işlemler aynı şekilde ancak kullanılan eşleştirme fonksiyonundan elde edilen katsayılar farklıdır. Model Composer'da üretilen IP Vivado'ya aktarılmıştır. Burada blok tasarım oluşturulmuş ve ilk olarak Microblaze eklenmiştir. Diğer bloklar da tasarıma eklenip devre sentezlenmiştir. Vitis'te giriş çıkışlar verilerek simülasyonu yapılmıştır. Simülasyonda görüldüğü gibi **Re** ve **Im** girişleri değişmektedir ancak her gelen veriden sonra **ap_start** işaretinin 1 olması gerekmektedir ancak üretilen IP'nin giriş ve çıkış sürücüleri arasındaki hatta uyumsuzluk olması sebebiyle devre istendiği gibi çalışmamaktadır. Bu soruna üretilen alternatif çözümler projenin son bölümünde verilmiştir. Bu sebepten dolayı projenin bu bloğu tüm sisteme entegre edilememiştir.



Şekil 8.62 : Menzil işleme blok tasarımı



Şekil 8.63 : Menzil işleme Vivado simülasyonu

Projenin diğer iki bloğu olan sinir ağı ve kripto blokları birleştirilmiştir. İlk olarak sinir ağının test edilebilmesi için sabit değerler giriş olarak verilmiştir. Bu değerler menzil doppler görüntü oluşturma algoritmasının sonuçları değil ancak resim verileri tipinde 256 test verisi sisteme sabit olarak verilmiştir. Bu veriler sinir ağına girer ve görüntüde su olarak belirlenen değerler tüm veriler içinde %30 oranını geçtiğinde giriş verileri yer istasyonuna gönderilmeye karar verilmiştir. Gönderilmeden önce ise kripto bloğuna girecektir. Bunun için belirlediğimiz çalışma sıralaması şu şekilde tasarlanmıştır.

1-İlk olarak anahtar değişimi algoritması çalıştırılmıştır. Bu algoritma tarafından oluşturulan 256 bit uzunluğundaki ortak anahtardan AES için kullanılacak olan 128 bit ana anahtar belirlenmiştir.

2-Bütün kripto algoritmaları ardışıl olarak tasarlandığından dolayı anahtar genişletme işlemi 128 bitlik ana anahtar belirlendikten sonra çalıştırılmıştır.

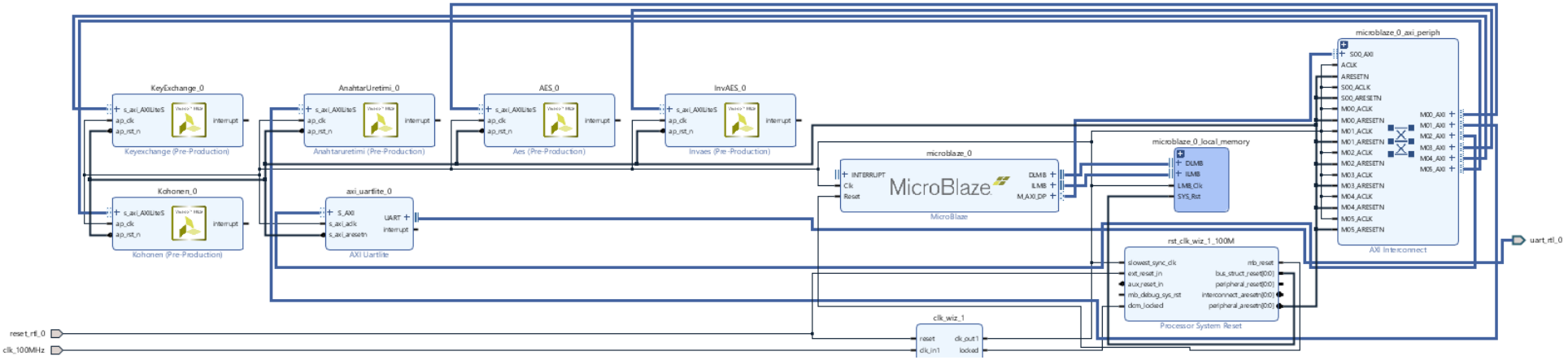
3-Belirlenen ana anahtar, tur anahtarları ve sinir ağından gelecek olan veriler AES şifreleme bloğunun girişlerini oluşturmuştur.

4-Yapılan işlemlerin doğruluğunu saptayabilmek amacıyla AES şifreleme bloğunun ardından AES şifre çözme bloğu kullanılmıştır.

Bu işlemleri yapabilmek amacıyla her bir bloğun anlatıldığı bölümlerde olduğu gibi IP'ler üretilmiş ve Vivado blok diyagrama aktarılmıştır. Elde edilen tasarım Şekil 8.64'te gösterilmiştir.

Bu sistem için yazılan Vitis'te yazılan Microblaze kodu Şekil 8.65'te gösterilmiştir . Kodlar github hesabında paylaşılmıştır.

Kontrol için yapılan simülasyonlar ise Şekil 8.66, 8.67, 8.68'de verilmiştir.

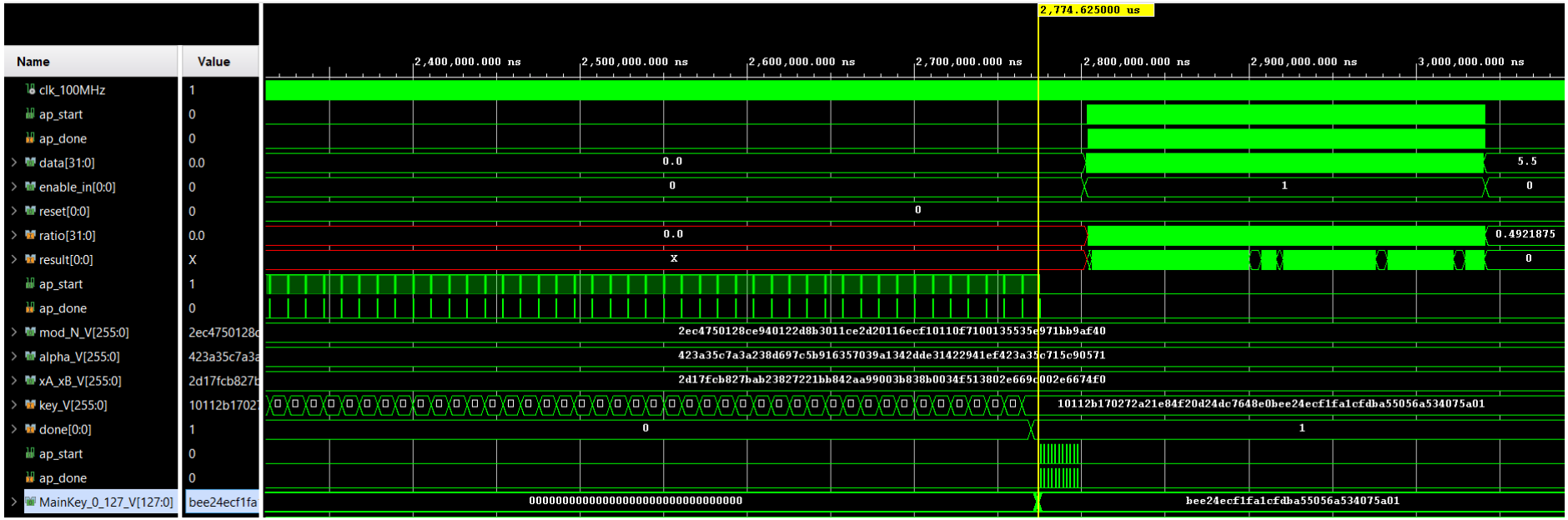


Şekil 8.64 : Sinir ağı ve kripto bloklarının birleştirilmiş tasarımı

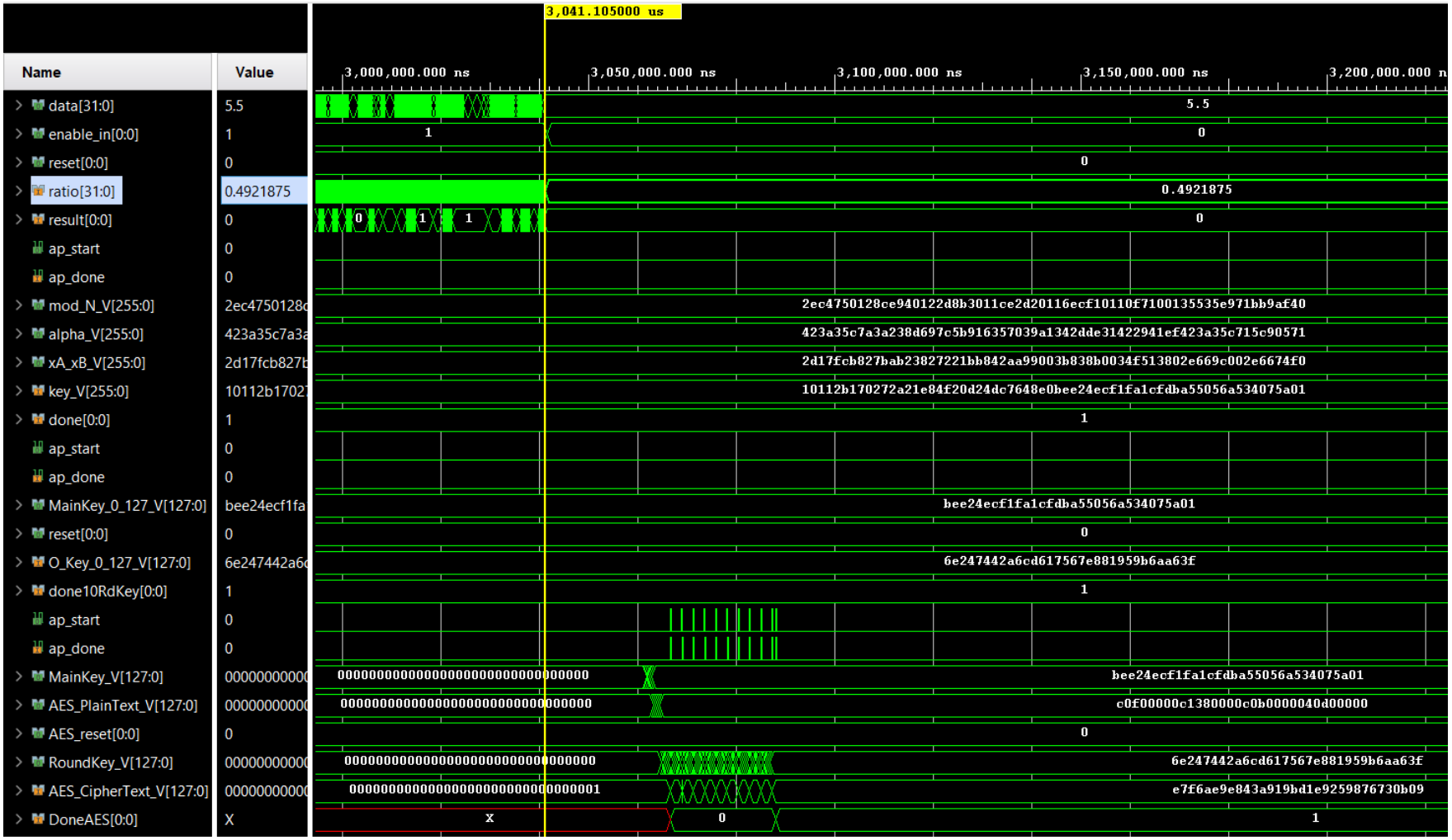
```
int main()
{
    const int data[] = {0xc0f00000,0xc1380000,0xc0b00000,0x40d00000,0x41780000,0x3f00

    struct DataArray mainKeys = KeyExchange();
    struct DataArrays roundKeys = RoundKeyExpansion(mainKeys);

    int ratio = NeuralNetwork(data);

    struct DataArray encrypted;
    struct DataArray decrypted;

    if(ratio > 0.3){ // kontrol %30

            struct DataArray image;
            for(int f=0;f<4;f++)
            {
                image.datas[f] = data[f];
            }

            encrypted = CryptographyEncryption(image, mainKeys, roundKeys);
            decrypted = CryptographyDecryption(encrypted, mainKeys, roundKeys);
    }

    return encrypted;
}
```

Şekil 8.65 : Sinir ağı ve kripto modülü kontrol kodu

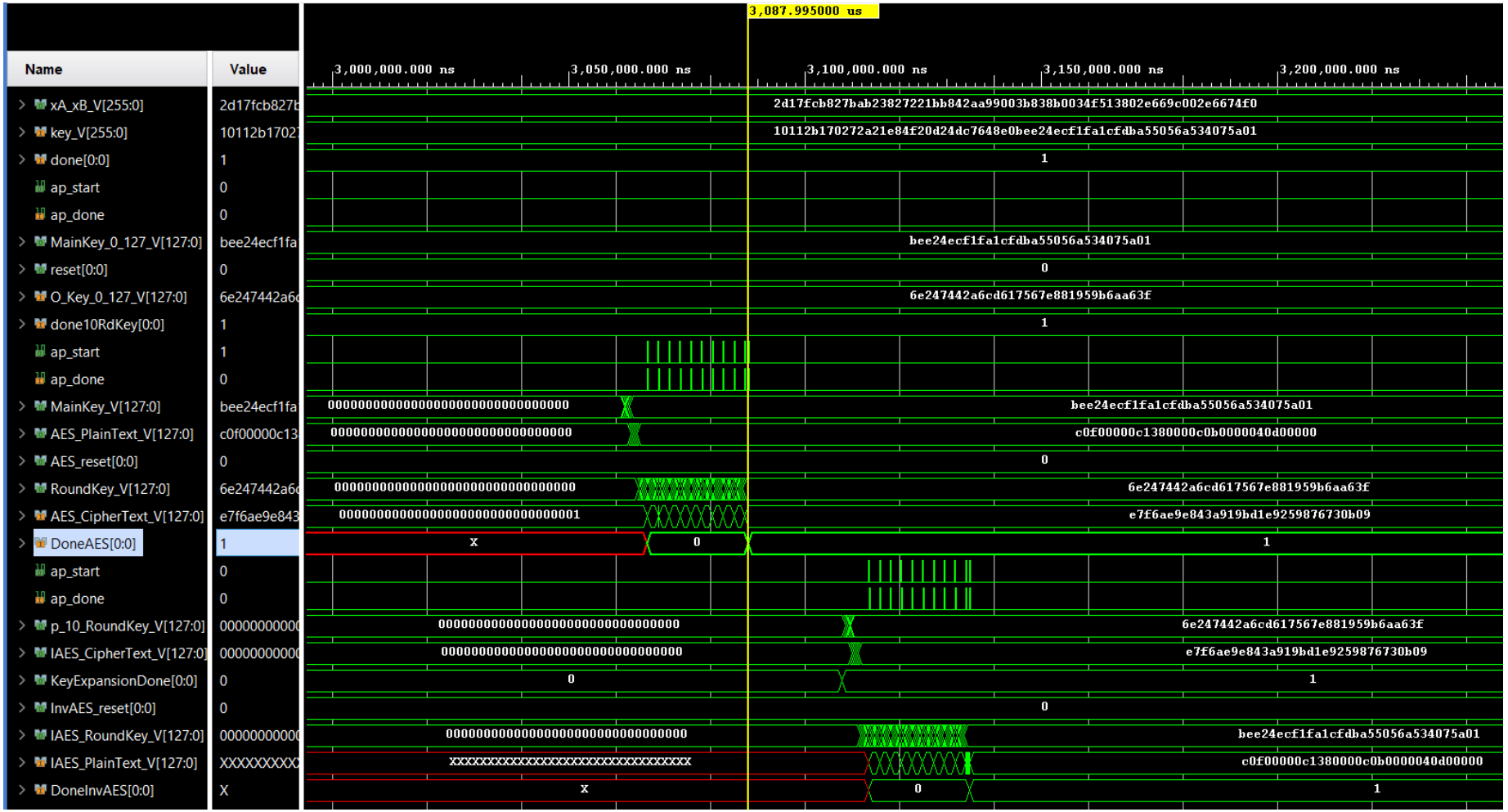
Anahtar değişim algoritmasının bittiği **done** sinyalinin ‘1’ vermesiyle anlaşılmaktadır. **done** sinyali ‘1’ olduktan sonra **Key_V** sinyalinden okunan değer bu algoritma sonucu oluşan 256 bitlik ortak anahtardır.(Şekil 8.66) **MainKey_0_127_V** sinyali AES’te kullanacağımız ana anahtar olarak belirlenmiştir. Şekil 8.67’de verilen simülasyonda sinir ağı işlemi tamamlanmış ve buradan alınan **ratio** sinyali resimde bulunan su oranını belirtmektedir. Bu değer belirlenen %30 değerinden fazla olduğu için veriler kripto bloğuna iletilmiştir. AES şifreleme bloğuna sinir ağından gelen 32 bitlik ilk 4 veri verilmiştir. Bu veriler **AES_PlainText_V** sinyalinden gözlemlenebilmektedir. Şifrelenen bu veriler şifre çözme bloğuna iletilmiştir. Şekil 8.68’de şifre çözme simülasyonu gösterilmiştir. Burada şifreleme bloğundan gelen şifrelenmiş verinin şifresi çözülür. **IAES_PlainText_V** sinyali ile sinir ağından gelen veriler ile karşılaştırıldığında şifreleme ve şifre çözme bloklarının istenen şekilde çalıştığı gözlemlenmiştir.



Şekil 8.66 : Anahtar değişim algoritması simülasyon sonucu ve ana anahtar elde edilmesi



Şekil 8.67 : Sinir ağı ve AES şifreleme simülasyon sonucu



Şekil 8.68 : AES şifreleme ve şifre çözme simülasyonu

9. GERÇEKÇI KISITLAR, SONUÇLAR VE ÖNERİLER

9.1 Gerçekçi Tasarım Kısıtları

Bu bölümde projenin teknik kısmının dışında kalan ekonomik, sosyal ve çevreye etki bakımından bilgiler verilecektir. Ayrıca tasarımın uyması gereken standartlar ve sağlık-güvenlik etkileri hakkında bilgiler verilecektir. Son kısımda ise projenin geliştirlmesi gereken yönleri ve bir sonuç kısmı verilerek bu kısım tamamlanacaktır.

9.1.1 Maliyet

Proje kapsamında kullanılacak MATLAB ve Vivado yazılımı öğrenci lisansına sahip olmamız sebebiyle maliyete dahil olmayacaktır.

Mühendis maaşı ortalama 8000 lira olarak alınırsa 1 yılda toplam 3 mühendisin çalışmasıyla 288.000 lira maaş gideri olacaktır.

Bunun dışında algoritmamızın çalıştırılacağı FPGA kartı alınması gerekmektedir. Burada sistem tasarlandıktan sonra gereksinimlere göre gereken kart belirlenerek satın alınmalıdır. Ortalama 200 dolar bir maliyet oluşacaktır. Toplamda 290.000 liralık bir maliyet oluşması beklenmektedir.

9.1.2 Standartlar

Bu proje askeri amaçla kullanılacak ise askeri yazılım standartlarına uyması gerekmektedir. Uzay araçlarında kullanılacak bir sistem tasarlanacak ise Ulusal Uzay protokollerine uygun bir tasarım yapılmalıdır. Bu tasarım yapılırken kullanılacak veriler açık kaynaklı paylaşılmıştır. Bunun dışında kullanılan yöntemler ise kendi sistemimize özgün olması gerektiğinden kopyalama yapılamayacaktır.

9.1.3 Sosyal, çevresel ve ekonomik etki

Bu sistemin benzeri bir sistem ülkemizde henüz bulunmamaktadır. Tasarlayacağımız sistem sonucunda ülkemiz, böylesine kritik bir teknolojiye sahip olmasıyla önemli bir

güce ulaşacaktır. Yerli ve milli olarak üretilmesiyle de ekonomik olarak bir gelir elde etmemiz yanında stratejik bir güce de ulaşacaktır.

Sistemin askeri amaçla kullanılması sırasında düşman unsurlarının engellemesiyle görev yapamaz hale getirilebilir ancak fiziksel güvenlik önlemleriyle de bu sorun aşılabilir.

9.1.4 Sağlık ve güvenlik riskleri

Sonuçta elde edilecek ürünün sağlık açısından olası herhangi bir etkisi yoktur. Burada sistemi kullanacak kişi ya da kurumların güvenlik ihtiyaçları oluşabilmektedir. Askeriye ve kamunun kullanımı için sistemin veri güvenliğinin en üst seviyede olması gerekmektedir. Oluşabilecek veri güvenliği problemleri ise sistem tasarlanırken gerçekleştirilen kriptolama işlemi sayesinde giderilecektir.

9.2 Sonuçlar

Bitirme projesi kapsamında yapılan SAR işaret işleme, yapay sinir ağı ve AES-anahtar değişimi gerçeklemeleri FPGA tabanlı olarak tasarlanmıştır. İlk olarak SAR işaret işlemeyi değerlendirecek olursak MATLAB'da Menzil Doppler algoritması tatmin edici seviyede bir görüntü elde edilmiştir. Burada algoritma doğrulaması yapıldıktan sonra FPGA gerçeklemesine geçilmiştir. Bunun için Model Composer programı ile model tabanlı olarak bir sistem tasarlanmıştır ancak tasarlanan sistem Vivado ortamına geçirildiğinde birtakım sorunlarla karşılaşılmıştır. Bunlar program kaynaklı olmakla birlikte bu sorunların programın yeni versiyonlarıyla birlikte ortadan kalkacağı düşünülmektedir. Kullanılan test verileri ile oluşturulan görüntüler sinir ağı alt sistemine iletilmiştir.

Yapay sinir ağı eğitiminde seçilmiş olan öz düzenlemeli ağ python ortamında eğitilmiştir. Eğitimi detaylandırmak için bazı eğitim verileri ile ikinci bir eğitim daha yapılmıştır. Eğitim sonucu yapılan testlerde öz düzenlemeli ağın sisteme uygun olduğu görüntüler üzerinde gözlemlenmiştir. Bunun üzerine test algoritması Model Composer ortamında gerçeklenerek FPGA'de Microblaze'le kontrol edilebilecek bir sisteme dönüştürülmüştür. Sistemin testleri Vivado ve Vitis ortamında yapılarak başarıyla sonuçlanmıştır. Yapay sinir ağı bloğu tek başına çalışabilir halde tasarlanmıştır.

Kriptoloji modülü iki ana bölümden oluşmuştur. Bu iki bölümden ilki AES şifreleme ve şifre çözme algoritmalarıdır. Bu algoritmalar hem kombinezonsal hem de ardışıl

şekilde tasarlanmıştır. Her iki tasarım da Model Composer testiyle doğrulanmıştır. Model Composer aracılığıyla üretilen IP'ler Vivado ortamında, Vitis'te yazılan kodlarla Microblaze kullanılarak test edilmiştir. Burada elde edilen sonuçlar da FIPS 197 dökümanıyla[11] karşılaştırılarak sistemin doğru çalıştığı gözlemlenmiştir. Kriptoloji modülünün diğer ayağı olan Diffie Hellman anahtar değişimi algoritması Model Composer ile ardışıl devre tasarımı gerçekleştirilmiş, test edilmiş ve doğrulanmıştır. AES şifreleme ve şifre çözme algoritmalarında olduğu gibi bu algoritmada da Vivado üzerinde sistem edilmiştir ve doğru çalıştığı gözlemlenmiştir. Kriptoloji bloğu tek başına çalışır hale gelmiştir.

Sistemin tüm blokları birbirine bağlanarak son hale getirilememiştir ancak SAR görüntü oluşturma bloğu IP'si Vivado'da çalıştırıldığında tüm sistem kolayca birbirine entegre edilebilir haldedir.

9.3 Geleceğe Yönelik Öneriler

Son aşamada algoritmanın Vivado doğrulaması ve sistem entegrasyonu yapılırken karşılaşılan Model Composer FFT işlemi ile Vivado FFT IP'sinin vektör ve skaler işlem yapma farklılıkları bir sorun oluşturmuştur. Bu sorun için farklı yollar denenebilir. İlk olarak Vivado'da FFT işlemi yapan donanımı tasarlayıp ayrı bir blok olarak eklemek bir öneri olabilir. Bir başka çözüm ise Model Composer ile üretilen FFT IP'sinin çıkışları üzerine bir üst modül ve bu modülün AXI Lite Slave arayüzü de eklenerek diğer modüllerle çalışması mümkün olacağı düşünülmektedir. Bu çözümler dışında farklı bir yapı da kullanılabilir ancak Model Composer kullanılacaksa bu iki önerinin tercih edilmesi doğru olacaktır. Aynı zamanda MATLAB'ın FFT işlemi için kullanılan bilgisayarda en fazla 256 boyutunda işlemler yapılabilmiştir. Daha yüksek kaynaklara sahip bilgisayarlarda daha yüksek boyutlu işlemler yapılabilir.

SAR görüntüleme algoritması için Menzil Doppler algoritması dışında algoritma gerçeklemeleri yapılarak görüntü kalitesi daha da iyileştirilebilir. Menzil Doppler algoritması içinse içinde kullanılan FFT blokları RTL seviyesinde Vivado'da paralel şekilde işlem yapacak şekilde tasarlanarak hızlandırılabilir.

Sinir ağında ise farklı algoritmalar denenerek veya Kohonen ağı algoritması geliştirilerek doğruluk oranı arttırılabilir.

Kriptoloji bloğunda tasarlanan bazı modüllerin alan, güç gibi karşılaştırılmaları yapılıp en optimum durum tasarlanmalıdır. Tasarlanacak olan modüller seçilecek FPGA ve Microblaze ile belirlenmelidir. Durumu göre değişebilecek olan ilk tasarım kombinezonsal AES şifreleme ve şifre çözme algoritması ile ardışıl AES şifreleme ve şifre çözme algoritmasıdır. Kombinezonsal AES şifreleme ve şifre çözme algoritması FPGA üzerinde daha fazla yer kaplarken Microblaze'e daha az iş gücü sağlar.Ardışıl AES şifreleme ve şifre çözme algoritması ise FPGA üzerinde daha az yer kaplarken Microblaze'e daha fazla iş gücü sağlar ve daha fazla hafıza ihtiyacı doğurur. İkinci tasarım ise AES şifre çözme algoritmasındaki sütun karıştırma işlemi tersi ile FIPS 197 dökümantasyonu [11] sütun karıştırma işlemi tersidir. Projede kullanılan sütun karıştırma işlemi tersi tasarımında sadece xor işlemi kullanılmıştır. Yapılacak olan karşılaştırmaya göre daha az yer kaplama durumunda bu tasarım diğer tasarımlar için de kullanılabilir.

Bir insansız hava aracı ve SAR RF donanımı tasarlanarak bu sistemler birleştirilebilir ve sahada kullanılabilir bir ürüne dönüştürülebilir.

KAYNAKLAR

[1] Nicolas, Jean-Marie & Le Hégarat-Mascle, Sylvie. (2008). Processing of Synthetic Aperture Radar Images, John Wiley & Sons, Ltd., sf. 1-55.

[2] Url-1<*https://www.boston.co.uk/info/nvidia-kepler/what-is-gpu-computing.aspx* >, erişim tarihi 25.05.2021.

[3] Url-2< *https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html*>**,** erişim tarihi 25.05.2021.

[4] G. F. Araujo, **R. d'Amore ve D. Fernandes.** (2018). *Cost-sensitive FPGA implementation of SAR range-doppler algorithm*. In IEEE Aerospace and Electronic Systems Magazine, Cilt. 33, No. 9, sf. 54-68, doi: 10.1109/MAES.2018.170120.

[5] R. Kędzierawski, W. Czarnecki, C. Leśnik ve J. Le Caillec (2011). MATLAB/simulink applications for SAR system design with FPGA, *2011 12th International Radar Symposium (IRS)*, Leipzig, pp. 35-40.

[6] Carrara W. G., Goodman R. S. ve Majewski R. M. (1995). Spotlight synthetic aperture radar: "Signal processing algorithms", Artech House, Norwood, MA.

[7] Dastgir, N., 2007. Processing SAR Data Using Range Doppler and Chirp Scaling Algorithms, *Master of Science Thesis*, School of Architecture and Built Environment, Royal Institute of Technology (KTH), Sweden.

[8] S. Haykin. (2009). "Learning Processes," in Neural networks and learning machines, Upper Saddle River, NJ: Prentice Hall, sf. 65–69.

[9] S. K. Pal & S. Mitra, (1992). "Multilayer perceptron, fuzzy sets, and classification," in IEEE Transactions on Neural Networks, sayı. 3, no. 5, sf. 683-697.

[10] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt and B. Scholkopf, (1998). "Support vector machines," in IEEE Intelligent Systems and their Applications, sayı. 13, no. 4, sf. 18-28.

[11] S. Albawi, T. A. Mohammed and S. Al-Zawi, (2017). "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), sf. 1-6.

[12] J. Li, C. Wang, S. Wang, H. Zhang ve B. Zhang, (2017). "Classification of very high resolution SAR image based on convolutional neural network," *2017 International Workshop on Remote Sensing with Intelligent Processing* (RSIP), Shanghai, sf. 1-4.

[13] Z. Ren, B. Hou, Z. Wen ve Jiao L. (2018). "Patch-Sorted Deep Feature Learning for High Resolution SAR Image Classification," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, Cilt. 11, No. 9, sf. 3113-3126.

[14] T. Kohonen. (1990). "The self-organizing map," in Proceedings of the IEEE, sayı. 78, no. 9, sf. 1464-1480.

[15] S. A. E. Rahman, (2015)**.** "Hyperspectral Imaging Classification Using ISODATA Algorithm: Big Data Challenge," 2015 Fifth International Conference on e-Learning (econf), sf. 247-250.

[16] M. Awad, K. Chehdi and A. Nasri, (2007). "Multicomponent Image Segmentation Using a Genetic Algorithm and Artificial Neural Network," in IEEE Geoscience and Remote Sensing Letters, sayı. 4, no. 4, sf. 571-575.

[17]E. Harison & N. Zaidenberg, (2018) "Survey of Cyber Threats in Air Traffic Control and Aircraft Communications Systems", Cyber Security: Power and Technology, Springer International Publishing, , ss. 199-217.

[18] Url-3<https://tr.wikipedia.org/wiki/Kriptoloji>, erişim tarihi 05.06.2021

[19] Diffie, W. and Hellman, M.E., 1976. New Directions in Cryptography. IEEE Transactions on Information Theory, sayı. 22, sf. 644-654.

[20] FIPS 197. (2001). Advanced Encryption Standard. National Institute of Standard and Technology, sf. 1

[21] Url-4< *https://www.mathworks.com/products/matlab/*>,erişim tarihi 29.05.2021.

[22] Xilinx. (2018). Model Composer User Guide (UG1262).

[23] Url-5<*https://tr.wikipedia.org/wiki/VHDL*>, erişim tarihi 04.06.2021.

[24] Xilinx. (2018). Model Based Design Using Model Composer(UG1259)

[25]Url-6<*https://en.wikipedia.org/wiki/Xilinx_Vivado* > , erişim tarihi 27.05.2021.

[26] Xilinx. (2020). Vitis Unified Software Platform Documentation User Guide (UG1400)

[27] Karabayır, O., 2010. Yapay Açıklık Radarı İşaret İşleme ve Doppler Merkez Frekansı Kestirimi, Yüksek Lisans Tezi, İTÜ Fen Bilimleri Enstitüsü, İstanbul.

[28] European Space Agency (ESA), Web sayfası: http://earth.esa.int

[29] Nicolas, J. M &Adragna F. (2008). *Processing of Synthetic Aperture Radar Images*. Hoboken, NJ.: Wiley.

[30] Carrara W. G., Goodman R. S. ve Majewski R. M. (1995). Spotlight synthetic aperture radar: "Signal processing algorithms", Artech House, Norwood, MA, sf. 13-18.

[31] Carrara W. G., Goodman R. S. ve Majewski R. M. (1995). Spotlight synthetic aperture radar: "Signal processing algorithms", Artech House, Norwood, MA, sf. 22-37.

[32] Naeim D., 2007. *Processing SAR datausing Range Doppler and Chirp ScalingAlgorithms*, Master's of Science Thesisin Geodesy Report No. 3096TRITA-GIT EX 07-005 School of Architecture and Built Environment Royal Institute of Technology (KTH), Stockholm, Sweden.

[33] Giorgio F., Riccardo L., 1999. *Synthetic Aperture Radar Processing*, pp. 30-70.

[34]Okunakul, H.B., 2006. Menzil-Doppler Ve Çırpı Tarama Algoritmaları Yardımıyla SAR Görüntü Oluşturulması, Yüksek Lisans Tezi, İTÜ Fen Bilimleri Enstitüsü, İstanbul.

[35]Deliormanlı,S., 2009. Yapay Açıklıklı Radar Ham Verilerinden Görüntü Oluşturulması, Yüksek Lisans Tezi, İTÜ Fen Bilimleri Enstitüsü, İstanbul.

[36] Curlender, J. and Mcdonough, R., 1991. *Synthetic Aperture Radar Systems and Signal Processing*, John Wiley & Sons, Inc., USA.

[37] Rahman, S. (2010). Focusing Moving Targets Using Range Migration Algorithm in Ultra Wideband Low Frequency Synthetic Aperture Radar. Blekinge Institute of Technology.

[38] Pritsker, D. (2015). Efficient global back-projection on an FPGA. In *Proceedings of the 2015 IEEE Radar Conference* (RadarCon), Cilt. 1, sf. 204–209.

[39]L. Pei, Y. Jin, T. Long and Y. Zhang, "DSP implementation of SPECAN algorithm," 2009 IET International Radar Conference, 2009, sf. 1-4.

[40] Raney, R.K., Runge, H., Bamler, R., Cumming, I. and Wong, F., 1994. Precision SAR Processing Using Chirp Scaling, IEEE Transactions on Geosciences and Remote Sensing, 32, 786-799.

[41] Akliouat, Hacene & Smara, Youcef & Bouchemakh, Lynda. (2007). Synthetıc Aperture Radar Image Formaton Process: Applıcatıon To A Regıon Of North Algerıa. Proc. 'Envisat Symposium 2007', Montreux

[42] Schlutz, M. (2009). Synthetic Aperture Radar Imaging Simulated in MATLAB, Yüksek lisans tezi, California Polytechnic State University, San Luis Obispo

[43] Sandwell D.T. (2002). Sar Image Formation: ERS Sar Processor Coded in MATLAB. Erişim tarihi Mayıs 5, 2021. Erişim adresi: https://engineering.purdue.edu/~bethel/sar_image_formation.pdf

[44] Url-7*https://upload.wikimedia.org/wikipedia/commons/1/1c/*>, erişim tarihi 25.05.2021.

[45] A. K. Jain, Jianchang Mao and K. M. Mohiuddin, "Artificial neural networks: a tutorial," in Computer, sayı. 29, no. 3, sf. 31-44.

[46] H. Yin and N. M. Allinson. (1995). "On the Distribution and Convergence of Feature Space in Self-Organizing Maps," in Neural Computation, sayı. 7, no. 6, sf. 1178-1187.

[47] Url-8<http://www.lohninger.com/helpcsuite/kohonen_network_-_background_information.htm >, erişim tarihi 01.06.2021.

[48] R. Huang, C. Cui, W. Sun and D. Towey, (2020)."Poster: Is Euclidean Distance the best Distance Measurement for Adaptive Random Testing?," 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), , sf. 406-40.

[49] H. Guo,(2011) "A Simple Algorithm for Fitting a Gaussian Function [DSP Tips and Tricks]," in IEEE Signal Processing Magazine, sayı. 28, no. 5, sf. 134-137.

[50] J. Vesanto and E. Alhoniemi, (2000)."Clustering of the self-organizing map," IEEE Transactions on Neural Networks, sayı. 11, no. 3, sf. 586-600.

[51] Url-7<*https://tr.wikipedia.org/wiki/Kriptografi*> erişim tarihi 02.06.2021.

[52] Büyükkaya E. (2017). Raspberry Pi üzerinde AES algoritmasına yan kanal analizi ve ölçüm iyileştirme. (Yüksek lisans tezi). İstanbul Şehir Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

[53]M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini and Y. Khamayseh, "Comprehensive study of symmetric key and asymmetric key encryption algorithms," *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1-7

[54] Ayushi. A (2010). A Symmetric Key Cryptographic Algorithm, International Journal of Computer Applications (0975 - 8887), pp. 2

[55] Hatun E., 2018. Raspberry Pi Üzerinde RSA Algoritmasına Yan Kanal Analizi .(Yüksek lisans tezi). İstanbul Teknik Üniversitesi.

[56] FIPS 46-3. (1999). Data Encryption Standard. National Institute of Standards and Technology (NIST).

[57]Url-8<*https://tr.wikipedia.org/wiki/Ayrık_logaritma*> erişim tarihi 02.06.2021.

[58]Yerlikaya T., Buluş E., Buluş N., Asimetrik Şifreleme Algoritmalarında Anahtar Değişim Sistemleri

[59] Levi A., Özcan M., 'Açık Anahtar Tabanlı Şifreleme Neden Zordur?' Türkiye Bilişim Konferansı 2002

[60] Aksuoğlu, A., (2010). RSA algoritmasının iyileştirilmesi için yeni bir yaklaşım. (Yüksek lisans tezi). Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Eskişehir.

[61] Url-9<https://www.nist.gov> erişim tarihi 04.06.2021.

[62] Şahin, F., (2015), "Modern blok şifreleme algoritmaları," İstanbul Aydın Üniversitesi Dergisi, no. 17, pp. 47-60.

[63] Daemen J., Rijmen V., AES submission document on Rijndael, June 1998.

[64]Url-10<https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development> erişim tarihi 03.06.2021.

[65] Ferguson, N., Schneier B., 2003: Practical Cryptography. Wiley Pulishing, Inc., Indianapolis, Indiana.

[66] Wadday A.G., Wadi S.M., Mohammed H.J., Abdullah A.A. "Study of WiMAX Based Communication Channel Effects on the Ciphered Image Using MAES Algorithm." International Journal of Applied Engineering Research,sayı.13, no.8, sf.6009- 6018, 2018.

[67] Doğan, A. Y., 2008: "AES Algoritmasının FPGA Üzerinde Düşük Güçlü Tasarımı", İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Türkiye.

[68] William, Stallings, and William Stallings. Cryptography and Network Security, 5/E. Pearson Education India, 2006

[69]Url-11<*https://xilinx.github.io/Vitis_Libraries/security/2019.2/guide_L1 /internals/iaes.html* >, erişim tarihi 01.06.2021.

[70] Bayam K.A, (2007). Differential Power Analysis Resistant Hardware Implementation of the RSA Cryptosytem, Yüksek Lisans Tezi, Istanbul

[71] Farooq, S.M.; Hussain, S.M.S.; Kiran, S.; Ustun, T.S. Certificate Based Authentication Mechanism for PMU Communication Networks Based on IEC 61850-90-5. Electronics 2018, 7, 370

[72] Montgomery, P.L., (1985). Modular Multiplication without Trial Division, Mathematics of Computation, 44, pp. 519-521.

[73] Menezes, A.J., Van Oorschot, P.C., and Vanstone, S.A., (1996). Handbook of Applied Cryptography, CRC Press.

[74] Batina, L., Örs, S.B., Preneel, B., and Vandewalle, J., 2003. Hardware Architectures for Public Key Cryptography. The VLSI Journal Integration, 34, pp. 1-64, Elsevier Science Publishers B. V.

[75]Url-12<*https://www.xilinx.com/products/designtools/vivado/simulator.html# waveform* > , erişim tarihi 27.05.2021.

[76]Url-13<*https://www.xilinx.com/content/dam/xilinx/imgs/block-diagrams/vivado-ip-integrator.png* > , erişim tarihi 27.05.2021.

[77] Url-15< *https://www.xilinx.com/support/answers/75527.html* > , erişim tarihi 27.04.2021.

[78] Url-14< *https://riscv.org/*> , erişim tarihi 27.04.2021.

[79] Url-16<*https://www.xilinx.com/products/design-tools/vivado/integration/sysgen. html* > , erişim tarihi 10.06.2021.

[80] S. M. Umar Talha, M. Asif, H. Hussain, A. Asghar ve H. Ameen (2016). "Efficient advance encryption standard (AES) implementation on FPGA using Xilinx system generator" *2016 6th International Conference on Intelligent and Advanced Systems* (ICIAS), Kuala Lumpur, sf. 1-6.

[81] M. A. Siddique, K. Aslam, A. Afroz, and F. H. Rizvi, "Implementation of Advanced Encryption Standard (AES) 192 Bit on FPGA", jictra, sf. 35-46, 2018.

[82] Ordu L.,(2006) "AES Algoritmasinin FPGA Uzerinde Gerceklenmesi ve Yan Kanal Analizi Saldirilarina Karsi Guclendirilmesi", Istanbul Teknik Universitesi, Fen Bilimleri Enstitusu.

Github: https://github.com/aydinfurkan/Graduation-Project

.

ÖZGEÇMİŞ-1

Ad-Soyad **:** Furkan CAN

Doğum Tarihi ve Yeri **:** 22.05.1998 - İstanbul

E-posta **:** canf16@itu.edu.tr

Lisans : İstanbul Teknik Üniversitesi –Elektronik ve Haberleşme Mühendisliği (2016-2021)

DENEYİM:

- 2020-2021 TÜBİTAK BİLGEM Bursiyerlik
- 2020 TÜBİTAK BİLGEM Stajı
- 2019 Akustik Algılama ve Görüntüleme Laboratuvar Stajı
- 2018 Alpplas AŞ ARGE Stajyeri

ÖZGEÇMİŞ-2

Ad-Soyad **:** Furkan AYDIN

Doğum Tarihi ve Yeri **:** 29.12.1997

E-posta **:** furkan139aydin@gmail.com

Lisans : İstanbul Teknik Üniversitesi –Elektronik ve Haberleşme Mühendisliği (2016-2021)

DENEYİM:

- 2020-2021 TESODEV Stajı
- 2019 Kafein Yazılım Stajı
- 2019 Huawei Stajı

ÖZGEÇMİŞ-3

Ad-Soyad **:** Fatma GÖK

Doğum Tarihi ve Yeri **:** 05.04.1998- Samsun

E-posta **:** ftmgk@windowslive.com

Lisans : İstanbul Teknik Üniversitesi –Elektronik ve Haberleşme Mühendisliği (2016-2021)

DENEYİM:

- 2020 Electra IC Stajı
- 2020 TÜBİTAK BİLGEM Stajı
- 2019 Huawei Stajı
- 2019 Akustik Algılama ve Görüntüleme Laboratuvar Stajı