İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

XİLİNX MODEL COMPOSER ARACINI KULLANARAK KIRMIK ÜSTÜ SİSTEM TASARIMI

LİSANS BİTİRME TASARIM PROJESİ

SALİM USLU

MUSTAFA MERT ESEN

İBRAHİM GÜVEN

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

MAYIS, 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

XİLİNX MODEL COMPOSER ARACINI KULLANARAK KIRMIK ÜSTÜ SİSTEM TASARIMI

LİSANS BİTİRME TASARIM PROJESİ

SALİM USLU
040130064

MUSTAFA MERT ESEN
040140099

İBRAHİM GÜVEN
040150070

Proje Danışmanı: Doç. Dr. S. Berna Örs Yalçın

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

MAYIS, 2019

İTÜ, Elektronik ve Haberleşme Mühendisliği Bölümü'nün ilgili Bitirme Tasarım Projesi yönergesine uygun olarak tamamen kendi çalışmamız sonucu hazırladığımız "MODEL COMPOSER ARACINI KULLANARAK KIRMIK ÜSTÜ SİSTEM TASARIMI" başlıklı Bitirme Tasarım Projesi'ni sunmaktayız. Bu çalışmayı intihal olmaksızın hazırladığımızı taahhüt eder; intihal olması durumunda bitirme tasarım projesinin başarısız sayılacağını kabul ederiz.

Salim USLU .....
(040130064)

Mustafa Mert ESEN .....
(040140099)

İbrahim GÜVEN .....
(040150070)

Proje Danışmanı : Doç. Dr. S. Berna ÖRS YALÇIN

.....

ÖNSÖZ

Bitirme tasarım proje çalışmamız boyunca her konuda anlayışlı olan ve yardımını hiçbir zaman esirgemeyen Berna Hocamıza ve projemizde bilgi ve tecrübeleriyle bize yardımcı olan Mehmet Akif Özkan'a sonsuz teşekkür eder, en içten saygılarımızı sunarız.

Mayıs 2019

Salim Uslu

Mustafa Mert Esen

İbrahim Güven

İÇİNDEKİLER

Sayfa

KISALTMALAR

AES **:** Advanced Encrytion Standard

AMDF **:** Average Magnitude Difference Fucntion

AXI **:** Advanced Extensible Interface

DES **:** Data Encryption Standard

FIPS **:** Federal Information Processing Standars

FPGA **:** Field Programmable Gate Array

GF **:** 4.2.1

GSM **:** Global System for Mobile Communications

HLS **:** High Level Synthesis

Hz **:** hertz

IEEE **:** The Institute of Electrical and Electronics Engineers

IP **:** Intellectual Property

kbps **:** kilo bit per second

kHz **:** kilohertz

LPC **:** Lineer Predictive Coding

MEMS **:** Micro Electro MEchanical Systems

MHz **:** megahertz

ms **:** milisecond

PCM **:** Pulse Code Modualtion

PDM **:** Pulse Density Modulated

PWM **:** Pulse Width Modulated

RF **:** Radio-Frequency

SPN **:** Substitution Permutation Network

VHDL **:** Very High –Speed Integrated Circuit Hardware Description Language

LSF **:** Linear Spectral Frequencies

ŞEKİL LİSTESİ

Sayfa

XİLİNX MODEL COMPOSER ARACINI KULLANARAK KIRMIK ÜSTÜ SİSTEM TASARIMI

ÖZET

Model Composer, Xilinx tarafından MathWorks Simulink [2] ortamına entegre edilmiş, model temelli tasarıma ve tasarımın anlık simülasyonuna imkan sağlayan ayrıca tasarımı Xilinx FPGA'leri üzerinde gerçeklenebilir hale getiren bir araçtır. Model Composer'da, görüntü işleme ,mantıksal işlemler ve bit manipülasyonları gibi işlemleri gerçekleştiren hazır bloklar bulunmaktadır ve aynı zamanda Simulink ortamında bu bloklarla oluşturulan model temelli tasarımların simülasyonu yapılabilmektedir. Ayrıca Model Composer kütüphanesinde kullanıma hazır olarak bulunan bloklar dışında kullanıcı tarafından C/C++ programlama dilleri ile tanımlanmış fonksiyonlar özelleştirilmiş blok olarak Model Composer kütüphanesine eklenebilmektedir.

Bu proje kapsamında Model Composer ortamında bir ses şifreleme algoritmasına ait donanımın tasarlanması hedeflenmiştir. Bu hedef doğrultusunda Model Composer üstünde Federal Bilgi İşleme Standartları'na (Federal Information Processing Standards – FIPS) uygun olarak bir Gelişmiş Şifreleme Standardı (Advanced Encryption Standard - AES) algoritması gerçeklenmiştir. Şifrelenerek bir nevi gürültü görünümüne sahip olan ses sinyalinin mobil haberleşme kanalı ile bozulmadan iletilebilmesi amacıyla bir Doğrusal Öngörü ile Kodlama (Linear Predictive Coding – LPC) [12] algoritmasının da proje dahilinde Model Composer ortamında gerçeklenmesi hedeflenmiştir. Tasarlanan LPC algoritması sayesinde, AES ile şifrelendikten sonra gürültü görünümüne sahip olan ve herhangi bir şekilde anlamlandırılması mümkün olmayan ses sinyalinin paketler halinde LPC kod kitapçıklarına karşılık gelen anlamsız insan seslerine dönüştürülerek mobil haberleşme kanallarında kullanılan gürültü filtrelerinde bozulmadan iletilebilecektir. Şifrelemeyle benzer şekilde, şifre çözme işlemlerinin de Model Composer ortamında yapılan tasarım ile gerçekleştirilmesi amaçlanmış ve yine simülasyonları da bu ortamda yapılmıştır.

Proje kapsamında yapılacak tasarımların DIGILENT Nexys4 DDR FPGA kartı üzerinde gerçeklenmesi hedeflenmiştir. Nexys4 DDR üzerinde bulunan MEMS mikrofon yardımıyla ortamdan alınan analog ses sinyali örneklenerek sayısal veriye dönüştürülmüş ve tasarlanan şifreleme donanımı ile şifrelenmiştir. Aynı şekilde şifrelenmiş ses sinyali üzerinde şifre çözme işlemi yapılmıştır. Alınan ses sinyali şifreleme ve şifre çözme işlemlerine tabi tutulduktan sonra yine kart üzerinde bulunan ses çıkış portu ile analog bir ses sinyali olarak dış ortama aktarılmıştır.

Seçilen algoritmalar, yapılan tasarımlar ve kullanılacak FPGA kartı ışığında tüm proje bir bütün olarak değerlendirilecek olursa, proje konu itibariyle ses şifrelemesi yapan bir modülün tasarımıdır. Bu bağlamda ortamdan alınan analog ses verisi FPGA kartı mikrofonunda sayısallaştırılacak, AES ile şifrelenecek, şifreleme sonucu oluşan ve gürültüden ayırt edilebilmesi zor olan şifrelenmiş ses sinyali veri paketleri halinde LPC bloğundan geçerek anlamsız insan seslerine dönüştürülmesi sağlanacaktır. Böylelikle insan sesi özellikleri taşıyan şifrelenmiş ses sinyali mobil haberleşme kanalı üzerinden filtrelerde bozulmadan güvenli bir şekilde alıcıya iletilecektir. Alıcı tarafına ulaşan şifrelenmiş ses verisi için aynı işlemler tersi yönünde uygulanarak güvenli bir haberleşme imkanı sağlanacaktır. Tasarım aracı olarak kullanılan Model Composer,

ses şifreleme için kullanılacak blokların kütüphanesinde yetersiz olmasından dolayı, projenin donanım ortamına aktarılmasına imkan vermemiş olsa da proje bu alanda yapılacak tasarımlar için bir referans olacaktır.

SYSTEM ON CHIP DESIGN USING XILINX MODEL COMPOSER

SUMMARY

Model Composer is a tool, which works on MathWorks Simulink environment and it enables model based design to be implemented on Xilinx FPGA boards. Xilinx Model Composer library includes blocks that can be used in image processing, logical operations and bit manipulations and model based design simulations can be realized using these blocks. Additionally, model based designs can be realized using user defined blocks that can be imported to Model Composer library using functions that are written in C/C++ programming languages.

The goal of this project is to design a hardware that can encrypt sound signals, using Model Composer. Therefore, following the instructions of Federal Information Processing Standards (FIPS), Advanced Encryption Standard (AES) algorithm is designed using Model Composer. To transmit the encrypted data using wireless channel, Linear Predictive Coding (LPC) algorithm is also realized using Model Composer. The benefit of using LPC is that the data which is encrypted by AES is converted to a suitable state for wireless transmission and transmitted in packets without deformation. Encryption as well as decryption processes are aimed to be designed using Model Composer.

The FPGA board on which the designs are implemented, is selected as Nexys4 DDR, because it has a MEMS microphone that can be used to sample analog sound signals. Sampled sound signals are then encrypted and decrypted with AES and both after encryption and decryption processes, the processed sound signal is outputted through the onboard mono sound output port as an analog sound signal.

Considering the chosen algorithms, designs and implementation on FPGA as a whole, this is a project that samples and encrypts sound signal, converts said sound signal data to human voice for lossless wireless transmission, transforms human voice to encrypted sound data and then decrypts sound data to find the original sampled data and outputs through the onboard mono sound output port. The lack of necessary design blocks in Model Composer library prevented the hardware implementation of the design, but this project can be used as a reference for future projects.

1. GİRİŞ

Bu proje kapsamında Model Composer ortamında bir ses şifreleme algoritmasına ait donanımın tasarlanması hedeflenmiştir. Bu hedef doğrultusunda Model Composer üstünde Federal Bilgi İşleme Standartları'na (Federal Information Processing Standards – FIPS) uygun olarak bir Gelişmiş Şifreleme Standardı (Advanced Encryption Standard - AES) [6] algoritması gerçeklenmiştir. Şifrelenerek bir nevi gürültü görünümüne sahip olan ses sinyalinin mobil haberleşme kanalı ile bozulmadan iletilebilmesi amacıyla bir Doğrusal Öngörü ile Kodlama (Linear Predictive Coding – LPC) [12] algoritmasının da proje dahilinde Model Composer ortamında gerçeklenmesi hedeflenmiştir. Tasarlanan LPC algoritması sayesinde, AES ile şifrelendikten sonra gürültü görünümüne sahip olan ve herhangi bir şekilde anlamlandırılması mümkün olmayan ses sinyalinin paketler halinde LPC kod kitapçıklarına karşılık gelen anlamsız insan seslerine dönüştürülerek mobil haberleşme kanallarında kullanılan gürültü filtrelerinde bozulmadan iletilebilecektir. Şifrelemeyle benzer şekilde, şifre çözme işlemlerinin de Model Composer ortamında yapılan tasarım ile gerçekleştirilmesi amaçlanmış ve yine simülasyonları da bu ortamda yapılmıştır.

Proje kapsamında yapılacak tasarımların DIGILENT Nexys4 DDR FPGA kartı üzerinde gerçeklenmesi hedeflenmiştir. Nexys4 DDR üzerinde bulunan MEMS mikrofon yardımıyla ortamdan alınan analog ses sinyali örneklenerek sayısal veriye dönüştürülmüş ve tasarlanan şifreleme donanımı ile şifrelenmiştir. Aynı şekilde şifrelenmiş ses sinyali üzerinde şifre çözme işlemi yapılmıştır. Alınan ses sinyali şifreleme ve şifre çözme işlemlerine tabi tutulduktan sonra yine kart üzerinde bulunan ses çıkış portu ile analog bir ses sinyali olarak dış ortama aktarılmıştır.

Seçilen algoritmalar, yapılan tasarımlar ve kullanılacak FPGA kartı ışığında tüm proje bir bütün olarak değerlendirilecek olursa, proje konu itibariyle ses şifrelemesi yapan bir modülün tasarımıdır. Bu bağlamda ortamdan alınan analog ses verisi FPGA kartı mikrofonunda sayısallaştırılacak, AES ile şifrelenecek, şifreleme sonucu oluşan ve gürültüden ayırt edilebilmesi zor olan şifrelenmiş ses sinyali veri paketleri halinde

LPC bloğundan geçerek anlamsız insan seslerine dönüştürülmesi sağlanacaktır. Böylelikle insan sesi özellikleri taşıyan şifrelenmiş ses sinyali mobil haberleşme kanalı üzerinden filtrelerde bozulmadan güvenli bir şekilde alıcıya iletilecektir. Alıcı tarafına ulaşan şifrelenmiş ses verisi için aynı işlemler tersi yönünde uygulanarak güvenli bir haberleşme imkanı sağlanacaktır. Tasarım aracı olarak kullanılan Model Composer, ses şifreleme için kullanılacak blokların kütüphanesinde yetersiz olmasından dolayı, projenin donanım ortamına aktarılmasına imkan vermemiş olsa da proje bu alanda yapılacak tasarımlar için bir referans olacaktır.

1.1 Rapor ile İlgili Açıklamalar

Model Composer, Matlab Simulink ortamında çalışarak model temelli tasarımların Xilinx programlanabilir cihazları için otomatik kod üretimine olanak sağlayarak tasarım sürecini hızlandırır [1]. Simulink ortamı görsel arayüzü ile model temelli tasarıma olanak sağlamakta ve sinyallerin giriş çıkış değerlerini gösterirken tasarımın genel durumu hakkında bilgi vermektedir [2].

Bu projede Xilinx Model Composer kullanılarak, aracın donanım tasarımı alanındaki etkinliğinin incelenmesi amaçlanmaktadır.

1.2 Proje Önerisinde Önerilen Çalışma Planı ve Muhtemel Değişiklikler

Projede ilk olarak Model Composer ortamının tanınması planlanmıştır. Bu bağlamda kullanılacak örnek projeler incelenerek uygulaması yapılacak ve üretilen donanım çıktıları kontrol edilecektir. Bu esnada kazanılan kullanım becerileri ışığında temel projeler üretilerek aracın etkinliğinin kontrol edilmesi ve ortamın sunduğu imkanların keşfedilmesi amaçlanmıştır. Araç hakkında belirli bir tecrübe kazanıldıktan sonra, Model Composer aracılığı ile tasarlanacak asıl proje için konu seçilmesi planlanmıştır. Konu seçildikten sonra Model Composer aracı ile gerekli donanımın tasarlanması ve Alanda Programlanabilir Kapı Dizini (Field Programmable Gate Array – FPGA) üzerinde gerçeklenmesi düşünülmüştür.

1.3 Proje Önerisinde Önerilen Çalışma Planı Gerçeklenme Düzeyi

Kullanılacak olan Model Composer'ın yeni bir araç olması nedeniyle bu projede başlangıç olarak Xilinx tarafından sağlanan eğitim dökümanları incelenmiştir. Bu

bağlamda dökümanların içerdiği örnek projeler uygulanmıştır. Bunun yanında tarafımızca ortamın daha iyi anlaşılması ve araca ilişkin kullanım becerilerinin geliştirilmesi amacıyla kombinezonsal ve ardışıl tipte tasarım yapılmış ve çıktılar kontrol edilmiştir.

Öngörülen çalışma planında, Model Composer tarafından üretilen kodun kontrolü kombinezonsal devre tasarımlarından önce gelmektedir, fakat Model Composer dökümanlarında Xilinx tarafından hazırlanmış tasarım örnekleri ile üretilen kodun karmaşıklığı nedeniyle, kontrol aşaması kombinezonsal devre tasarımdan sonraya alınmıştır.

Konu olarak seçilen ses şifrelemesi yapan donanımın model composer ortamında gerçeklenmesi ve gerçeklenecek olan donanımın vivado ortamına aktarılması adımları henüz tamamlanmamıştır.

1.4 Literatür Araştırması

Model Composer, Xilinx tarafından 2018'in ilk yarısında tasarımdan donanıma geçerken oluşan aksaklıkları gidermek amacıyla Simulink ortamına bir eklenti olarak tasarlanmıştır [1] Yeni bir araç olması nedeniyle literatürde bu araç üzerine bir çalışma bulunmadığından, sadece geliştirici tarafından sunulan kaynaklar üzerinde inceleme yapılabilmiştir. Bu bağlamda "*Model-Based Design Using Model Composer*" [3] ve "*Model Composer User Guide*" [1] dökümanları, araç hakkında yapılan incelemenin temel kaynakları olmuştur. Bahsedilen kaynaklar araç hakkında temel bilgilerin edinilmesini ve örnek projeler ile aracın kullanılmasına ilişkin becerilerin geliştirilmesini sağlamıştır.

LPC algoritması temelinde analog ses verisini çözümlemek için kullanılır. Kısaca anlık ses bilgisini bulmak için önceki değerleri kullanmaktır. İlk olarak 1984 yılında Federal Standart 1015 standartı içinde, United States Department of Defence tarafından yayımlanmıştır. LPC insanda vokal sistemin bir matematiksel karşılığıdır ve Mobil Haberleşme için Küresel Sistem(Global System for Mobile – GSM) hattı üzerinde dijital olarak ses iletimi yapılamayacağından LPC algoritması ile analog ses verisini işlemeyi gerektirir [12].

1.5 Proje Önerisinde Belirtilen Aşamaların Gerçekleşme Düzeyi

Bu projenin yapılması için izlenecek adımlar proje planında şu şekilde belirlenmiştir:

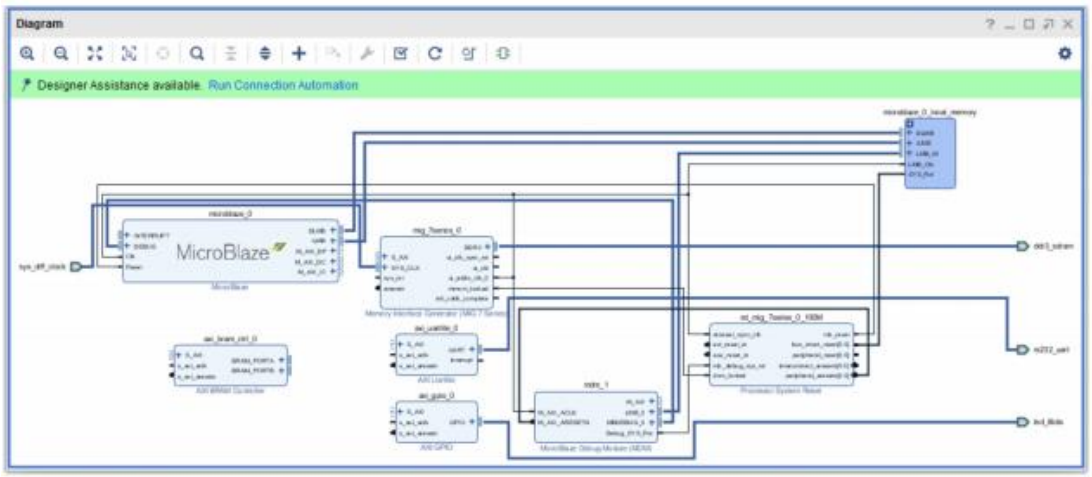
- Model Composer ve Simulink ortamının, üreticinin sağladığı yardımcı dökümanlar ile tanınması,
- Model Composer tarafından tasarlanan devrenin etkinliğinin kontrol edilmesi,
- Model Composer tarafından üretilen donanım tanımlama dili kodunun anlaşılabilirliğinin ve doğruluğunun kontrol edilmesi,
- Model Composer ve Simulink ortamlarının daha iyi anlaşılması için basit kombinezonsal devre tasarımları yapılması,
- Model Composer ve Simulink ortamlarının daha iyi anlaşılması için basit ardışıl devre tasarımları yapılması,
- Üretilen kırmık üstü sistem üstünde çalışan programların yazılması,
- Uygulama alanı ve gerçeklenecek algoritmanın seçilmesi,
- Seçilen algoritmanın Simulink ortamında tasarlanması, test edilmesi ve geliştirilmesi,
- Simulink'te hazırlanmış projenin Model Composer ile donanım tanımlama dillerine aktarılması,
- Model Composer ile üretilen otomatik kodların Vivado aracı ile FPGA üzerinde gerçeklenmesi,
- Vivado aracı ile mikroişlemci ve otomatik üretilen donanımlardan oluşan kırmık üstü sistemin gerçeklenmesi

2. KULLANILAN ARAÇLAR

2.1 Xilinx Vivado Tasarım Ortamı

Xilinx tarafından oluşturulan Vivado Tasarım Ortamı (Vivado Design Suite) uygulaması yapılacak tasarımın en başından FPGA üzerinde gerçeklenene kadar uygulanacak bütün adımların herhangi bir ara dosya formatı oluşturmadan tek bir uygulama üzerinde yapılmasına imkân sağlar [23]. Bu da çalışma süresinin azaltılmasını ve hata ayıklama işleminin daha etkin bir biçimde yapılmasına olanak sağlar.

Microblaze işlemcileri kullanırak oluşturulacak kırmık üstü sistemler için oluşturulmuş özelleştirilmiş Zihni Mülkiyetler (Intellectual Property – IP) Gelişmiş Genişletilebilir Arayüzüne (Advenced Extensible Interface – AXI) uygun olarak tasarlanmalıdır [24]. Microblaze işlemcisi tasarımda birlikte kullanılan IP'leri AXI üzerinden kontrol etmektedir.



Şekil 2.1 : Microblaze Sistemi[24]

2.2 Model Composer

Bu bölümde, Xilinx Model Composer'ın sunduğu faydalardan kısaca bahsedilecek, araç ile sunulan hazır kütüphane hakkında bilgi verilecek ve aracın kullanımının pekiştirilmesi için gerçekleştirilmiş çalışmalar örnekleriyle gösterilecektir.

2.2.1 Giriş

Xilinx tarafından yayınlanan kullanıcı klavuzuna göre Model Composer, Xilinx programlanabilir cihazları için Mathworks Simulink ortamında otomatik kod üretimi yaparak üretime giden yolu hızlandıran model tabanlı bir tasarım aracıdır[1]. Model Composer kullanılarak yapılan tasarımların, Vivado Tasarım Ortamı'nda Microblaze işlemcileri ile kullanılmasına, Model Composer ile üretilen IPlerin AXI uyumlu olması izin verir.

Matlab üzerine entegre edilmiş bir sistem olan Simulink, modelleme için grafiksel ortam sağlayan bir araçtır. Bunun yanında simülasyon, analiz ve tasarımların sistem düzeyinde doğrulanmasını sağlar.

Model Composer sunduğu kütüphaneler sayesinde simulink ortamında medş temelli tasarım yapılmasına olanak sağlar. Bununla birlikte kullanıcı tarafından blok oluşturulması da mümkündür. Tasarımcı tarafından C/C++ dillerinde yazılan kodlar da Model Composer'a özelleştirilmiş bloklar olarak aktarılabilir. Bu blokların kullanılmasıyla gerçeklenen algoritmalar Vivado Yüksek Seviye Sentez (High-Level Synthesis- HLS) ortamının otomatik optimizasyon ve üst düzey sentez teknolojisi ile ürün kalitesinde Zihni Mülkiyet'e (Intellectual Property-IP) dönüştürülebilir. Böylece üretilen IP kırmık üstü sistem tasarımı projelerinde kullanılabilecek nitelikte olacaktır.

Model Composer, Simulink üzerinde çalışan, optimize edilmiş bloklar içeren bir kütüphanedir. Bu küpüthane içerik olarak;

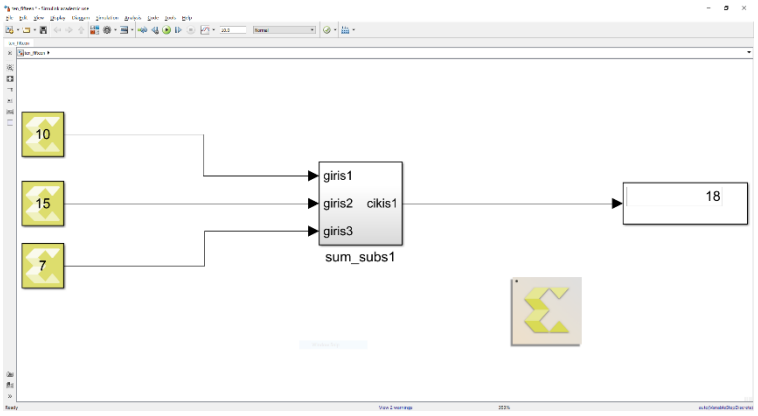
- Computer Vision: Sayısallaştırılmış görüntü için analiz, optimizasyon ve manipülasyon yapan bloklar,
- Logic and Bit Operations: Bileşik mantıksal işlemleri ve bit bazında işlemleri destekleyen bloklar,
- Lookup Table: Giriş indeksine göre temel davranışın kullanıcı tarafından sağlanabileceği bloklar,
- Math Functions: Matametiksel fonksiyonlar içeren bloklar,
- Ports and Subsystems: Alt sistem ve giriş çıkış port işlemlerini gerçekleştirmeye yarayan bloklar,
- Relational Operations: Girişler arasında karşılaştırma yapılmasına olanak sağlayan bloklar,

- Signal Attiributes: Bloklar arasında giriş ve çıkışların data tipleri ve boyutları olarak uyumlu hale getirilmesine olanak sağlayan bloklar,
- Signal Operations: sinyalin zaman değişkeni üzerinde basit işlemler ile yeni sinyal üretimi yapmayı sağlayan bloklar,
- Signal Routing: Sinyal kaynaklarını ve sinyal yollarını izleyen bus seçici gibi bloklar,
- Sinks: Blokların fiziksel sinyal çıktılarını gösteren bloklar,
- Source: Sinyal üreten bloklar,
- Tools: Modelin implementasyon ve arayüzünü kontrol eden

bloklar içermektedir.

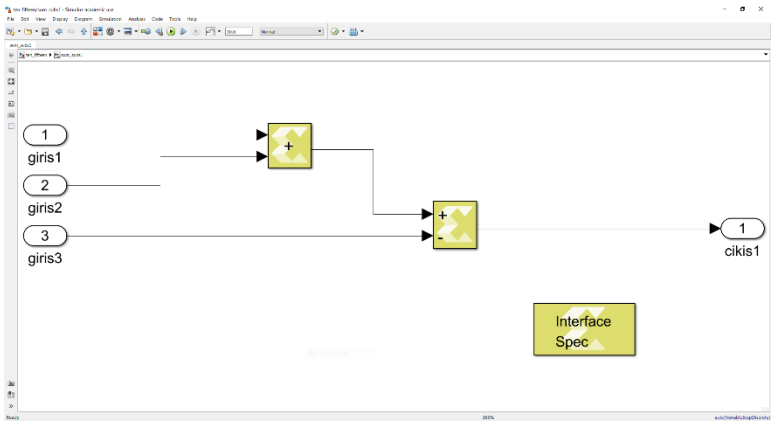
2.2.2 Kombinezonsal Devre Tasarımı

Mantık devrelerini kombinezonsal ve ardışıl devreler olarak ikiye ayırmak mümkündür. Kombinezonsal devreler, çıkış değerlerini girişlerin sadece güncel değerlerini kullanarak belirleyen devrelerdir [4]. Bu bağlamda, Model Composer kullanılarak basit kombinezonsal tasarımlar yapılması amaçlanmıştır. Yapılan bir kombinezonsal devre tasarımı aşağıda verilmiştir.



Şekil 2.2 : Basit kombinezonsal devrenin tepe modülü.

Bu tasarım, giris1 ve giris2'yi toplar ve giris3'ü bu toplamdan çıkararak sonucu verir. Bu tasarımdaki "sum_subs1" alt modülünün iç yapısı verilmiştir (Şekil 2.2).



Şekil 2.3 : Basit kombinezonsal devrenin iç yapısı.

2.2.3 Kontrol Aşaması

Model Composer kullanılarak tasarlanan basit kombinezonsal devreye ait, Model Composer tarafından üretilen, Verilog [25] kodlarının, blok tasarımı ile aynı işlemi yapıp yapmadığı incelenmek istenmektedir.

Bu aşamada, Şekil 2.1'de verilen kombinezonsal devreye ait, Model Composer tarafından üretilen Verilog kodları incelenmiştir. Üretilen Verilog kodlarının, AXI uyumlu olduğu (Şekil 2.4) ve tasarım ile aynı şekilde çalıştığı görülmektedir (Şekil 2.5).

```
module sum_subs1_AXILiteS_s_axi
#(parameter
    C_S_AXI_ADDR_WIDTH = 6,
    C_S_AXI_DATA_WIDTH = 32
)(
    // axi4 lite slave signals
    input  wire                          ACLK,
    input  wire                          ARESET,
    input  wire                          ACLK_EN,
    input  wire [C_S_AXI_ADDR_WIDTH-1:0] AWADDR,
    input  wire                          AWVALID,
    output wire                          AWREADY,
    input  wire [C_S_AXI_DATA_WIDTH-1:0] WDATA,
    input  wire [C_S_AXI_DATA_WIDTH/8-1:0] WSTRB,
    input  wire                          WVALID,
    output wire                          WREADY,
    output wire [1:0]                    BRESP,
    output wire                          BVALID,
    input  wire                          BREADY,
    input  wire [C_S_AXI_ADDR_WIDTH-1:0] ARADDR,
    input  wire                          ARVALID,
    output wire                          ARREADY,
    output wire [C_S_AXI_DATA_WIDTH-1:0] RDATA,
    output wire [1:0]                    RRESP,
    output wire                          RVALID,
    input  wire                          RREADY,
    output wire                          interrupt,
    // user signals
    output wire                          ap_start,
    input  wire                          ap_done,
    input  wire                          ap_ready,
    input  wire                          ap_idle,
    output wire [7:0]                    giris1,
    output wire [7:0]                    giris2,
    output wire [7:0]                    giris3,
    input  wire [7:0]                    cikis1,
    input  wire                          cikis1_ap_vld
);
```

Şekil 2.4 : Model Composer tarafından üretilen, AXI uyumlu modülün giriş ve çıkışlar

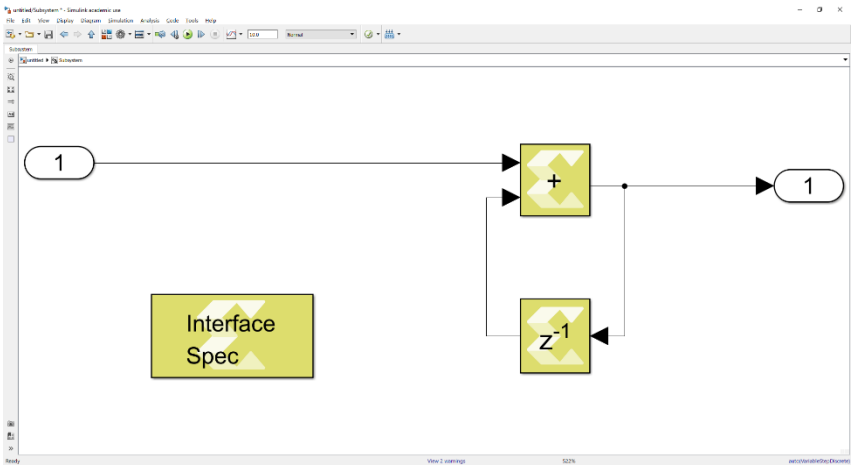
```
assign Sum_out1_fu_51_p2 = (giris2 + giris1);

assign ap_done = ap_start;

assign ap_idle = 1'b1;

assign ap_ready = ap_start;

always @ (*) begin
    ap_rst_n_inv = ~ap_rst_n;
end

assign cikis1 = (Sum_out1_fu_51_p2 - giris3);
```

Şekil 2.5 : Model Composer tarafından üretilen kodun ilgili kısmı

2.2.4 Ardışıl Devre Tasarımı

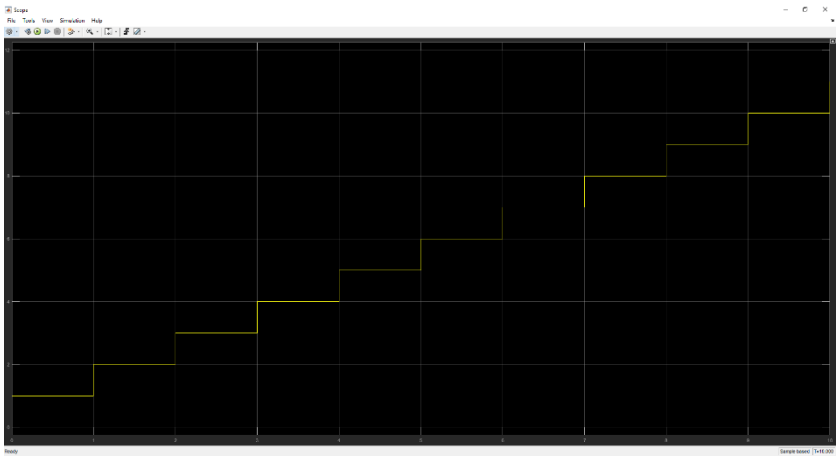
Bazı sayısal sistemlerde, çıkış değerlerinin bulunması için girişlerin güncel değerlerinin ve çıkışların eski değerlerinin, geri beslemeden faydalanılarak, kullanılması gereklidir. Kombinezonsal devreleri kullanmak, bu noktada mümkün değildir [5]. Ardışıl devrelerin Model Composer kullanılarak gerçeklenebilirliği incelenmesi amaçlanmıştır.

Model Composer ile tasarım anlayışını geliştirmek amacıyla bu aşamada, geri besleme ile çalışan basit bir asenkron ardışıl devre tasarlanmıştır (Şekil 2.6).



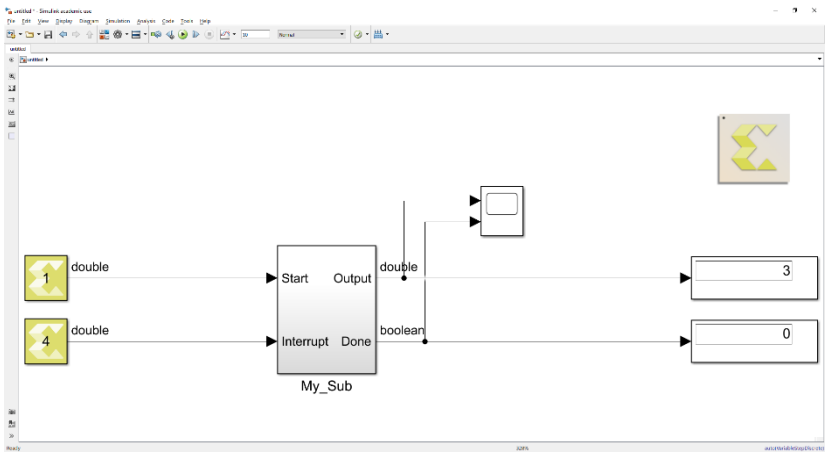
Şekil 2.6 : Geri besleme ile çalışan basit ardışıl devre.

Bu tasarımda, giriş değeri, bir gecikme ile kendisiyle toplanmaktadır. İlk giriş değeri olarak “1” verildiğinde, çıkış değerinin her saat döngüsündeki değeri, ilk on saat döngüsü için Şekil 2.7’te verilmiştir.



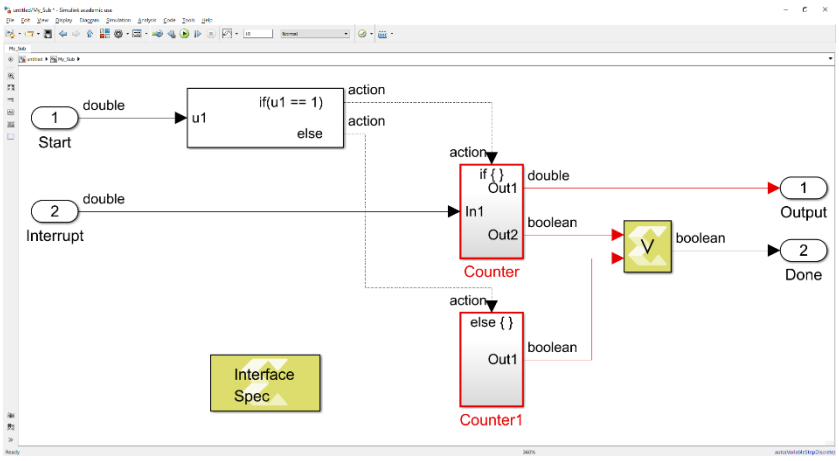
Şekil 2.7 : Geri besleme ile çalışan basit ardışıl devreye ait çıkışın, zamanla değişen değerleri

Sonraki aşamada, Şekil 2.6'te verilen asenkron ardışıl devre geliştirilerek, belirli bir değere kadar sayan ardışıl bir devre tasarlanmıştır (Şekil 2.8). Bu devrede "Output", "Start" sinyali "1" değerini aldığı sürece, "Interrupt" değerine kadar artarak sayar ve bu değere ulaşıldığında "Done" sinyali "1" değerini alır ve bir sonraki saat döngüsünde "Output" eski değerine gelirken "Done", "0" değerini alır.

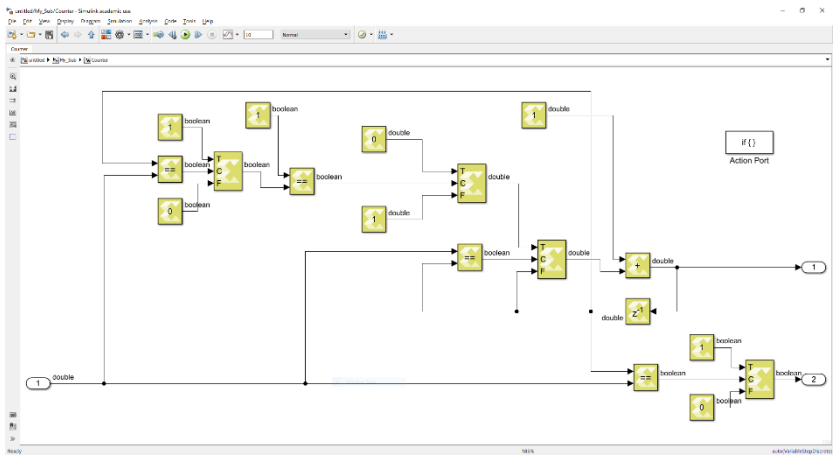


Şekil 2.8 : Ardışıl sayıcı devresinin tepe modülü.

Şekil 2.8'da verilen tasarımın iç yapısı Şekil 2.9 ve Şekil 2.10'da verilmiştir.



Şekil 2.9 : My_Sub modülünün iç yapısı.



Şekil 2.10 : Counter modülünün iç yapısı.

3. GELİŞMİŞ ŞİFRELEME STANDARDI

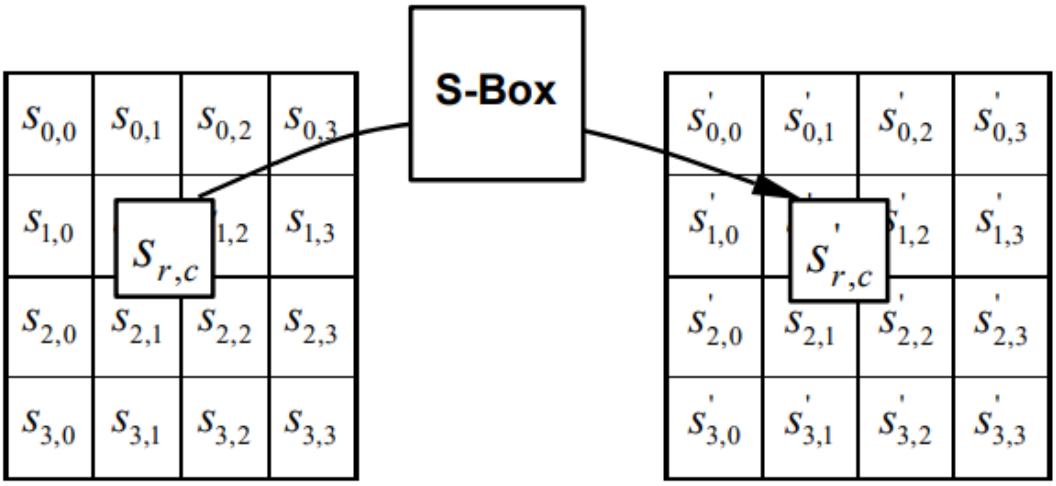
3.1 Giriş

AES, 128, 192 ve 256 bitlik üç farklı boyuttaki giriş, çıkış, durum ve anahtar kullanılarak, şifreleme ve şifre çözme işlemlerini simetrik olarak gerçekleştirebilen bir tasarımdır [6]. Kullanılan anahtar boyutu büyüdükçe AES algoritması daha yavaş çalışmaktadır. Bu nedenle 128 bitlik anahtar kullanılarak en hızlı tasarım elde edilmektedir.AES, bitleri her turda değiştiren Yer Değiştirme-Permütasyon Ağı (Substitution- Permutation Network) mekanizması ile gerçeklenir. AES'in bu şekilde tasarlanması, şifre çözme işlemi sırasında yapılan her dönüşümün, şifrelemede karşılık düşen dönüşümün tersi olmasına olanak sağlar. Bu da şifrelemede yapılan her dönüşümü ters sırada yaparak şifre çözmeye olanak sağlar. Bu mekanizmanın her turunda şifreleme anahtarından anahtar genişletme yöntemiyle türetilen tur anahtarı, iki boyutluk bir diziye karşılık gelmektedir [7]. Tur anahtarı, güncel durumla XOR işlemine sokulur ve sonuç, yer değitirme tablosundan karşılık gelen değerle değiştirilir. Bu işlem son tura ulaşılıp çıkış elde edilene kadar devam eder.

AES içerisindeki dönüşümler, durum adı verilen 4x4 boyutundaki bir matrise uygulanır. Şifreleme ve şifre çözme işlemlerinin ilk turunda, giriş değeri duruma kopyalanır [6].

3.2 128 Bit Gelişmiş Şifreleme Standardı Şifrelemesinin Özellikleri

AES, 128 bit boyutundaki giriş, çıkış, durum ve anahtarı, her elemanı 8 bit veya 1 byte olan 4x4 matris olarak işlemektedir. Şifrelemenin ilk turuna geçmeden önce, anahtar ve giriş değeri exorlanmaktadır [6]. Şifrelemenin ilk turundan onuncu turuna kadar SubByte, ShiftRows ve MixColumns dönüşümleri yapılırken; son turda MixColumn dönüşümü es geçilir ve çıkış değeri, durumun o anki değeri ile son turdaki anahtarın exorlanması ile elde edilir. Çıkış da her elemanı 8 bit olan 4x4 bir matrisdir.



Şekil 3.1 : SubByte dönüşümü [6]

3.2.1 SubByte dönüşümü

SubByte, S-box adındaki tersi alınabilir bir yer değiştirme tablosunu kullanan, lineer olmayan bir byte değiştime dönüşümüdür. S-box, iki adet dönüşüm içermektedir [6].

- {00} elemanının kendisi ile eşleşmesi için sonlu GF(2^8) alanında çarpımsal ters işlemi gerçekleştirilir.
- Aşağıdaki dönüşüm GF(2) üstünde uygulanır:

$$bi' = bi \oplus b(i+4)mod8 \oplus b(i+5)mod8 \oplus b(i+6)mod8 \oplus b(i+7)mod8 \oplus ci \quad (3.1)$$

i, 0 ve 8 arasında değer alırken; bi, byteın i numaralı bitini ve ci, c değeri {63} iken c'nin i numaralı bitini temsil etmektedir. bi' byte değerinin i numaralı yeni değerini göstermektedir.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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

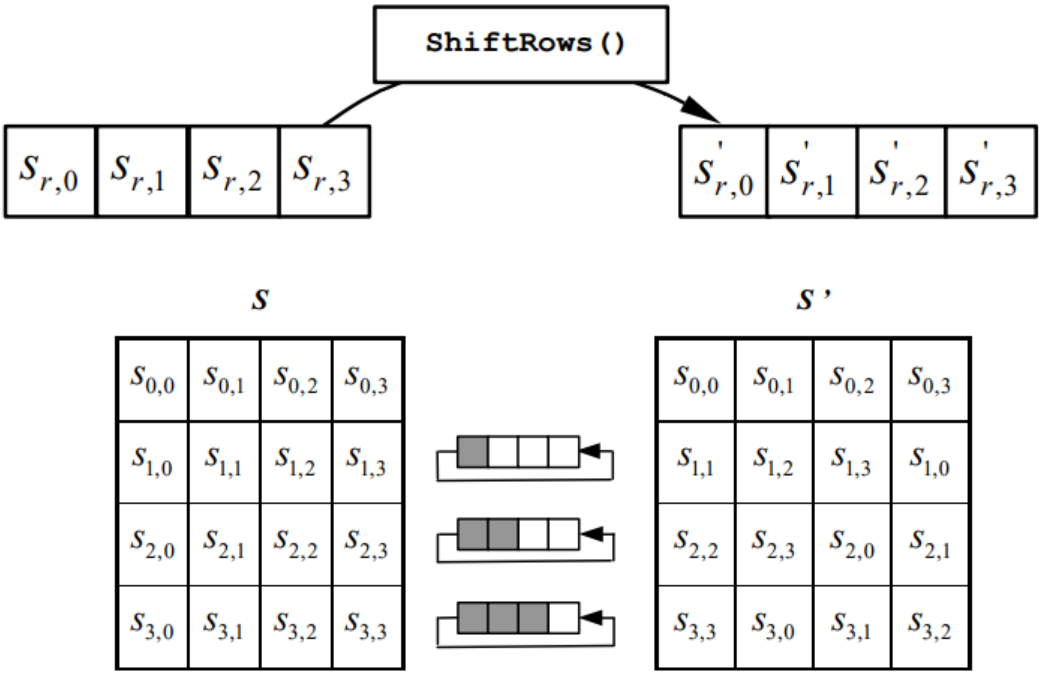
Şekil 3.2 : S-box tablosunun hexadecimal değerleri [6]

3.2.2 ShiftRows dönüşümü

ShiftRows, durum matrisinin ilk satırı dışındaki satırları farklı değerlerde döngüsel olarak kaydırır [6]. Kaydırma işlemi yapılmayan tek satır ilk satırdır. ShiftRows dönüşümüne ait bağıntı su şekildedir:

$$Sr, c' = Sr, (c + kaydır(r, Nb) mod Nb) \quad (3.2)$$

kaydır(r, Nb) fonksiyonu, durum matrisinde aynı satırda bulunan elemanların sütun değeri en yüksek olan ile en düşük olanın yerini değiştirmektedir.



Şekil 3.3 : ShiftRows dönüşümü [6]

3.2.3 MixColumns dönüşümü

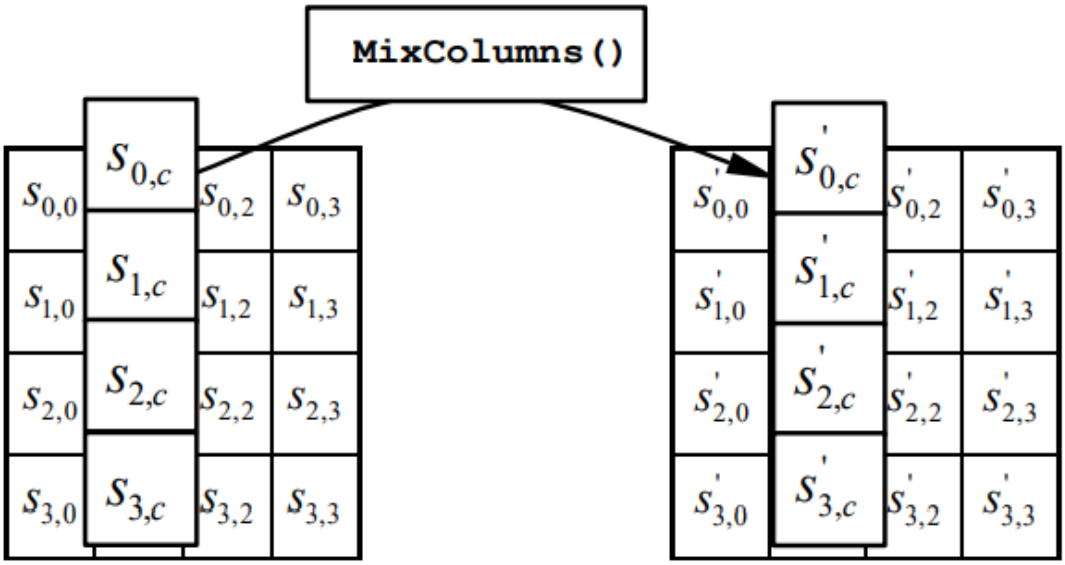
MixColumns dönüşümünde, durum matrisine ait her sütun bir polinom olarak ele alınır ve GF(2^8) üzerinde polinom olarak incelenir; $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ olarak tanımlanmış a(x) polinomu ile çarpılır. Yeni sütun değerleri şu şekilde bulunmaktadır [6]:

$$s0,c' = (\{02\} \bullet s0,c) \oplus (\{03\} \bullet s1,c) \oplus s2,c \oplus s3,c$$

$$s1,c' = s0,c \oplus (\{02\} \bullet s1,c) \oplus (\{03\} \bullet s2,c) \oplus s3,c$$

$$s2,c' = s0,c \oplus s1,c \oplus (\{02\} \bullet s2,c) \oplus (\{03\} \bullet s3,c)$$

$$s3,c' = (\{03\} \bullet s0,c) \oplus \bullet s1,c \oplus s2,c \oplus (\{03\} \bullet s3,c) \qquad (3.3)$$

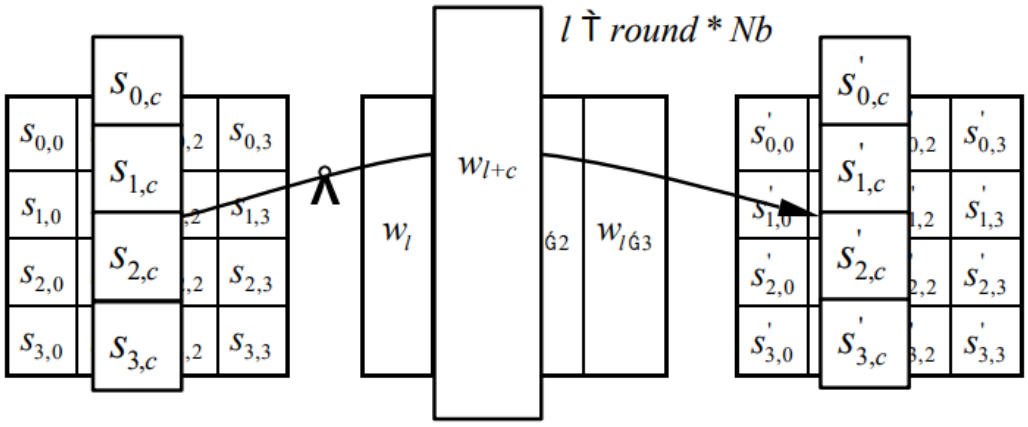


Şekil 3.4 : MixColumns dönüşümü [6]

3.2.4 AddRoundKey dönüşümü

Bu aşamada, o anki turda üretilen anahtar matrisi ile durum matrisinin sütunları ayrı ayrı exorlanır ve yeni durum matrisi elde edilir [6]. Bu dönüşümün bağıntısı şu şekildedir:

$$[s0, c', s1, c', s2, c', s3, c'] = [s0, c, s1, c, s2, c, s3, c] \oplus [wround * Nb + c] \quad (3.4)$$



Şekil 3.5 : AddRoundKey dönüşümü [6]

3.3 Anahtar Genişletilmesi

Anahtar genişletme işleminde, şifreleme anahtarı kullanılarak anahtar listesi üretilmektedir [6]. AES 128 için 4x4 matrislerden oluşan, şifreleme anahtarının da

dahil olduğu onbir adet anahtar bu listede saklanmaktadır. Bu aşamada, daha önce bahsedilen S-box yapısı, SubWord() fonksiyonu içerisinde kullanılmaktadır. RotWord() fonksiyonu [a_0, a_1, a_2, a_3] olarak aldığı girişi, [a_1, a_2, a_3, a_0] olarak çıkışa vermektedir. Rcon yapısı, 32 bit boyutunda kelime içeren bir rom yapısıdır.

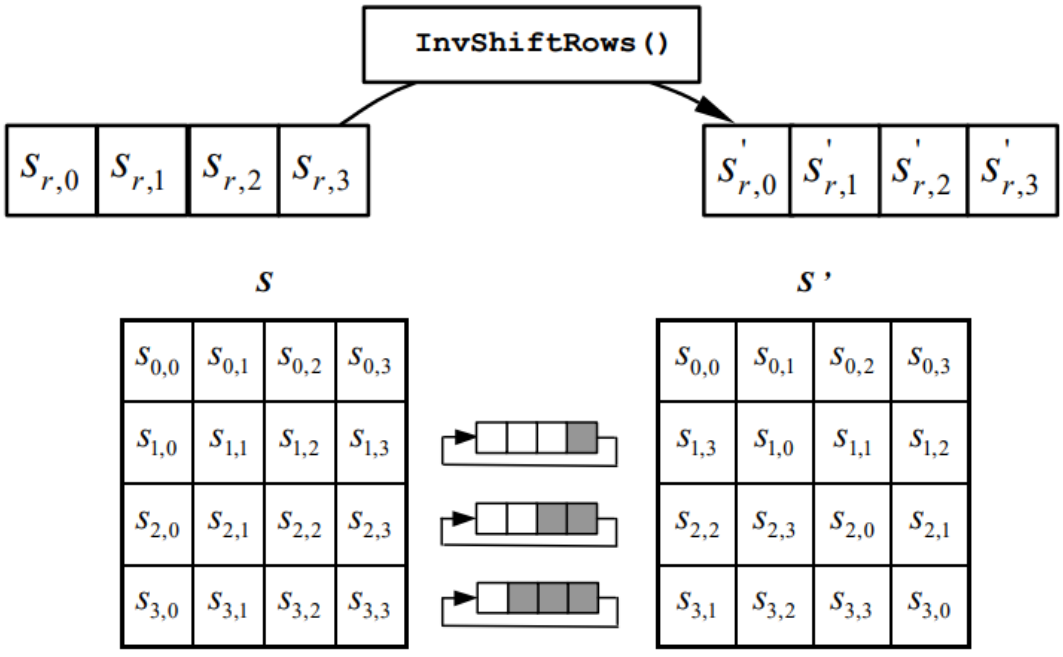
3.4 128 bit AES şifre çözmenin özellikleri

Şifreleme işleminde yapılan dönüşümler tersi alınabilir olduğu için, şifre çözme işlemi sırasında yapılan dönüşümler, şifreleme işlemi sırasında yapılan dönüşümlerin ters sırada uygulanmış hali olacaktır [6]. Şifre çözme işlemi sırasında InvShiftRows, InvSubBytes, InvMıxColums ve AddRoundKey dönüşümleri yapılmaktadır. Şifreleme işleminin çıkışı, şifre çözme modulünün girişi olarak kabul edilmektedir ve durum matrisine aktarılmadan önce tura ait anahtar ile exorlanmaktadır. Durum matrisine on tur boyunca InvShiftRows, InvSubBytes, InvMıxColums ve AddRoundKey dönüşümleri uygulanmaktadır ve sonuncu turda sadece InvMıxColums dönüşümü uygulanmamaktadır.

3.4.1 InvShiftRows dönüşümü

ShiftRows dönüşümün tersidir ve dönüşüme ait bağıntı şu şekildedir [6]:

$$Sr, (c + kaydır(r, Nb) mod Nb) = Sr, c' \quad (3.5)$$



Şekil 3.6 : InvShiftRows dönüşümü [6]

3.4.2 InvSubBytes dönüşümü

InvSubBytes dönüşümü SubBytes dönüşümün tersidir ve S-box yer değiştirme matrisinin tersini kullanarak durum matrisini dönüştürür [6].

		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 3.7 : Inverse S-box tablosu hexadecimal değerleri [6]

3.4.3 InvMixColumns dönüşümü

MixColumns dönüşümün tersidir ve durum matrisinin her bir sütununu dört elemanlı bir polinom olarak GF(2^8) üzerinde değerlendirilir ve $a^{-1}(x)$ = $\{0b\}x^3+\{0d\}x^2+\{09\}x+\{0e\}$ polinomuyla çarpılır [6].

Bu dönüşümün sonucu, şu bağıntıda verilmiştir.

$$s0,c' = (\{0e\} \bullet s0,c) \oplus (\{0b\} \bullet s1,c) \oplus (\{0d\} \bullet s2,c) \oplus (\{09\} \bullet s3,c)$$

$$s1,c' = (\{09\} \bullet s0,c) \oplus (\{0e\} \bullet s1,c) \oplus (\{0b\} \bullet s2,c) \oplus (\{0d\} \bullet s3,c)$$

$$s2,c' = (\{0d\} \bullet s0,c) \oplus (\{09\} \bullet s1,c) \oplus (\{0e\} \bullet s2,c) \oplus (\{0b\} \bullet s3,c)$$

$$s3,c' = (\{0b\} \bullet s0,c) \oplus (\{0d\} \bullet s1,c) \oplus (\{09\} \bullet s2,c) \oplus (\{0e\} \bullet s3,c) \quad (3.6)$$

4. DOĞRUSAL ÖNGÖRÜ YÖNTEMİ İLE SESİN KODLANMASI

Sesin sıkıştırılması ya da sesin kodlanması, konuşma sinyalinin uygulama alanına göre daha az parametre ile ifade edilmesidir. Bu yöntemler her ne kadar kayıplı olsalarda insan kulağı bu kayıpları algılayabilecek yeterlilikte olmadıklarından ses sıkıştırılması ve kodlanması işlemlerindeki bu kayıplar kabul edilebilir kayıplardır. Sıkıştırılan ses içinde sesli kısım kodlanır, sessiz kısım kodlanmaz bu sayede ses için kullanılacak parametre ve bit sayısında düşüş sağlanır.

Haberleşme teknolojisinin en temel problemlerinden biri ise bant genişliğinin verimli kullanılması ve sesin sıkıtıştırılmasıdır [9]. Bant genişliğinin daha fazla olduğu ve hâlen artmakta olan, daha hızlı iletim sağlayan veri bağlantılarının (data links) aksine ses iletişimi (speech communication) telekomünikasyonda en baskın ve yaygın hizmettir. Uzun yıllardır çalışılan konuşmanın kodlanması ve sıkıştırılması konusu son 20 ila 30 yıl içinde öneminin arttığı gözlemlenmiştir.

Konuşmanın sıkıştırılması alanındaki çalışmaların başlangıcı olarak kabul edilen Darbe Kod Modülasyonu (Pulse Code Modulation - PCM) 1930'lu yıllarda hayatımıza girmiştir [10]. Daha sonra artan maliyetler sonucu 1970'li yıllarda Doğrusal Öngörü ile Kodlama (Linear Predictive Coding - LPC) yöntemi geliştirilmeye başlandı ve 1984 yılında Federal Standart 1015, Birleşik Devletler Savunma Bakalınlığı (United States Department of Defence) tarafından LPC algoritması standaştırılarak yayınlandı.

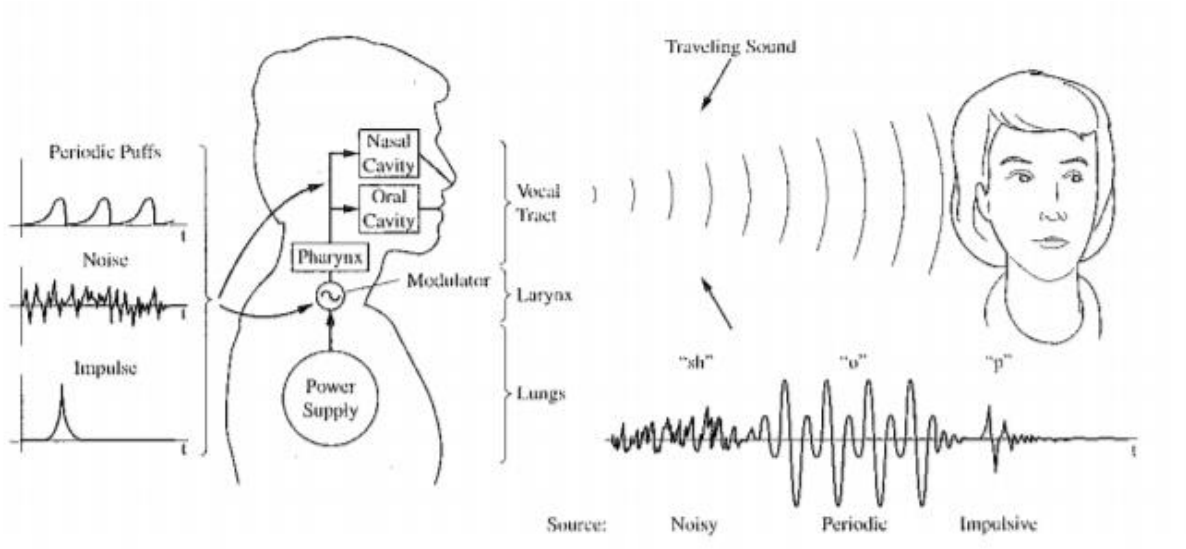
Konuşma kodlayıcılar konuşmanın kodlanmasında ve sıkıştırılmasında kullanılan sistemlerdir [11]. Bu sistemler darbe biçimi tabanlı ve model parametreleri tabanlı olarak ikiye ayrılırlar. LPC ise model parametreleri tabanlı bir ses kodlama metodudur. Bu metodda konuşma, parametreler cinsinden ifade edilip tekrar sentezlendiğinden konuşmanın tam olarak elde edilebilmesi mümkün değildir. Bu metod sisteme karmaşıklık getirse bile, darbe biçimi tabanlı konuşma kodlayıcılarının aksine daha iyi bir sıkıştırma oranı vermektedir.

Konuşmanın sıkıştırılmasında en önemli etkenlerden biri bit oranıdır. Bu oran sıkıştırılacak konuşmadaki sıkıştıma miktarı hakkında bilgi verir. 8 kHz ile örneklenen konuşmalar 8 bit ile ifade edilirse, konuşma 64 kbps (kilo bit per second) bit rate olarak ifade edilebilir. 8 kHz de 64 kbps'den daha düşük bit rate'e sahip konuşmalar sıkıştırılmış olarak ifade edilir [12]. LPC, 2.4 kbps bit rate oranına kadar konuşmaya sıkıştırma yapabilen bir sistemdir. Konuşmanın sıkıştırılması konusunda günümüze

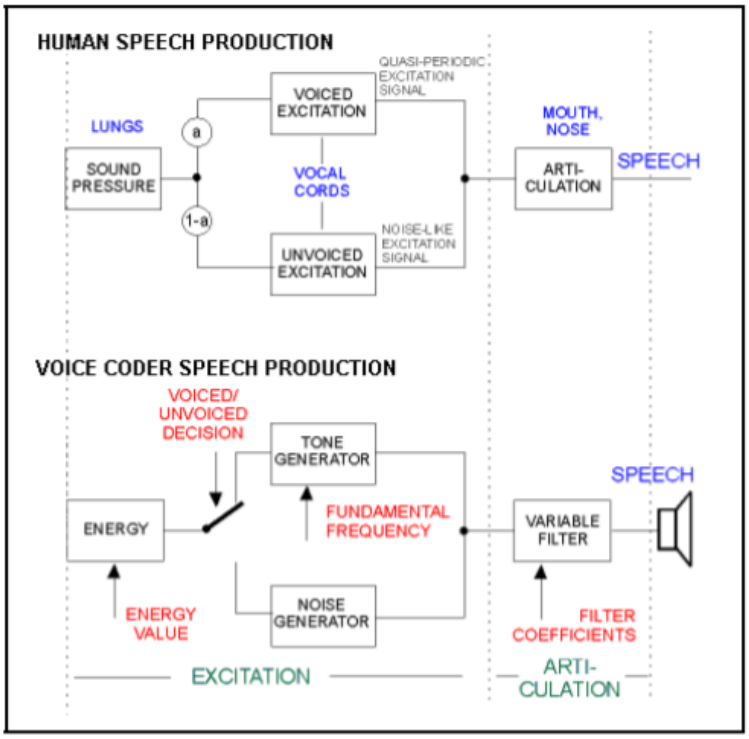
kadar bir çok çalışma yapılmıştır. Bu çalışmaların çoğunda konuşmanın frekans aralığının (300 Hz -3400 Hz) sınırlı olması ve konuşmanın yavaş anatomik hareketlerden oluşması üzerinde durulmuştur.

4.1 İnsanlarda Konuşmanın Modellenmesi

İnsanlarda ses, akciğerlerden gelen hava gırtlakta bulunan ses tellerini titreştirir. Ses telleri, telli müzik aletlerindeki gibi titreştiğinde akciğerlerden gelen havanın şiddetiyle birlikte insan sesinin çıkmasını sağlar [13]. Akciğerlerden gelen havanın şiddeti sesin genlik değerini belirleyip sesin ne kadar gürültülü olduğunu analiz etmemizi sağlar. Havanın izlediği yollar vokal sistem olarak adlandırılır. Vokal sistem, boğaz (gırtlak, soluk borusu), burun ve ağızdan oluşur. Ses tellerinin titreşim farklılıklarına bağlı olarak da üretilen seslerin farklılığını sağlar. Sesli konuşma matematiksel olarak incelendiğinde konuşmanın periyodik olduğu gözlemlenir. Sessiz konuşma ise rastgele olduğu gözlemlenir. İnsan sistemi modellendiğinde akciğer, ses kaynağı, vokal sistem sesin farklılığını, genliğini oluşturan filtre olarak tanımlanır.



Şekil 4.1 : İnsanda ses modeli [14]



Şekil 4.2 : Ses modeli ile LPC yapısı karşılaştırılması[12]

Şekil 4.2'de de görüldüğü üzere insanda konuşmanın modellenmesi ile LPC'nin modellenmesi hemen hemen aynı modelle anlatılabilir. Akciğerler kaynak olarak, ses telleri ise sesli ve sessiz duruma göre ton üretici veya gürültü üretici olarak kullanılan sistem olarak modellenmiştir. Ağız ve burun, uyarılma süzgeci olarak modellenmiştir.

4.2 Doğrusal Öngörü Kodlaması

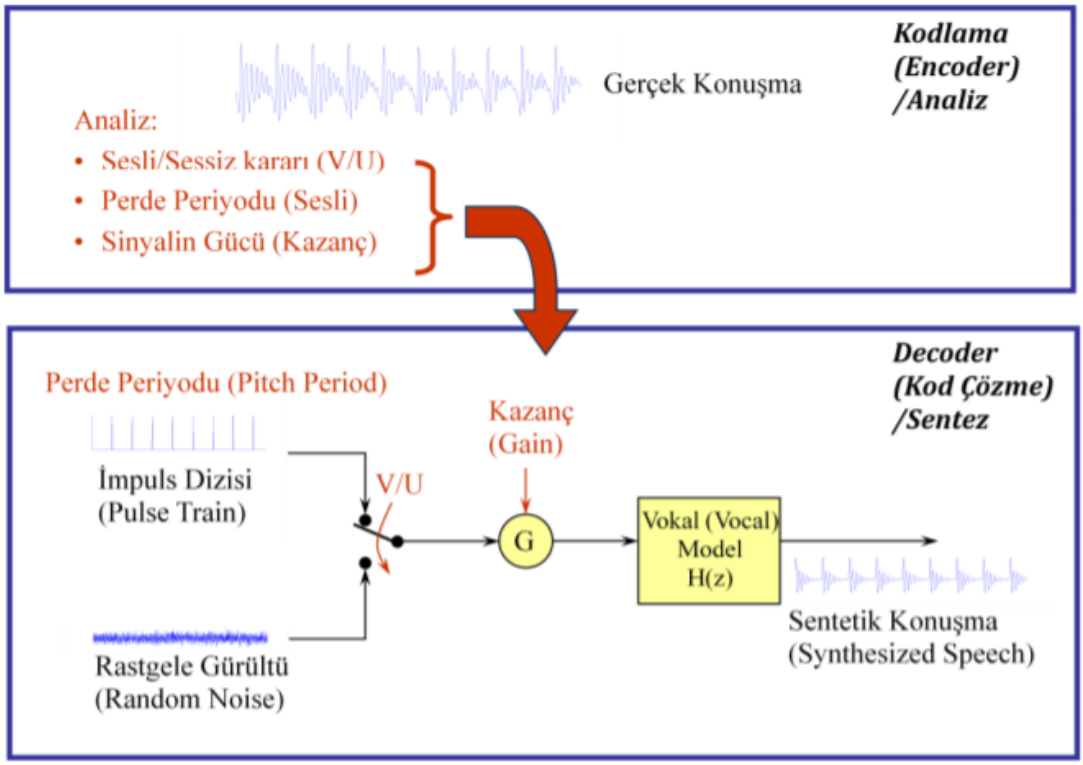
Konuşma analizlerinde kullanılan LPC ile ilgili ilk yayı nlar Atal ve Schroeder, 1968, 1970; Atal ve Hanauer, 1971; Markel, 1971, 1972 olarak kabul görülmektedir [16]. Bu yayınlardan itibaren günümüze kadar LPC hakkında birçok kez bildiri, tez, rapor yazılmıştır. Bu çalışmalarla günümüzde kullanılanılan konuşma analiz yöntemlerine zemin hazırlanmış ve konuşma analizinin geliştirilmesi sağlanmıştır.

Günümüzde konuşma analizinde kullanılan en önemli uygulamalardan biri LPC'dir. Doğrusal öngörünün temelinde, sessiz sesler içinse rastgele darbeler ile uyarılmış, sesli sesler için periyoduk ve düzgün darbeler üreten, zamanla değişen doğrusal bir sistemin çıkışı olarak modellenmesi amaçlanır [17]. LPC algoritması konuşma üretmek için kullanılacak olan parametrelerin (seslilik, formantlar, periyot bilgisi (pitch), vb.)

bulunmasında ve konuşmanın olabilecek en düşük bit hızlarında (bit rate) iletilmesini modellemede kullanılan en temel metotlardan biri haline gelmiştir [12].

4.3 Doğrusal Öngörü ile Kodlama Algoritması Çalışma Şekli

LPC algoritmasının temelinde iki kısım bulunmaktadır. Bu kısımlardan ilki konuşma analizi, ikincisi ise konuşmanın kodlanmasıdır. Konuşma analizi kısmında doğrusal öngörü algoritması ile konuşma analiz edilir. Bu analiz sayesinde konuşmaya ilişkin parametreler (seslilik, formantlar, periyot bilgisi (pitch), vb.) elde edilmiş olunur. Bu paramtreler ile konuşmanın kodlanma kısmına geçilir. Bu kısımda belirlenen parametreler sayesinde konuşma kodlanır, konuşmaya ilişki sayısal bir veri elde edilir. Konuşma yavaş anatomik hareketlerden oluştuğu için konuşma, kısa çerçevelere (20ms, 40 ms)bölündükten sonra elde edilen konuşma aralıkları durağan halde kabul edilip analiz edilir.



Şekil 4.3 : LPC temel yapısı [18].

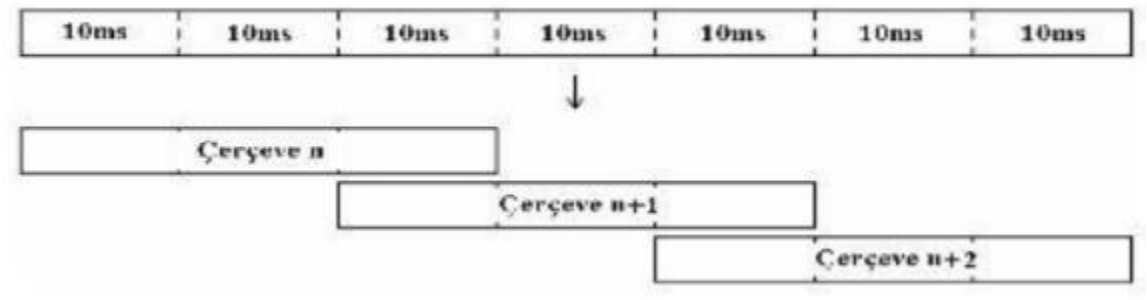
Konuşmanın analiz (encoder) kısmında örneklenen konuşmalar yukarıda bahsedilen kısa aralıklı çerçeveler ile çerçevelenip konuşma parçalarına ayrılır. Bu parçalar geri kalan kısımda tek tek incelenir. Her parça için parça içindeki sinyalin gücü bulunarak kazanç hesabı yapılır. Sinyalı periyodikliğine bakılarak sesli veya sessiz olduğu

anlaşılır. Sesli parçaların periyot bilgisi olduğundan bu parçaların perde periyotları hesaplanır. Daha sonrasında LP katsayıları (lineer prediction coefficients) tekrardan her bir parça için esaplanır ve voka sistem hazırlanır. Böylelikle analiz kısmında bulacağımız parametreler elde edilmiş olur.

Konuşmanın sentez (decoder) kısmında analiz kısmında elde ettiğimiz parametrelere ilişkin konuşma sinyali tekrar elde edilir. Şekil 4.3'de de görüldüğü üzere sesli sinyaller için impuls dizisi (impuls train), sessiz kısımlar için rastgele gürültü (random noice) seçilir. Daha sonrasında analiz kısmında hesaplanmış olan kazanç değeri ile çarpılır. Kazanç ile çarpılmış sinyal vokal sistem bloğuna giriş olarak verilir. Analiz kısmında elde edilen katsayılar bu blokta bir filtre görevi görerek gelen sinyalden ses üretir. Bu sese sentetik konuşma (synthesized speech) adı verilir.

4.3.1 Çerçeveleme

Gerçek zamanlı sistemlerde konuşmanın tamamının işlemden geçirilmesi için sistemin çok yüksek düzeyde hafızası olması gerekmektedir. Bu doğrultuda sistem tasarımının maliyeti artmaktadır. Konuşmanın işlenebilmesi ve bu maliyetlerden kurtulunması için konuşmanın durağan bir sinyal olarak kabul edildiği çerçevelerce çerçevelenmesi gerekmektedir [19]. Çerçeveleme işlemi yapılırken analiz edilen konuşmaya ait bazı bilgiler, çerçevelerin geçiş bölgelerine denk geldiğinden kaybolabilir. Analiz edilen knouşmanın yarısı bir çerçeveye diğer yarısı diğer çerçeveye denk gelebilir. Bu durumun önüne geçmek amacıyla çerçeveleme yöntemi olarak çerçeveler örtüştürülür. Çerçevelerin örtüştürülme ölçütlerinden %25 ve %50 yaygın olanlarıdır.

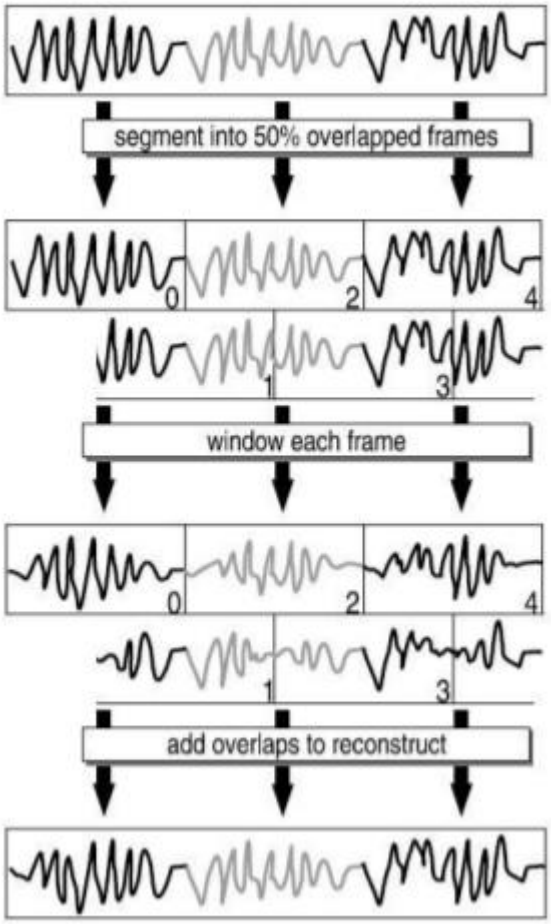


Şekil 4.4 : Sesin örtüştürülmesi [15].

Şekil 4.4 70ms uzunluğundaki konuşma parçasının 3'te 1 lik bir örtüştürme oranıyla örtüştürülmüştür. Çerçeveleme yaparken örtüştürme kullanmanın dezavantajı,

işenecek ses verisinin artmasıdır. Şekil 4.4'te de görüldüğü üzere 70ms'lik bir ses verisi 90ms boyutunda işlenecektir.

Analiz edilen çerçevelenmiş ses verisinden tekrar ses elde edilmesi, çerçeveleme için önemli bir konudur. Örtüştürülme yapılmış olan ses verisi tekrar elde edilirken çerçeveler arka arkaya eklenirse, ilk başta kullanılan ses verisinden uzun ve karmaşık bir ses üretilmiş olunur.



Şekil 4.5 : Çerçeveleme, örtüştürme ve sentez [19].

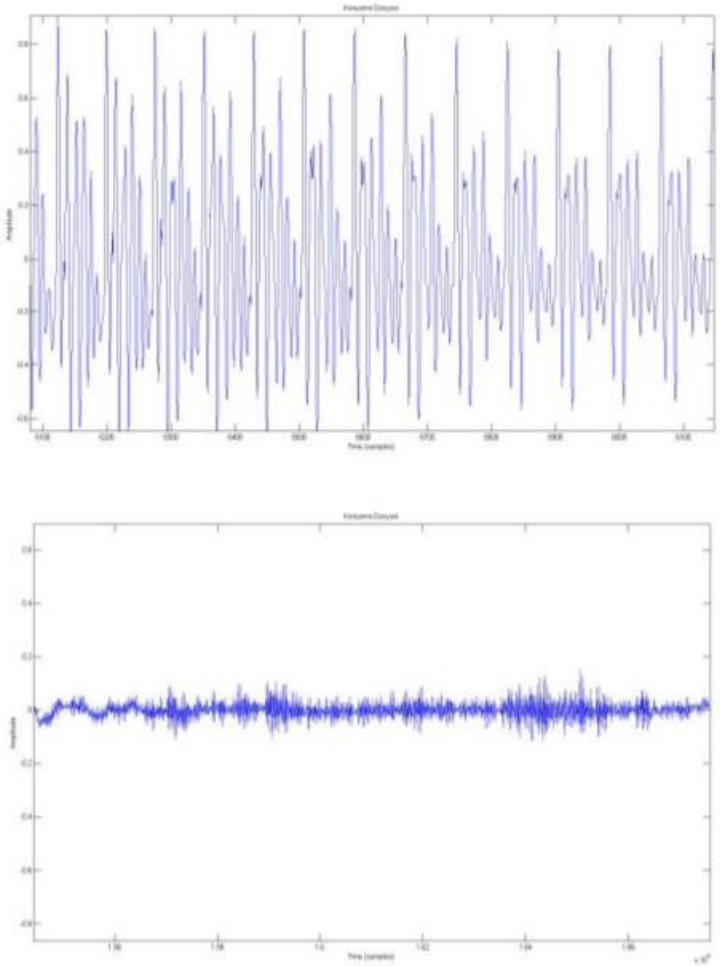
Bu problemi gidermek amacıyla şekil 4.5 şemasında da olduğu gibi analizi yapılmış olan sesin kullanılacak uygulamaya göre Hamming, Bartlett, Blackmann gibi çeşitli pencereleme yöntemleriyle pencerelenmesidir. Pencerelenen ses verileri tekrardan birleştirilir. Böylece fazladan üretilen ses kısımları sistemden atılmış olur.

$$\text{Hamming Penceresi: } w[m] = \begin{cases} 0.54 - 0.46\cos\frac{2\pi m}{N-1} & 0 \le m \le N-1 \\ 0 & diğer \end{cases} \quad (4.1)$$

$$\text{Blackman Penceresi: } w[m] = \begin{cases} 0.42 - 0.52\cos\frac{2\pi m}{N-1} + 0.08\cos\frac{4\pi m}{N-1} & 0 \le m \le N-1 \\ 0 & diğer \end{cases} \quad (4.2)$$

4.3.2 Çerçevelenen Sesin Seslilik Sınıflandırması

Konuşma sinyalleri sesli ve sessiz olarak iki şekilde sınıflandırılabilir [13]. Sesli kısımların sessiz kısımlara kıyasla bir periyodikliği, yüksek enerjisi, yüksek genliği gibi özellikleri bulunur. Sessiz kısımlar ise daha önceden de bahsedildiği üzere ratgele sinyallerden, gürültüden oluşmaktadır. Konuşma sinyalleri sınıflandırılırken, sinyalin öncelikle çerçevelenip durağan sinyaller olarak incelenmesi gerekmektedir. Bu işlem daha önceden bahsedildiği üzere çerçeveleme ve pencereleme olarak yapılmaktadır. Konuşma sinyalinin inelenmesinde başlıca faktörler, sinyalin enerjisi ve enerji dağılımı (Low band to Full band ratio), periyodikliği, tepeli yapıya sahip olması (peakiness), ilk LP tahmin katsayısı, sıfırdan geçme oranı (zero-crossing rate) gibi faktörlerdir.



Şekil 4.6 : Sesli ve sessiz ses örnekleri [13].

Şekil 4.6'da görüldüğü üzere sesli sesin belirli bir periyodikliği, tepeli yapısı, yüksek genliği ve yüsek enerjisi bulunmaktadır. Sessiz ses ise rastgele sinyaller içeren gürültü yapısındadır.

4.3.3 Doğrusal öngörü ile kodlama parametrelerinin elde edilmesi

LPC, United States Department of Defence tarafından 1984 yılında 1015 federal standartı içinde standaştırılarak yayımlanmıştır. Doğrusal öngörülü kodlamanın temelinde kodlanacak sinyalin anlı değerinin o andan daha önceki değerler ile ifade edilmesi vardır. Başka bir deyişle isminden de anlaşılacağı üzere anlık değerin öngörülebilmesdir.

$$s[n] \approx A_1 * s[n-1] + A_2 * s[n-2] + \cdots + A_p * s[n-p] \qquad (4.3)$$

Temel olarak ayrık zamandaki ifadesi (4.3) denklemindeki gibidir. Bu bir fark denklemidir. Z domeninde incelendiği takdirde bir filtredir.

Daha önce bahsedildiği gibi 8 kHz de örneklemiş bir işareti 8 bit ile ifade edilirse 64 kbps'lik bit rate olarak ifade edilir. LPC bit rate'i 2.4 kbps olacak şekilde sıkıştırma imkanı verir. Bu da incelenecek verinin daha iyi iletimine ve incelenmesine olanak verir. Bu bit rate'i veren sistemlerde LPC10 algoritması kullanılmaktadır. LPC10 ismindeki 10 ifadesi denklem (4.3)'deki p değerinin yani derecesinin ifadesidir.

4.3.3.1 Doğrusal öngörü katsayılarının bulunması

Denklem (4.3) öz ilişki katsayıalrını bulmak amacıyla tekrardan düzenlenirse denklem (4.4) elde edilir.

$$s[n] = Gu[n] + \sum_{k=1}^{p} A_k s[n-k] \qquad (4.4)$$

Bu denklemde 'n' anlık değer, 'u[n]' sesli ses parçaları için impuls dizisi, sessiz parçalar için rastgele gürültüye, 'G' Kazanç değerine, 's[n]' sinyal değerine 'A_k değeri ise LPC katsayılarına eşit olmaktadır [12]. Denklem (4.4) Şekil 4.3 ile görsel olarak ifade edilebilir. Tezin kapsamı gereği modelin MATLAB Simulink ortamında Model Composer kullanılarak gerçeklemesi amaçlandığından dolayı, matematiksel gösterimler ve çıkarımlar üstünde detaylı bir anlatıma gidilmeyecektir. Teorik çıkarımlar ve matematiksel modeller detaylı olarak teorik kaynaklar üzerinden incelenebilir [9,13].

Denklem (4.4)'deki vokal kısım doğrusal öngörü ile sentezlenmiş sentetik ses olarak tanımlanırsa denklem (4.5) elde edilir [20].

$$\tilde{s}[n] = \sum_{k=1}^{p} A_k s[n-k] \quad (4.5)$$

Sentezlenen ses ile orijinal ses arasındaki hatayı bulmak istersek, (4.4) ile (4.5) arasındaki farka bakmamız yeterli olacaktır.

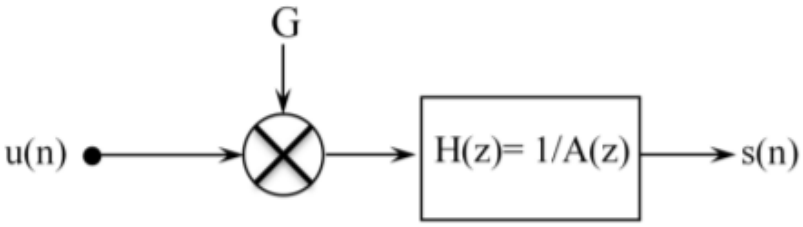
$$e[n] = s[n] - \tilde{s}[n] = s[n] - \sum_{k=1}^{p} A_k\, s[n-k] \quad (4.6)$$

Hata z domeninde incelenirse;

$$E(Z) = S(Z) - \tilde{S}(Z) \sum_{k=1}^{p} A_k z^{-k} = S(Z)\left[1 - \sum_{k=1}^{p} A_k z^{-k}\right] = S(Z)A(Z) \quad (4.7)$$

Transfer fonkisyonu (4.7) kullanılarak çıkartılabilinir:

$$A(Z) = \frac{E(Z)}{S(Z)} = 1 - \sum_{k=1}^{p} A_k z^{-k} \quad (4.8)$$



Şekil 4.7 : LP modeli[5].

Sinyal ve hata fonksiyonları, A_k katsayılarının bulunabilmesi için kısa zaman zarfında periyodik olarak kabul edilerek (4.9) ve (4.10) denklemlerinde tanımlanır.

$$s_n[m] = s[n+m] \quad (4.9)$$

$$e_n[m] = e[n+m] \quad (4.10)$$

Sistemin ortalama karesel hatası;

$$\mathrm{E_n} = \sum_{\mathrm{m}=-\infty}^{\infty} \mathrm{e}^2[\mathrm{m}] = \sum_{\mathrm{m}} \left(\mathrm{s_n}[\mathrm{m}] - \sum_{\mathrm{k}=1}^{\mathrm{p}} \mathrm{A_k}\, \mathrm{s}[\mathrm{m}-\mathrm{k}]\right)^2 \quad (4.11)$$

Tahmin katsayılarını bulmak için ortalam karesel hatanın minimum olması gerekmektedir. Minimum ortalama karesel hata için ortalama karesel hatanın bütün A_k katsayılarına göre kısmı türevi alınarak sıfıra eşitlenir. Bu sayede katsayı adedi kadar denklem elde edilir. LPC10 algoritması için 10 katsayı elde edilir. Bulunan denklemler çözülerek katsayıalr hesaplanmış olunur.

$$\frac{dE_n}{dA_k} = 0, \qquad k = 1,2,\dots,p \qquad (4.12)$$

$$\sum_m s_n[m-i]\, s_n[m] = \sum_{k=1}^{p} A_k \sum_m s_n[m-i]\, s_n[m-k] \qquad (4.13)$$

Özel bir ifade olan kovaryans bağıntısı, denklem (4.14) de tanımlanmıştır.

$$\varphi(i,k) = \sum_m s_n[m-i]\, s_n[m-k] \qquad (4.14)$$

Bu tanımla birlikte denklem (4.13) ve (4.14) birleştirilirse;

$$\varphi(i,0) = \sum_{k=1}^{p} A_k * \varphi(i,k) \qquad (4.15)$$

Ortalama karesel hesabı minimuma indirmek için (4.15) de değişiklik yapmamız gerekir.

$$E_n = \varphi(0,0) = \sum_{k=1}^{p} A_k * \varphi(0,k) \qquad (4.16)$$

A_k katsayılarının bulunabilmesi için $\varphi(i,k)$ 'nın $1 \le i \le p,\ 0 \le k \le p$ durumlarında oluşturulması gerekmektedir [12]. Oluşturulan $\varphi(i,k)$ 'p' tane denklemin çözümü sonrası A_k katsayıları hesaplanır.

Katsayıalrın hesaplanması için matematiksel tasarımlar incelendiğinde kovaryans bağıntısındaki 'm' değeri için bir sınır bulunmamaktadır. Bu sınırsızlık denklem çözümleri için gerçek bir sonuç çıkartamamaktadır. Bu sınırsızlığa çözüm olarak öz ilişki (autocorrelation) metodu önerilmiştir [9]. Bu metotta m değeri için $0 \le m \le N-1$ sınır getirilip bu sınır dışında sinyal değeri 0 olarak alınır.

Öz ilişki ve kovaryans eşitliklerinin birbiri ile bağıntısı;

$$r_n(\tau) = r_n(|i-k|) = \varphi_n(i,k) \qquad (4.17)$$

(4.17) denklemi ile (4.18) denklemi kullanarak (4.15) denklemi öz ilişki cinsinden ifade edilirse;

$$r_n(i) = \sum_{m=1}^{p} s_n[m]\, s_n[m+i] \qquad (4.18)$$

$$\sum_{k=1}^{p} A_k r_n(i-k) = r_n(i) \qquad (4.19)$$

(4.19) denklemleri yazılıp matris formda incelenirse;

$$\begin{bmatrix} R_n(0) & R_n(1) & \dots & R_n(p-1) \\ R_n(1) & \vdots & \vdots & R_n(p-2) \\ \vdots & \vdots & \vdots & \vdots \\ R_n(p-1) & \cdots & \cdots & R_n(0) \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix} \quad (4.20)$$

(4.20) matrisi incelendiğinde, matrisin simetrik ve köşegen üzerindeki elemanlar eşittir. Bu tip matrisler Toeplitz yapısı olarak adlandırılır ($R_{n_{i,j}} = R_{n_{i+1,j+1}}$). Katsayıların bulunması için matris çözülmesi gerekmektedir. Her ne kadar mxm tipinde bir matris olsa da sayısal sistemlerde bu çözüm hataya sebebiyet vermektedir. Matris çözümü için en çok tercih edilen yöntem Levinson – Durbin algoritmasıdır [9,13,20].

Algoritma aşşağıdaki adımlarla matris çözümüne ulaştırır [21].

$$1.\ s_0 = r(0) \quad (4.21)$$

$2.\ i = 0,1, \dots, p-1$ içinaşşağdıaki adımlar takip edilir.

$$a.\ \rho_{i+1} = \frac{r(i+1)+\sum_{k=1}^{i} A_{i,k} r(i+1-k)}{s_i} \quad (4.22)$$

$$b.\ s_{i+1} = s_i(1-\rho_{i+1}^2) \quad (4.23)$$

$$c.\ A_{i+1,k} = A_{i,k} - \rho_{i+1} A_i\ ,\ \ 1 \le k \le i \quad (4.24)$$

$$d.\ A_{i+1,k+1} = -\rho_{i+1} \quad (4.25)$$

Bu işlemler sonucunda bulunan $A_{i+1,k+1}$ katsayıları (4.8) denklemindeki A_k katsayılarına yani LPC katsayıalrına eşittir. Bu katsayılar ile LPC'nin sentez kısmında bulunan vokal model (Şekil 4.3) tasarlanmış olur.

4.3.4 Kazanç hesabı

Analiz edilen sesin tekrar elde edilebilmesi için sentetik ses üretimi üzerine konuşulmuştu. Bu sentetik sesin analiz edilen ses ile aynı olması istenmektedir. Bu yüzden enerji düzeyleri aynı olması gerekmektedir. Filtrenin girişine verilen sinyal 'G' kazancı ile çarpılarak bu eşitlik sağlanmaktadır.

$$\mathrm{Gu[n]} = \mathrm{s[n]} - \sum_{\mathrm{k=1}}^{\mathrm{p}} \mathrm{A_k s[n-k]} = \mathrm{e(n)} \quad (4.26)$$

Kazanç hesabı her ne kadar hata sinyali üzerinden bulunması çok güvenilir olmasa da kazanç, hata sinyalinin enerjisine göre seçilmektedir [13].

4.3.5 Perde (periyot) bilgisinin eldesi

Ses verisinin analiz edildikten sonra sentez kısmında tekrar düzgün bir şekilde elde edilebilmesi için sesli kısım için periyot bilgisine ihtiyaç vardır. Bu periyot bilgisi ile ortaya çıkan sesin kalitesi gerçek sese çok yakın olacaktı. Analiz edilecek sesin periyodik olarak mükkemmel olmaması, gürültü ve yankının eklenmesi periyot bilgisi elde etmekte problem olşturmaktadır. Sesin periyot bilgisini saptamak için otokoreasyon yöntemi kullanılabilir. Bunun sebebi olarak otokorelasyon, sinyalin kendine benzerlğinin bir ölçütü olmasıdır [13]. Bu yöntemde periyot bilgisi için sinyal gecikmesi arttırılarak otokorelasyon katsayıları hesaplanır. Hesaplanan katsayılardan en büyüğünün olduğu gecikme periyot bilgisi verir. Yöntemin dezavantajı olarak her gecikme değerinde hesaplama yapmayı gerektirir. Bu da zaman kaybı ve maaliyet artışına sebebiyet vermektedir.

$$AMDF[l] = \sum_{n=0}^{N-1} |S[n] - S[n-l]| \qquad (4.27)$$

Denklem (4.27), genlik fark fonksiyonudur (AMDF, Average Magnitude Difference Function) ve periyot bilgisi bulmak için performans olarak otokorelasyon yöntminden çok daha iyidir.

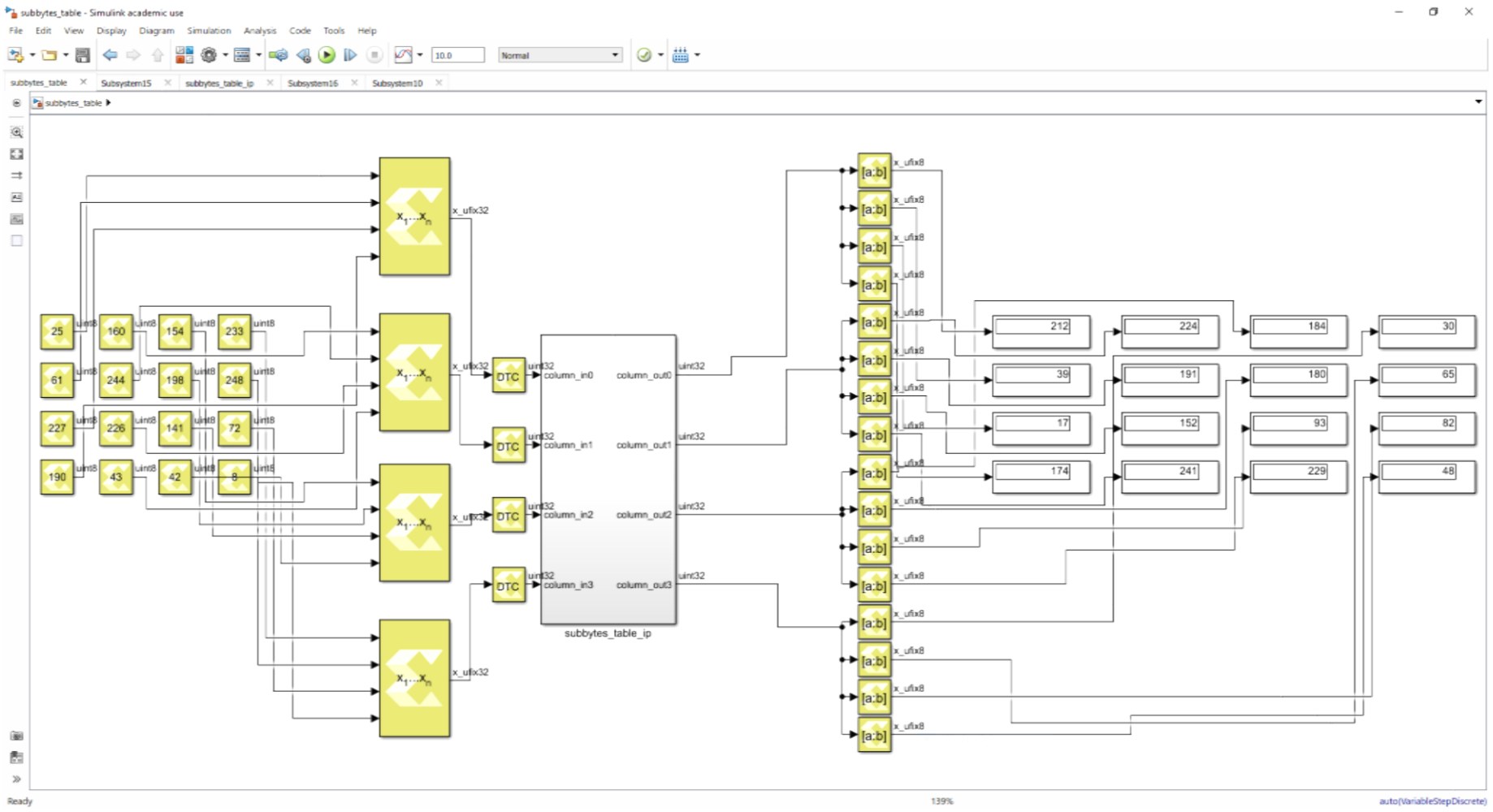
5. SEÇİLEN ALGORİTMALARIN MODEL COMPOSER İLE GERÇEKLENMESİ

5.1 Gelişmiş Şifreleme Standardı Algoritmasının Model Composer Ortamında Gerçeklenmesi

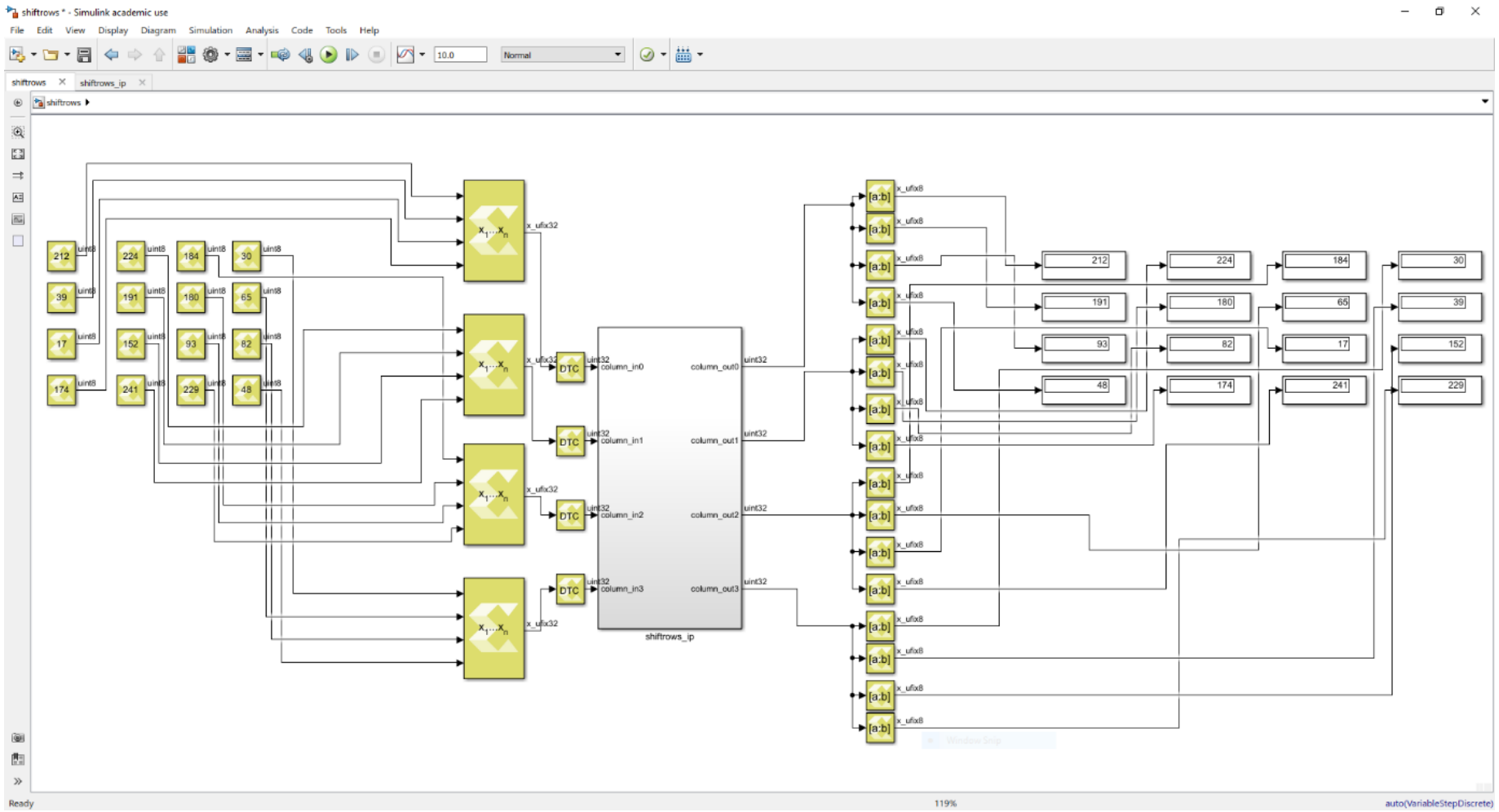
Bu bölümde, FIPS yönergeleri baz alınarak, AES algoritmasının Model Composer ortamında model temelli tasarımı gerçeklenmiştir. Model Composer'ın yeni bir araç olması ve geliştirilmeye hala devam ediliyor olması nedeniyle, AES tasarımının IP olarak vivado ortamına aktarılması sağlanamamıştır.

Dördüncü bölümde teorik bilgisi verilen SubBytes, ShiftRows ve MixColumns'un Model Composer ile yapılan gerçeklemeleri Şekil 5.1, Şekil 5.2 ve Şekil 5.3'te görülmektedir. Şekil 5.4'te tur anahtarını üreten modül görülmektedir. Şekil 5.5'te birinci tur özelinde çalışan şifreleme tasarımı görülmektedir. Şekil 5.6'da şifreleme donanımının on tur sonrasındaki değerleri görülmektedir. Tasarımlara ait giriş ve çıkış değerlerinin kontrolü yapılmıştır [6].

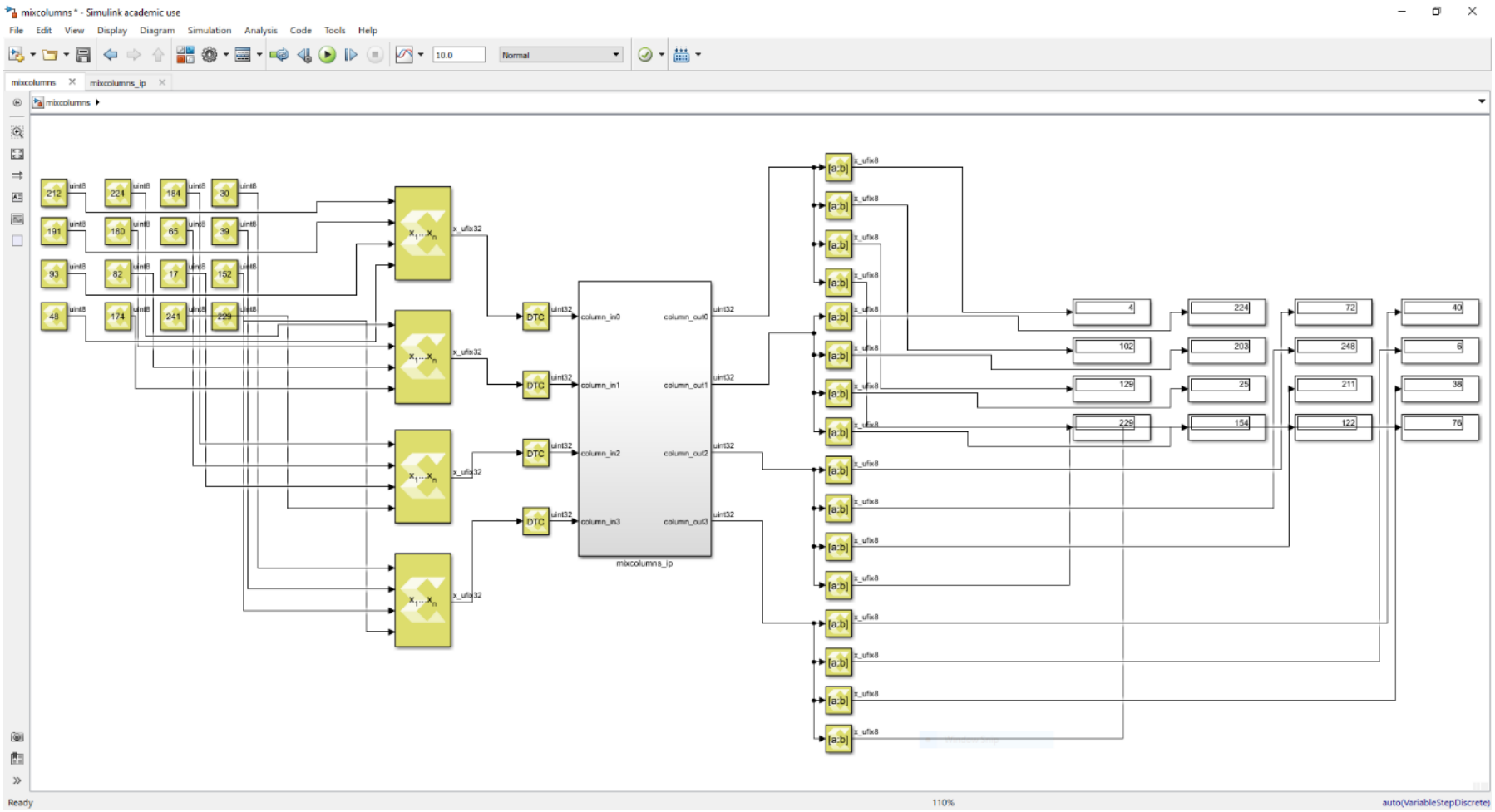
Dördüncü bölümde teorik bilgisi verilen InvSubBytes, InvShiftRows ve InvMixColumns'un Model Composer ile yapılan gerçeklemeleri Şekil 5.7, Şekil 5.8 ve Şekil 5.9'da görülmektedir. Şekil 5.10'da şifreleme ve şifre çözme donanımlarına ait tasarımlar birarada görülmektedir.



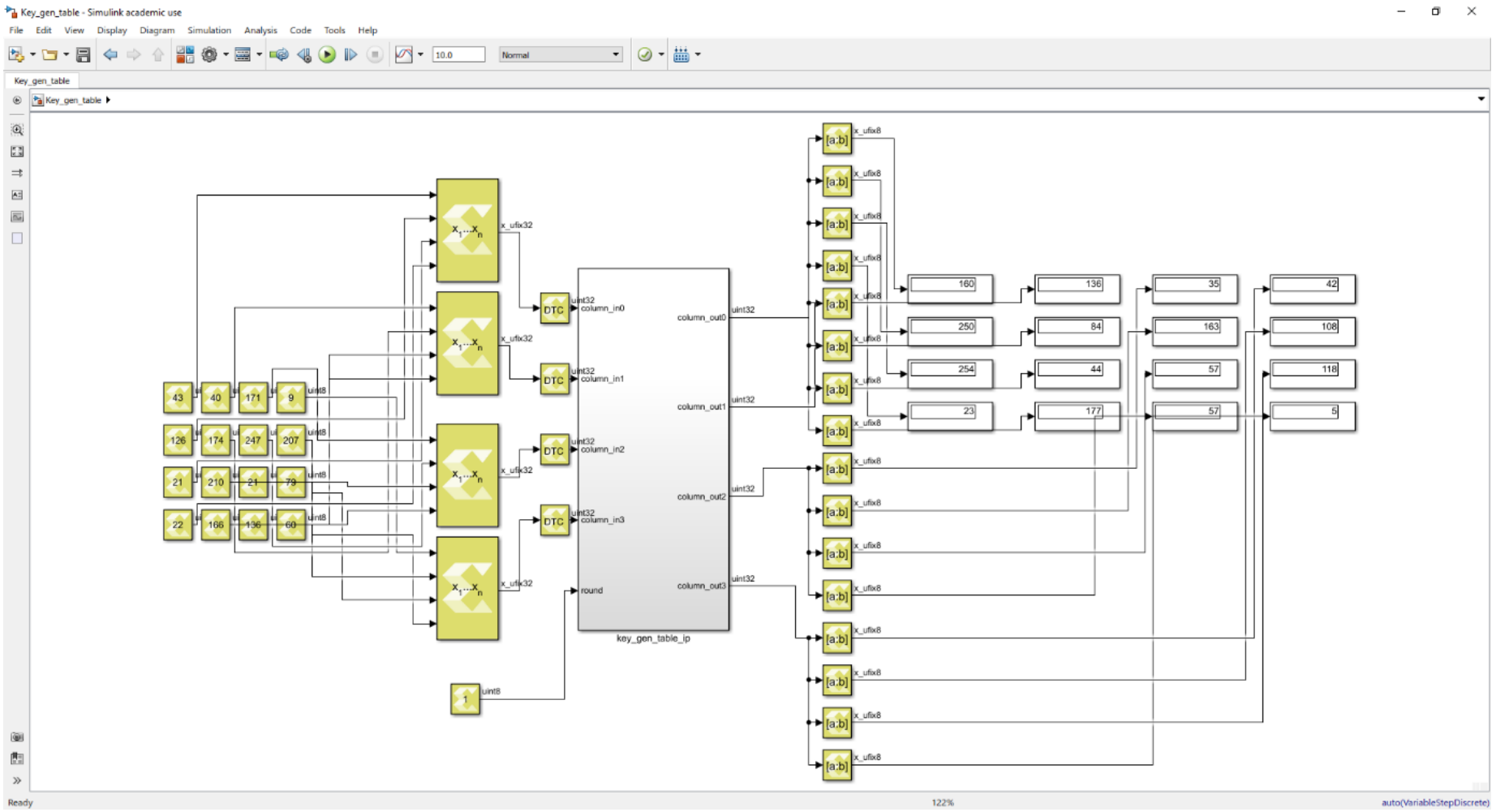
Şekil 5.1 : SubByte Model Composer simülasyonu



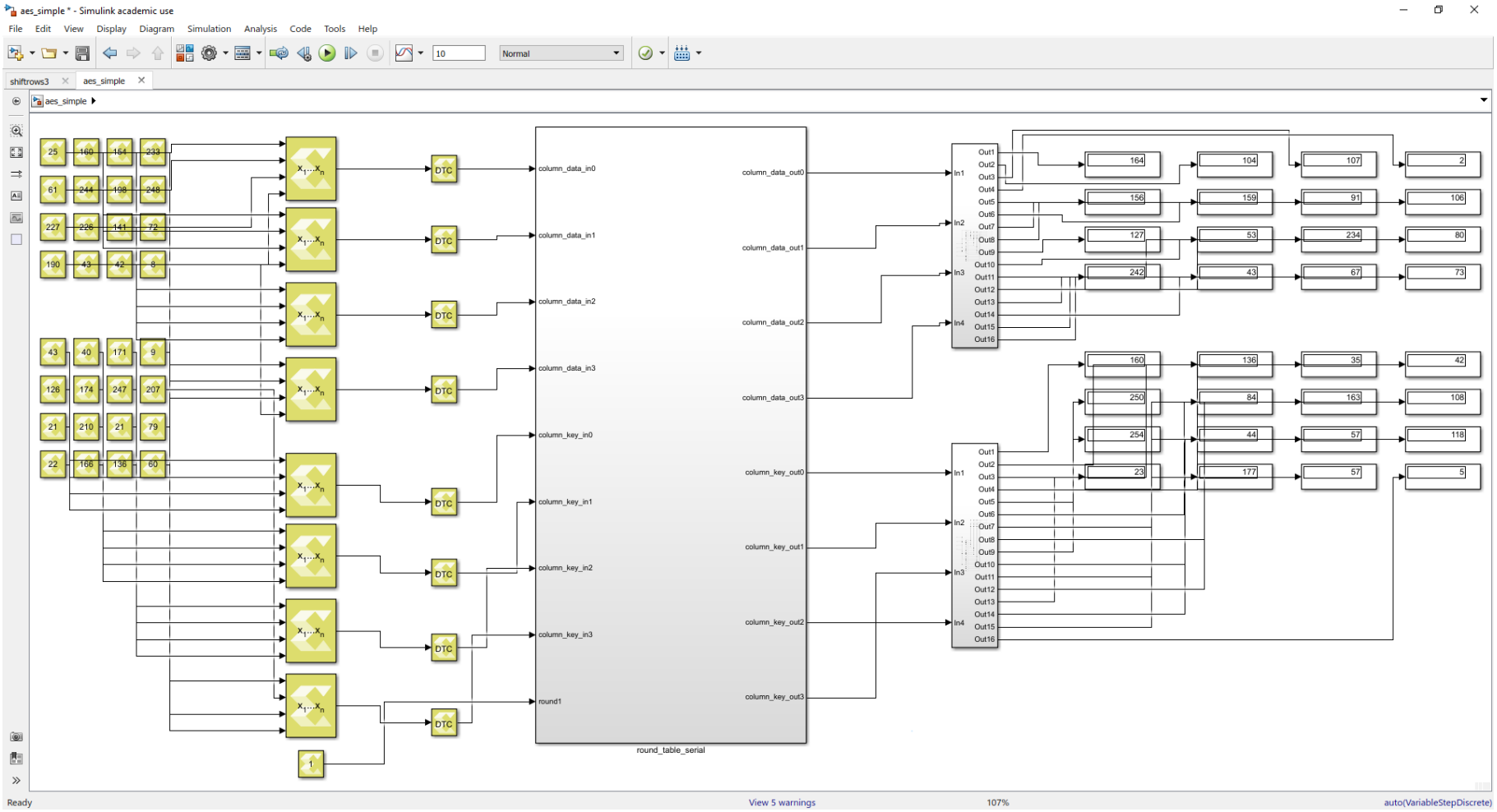
Şekil 5.2 : ShiftRows Model Composer Simülasyonu



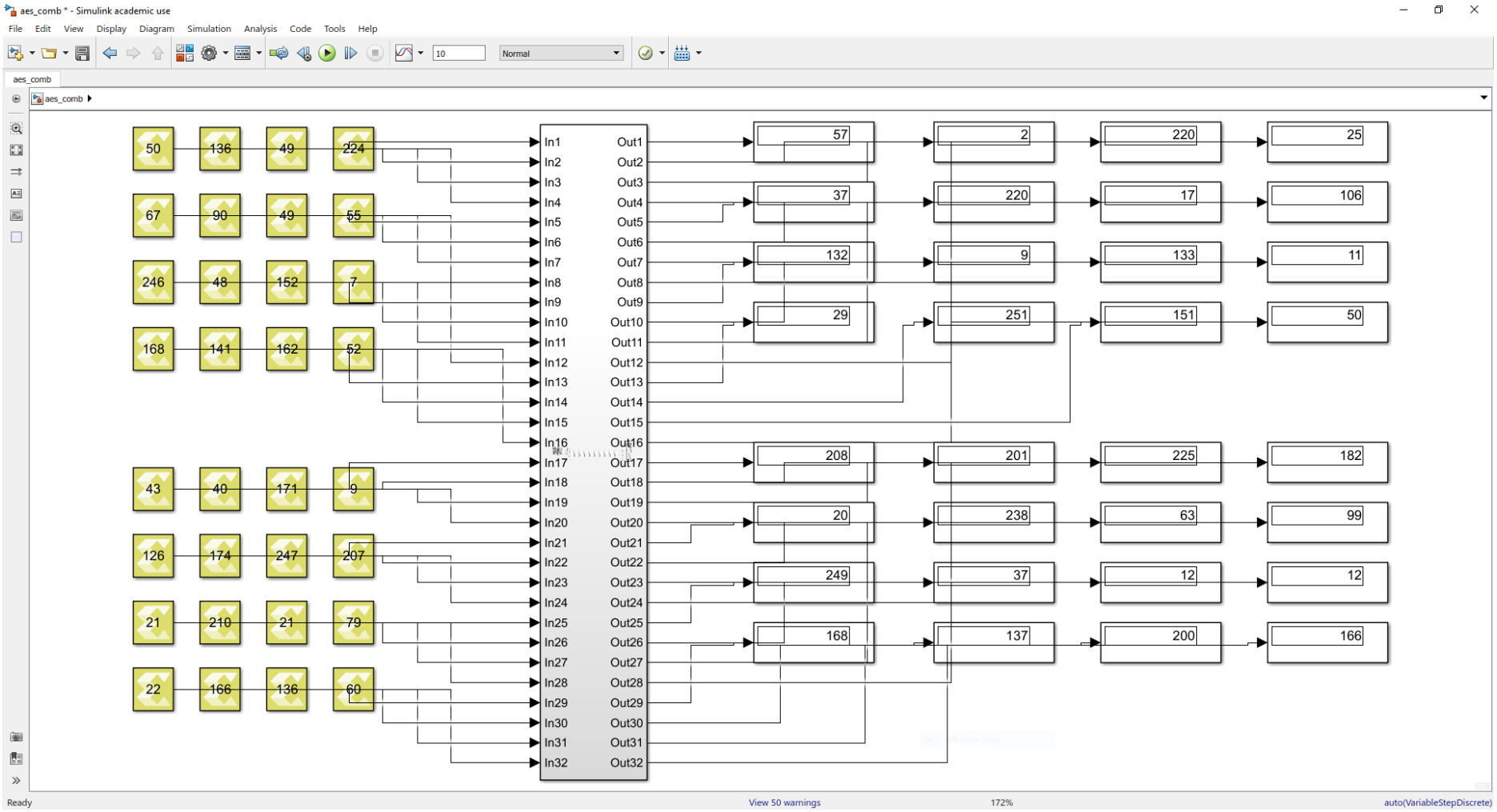
Şekil 5.3 : MixColumns Model Composer simülasyonu



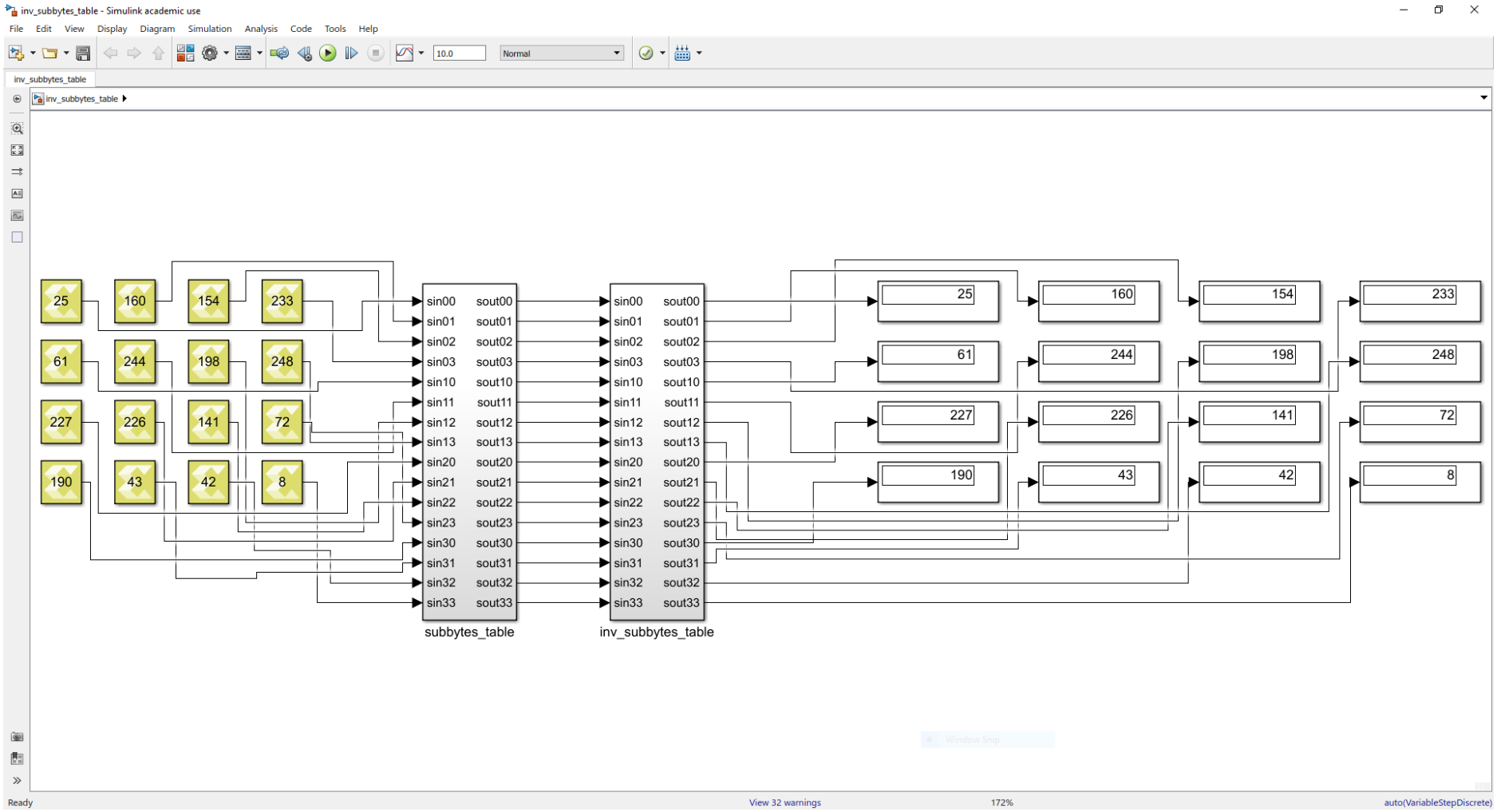
Şekil 5.4 : Model Composer Anahtar Üreten Blok Tasarım



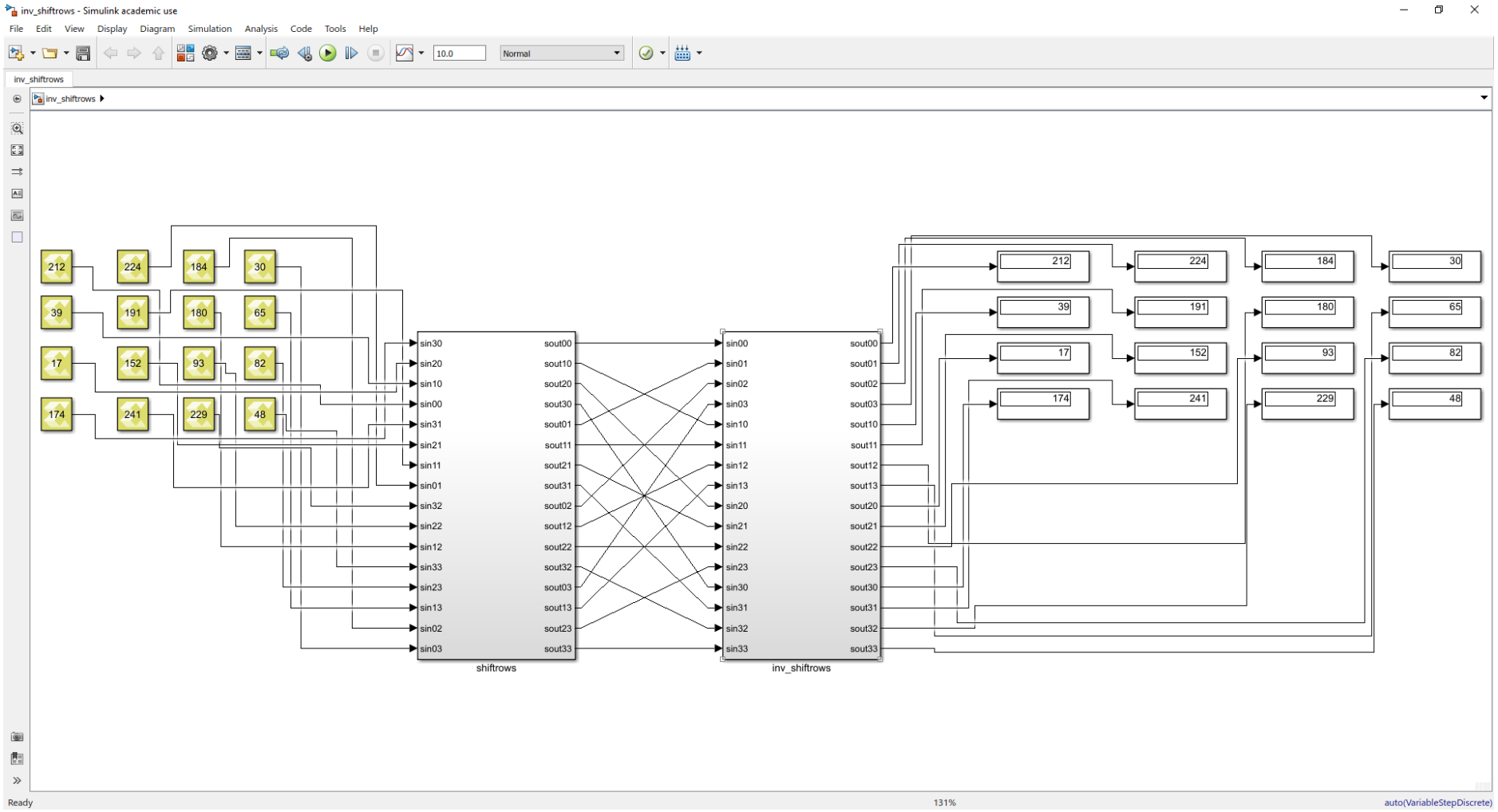
Şekil 5.5 : 1. tur için AES Model Composer simülasyonu



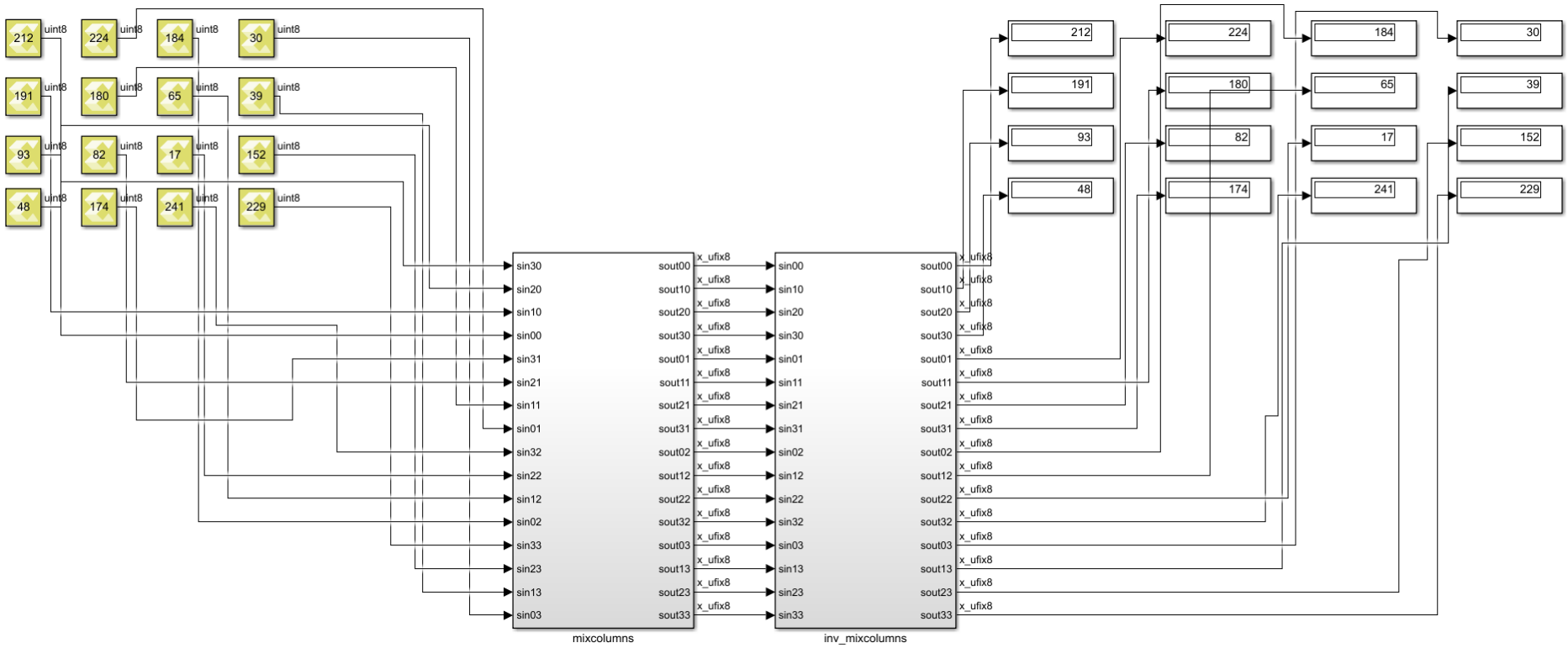
Şekil 5.6 : 10 tur çalışan AES Model Composer simülasyonu



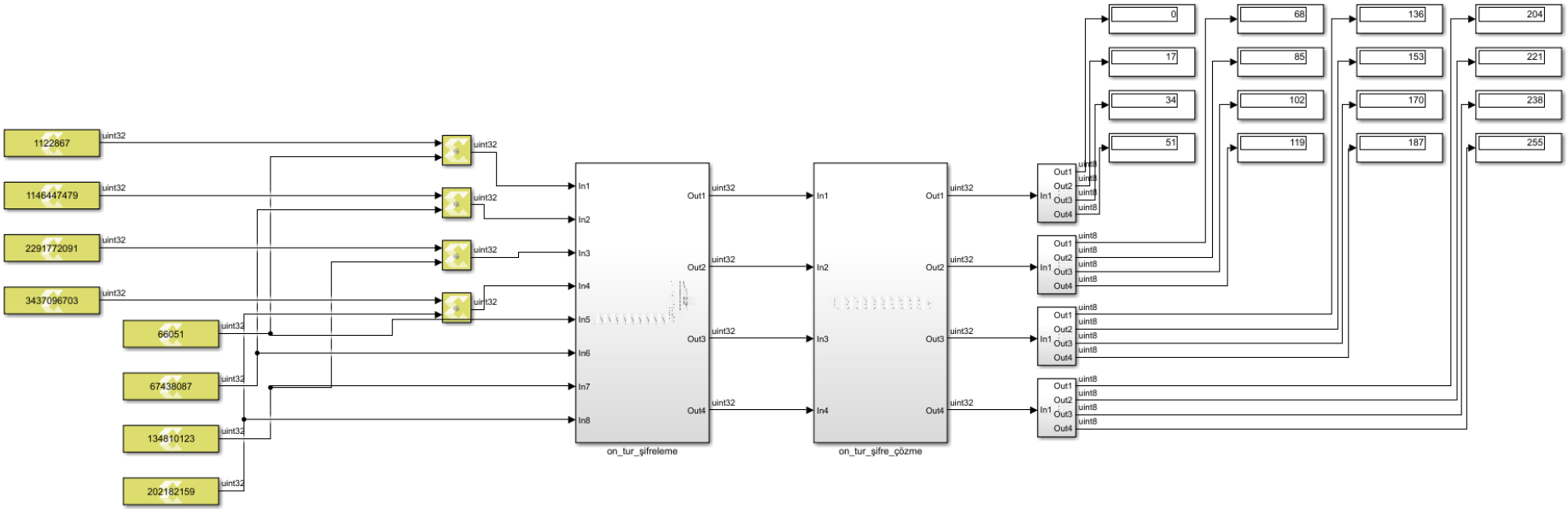
Şekil 5.7 : InvSubByte Model Composer simülasyonu



Şekil 5.8 : InvShiftRows Model Composer simülasyonu



Şekil 5.9 : InvMixColumns Model Composer simülasyonu



Şekil 5.10 : AES şifreleme ve şifre çözme Model Composer simülasyonu

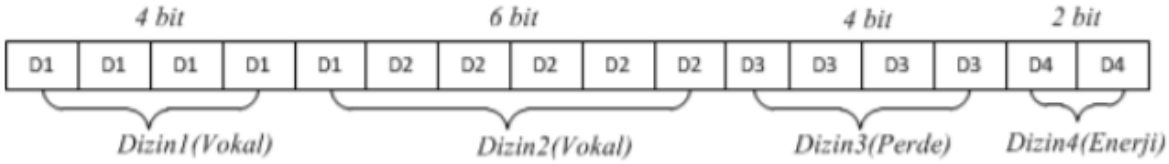
5.2 Doğrusal Öngörü Yöntemiyle Kodlama Algoritmasının Model Composer Ortamında Gerçeklenmesi

5.2.1 Doğrusal öngörü ile kodlama sentez kısmının gerçeklenmesi

GSM üzerinden taşınacak olan sesin bit dizisi olarak Radyo Frekansına (RF, Radio-Frequency) aktarılamayacağı bilinmektedir. Gelen ses verisinin RF katına iletiminin sağlanması için daha önceki bölümlerde anlatıldığı üzere bulunacak LPC parametreleri ile sentetik ses sentezlenip RF katına aktarım yapılması planlanmıştır.

5.2.1.1 Bit dizisi ayrıştırma

LPC sentez kısmına gelecek olan 16 bitlik bit dizisi, ilgilenilecek parametreler için ayrıştırılır. Bu ayrıştırma bu içerikler filtre için katsayılar (vokal kısım), perde periyot bilgisi, kazanç bilgisi olmak üzere 3'e ayrılır.



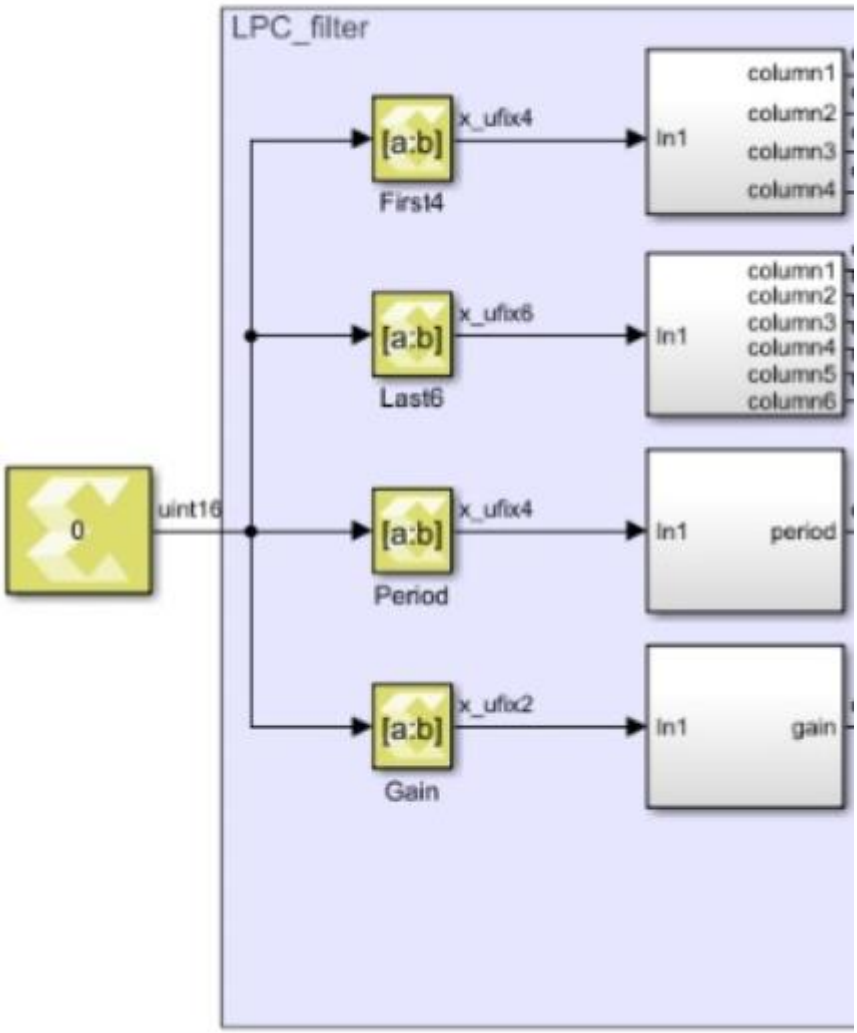
Şekil 5.11 : Ayrıştırma İşlemi [12].

5.2.1.2 Doğrusal öngörü ile kodlama katsayıların model composer ile belirlenmesi

LPC katsayılarını içeren bit dizisi toplamda 10 bit içermektedir. 10 bit olmasının sebebi öngörü derecesinin 10 olmasından kaynaklanmaktadır. LPC değerlerinin tutulacağı kod kitap boyutlarının minimum düzeyde olması istenmektedir. Tek bir kod kitabı üzerinde tutulmak istendiğinde 210 = 1024 adet farklı kod kitabı satırı belirlenmesi gerekmektedir. Bu boyutlar işlem yükünü arttıracağından bu yapının bölünmesi önerilmiştir. 4 bitlik ve 6 bitlk olarak bölünmesi önerilmiştir. Bu sayede 24 +26 =80 ade farlı kod kitabı satırı belirleyerek işlem gücü azaltılır.

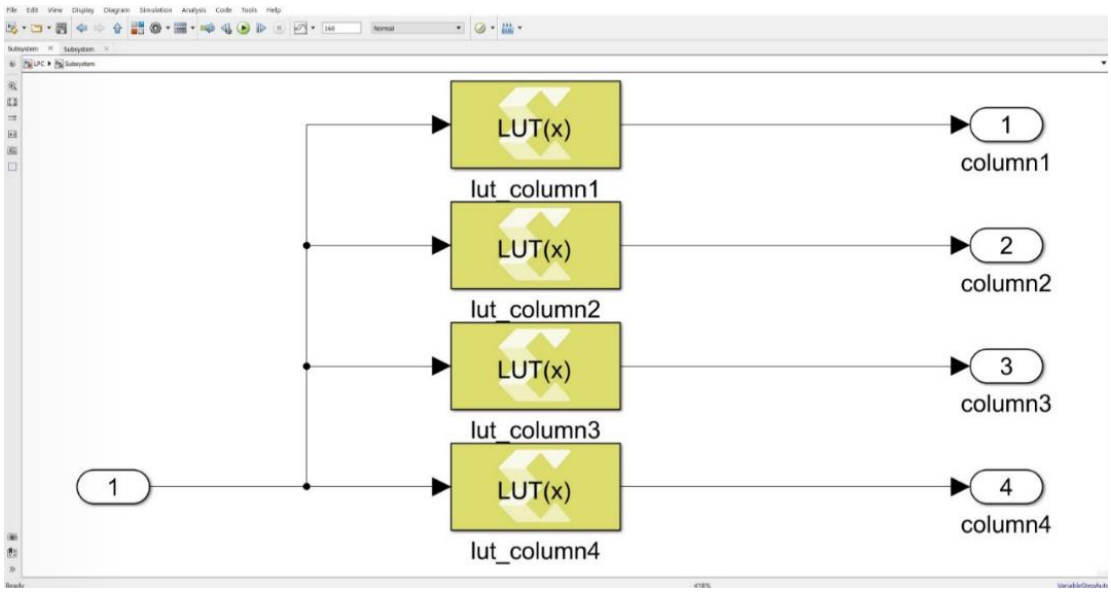
Şekil 5.12'da gözüktüğü üzere bahsettiğimiz ayrıştırma işlemleri Model Composer ile Simulink ortamına ktarılmıştır. Aktarım yapılırken bahsedilen ayrıştırma işlemlerine dikkat edilerek işlemler yapılmıştır. Sentez girişine gelen 16 bitlik bit dizisi, ilk 4 bit son 6 bit şeklinde LPC katsayıları, 4 bitlik perde bilgisi ve 2 bitlik kazanç bilgisi olarak ayrılmıştır. First4 bloğu gelen bit dizisinin en anlamlı ilk 4 bitini, Last6 bloğu kalan

bitlerden en anlamlı 6 biti, Period bloğu kalan bitlerden 4 biti ve Gain bloğu ise son kalan 2 biti seçmektedir.



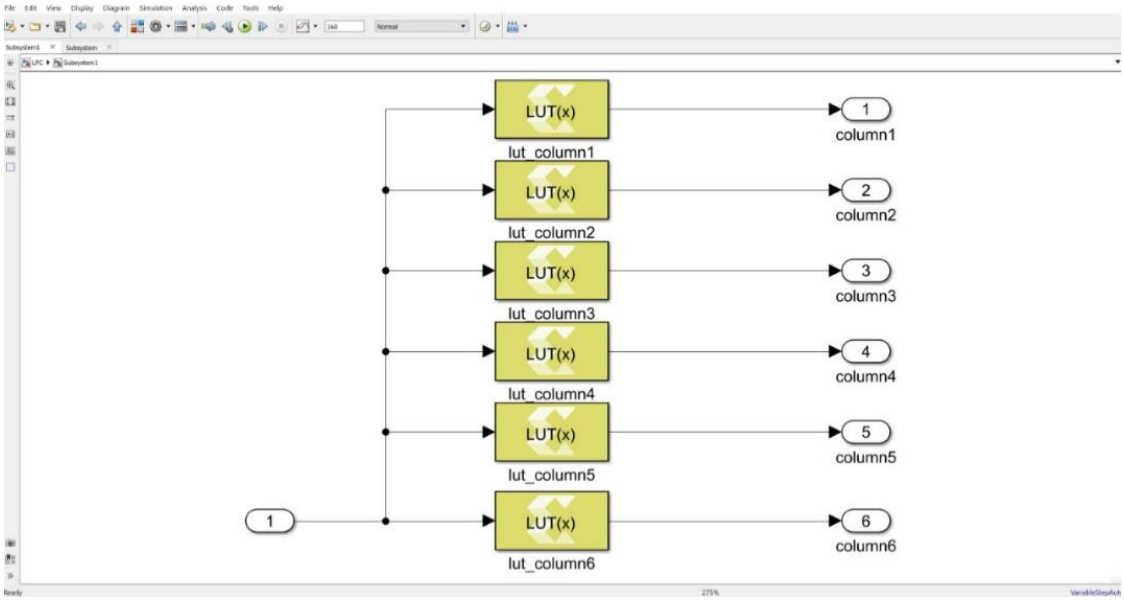
Şekil 5.12 : Model Composer ile ayrıştırma

Bu seçimler yapıldıktan sonra seçilen bitler, '[5]' kaynağından kaynak sahibinin izni ile, Akif Özkan tarafından hazırlanan kod bloklarına iletilmiştir. Bu kod bloklarında NTIMIT veri tabanından taranan LPC katsayıları, perde bilgileri ve kazanç bilgileri bulunmaktadır. Gelen bit dizisine göre kod bloklarından LPC katsayıları, perde bilgisi ve kazanç bilgisi seçilir.



Şekil 5.13 : First4 Kod blok Subsystem yapısı

LPC katsayılarını belirlerken Akif Özkan'dan alınan kod blokları uygu şekilde Model Composer kullanılarak Simulink ortamına aktarıldı. First4 için gelen 4 bitlik bit dizisine göre LookUp Table içindeki LPC katsayı değerleri seçilir. Toplamda 10 adet katsayı olacağından ilk 4 ve son 6 olarak LookUp Table ayırmı yapılmıştır. Gelen bit dizisine göre First4 için bütün LookUp Table'lardan aynı bit değeri sayısı seçilir (0000 biti için ilk değerler).



Şekil 5.14 : Last6 Kod Blok Subsystem Yapısı

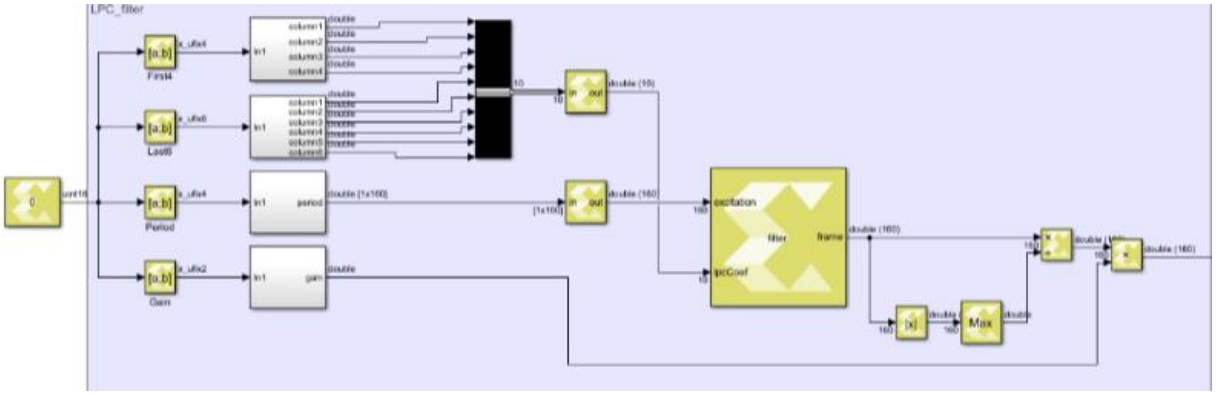
Aynı şekilde Last6 Subsytem'i için gelen 6 bitlik değere göre her bir LookUp Table'dan, gelen bit dizisi değerine göre LPC katsayıları seçilir (000000 için ilk

değerler). Bunun sonucunda katsayılar ile filtre modeli tasarlanacak ve vokal model oluşturulacaktır.

5.2.1.3 Perde Periyodu ve Kazanç Bilgisinin Model Composer ile Belirlenmesi

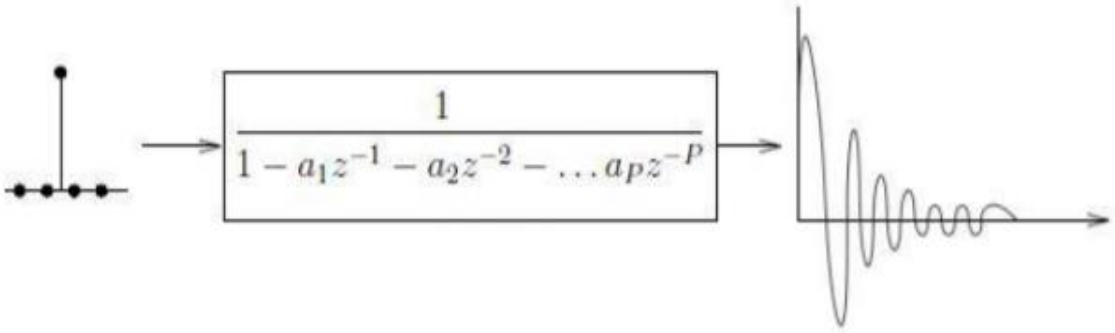
Perde periyodu 4 bitlik veriden elde edileceği için toplam 16 adet periyot bilgisi bulunmaktadır. Kazanç bilgisi ise 2 bitlik veriden elde edileceği için toplam 4 adet kazanç bilgisi bulunmaktadır. Bu yüzden hem periyot hem kazanç bilgileri, birer LookUp Table ile tutulup gelecek bit dizisine göre sonuç alınmaktadır (0000 bit değeri için ilk değer, 00 bit değeri için ilk değer). Seçilen periyot bilgisine göre bir uyarı sinyali oluşturulur. Bu uyarı sinyali 160 örnek içerir. Bunun sebebi sesin 8kHz de örneklenip 2.4kbps'da sıkışıtırlmasıdır. Periyot bilgisine göre önek değerlerinden bazıları 1, geri kalan kısım 0 ile modellenir. Konvolüsyon yapılacağından b değerler sentetik seste periyot bilgisini verecektir.

5.2.1.4 Sentez işlemleri



Şekil 5.15 : LPC Sentez Şeması

Model Composer ortamının kısıtlı blok şemaları olduğudan dolayı filter bloğu C kodu ile tasarlanıp Simulink ortamına blok olarak aktarılmıştır.



Şekil 5.16 : Sentez filtresi

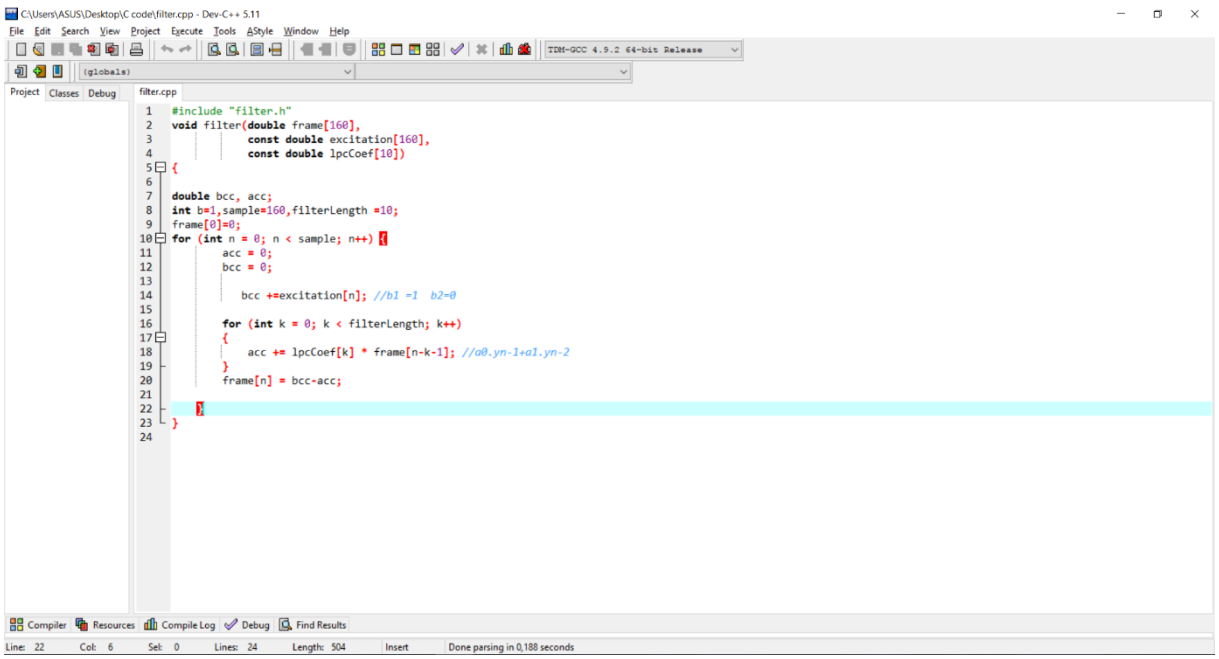
Filtre Bloğu Şekil 5.16'de de görüldüğü üzere bir fark denkleminin z domenindeki karşılığıdır. Zaman domenindeki karşılığı Şekil 5.17'te verilmiştir. Zaman domeni karşılığı C ortamında daha kolay gerçekleneceğinden zaman domeni tercih edilmiştir.

$$a(1)\,y(n) = b(1)x(n) + b(2)x(n-1) + \ldots + b(n_b+1)x(n-n_b)$$
$$-a(2)\,y(n-1) - \ldots - a(n_a+1)\,y(n-n_a).$$

Şekil 5.17 : Sentez filtresi fark denklemi

Fark denkleminde katsayılar a ile, excitation (uyarma sinyali) x(n) ile ifade edilir. B katsayıları Şekil 5.16 düşünüldüğünde b(1) =1, diğer değerler sıfırdır.

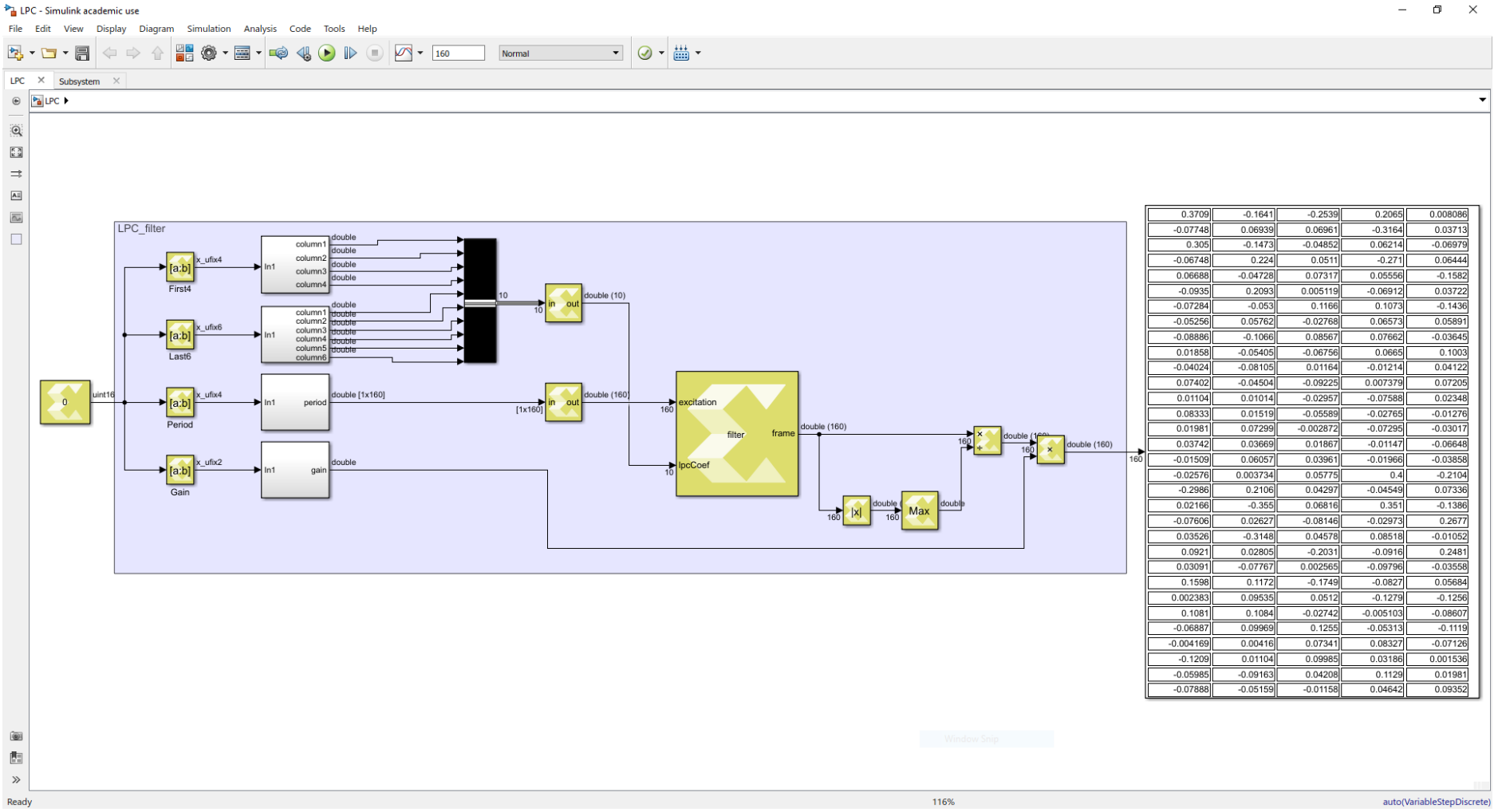
Filtre giriş değeri olan excitation değerlerini Şekil 5.16'daki katsayıları, gelen bit dizisine göre seçtiğimz katsayılar olan filtresine giriş oalrak verir. Z domeninde çarpma işlemi olan bu işlem zaman domeninde konvolüsyona denk düşmektedir. Tasarım kolaylığı için zaman domeni tercih edildiğine değinilmişti. Bu sebeple filter bloğunda konvolüsyon işlemi yapılmaktadır.



Şekil 5.18 : Filtre bloğu C kod karşılığı

Filtre çıkışında 160 örnekli bir veri elde edilmektedir. Bu aslıda sentetik bir ses verisidir ve periyodiktir. Filtre, gelen parametreler doğrultusunda filtre çıkışındaki sentetik ses verisinde kırpılmalar görülebilir. Bunu önlemek amacıyla filtre çıkışındaki değerleri normalize edilme işleminden geçirilir. Sonuç olarak kırpılmalardan kurtulunmuş olunur. Normalize işlemi sentetik ses verisinin mutlak en büyük değerinin her bir değere bölünerek bütün değerleri 0 ila 1 arasında indirme işlemidir. Bu işlem yapıldıktan sonra normalize edilmiş sentetik ses değerlerinin her biri kazanç ile çarpılarak orijinal sese en yakın sentetik ses verisi normalize edilmiş oalrak elde edilmiş olunur.

Sentetik ses bu süreçlerden sonra RF katına aktarılıp iletim sağlandıktan sonra LPC algoritmasının analiz kısmını içeren bloğa aktarılır. Akif Özkan'ın kullanmış ve hazırlamış olduğu kodlar incelendiğinde sentez kısmı tasarlanmış, analiz kısmı için Model Composer'ın yetersiz kaldığı tespit edilmiştir. Tasarım analiz kısmına kadar tamamlanmış, LPC sentez bloğunun simulasyonu yapılmıştır. Simulasyon değerleri ile baz alınan Akif Özkan tarafında hazırlanan MATLAB tasarımı karşılaştırılmış ve sonuçların tutarlı olduğu görülmüştür.



0.3709	-0.1641	-0.2539	0.2065	0.008086
-0.07748	0.06939	0.06961	-0.3164	0.03713
0.305	-0.1473	-0.04852	0.06214	-0.06979
-0.06748	0.224	0.0511	-0.271	0.06444
0.06688	-0.04728	0.07317	0.05556	-0.1582
-0.0935	0.2093	0.005119	-0.06912	0.03722
-0.07284	-0.053	0.1166	0.1073	-0.1436
-0.05256	0.05762	-0.02768	0.06573	0.05891
-0.08886	-0.1066	0.08567	0.07662	-0.03645
0.01858	-0.05405	-0.06756	0.0665	0.1003
-0.04024	-0.08105	0.01164	-0.01214	0.04122
0.07402	-0.04504	-0.09225	0.007379	0.07205
0.01104	0.01014	-0.02957	-0.07588	0.02348
0.08333	0.01519	-0.05589	-0.02765	-0.01276
0.01981	0.07299	-0.002872	-0.07295	-0.03017
0.03742	0.03669	0.01867	-0.01147	-0.06648
-0.01509	0.06057	0.03961	-0.01966	-0.03858
-0.02576	0.003734	0.05775	0.4	-0.2104
-0.2986	0.2106	0.04297	-0.04549	0.07336
0.02166	-0.355	0.06816	0.351	-0.1386
-0.07606	0.02627	-0.08146	-0.02973	0.2677
0.03526	-0.3148	0.04578	0.08518	-0.01052
0.0921	0.02805	-0.2031	-0.0916	0.2481
0.03091	-0.07767	0.002565	-0.09796	-0.03558
0.1598	0.1172	-0.1749	-0.0827	0.05684
0.002383	0.09535	0.0512	-0.1279	-0.1256
0.1081	0.1084	-0.02742	-0.005103	-0.08607
-0.06887	0.09969	0.1255	-0.05313	-0.1119
-0.004169	0.00416	0.07341	0.08327	-0.07126
-0.1209	0.01104	0.09985	0.03186	0.001536
-0.05985	-0.09163	0.04208	0.1129	0.01981
-0.07888	-0.05159	-0.01158	0.04642	0.09352

Şekil 5.19 : Simulasyon sonuçları (Analog veri karşılığı)

5.2.2 Doğrusal öngörü ile kodlama analiz kısmının gerçeklenmesi

Sentez işlemi yapıldıktan sonra sentetik ses RF katına aktarılıp iletimi sağlanır. Bu iletimden sonra LPC algoritmasının analiz kısmına aktarılır. Akif Özkan'ın hazırlamış olduğu kodlar incelendiğinde analiz kısmının Model Composer kütüphanesi ile tasarlanamayacağı tespit edilmiştir. Tasarımın devamı ve tutarlılığı için LPC analiz kısmı Model Composer ve Simulink kütüphaneleri kullanılarak Akif Özkan'ın kodlarını inceleyerek blok tasarımı olarak tasarlanmak istenmiştir. Bu kapsamda Model Composer ile tasarlanan sentez kısmında da analiz kısmının uygunluğu için değişikliklere gidilmiştir.

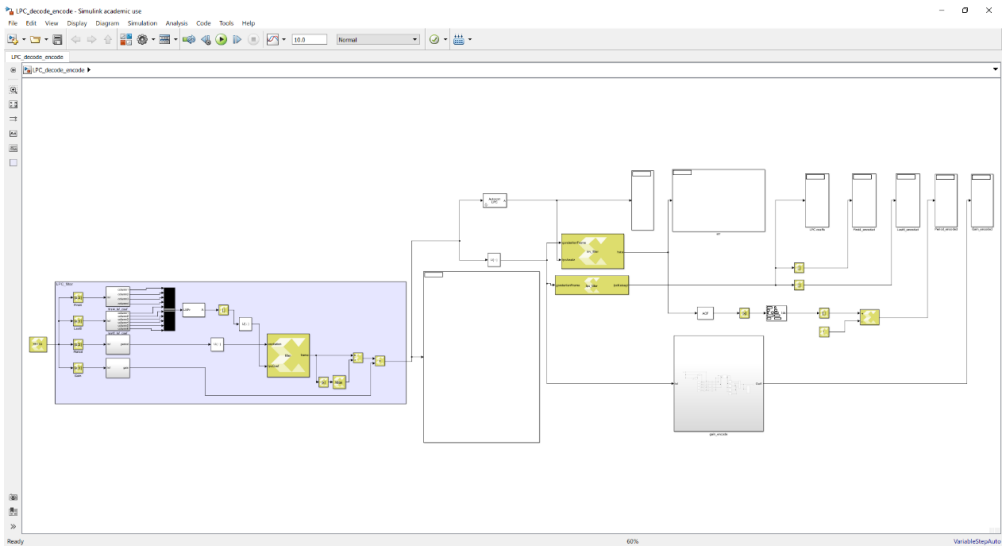
5.2.2.1 Sentez kısmında yapılan değişiklikler

Tasarımın bütün olarak çalışması için sentez kısmını analiz kısmına uydurma gerekliliği doğmuştur. Yapmış olduğumuz sentez kısmındaki değişiklikler şu şeklide sıralanabilir; bit dizisi ayrıştırma, doğrusal tayf frekansları (Linear Spectral Frequencies-LSF)-LPC dönüşümü, kod kitapları düzenlemesi.

Bit dizisi ayrıştırma düzenlemesi için seçilen 6 bit 4 bit ayrıştırma yerine daha tutarlı olan 5 bit 5 bit ayrıştırma yöntemi seçilmiştir. Kod kitapları içeriği 80 satırdan 64 satıra düşürülmüştür.

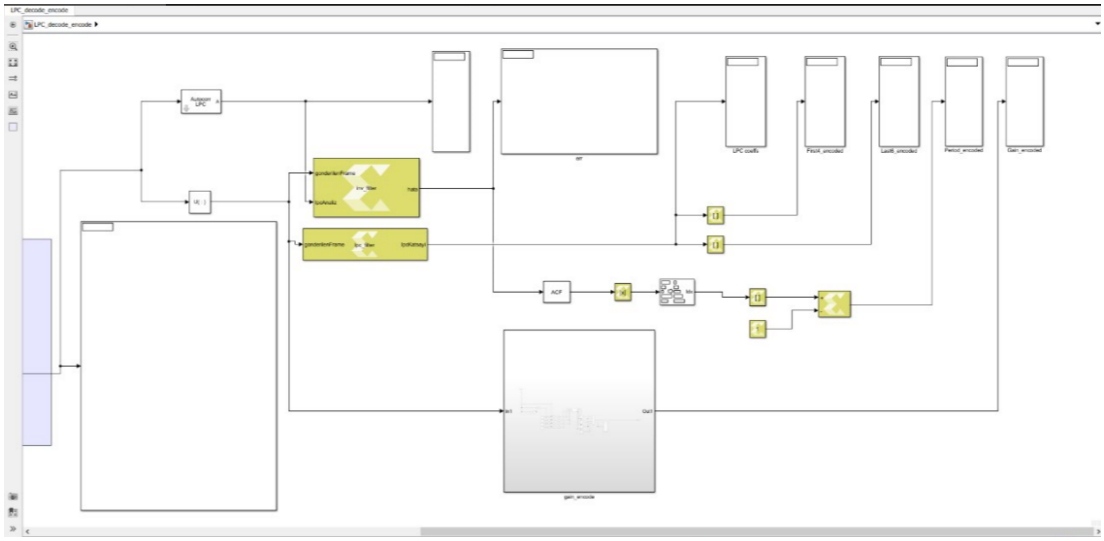
Kod kitapları içerikleri LPC katsayıları yerine LSF katsayıları ile değiştirilmiştir.

Kod kitaplarından seçilen LSF katsayıları filtrede kullanılamayacağı için Simulink'de bulunan lsf to lpc dönüştürücü blok kullanılarak LPC katsayıları elde edilmiştir.



Şekil 5.20 : LPC sentez ve analiz kısmı tamamı

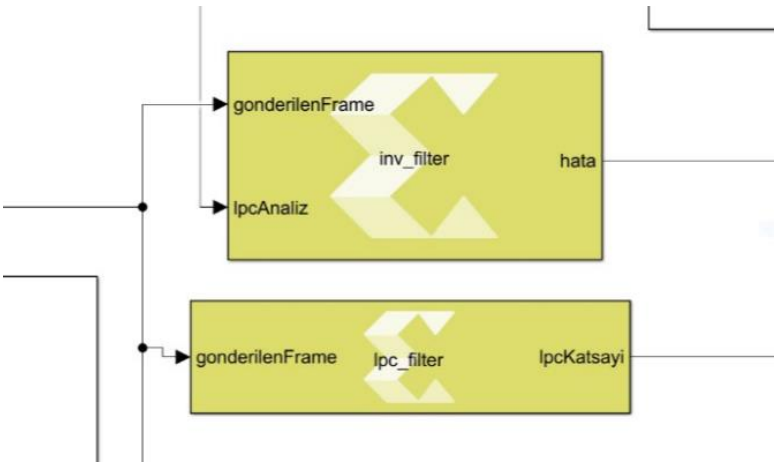
5.2.2.2 Analiz işlemleri



Şekil 5.21 : LPC analiz şeması

LPC katsayılarının elde edilmesi

Sentez kısmından çıkan sentetik ses öncelikle öz ilişki bağıntısı ile LPC katsayıları üretecek bloğa gönderilir. Bu blokta üretilen LPC katsayıları tahmini olarak adlandırılabilir. Bu analiz LPC katsayıları ile sentetik ses bir ters filtreye sokularak hata üretilir. Aynı zaman zarfında sentez kısmında kullanılan filtrenin tersi kullanılarak kullanılan asıl LPC katsayıları elde edilir. Bu filtreler Model Composer ortamında bulunmadığından custom block olarak tasarlanıp sisteme aktarılmıştır. LPC katsayılarını kod kitaplarında aratmak için tekrardan LPC-LSF dönüşümü yapılır ve bu LSF değerleri kod kitaplarında taratılmak üzere saklanır.



Şekil 5.22 : Özelleştirilmiş blok tasarımları

lpc_filter.cpp inv_filter.cpp

```cpp
#include "lpc_filter.h"
void lpc_filter(double lpcKatsayi[10],
                const double gonderilenFrame[11])


{
    double bcc,acc,ccc;
    for(int k=0; k<10;k++){
        bcc=0;
        acc=0;
        bcc +=gonderilenFrame[k+1];
        ccc=0;
        for(int i=0;i<11;i++){
            if(k-i>0){
                ccc +=lpcKatsayi[i] * gonderilenFrame[k-i];
            }
        }acc =(bcc+ccc)/gonderilenFrame[0];
        lpcKatsayi[k]= -acc;
    }
}
```

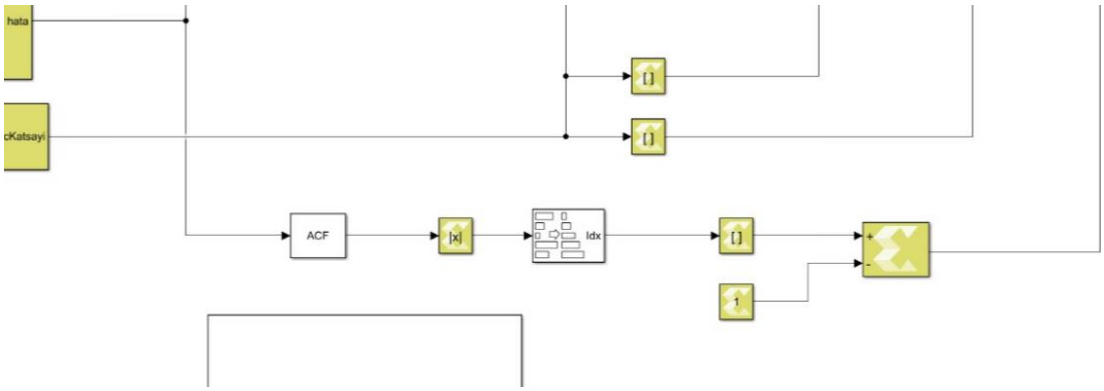
lpc_filter.cpp inv_filter.cpp

```cpp
#include "lpc_filter.h"
void lpc_filter(double lpcKatsayi[10],
                const double gonderilenFrame[11])


{
    double bcc,acc,ccc;
    for(int k=0; k<10;k++){
        bcc=0;
        acc=0;
        bcc +=gonderilenFrame[k+1];
        ccc=0;
        for(int i=0;i<11;i++){
            if(k-i>0){
                ccc +=lpcKatsayi[i] * gonderilenFrame[k-i];
            }
        }acc =(bcc+ccc)/gonderilenFrame[0];
        lpcKatsayi[k]= -acc;
    }
}
```

Şekil 5.23 : Özelleştirilmiş blok C kodları

Perde bilgisinin elde edilmesi

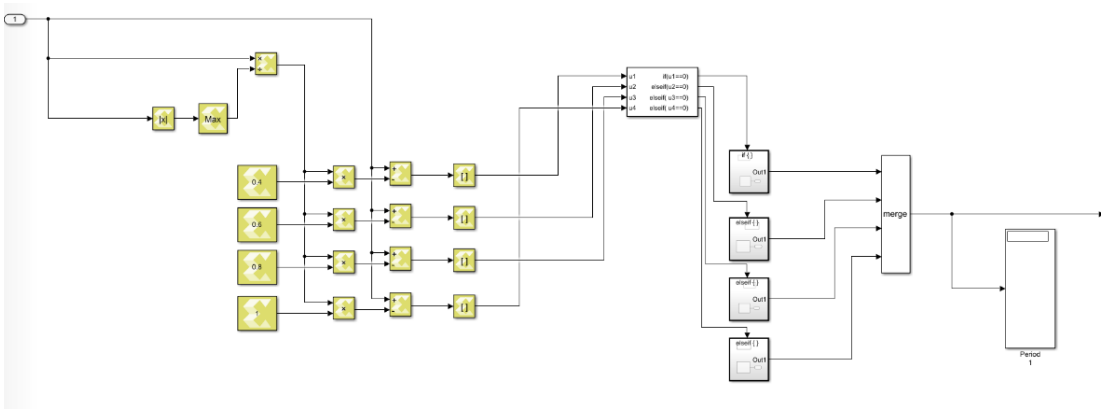
Üretilen hata değerlerinin özilişki bağıntı değerleri hesaplanır. Özilişki bağıntısı tanım olarak sinyalin kendi örnekleri arasındaki benzerliğin bir ölçütüdür. Bu ölçüt sinyalin zamanda ötelenmiş hali ile kendisinin karşılaştırılması ile oluşur. En yüksek olduğu öz ilişki değeri sinyallerin en benzer olduğu duruma denk düşmektedir. Bu bağlamda bu öteleme bilgisi bize periyod bilgisini vermektedir. Üretilen hata sinyalinin mutlak özilişki değerli artan şekilde sıralanırken aynı zamanda indeks değerleri de sıralanmıştır. En yüksek değer en son değerde indeks değeri 1 olacaktır. Bu ilk sinyaller arasındaki özilişkidir. Bundan bir önceki indeks değerinin 1 eksiği bize periyod bilgisini verecektir.



Şekil 5.24 : Perde (periyod) bilgisinin elde edilmesi

Kazanç bilgisinin elde edilmesi

Kazanç hesabı yapılırken, analiz kısmına gelen sentetik ses verisi normalize edilir. Bu işlem ise ses verisinin her değerini mutlak en yüksek değerine bölünmesi şekkinde tanımlanır. Ardından normalize edilmiş veri kazanç kod kitabında bulunan 4 farklı değerle tek tek çarpılır ve sentetik ses verisinden çıkartılır. Bu 4 farklı çıkartma işlemlerinden mutlak minimum olan değerdeki kazanç değeri, sinyalin kazanç değerini verecektir.



Şekil 5.25 : Kazanç bilgisinin elde edilmesi

Bit dizisi üretilmesi

Bulunan değerler kod kitaplarında taratılacaktır. Bu tarama işlemi için Simulink ve Model Composer ortamı yeter düzeydedir fakat tasarıma çok karmaşıklık getirmektedir. Bu sebepten dolayı kod kitaplarında değer taraması tasarımda yer almamaktadır. Teorik olarak anlatmak gerekirse taranacak değerlerin indeks değerleri alınıp bit dizisi olarak birleştirildikten sorna deşifreleme bloğuna aktarılıp deşifre ediecektir. Tasarım Akif Özkan'ın kodları incelenerek tasarlanmış olup simulasyonu yapılmştır. Sistemin doğru çalıştığı tespit edilmiştir.

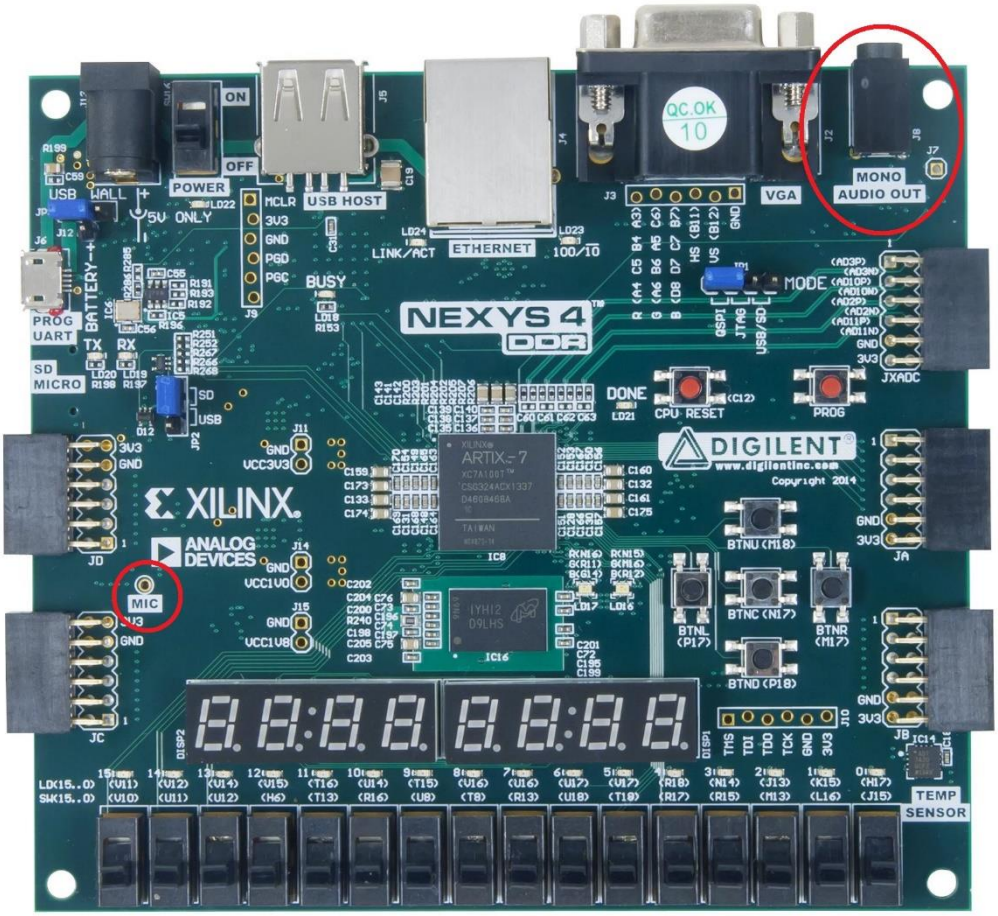
6. SES ŞİFRELEME ALGORİTMASININ ALANDA PROGRAMLANABİLİR KAPI DİZİLERİ ÜZERİNDE GERÇEKLENMESİ

Bu bölümde Model Composer ortamında üretilecek olan şifreleme algoritmasına ait donanımın Xilinx firmasının araçlarından faydalanarak nasıl gerçekleme yapılacağından bahsedilmiştir.

Proje kapsamında Model Composer ortamında üretilen ses şifreleme donanımının Xilinx araçları ile FPGA üzerinde gerçeklenerek FPGA kartının bir şifreleme modülü olarak kullanılması amaçlanmıştır. Bu bağlamda, FPGA kartı ortamdan sesi alacak, gerekli şifreleme işlemlerinden sonra güvenli bir şekilde iletilmesini sağlayacaktır. Bu amaç doğrultusunda, uygulama yapılacak kartın seçilmesi, seçilen kartın tanınması, Xilinx'in sunduğu imkanlar doğrultusunda yapılacak tasarımlar için kullanılabilecek araçların belirlenmesi ve tanınması, ses şifreleme için uygun donanım ortamının sağlanması, hazırlanan şifreleme algoritmasının FPGA kartı üzerinde gerçeklenmesi amaçlanmıştır.

6.1 Nexsys4 DDR

Bir DIGILENT firması ürünü olan Nexsys4 DDR, Xilinx tarafından sağlanan Artix 7 FPGA tabanlı, kullanıma hazır bir sayısal devre geliştirme platformudur[22]. İçinde bulundurduğu işlevsel donanımlar ve çevre birimleri sayesinde basit kombinezonsal devre yapılarından karmaşık işlemci yapılarına kadar birçok uygulama için uygun ortam sağlamaktadır. Çevre birimi olarak bir Mikroelektrik-Mekanik Sistem(MicroElectrical-Mechanical System, MEMS) dijital mikrofon, sıcaklık sensörü gibi birçoğunu içinde bulunduran Nexys4 DDR başka bir bileşene ihtiyaç duymadan birçok tasarımın gerçeklenebilmesine imkân sağlamaktadır. Aşağıda Nexys4 DDR kartı fotoğrafı verilmiş, mikrofon ve ses çıkışı belirtilmiştir.



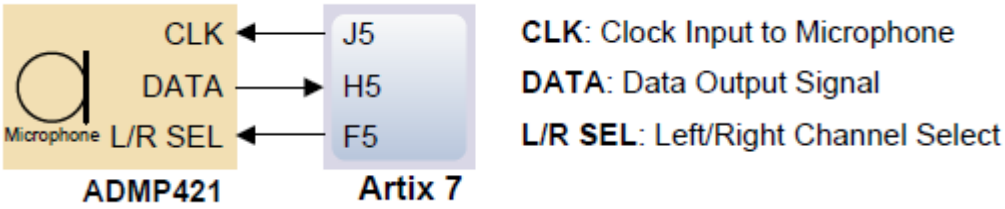
Şekil 6.1 : Nexys4 DDR [22].

Nexys4 kartı Xilinx'in yeni Vivado Design Suite ve ISE araç setleri ile uyumlu çalışmaktadır. Xilinx Vivado Design Suite için ücretsiz Webpack paketi sunmakta ve bu sayede tasarımlar ek bir maliyet gerektirmeden kart üzerinde uygulanabilmektedir.

6.1.1 Mikrofon

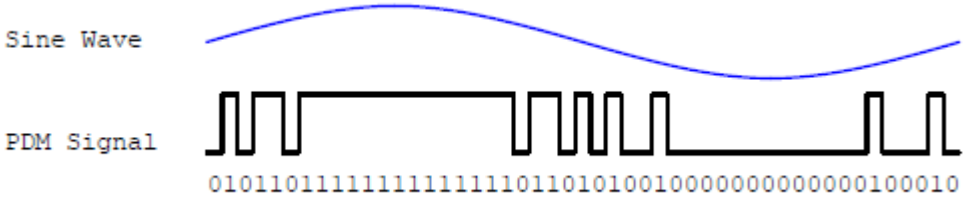
Seçilen Nexys4 kartı bir ses işleme aracı olarak kullanılacaktır. Bunun için önemli olan donanımlar sesin sayısal veriye aktarılmasını sağlayacak bir mikrofon ve yine şifrelenen sesin sayısal veriden ses olarak dış ortama aktarılmasını sağlayacak ses çıkışına ihtiyaç duyulacaktır.

Nexys4 kartı üzerinde Analog Devices firmasına ait ADMP421 yongası içeren çok yönlü MEMS mikrofon bulunmaktadır [22]. Mikrofon tarafından sayısallaştırılmış veri darbe yoğunluğu modülasyonlu (Pulse Density Modulated - PDM) formatta olmaktadır. Aşağıda mikrofonun yapısı gösterilmiştir.



Şekil 6.2 : Mikrofonun Yapısı [22].

PDM veri bağlantıları cep telefonu ve tabletlerde sık kullanılan bir yöntemdir. PDM sinyalinin frekansı 1MHz ile 3MHz aralığında olmaktadır. Bir PDM bit dizisinde "1" pozitif dürtüye(pulse) denk gelirken "0" değerleri negatif dürtüye denk düşmektedir. Aşağıda bir sinüs dalgasının PDM sinyaliyle nasıl temsil edildiği gösterilmektedir.



Şekil 6.3 : Sinüs Dalgasının PDM Sinyaliyle Temsil Edilmesi [22].

Mikrofonun saat girişi örnekleme frekansına ve kullanıcı uygulamanın hassasiyet gereksinimine göre 1 MHz ile 3.3 MHz arasında olabilmektedir.

6.1.2 Ses Çıkış Portu

Kart üzerinde bulunan ses çıkış portu derecesi dört olan bir Sallen-Key Butterworth alçak geçiren filtre ile içermektedir [22]. Filtrenin girişi sinyal genişlik modülasyonlu (Pulse Width Modulated - PWM) veya FPGA mikrofonu tarafından üretilen PDM formatında olabilmektedir. Alçak geçiren filtre girişindeki sayısal veriyi analog voltaj değerlerine dönüştürmek için bir yeniden yapılandırma filtresi olarak görev yapar.

Genel olarak FPGA yapısına bakılacak olursa girişten alınan sinyalin herhangi bir değişikliğe uğratılmadan çıkışa verilebileceği görülmektedir.

6.2 Ses Sinyalinin Şifrelemeye Hazır Hale Getirilmesi

Proje amaç itibariyle sesin şifrelenmesini sağlayacak algoritmanın Model Composer ortamında tasarlanarak uygun donanımın oluşturulmasını hedeflemektedir. Oluşturulan donanımlar sayısal veri üzerinde çalışacağından ses sinyalinin analog ortamdan alınarak sayısal veriye dönüştürülmesi ve yine aynı şekilde şifrelenmiş sayısal ses verisinin analog ortama aktarılması gerekmektedir. Bu bağlamda Nexys4 kartı üzerinde bulunan mikrofon ve ses çıkışlarının tasarımın genel hedeflerine uygun şekilde kullanılmasını sağlayacak donanımların tasarlanması bu bölümün konusunu oluşturmaktadır.

Bölüm 7.1'de belirtildiği gibi Nexys4 kartı üzerinde bulunan mikrofon analog ses sinyallerini PDM formatında sayısallaştırmaktadır. Yine kart üzerinde bulunan ses çıkış portu da sayısal PDM sinyalini analog hale getirebilmektedir. Kartın sunduğu bu avantaj doğrultusunda mikrofondan alınan veri üzerinde herhangi bir ek işlem yapmadan çıkışa gönderilebilecektir. Bu bağlamda ilk adım olarak mikrofon için tasarlanan bir sürücü devre ile mikrofondan alınan verinin ses çıkış portu üzerinden dış ortama aktarılması hedeflenmiştir.

Tasarımın ilk aşamasında mikrofon için bir saat sinyali oluşturulması hedeflenmiş, fakat kart üzerinde bulunan kristal saat üreticisinin mikrofon örnekleme periyoduna uygun frekansta saat üretmeye imkân vermediği saptanmıştır. Bu nedenle mikrofon sürücüsü için kristalden alınan 100 MHz frekansında saat periyodunu 3.125 MHz'e düşürecek bir frekans bölücü tasarlanmış böylece mikrofonun örnekleme frekansının üst limitine yakın bir saat işareti oluşturularak alınacak örneğin imkânlar dâhilinde en yüksek kalitede olması hedeflenmiştir.

Tasarım için gerekli diğer tanımlamalar ve bağlantılar yapıldıktan sonra tasarım tamamlanmış ve Vivado ortamında sentez ve implemantasyon adımları tamamlanarak FPGA üzerinde gerçeklenmiştir.

Tasarlanan donanım FPGA üzerine aktarıldıktan sonra karta bir hoparlör bağlantısı yapılmış ve mikrofondan alınan sesin hoparlöre iletildiği görülmüştür. Böylece kart üzerinde analog ses işareti sayısallaştırıldıktan sonra tekrar analog ortama dönülmüş ve ses iletimi sağlanmıştır. Bununla birlikte Nexys4 DDR kartı ses şifreleme algoritmalarının gerçeklenmesi için uygun hale getirilmiştir.

Kullanıcı uygulamanın getireceği gerekliliklerle birlikte ihtiyaç halinde mikrofon sürücüsü üzerinde küçük değişiklikler ve eklemeler yapılarak genel ihtiyaca cevap verecek hale getirilebilecektir.

7. GERÇEKÇI KISITLAR, SONUÇLAR VE ÖNERİLER

7.1 Çalışmanın Uygulama Alanı

Bu çalışmanın uygulama alanları aşağıda belirtilmiştir:

- Simulink ortamının sunduğu görsel arayüz ve Model Composer kütüphanesinin sunduğu hazır blokların kullanımı ile kırmık üstü sistem tasarımını ve tasarlanan sistemin Simülink ortamında simülasyonu yapılarak test edilmesini kolaylaştırmak.
- Tasarlanan sistemdeki hataların ayıklanmasını ve sistemin optimize edilmesini, Simulink'in gelişmiş arayüzünü kullanarak basitleştirmek.
- Model Composer ve Simulink ile üretilen otomatik kodların Vivado aracı ile FPGA üzerinde gerçeklenmesi.

7.2 Gerçekçi Tasarım Kısıtları

7.2.1 Maliyet

Projenin yapılabilmesi için gerekli olan ilk materyal, bir FPGA kartıdır. FPGA kartı ihtiyacı, fakülte tarafından, gruptaki her öğrenciye bir adet verilecek şekilde giderilecektir. FPGA kartı üzerinde kırmık üstü sistem tasarımı yapılabilmesi için, Xilinx'in Vivado HL System Edition geliştirme ortamı ve Model Composer aracı kullanılacaktır. Vivado HL System Edition ve Model Composer için gerekli olan lisanslar, gruptaki her öğrenciye, fakülte tarafından sağlanacaktır.

Yeni başlayan bir mühendisin maaşı $250 olarak varsayılmıştır.

(4 kişi x $250 x 6 ay = $6000)

Nexys 4DDR FPGA kartının maliyeti $360'dır.

Xilinx Vivado HL System Edition lisans maliyeti $3495'dır.

MATLAB yıllık lisans maliyeti $940'dır.

Projenin toplam maliyeti $10.795 olarak hesaplanmıştır.

7.2.2 Standartlar

Bu projede amaç Model Composer ve Simulink ortamında güvenilir ve düşük maliyetli bir kırmık üstü sistem tasarlamaktır. Tasarlanan sayısal sistem genel hatlarıyla IEEE standartlarına uygun olacaktır. Bununla birlikte tasarlanacak AES algoritması için FIPS yönergelerine uyulmuştur.

7.2.3 Sosyal, çevresel ve ekonomik etki

Kırmık üstü sistem yapılırken özellikle görüntü işleme gibi özel konularda, algoritma geliştirme becerisinin yanında, tasarlanan algoritmanın donanım ortamında uygulanabilmesi için donanım tanımlama dili bilgisi de gerekmektedir. Simulink ile tasarlanan kırmık üstü sistem, Model Composer ile kolaylıkla donanım ortamına aktarılabilecektir. Bu sayede iş yükü hafifletilecektir.

7.2.4 Sağlık ve güvenlik riskleri

Herhangi bir sağlık ve güvenlik riski bulunmamaktadır.

7.3 Sonuçlar

Bu bitirme tasarım projesinde, Model Composer ortamında bir şifreleme algoritmasının gerçeklenmesi için çalışılmıştır. Gerçeklenecek şifreleme algoritması olarak AES seçilmiştir. AES, şifreleme ve şifre çözme işlemlerini gerçekleştirirken yapılan dönüşümlerin birbirinin tersi olması tasarımda kolaylık sağlamaktadır. AES, 128, 192 ve 256 bit boyutundaki giriş, çıkış, durum matrisi ve anahtarlarla çalışabilmektedir. 128 bit ile yapılan tasarım daha hızlı olduğundan bu proje için seçilmiştir. Model Composer ile AES algoritmasının simülasyonları yapılmış ve VHDL ile de donanım üzerinde gerçeklenmiştir.

Şifrelenen konuşma verisi GSM hattında taşınabilmesi için analog veriye çevirilmesi gerekmektedir. Bu tasarım için en uygun algoritma olarak LPC algoritması seçilmiştir. LPC girişine gelecek olan 16 bitlik ses verisini ayrıştırıp kod bloklarından gerekli parametreleri alacaktır. Alınan parametreler ışığında LPC sentez kısmı gerçeklenir. LPC katsayıları ile tasarlanan filtreden periyod bilgisi geçirilerek zaman domeninde konvolüsyon edilir. Daha sonra veri, kazanç bilgisi ile çarpılarak çıkışa aktarılır.

Analog hale dönüşen veri GSM hattına aktarılır ve LPC'nin analiz kısmına giriş olarak verilir. Analiz kısmında ise gelen ses verisinden ilk önce LPC katsayıları ahmin edilerek hata bulunur. Bu hata ile verinin periyot bilgisi, kazanç değeri ve en uygun LPC katsayıları saptanır. Bu değerler kod bloklarında taratıldıktan sonra bit dizisi halinda dijital veri olarak şifre çözüm bloğuna iletilir. Tasarımın gerçeklenmesi kısmında öncelikle sentez ve analiz kısmının Model Composer üzerinde IP'leri üretülmek istenmiştir. Fakat Model composer ve Simulink ortamının LPC algoritmasını gerçeklemek için yeterli düzeyde materyalleri, blok tasarımları bulunmamaktadır. Sistemin çalışıp çalışmadığını gözlemlemek amacıyla 2. Aşama olarak sistemin simulasyonu yapılmak istenmiştir. Tekrardan simulink ve Model Composer kütüphanesinin simulasyon için dahi yeterli blok tasarımları bulunmamaktadır. Buna çözüm olarak Model Composer'ın özelleştirilmiş blok tasarımı yöntemi kullanılarak sistem tasarımı için sisteme özelleştirilmiş blok tasarmları eklenmiştir. Bu blok tasarımlarında IP üretimi mümkün değildir. Simulink ve Model Composer ortamlarının birlikte kullanılması sonucu sistem tasarımı yapılmıştır. Yapılan sistem simule edilmiş ve sonuçların kaynak kodlarıyla karşılaştırılması sonucu doğru çıktığı gözlemlenmiştir.

Projenin gerçekleneceği platform olarak Nexys4 DDR FPGA kartı seçilmiştir. Nexys4 DDR üstünde tümleşik olarak MEMS mikrofon ve ses çıkış portu bulunmaktadır. Bu sayede ek bir donanıma ihtiyaç duyulmadan, ortamdan analog olarak alınan ses sinyali sayısallaştırılarak işlenmeye hazır hale getirilebilmektedir.

Yapılan çalışmada, Nexys4 DDR üzerinde bulunan mikrofondan 3.125 MHz örnekleme frekansıyla sayısallaştırılan ses sinyali AES ile şifrelenmiş, şifrelenen ses kart üstünde bulunan ses çıkışına verilmiş, şifrelenmiş ses sinyalinin deşifre işlemleri aynı kart üzerinde gerçekleştirilmiş ve şifresi çözüldükten sonra ses çıkışına verilmiştir.

LPC algoritması içerisinde bulunan sentetik ses üreten donanım Model Composer ortamında gerçeklenmiş ve testleri yapılmıştır. LPC algoritması içerisinde bulunan sentetik sesten orijinal ses üretimi yapan donanım Model Composer ortamında Simulink bloklari ve özelleştirilmiş bloklar kullanılarak gerçeklenmiş ve testleri yapılmıştır. Yapılan testler sonucunda sistemin düzgün çalışıtığı gözlemlenmiştir fakat Simulink blokları ve özelleştirilmiş bloklar sebebiyle sistemin donanıma aktarılması için IP üretimi yapılamamaktadır.

Model Composer'ın yapılan tasarımlarda sadece Xilinx kütüphanesine ait bloklar kullanılmasına rağmen FPGA üzerinde gerçeklemeye uygun IP üretimini gerçekleştirememiştir. Model Composer ortamında yapılacak tasarımlarda hedeflenen davranışa ulaşmak için özellikle karmaşık işlemlerde, yapılacak tasarımın genel davranışına olabildiğince yakın hazır blok yapılarından faydalanılarak tasarımın mümkün olduğunca basit tutulması gerektiği mevcut kütüphane imkanlarıyla tecrübe edilmiştir. Model Composer kütüphanesinde hazır blok olarak sunulmayan bir algoritmanın temel alt bloklar kullanılarak tasarlanmasında, hedeflenen devre davranışı karmaşıklığı arttıkça veya hedeflenen davranışa ulaşmak için kullanılacak blokların seviyesi temele indikçe kullanılacak blok sayısı arttığı gibi iş yükü de fazlaca artarak asıl olarak hedeflenen iş yükünün azaltılması hedefinden uzaklaşılmaktadır. İş yükü göze alınarak tasarımın yapılmasına karar verilmiş olsa bile, iterasyonel olarak bir hesap yapılması gereken durumlarda, veri yolunun giriş parametreleriyle belirlenecği durumlarda, bir önceki çıkış değerinin belirli durumlarda giriş olarak kullanılacağı tasarımlarda mevcut Model Composer kütüphanesi bloklarının Simulink ortamında simulasyona imkan vermiş olsa bile IP üretimine geçilemediği tecrübe edilmiştir.

7.4 Geleceğe Yönelik Öneriler

Bu çalışma, Model Composer'ın sayısal sistem tasarımı alanındaki uygunluğunu test etmek amacıyla yapılmıştır. Bu proje kapsamında, Model Composer'da bulunan hazır blokların bir kısmından yararlanılmıştır. Bu nedenle, Xilinx tarafından sunulan ve görüntü işlemeye yönelik hazırlanmış blokların etkinliğinin, ileride yapılan tasarımlarda test edilmesi önerilmektedir. Yapılan tasarımı donanıma aktarmada şu anda yaşanan sıkıntılar giderildiği taktirde, bu ortamda yapılacak tasarımların temelini oluşturacaktır.

KAYNAKLAR

[1] **Xilinx**. (2018). , Model Composer User Guide

[2] **MathWorks**. (2015). , Simulink User's Guide, pp: 186-187

[3] **Xilinx**. (2018). , Model Based Design Using Model Composer

[4] **Wakerly, J. F.** (2006). Digital Design Principles and Practices (4th ed.). United States of America: Pearson Education Inc., pp. 183

[5] **Bakshi, U. A., Godse, A. P.** (2009). Analog and Digital Electronics (1st ed.). India: Technical Publications Pune, pp. 301

[6] **FIPS 197.** (2001). Advanced Encryption Standard. Nationa Institute of Standard and Technology, pp. 1

[7] **Boneh, D.** (2014). Cryptography I. Retrieved from Stanford University, https://class.coursera.org/crypto-009

[8] **Xilinx.** (2008). MicroBlaze Processor Reference Guide, pp: 10

[9**] Kondoz, A. M.,** (2004). "Digital Speech:Coding for Low Bit Rate Communication Systems", John Wiley&Sons.

[10] **FS 1016**, (1990). "The Proposed Federal Standard 1016 4800 bps Voice Coder: CELP", Joseph P. Campbell Jr., Thomas E. Tremain and Vanoy C. Welch. Speech Technology Magazine, pp. 58-64.

[11] **Jayant, N. S. and Noll, P.**, (1984). "Digital Coding of Waveforms: Principles and Applications to Speech and Video", Prentice-Hall, Englewood Cliffs, New Jersey.

[12] **Bradbury**, (2000). Linear predictive coding. University of Biritish Colombia, J..

[13] **Özkan, A.**, (2011). *GSM Ağı Üzerinden Güvenli Ses İletimi.* (Lisans Tezi), İ.T.Ü. Elektrik Elektronik Fakültesi, İstanbul.

[14] **Oppenheim, A. V. & Schafer, R. W.**, (1975). "Digital Signal Processing", Prentice Hall.

[15] **Kırçiçek, Y.**, (2007). *Doğrusal Öngörü İle Konuşma İşareti Kodlayıcısı Tasarımı*. (Yüksek Lisans Tezi). Y.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[16] **DELLER, J. R., PROAKIS, J. G. & HANSEN, J. H. L.**, (1999). Discrete-Time Processing of Speech Signals, IEEE Press Classic Reissue, pp. 266-330.

[17] **Meral, O.**, (2018). *DOĞRUSAL ÖNGÖRÜLÜ KODLAMA VE ADAPTİF ALGORİTMA TABANLI KONUŞMACI TANIMA.* (Yüksek Lisans Tezi) Elektrik-Elektronik Mühendisliği Anabilim Dalı Elektrik-Elektronik Mühendisliği Programı, İstanbul Üniversitesi Fen Bilimleri Enstitüsü, İstanbul

[18] **Bae, S. H.**, (2004). "LPC10 2.4 kbps federal Standard in speech coding",Georgia Institue of Technology, accessed from http://users.ece.gatech.edu/~juang/8873/Bae-LPC10.ppt in 20.05.2011.

[19] **McLoughlin, I.**, (2009). "Applied Speech and Audio Processing", Cambridge University Press, New York.

[20] **Tunçay, S.**, (2012), *Bir GSM Modemin FPGA Üzerinde Gerçeklenmesi*. (Lisans Tezi). İ.T.Ü Elektrik Elektronik Fakültesi, İstanbul.

[21] **Gomez, P.**, (2004). *Speech Coding & Linear Predictive Coding (LPC)*. (PhD Thesis). Florida International University, Miami.

[22] **Digilent.** (2016). Nexsy4 DDR™ FPGA Board Reference Manual,pp 1-28

[23] **Xilinx.** (2018). *Vivado Design Suite User Guide Getting Started*, pp 5

[24] **Xilinx.** (2018). Vivado Design Suite Tutorial

[25] **Reese, R. B., Thornton, M. A.**, (2006). Introduction to Logic Synthesis Using Verilog HDL. Morgan & Claypool Publishers.

ÖZGEÇMİŞ

Ad-Soyad **:** Salim Uslu

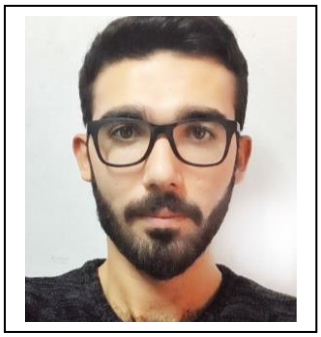
Doğum Tarihi ve Yeri **:** Burdur, 12/07/1995

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

Lise : Denizli Erbakır Fen Lisesi; 2009 - 2019

Lisans : İstanbul Teknik Üniversitesi, Elektronik ve Haberleşme Mühendisliği; 2013 - 2019

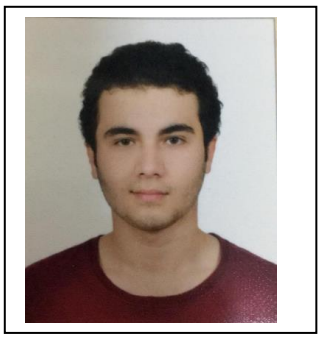
ÖZGEÇMİŞ



Ad-Soyad : Mustafa Mert ESEN
Doğum Tarihi ve Yeri : 27.07.1996 NİĞDE
E-posta : esenmu@itu.edu.tr

Niğde ili merkez ilçesinde 1996 yılında doğdum. 2010 Yılında ilk ve orta öğretimimi aldığım Niğde Merkez Atatürk İlköğretim Okulundan Mezun olarak Niğde Fen Lisesine başladım. Lise eğitimimi tamamladıktan sonra lisans eğitimime 2014 yılında İstanbul Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümünde başladım.

ÖZGEÇMİŞ



Ad-Soyad : İbrahim GÜVEN
Doğum Tarihi ve Yeri : 05.09.1997 BİLECİK
E-posta : ibrahimguven148@gmail.com

Bilecik ili merkez ilçesinde 5 Eylül 1997 yılında doğdum. Ilk ve ortaokul eğitimimi Atatürk İlköğretim Okulu'nda tamamladım. Lise eğitimimi Bilecik Anadolu Öğretmen Lisesinde tamamladım. Daha sonrasında İstanbul Teknik Üniversitesi Elektronik ve Haberleşme Bölümü lisans eğitim programını kazanarak İstanbul'a geldim.