**İSTANBUL TEKNİK ÜNİVERSİTESİ**
**ELEKTRİK – ELEKTRONİK FAKÜLTESİ**

# GÜVENLİ TELEKONFERANS SİSTEMİNİN FPGA ÜZERİNDE TASARIMI VE GERÇEKLENMESİ

**BİTİRME ÖDEVİ**

**Onur TİRYAKİOĞLU**

**040030435**

**Bölümü: Elektronik ve Haberleşme Mühendisliği**

**Programı: Elektronik Mühendisliği**

**Danışmanı: Yrd. Doç. Dr. Sıddıka Berna Örs YALÇIN**

**MAYIS 2008**

**İÇİNDEKİLER**

**ÖNSÖZ**

İlgi alanlarımın başında bulunan sayısal tasarım ve kriptoloji konularının ikisiyle birden, bitirme tasarım projemde uğraşmış ve sonuca ulaşmış olmaktan mutluluk duyuyorum. Kriptoloji dalının benim için sadece ilgi alanı olmaktan çıkıp bundan sonraki hayatımın bir parçası olmasına vesile olmuş olan, onlarca öğrencinin danışmanlığını yapıyor olmasına rağmen kısıtlı zamanında bana her zaman vakit ayırıp sorularımı cevaplayan, yol gösteren, destek olan saygı değer danışman hocam Yrd. Doc. Dr. S. Berna Örs Yalçın'a sonsuz teşekkürlerimi sunarım.

Sayısal elektronik derslerini aldığım değerli hocam Prof. Dr. Ece Olcay Güneş'e ve öğrencilere her zaman güler yüzle yardımcı olan bitirme projemde de yardımlarını esirgemeyen Araş. Gör. Yük. Müh. Levent Aksoy'a teşekkür ederim.

Son olarak tüm arkadaşlarıma, kardeşime özellikle anneme ve babama karşılıksız desteklerinden ötürü teşekkürü bir borç bilirim.

**Onur Tiryakioğlu**

**Mayıs 2008**

**KISALTMALAR**

| | |
|---|---|
| **AES** | : Advanced Encryption Standard |
| **DES** | : Data Encryption Standard |
| **FPGA** | : Field Programmable Gate Array |
| **LUT** | : Look-Up Table |
| **MUX** | : Multiplexer (Çoklayıcı) |
| **ROM** | : Read-Only Memory |
| **GF** | : Galois Field |
| **NIST** | : National Institute of Standards and Technology |
| **FIPS** | : Federal Information Processing Standarts |

# TABLO LİSTESİ

# ŞEKİL LİSTESİ

# GÜVENLİ TELEKONFERANS SİSTEMİNİN FPGA ÜZERİNDE TASARIMI VE GERÇEKLENMESİ

## ÖZET

Günümüzde teknolojinin ulaştığı noktada, kişilerin özel hayatlarına çok kolay müdahale edilmekte, mahremiyetleri hiçe sayılarak her türlü kişisel bilgilerine ulaşılabilinmektedir. İnternet üzerinden yapılan her türlü harcama, alışveriş, posta gönderimi veya anlık mesajlaşma uygulamaları dahil ev telefonu veya cep telefonu konuşmaları dahi dinlenilebilmekte kayıt altına alınabilmektedir. Teknolojiden ancak teknolojiyi kullanarak korunabiliriz. Kriptoloji bilimi işte bu yüzden özellikle günümüzde daha da fazla önem kazanmaktadır. Her geçen gün internetteki bilgilerimizi nasıl daha güvenli saklarız diye yeni çalışmalar yapılmakta. Bu çalışma da ev telefonlarımız kullanılarak nasıl güvenli iletişim kurabileceğimiz üzerine bir araştırma yapmakta ve bir çözüm önerisi getirmekte.

Gelişmiş Şifreleme Standardı (AES) en güçlü şifreleme algoritmalarından biri olması sebebi ile bu çalışmada kullanılması uygun görüldü. AES algoritması hızlı, güvenli ve kolay şifreleme sağlayacak FPGA donanım aracı üzerinde tasarlanarak gerçeklendi. Dahası iki ayrı cihaz arasında güvenli iletişim gerçeklenerek evlerimizde de kullanılabilme ihtimalini gözler önüne serdi.

Bu çalışmada ilk olarak AES algoritmasının matematiksel tanımlamaları yapılmakta ve matematik temeli açıklanmaktadır. Matematiksel temel üzerine simetrik blok şifreleme algoritmasının yapısı detaylı olarak şekillerle anlatılmaktadır. Daha sonra FPGA üzerinde nasıl gerçeklendiği, ne kadar yer kapladığı, nasıl çalıştığı anlatılmıştır. En sonunda da genel olarak nasıl bir sistem kurulması amaçlandığı anlatılmış ve kurulacak sistemin yapısı açıklanmıştır. Gerçekleme aşamasında da iki kartın birbirleriyle nasıl iletişim kurduğu açıklanmıştır.

**DESIGNING AND IMPLEMENTING A SECURE TELECONFERENCE SYSTEM ON FPGA CHIP**

## SUMMARY

Today people have a big security weakness in their everyday life. Hackers can reach up to innocent and also careless people's checks, mails, instant messages etc. and some of them can listen to others phone calls. In this technological level we should use cryptologic solutions. People in this industry always work to make solutions better. As a matter of the fact that this paper is about a secure teleconference system that normal people can use it in their home phones.

Advanced Encryption Standard is one of the most secure encryption algorithm now since I prefer to use it in my study. Implementing AES algorithm on FPGA is really fast, secure and simple encrypting solution for my system. AES is implemented on two FPGA Starter Kits and telecommunicated with each other. This showed us, there is a great chance to use it at home.

In this study, first of all AES algorithm's mathematic preliminaries are explained and fundamental operations are shown again in first part. Second part Advanced Encryption Standard is reported with every detail in it and then implementing on FPGA is expressed systematically. At least part, full custom teleconference system tried to be describing with its general structure. And two FPGA board communicated with each other clearly.

# 1. GİRİŞ

Modern kriptoloji genel olarak ikiye ayrılmaktadır, simetrik ve asimetrik kriptoloji. Asimetrik şifreleme açık anahtar ilkesine dayanmaktadır. Gizlenmek istenen metin herkesin bildiği bir anahtar ile şifrelenir ve ancak gizli bir anahtarla çözülebilir. Simetrik algoritmalarda ise tek bir gizli anahtar bulunur; şifreleme ve şifre çözme için bu anahtara ihtiyaç duyulur. Simetrik şifreleme, blok şifreleme ve dizi şifreleme olmak üzere iki ana başlığa ayrılabilir. Bu çalışma kapsamında bir blok şifreleme türü olan Gelişmiş Şifreleme Standardı kullanılmıştır [7].

Gelişmiş Şifreleme Standardı (Advanced Encryption Standardı: AES) 2001'de elektronik verinin saklanması için kullanılmak üzere federal bilgi işleme standardı (Federal Information Processing Standards: FIPS) olarak Amerikan Ulusal Standartlar Ve Teknoloji Enstitüsü (National Institute of Standards and Technology: NIST) tarafından yayınlanmıştır. AES algoritması günümüzde en güvenli simetrik şifreleme algoritmalarından biridir ve yaygın bir şekilde kullanılmaktadır [8]. AES algoritması kendinden önce kullanılan daha kısa bir anahtar uzayına ve daha az güvenli olan Veri Kodlama Standardının (Data Encryption Standart: DES) [9] artık ömrünü tamamlamasından dolayı alternatif olarak yazılmıştır.

NIST 1977 yılında bir simetrik şifreleme algoritması olan DES'i standart olarak belirlemiştir. DES uzun yıllar güvenilir bir algoritma olarak kullanılmış olup yeni bilgisayarların işlem gücü karşısında eski gücünü koruyamamıştır. Bunun sonucu olarak DES algoritması kırılmış yerini daha güvenli olan TDES'e bırakmıştır. DES ve onun türevi olan TDES günümüz bilgisayarlarına dayanmaları imkansız hale geldiği için NIST başka bir standart olan AES'i getirmiştir.

## 1.1 Çalışmanın Amacı

Çalışma AES algoritmasının FPGA üzerinde gerçeklenmesini ve bu algoritma kullanılarak güvenli telekonferans sistemi oluşturulabileceğini önermektedir.

Araç olarak donanım kullanılması çalışmanın başlıca amacını oluşturur. Çünkü donanımsal şifreleme en hızlı, en güvenli ve en doğru şifrelemeyi sağlar. Yazılımsal şifreleme sadece daha ucuza mal edilebilir ve daha esnek yapıya sahip olurlar. Nitekim bu çalışmada da donanım olarak FPGA ve tasarım dili olarak da VHDL seçilmiştir.

İlk olarak AES algoritmasının matematik temellerinden bahsedilmiş, daha sonra nasıl bir yapısı olduğu ve nasıl gerçeklendiği anlatılmıştır. Çalışmanın sonunda da birbiriyle iletişim kurabilen 2 FPGA kartı gerçekleştirilmiştir.

# 2. SONLU ALANLAR ARİTMETİĞİ

## 2.1 Giriş

Normal aritmetik işlemlerde, her işlem sonrasında sonuç uzunluğu değişmektedir. Çarpma veya toplama yapıldığında sonucun gösterileceği bit sayısı da artmaktadır. Bu yönüyle normal aritmetiğin birçok uygulamada kullanılması sakıncalar doğurmaktadır. Bu sebeple Rijndael algoritmasında olduğu gibi kriptografide sonlu alanlar aritmetiğinin kullanımı yaygın olarak yer almaktadır. Sonlu alanlar aritmetiğinde, tanımlı tüm işlemler yine aynı uzayda sonuçlar üretmekte, yani sonucun uzunluğu değişmemektedir. Kriptografide kullanılan 'Galois Alanı' sonlu uzayın bir alt kümesidir.

## 2.2 Galois Alanı

$GF(p^n)$ gösteriminde p Galois Alanının karakteristiği olup kaç farklı eleman içereceğini gösterir. Asal bir sayı seçilir ve Rijndael Algoritmasında bu değer 2'dir. n ise toplam kaç GF(p) elemanı içerdiğini gösterir. $GF(2^8)$ alanında 256 adet farklı eleman bulunur. Bu sayılar her biri 1 bayt ile ifade edilip yapılan işlemlerde n. dereceden indirgeme polinomu ile indirgeme yapılır. Yapılan bu indirgeme ile sonucun yine aynı sonlu alanda oluşması sağlanmış olur.

## 2.3 Galois Alanında Elemanların Gösterimi

Galois Alanında elemanlar ikilik (binary), onaltılık (hexadecimal) ve polinom şeklinde gösterilirler. İkilik ve onaltılık gösterimler bilinen gösterim şekillerindendir. Galois Alanında toplama ve çarpma işlemleri, polinom gösteriminde tanımlanmıştır.

### 2.3.1 Polinom Gösterimi

$b \in GF(2^n)$'nin polinomsal gösterimi şu şekildedir;

$$b(x) = \sum b_i * x^i = b_{n-1} * x^{n-1} + b_{n-2} * x^{n-2} + \ldots + b_1 * x + b_0 \qquad (2.1)$$

Örnek olarak;
8 bitlik bir eleman olan {00011011} elemanı için (2.1) denklemini uygularsak,

$$b(x) = x^4 + x^3 + x + 1$$

sonucu oluşur.

### 2.4 Galois Alanında Aritmetik İşlemler

Galois Alanında toplama ve çarpma işlemleri, karakteristiğinin 2 olması sayesinde özel olarak tanımlanmışlardır. Aşağıda detaylı olarak incelemesi görülmektedir.

#### 2.4.1 Toplama İşlemi

Polinom toplama işleminde aynı üstel değere sahip sayılar toplanırlar ve oluşan ara sonuca modulo 2 işlemi uygulanır. Bu işlemle sonucun sonlu alanda kalması sağlanır. Örneğin;

$$a = \{00010001\} \Rightarrow a(x) = x^4 + 1$$
$$b = \{11010011\} \Rightarrow b(x) = x^7 + x^6 + x^4 + x + 1$$
$$a(x) + b(x) = x^7 + x^6 + 2x^4 + x + 2$$

Modulo 2 uygulandıktan sonra,

$$c(x) = x^7 + x^6 + x$$

haline gelir.
Oluşan c sayısını (c={11000010}) bulmanın başka bir yolu daha vardır. O da a ve b sayılarının her bir bitini özel veya işlemine sokmaktır. Çünkü GF(2) sonlu alanında her bir bit 2 farklı değer alabilir bu da devrelerde lojik 1 ve 0'a karşılık gelmektedir [4].

$$c(x) = a(x) + b(x) = \sum_{i=1}^{n-1} (a_i \oplus b_i) x^i \qquad (2.2)$$

Denklem uygulanırsa,

$c = \{00010001\} \oplus \{11010011\} = \{11000010\}$

c değerinin kolayca bulunduğu görülebilir.
Başka bir örnekle bu gösterimler daha iyi ifade edilebilir [3].

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ (polinom gösterimi)

$\{01010111\} \oplus \{10000011\} = \{11010100\}$ (ikili gösterim)

$\{57\} \oplus \{83\} = \{d4\}$ (on altılı gösterim)

### 2.4.2 Çarpma İşlemi

Rijndael algoritmasında GF($2^8$)'de çarpma işlemi, iki polinomun çarpımının 8. dereceden indirgenemez polinom ile modulo alınması işleminden oluşur. Şöyle söylenebilir ki, 8. dereceden indirgenemez polinomun böleni tektir ve sadece kendisidir. AES için bu sayı polinom gösterimi ile,

$m(x) = (x^8 + x^4 + x^3 + x + 1)$ 'dir.

Onaltılık gösterimle de {01}{1b} sayısıdır.
Örneğin, {57}*{83}={c1} olur. Polinom gösterimi ile çözecek olursak,

$$\begin{aligned}(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) &= x^{13} + x^{11} + x^9 + x^8 + x^7 + \\ &\quad x^7 + x^5 + x^3 + x^2 + x + \\ &\quad x^6 + x^4 + x^2 + x + 1 + \\ &= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1\end{aligned}$$

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ modulo } (x^8 + x^4 + x^3 + x + 1)$$
$$= (x^7 + x^6 + 1).$$

Modulo işlemi nedeni ile sonucun derecesi 8'den küçük olmaktadır. Bu da o sayıyı 2 bayt ile ifade edebilmemizi sağlamış olur. Fakat toplama işlemindeki gibi kolay bir yol çarpma işleminde mevcut değildir. Hatırlayacağınız gibi bit seviyesinde özel veya işlemi ile toplama yapılabilmekteydi. Aynı şekil çıkarma işlemi de toplamanın aynısı olmaktadır. Çarpmaya gelince her bir çarpma işlemi için bu modulo işlemini de içine alan bir çarpma bloğu oluşturmak gerekmektedir [3].

# 3. AES – GELİŞMİŞ ŞİFRELEME STANDARDI

## 3.1 Giriş

Rijndael algoritması NIST tarafından 2000 yılında yeni şifreleme standardı olarak Gelişmiş Şifreleme Standardı ismiyle tescillidir.

NIST tarafından 1997 yılında DES algoritmasının yerine geçmesi için, yeni bir algoritma bulunmasını sağlamak üzere bir yarışma düzenlendi. 1998 yılında 15 farklı algoritma önerildi ve 1999 yılında 5 finalist seçildi. Dört yıl boyunca süren değerlendirme ve eleme süreci sonrasında, 2000 yılında, sonuç açıklandı. NIST, Joan Daemen ve Vincent Rijmen tarafından tasarlanan, Rijndael algoritmasının hiçbir değişiklik talep edilmeden Gelişmiş Kodlama Standardı (AES: Advanced Encryption Standard) olarak kullanılacağını ilan etti [1].

Diğer algoritmalar Rc6, Serpent, Blowfish ve Mars da iyi algoritmalardı ve bugün bazı alanlarda kullanımı görülebilmekte. Rijndael algoritmasında anahtar uzunluğu 128, 192, 256 bit uzunluklarında değişmektedir. Bu çalışmada 128 bitlik anahtar uzayından 128 bit uzunlukta üretilen anahtar kullanılmaktadır.

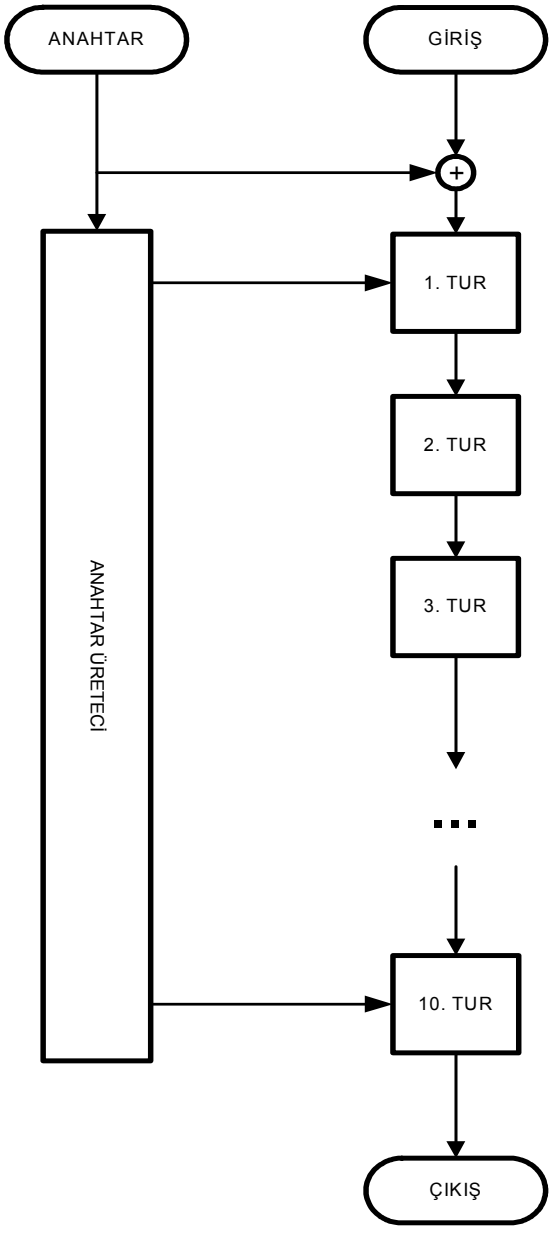
## 3.2 Rijndael Algoritmasının Yapısı

AES [1] algoritmasında giriş, çıkış ve matrisler 128 bitliktir. Matris 4 satır, 4 sütun (4x4), 16 bölmeden oluşur. Bu matrise ‘durum’ denmektedir. Durumun her bölmesine bir baytlık veri düşer. Her satır da 32 bitlik bir kelimeyi meydana getirir. Bu gösterimde $N_b = 4$ olmaktadır. Anahtar uzunluğu 128, 192 veya 256 olabilmektedir. Bu bit sayılarındaki farklılık AES tur döngülerinin sayısını değiştirmektedir. Aşağıdaki Tablo 3.1’de farklı anahtar uzunlukları için gereken tur sayıları gösterilmiştir.

**Tablo 3.1** Tur Sayısının Anahtar Uzunluğuna Göre Değişimi

| | Kelime Uzunluğu | Tur Sayısı |
|---|---|---|
| AES-128 | 4 | 10 |
| AES-192 | 6 | 12 |
| AES-256 | 8 | 14 |

Bu çalışmada kullanılan anahtar uzunluğu 128 bit olduğundan 10 turluk işlem gerekmektedir. Şekil 3.1'de algoritmanın genel yapısı 10 tur ile birlikte gösterilmiştir.

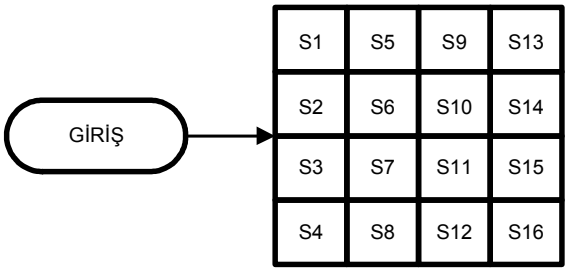



**Şekil 3.1** AES Algoritması Genel Görünüm

Her bir turda 4 farklı alt işlem gerçekleştirilir. Bunlar sırası ile bayt değiştirme, satır kaydırma, sütun karıştırma ve tur anahtarı ile toplamadır. 10 tur sonrasında giren veri şifrelenmiş olarak dışarı çıkmaktadır. İlk tura anahtar ilk haliyle katılmakta diğer turlara yeni üretilen anahtarlar sokulmaktadır [3].

### 3.3 Şifreli Metinin Oluşumu

Girişten gelen metin 128 bitlik parçalara bölünür. Her parça durum matrisine yerleştirilir. Bu yerleştirme Şekil 3.2'de gösterilmiştir. Durum matrisi oluşturulduktan sonra, artık üzerinde tüm işlemler yapılabilir duruma gelmiş demektir. Aynı şekilde önceden alınan 128 bitlik anahtar da bu durum matrisi halinde işlem görür. Giriş metninin yazıldığı durum matrisi ilk olarak anahtar ile toplanır.

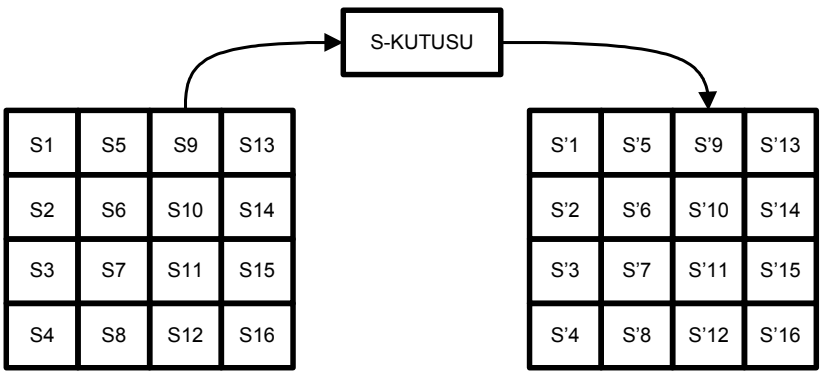



**Şekil 3.2** Girişin Alındığı 4x4'lük durum matrisi

Ardından algoritmanın temeli sayılabilecek bir tur işlemine tabi tutulur. Bu tur işleminin içeriği sıradaki başlıklarda detaylı olarak anlatılmıştır.

#### 3.3.1 Bayt Değiştirme

İlk işlem algoritmanın tek doğrusal olmayan işlemi olan bayt değiştirmedir. Durum matrisinin her elemanı, değerleri önceden hesaplanarak oluşturulmuş S-kutusundaki değerlerle değiştirilir (Şekil 3.3).




**Şekil 3.3** S-kutusundaki Değer ile Bayt Değişimi

S-kutusundaki değerler hesaplanırken önce eldeki değerin çarpmaya göre tersi alınır daha sonra da c={63} olmak üzere (3.1) denkleminden yeni değeri hesaplanır.

$$\overline{b_i} = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i \tag{3.1}$$

Bu işlem matris formunda yazılacak olursa ortaya (3.2) denklemi çıkar [3].

$$\begin{bmatrix} \overline{b_7} \\ \overline{b_6} \\ \overline{b_5} \\ \overline{b_4} \\ \overline{b_3} \\ \overline{b_2} \\ \overline{b_1} \\ \overline{b_0} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \tag{3.2}$$

Bu işlemlerin sonunda oluşabilecek tüm değerler S-kutusunda sonradan kullanılmak üzere toplanabilir. Tablo 3.2'de S-kutusundaki tüm değerler yazılmıştır.

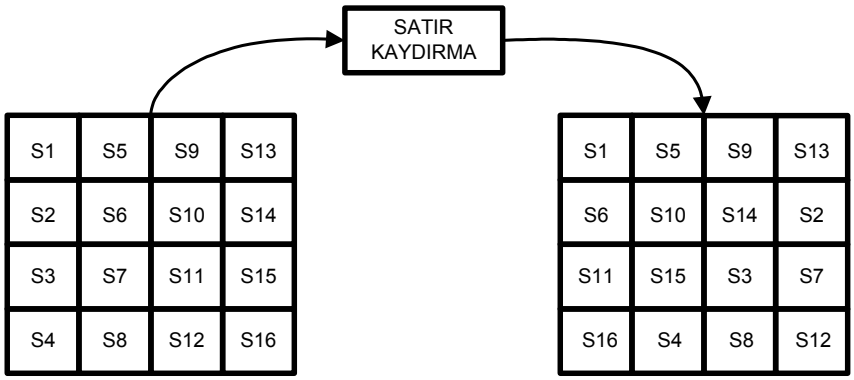
**Tablo 3.2** S-kutusu Değerleri [3]

| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 63 | 7c | 77 | 7b | f2 | 6b | 6f | C5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| **1** | ca | 82 | c9 | 7d | fa | 59 | 47 | F0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| **2** | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| **3** | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| **4** | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| **5** | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| **6** | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| **7** | 51 | a3 | 40 | 8f | 92 | 9d | 38 | F5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| **8** | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| **9** | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| **A** | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| **B** | e7 | c8 | 37 | 6d | 8d | d5 | 4e | A9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **C** | ba | 78 | 25 | 2e | 1c | a6 | b4 | C6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| **D** | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| **E** | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| **F** | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

### 3.3.2 Satır Kaydırma

Satır kaydırma işleminde satırlar sırayla çevrimsel şekilde kaydırılırlar. Yani ilk satır değiştirilmez, ikinci satır sola 1 ötelenir, üçüncü satır sola veya sağa 2 ötelenir ve son satır sola 3 ötelenir veya sağa 1 ötelenir. Taşan bölmeler kaydırmanın başına eklenir. Şekil 3.4'de bu işlemin nasıl yaptığı görsel olarak belirtilmiş, bir satır kaydırma örneği verilmiştir [2].




**Şekil 3.4** Satır Kaydırma Örneği

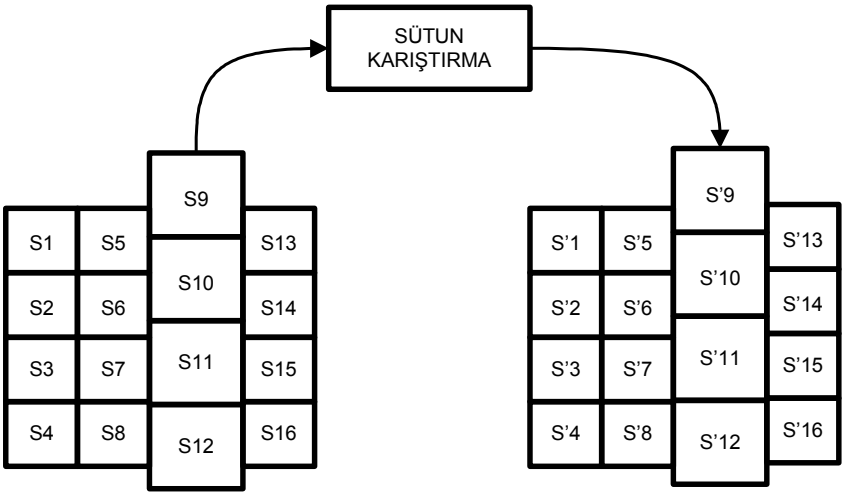
Bu işlem ile 128 bit verinin baytları yer değiştirmiş olup çözülmesi zorlaştırılmıştır [3].

### 3.3.3 Sütun Karıştırma

Bu işlemde eski sütunun elemanları kullanılarak yeni sütun elde edilmektedir. Bu yapılırken yeni sütunun elemanları eski sütunun her elemanı hesaba katılarak tek tek hesaplanır. Yapılan hesap çarpma ve toplama işlemlerinden oluşur [2]. Çarpma işleminde belirli bir sabit sayı kullanılır. Şekil 3.6 üzerinden anlatmak gerekirse, sabit sayı değeri a(x) olup yeni sütun, denklem (3.3)'deki gibi eski sütunun bu sayı ile çarpımından meydana gelir.

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

$s'(x) = a(x) * s(x) \qquad (\text{modulo } x^4 + 1)$ (3.3)




**Şekil 3.5** Sütun Karıştırma İşlemi

Denklem (3.3)'ü açarak matris formunda yazacak olursak ortaya denklem (3.4) çıkar.

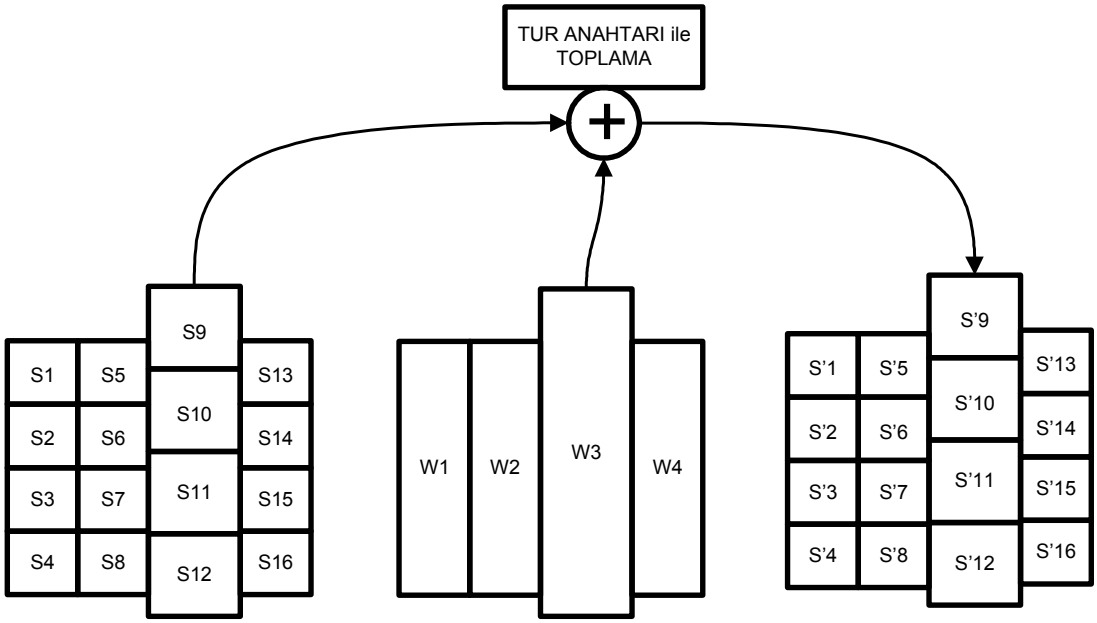
$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{3.4}$$

(3.4)'deki matris açılırsa da (3.5)'deki denklem ortaya çıkar [3].

$$\begin{aligned} s_0' &= (\{02\} * s_0) \oplus (\{03\} * s_1) \oplus s_2 \oplus s_3 \\ s_1' &= s_0 \oplus (\{02\} * s_1) \oplus (\{03\} * s_2) \oplus s_3 \\ s_2' &= s_0 \oplus s_1 \oplus (\{02\} * s_2) \oplus (\{03\} * s_3) \\ s_3' &= (\{03\} * s_0) \oplus s_1 \oplus s_2 \oplus (\{02\} * s_3) \end{aligned} \tag{3.5}$$

### 3.3.4 Tur Anahtarıyla Toplama

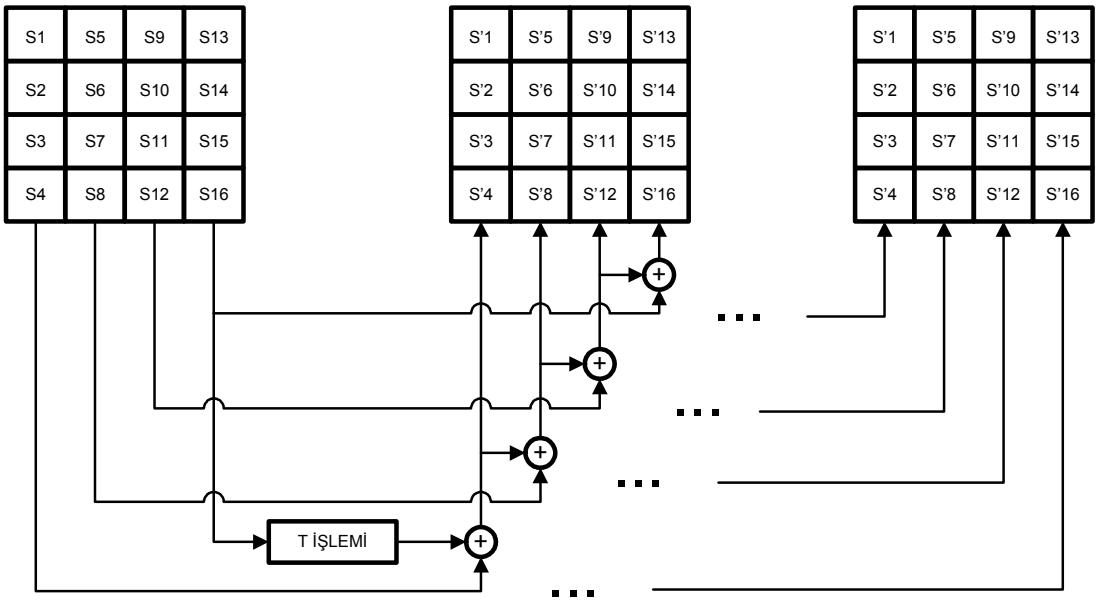
Her turda daha önce saydığımız işlemlerle birlikte bir de tur anahtarı oluşturma işlemi yapılmaktadır ve her turda sonuçta oluşan durum ile o tur için hazırlanmış olan yeni anahtar toplama işlemine tabi tutulur. Bu işlem sonlu alanlarda yapılan toplama işlemidir ve bit mertebesinde özel veya işlemine karşılık düşer. 128 bitlik durum matrisi ile 128 bitlik ara anahtar değeri bit bit özel veya elemanı ile toplanırlar. Şekil 3.6'da nasıl yapıldığı daha detaylı gösterilmektedir [3].




**Şekil 3.6** Tur Anahtarı ile Toplama İşlemi

## 3.4 Anahtar Üretme

AES algoritması anahtarı alır ve bir dizi işlemden geçirerek işlem sayısı kadar anahtar oluşturur. Bu sayı 128 bitlik uzunluk için 10'dur. 10 farklı anahtar oluşturulur ve oluşan son anahtar şifreyi çözmede kullanılan ilk anahtar haline gelir. Çözerken de aynı işlemler benzer olarak tersten yürütülerek kullanılır. Şekil 3.7'de de görüldüğü gibi anahtar üretmede her yeni oluşturulan yeni anahtar kendinden önceki anahtarlar kullanılarak elde edilir.

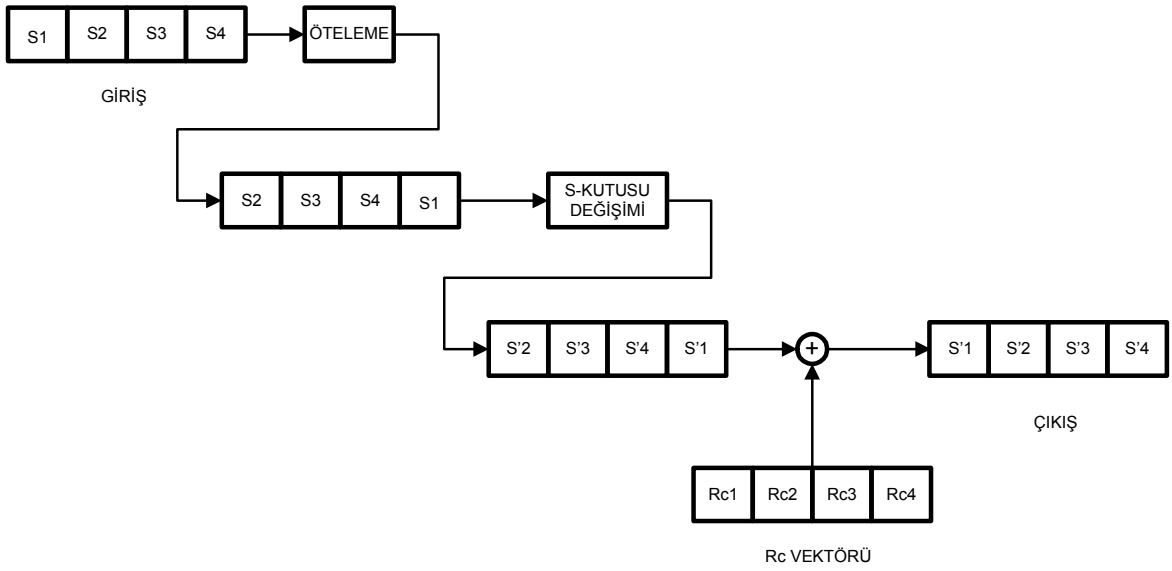



**Şekil 3.7** Anahtar Üretimi

Şekilde de görüldüğü üzere, yeni anahtarın oluşmasındaki temel işlem bir önceki ile dört önceki satırın toplanması. Ancak bir istisna nokta var ki o da her 4'ün katı olan satırda toplamadan önce bir dizi işlemden (T işlemi) daha geçirilir. Bu işlemler öteleme, S kutusundan geçirme ve Tablo 3.3'deki Rc(x) vektörü ile toplama işlemidir [3] (Şekil 3.8).

**Tablo 3.3** Rc(x) Vektörleri

| **Tur Sayısı** | **Rc Değeri** |
|---|---|
| **1** | 01 00 00 00 |
| **2** | 02 00 00 00 |
| **3** | 04 00 00 00 |
| **4** | 08 00 00 00 |
| **5** | 10 00 00 00 |
| **6** | 20 00 00 00 |
| **7** | 40 00 00 00 |
| **8** | 80 00 00 00 |
| **9** | 1b 00 00 00 |
| **10** | 36 00 00 00 |




**Şekil 3.8** T-İşlemi

### 3.4.1 Anahtar Üretimi Örneği

Aşağıda bir anahtar üretim örneği verilmiştir. Tablo 3.4'te ayrıntılı olarak her adımdan sonra oluşan yeni değerler okunabilir.

Anahtar: `2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c`

**Tablo 3.4** Anahtar Üretimine Örnek [3]

| i | 1 Önceki Değer | Ötelemeden Sonra | S Kutusundan Sonra | Rc Değeri | Rc ile Toplamadan Sonra | 4 Önceki Değer | Oluşan Değer |
|---|---|---|---|---|---|---|---|
| **4** | 09cf4f3c | cf4f3c09 | 8a84eb01 | 1000000 | 8b84eb01 | 2b7e1516 | a0fafe17 |
| **5** | a0fafe17 | | | | | 28aed2a6 | 88542cb1 |
| **6** | 88542cb1 | | | | | abf71588 | 23a33939 |
| **7** | 23a33939 | | | | | 09cf4f3c | 2a6c7605 |
| **8** | 2a6c7605 | 6c76052a | 50386be5 | 2000000 | 52386be5 | a0fafe17 | f2c295f2 |
| **9** | f2c295f2 | | | | | 88542cb1 | 7a96b943 |
| **10** | 7a96b943 | | | | | 23a33939 | 5935807a |
| **11** | 5935807a | | | | | 2a6c7605 | 7359f67f |
| **12** | 7359f67f | 59f67f73 | cb42d28f | 4000000 | cf42d28f | f2c295f2 | 3d80477d |
| **13** | 3d80477d | | | | | 7a96b943 | 4716fe3e |
| **14** | 4716fe3e | | | | | 5935807a | 1e237e44 |
| **15** | 1e237e44 | | | | | 7359f67f | 6d7a883b |
| **16** | 6d7a883b | 7a883b6d | dac4e23c | 8000000 | d2c4e23c | 3d80477d | ef44a541 |
| **17** | ef44a541 | | | | | 4716fe3e | a8525b7f |
| **18** | a8525b7f | | | | | 1e237e44 | b671253b |
| **19** | b671253b | | | | | 6d7a883b | db0bad00 |
| **20** | db0bad00 | 0bad00db | 2b9563b9 | 10000000 | 3b9563b9 | ef44a541 | d4d1c6f8 |

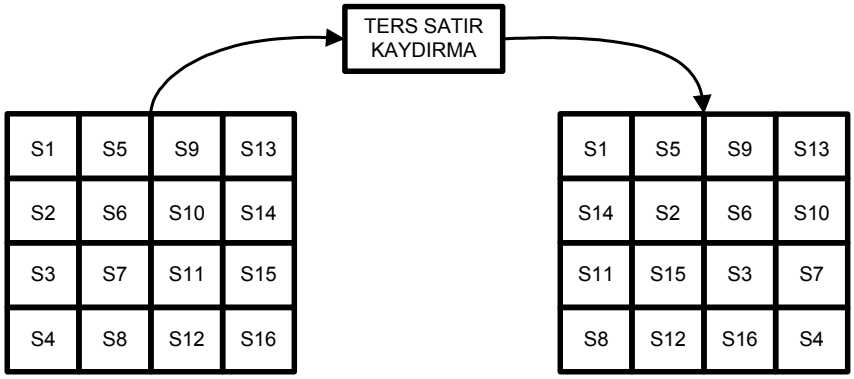
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **21** | d4d1c6f8 | | | | | a8525b7f | 7c839d87 |
| **22** | 7c839d87 | | | | | b671253b | caf2b8bc |
| **23** | caf2b8bc | | | | | db0bad00 | 11f915bc |
| **24** | 11f915bc | f915bc11 | 99596582 | 20000000 | b9596582 | d4d1c6f8 | 6d88a37a |
| **25** | 6d88a37a | | | | | 7c839d87 | 110b3efd |
| **26** | 110b3efd | | | | | caf2b8bc | dbf98641 |
| **27** | dbf98641 | | | | | 11f915bc | ca0093fd |
| **28** | ca0093fd | 0093fdca | 63dc5474 | 40000000 | 23dc5474 | 6d88a37a | 4e54f70e |
| **29** | 4e54f70e | | | | | 110b3efd | 5f5fc9f3 |
| **30** | 5f5fc9f3 | | | | | dbf98641 | 84a64fb2 |
| **31** | 84a64fb2 | | | | | ca0093fd | 4ea6dc4f |
| **32** | 4ea6dc4f | a6dc4f4e | 2486842f | 80000000 | a486842f | 4e54f70e | ead27321 |
| **33** | ead27321 | | | | | 5f5fc9f3 | b58dbad2 |
| **34** | b58dbad2 | | | | | 84a64fb2 | 312bf560 |
| **35** | 312bf560 | | | | | 4ea6dc4f | 7f8d292f |
| **36** | 7f8d292f | 8d292f7f | 5da515d2 | 1b000000 | 46a515d2 | ead27321 | ac7766f3 |
| **37** | ac7766f3 | | | | | b58dbad2 | 19fadc21 |
| **38** | 19fadc21 | | | | | 312bf560 | 28d12941 |
| **39** | 28d12941 | | | | | 7f8d292f | 575c006e |
| **40** | 575c006e | 5c006e57 | 4a639f5b | 36000000 | 7c639f5b | ac7766f3 | d014f9a8 |
| **41** | d014f9a8 | | | | | 19fadc21 | c9ee2589 |
| **42** | c9ee2589 | | | | | 28d12941 | e13f0cc8 |
| **43** | e13f0cc8 | | | | | 575c006e | b6630ca6 |

### 3.5 Şifreli Metini Çözme

AES algoritmasında oluşturulan şifreli metin kolaylıkla, ters işlemlerle tekrar çözülerek giriş metni elde edilebilmektedir. Bu çözme işi için yapılan işlemler ters satır kaydırma, ters S kutusundan geçirme, ters sütun karıştırma ve tur anahtarı ile toplama işlemidir.

#### 3.5.1 Ters Satır Kaydırma İşlemi

Şifreleme yaparken kullanılan satır kaydırma işleminin sola değil de sağa kaydırarak yapılmasıdır. Şekil 3.9'da detaylı olarak gösterimi bulunmaktadır.




**Şekil 3.9** Ters Satır Kaydırma İşlemi

### 3.5.2 Ters S-kutusunda Geçirme İşlemi

Bu kez de değişim uygulanan S-kutusunda bir değişiklik yapılmaktadır. Eski değerlere geri dönebilmek için S-kutusunu oluşturan değerler farklı bir metodla hesaplanır. Sonuç olarak oluşan kutu; (Tablo 3.5) normal S-kutusunda elde ettiğimiz değeri tekrar girişine uyguladığımızda bize ilk verdiğimiz değeri geri verecek şekilde düzenlenmiş halinden ibarettir.

**Tablo 3.5** Ters S-kutusu Değerleri [3]

| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **a** | **b** | **c** | **d** | **e** | **f** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 52 | 9 | 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** | 8 | 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 | 0 | 8c | bc | d3 | 0a | f7 | e4 | 58 | 5 | b8 | b3 | 45 | 6 |
| **7** | d0 | 2c | 1e | 8f | ca | 3f | 0f | 2 | c1 | af | bd | 3 | 1 | 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 | 7 | 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 | 4 | 7e | ba | 77 | d6 | 26 | e1 | 69 | 14 | 63 | 55 | 21 | 0c | 7d |

### 3.5.3 Ters Sütun Karıştırma İşlemi

Bu işlemde sütunlar şifreleme sırasındaki sütun karıştırma işleminden farklı bir sabit polinomla çarpılır. Bu polinom ve işlemler denklem (3.6), (3.7)'de gösterilmişlerdir [3].

$$a(x) = 0\text{B} * x^3 + 0\text{D} * x^2 + 09 * x + 0\text{E} \tag{3.6}$$

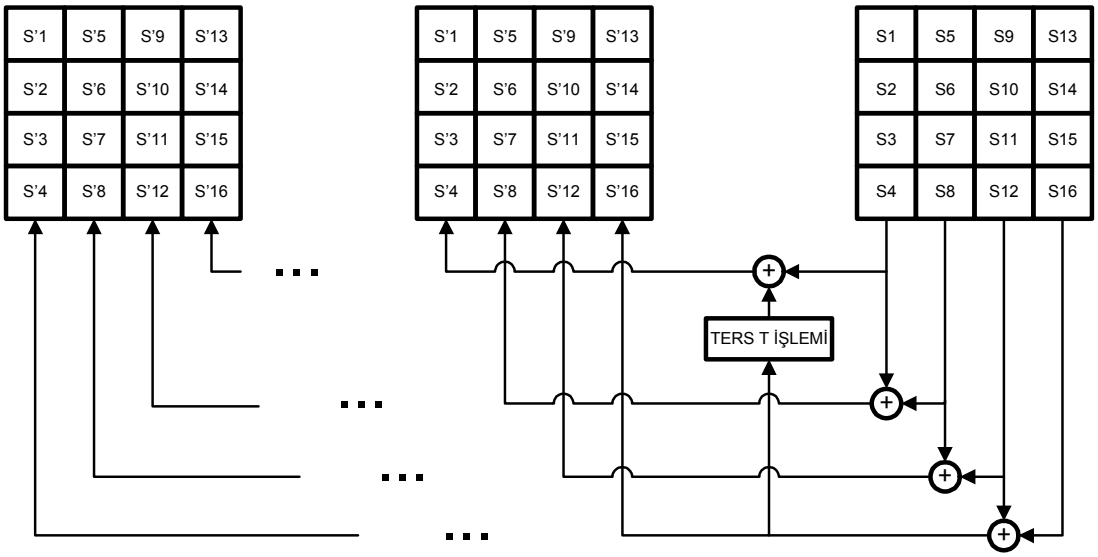
$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0\text{E} & 0\text{B} & 0\text{D} & 09 \\ 09 & 0\text{E} & 0\text{B} & 0\text{D} \\ 0\text{D} & 09 & 0\text{E} & 0\text{B} \\ 0\text{B} & 0\text{D} & 09 & 0\text{E} \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{3.7}$$

### 3.5.4 Çözme İşleminde Tur Anahtarı İle Toplama

Ters tur anahtarı ile toplama işlemi demeyişimin sebebi tur anahtarı ile toplamanın şifreleme ve şifre çözme işlemlerinde aynı şekilde uygulanıyor olmasıdır. Bit bit özel veya işleminden ibaret olması sebebi ile tekrar özel veya işlemine tabi tutulması eski haline dönmesine sebep olmaktadır. Sonuç olarak şifreleme işlemindeki tur anahtarı ile toplama işleminin aynısı yürütülür.

### 3.5.5 Ters Anahtar Üretimi

Şifreleme işleminde oluşan son anahtar, çözme işleminin ilk anahtarı olarak kullanılır. Anahtar üretiminin tersi işlemler ile ters anahtar üretimi gerçekleştirilir. Şekil 3.10'da gördüğümüz gibi önce elimizdeki anahtarın sütunlarını kayıt altına alırız.




**Şekil 3.10** Ters Anahtar Üretimi

Sütunlara baştan itibaren $w_0$,$w_1$,$w_2$,$w_3$, dersek; önce $w_2$ ve $w_3$'ü özel veya ile toplar yeni $w'_0$ değerini elde ederiz. Aynı şekilde $w_1$ ve $w_2$ de $w'_1$ değerini verir. Bu şekilde geriye doğru giderek yeni durum oluşturulur. Son olarak yeni oluşan $w_0$ sütununu 1 sola öteledikten sonra her sütunu S-kutusundan geçiririz ve ters Rc katsayısı ile çarparız. Böylece yeni anahtar oluşmuş olur. Tablo 3.6'da ters Rc değerleri verilmiştir [3].

**Tablo 3.6** Ters Rc(x) Vektörleri

| **Tur Sayısı** | **Ters Rc Değeri** |
|---|---|
| **1** | 36 00 00 00 |
| **2** | 1b 00 00 00 |
| **3** | 80 00 00 00 |
| **4** | 40 00 00 00 |
| **5** | 20 00 00 00 |
| **6** | 10 00 00 00 |
| **7** | 08 00 00 00 |
| **8** | 04 00 00 00 |
| **9** | 02 00 00 00 |
| **10** | 01 00 00 00 |

### 3.6 Şifreli Metin Oluşturma ve Çözme Örneği

128 bit giriş metni, 128 bit anahtar ile şifreleme ve ardından bu şifreli metni çözme işlemlerini adım adım giderek ne gibi değişikliklerin meydana geldiği Tablo 3.7 ve Tablo 3.8'de ayrıntılı olarak gösterilmiştir.

**Tablo 3.7** Şifreleme Adımları örneği [3]

| **Giriş Metni:** | 00112233445566778899aabbccddeeff |
|---|---|
| **Anahtar:** | 000102030405060708090a0b0c0d0e0f |
| **round[ 0].input** | 00112233445566778899aabbccddeeff |
| **round[ 0].k_sch** | 000102030405060708090a0b0c0d0e0f |
| **round[ 1].start** | 00102030405060708090a0b0c0d0e0f0 |
| **round[ 1].s_box** | 63cab7040953d051cd60e0e7ba70e18c |
| **round[ 1].s_row** | 6353e08c0960e104cd70b751bacad0e7 |
| **round[ 1].m_col** | 5f72641557f5bc92f7be3b291db9f91a |

| | |
|---|---|
| **round[ 1].k_sch** | d6aa74fdd2af72fadaa678f1d6ab76fe |
| **round[ 2].start** | 89d810e8855ace682d1843d8cb128fe4 |
| **round[ 2].s_box** | a761ca9b97be8b45d8ad1a611fc97369 |
| **round[ 2].s_row** | a7be1a6997ad739bd8c9ca451f618b61 |
| **round[ 2].m_col** | ff87968431d86a51645151fa773ad009 |
| **round[ 2].k_sch** | b692cf0b643dbdf1be9bc5006830b3fe |
| **round[ 3].start** | 4915598f55e5d7a0daca94fa1f0a63f7 |
| **round[ 3].s_box** | 3b59cb73fcd90ee05774222dc067fb68 |
| **round[ 3].s_row** | 3bd92268fc74fb735767cbe0c0590e2d |
| **round[ 3].m_col** | 4c9c1e66f771f0762c3f868e534df256 |
| **round[ 3].k_sch** | b6ff744ed2c2c9bf6c590cbf0469bf41 |
| **round[ 4].start** | fa636a2825b339c940668a3157244d17 |
| **round[ 4].s_box** | 2dfb02343f6d12dd09337ec75b36e3f0 |
| **round[ 4].s_row** | 2d6d7ef03f33e334093602dd5bfb12c7 |
| **round[ 4].m_col** | 6385b79ffc538df997be478e7547d691 |
| **round[ 4].k_sch** | 47f7f7bc95353e03f96c32bcfd058dfd |
| **round[ 5].start** | 247240236966b3fa6ed2753288425b6c |
| **round[ 5].s_box** | 36400926f9336d2d9fb59d23c42c3950 |
| **round[ 5].s_row** | 36339d50f9b539269f2c092dc4406d23 |
| **round[ 5].m_col** | f4bcd45432e554d075f1d6c51dd03b3c |
| **round[ 5].k_sch** | 3caaa3e8a99f9deb50f3af57adf622aa |
| **round[ 6].start** | c81677bc9b7ac93b25027992b0261996 |
| **round[ 6].s_box** | e847f56514dadde23f77b64fe7f7d490 |
| **round[ 6].s_row** | e8dab6901477d4653ff7f5e2e747dd4f |
| **round[ 6].m_col** | 9816ee7400f87f556b2c049c8e5ad036 |
| **round[ 6].k_sch** | 5e390f7df7a69296a7553dc10aa31f6b |
| **round[ 7].start** | c62fe109f75eedc3cc79395d84f9cf5d |
| **round[ 7].s_box** | b415f8016858552e4bb6124c5f998a4c |
| **round[ 7].s_row** | b458124c68b68a014b99f82e5f15554c |
| **round[ 7].m_col** | c57e1c159a9bd286f05f4be098c63439 |
| **round[ 7].k_sch** | 14f9701ae35fe28c440adf4d4ea9c026 |
| **round[ 8].start** | d1876c0f79c4300ab45594add66ff41f |
| **round[ 8].s_box** | 3e175076b61c04678dfc2295f6a8bfc0 |
| **round[ 8].s_row** | 3e1c22c0b6fcbf768da85067f6170495 |
| **round[ 8].m_col** | baa03de7a1f9b56ed5512cba5f414d23 |
| **round[ 8].k_sch** | 47438735a41c65b9e016baf4aebf7ad2 |
| **round[ 9].start** | fde3bad205e5d0d73547964ef1fe37f1 |
| **round[ 9].s_box** | 5411f4b56bd9700e96a0902fa1bb9aa1 |
| **round[ 9].s_row** | 54d990a16ba09ab596bbf40ea111702f |
| **round[ 9].m_col** | e9f74eec023020f61bf2ccf2353c21c7 |
| **round[ 9].k_sch** | 549932d1f08557681093ed9cbe2c974e |
| **round[10].start** | bd6e7c3df2b5779e0b61216e8b10b689 |
| **round[10].s_box** | 7a9f102789d5f50b2beffd9f3dca4ea7 |
| **round[10].s_row** | 7ad5fda789ef4e272bca100b3d9ff59f |
| **round[10].k_sch** | 13111d7fe3944a17f307a78b4d2b30c5 |
| **round[10].output** | 69c4e0d86a7b0430d8cdb78070b4c55a |

**Tablo 3.8** Şifre Çözme Adımları örneği [3]

| | |
|---|---|
| **round[ 0].iinput** | 69c4e0d86a7b0430d8cdb78070b4c55a |
| **round[ 0].ik_sch** | 13111d7fe3944a17f307a78b4d2b30c5 |
| **round[ 1].istart** | 7ad5fda789ef4e272bca100b3d9ff59f |
| **round[ 1].is_row** | 7a9f102789d5f50b2beffd9f3dca4ea7 |
| **round[ 1].is_box** | bd6e7c3df2b5779e0b61216e8b10b689 |
| **round[ 1].ik_sch** | 549932d1f08557681093ed9cbe2c974e |
| **round[ 1].ik_add** | e9f74eec023020f61bf2ccf2353c21c7 |
| **round[ 2].istart** | 54d990a16ba09ab596bbf40ea111702f |
| **round[ 2].is_row** | 5411f4b56bd9700e96a0902fa1bb9aa1 |

| | |
|---|---|
| **round[ 2].is_box** | fde3bad205e5d0d73547964ef1fe37f1 |
| **round[ 2].ik_sch** | 47438735a41c65b9e016baf4aebf7ad2 |
| **round[ 2].ik_add** | baa03de7a1f9b56ed5512cba5f414d23 |
| **round[ 3].istart** | 3e1c22c0b6fcbf768da85067f6170495 |
| **round[ 3].is_row** | 3e175076b61c04678dfc2295f6a8bfc0 |
| **round[ 3].is_box** | d1876c0f79c4300ab45594add66ff41f |
| **round[ 3].ik_sch** | 14f9701ae35fe28c440adf4d4ea9c026 |
| **round[ 3].ik_add** | c57e1c159a9bd286f05f4be098c63439 |
| **round[ 4].istart** | b458124c68b68a014b99f82e5f15554c |
| **round[ 4].is_row** | b415f8016858552e4bb6124c5f998a4c |
| **round[ 4].is_box** | c62fe109f75eedc3cc79395d84f9cf5d |
| **round[ 4].ik_sch** | 5e390f7df7a69296a7553dc10aa31f6b |
| **round[ 4].ik_add** | 9816ee7400f87f556b2c049c8e5ad036 |
| **round[ 5].istart** | e8dab6901477d4653ff7f5e2e747dd4f |
| **round[ 5].is_row** | e847f56514dadde23f77b64fe7f7d490 |
| **round[ 5].is_box** | c81677bc9b7ac93b25027992b0261996 |
| **round[ 5].ik_sch** | 3caaa3e8a99f9deb50f3af57adf622aa |
| **round[ 5].ik_add** | f4bcd45432e554d075f1d6c51dd03b3c |
| **round[ 6].istart** | 36339d50f9b539269f2c092dc4406d23 |
| **round[ 6].is_row** | 36400926f9336d2d9fb59d23c42c3950 |
| **round[ 6].is_box** | 247240236966b3fa6ed2753288425b6c |
| **round[ 6].ik_sch** | 47f7f7bc95353e03f96c32bcfd058dfd |
| **round[ 6].ik_add** | 6385b79ffc538df997be478e7547d691 |
| **round[ 7].istart** | 2d6d7ef03f33e334093602dd5bfb12c7 |
| **round[ 7].is_row** | 2dfb02343f6d12dd09337ec75b36e3f0 |
| **round[ 7].is_box** | fa636a2825b339c940668a3157244d17 |
| **round[ 7].ik_sch** | b6ff744ed2c2c9bf6c590cbf0469bf41 |
| **round[ 7].ik_add** | 4c9c1e66f771f0762c3f868e534df256 |
| **round[ 8].istart** | 3bd92268fc74fb735767cbe0c0590e2d |
| **round[ 8].is_row** | 3b59cb73fcd90ee05774222dc067fb68 |
| **round[ 8].is_box** | 4915598f55e5d7a0daca94fa1f0a63f7 |
| **round[ 8].ik_sch** | b692cf0b643dbdf1be9bc5006830b3fe |
| **round[ 8].ik_add** | ff87968431d86a51645151fa773ad009 |
| **round[ 9].istart** | a7be1a6997ad739bd8c9ca451f618b61 |
| **round[ 9].is_row** | a761ca9b97be8b45d8ad1a611fc97369 |
| **round[ 9].is_box** | 89d810e8855ace682d1843d8cb128fe4 |
| **round[ 9].ik_sch** | d6aa74fdd2af72fadaa678f1d6ab76fe |
| **round[ 9].ik_add** | 5f72641557f5bc92f7be3b291db9f91a |
| **round[10].istart** | 6353e08c0960e104cd70b751bacad0e7 |
| **round[10].is_row** | 63cab7040953d051cd60e0e7ba70e18c |
| **round[10].is_box** | 00102030405060708090a0b0c0d0e0f0 |
| **round[10].ik_sch** | 000102030405060708090a0b0c0d0e0f |
| **round[10].ioutput** | 00112233445566778899aabbccddeeff |

# 4. AES ALGORİTMASININ FPGA ÜZERİNDE GERÇEKLENMESİ

AES algoritmasının FPGA üzerinde gerçeklenmesi için seçilmiş olan donanım programlama dili VHDL'dir. Tüm modüller VHDL kullanılarak tasarlanmıştır. Tasarım ortamı olarak da Xilinx firmasının ISE 9.2i derleyicisi kullanılmıştır.

## 4.1 Giriş

'Sahada Programlanabilir Kapı Dizileri'(FPGA) bugün kolayca ve ucuza satın alınabilmekte ve de deneme amacıyla kullanılıp daha sonra ASIC tasarıma geçebilme olanağı sağlamaktadır. Aslında çok farklı ortamlarda gerçeklenebilen AES algoritması hız, güvenlik ve basitlik sağlaması sebebi ile FPGA üzerinde kullanılması uygun görülmüştür. Bu bölümde tasarımın başından sonuna zaman açısında hiyerarşik bir biçimde nasıl meydana getirildiği anlatılacaktır. Yazılan tüm kodlar ödevle birlikte verilen CD içerisinde bulunmaktadır.

## 4.2 AES Algoritması İçin VHDL Paket Oluşturulması

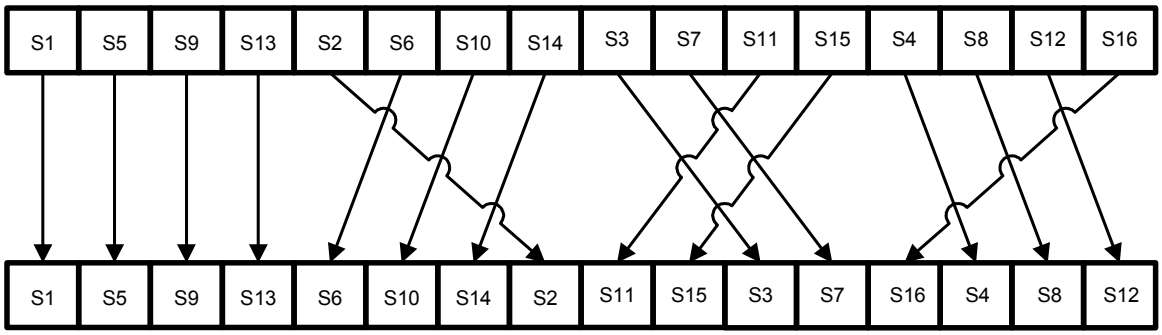
VHDL'de matris tanımlamaları, fonksiyon tanımlamaları veya ROM kullanımı için paket oluşturulması gerekir. Oluşturulan pakette satır ve sütun tanımlamaları yapılmış olup, S-kutusu ROM'a yazılmıştır. Diğer kullanılan alt ve üst modüllerde gerekli olduğu zaman koda bu paketin kullanılması için gereken kod eklenir.

## 4.3 Tur Anahtarı ile Toplama İşleminin Gerçeklenmesi

Algoritmanın görece en kolay yazılabilen ve gerçeklenen bölümü olması sebebi ile ilk olarak tur anahtarı ile toplama işlemi gerçekleştirildi. Bu işlem, 128 bitlik 2 girişin bir biriyle özel veya elemanı kullanılarak toplanıp çıkışa verilmesinden oluşuyor. FPGA'da özel veya gibi işlemleri yapmak için içinde bulunan hazır kapılardan gereken LUT'lar (Look Up Table) oluşturulmaktadır. Bu modül için gereken Slice adedi 74, kullanılan LUT sayısı 128 dir.

### 4.4 Satır Kaydırma İşleminin Gerçeklenmesi

Satır kaydırma işlemi de basitçe ‘for’ döngüsü kullanılarak tasarlanmıştır. Satır kaydırmada temel prensip giren 128 bitlik verinin 128 bit çıkışta bazı baytlarının yerinin değişmiş olması olduğundan bu işlem için FPGA üzerinde sadece çıkış yollarının farklı girişlere bağlanmış olması yeterlidir. Bu yüzden de her hangi bir eleman kullanılmaz ve her hangi bir yer kaplaması da mümkün değildir. Şekil 4.1’de eleman kullanmadan nasıl yapıldığı görülmektedir.




**Şekil 4.1** Satır Kaydırma İşleminin Gerçeklenmesi

### 4.5 Sütun Karıştırma İşleminin Gerçeklenmesi

Bu işlemde 2 modül kullanılmaktadır. Birincisi yani üst modülde hangi çarpma işlemlerinin yapılacağı gösterilmiş olup alt modülde de çarpma işleminin nasıl yapıldığı tanımlanmıştır. Bilindiği gibi sonlu alanlarda yapılan çarpma işleminde sonucun bir polinoma göre modulo işleminden geçirilmesi gerekiyor. Bu yüzden çarpma işlemi matematiksel olarak analiz edilip hangi işlemlerin yapılması gerektiği hesaplanmış ve ayrı ayrı yazılmıştır. Denklem (4.1)’de girişin a(x) vektörü ile nasıl çarpıldığı ve sonucun nasıl bulunduğu açık ifadesi denklem (3.5) kullanılarak gösterilmiştir.

a(x) ve b(x) giriş, c(x) de çıkış olmak üzere,

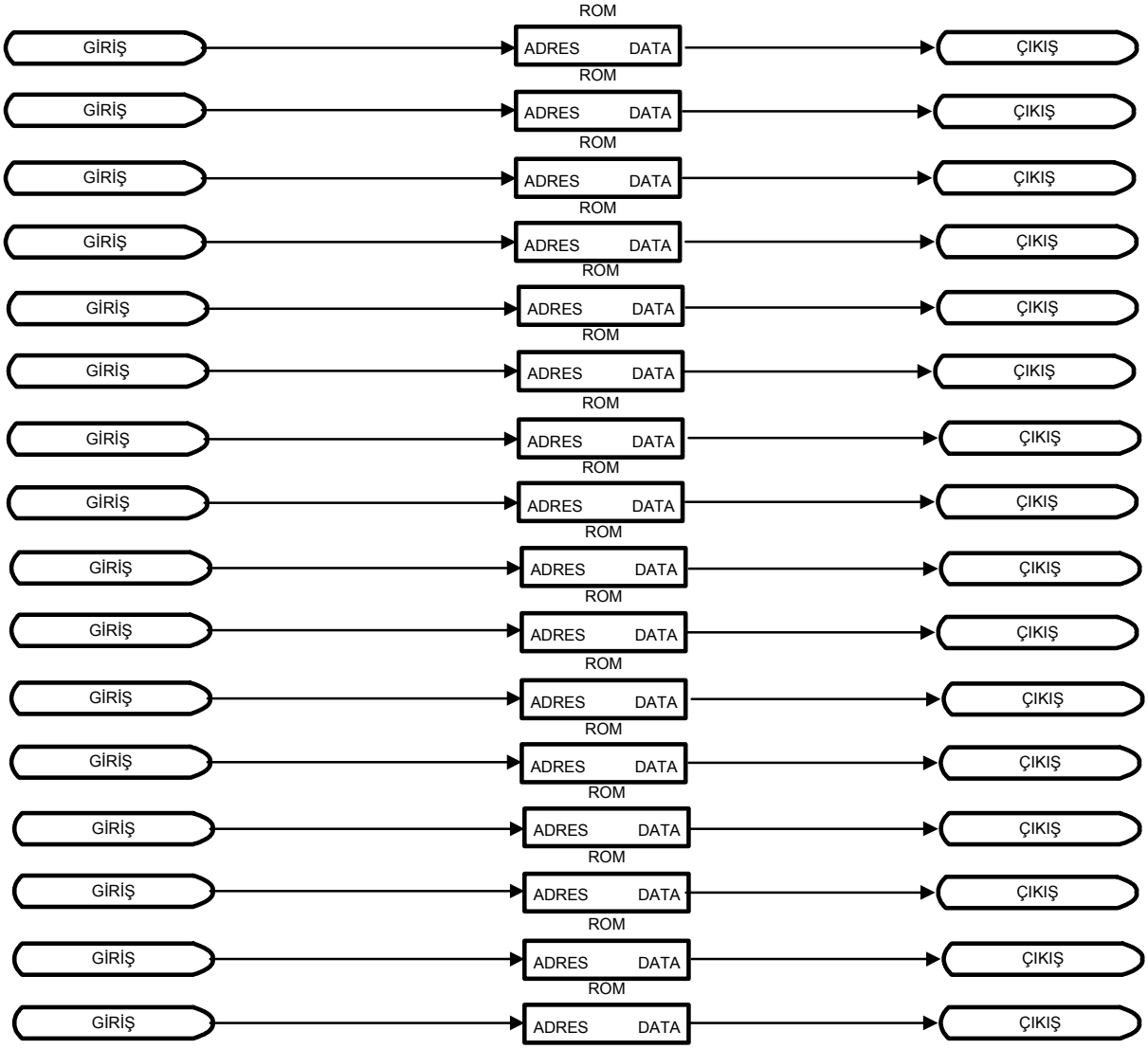
```
c(7)<=  (a(7) + b(0)) ⊕ (a(6) + b(1)) ⊕ (a(5) + b(2)) ⊕ (a(4) + b(3))
        ⊕ (a(3) + b(4)) ⊕ (a(2) + b(5)) ⊕ (a(1) + b(6)) ⊕ (a(0) + b(7))
        ⊕ (a(7) + b(7)) ⊕ (a(7) + b(5)) ⊕ (a(6) + b(6)) ⊕ (a(5) + b(7))
        ⊕ (a(7) + b(4)) ⊕ (a(6) + b(5)) ⊕ (a(5) + b(6)) ⊕ (a(4) + b(7));
```

```
c(6)<= (a(6) + b(0)) ⊕ (a(5) + b(1)) ⊕ (a(4) + b(2)) ⊕ (a(3) + b(3))
       ⊕ (a(2) + b(4)) ⊕ (a(1) + b(5)) ⊕ (a(0) + b(6)) ⊕ (a(7) + b(6))
       ⊕ (a(6) + b(7)) ⊕ (a(7) + b(4)) ⊕ (a(6) + b(5)) ⊕ (a(5) + b(6))
       ⊕ (a(4) + b(7)) ⊕ (a(7) + b(3)) ⊕ (a(6) + b(4)) ⊕ (a(5) + b(5))
       ⊕ (a(4) + b(6)) ⊕ (a(3) + b(7));


c(5)<=  (a(5) + b(0)) ⊕ (a(4) + b(1)) ⊕ (a(3) + b(2)) ⊕ (a(2) + b(3))
      ⊕ (a(1) + b(4)) ⊕ (a(0) + b(5)) ⊕ (a(7) + b(5)) ⊕ (a(6) + b(6))
      ⊕ (a(5) + b(7)) ⊕ (a(7) + b(3)) ⊕ (a(6) + b(4)) ⊕ (a(5) + b(5))
      ⊕ (a(4) + b(6)) ⊕ (a(3) + b(7)) ⊕ (a(7) + b(2)) ⊕ (a(6) + b(3))
      ⊕ (a(5) + b(4)) ⊕ (a(4) + b(5)) ⊕ (a(3) + b(6)) ⊕ (a(2) + b(7));

c(4)<=  (a(4) + b(0)) ⊕ (a(3) + b(1)) ⊕ (a(2) + b(2)) ⊕ (a(1) + b(3))
      ⊕ (a(0) + b(4)) ⊕ (a(7) + b(4)) ⊕ (a(6) + b(5)) ⊕ (a(5) + b(6))
      ⊕ (a(4) + b(7)) ⊕ (a(7) + b(2)) ⊕ (a(6) + b(3)) ⊕ (a(5) + b(4))
      ⊕ (a(4) + b(5)) ⊕ (a(3) + b(6)) ⊕ (a(2) + b(7)) ⊕ (a(7) + b(1))
      ⊕ (a(6) + b(2)) ⊕ (a(5) + b(3)) ⊕ (a(4) + b(4)) ⊕ (a(3) + b(5))
      ⊕ (a(2) + b(6)) ⊕ (a(1) + b(7)) ⊕ (a(7) + b(7)) ;
                                                                    (4.1)
c(3)<=  (a(3) + b(0)) ⊕ (a(2) + b(1)) ⊕ (a(1) + b(2)) ⊕ (a(0) + b(3))
       ⊕ (a(7) + b(1)) ⊕ (a(6) + b(2)) ⊕ (a(5) + b(3)) ⊕ (a(4) + b(4))
       ⊕ (a(3) + b(5)) ⊕ (a(2) + b(6)) ⊕ (a(1) + b(7)) ⊕ (a(7) + b(7))
       ⊕ (a(7) + b(6)) ⊕ (a(6) + b(7)) ⊕ (a(7) + b(5)) ⊕ (a(6) + b(6))
      ⊕ (a(5) + b(7)) ⊕ (a(7) + b(3)) ⊕ (a(6) + b(4)) ⊕ (a(5) + b(5))
      ⊕ (a(4) + b(6)) ⊕ (a(3) + b(7)) ⊕ (a(7) + b(4)) ⊕ (a(6) + b(5))
      ⊕ (a(5) + b(6)) ⊕ (a(4) + b(7));

c(2)<=  (a(7) + b(2)) ⊕ (a(6) + b(3)) ⊕ (a(5) + b(4)) ⊕ (a(4) + b(5))
       ⊕ (a(3) + b(6)) ⊕ (a(2) + b(7)) ⊕ (a(7) + b(6)) ⊕ (a(6) + b(7))
       ⊕ (a(7) + b(3)) ⊕ (a(6) + b(4)) ⊕ (a(5) + b(5)) ⊕ (a(4) + b(6))
       ⊕ (a(3) + b(7)) ⊕ (a(2) + b(0)) ⊕ (a(1) + b(1)) ⊕ (a(0) + b(2));


c(1)<=  (a(1) + b(0)) ⊕ (a(0) + b(1)) ⊕ (a(7) + b(7)) ⊕ (a(7) + b(2))
       ⊕ (a(6) + b(3)) ⊕ (a(5) + b(4)) ⊕ (a(4) + b(5)) ⊕ (a(3) + b(6))
       ⊕ (a(2) + b(7)) ⊕ (a(7) + b(5)) ⊕ (a(6) + b(6)) ⊕ (a(5) + b(7))
       ⊕ (a(7) + b(1)) ⊕ (a(6) + b(2)) ⊕ (a(5) + b(3)) ⊕ (a(4) + b(4))
       ⊕ (a(3) + b(5)) ⊕ (a(2) + b(6)) ⊕ (a(1) + b(7));


c(0)<=  (a(0) + b(0))     ⊕ (a(7) + b(6)) ⊕ (a(6) + b(7)) ⊕ (a(7) + b(5))
       ⊕ (a(6) + b(6)) ⊕ (a(5) + b(7)) ⊕ (a(7) + b(1)) ⊕ (a(6) + b(2))
       ⊕ (a(5) + b(3)) ⊕ (a(4) + b(4)) ⊕ (a(3) + b(5)) ⊕ (a(2) + b(6))
       ⊕ (a(1) + b(7));
```

Üst modülde 16x4 yani 64 tane çarpım işlemi tanımlanması gerekiyor. a(x) vektöründe 4 adet bayt bulunuyor fakat bu baytlardan 2'si aynı ({01}) olmasından dolayı çarpma sayısı 16x3 yani 48'e düşmekte. Sonuç olarak sütun karıştırma işlemi FPGA üzerinde 133 Slice ve 244 LUT harcamaktadır.

### 4.6 Bayt Değiştirme İşleminin Gerçeklenmesi

S-kutusu kullanılarak yapılan bayt değiştirme işleminde bu tasarım için S-kutusu değerleri ROM'a yazılarak kullanılmıştır. Değiştirilmesi gereken her değer için bir S-kutusu gerektiğinden toplamda 16 adet S-kutusu bulunmaktadır. S-kutusundaki değerler AES_pack.vhd paket dosyasına yazılmıştır. 256x8 bitlik 16 adet ROM kullanılmıştır. Devrenin genel görünümü Şekil 4.2'de gösterilmiştir.




**Şekil 4.2** S-kutusu Dönüşümünün Gerçeklenmesi

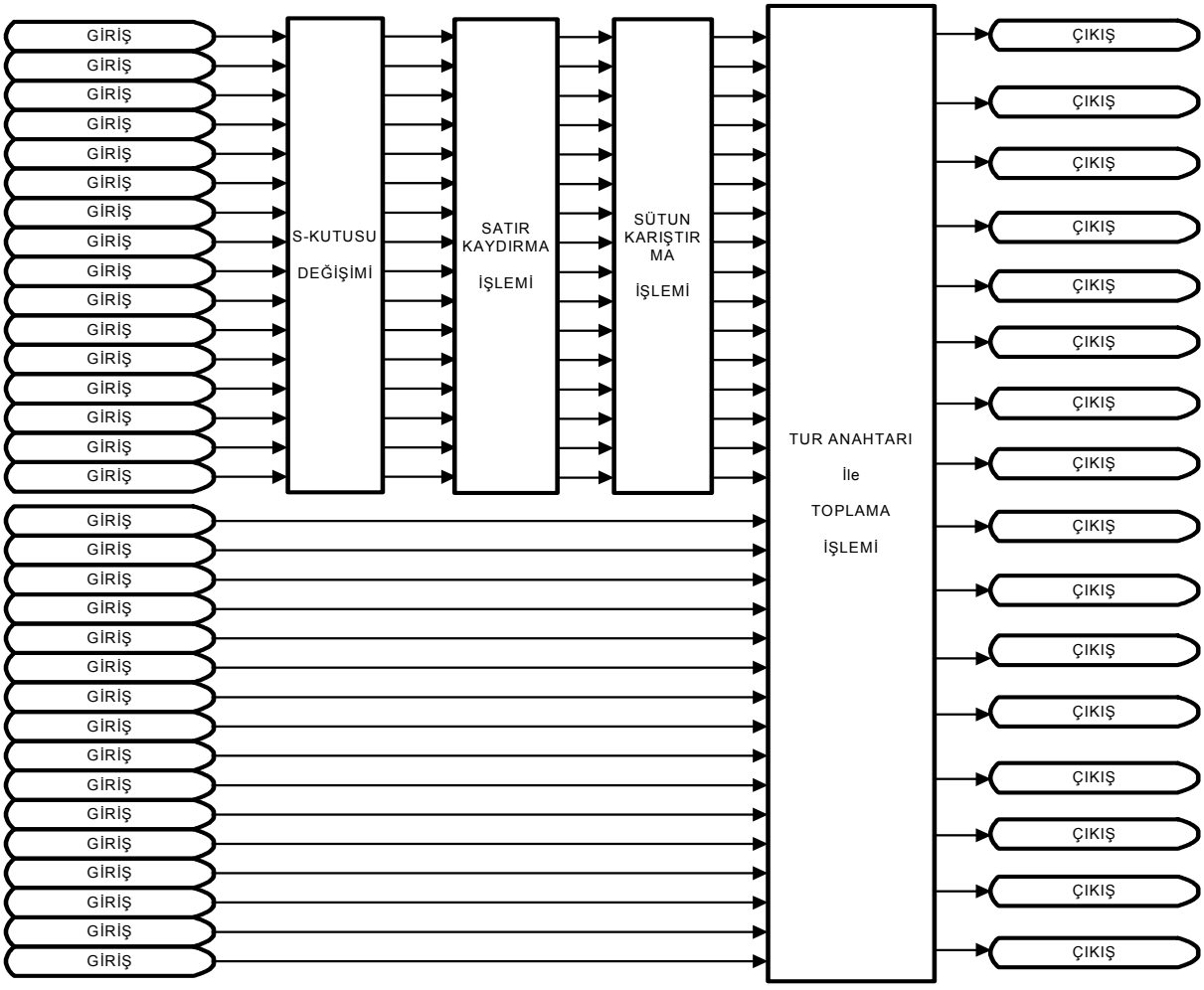
### 4.7 Anahtar Oluşturma İşleminin Gerçeklenmesi

Anahtar üretmek için önce bir çevrimlik blok oluşturulmuştur. Bu blok bir sayaca bağlı olarak çalışmaktadır ve hangi turda ise ona göre katsayıyı seçerek o işlemi

gerçekleştirmektedir. İlk önce T işlemi için 4'ün katı olan sütun öteleme işleminden geçirilir. Daha sonra S-kutusundan geçirme işlemine tabi tutulur. Ancak bu kez S-kutusu 4 baytlıktır. Çünkü sadece bir sütunun değerleri değiştirilmektedir. Ardından S-kutusundan çıkan değer tura uygun Rc vektörü ile çarpılır. Ve geri kalan işlem toplama işlemleridir. Burada S-kutusu yine ROM'a yazılmıştır. Geri kalan işlemler için 440 Slice 853 LUT kullanılmaktadır.

### 4.8 Bir Tur Dönüşümünün Gerçeklenmesi

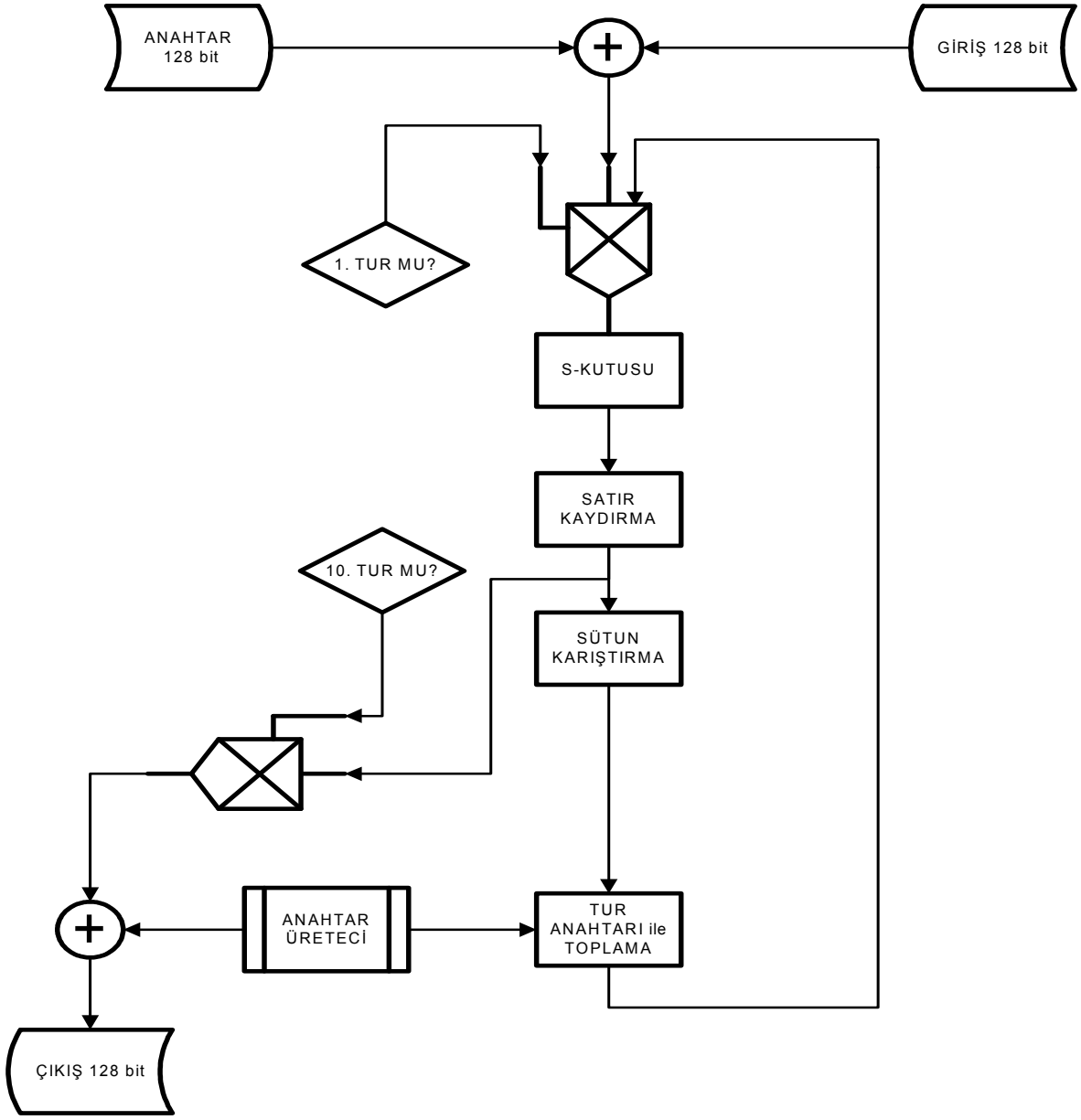
Tur anahtarı ile toplama, satır öteleme, sütun karıştırma, S-kutusundan geçirme işlemleri artarda dizilirler ve kombinezonsal olarak birbirine bağlanırlar (Şekil 4.3). Yani bir saat darbesinde tüm bu işlemler olup bitmişlerdir. 128 bit veri girer ve ilk turun sonunda ilk anahtar ile şifrelenmiş olur. Bu şekilde arka arkaya dizildiklerinde 1210 Slice yer kaplıyorlar.




**Şekil 4.3** Bir Tur Dönüşümünün Genel Görünümü

### 4.9 On Tur Dönüşümünün Gerçeklenmesi

Çıkan ara değer yeni oluşturulan anahtar ile yeni tura sokulur. Bu turda da bir saat darbesinde işlemler tamamlanır ve ikinci şifreli ara durum oluşur. Bunun gibi 9 tur ardışıl olarak gerçekleştikten sonra son turda bir farklılık yapılır. Son turda yani 10. turda sütun karıştırma işlemi yapılmaz. Bu değişiklik tasarımda bir çoğullayıcı kullanılarak çözülmüştür. Çoğullayıcı sütun karıştırma işleminden önceki veriyi 10. tura kadar sütun karıştırma bloğuna, 10. turda ise tur anahtarı ile toplama bloğuna aktarmaktadır. Şekil 4.4'te AES algoritmasının 10 turunun çalışması gösterilmiştir.




**Şekil 4.4** AES Algoritmasının 10 Turu

## 4.10 Şifre Çözme İşleminin Gerçeklenmesi

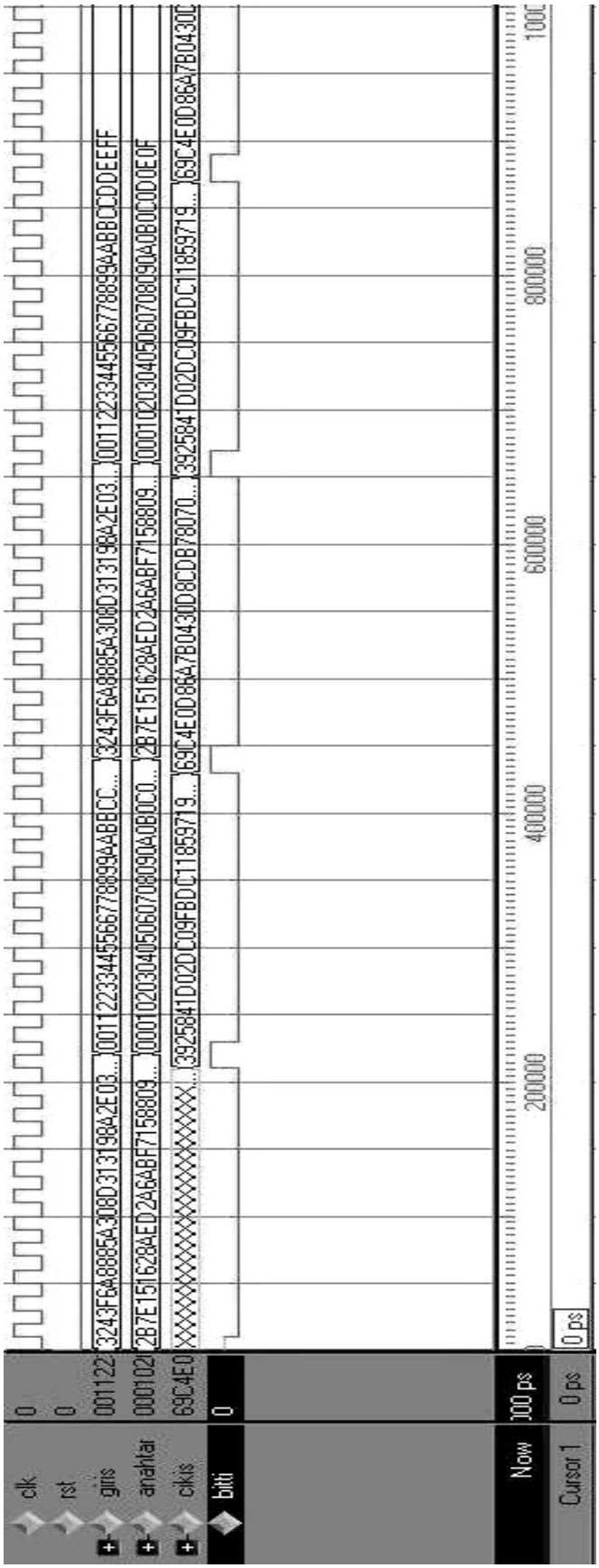
Şifreleme işlemini oluşturan devre tasarlandıktan sonra sıra şifre çözme işlemine gelmiştir. Şifreleme yapan devre elde olduğu için şifre çözme işlemini yapmak daha kolaydır çünkü yapılması gerekli değişiklikler şifreleme tasarımı üzerinde değişiklikler yapılarak oluşturulabilir. Şifre çözmede, satır kaydırma işlemi benzer şekilde yapılır yine yer kaplamaz. S-kutusu için gereken ters değerler pakete yazılır. Ters S-kutusundan geçirme işlemi uygulanır. Sütun karıştırma işleminde şifreleme yaparken kullanılan çarpma bloğu aynen kullanılır yalnız çarpma vektörü farklıdır. Şifre çözmede, sütun karıştırmada kullanılan vektör denklem (4.2)'de verilmiştir.

$$a(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \quad (4.2)$$

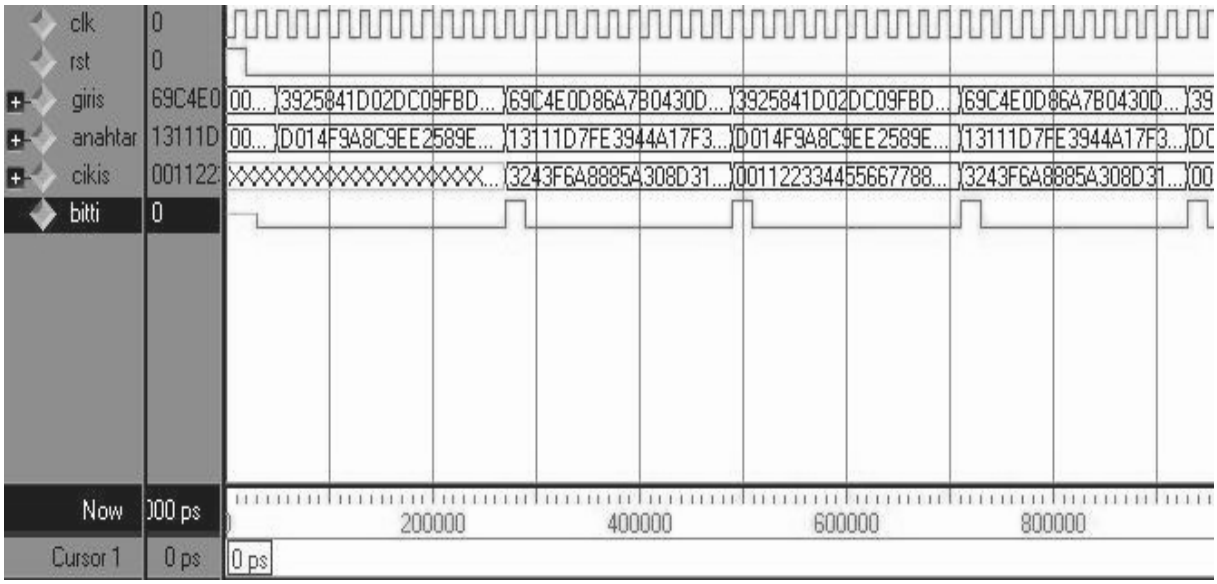
Bu vektörün farklı olması çarpma bloğu sayısını arttırmaktadır. Şifrelemede bu vektör içinde 2 tane {01} baytı olduğu için 16 tane çarpma bloğundan kar etmiştik fakat ters yönde 4 bayt da birbirinden farklı olduğu için mecburen 64 tane çarpma bloğunu da kullanmak zorunda kalırız. Sütun karıştırma işlemi de tamamlandıktan sonra tur anahtarı ile toplama işlemine sıra gelir. Bu işlem zaten aynıdır, özel veya elemanı ile toplama yapılır. Bu arada anahtar üretimi de ters yönde çalışmalıdır. Ters anahtar üretimi bölüm 3.5.5'de anlatıldığı gibi gerçeklenir. Şifre çözme işleminde bu kez ilk tur da sütun karıştırma işlemi bulunmaz ve son turda tek bir tur anahtarı ile toplama işlemi yapılır. Tüm gerekli değişiklikler yapıldıktan sonra çarpma bloklarının da etkisi ile şifre çözme işleminin daha çok yer kapladığı görülür. Devrelerin çalışmalarının simülasyonları bir sonraki başlıkta anlatılmıştır.

## 4.11 Şifreleme ve Çözme İşlemlerinin Doğrulanması

Yazılan tüm kodlar Xilinx ISE 9.2i sentezleme aracı ile sentezlenmiş ve Modelsim XE III 6.2g kullanılarak test edilmiştir. Şekil 4.5'de şifreleme örneğinin simülasyon sonucu verilmiştir. Sonuçların doğruluğuna [1] numaralı kaynaktaki değerlerle karşılaştırılarak karar verilmiştir. Şekil 4.6'da da şifre çözme örneği verilmiştir.




**Şekil 4.5** Şifreleme Simülasyon sonucu

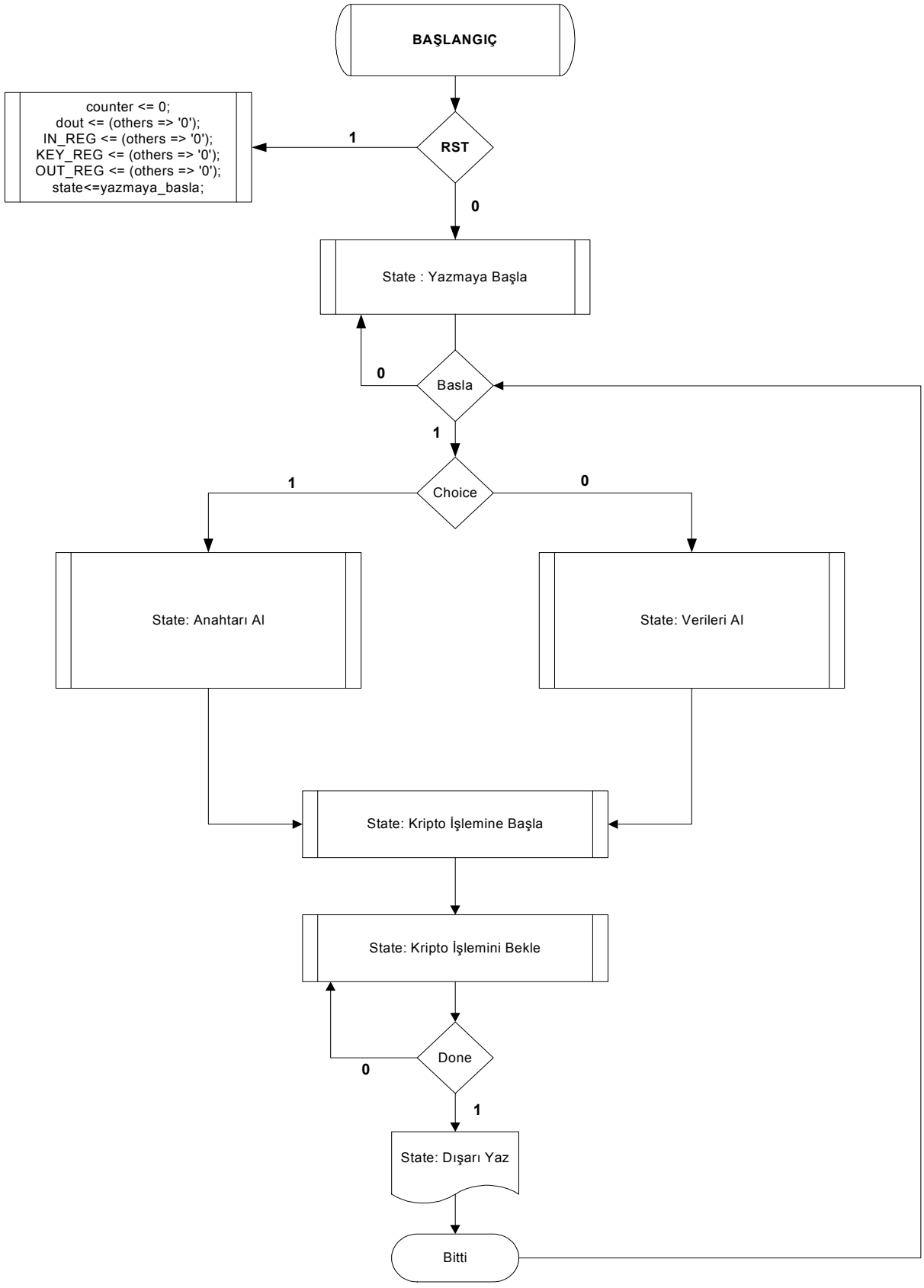



**Şekil 4.6** Şifre Çözme Simülasyon Sonucu

Bu haliyle devrenin şifreleme işlemi 964 Slice yer kaplamaktadır. Saat periyodu 6.328ns en yüksek frekans da 158.033MHz olmaktadır. Şifre çözme işlemi 1273 Slice yer kaplamaktadır. En küçük saat periyodu 11.457 ns olabilmekte ve en yüksek 87.280MHz hızında çalışabilmektedir.

### 4.12 Verilerin Okunması ve Yazılması İçin Gerekli Modüllerin Gerçeklenmesi

Şifreleme ve çözme işlemlerinde giriş çıkışlar hep 128 bit olmaktaydı. Bu şekilde kullanılması durumunda FPGA'nın pinleri yetersiz kalmaktadır. Bu sebeple devreye verilerin daha küçük parçalarda alınmasına karar verilmiştir ve bir katlayıcı devresi yazılmıştır. Örnek olarak 8 bitlik parçalar ile içeri alınan verinin döner yazmaçlarla 128 bitlik yazmaca kayıt edilmesi sağlanmıştır. Ayrıca içeri yazma ve dışarı yazma işlemleri arasında şifreleme veya şifre çözme işlemleri yapılacağı için bunların sıra ile yapılması gerekmektedir. Bu yüzden sıra ile işlerin yürümesini sağlayan bir sonlu durum makinesi yapılmıştır. Bu devrede veriler önce 8'er bit içeri alınmakta şifreleme işlemine gönderilmekte şifreleme işlemi bittiğinde tekrar 8'er bit dışarı yazmaktadır. Devrenin çalışma şekli Şekil 4.7'de gösterilmiştir. Şifre çözme işleminde de aynı şekilde şifrelenmiş olarak gelen veriler 8 bit parçalar halinde geleceği için onları o şekilde alarak 128 bit haline getirmesi ve şifreyi çözerek dışarı yine 8'er bit parçalar halinde vermesi için bir katlayıcı daha yazılmıştır.




**Şekil 4.7** AES İçin Katlayıcı Üst Modülü

# 5. SİSTEMİN GENEL AÇIKLAMASI

## 5.1 Giriş

İletişim araçlarımızdan, ev telefonunun bir benzeri olarak düşünülen sistem normal telefondan farklı olarak şu avantajları sunmaktadır; güvenli iletişim, düşük maliyet ile yüksek güvenlik basit donanım, hızlı şifreleme yapabilme olanağı. Sistemde güvenliği sağlamak için Amerikan Ulusal Standartlar Ve Teknoloji Enstitüsü (National Institute of Standards and Technology (NIST)) tarafından yayınlanan bir Federal Bilgi İşleme Standardı (Federal Information Processing Standards) olan AES yani Gelişmiş Kodlama Standardı (Advanced Encryption Standard) kullanılmıştır. Günümüzde birçok yerde kullanılan bu standart, güvenilirliği ispatlanmış güncel bir algoritmadır. Yüksek güvenlik sağlanabilmesi için donanım kullanılması gerekmektedir, zira yazılım ile gerçeklenen güvenlik sistemleri günümüzde bilgisayar ağlarının ne denli geniş, internetin, işletim sistemlerinin ne büyük açıkları olduğu düşünülürse, donanım yazılımsal şifrelemeden kat kat daha güvenlidir. Ayrıca donanım olarak seçilen FPGA hem çok yüksek hızlara çıkabilmekte hem de kolayca ve ucuza bulunabilmektedir. Yapılacak donanım ev telefonu hattına bağlanabilir çünkü ev telefonu hatları ses harici verileri de taşıyabilmektedir, örnek olarak ADSL gibi yüksek hızlı iletişim mümkündür.

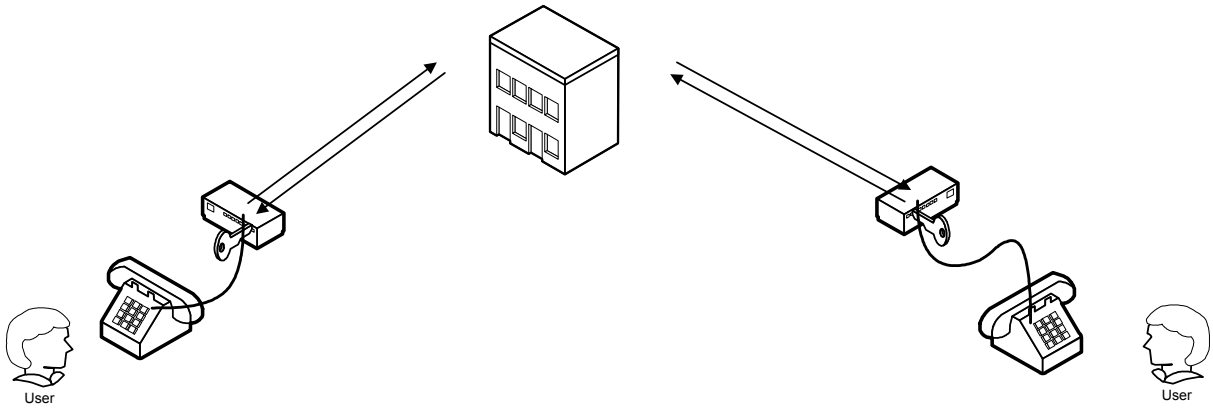
## 5.2 Sistemin Genel Yapısı

Haberleşmenin kolayca sağlanabilmesi için normalde kullandığımız ev telefonlarının bağlı olduğu telekom operatörünü kullanmayı seçtik. Yapılacak telekonferansta herkes operatör aracılığı ile birbirine bağlı olacak ancak operatöre sadece şifreli metin gideceği için elinde anahtar olmadan dinlemesinin anlamı olmayacaktır. Ayrıca konuşmacılar önceden anahtarı biliyor olmalıdırlar. Simetrik şifreleme yapılacağı için tek anahtar kullanılmaktadır.

Sistemi basitleştirmek için sadece iki kullanıcı olduğunu düşünelim. İki kullanıcıda da hem şifreleme hem de şifre çözme işleminin yapılabiliyor olması gerekmektedir.

Yani her telefona konacak olan FPGA çipine hem şifreleme hem de şifre çözme devresi yüklenecektir.

Kullanıcılardan biri diğeri ile konuştuğu sırada sesi analog sayısal çeviriciden geçerek sayısal veri haline getirilecektir. Bu veri 8'er bit parçalar ile toplanacak 128 bit olduğunda şifrelenecek ve yine 8'er bit olarak dışarı yollanacaktır. Burada veri kaybının yaşanmamasının püf noktası şifreleme ve şifre çözme işlemlerinin verileri alma ya da yazma işlemlerine göre çok hızlı bir şekilde gerçekleşmesidir. Bu sayede şifreleme işlemi sırasındaki çok küçük gecikme zaten insan kulağının hissedemeyeceği kadar kısa olduğu için herhangi bir sorun yaşanmayacaktır. Gönderilen şifreli veri karşı tarafa ulaştığında bu kez onun cihazına girecek şifre çözme işleminden geçecek ve en son sayısal analog çeviriciden de geçerek kulağına ses olarak ulaşacaktır. Anahtar cihazlara eklenebilecek olan 10 rakam ve 6 harfli bir klavye aracılığı ile girilebilir(Şekil 5.1) [6].




**Şekil 5.1** Güveli İletişim Örneği

# 6. FPGA DENEME KARTI ÜZERİNDE ÇALIŞTIRILMASI

## 6.1 Giriş

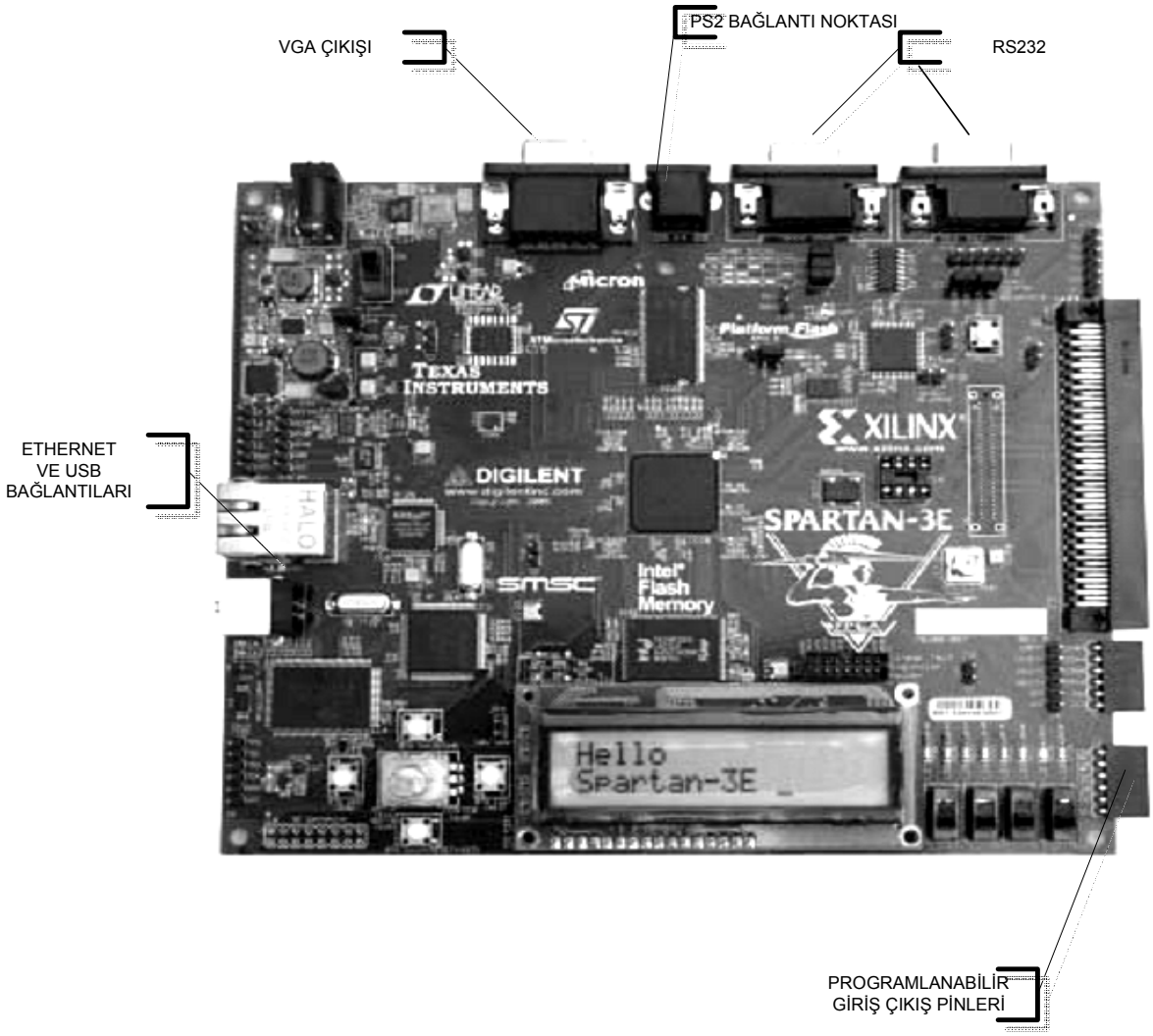
Devrenin gerçeklenmesi için Xilinx firmasının ürettiği Spartan 3E FPGA'sı kullanılmıştır. Bu FPGA yaklaşık 500 bin kapı elemanı içermekte ve 320 tane pini bulunmaktadır. FPGA'lara yükleme yapma, üzerinde çeşitli testler gerçekleştirmek için deneme kartından (Spartan 3E Starter Kit) yararlanıldı. Kart üzerinde bulunan FPGA'ya yazılan kodlar yüklenmiş ve çalıştırılarak doğruluğu gözlenmiştir. Öncelikle 6.2'de deneme kartının çeşitli özelliklerinden bahsedilmiştir. Ardından yapılan yüklemeler ve gözlemler anlatılmıştır.

## 6.2 Spartan 3E Deneme Kartı

Bu kart, üzerinde çeşitli uygulamalar geliştirilmek için tasarlanmış çok fonksiyonel bir deneme kartıdır. Üzerinde 1 VGA çıkışı, 2 RS232 çıkışı, 1 adet 100 pinli Hirose çıkışı, 12 adet programlanabilir giriş/çıkış pini, 4 sürgülü düğme, 4 bas çek tuş, 8 adet led, 1 adet çevirmeli düğme, USB ve Ethernet girişi, PS2 girişi, SMA anten girişi, saat osilatör girişi vs. giriş çıkış bulunmaktadır. Ayrıca 1 adet 16x2 karakter LCD ekranı, 1'er de ADC ve DAC'si bulunmaktadır (Şekil 6.1). Bunların dışında üzerinde 512 Mbit DDR SDRAM, 128 Mbit Flash PROM, 16 Mbit Serial Flash ve 50 MHz saat osilatörü bulunmaktadır. Bu devrede de üzerinde bulunan 50 MHz'lik saat kaynağından yararlanılmıştır [5].

Bu tasarımda kartın birkaç fonksiyonu kullanılmıştır. Bunlar; yükleme yapmak için USB girişi, başla, reset gibi girişler için düğmeler, hangi işlemin yapıldığını anlamak için ledler, oluşan şifreli metnin ve çözüm işleminden sonra oluşan metnin görülmesi için karakter LCD ekran ve son olarak iki FPGA deneme kartının birbiriyle iletişim kurması için programlanabilir giriş çıkış pinleridir.

LCD ekranın kullanılması için de özel olarak bir modül tasarlanmıştır. VHDL dili kullanılan bu modül ile kart üzerindeki LCD ekran istenilen değeri göstermesi için programlanmıştır.

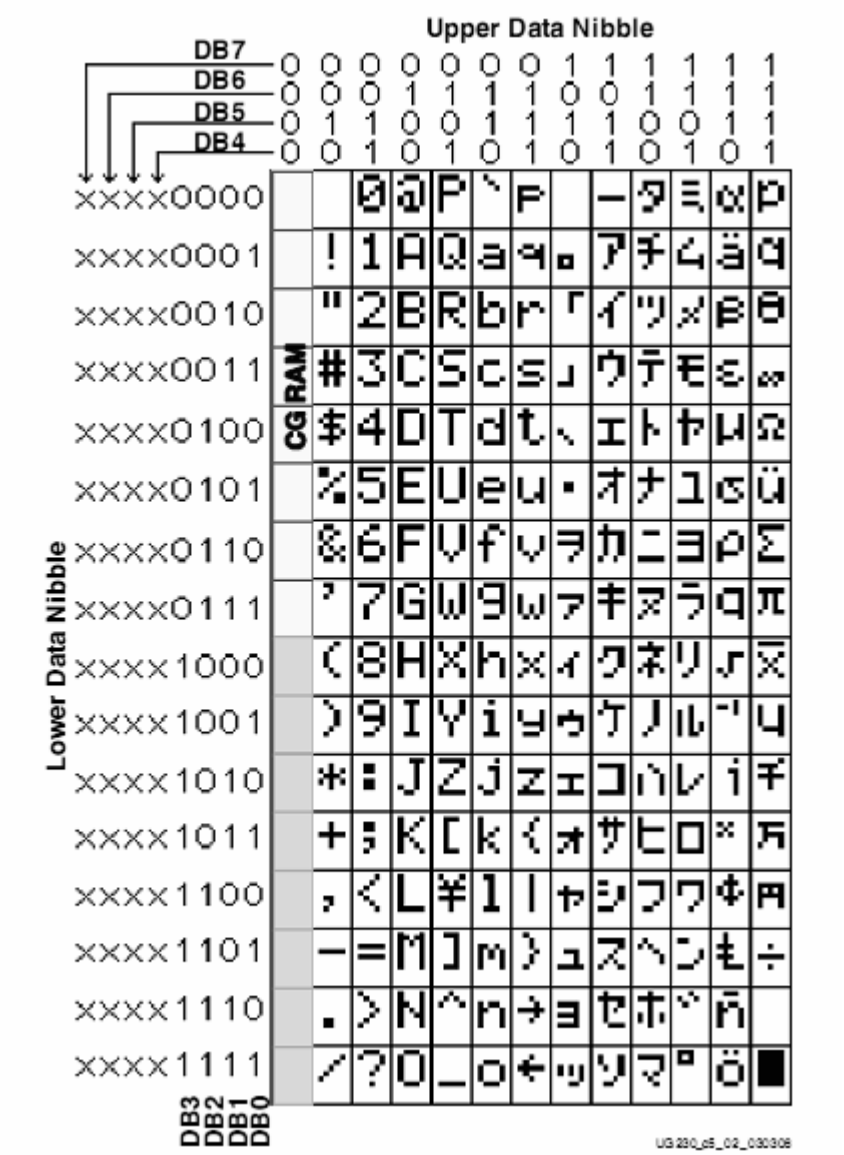



**Şekil 6.1** Spartan 3E Deneme Kartı [4]

### 6.2.1 Spartan 3E Karakter LCD Ekranı

Kart üzerindeki LCD ekran 2 satır x 16 karakter gösterebilmektedir. Gösterilen harf ve rakamlar İngilizce ASCII kodu ve Japonca Kana alfabesine göre yapılandırılmışlardır. Kart üzerinde LCD ekranın grafik kontrolörü olarak Sitronix ST7066U kullanılmış. Şekil 6.2'de LCD ekranın karakter seti görülmektedir. Tasarımda ilgili baytları göstermesi için {0011XXXX} adresleri kullanılmıştır. Bu kullanım sonucu 0'dan 9'a kadar olan rakamlar aynı şekilde; a,b,c,d,e,f harfleri için ':', ';', '<', '=', '>' ve '?' sembolleri karşılık gelmiştir. Ayrıca 128 bitin tamamı görülemeyeceği için sadece ilk 32 bitinin görülmesi sağlanmıştır ve bu da yeterli

olmaktadır. Çünkü şifreleme sırasında 1 bit bile farklılık oluşsa 128 bitin tamamı farklı değer almaktadır. Yani ilk 32 biti doğru olan verinin tamamının da doğru olacağını söyleyebiliriz [5].




**Şekil 6.2** Spartan 3E LCD Ekran Karakter Seti [4]

### 6.2.2 ADC ve DAC'lerin Kullanımı

Kart üzerinde bir tane sayısal analog bir tane de analog sayısal dönüştürücü bulunmaktadır. Analog sayısal dönüştürücü, analog veriyi 14 bitlik sayısal veriye çevirmektedir. Bu işlemi denklem (6.1)'deki gibi yapmaktadır.

$$D[13:0] = \text{Kazanç} \times \frac{V_{in} - 1.65V}{1.25V} \times 8192 \qquad (6.1)$$

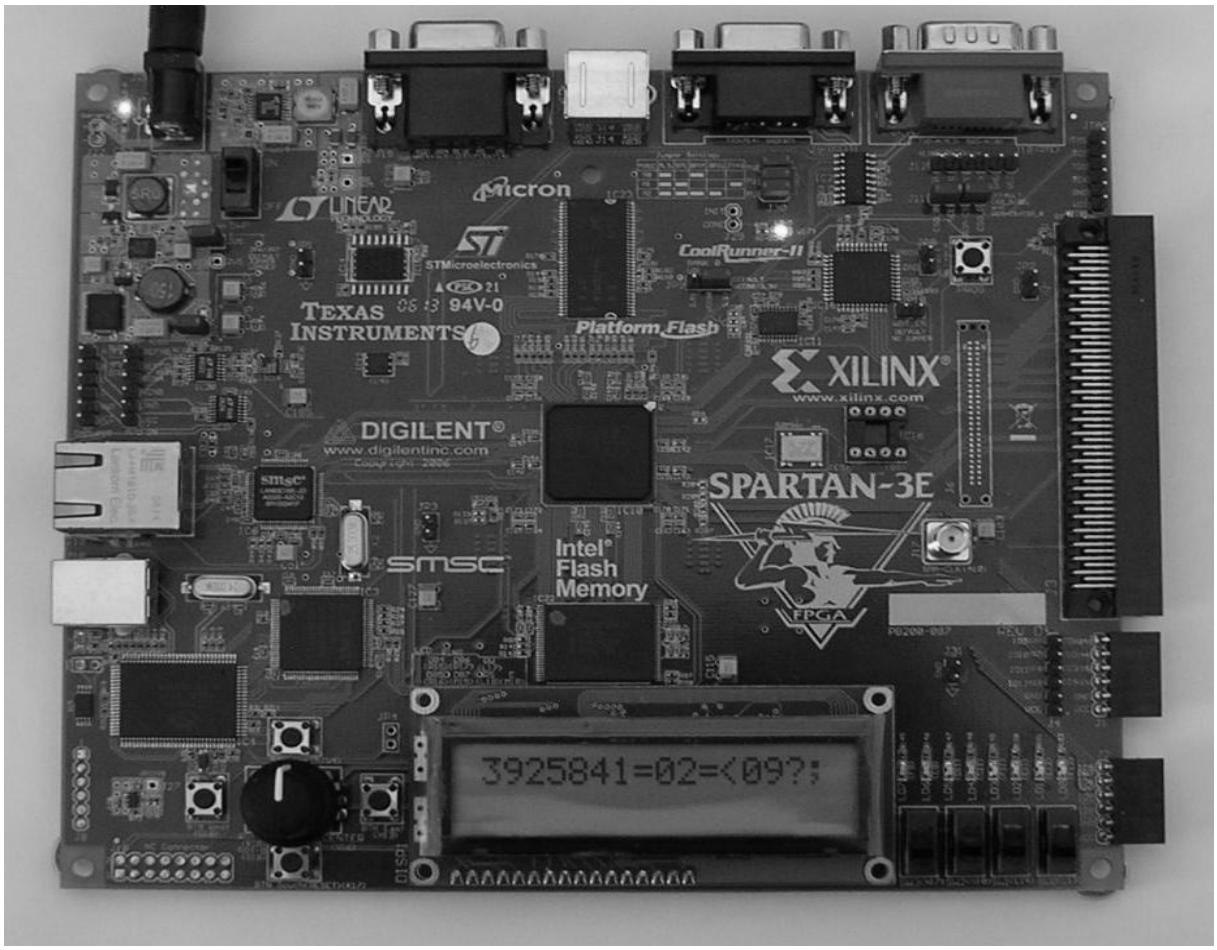
Buradaki kazan analog verinin ilk öce girdiği ayarlanabilir preamplifikatörün kazancıdır. Sonuçta oluşan 14 bit, 2'ye tümleyen şeklinde oluşmaktadır ve $-2^{13}$ ile $2^{13}-1$ arasında değişmektedir.

Sayısal analog çevirici ise 12 bit işaretsiz veriyi analog değer dönüştürmektedir. Bunu denklem (6.2)'ye göre yapmaktadır [5].

$$Vout = \frac{D[11:0]}{4096} \times V_{\text{Re } ferans} \tag{6.2}$$

$V_{Referans}$ = 3.3 V veya 2.5 V

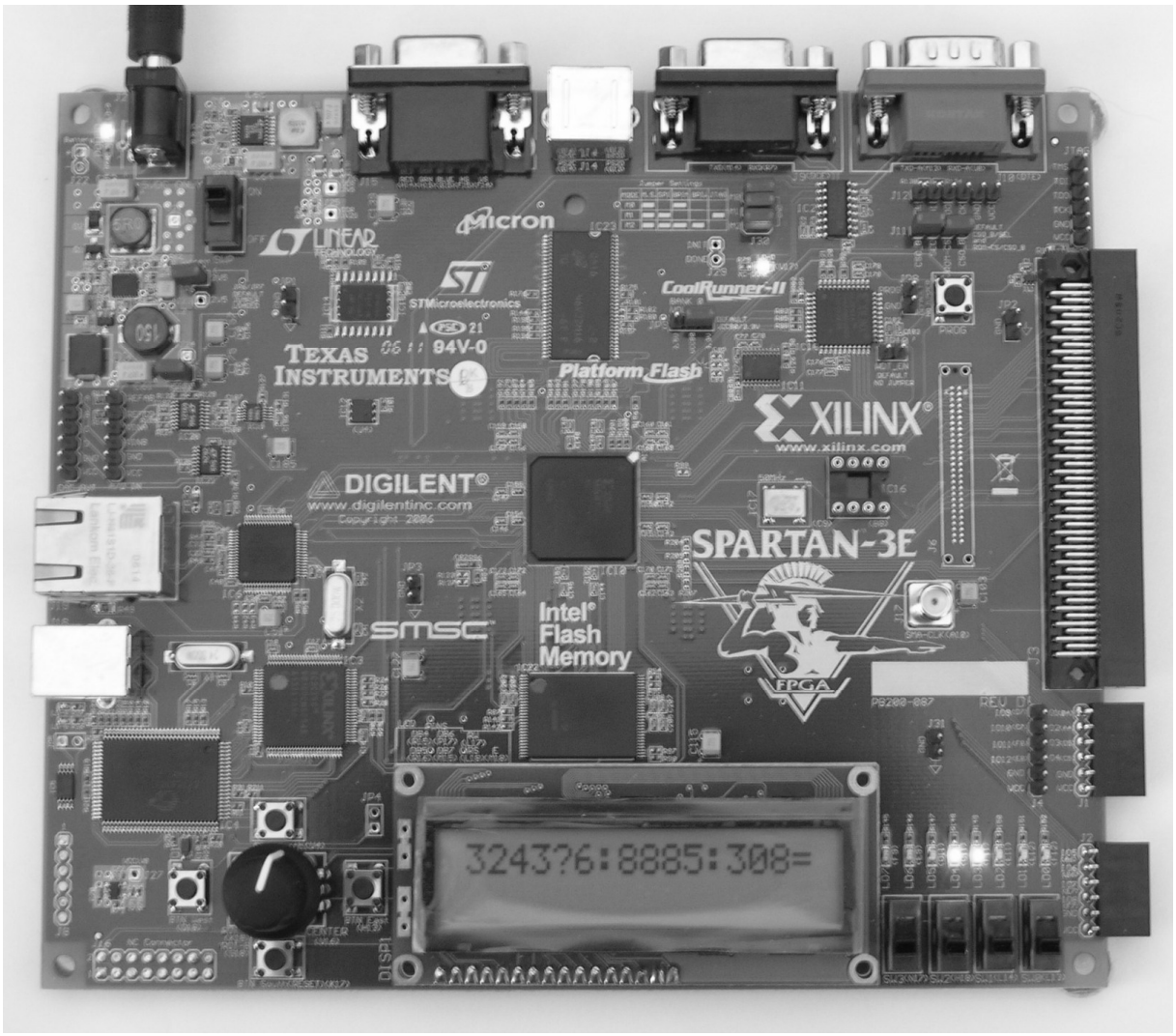
## 6.3 Kart Üzerine Yükleme Yapılması




**Şekil 6.3** Şifreleme İşleminin Kart Üzerinde Çalıştırılması

Yükleme işlemi için USB arabirimi kullanılmış olup 1. karta şifreleme 2. karta da şifre çözme işlemini yapan kodlar yüklenmiştir. 2 kartta da içine gömülen anahtar ve girişler işlemden geçerek çıkışları kart üzerindeki LCD ekrana yazdırıldı. Şekil 6.4'te şifreleme işleminin sonucunun LCD ekrana yazılması görülüyor.

Aynı şekilde şifre çözme işlemi de Şekil 6.3'te LCD ekranda görülen değer ve anahtar da giriş olarak girilerek tekrar düz metin elde edildi. Bu işlemin resmi de Şekil 6.4'de görülmektedir.




**Şekil 6.4** Şifre Çözme İşleminin Kart Üzerinde Çalıştırılması

Bu iki işlemin de çalıştığı kesinleştikten sonra sırada kartların birbirleriyle olan iletişimi geliyor. Bölüm 6.4'de kartlar arasında nasıl iletişim kurulduğu anlatılmaktadır.

### 6.4 Kartlar Arasındaki İletişimin Gerçekleştirilmesi

İletişim sağlanması için tüm giriş çıkışları belirleyen, saat frekansını, reset komutunu aldığı 'ucf' dosyası yazılmıştır. Yani hangi pinin nereye bağlanacağına karar verildiği dosya yazılmıştır. Tablo 6.1'de şifreleme için kullanılan ucf dosyası görülmektedir.

**Tablo 6.1** Şifreleme İşlemi İçin Pin Bağlantıları

```
NET "clk"  LOC = "c9"  ;
#NET "finish"  LOC = "E9"  ;
NET "lcd_4"  LOC = "r15"  ;
NET "lcd_5"  LOC = "r16"  ;
NET "lcd_6"  LOC = "p17"  ;
NET "lcd_7"  LOC = "m15"  ;
NET "lcd_e"  LOC = "m18"  ;
NET "lcd_rs"  LOC = "l18"  ;
NET "lcd_rw"  LOC = "l17"  ;
NET "sf_ce0"  LOC = "d16"  ;
NET "rst"  LOC = "n17"  ;
#NET "basla1"  LOC = "k17"  ;
NET "dout<0>"  LOC = "b4" | IOSTANDARD = LVTTL  | SLEW = SLOW  | DRIVE = 6 ;
NET "dout<1>"  LOC = "a4" | IOSTANDARD = LVTTL  | SLEW = SLOW  | DRIVE = 6 ;
NET "dout<2>"  LOC = "d5" | IOSTANDARD = LVTTL  | SLEW = SLOW  | DRIVE = 6 ;
NET "dout<3>"  LOC = "c5" | IOSTANDARD = LVTTL  | SLEW = SLOW  | DRIVE = 6 ;

NET "basla1"  LOC = "D18";# | IOSTANDARD = LVTTL  | SLEW = SLOW  | DRIVE = 6
;
NET "f1tof2"  LOC = "B6" | IOSTANDARD = LVTTL  | SLEW = SLOW  | DRIVE = 6 ;
NET "f2tof1"  LOC = "e7" | IOSTANDARD = LVTTL  | SLEW = SLOW  | DRIVE = 6 ;
NET "say<4>" LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "say<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "say<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "say<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "say<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
```

Tabloda görülen değerlerden ilki saat frekansının verildiği c9 pini. Ardından gelen 8 pin LCD ekranı kullanmak için ayarlanan pinler. Reset komutu için bir tuşun bağlı olduğu pin numarası verilmiş. 4 adet data çıkışının yapılacağı çıkış pinleri, kontrol pinleri ve son 5 tane ledleri kontrol eden pinler girilmiş. İşte bu şekilde istediğimiz pinleri istediğimiz giriş çıkışla ilişkilendiren ucf dosyasını hazırladıktan sonra FPGA'yı istediğimiz şekilde kontrol edebiliyoruz.

2 türlü iletişim şekli gerçekleştirildi. İlki 2 adet kontrol işareti kullanmakta. Alıcı aldım işareti için ilgili bayrağını kaldırmakta ve gönderici de gönderi hazır bayrağını kaldırmakta. Bu şekilde iletişim sorunsuz bir şekilde gerçekleştirilmekte. 2. yol ise sadece tek bir kontrol işareti kullanılarak yapılan tek yönlü iletişim. Bu yolda gönderi hazır işareti verilmekte ve bu işareti gören alıcı veriyi kaydetmektedir. Bu yol daha kullanışlıdır çünkü çift yönlü iletişime geçildiğinde 2 adet kontrol işareti gerekecektir. Bir önceki yolda çift yönlü iletişime geçildiğinde 4 tane kontrol işareti kullanılması gerekecektir. Bu yüzden 2. yol tercih edilmiştir. Tablo 6.2'de örnek olarak kullanılan değerlerin adım adım üretilmesi gösterilmektedir.

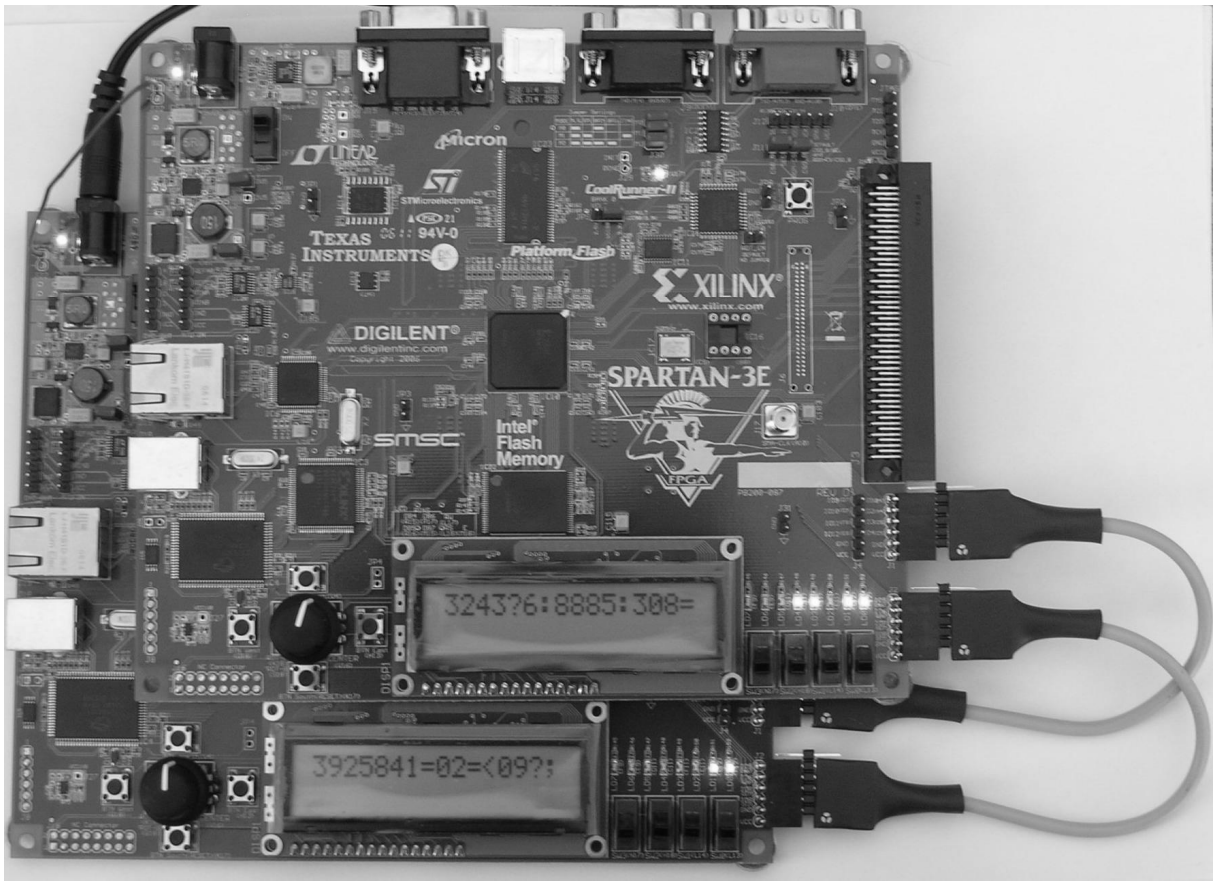
**Tablo 6.2** Şifreleme Örneği [3]

| Tur Sayısı | Başlangıç Değ. | S-kutusundan S. | Satır Kayd. Sonra | Sütun Karış. S. | Tur Anahtarı | |
|---|---|---|---|---|---|---|
| | **32 88 31 e0** | | | | **2b 28 ab 9** | |
| | **43 5a 31 37** | | | | **7e ae f7 cf** | ⊕ |
| input | **f6 30 98 7** | | | | **15 d2 15 4f** | |
| | **a8 8d a2 34** | | | | **16 a6 88 3c** | |
| | 19 a0 9a e9 | d4 e0 b8 1e | d4 e0 b8 1e | 4 e0 48 28 | a0 88 23 2a | |
| 1 | 3d f4 c6 f8 | 27 bf b4 41 | bf b4 41 27 | 66 cb f8 6 | fa 54 a3 6c | ⊕ = |
| | e3 e2 8d 48 | 11 98 5d 52 | 5d 52 11 98 | 81 19 d3 26 | fe 2c 39 76 | |
| | be 2b 2a 8 | ae f1 e5 30 | 30 ae f1 e5 | e5 9a 7a 4c | 17 b1 39 5 | |
| | a4 68 6b 2 | 49 45 7f 77 | 49 45 7f 77 | 58 1b db 1b | f2 7a 59 73 | |
| 2 | 9c 9f 5b 6a | de db 39 2 | db 39 2 de | 4d 4b e7 6b | c2 96 35 59 | ⊕ = |
| | 7f 35 ea 50 | d2 96 87 53 | 87 53 d2 96 | ca 5a ca b0 | 95 b9 80 f6 | |
| | f2 2b 43 49 | 89 f1 1a 3b | 3b 89 f1 1a | f1 ac a8 e5 | f2 43 7a 7f | |
| | aa 61 82 68 | ac ef 13 45 | ac ef 13 45 | 75 20 53 bb | 3d 47 1e 6d | |
| 3 | 8f dd d2 32 | 73 c1 b5 23 | c1 b5 23 73 | ec 0b c0 25 | 80 16 23 7a | ⊕ = |
| | 5f e3 4a 46 | cf 11 d6 5a | d6 5a cf 11 | 9 63 cf d0 | 47 fe 7e 88 | |
| | 3 ef d2 9a | 7b df b5 b8 | b8 7b df b5 | 93 33 7c dc | 7d 3e 44 3b | |
| | 48 67 4d d6 | 52 85 e3 f6 | 52 85 e3 f6 | 0f 60 6f 5e | ef a8 b6 db | |
| 4 | 6c 1d e3 5f | 50 a4 11 cf | a4 11 cf 50 | d6 31 c0 b3 | 44 52 71 0b | ⊕ = |
| | 4e 9d b1 58 | 2f 5e c8 6a | c8 6a 2f 5e | da 38 10 13 | a5 5b 25 ad | |
| | ee 0d 38 e7 | 28 d7 7 94 | 94 28 d7 7 | a9 bf 6b 1 | 41 7f 3b 0 | |
| | e0 c8 d9 85 | e1 e8 35 97 | e1 e8 35 97 | 25 bd b6 4c | d4 7c ca 11 | |
| 5 | 92 63 b1 b8 | 4f fb c8 6c | fb c8 6c 4f | d1 11 3a 4c | d1 83 f2 f9 | ⊕ = |
| | 7f 63 35 be | d2 fb 96 ae | 96 ae d2 fb | a9 d1 33 c0 | c6 9d b8 15 | |

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | e8 | c0 | 50 | 1 | 9b | ba | 53 | 7c | 7c | 9b | ba | 53 | ad | 68 | 8e | b0 | | f8 | 87 | bc | bc |
| | f1 | c1 | 7c | 5d | a1 | 78 | 10 | 4c | a1 | 78 | 10 | 4c | 4b | 2c | 33 | 37 | | 6d | 11 | db | ca |
| 6 | 0 | 92 | c8 | b5 | 63 | 4f | e8 | d5 | 4f | e8 | d5 | 63 | 86 | 4a | 9d | d2 | ⊕ | 88 | 0b | f9 | 0 | = |
| | 6f | 4c | 8b | d5 | a8 | 29 | 3d | 3 | 3d | 3 | a8 | 29 | 8d | 89 | f4 | 18 | | a3 | 3e | 86 | 93 | |
| | 55 | ef | 32 | 0c | fc | df | 23 | fe | fe | fc | df | 23 | 6d | 80 | e8 | d8 | | 7a | fd | 41 | fd | |
| | 26 | 3d | e8 | fd | f7 | 27 | 9b | 54 | f7 | 27 | 9b | 54 | 14 | 46 | 27 | 34 | | 4e | 5f | 84 | 4e | |
| 7 | 0e | 41 | 64 | d2 | ab | 83 | 43 | b5 | 83 | 43 | b5 | ab | 15 | 16 | 46 | 2a | ⊕ | 54 | 5f | a6 | a6 | = |
| | 2e | b7 | 72 | 8b | 31 | a9 | 40 | 3d | 40 | 3d | 31 | a9 | b5 | 15 | 56 | d8 | | f7 | c9 | 4f | dc | |
| | 17 | 7d | a9 | 25 | f0 | ff | d3 | 3f | 3f | f0 | ff | d3 | bf | ec | d7 | 43 | | 0e | f3 | b2 | 4f | |
| | 5a | 19 | a3 | 7a | be | d4 | 0a | da | be | d4 | 0a | da | 0 | b1 | 54 | fa | | ea | b5 | 31 | 7f | |
| 8 | 41 | 49 | e0 | 8c | 83 | 3b | e1 | 64 | 3b | e1 | 64 | 83 | 51 | c8 | 76 | 1b | ⊕ | d2 | 8d | 2b | 8d | = |
| | 42 | dc | 19 | 4 | 2c | 86 | d4 | f2 | d4 | f2 | 2c | 86 | 2f | 89 | 6d | 99 | | 73 | ba | f5 | 29 | |
| | b1 | 1f | 65 | 0c | c8 | c0 | 4d | fe | fe | c8 | c0 | 4d | d1 | ff | cd | ea | | 21 | d2 | 60 | 2f | |
| | ea | 4 | 65 | 85 | 87 | f2 | 4d | 97 | 87 | f2 | 4d | 97 | 47 | 40 | a3 | 4c | | ac | 19 | 28 | 57 | |
| 9 | 83 | 45 | 5d | 96 | ec | 6e | 4c | 90 | 6e | 4c | 90 | ec | 37 | d4 | 70 | 9f | ⊕ | 77 | fa | d1 | 5c | = |
| | 5c | 33 | 98 | b0 | 4a | c3 | 46 | e7 | 46 | e7 | 4a | c3 | 94 | e4 | 3a | 42 | | 66 | dc | 29 | 0 | |
| | f0 | 2d | ad | c5 | 8c | d8 | 95 | a6 | a6 | 8c | d8 | 95 | ed | a5 | a6 | bc | | f3 | 21 | 41 | 6e | |
| | eb | 59 | 8b | 1b | e9 | cb | 3d | af | e9 | cb | 3d | af | | | | | | d0 | c9 | e1 | b6 | |
| 10 | 40 | 2e | a1 | c3 | 9 | 31 | 32 | 2e | 31 | 32 | 2e | 9 | | | | | ⊕ | 14 | ee | 3f | 63 | = |
| | f2 | 38 | 13 | 42 | 89 | 7 | 7d | 2c | 7d | 2c | 89 | 7 | | | | | | f9 | 25 | 0c | 0c | |
| | 1e | 84 | e7 | d2 | 72 | 5f | 94 | b5 | b5 | 72 | 5f | 94 | | | | | | a8 | 89 | c8 | a6 | |

| | | | |
|---|---|---|---|
| **39** | **2** | **dc** | **19** |
| **25** | **dc** | **11** | **6a** |
| **84** | **9** | **85** | **0b** |
| **1d** | **fb** | **97** | **32** |

Tablo 6.2'deki şifreleme örneği FPGA kartları arasında iletişim sağlanarak denenmiştir ve Şekil 6.5'da görülmektedir. Son yazılan modüller de eklendikten sonra yani katlayıcı ve LCD'ye yazıcı modül sonrasında şifreleme işlemi için sentez sonucuna göre 1074 Slice yer kaplamakta ve en fazla 133 MHz hızda çalışmaktadır. Şifre çözme işlemi ise 1311 Slice yer kaplamakta ve 87 MHz hızda çalışabilmektedir.




**Şekil 6.5** FPGA Kartları Arasında İletişim Kurulması

## 7. SONUÇLAR ve TARTIŞMA

İlk olarak, bu çalışmada AES algoritmasının FPGA üzerinde tasarım ve gerçeklenme işi yapılmıştır. Ardından bir deneme kartı kullanarak çeşitli uygulamalar denenmiştir. Tek başına şifreleme ve şifre çözme işlemi, birbirleriyle iletişim kurarak şifreli metin gönderme, şifreli metini çözerek düz metini elde etme, kart üzerindeki LCD ekran kullanılarak görsel olarak takip edebilme, analog sayısal ve sayısal analog dönüştürücüler kullanarak analog veriyi şifreleme ve tekrar şifreli veriyi çözerek analog veriyi elde etme vb. çeşitli uygulamalar gerçekleştirilmiştir. Görülüyor ki FPGA'lar kullanarak yüksek hızlı ve doğruluklu birçok çalışma yapabiliriz. Yüksek doğruluğun en önemli olduğu alan olan Kriptografide de FPGA'lardan yararlanmak gereklidir. Günümüzün en yüksek güvenlikli simetrik algoritmalarından olan AES ile hızlı, basit ve güvenli iletişimin mümkün olduğunu görmüş olduk. Eğer çalışma toparlanıp bir cihaz haline getirilirse insanlar internet haricinde yazılımsal şifrelemelerden çok daha güvenli olan donanımsal şifreleme ile ev telefonlarında da şifreli iletişim olanağından faydalanarak daha güvenli iletişim kurabileceklerdir.

Çalışmada donanım tasarlama dili olarak VHDL kullanıldı. VHDL bilimsel çalışmalarda Verilog'dan daha fazla kullanılmaktadır çünkü daha karmaşık yapılarda VHDL daha etkilidir. Sonuçların analizi ve simülasyon aşamalarında da Mentor Graphics'in ModelSim programı kullanıldı. Bu program da diğer simülatörlere göre daha yüksek doğruluk sağlamaktadır.

**KAYNAKLAR**


[1] **Daemen, J. ve Rijmen, V.**, 2002. The Design of Rijhndael AES-The Advanced Encryption Standart

[2] **Gladman B.**, 2002. A Specification for Rijndael, the AES Algorithm Mayıs 2002., s. 2 - 20

[3] **FIPS 197**, 2001. Advanced Encryption Standard. National Institute of Standarts and Technology (NIST).

[4] **J¨org J. Buchholz**, 2001. Matlab Implementation of the Advanced Encryption Standard report , Hochschule Bremen Almanya s. 5 – 28

[5] **Xilinx**, 2006. Spartan-3E Starter Kit Board User Guide, UG230 (v1.0) March 9.

[6] **Steer D. G., Strawczynski L., Diffie W., Wiener M.**, 1990. A secure audio teleconference system, Proceedings on Advances in cryptology, February , Santa Barbara, California, United States p.520-528.

[7] **Menezes A., Oorschot van P., Vanstone S.**, 1996. Handbook of Applied Cryptography, CRC Press, Chapter 1.

[8] **Yerlikaya, T., Buluş, E., Arda, D.,** 2004. AES Aday şifreleme Algoritmalarının Yazılım ve Donanım Performans Karşılaştırması ve Uygulamalar, Elektrik Elektronik Bilgisayar Mühendisliği Sempozyumu (ELECO 2004), Bursa-TÜRKYE, s. 1

[9] **FIPS 46-3,** 1999. Data Encryption Standard. National Institute of Standarts and Technology (NIST).

**EK-A**

Çalışmada ek olarak verilmiş CD-ROM içerisinde, şifreleme, şifre çözme, LCD kullanımı ve katlayıcı işlemlerini FPGA üzerinde gerçekleyen VHDL dosyaları bulunmaktadır.

## ÖZGEÇMİŞ

11.01.1986 yılında Tekirdağ'ın Saray ilçesinde dünyaya gelen Onur Tiryakioğlu 1991'de bu ilçede ilkokula başladı. Ortaokulu Çorlu Anadolu Lisesinde tamamladıktan sonra liseyi Edirne Fen Lisesinde yatılı olarak okudu. Mezun olduğu yıl 2003'te İstanbul Teknik Üniversitesi Elektronik Mühendisliğini kazandı ve yabancı dil hazırlıkla beraber 2008'de mezun oldu. İlgi alanları arasında sayısal dizayn, kriptografi, gömülü sistem tasarımı bulunmakta. Tasarım için VHDL ve Verilog donanım tanımlama dillerini, yabancı dil olarak da İngilizce ve Japoncayı bilmektedir.