İSTANBUL TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

KRİPTOLOJİ UYGULAMALARINDA KULLANILACAK BİR İŞLEMCİNİN TASARLANARAK FPGA ÜZERİNDE GERÇEKLENMESİ

BİTİRME ÖDEVİ

ONUR ŞAHİN

040070112

Bölümü: Elektronik ve Haberleşme Mühendisliği Bölümü

Programı: Elektronik Mühendisliği

Danışmanı: Doç. Dr. Sıddıka Berna Örs Yalçın

MAYIS 2012

ÖNSÖZ

Tez sürecim boyunca yardımını benden esiremeyen, değerli vaktini ayırıp bilgi ve tecrübesiyle bana yol gösteren saygı değer danışman hocam Doç. Dr. Sıddıka Berna Örs Yalçın'a saygı ve teşekkürlerimi sunarım.

Ayrıca, eğitim hayatım boyunca maddi ve manevi destekleriyle her zaman yanımda olan aileme de teşekkürü bir borç bilirim.

Onur ŞAHİN

Mayıs 2012

İÇİNDEKİLER

KISALTMALAR

FPGA	**:** Field Programmable Gate Array
AES	**:** Advanced Encryption Standard
ASIP	**:** Application Specific Instruction Set Processor
GPP	**:** General Purpose Processor
PC	**:** Personal Computer
RISC	**:** Reduced Instruction Set Computing
CISC	**:** Complex Instruction Set Computing
ALU	**:** Arithmetic Logic Unit
MİB	**:** Mikroişlemci Birimi
BRAM	**:** Block RAM

KRİPTOLOJİ UYGULAMALARINDA KULLANILACAK BİR İŞLEMCİNİN TASARLANARAK FPGA ÜZERİNDE GERÇEKLENMESİ

ÖZET

Genel amaçlı işlemciler, birçok farklı alandaki uygulamalarda kullanılabilmeye olanak sağlaması açısından, hemen hemen bütün uygulamalarda gerçekleştirilmesi gerekli olan bazı temel ve genel amaçlı buyrukları komut setinde barındıran işlemcilerdir. Bu tip işlemciler genel amaçlı kullanıma uygun olarak tasarlandıklarından geniş kullanım alanlarına sahiptirler. Fakat, bu işlemcilerin sağladıkları komutlar, bazı özel uygulamaları (kriptoloji, işaret işleme….) gerçekleştirmede performans, kaynakların verimli kullanımı ve enerji tüketimi gibi unsurlar göz önünde bulundurulduğunda beklenen ölçütlerin sağlanmasında yeterli olamamaktadırlar. Ayrıca genel amaçlı işlemciler, güvenlik kaygısı oluşturan uygulamalarda tehditlere açık olduklarından tercih edilmemektedirler. Bu uygulamada genel amaçlı komutlarının yanı sıra gelişmiş şifreleme standardı (Advanced Encryption Standard-AES) algoritmasının hızlı ve kolay bir şekilde gerçekleştirilmesini sağlayacak olan komutları da barındıran bir işlemci tasarlanacaktır.

AES algoritması günümüzde kriptoloji uygulamalarında en yaygın olarak kullanılan ve en çok kabul gören şifreleme algoritmalarından biridir. AES algoritmasına özel komutların eklenmesi ile işlemcinin veri şifreleme/şifre çözme işlemleri için performansının artırılması amaçlanmaktadır.

Bu çalışmada ilk olarak AES komutlarını gerçekleştirecek olan donanımlar tasarlanmış ve yapılan simülasyonlar ile test edilmiştir. Bu komutların ekleneceği genel amaçlı işlemci de yüksek hızlı tümleşik devre tanımlama dili (Very high speed integrated circuit Hardware Description Language-VHDL) kullanılarak gerçekleştirilmiştir. Oluşturulan işlemcinin donanımsal yapısı ve işleyişi tez içeriğinde ayrıntılı bir şekilde ele alınmıştır. Projenin sonunda kendi geliştirdiğimiz

işlemci üzerinde koşacak bir şifreleme programı yazılmış ve başarıyla gerçekleştirilmiştir.

DESIGN AND IMPLEMENTATION OF A CRYPTOGRAPHIC ASIP ON FPGA

SUMMARY

In order to be used in many areas, general purpose processors (GPPs) have instruction sets that contain most of common operations need by all kind of applications. As they are designed to serve general facilities for any kind of application, their applications cover wide range of areas. However, considering performance, efficient usage of sources and power consumption, these processors might not satisfy requirement for some specific applications such as cryptography, signal processing etc.. Moreover, general purpose processors might lead to security problems as they are open to any attacks. In this project, a processor with instruction set that is extended by instructions for performing AES cryptography algorithm is to be designed.

Advanced Encryption Standard (AES) is one of the most widely used and accepted cryptography algorithm used to encrypt/decrypt data blocks. Aim of this project is to increase the performance of a general purpose processor for performing AES algorithm by adding special instructions.

At the initial stages of the project, hardware units that will perform AES operations are designed and verified by simulations. The general purpose processor that we will extend its instruction set by AES functions, is also developed by VHDL hardware design language. Functionality and hardware architecture of this processor is detailed in the following sections of this thesis. Finally, we developed a program to perform AES algorithm using instructions from our instruction-set. After running this program in our processor, successful encryption is observed.

1. GİRİŞ

Kişisel bilgisayarlarda (PC) kullanılan işlemciler günümüzde üretilen milyarlarca işlemcilerin yalnızca küçük bir bölümünü oluşturmaktadır. İşlemciler artık hemen hemen bütün elektronik cihazlar (cep telefonları, PDA, kameralar, duyarga düğümleri…) üzerinde bulunmaktadır ve kullanılan işlemcilerin özellikleri ve türü uygulamadan uygulamaya değişmektedir. Bu tip uygulamaların geliştirilebilmesi açısından kullanılan iki temel yaklaşım bir genel amaçlı işlemci (General Purpose Processor, GPP) kullanmak ve ya uygulamaya özel tümleşik devre (Application Specific Integrated Circuit, ASIC) ile gerekli fonksiyonların gerçekleştirilmesini sağlamaktır.

Genel amaçlı işlemciler kolay programlanabilme özelliklerinden dolayı uygulama alanları çok geniştir fakat kullanılacağı uygulamaya göre bazı dezavantajları da birlikte getirebilmektedir. Örneğin, kriptografi algoritmalarının bu tip işlemciler üzerinde yazılımsal olarak gerçekleştirilmesi esneklik açısından kolaylık sağlasa da, veri işleme gücü, bellek ve enerji bakımından kısıtlı olan bu işlemciler performans açısından oldukça yetersiz kalmaktadır [1]. ASIC'ler ile de bu performans ve enerji sorunları aşılabilmektedir fakat günümüzde ASIC tasarımı ve üretimi oldukça maliyetli bir iştir. Bu ikilemin, son yıllarda kullanımlarında önemli bir artış gösteren uygulamaya özel komut setli işlemciler (Application Specific Instruction Set Processor, ASIP) ile çözümü yoluna gidilmektedir. Bu tip işlemcilerde geliştirilen uygulamalar, genellikle standart yazılım uygulamalarına göre daha yüksek performans sağlamasının yanı sıra yardımcı işlemci (co-processor) gibi sadece belirli bir uygulamaya adanmış donanım bloklarına göre de daha az silikon alanı kaplamaktadır [2]. ASIP'ler yazılımın sağladığı esneklik ile donanımların sağladığı performans avantajlarından birlikte yararlanılabilmesini sağlayan işlemcilerdir.

Ayrıca, günümüzde çoğu modern elektronik araçlar (PC, cep telefonları, ağ yönlendiricileri, akıllı kartlar, ağ duyargaları…) önemli verilere erişmekte, bu verileri işleyip depolamakta veya başka birimlerle haberleşme kanallarıyla iletmektedir. Bu durum dolayısıyla, bu araçların tasarımlarında güvenlik unsuru bir

yer teşkil etmektedir. Bu uygulamaların temelini oluşturan gömülü sistemler de bu yüzden güvelik talepleriyle çokça karşı karşıya kalmakta ve bu ihtiyaçlara cevap verebilme durumunda olması gerekmektedir. Bu ihtiyaçların karşılanması için kullanılan yöntemlerden biri, bu bitirme çalışmasında da üzerinde çalışılan, kriptografi uygulamalarına özgü fonksiyonlar gerçekleyebilen ve güvenlik unsuru ön plan da tutulan bir ASIP işlemcinin gerçekleştirilmesidir.

Bu çalışmada, en çok kabul gören ve kullanılan blok şifreleme algoritmalarından gelişmiş şifreleme standardı (Advanced Encryiption Standard, AES) algoritmasını, eklenen özel donanım blokları ile daha verimli ve hızlı bir şekilde gerçekleyebilecek bir ASIP işlemcinin tasarlanması amaçlanmıştır. ASIP işlemcilerin tasarımında kullanılan en yaygın prosedür genel amaçlı bir işlemcinin komut setinin genişletilmesi (Instruction Set Extension) ve oluşan boş yerlere eklenen donanımları çalıştırabilecek komutların uygun bir şekilde yerleştirilmesidir. Bu tip bir yaklaşım, standart bir işlemciye herhangi bir uygulamaya özel fonksiyonları daha yüksek performansta, işlemcinin genel yapısının korunarak, eklememizi sağlayan basit ve etkili bir metottur.

Tezin 2. bölümünde mikroişlemciler hakkında genel bilgilere yer verilmiştir. Mikroişlemciyi oluşturan temel birimler, genel bazı kavramlar ve farklı işlemci mimarileri 2. kısımda irdelenmiştir. 3. kısımda AES algoritması açıklanmış, şifreleme, anahtar üretilmesi ve şifre çözme işlemlerinde yapılan işlemler ele alınmıştır. 4. kısımda işlemciyi oluşturan ve komutları işleyecek olan donanım birimlerinin tasarımı ve gerçekleştirilen modüllerin işlevleri açıklanmıştır Bir sonraki kısımda bu birimlerin bir işlemci mimarisi oluşturacak şekilde birbirine nasıl bağlandığı ve senkron bir şekilde çalışmasının nasıl sağlandığı ele alınmıştır.

2. MİKROİŞLEMCİ YAPILARI VE TEMEL BLOKLARI

2.1 Giriş

Mikroişlemci ve ya kısaca işlemci, makine kodu şeklinde kodlanmış program parçalarını çözümleyip gerekli işlemleri yürüterek, merkezi işlemci biriminin fonksiyonlarını gerçekleştiren programlanabilir bir sayısal tümdevre yapısıdır. Intel firmasının, başta yüksek performanslı hesap makinelerinde kullanmayı öngörerek tasarladığı 4 bit'lik 4004 işlemcisi, 1971 yılında piyasaya sürülmüş ve bir çok otorite tarafından ticari değeri olan ilk tümleşik işlemci devresi olarak gösterilmiştir. 4004 işlemcisinin yanında, Amerikan donanmasındaki F-14 savaş uçaklarında uçuş kontrolünün sağlanması amacıyla Garrett AiResearch firmasının geliştirdiği CADC (Central Air Data Computer) ve Texas Instruments firmasının önceden programlanmış belirli işlevleri gerçekleştirebilecek şekilde tasarladığı TMS 1000 de benzer dönemde ortaya çıkan ve işlemci teknolojisinin temellerini atan diğer mikroişlemci yapılarıdır.

1970'lerde temelleri atılan mikroişlemcilerin sağladığı esneklik ve tasarım kolaylığı, bu tümleşik devre yapılarının popülerliğinin hızla yükselmesine olanak sağlamıştır. Mikroişlemci teknolojisinin, performans, hız, tümdevre alanı gibi özelliklerinin yanında, sağladığı işlevler de göz önünde bulundurulduğunda elektronik endüstrisinde en çok gelişme gösteren alan olarak nitelendirmek yanlış olmayacaktır. Tümdevre teknolojisi ile paralel gelişme gösteren mikroişlemci yapılarının gelişimin Moore yasasını doğrular şekilde olduğu göze çarpmaktadır. Bilindiği üzere Moore yasası, belirli bir silikon alan üzerine sığdırılabilecek tranzistör sayısının en düşük bileşen maliyetine göre her 2 yılda bir iki katına çıkacağını öngörmektedir [3].

Mikroişlemciler günümüzde elektronik sistem altyapısı olan çoğu uygulamada sistemin durumunu denetleyerek, kontrol etme işlevlerini yazılım ile gerçekleştirebilmekte olduklarından birçok uygulamada sistemin en kritik yapısını oluşturmaktadır. Mikroişlemci temelli sistemler günümüzde önemini giderek artırmaktadır. Bu sistemlerin işaret işleme, kontrol sistemleri, haberleşme,

mikrobilgisayarlar, robotik gibi önemli alanları da içinde bulunduran geniş bir yelpazede önemli uygulamaları bulunmaktadır.

2.2 İşlemci Mimarileri

Mikroişlemciler komut sayıları, adresleme kipleri ve bellek organizasyonlarına göre sınıflandırılabilmektedirler.

İşlemcilerin gelişme sürecinin başlarında izlenen yol, tümdevre teknolojisinde yaşanan ilerleme ve sayısal donanımdaki ucuzlama sayesinde işlemcilerin karmaşıklığını artırmak yönünde oldu. Bu süreçte tasarımcılar, işlemcilerin buyruk kümelerine birçok farklı fonksiyonu gerçeklemek üzere ve karmaşık adresleme kiplerini de destekleyebilen çok sayıda buyruk eklediler. Burada amaçlanan, işlemciler üzerinde geliştirilecek uygulamaların kullanıcılar için makine düzeyinden kullanıcı düzeyine yaklaştırılması ve daha yüksek seviyeli bir dil ile programlanabilmeye olanak sağlayan işlemciler tasarlanabilmesidir. Bu doğrultuda tasarlanan işlemciler CISC (Karmaşık Komut Setli Bilgisayar) olarak adlandırılmaktadır.

Ancak giderek karmaşık bir hal alan ve çok sayıda buyruğu gerçeklemek üzere donanım birimleri barındıran CISC işlemci yapısının performans açısından dezavantaj sağlayabileceği düşüncesi tasarımcıları başka mimariler üzerine düşünmeye sevk etti. 1980'lerin başında da bilgisayar tasarımcıları az sayıda buyruğu bulunan ve sık sık belleğe erişmeyen makinelerin mikroişlemci birimi içerisinde daha hızlı olduklarını tavsiye ettiler [4]. Bu doğrultuda da RISC (Azaltılmış Buyruk Kümeli Bilgisayar) mimarisi ortaya çıkmıştır.

2.2.1 RISC

RISC ifadesi "Reduced Instruction Set Computer", yani azaltılmış buyruk kümeli bilgisayar anlamına gelmektedir. Adından da anlaşılacağı üzere, RISC işlemciler CISC işlemcilere göre daha az sayıda komut içermektedir. Daha az sayıda ve daha basit buyruklar ile işlem süresi azaltılabilmektedir. RISC işlemcilerin temel ve ortak bazı özellikleri şunlardır [4]:

- Az sayıda buyruk ve adresleme kipi
- Sadece LOAD ve STORE buyrukları ile bellek erişimi gerçekleştirilmesi

- Sabit uzunluklu, kolay çözülebilen buyruk yapısı
- Mikroprogramlı denetim yerine donanımsal denetim gerçekleştirilmesi
- Buyrukların tek evrede icra edilmesi

RISC işlemcilerdeki önemli özelliklerden biri buyruk uzunluğunun sabit olması ve yapısının kolay çözülmeye imkân sağlayabilmesidir. Örnek bir RISC işlemci buyruk yapısı Şekil 2.1'de'verilmiştir.

Şekil 2.1 16 Bit'lik Örnek Bir Komut Yapısı [5]

Şekil 2.1'deki komut yapısı incelenecek olursa, buyruğun belli bölgelerinin farklı özel anlamlar taşıdığı açıkça görülmektedir Örneğin, ilk 3 bit işlem kodunu (opcode) belirtmektedir ve mikroişlemci bu 3 bite bakarak hangi işlemi yürütmesi gerektiği anlar. Sonraki iki 3 bit'lik blok ile de, üzerinde işlem yürütülecek olan saklayıcıların adresi belirlenmektedir. Son 7 bit'lik kısımda ise işleme doğrudan sokulmak istenen sayılar yer almaktadır. Komut çözme işlemi (Instruction Decoding), bu buyruktan yola çıkarak işlemcinin yapacağı işi ve hangi saklayıcılara kullanmak üzere erişmesi gerektiğini belirlemesidir.

Günümüzde RISC ailesine ait bazı önemli işlemciler ARM, AMD 29k, SPARC ve PowerPC'dir.

2.2.2 CISC

CISC işlemcilerde kullanılan buyruklar, yüksek seviyeli dillerin sağladıkları özelliklere benzer işlevleri gerçekleştirmek üzere tasarlanmışlardır. CISC işlemcilerin bazı belirgin özellikleri [4];

- 100-250 arasında değişen çok sayıda buyruk
- 5-20 arasında değişen çok sayıda adresleme kipi
- Değişken uzunluklu buyruk yapıları
- Bellekten veri kullanabilen buyruklar

Yukarıda verilen özelliklerde belirtildiği üzere, CISC işlemcilerde komut uzunlukları sabit olmayabilmektedir. Bu tip işlemcilerde komut çözme işlemi aşama aşama gerçekleştirilir. Komutun bütünüyle çözülmesinin kaç aşamada tamamlanacağı farklı

buyruklar için farklı olabilmektedir. Bellekten okunacak verinin ve ya buyruğun boyutu, her aşamada komuta bakılarak tekrar değerlendirilebilir.

CISC işlemcilerin karmaşık işlevleri gerçekleştirilebilen komutlarına örnek olması açıcından MC68000 işlemcisinin MOVEM buyruğu gösterilebilir. Örneğin;

MOVEM.W (A7)+, D0-D5/D7/A0-A6

komutu ile A7 saklayıcısındaki (yığın göstergesi) adresten başlanarak artan adreslerden okunan değerler, sağ tarafında verilen saklayıcılara sırayla yazılmaktadır. Bu işlem çok sayıda bellek okuma ve yazma işlemini tek komut ile gerçekleştirmemizi sağlamaktadır.

Motorola'nın 68000 ailesi ve Intel'in 8086 işlemcisi CISC yapısının kullanıldığı işlemcilere örnek teşkil etmektedir.

Bazı örnek RISC ve CISC işlemcilerin genel özellikleri Tablo 2.1'de karşılaştırılmıştır.

Tablo 2.1 Bazı CISC ve RISC İşlemcilerin Özelliklerinin Karşılaştırılması [6]

	CISC		**RISC**
Karakteristik	**VAX 11/780**	**Intel 486**	**MIPS R4000**
Komut Sayısı	303	235	94
Adresleme Kipi Sayısı	22	11	1
Buyruk Boyutu (Bayt)	2-57	1-12	4
Saklayıcı Sayısı	16	8	32

2.3 Donanım Mimarileri

Bir önceki kısımda işlemcileri, buyruklarının sağladığı işlev ve sayılarına göre RISC ve CISC olarak ikiye ayırmıştık. İşlemcileri birbirinden farklı kılan diğer bir temel özellik de işlemci donanımlarının organizasyonu ile ilgilidir.

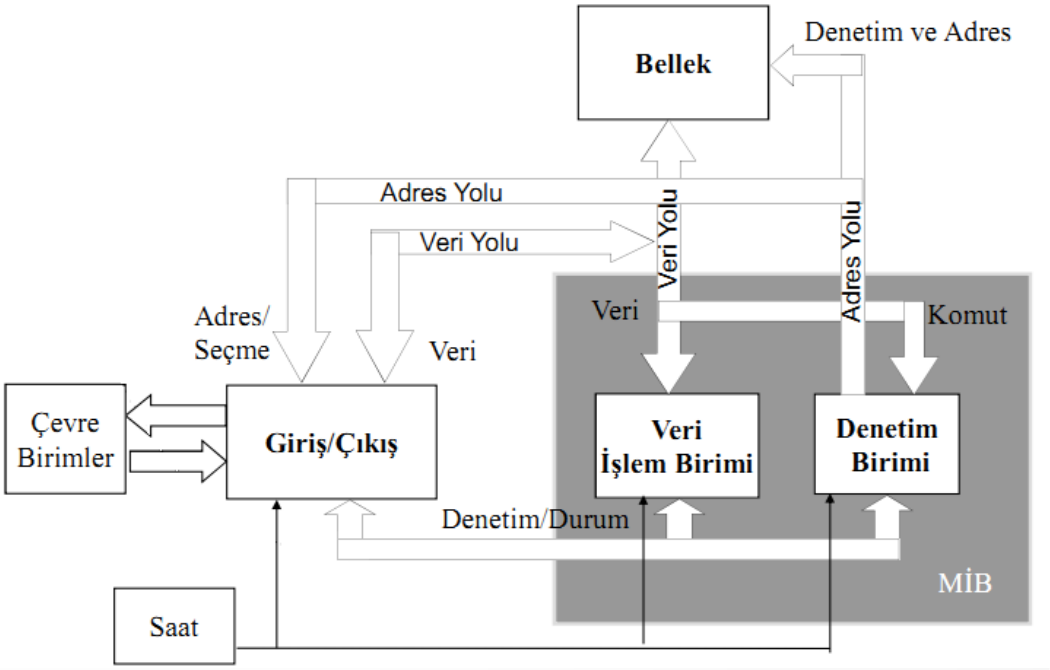
İşlemcinin kullanıcının istediği şekilde çalışması ve ya istenilen değerler üzerinde işlem yapması için kullandığı iki önemli bilgi komutlar ve verilerdir. Mikroişlemci birimi bu iki bilgiye erişerek çalışmasını sürdürmek durumundadır. Bu yüzden işlemci biriminin bu bilgilere ne şekilde ve ne hızda ulaşabildiği, ayrıca belirli bir

anda bu bilgilerden ne boyutta okuyabildiği mikroişlemcinin performans ve işlevselliğini belirleyen önemli bir etkendir.

Bazı işlemcilerde komut ve veriler aynı bellekte tutulurken bazı işlemcilerde farklı belleklerde tutulmaktadır.

2.3.1 Von Neumann Mimarisi

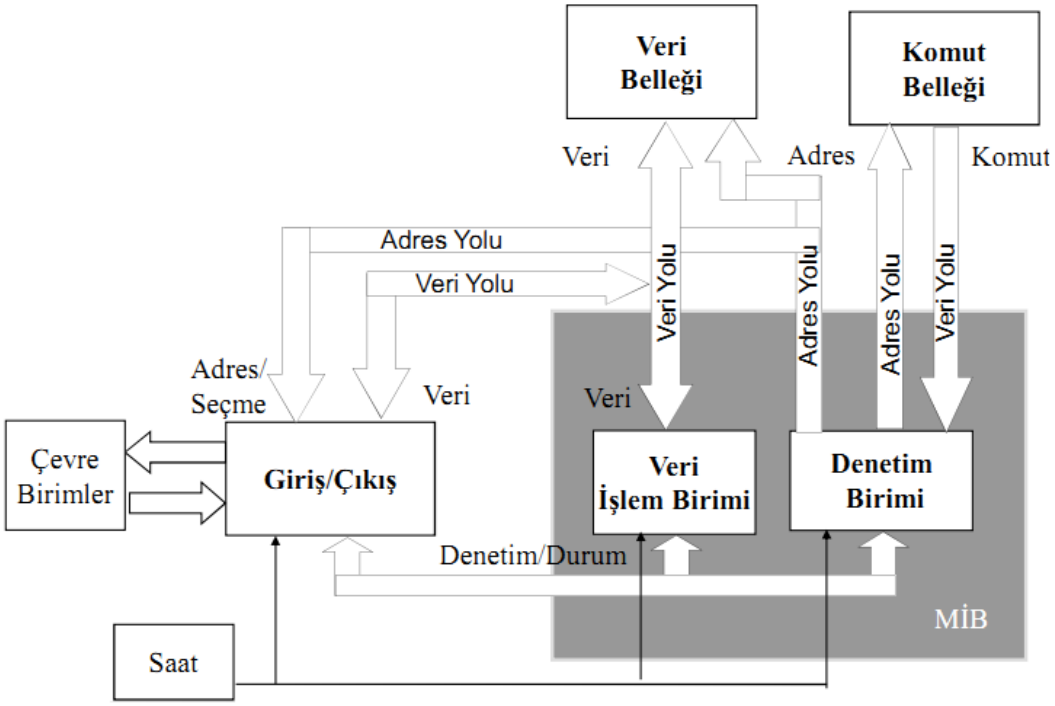
Komut ve verilerin aynı bellek yapısında saklandığı donanım mimarisidir. Komut ve verilere erişim aynı veri yolları üzerinden olduğundan, belleğe bağlı bir adres yolu ve bir veri yolu ile yazma ve okuma işlemleri yürütebilmektedir. Komut ve veriler bellekten farklı saat darbelerinde okunur. Von Neumann mimarisindeki örnek bir bilgisayar yapısının blok diyagramı Şekil 2.2'de verilmiştir.



Şekil 2.2 Von Neumann Mimarisi Blok Diyagramı [7]

2.3.2 Harvard Mimarisi

Komut ve verilerin farklı bellek birimlerinde saklandığı donanım mimarisidir. Dolayısıyla Harvard mimarisinde, Von Neumann mimarisindekinin aksine komut ve verilere erişim sağlanması için ayrı veri ve adres yolları mevcuttur. Bu mimarinin sağladığı avantaj, aynı anda veri ve buyruklara erişilebilmesidir. Harvard mimarisinin örnek bir blok yapısı Şekil 2.3'de verilmiştir.



Şekil 2.3 Harvard Mimarisi Blok Diyagramı [7]

2.4 Temel İşlemci Birimleri

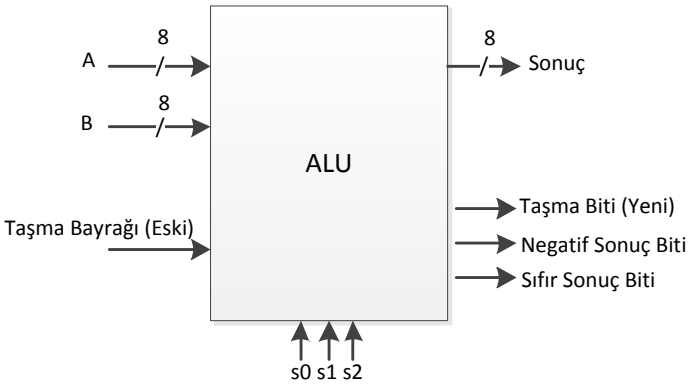
İşlemcilerin içerisindeki donanım blokları, buyrukları uygun sırayla bellekten okur ve gerekli verilere erişip, bu veriler üzerinde buyrukta tanımlanan işleri yürütür. Bu kısımda işlemcilerin senkron bir şekilde çalışmasını sürdürmesini sağlayan ve denetleyen bazı temel birimlerden söz edilecektir. Bu kısımda anlatılan donanım birimleri, veriler üzerinde işlem yapan aritmetik mantık birimi (Arithmetic Logic Unit-ALU), donanımların çalışmasını düzenleyen kontrol birimi, blokları birbirine bağlayan veri yolu yapıları ve iç saklayıcıların bulunduğu saklayıcı dosyasıdır.

2.4.1 Aritmetik Mantık Kaydırma Birimi (ALU)

Aritmetik mantık birimi (ALU) buyrukların yerine getirilmesi için başlatılan mikroişlemlerin donanımsal olarak gerçeklendiği birimdir. ALU, içerisinde temel aritmetik ve mantık işlemlerini gerçekleştirebilecek sayısal donanımları barındırır. Tasarlanan bazı ALU yapılarında kaydırma işlemlerini yürüten birim ALU dışında da tasarlanabilmektedir.

ALU bir kombinezonsal devredir, dolayısıyla saat darbesi gerektirmez ve işlem süresi kapı ve yolların oluşturduğu gecikmeler ile belirlenir. ALU girişine işlemlerde kullanılacak saklayıcılar ile yapılacak işlemleri ve hangi çıkışın sonuca aktarılacağını

belirleyecek seçim girişleri uygulanır. ALU bu seçim girişlerine göre belirlenen işlemi gerçekleştirerek çıkışa yazar.



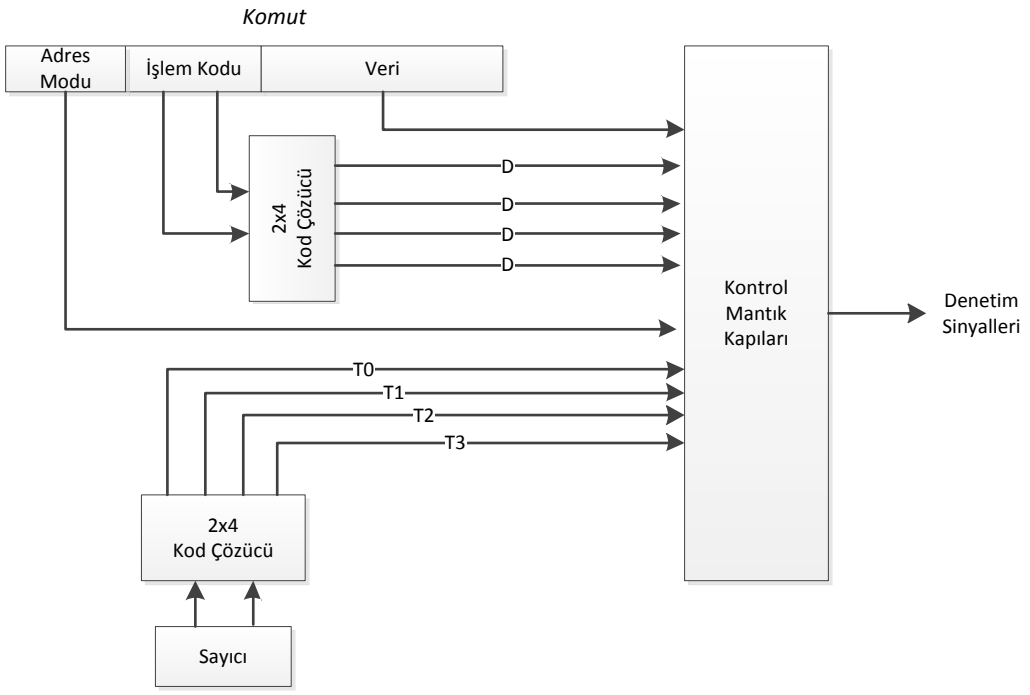
Şekil 2.4 Örnek Bir ALU Bloğu

Şekil 2.4'de örnek bir ALU bloğunun giriş ve çıkışları gösterilmiştir. Görüldüğü üzere ALU işlemleri ile oluşan sonuca göre durum bayraklarının güncellenmesi sağlanmaktadır.

2.4.2 Kontrol Birimi

Kontrol birimi ve ya diğer adıyla denetim birimi, işlemci içerisindeki donanım birimlerinin çalışmasının düzenlenmesinden ve senkronizasyonundan sorumludur. Bir işlemci mimarisi içerisinde saklayıcıların içeriği saklayıcıların güncellenmesi, ortak veri yoluna yazma işlemleri denetim sinyalleri tarafından yetkilendirildiğinde gerçekleşir. Denetim sinyalleri kontrol birimi tarafından üretilir.

Kontrol birimi, ürettiği yetkilendirme sinyallerinin yanında yürütülecek olan işlemlerin zamanlamasını da uygun bir şekilde düzenlemek durumundadır. İşlemcinin bellekten bir buyruğu alıp çözdükten sonra gerekli işlemleri yerine getirmesi birden fazla saat darbesi gerektirebilir. Bu yüzden kontrol birimi hangi saat darbesinde hangi birimlerin denetim sinyalleri ile aktif edileceğini bilmelidir. Bu tür durumlar için denetim biriminde sayıcı devreler kullanılır. Şekil 2.6'da örnek bir kontrol birimi blok diyagramı verilmiştir. İki bitlik sayıcının oluşturduğu T sinyalleri, kombinezonsal kontrol mantık devresine zamanlama bilgisi vermektedir.



Şekil 2.6 Kontrol Birimi

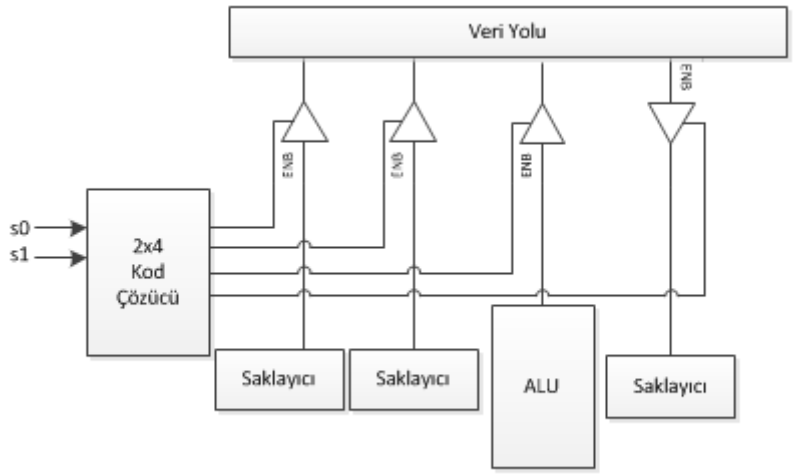
2.4.3 Saklayıcı Dosyası

Bir bilgisayar sisteminde genel amaçlı iç saklayıcılar, ALU işlemlerinde kullanılacak akümülatörler, adres saklayıcıları ve ya sistemin durumu hakkında bilgi içeren durum saklayıcılar gibi birçok saklayıcı bulunmaktadır. İç saklayıcılar bir saklayıcı dosyası yapısı içerisinde tutulur ve bellek erişimi yapılırken uygulanan prosedürlere benzer şekilde bu saklayıcılara yazma/okuma işlemleri yürütülür. Bazı sistemlerde özel amaçlı saklayıcılar saklayıcı dosyası dışında da bulunabilmektedir.

2.4.4 Veri Yolu Yapıları

Bir bilgisayar sisteminde birbiri ile haberleşme durumunda olan birçok saklayıcı ve donanım birimleri bulunabilmektedir. Çok sayıda birimin kendi aralarında veri alış verişi yapabileceği etkin ve denetimi kolay bir yöntem ortak veri yolu sistemidir. Ortak veri yolu sisteminde, saklayıcılar aynı veri hattı üzerinde ver aktarımını gerçekleştirirler ve belirli bir anda hangi saklayıcıların veri yoluna erişebileceği denetim sinyalleri ile belirlenir.

Ortak bir veri yolu yapısı, veri seçiciler (MUX) ve ya 3 durumlu tamponlar kullanılarak gerçekleştirilebilir. 3 durumlu tamponlar (buffer) kullanılarak gerçekleştirilebilecek bir veri yolu yapısı Şekil 2.4'de verilmiştir.

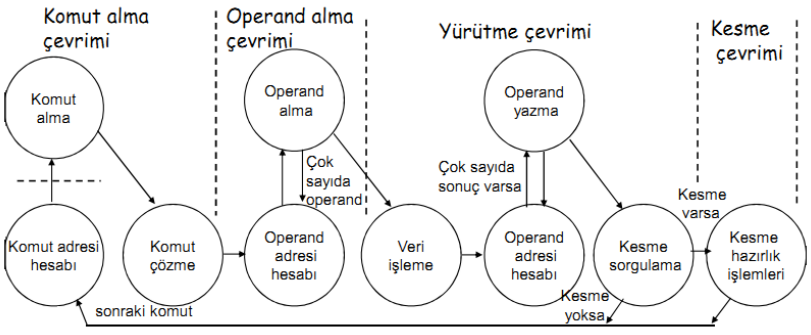


Şekil 2.5 Üç Durumlu Tamponlar ile Tasarlanan Örnek Bir Veri Yolu

2.5 Komut Süreci

İşlemci program belleğinde ve ya ortak belleğinde bulunan komutları uygun sıra ile alır ve yapılacak işlemleri belirleyip buyruğu icra eder. Bu da belirli bir sürecin sonunda mümkün olmaktadır. Bir işlemci sisteminde genel olarak bir buyruk süreci üç evreden oluşur ve bu süreç kontrol birimi tarafından denetlenir.

Bu süreçteki evrelerin ilki komutun bellekten alınıp çözülmesidir (Fetch and Decode). Bu aşamada program sayıcıda yüklü olan adresteki buyruk bellekten okunur ve kodu çözülür. Bir sonraki evrede buyruğun dolaylı adresleme kipinde verilmiş olması durumu için etkin adres hesaplanır ve etkin adresteki buyruk okunur. Son evrede ise, kontrol birimi buyruğu icra etmek üzere gerekli denetim sinyallerini üretir. Bir işlemcide, buyruğun alınıp icra edildikten sonra bir sonraki buyruğa geçene kadar geçirdiği tüm çevrimler Şekil 2.6'da gösterilmiştir.

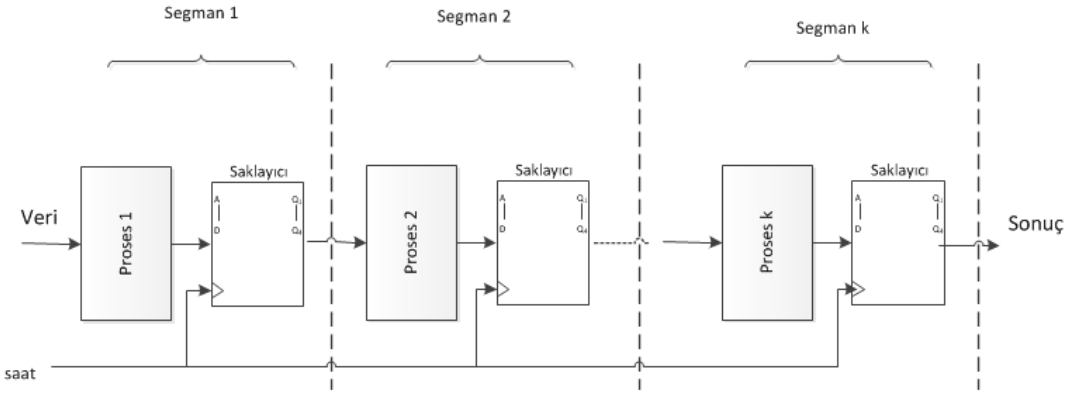


Şekil 2.6 İşlemci Çalışma Döngüsü [7]

2.6 İş Hattı (Pipeline)

İş hattı, büyük işlerin her biri paralel olarak çalışabilen küçük alt parçalara bölünmesi ile oluşturulan bir veri işleme yöntemidir. İş hattına örnek olarak bir otomobil fabrikasının işleyişini gösterebiliriz. Bir otomobilin bütünüyle baştan sona üretilmesi uzun bir süreç olarak gözükmesine rağmen iş hattı ile bu süreç hızlandırılmaktadır. Çünkü otomobilin her bir parçasının montajı ayrı robotlar tarafından paralel olarak yapılmaktadır. Örneğin bir otomobilin kapılarının montajı yapılırken bir yandan da başka bir otomobilin lastikleri monte edilebilmektedir. İşlemcilerde de bu prosedür uygulanarak işlemcinin hızının artırılması amaçlanmaktadır.

Genel bir iş hattı yapısı Şekil 2.7'de verilmiştir. İş hattının her bir bölümüne segman adı verilmektedir ve her segmanda alt parçalara bölünmüş olan işlemler yürütülmektedir. Her segmanın sonunda segman saklayıcıları bulunmaktadır ve alt işlemlerin sonuçları bu saklayıcılara yazılmaktadır.



Şekil 2.7 İş hattı Yapısı

Bir iş hattında, belirli bir süre zarfında hangi segmanda hangi işlerin yürütüldüğünü göstermek için uzay-zaman diyagramları kullanılır. 4 segmanlı bir iş hattının uzay-zaman diyagramı Şekil 2.8'de gösterilmiştir.

Saat Darbesi

Segman	1	2	3	4	5	6	7
1	T1	T2	T3	T4	T5	T6	
2		T1	T2	T3	T4	T5	T6
3			T1	T2	T3	T4	T5
4				T1	T2	T3	T4

Şekil 2.8 4 Segmanlı Bir İş Hattının Uzay-Zaman Diyagramı [7]

Verilen uzay-zaman diyagramında, yapılacak işler T ile gösterilmiştir. Görüldüğü üzere T1 işi ikinci saat darbesinde ikinci segmana geçtiğinde, birinci segmanda T2 işinin yürütülmesine başlanmaktadır. Her bir iş 4 saat darbesinde tamamlanmaktadır fakat iş hattı sayesinde dördüncü saat darbesinden sonra her saat darbesinde bir iş tamamlanabilmektedir.

Ancak iş hattının bu paralel veri işleyen yapısı bazı sıkıntıları da beraberinde getirmektedir. Bir iş hattı yapısında dallanma buyruğu icra edilirken, iş hattına bu buyruktan sonra gelen komutlar alınmış olacaktır. Dallanma gerçekleşmesi beklendiğinden iş hattına giren bu buyrukların gerçekleştirilmesi istenmez ve iş hattının boşaltılması gerekir. Bu duruma dallanma cezası (branch penalty) adı verilir. Bir diğer problem ise aynı veri üzerinde işlem yapacak olan buyrukların aynı anda iş hattı üzerinde bulunmasıdır. Farklı segmanlarda bulunan buyruklar veriyi değiştirmesi hatalı işlem yapılmasına yol açabilmektedir. Bu probleme de veri bağımlılığı (data dependency) adı verilmektedir.

3. GELİŞMİŞ ŞİFRELEME STANDARDI

3.1 Giriş

Belçikalı iki kriptografi uzmanı Joan Daemen ve Vincent Rijmen tarafından geliştirilen Rijndael algoritması 2000 yılında Amerika Devlet Standartlar Enstitüsü (NIST) tarafından Gelişmiş Şifreleme Standardı (AES) olarak isimlendirilerek elektronik veri güvenliğinin sağlanması amacıyla veri şifreleme standardı olarak ortaya konmuştur.

1970'lerin başında IBM tarafından geliştirilen veri şifreleme standardı (Data Encryption Standard-DES) algoritmasının teorik olarak zayıflığının gösterilmesi ve 54 bitlik anahtarın gelişen bilgisayar ve tümdevre teknolojileri sayesinde kırılabilir duruma gelmesi, 90'lı yıllar başladığında bu algoritmanın güvenliğinin sorgulanır duruma gelmesine sebep olmuştur. Bu sebepten ötürü NIST 1997 yılında bir yarışma başlatmış ve 4 yıl süren değerlendirmeler sonucunda Rijndael algoritmasını kazanan olarak açıklamıştır.

3.2 Rijndael Algoritması

Rijndael algoritması, anahtar uzunluğuna göre farklı sayıda döngü içeren ve her döngüde belirli işlemlerin, yenilenen anahtar da kullanılarak şifrelenecek veriye uygulanması ile gerçekleştirilen bir blok şifreleme algoritmasıdır. AES, kriptografik algoritmalar içinde simetrik anahtar algoritmaları grubundadır. Burada simetrik anahtar kavramı, bir metni şifrelemek ve şifrelenmiş metni çözmek için aynı anahtarın kullanılacağını betimler.

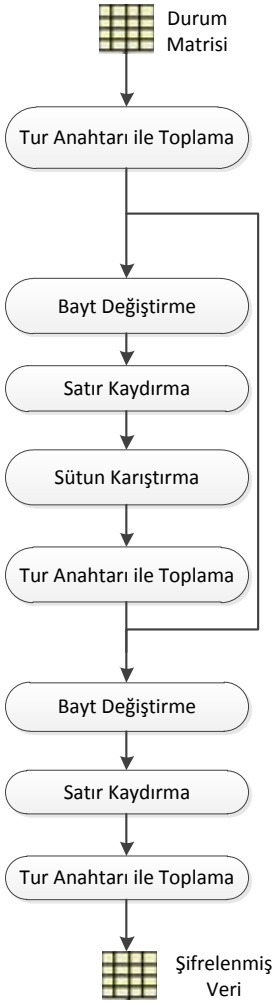
Rijndael algoritmasında, şifrelenecek ve ya şifresi çözülecek veriler 128 bit uzunluğundadır. Bu 128 bitlik veri, her bir elemanı 1 bayt yani 8 bit'e karşılık gelen, 4 satır ve 4 sütundan (4x4) oluşan bir matris ile ifade edilir. Bu matrise "durum" adı verilmektedir. Belirtildiği üzere AES algoritmasında tur sayısı kullanılan anahtarın uzunluğu ile belirlenmektedir. Tur sayısı arttıkça veri güvenliği artmakta fakat şifreyi

saklamak için gerekli bellek alanı ve yapılması gereken işlem sayısı artmaktadır. Şifre uzunluğuna göre tur sayısının değişimi Tablo 3.1'de gösterilmiştir.

Tablo 3.1 Tur Sayısı ile Anahtar Uzunluğu Arasındaki İlişki

	Veri	**Anahtar Uzunluğu**	**Tur Sayısı**
AES-128	128 bit	128 bit	10
AES-192	128 bit	192 bit	12
AES-256	128 bit	256 bit	14

AES algoritmasının blok yapısı Şekil 3.1'de verilmiştir. Burada tur oluşumu geri besleme ifade edilmiştir ve sayısı Tablo 3.1'de gösterilen değerleri alabilir. Görüldüğü üzere son turda sütun karıştırma işlemi gerçekleştirilmemektedir.



Şekil 3.1 AES Algoritmasının Blok Yapısı

3.3 Şifreleme İşlemi

AES algoritması 128 bit'lik veri bloklarının şifrelenmesini sağlamaktadır. Burada blok olarak tanımlanan, bir önceki kısımda "durum" olarak tanımladığımız 4x4 büyüklüğündeki matristir. AES algoritması işlemlerini bu matris yapısı üzerinde gerçekleştirdiğinden, şifrelenecek veri uygun bir şekilde durum matrisi halinde ifade edilmelidir.

Bir örnek ile verinin durum matrisi şeklinde ifade edilmesini göstertebiliriz. Şifrelenecek verinin onaltılık tabanda 32 43 F6 A8 88 5A 30 8D 31 31 98 A2 E0 37 07 34 olması durumda, oluşacak durum matrisi Şekil 3.2'de gösterildiği gibi olacaktır.

32	88	31	E0
43	5A	31	37
F6	30	98	07
A8	8D	A2	34

Şekil 3.2 Örnek Durum Matrisi

Oluşturulan bu durum matrisi her bir turda bayt değiştirme, satır kaydırma, sütun karıştırma ve tur anahtarı ile toplama işlemlerinden geçirilerek şifrelenmiş veri elde edilmektedir.

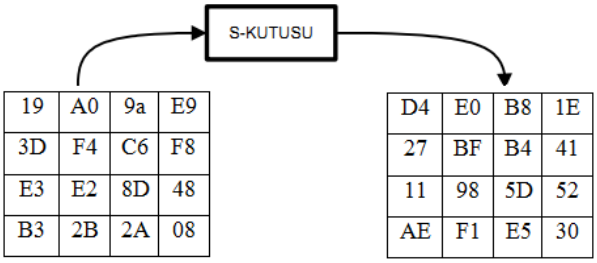
3.3.1 Bayt Değiştirme

Bayt değiştirme işlemi, durum matrisinin her bir baytı üzerinde bağımsız olarak işlem yapabilen doğrusal olmayan bir işlemdir. Bu işlemde durum matrisindeki her bir kutucuk, (bir bayt) S-kutusu adı verilen özel bir matris üzerinde, kendi değerine karşı gelen kutucuktaki değer ile değiştirilir. S-kutusunun yapısı Şekil 3.3'de gösterilmiştir.

Bu işlemin anlaşılması açısından örnek bir durum matrisine, bayt değiştirme işleminin uygulanması ile elde edilen yeni durum matrisi Şekil 3.4'de gösterilmiştir.

		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

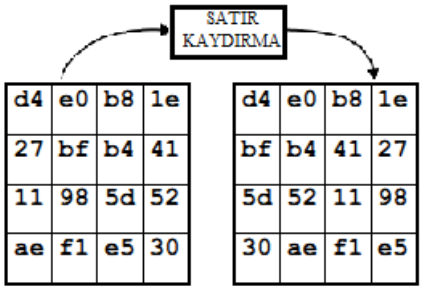
Şekil 3.3 S-kutusu Değerleri (Onaltılık Tabanda) [8]



Şekil 3.4 Örnek Bir Bayt Değiştirme İşlemi [8]

3.3.2 Satır Kaydırma

Satır kaydırma işleminde ilk satır aynı kalırken, ikinci satır 1, üçüncü satır 2, dördüncü satır ise 3 bayt sola ötelenir. Taşan baytlar ise satır sonuna eklenerek işlem sürdürülür. Şekil 3.4'deki bayt değiştirme işlemi sonucunda elde edilen matrise satır kaydırma işlemi uygulanması ile elde edilecek yeni durum matrisi Şekil 3.5'de verilmiştir.



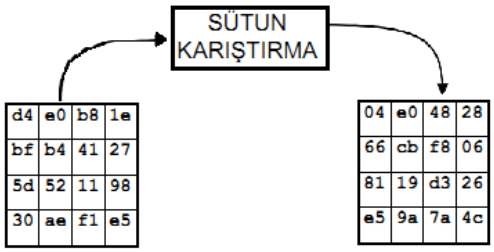
Şekil 3.5 Örnek Bir Satır Kaydırma İşlemi

3.3.3 Sütun Karıştırma

AES algoritmasının bu işleminde, durum matrisinin sütunları üzerinde işlem yapılmaktadır. Sütunlar Denklem 3.1'de verilen matris çarpımı işlemine tabi tutulur ve matrisin sütunları hesaplanan yeni sütun değerleri ile değiştirilir.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad (3.1)\ [8]$$

Örnek bir sütun karıştırma işlemi sonucunda elde edilen yeni durum matrisi Şekil 3.6'da verilmiştir.



Şekil 3.6 Örnek Bir Sütun Karıştırma İşlemi

3.3.4 Tur Anahtarı ile Toplama

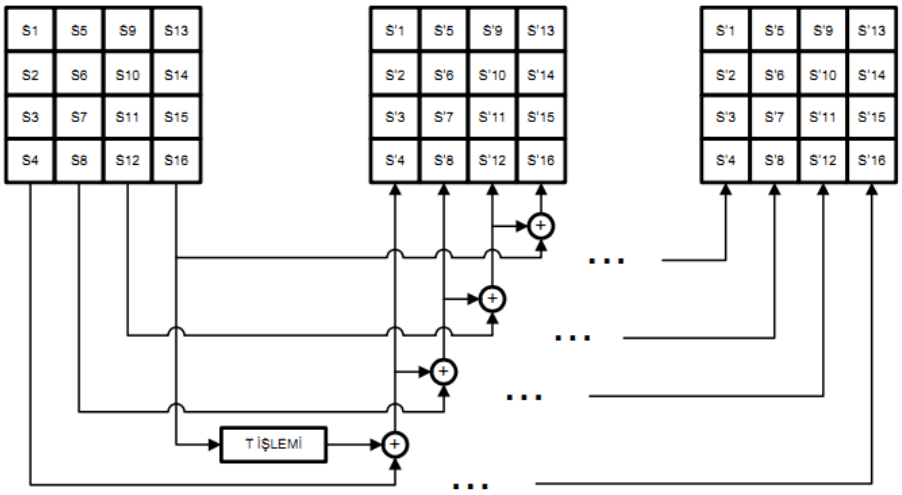
AES algoritmasında her bir tur için farklı bir anahtar kullanılarak şifreleme işlemi gerçekleştirilmektedir. Her bir tur için bu anahtarın nasıl hesaplanacağı 3.3.5 kısmında açıklanmıştır.

Tur anahtarı ile toplama işlemi, durum matrisi üzerinde, üretilen anahtar matrisi kullanılarak gerçekleştirilen toplama işlemine karşılık gelmektedir. Bu işlem sonlu alanlarda yapılan toplama işlemidir ve bit mertebesinde özel ve ya işlemine karşılık düşer.

3.4 Anahtar Üretilmesi

AES algoritmasında şifreleme işlemi yürütülürken, her tur için farklı bir anahtarın şifreleme işlemi ile birlikte (on the fly key expansion) ve ya daha önceden (pre-computed key schedule) hesaplanarak elde edilmesi gerekmektedir. Dolayısıyla

anahtar üretme işlemi de belirli sayıda tur içermektedir ve bütün anahtarlar bir önceki turda hesaplanan anahtarlar kullanılarak elde edilir.



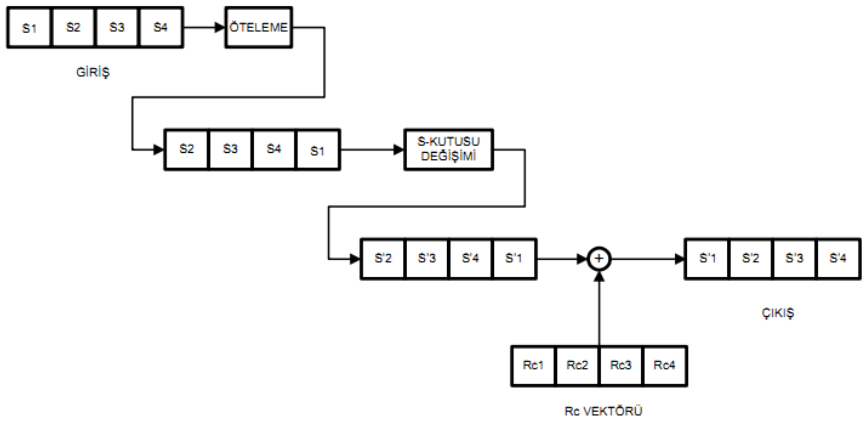
Şekil 3.7 Anahtar Üretme İşleminin Genel Yapısı [9]

Şekil 3.7'de görüldüğü üzere, anahtar üretimi işleminde her bir sütun hesaplanırken, o sütundan bir önceki ve dört önceki sütunlar özel veya işlemine tabi tutulmaktadır. Ancak Şekil 7'de de "T işlemi" olarak gösterilen blok, her bir matrisin ilk sütunu hesaplanırken uygulanmaktadır. T işlemi, öteleme, S kutusundan geçirme ve Tablo 3.2'de verilen Rc(x) vektörü ile toplama işleminden oluşan bir işlemler zinciri içermektedir.

Tablo 3.2 Rc(x) Vektörleri [9]

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

T işleminin genel blok yapısı Şekil 3.8'de gösterilmiştir.



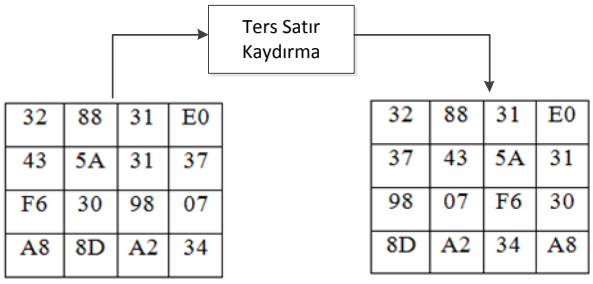
Şekil 3.8 T İşlemi Blok Yapısı [9]

3.5 Şifre Çözme İşlemi

Rijndael algoritmasında şifreli metni çözmek için uygulanan prosedürler şifreleme işlemi için yapılanlar ile oldukça benzerdir. Burada gerçekleştirilen işlemler şifrelemede yapılanların tersi olacak şekilde düzenlenmiştir.

3.5.1 Ters Satır kaydırma

Sifre çözme işleminde satır kaydırma işlemleri, şifrelemede yapılanın tersi olacak şekilde yani sağa doğru öteleme ile gerçekleştirilmektedir. Örnek bir ters satır kaydırma işlemi Şekil 3.9'da verilmiştir.



Şekil 3.9 Örnek Bir Ters Satır Kaydırma İşlemi

3.5.2 Ters Bayt Değiştirme

Şifreleme işleminde bayt değiştirme için bir S-kutusu kullanmıştık. Şifre çözme işleminde de gene aynı şekilde bir S-kutusu kullanılmaktadır. Fakat burada kullanılan S-kutusu ile elde edilen değerler, şifreleme işleminde elde edilen değerlerin tersidir ve ters S-kutusu olarak nitelendirilebilir. Yani bir veriyi önce S-kutusu sonrasında ise

ters S-kutusundan geçirdiğimizde aynı veriyi elde edebilmekteyiz. Dolayısıyla, şifre çözme işleminde kullanılan ters S-kutusunun değerleri farklıdır ve Şekil 3.10'da bu değerler gösterilmiştir.

		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 3.10 Ters S-kutusu Değerleri [8]

3.5.3 Ters Sütun Karıştırma

Bu işlemde şifrelemede olduğu gibi sütunlar üzerinde işlem yapılmaktadır fakat burada matris çarpımında kullanılan değerler farklıdır. Durum matrisinin sütunları $GF(2^8)$ alanında tanımlı polinomlar olarak ifade edilirler ve modülo $x^4 + 1$ işlemine tabi tutulduklarında elde edilecek $a^{-1}(x)$ polinomu, Denklem 3.2 ile verilmektedir.

$$a^{-1}(x) = \{0B\}x^3 + \{0D\}x^3 + \{09\}x + \{0E\} \quad (3.2)$$

$s'(x) = a^{-1}(x) \otimes s(x)$ ile ifade edildiğinde, bu işlem Denklem 3.3'de gösterildiği gibi matris çarpması halinde gösterilebilir.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad (3.3)\ [8]$$

3.5.4 Çözme İşleminde Tur Anahtarı ile Toplama

Şifreli metni çözme işleminin bu kısmında yapılan işlem, şifrelemede yapılan ile tamamen aynıdır. Hatırlanacağı üzere, şifrelemede tur anahtarı ile toplama işleminin bit düzeyinde özel veya işlemi ile yürütüldüğü açıklanmıştı. Dolayısıyla şifre çözme işleminde de aynı şekilde, durum matrisi tur anahtarındaki veriler ile özel veya işlemine tabi tutulmaktadır.

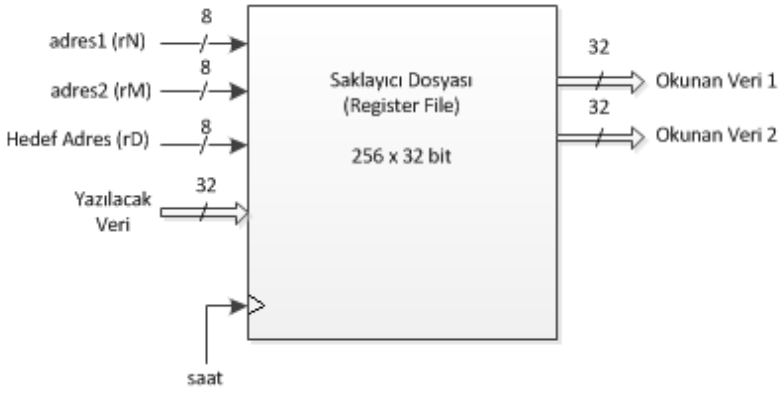
4. MİKROİŞLEMCİ BLOKLARININ VHDL İLE TASARLANMASI

Yüksek hızlı tümleşik devre donanımı tanımlama dili (Very high speed integrated circuit Hardware Description Language-VHDL), tümleşik devre ve ya FPGA'lar üzerinde gerçekleştirilecek olan sayısal donanımların tasarlanması ve denenmesi amacıyla kullanılan bir donanım tanımlama dilidir. VHDL ile kombinezonsal devre yapıları ve ya senkron ardışıl devre yapıları kolayca gerçekleştirilebilmektedir. VHDL ile tasarlanan donanımlar üzerindeki sinyallerin, ModelSim ve ya ISim gibi araçlar kullanılarak benzetimleri yapılabilmekte, bu şekilde tasarımın doğruluğu hakkında bilgi edinilebilmektedir.

Tezin bu bölümünde işlemci yapısını oluşturacak olan donanım bloklarının ayrı ayrı VHDL ile tasarlanması ele alınmıştır. Tasarlanan yapıların modüler yapısı gösterilmiş ve simülasyon sonuçları gösterilmiştir.

4.1 Saklayıcı Dosyasının Oluşturulması

Bu tez kapsamında tasarlanacak olan işlemci 32 bitlik bir RISC işlemcidir. Bu yüzden verileri tutacak olan iç saklayıcıları 32 bit uzunluğundadır. Ayrıca tasarımda akümülatör benzeri saklayıcılar kullanılmamıştır çünkü yapılan işlemler doğrudan saklayıcı dosyasındaki 256 saklayıcı üzerinde yürütülebilmektedir. Gerçekleştirilen saklayıcı dosyasının genel görünümü Şekil 4.1'de verilmiştir.

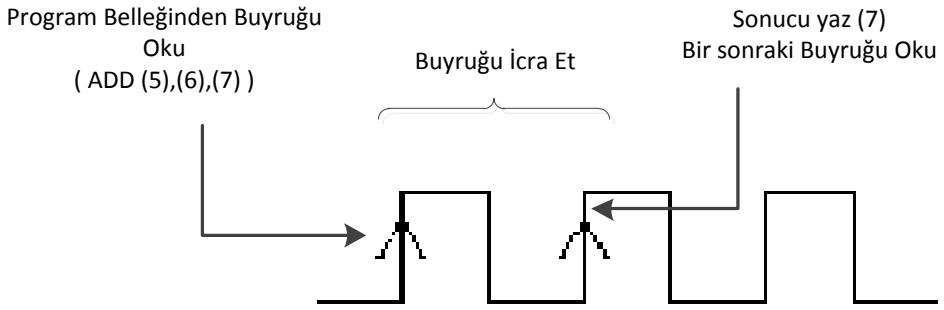


Şekil 4.1 Saklayıcı Dosyası

Oluşturulan saklayıcı dosyası yapısında yazma işlemi saat işaretine göre senkron bir şekilde yapılırken, okuma işlemi saat işaretinden bağımsız olarak asenkron bir şekilde gerçekleştirilmektedir. Hedef adres ve yazılacak veri girişleri ile, mikroişlemcide gerçekleştirilen işlemlerin sonucu saklayıcılara yazılmaktadır. Ayrıca adres1 ve adres2 girişlerindeki değerlere göre belirlenen saklayıcılar da asenkron olarak okunarak çıkışlara yazılmaktadır.

4.1.1 Saklayıcı Dosyasına Yazma Komutları

Saklayıcı dosyasından okuma işlemi asenkron olarak gerçekleştirilebildiğinden, okuma işlemi tanımlanan buyruklar içerisinde olaylı olarak yürütülebilmektedir. Örneğin, 5 ve 6 adresindeki saklayıcı değerlerinin toplanarak 7 adresine yazılması için bu işlemcide gerçekleştirilecek işlem sembolik olarak ADD (5),(6),(7) şeklinde gösterilmektedir. Burada buyruk alındıktan sonra 5 ve 6 adresleri asenkron olarak okunacaktır. Sonucun 7 adresine yazılması işlemi ise senkron bir şekilde yapılmaktadır ve buyruğun alındığı saat darbesini izleyen bir sonraki saat darbesinde işlem sonucu 7. saklayıcıya yazılır. Senkron bir şekilde gerçekleştirilen bu işlemin zaman diyagramı Şekil 4.2'de gösterilmiştir.



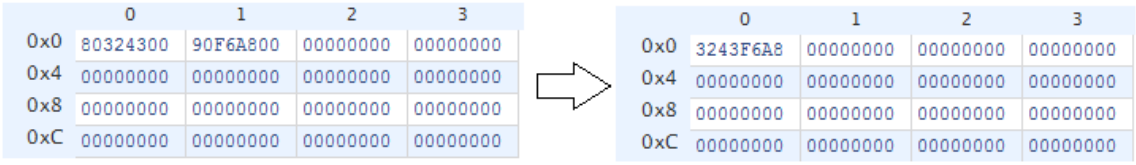
Şekil 4.2 Saat İşaretine Göre Buyruğun İcra Edilmesi

Saklayıcılara doğrudan bir değer yazılması için de gerekli komutlar işlemcinin komut setine eklenmiştir. 32 bitlik bir verinin komut içerisinde yer alması, komutun gereğinden fazla uzun olmasına sebep olacağından, saklayıcıların alt ve üst 16 bitlerine yazma işlemleri ayrı iki komut ile gerçekleştirilmiştir. Saklayıcı dosyasına yazma işlemlerini gerçekleştiren komutların yapısı Tablo 4.1'de verilmiştir.

Tablo 4.1 Saklayıcı Dosyasına Yazma Komutları

Makine Kodu (32 bit)	Sembolik İfade	İşlem
80kkkknn	MovH rN, <k>	rN = <k> (Üst 16 bit)
90kkkknn	MovL rN, <k>	rN = <k> (Alt 16 bit)

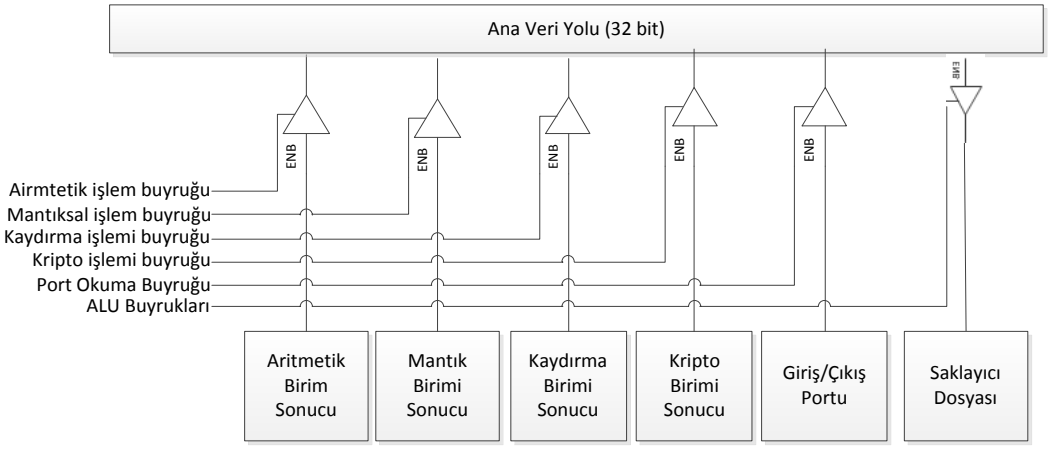
Tablo 4.1'de yazma komutlarının onaltılık tabandaki makine kodları verilmiştir. Örnek olarak MovH (00), 3243H (makine kodu: 80324300) ve MovL (00), F6A8H (makine kodu: 90F6A800) komutlarının program belleğine yazılarak işlemlerin yürütülmesi sonucunda saklayıcı dosyasında 0 adresindeki saklayıcıya onaltılık tabandaki 3243F6A8 sayısı yüklenmiştir. Program belleğinin ve benzetim sonucunda saklayıcı dosyasının görünümü Şekil 4.3'de gösterilmiştir.



Şekil 4.3 Verilen program Örneği İçin Program Belleği (solda) ve Saklayıcı Dosyasının (sağda) Görünümü

4.2 Veri Yolu Yapılarının Oluşturulması

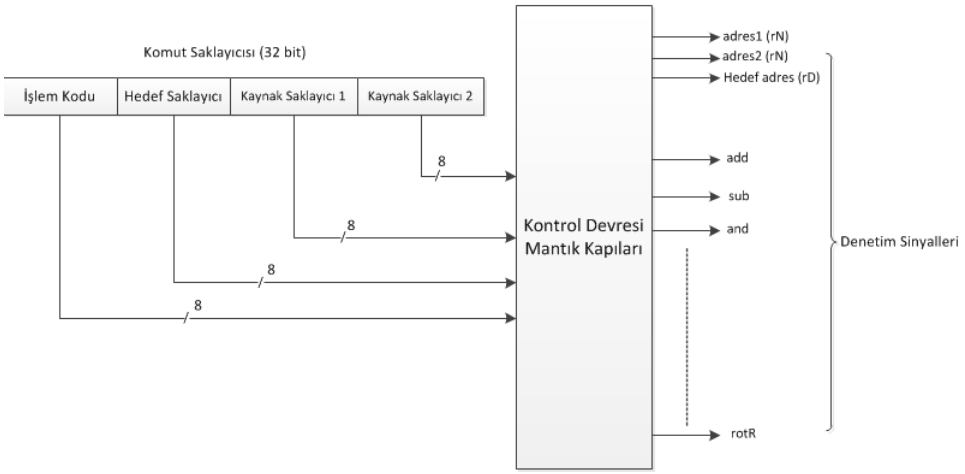
Tasarladığımız işlemcide 3 temel veri yolu bulunmaktadır; bir ana veri yolu ve 2 yardımcı veri yolu. Bütün veri yolları 32 bit kapasitesindedir. Yardımcı veri yolları saklayıcı dosyasından okunan verileri tutmaktadır ve busN ve busM olarak isimlendirilmişlerdir. Ana veri yoluna ise icra edilen buyruk sonucunda elde edilen veriler yazılmaktadır. Ana veri yolu aynı zamanda saklayıcı dosyasının girişine bağlı olduğundan, bu yoldaki veri saklayıcı dosyasına yazılabilmektedir. Ana veri yolu üstüne bağlı yapılar Şekil 4.4'de gösterilmiştir.



Şekil 4.4 Ana Veri Yoluna Bağlı Birimler

4.3 Komut Çözücü ve Kontrol Birimi

Tasarlanan bu işlemcide komutlar bir saat darbesinde tamamlanmaktadır. Dolayısıyla her saat darbesinde yeni bir buyruk alınmakta ve kontrol biriminde zamanlama amaçlı olarak bir sayıcının kullanılması gerekmemektedir. Saat darbesinin yükselen kenarında, program belleğinden alınan buyruk, komut saklayıcısına yazılmaktadır. Kontrol birimi bu komut saklayıcısını okumakta ve bellekten okunacak verileri, hangi işlemin yürütüleceğini ve ana veri yoluna hangi birimin yazacağını belirleyen kontrol sinyallerini üretmektedir. Kontrol biriminin genel görünümü Şekil 4.5'de gösterilmiştir.



Şekil 4.5 Kontrol Biriminin Genel Görünümü

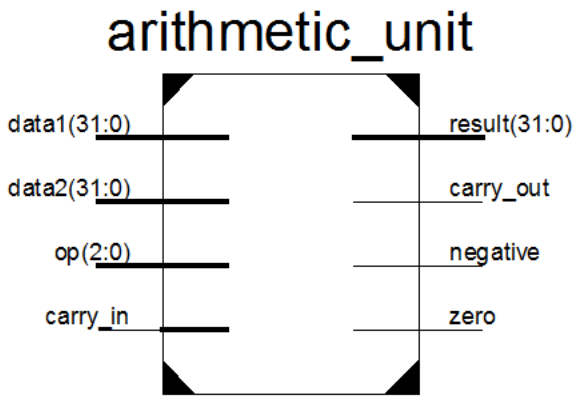
Şekil 4.5'de verilen kontrol birimi blok diyagramı, ALU komutları için geçerli olan yapıdır. ALU komutları ilk 8 biti işlem kodu, sonraki 8 bit hedef saklayıcı ve sonraki iki 8 bit de kaynak saklayıcılar olacak şekilde tasarlanmıştır. Dallanma ve giriş/çıkış işlemleri için tasarlanan komutların yapısı farklıdır. Ancak, kontrol birimi tarafından gerçekleştirilen işlemler Şekil 4.5'deki yapıya benzerdir.

4.4 Aritmetik Birim

Aritmetik işlemleri icra edecek olan donanım birimi aritmetik birimdir. Bu işlemci için tasarlanan aritmetik birim 8 farklı işlem gerçekleştirilebilmektedir.

4.4.1 Tasarlanan Modülün Yapısı

Aritmetik birimin girişine kaynak saklayıcıların değerlerini tutan busN ve busM yardımcı veri yolları bağlanmıştır. 8 farklı işlemi kodlamak üzere 3 bitlik seçim girişi bulunmaktadır ve uygulanan seçim girişine göre yapılacak aritmetik işlem belirlenmektedir. VHDL ile tasarlanan donanımın sentezlenmesi sonucu oluşan yapının görünümü Şekil 4.6'da verilmiştir.



Şekil 4.6 Aritmetik Birim

Görüldüğü üzere aritmetik birimin çıkışında elde, negatif ve sıfır durum bayrakları oluşmaktadır.Bu bayraklar yapılan aritmetik işlem sonuçlarına göre belirlenmektedir.

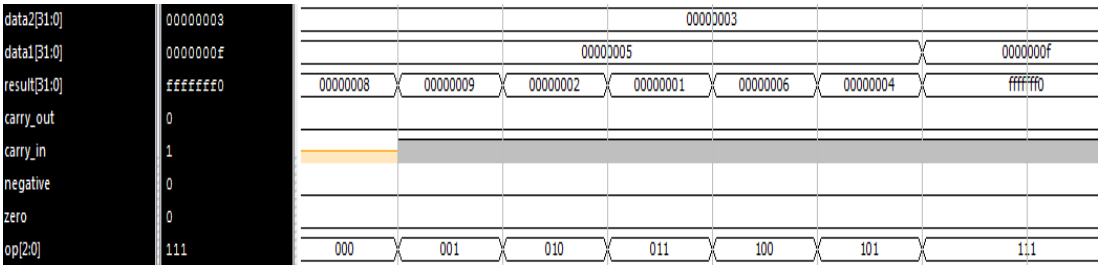
4.4.1 Aritmetik Birim Komutları

Aritmetik birimde 8 farklı işlem gerçekleştirilebilmektedir. Bu işlemler onaltılık tabandaki makine kodu karşılıkları ile birlikte Tablo 4.2'de verilmiştir.

Tablo 4.2 Aritmetik Birim Komutları

Makine Kodu	Sembolik İfade	İşlem
D0ddmmnn	ADD rN, rM, rd	rd = rN + rM
D1ddmmnn	ADDC rN, rM, rd	rd = rN + rM + c
D2ddmmnn	SUB rN, rM, rd	rd = rN - rM
D3ddmmnn	SUBC rN, rM, rd	rd = rN - rM - c
D40000nn	INC rN	rN = rN + 1
D50000nn	DEC rN	rN = rN - 1
D600mmnn	CMP rN, rM	rN ↔ rM
D70000nn	COM rN	rN = rN’

CMP işlemi diğerlerinden farklı olarak sadece karşılaştırma yapmakta yani aritmetik birim çıkışına herhangi bir sonuç yazmamaktadır. Tablo 2.2’de verilen komutları sırayla gerçekleştiren bir test kodu ile elde edilen sinyallerin durumu Şekil 4.7’de verilmiştir.



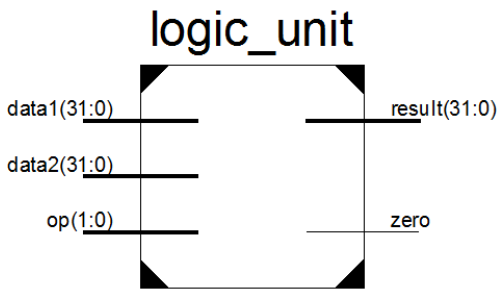
Şekil 4.7 Aritmetik Birim Simülasyon Sonucu

4.5 Mantık Birimi

Mantık birimi bit düzeyindeki mantıksal işlemlerin gerçekleştirilebilmesini sağlayan donanım bloğudur.

4.5.1 Tasarlanan Modülün Yapısı

Aritmetik biriminde olduğu gibi mantık biriminin de girişine kaynak saklayıcı değerlerini tutan busN ve busM yardımcı veri yolları bağlanmaktadır. Bu birim ve, veya, özel veya ve veri aktarımı olmak üzere dört farklı işlemi gerçekleştirmektedir. Sentez sonucunda oluşan modülün yapısı Şekil 4.8’de gösterilmiştir.



Şekil 4.8 Mantık Birimi

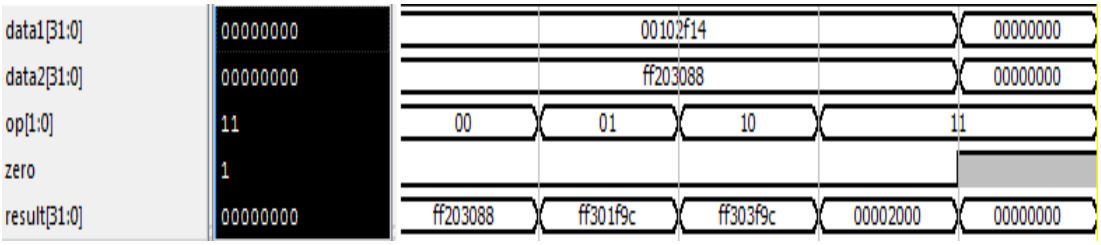
4.5.2 Mantık Birimi Komutları

Mantık birimi 3 temel mantıksal işlemi ve veri aktarma işlemini gerçekleştirmektedir. Veri aktarma komutu bir saklayıcıdaki değerin başka bir saklayıcıya da aktarılmasını sağlamaktadır. Mantık birimi komutlar onaltılık tabandaki makine kodları ile birlikte Tablo 4.3'de verilmiştir.

Tablo 4.3 Mantık Birimi Komutları

Makine Kodu	Sembolik İfade	İşlem
C0dd00nn	Mov rN, rd	rN = rd
C1ddmmnn	Xor rN, rM, rd	rd = rN xor rM
C2ddmmnn	Or rN, rM, rd	rd = rN or rM
C3ddmmnn	And rN, rM, rd	rd = rN and rM

Tablo 4.3'deki işlemleri verilen sırayla örnek bir değer için test eden kodun yazılması sonucunda elde edilen çıkış sinyalleri Şekil 4.9'da gösterilmiştir.



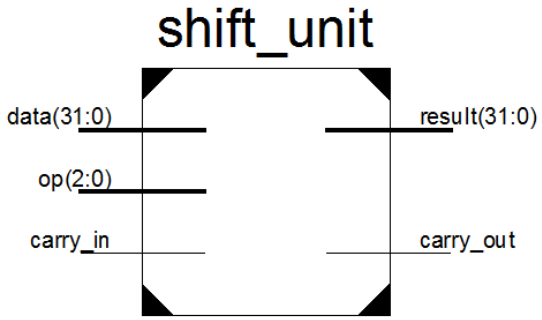
Şekil 4.9 Mantık Birimi Simülasyon Sonucu

4.6 Kaydırma Birimi

Kaydırma birimi verilerin sağa ve ya sola, eldeli ve ya eldesiz olarak kaydırılmasını sağlamaktadır.

4.6.1 Tasarlanan Modülün Yapısı

Kaydırma birimi 7 farklı kaydırma işlemini gerçekleştirebilmektedir. 7 farklı işlemin kodlanması için 3 bitlik seçim girişi bulunmaktadır. Ayrıca kaydırma işlemi ile elde bayrağı da değiştirebileceğinden, modülün elde çıkışı da bulunmaktadır. Modülün yapısı Şekil 4.10'da verilmiştir.

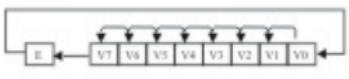
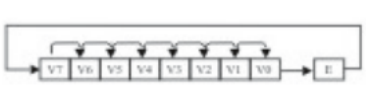
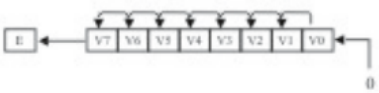
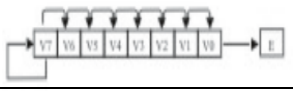


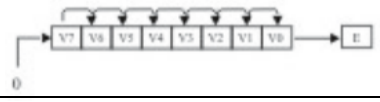
Şekil 4.10 Kaydırma Birimi

4.6.2 Kaydırma Birimi Komutları

Kaydırma biriminin komut listesi Tablo 4.4'de verilmiştir.

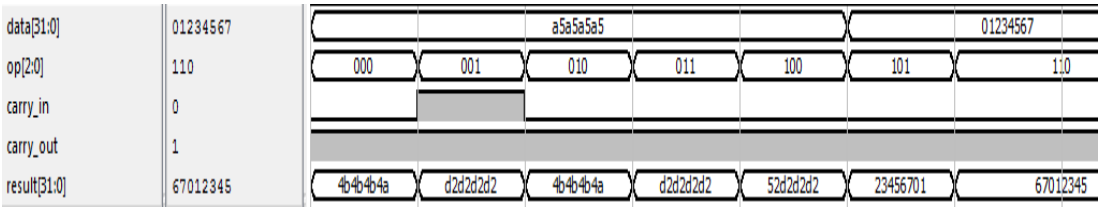
Tablo 4.4 Kaydırma Birimi Komutları

Makine Kodu	Sembolik İfade	İşlem
F00000nn	rotL rN	
E10000nn	rotR rN	
E20000nn	Asl rN	
E30000nn	Asr rN	

E40000nn	Lsr rN	
E50000nn	RotLB	Sola Bir Bayt Öteleme
E60000nn	RotRB	Sağa Bir Bayt Öteleme

Öteleme komutlarının gerçekleştirdiği işlerin daha iyi anlaşılması açısından, Tablo 4.4'de işlem kısmında öteleme diyagramları gösterilmiştir. Son iki öteleme işlemi, verinin sağa ve ya sola bir bayt ötelenmesi için oluşturulan komutlardır.

Verilen komutların sırayla örnek bir veri için test edilmesi ile simülasyon sonucu Şekil 4.11'de verilmiştir.



Şekil 4.11 Kaydırma Birimi Simülasyon Sonucu

4.7 Kripto Birimi

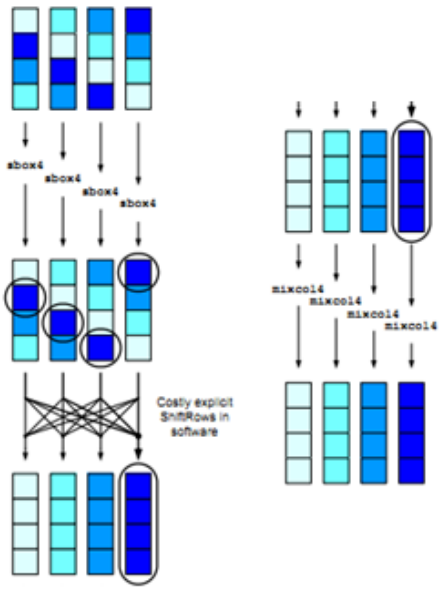
Tasarlanan bu işlemciyi standart genel amaçlı işlemcilerden farklı yapan en önemli kısmı kripto birimidir. Bu birim ile işlemciye genel amaçlı kullanıma uygun komutların yanı sıra, kriptoloji uygulamalarına özel işlemleri gerçekleştirecek donanımların da eklenmesi sağlanmıştır. Bu tip uygulamaya özel komut ve donanımları bünyesinde barındıran işlemciler ASIP (Application-Specific Instruction Set Processor) olarak nitelendirilmektedir. Bu tez kapsamında kendi geliştirdiğimiz bir işlemciye ek donanım ekliyor olsak da, ASIP işlemcilerin elde edilmesi amacıyla uygulanan bir diğer yöntem komut seti genişletilmesi (Instruction Set Extension) işlemidir. Bu yöntemde genel amaçlı işlemci çekirdeği hazır donanım olarak alınır ve işlemcinin gerekli denetim sinyalleri üretmesini sağlayacak şekilde düzenlemeler yapıldıktan sonra eklenecek donanımlar işlemci çekirdeğinin çevresine yerleştirilir.

Tasarlanan bu kripto birimi, 2. bölümde anlatılan AES algoritmasını bu işlemcide en verimli şekilde gerçekleştirecek şekilde düşünülmüştür. İşlemcide AES algoritmasına

özel komutların bulunması ile veri şifreleme ve ya şifreli veriyi çözme işlemleri çok daha hızlı ve verimli bir şekilde yapılabilmektedir.

4.7.1 Satır – Sütun Dönüştürme Problemi

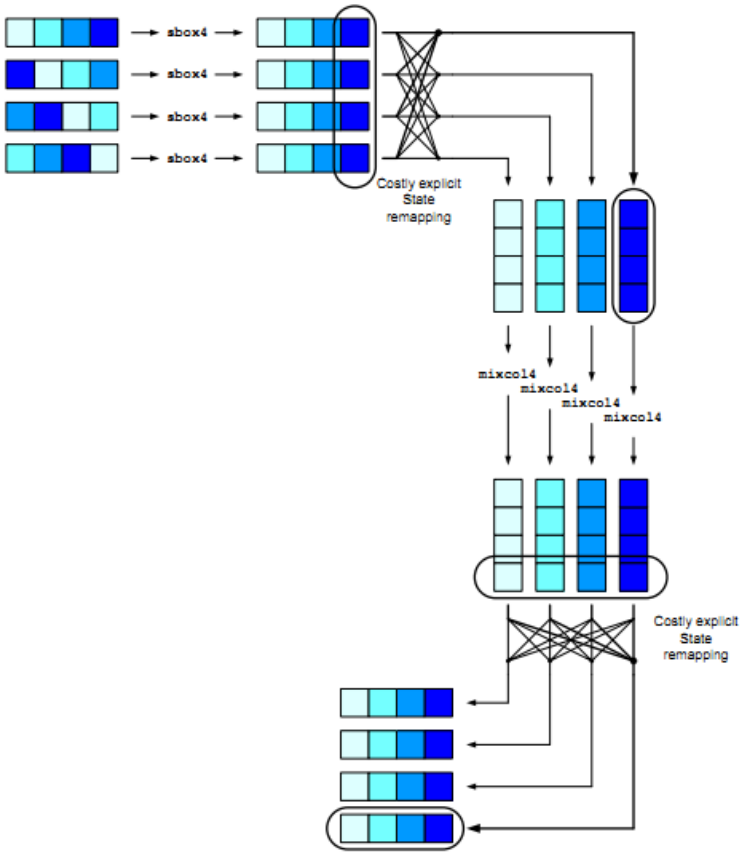
2. bölümde anlatıldığı üzere, AES algoritmasında veriler durum matrisi şeklinde ifade edilmektedir. Matrisin her bir sütun ve satırı 4 bayt yani 32 bit uzunluğundadır. 32 bit işlemcilerde AES algoritmasının gerçeklenmesi için durum matrisinin satır ve ya sütunları işlemcinin 32 bitlik saklayıcılarında tutulmalıdır. Fakat AES algoritması işlemlerinden satır kaydırma işlemi matrisin satırları üstünde işlem yaparken, sütun karıştırma işlemi matrisin sütunları üzerinde işlem yapmaktadır. Örnek olarak, işlemci saklayıcılarında satırları tuttuğumuzu düşünecek olursak sütun karıştırma işlemine sokulacak sütunu oluşturmak için 4 satırın da birer baytının okunarak bir sütun oluşturulması gerekir. Bu sütun oluşturma işlemini her bir sütun için gerçekleştirmemiz gerektiğini ve algoritma en az 10 tur içerdiğinden dolayı bu işlemin her turda tekrarlanacak olması gerektiğini düşünecek olursak, algoritmanın yazılım ile bu tip bir yaklaşım kullanılarak gerçekleştirilmesinin ciddi bir ek maliyet getireceği açıktır.



Şekil 4.12 Sütunlara Dayalı Olarak Yapılan İşlem [10]

Şekil 4.12'de saklayıcılarda sütunların tutulduğu durum için oluşan satır elde etme problemi gösterilmiştir. Satır kaydırma işleminin getirdiği yüksek maliyet Şekil 4.12'de net bir şekilde gözlenebilmektedir. Benzer problem ile satırların

saklayıcılarda tutulduğu durumda da karşılaşılacaktır. Satırlara dayalı yapılan yazılım gerçeklemesinde oluşan durum da Şekil 4.13'de gösterilmiştir.



Şekil 4.13 Satırlara Dayalı Olarak Yapılan İşlem [10]

4.7.2 AES Komutları

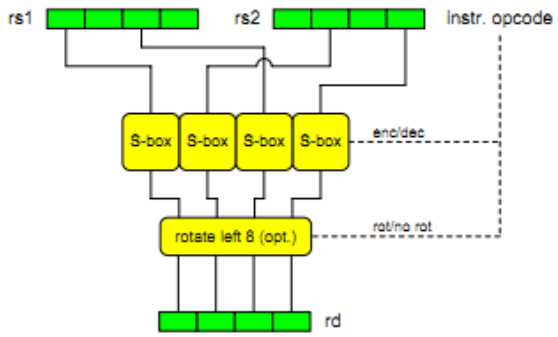
AES algoritmasını gerçekleştirmek üzere öne sürülmüş birçok komut yapısı bulunmaktadır. Uygun yöntemin seçilmesi için, tasarlanan genel amaçlı işlemcinin bir saat işareti çevriminde buyrukları icra ediyor olması ve 32 bit işlem yapabiliyor olmasının yanında gerçekleştirilecek mimariye uygun olması unsurları göz önünde bulundurulmuştur.

Bir önceki bölümde anlatılan satır-sütun dönüştürme problemine çözüm oluşturan ve tasarladığımız işlemci yapısına da uygun olarak görülen sbox4s [1], isbox4s [1], isbox4r [1], mixcol4s [1], ve imixcol4s [1] komutlarının işlemcimizin komut setine eklenmesine karar verilmiştir. Komutları icra edecek olan donanım birimleri de gene VHDL ile gerçekleştirilmiştir.

Bu komutlar sütunlara dayalı işlem yapmaktadırlar ve iki kaynak saklayıcı gerektirmektedirler. Bu komut yapılarının sağladığı en belirgin avantaj ise satır kaydırma işlemini dolaylı olarak icra etmeleridir. AES algoritmasını bu komutlar ile gerçeklerken satır kaydırma işlemine gerek duymamaktayız. Komutların yapıları bir sonraki kısımda irdelenmiştir.

4.7.2.1 sbox4s – isbox4s - sbox4r

Bu komutlar AES algoritmasının her turunda gerçekleştirilen bayt değiştirme ve ters bayt değiştirme işlemini gerçekleştirmektedir. Komutun yapısı Şekil 4.14'de gösterilmiştir.



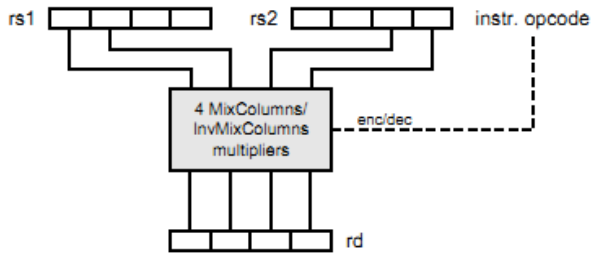
Şekil 4.14 sbox4s, isbox4s ve sbox4r Komutlarının Yapısı [10]

Şekil 4.14'den görüldüğü üzere girişte iki adet kaynak saklayıcı bulunmakta ve bu saklayıcıların şekilde gösterilen baytları alınarak S-kutusundan geçirilmektedir. S-kutusu çıkışlarının bağlı olduğu bayt kaydırma birimi bu donanım bloğunun anahtar üretimi için de kullanılmasını sağlamaktadır ve sbox4r komutu ile bu işlem gerçekleştirilmektedir. Sbox4s komutu şifreleme işlemi için kullanılırken isbox4s komutu şifre çözme için kullanılmaktadır ve S-kutuları yerine ters S-kutuları kullanılması ile elde edilmiştir.

Bayt değiştirme komutu bir sonraki kısımda açıklanan sütun karıştırma komutu (mixcol4s) ile birlikte bir sonraki kısımda test edilmiştir. Bu şekilde AES algoritmasının bir turunda gerçekleştirilen bayt değiştirme, satır kaydırma ve sütun karıştırma işlemlerinin sadece anlatılan iki komut ile gerçeklenebildiği gösterilmiştir.

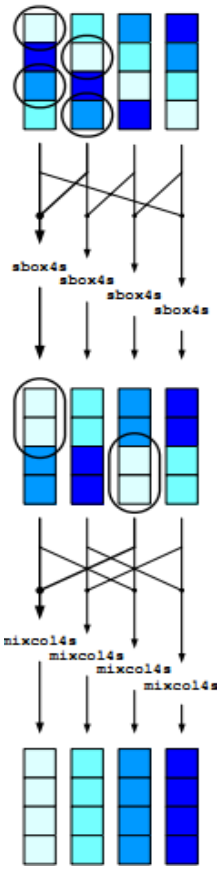
4.7.2.2 mixcol4s – imixcol4s

Sütun karıştırma ve ters sütun karıştırma işlemlerinin gerçekleştirilmesini sağlayan bu komutların yapısı Şekil 4.15'de gösterilmiştir.



Şekil 4.15 Mixcol4s ve imixcol4s Komutlarının Yapısı [1]

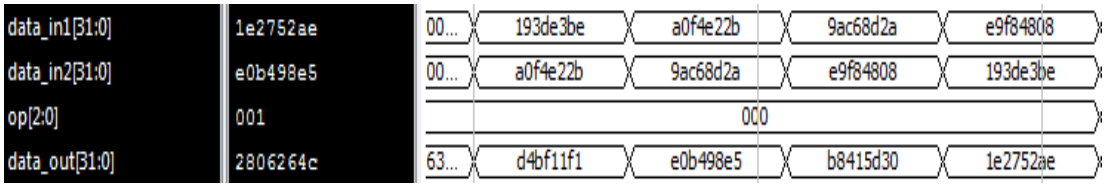
Bu komut yapısı gereğince, matrisin iki farklı sütununu tutan iki kaynak saklayıcısının şekilde gösterilen baytları alınarak tasarlanan sütun çarpıcı modülünden geçirilmektedir. Mixcol4s ve imixcol4s komutları arasındaki tek fark, imixcol4s için ters sütun çarpıcı modülünün kullanılmasıdır.



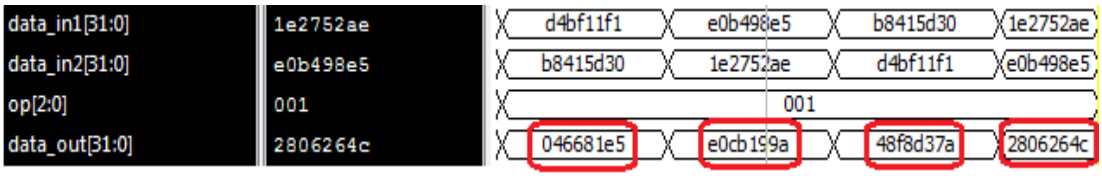
Şekil 4.16 Satır Kaydırma İşleminin Dolaylı Olarak Gerçekleştirilmesi [10]

Verilen komutlar ile satır kaydırma işleminin dolaylı olarak nasıl gerçeklenebileceği Şekil 4.16'da gösterilmiştir. Bu komutlar kullanılırken sbox4s işleminden geçirilen sütunlar bir ara matrise yazılmaktadır. Görüldüğü üzere kaynak saklayıcılarının şekide verildiği gibi seçilmesi durumunda, sütunlara dayalı işlem yapılmasına rağmen elde edilen sonuçta satır kaydırma işlemi gerçekleşmiş durumdadır. AES algoritmasını bu komutlar ile gerçeklerken dikkat edilmesi gereken bir unsur kaynak saklayıcılarının seçimidir. Satır kaydırma işleminin başarıyla dolaylı olarak gerçekleştirilebilmesi için kaynak saklayıcılar Şekil 4.16'da gösterilen şekilde seçilmelidir.

Şekil 4.17 ve Şekil 4.18'de bu komutlar kullanılarak yapılan bir örnek işlem gösterilmiştir. Bu örnekte Şekil 3.4'de verilen durum matrisi önce sbox4s, ardından da mixcol4s işlemlerine tabi tutulmuştur. İşlem sonucunda elde edilen verinin Şekil 3.6'da elde edilen sonuç ile aynı olması yapılan işlemlerin doğru olduğunu göstermektedir.



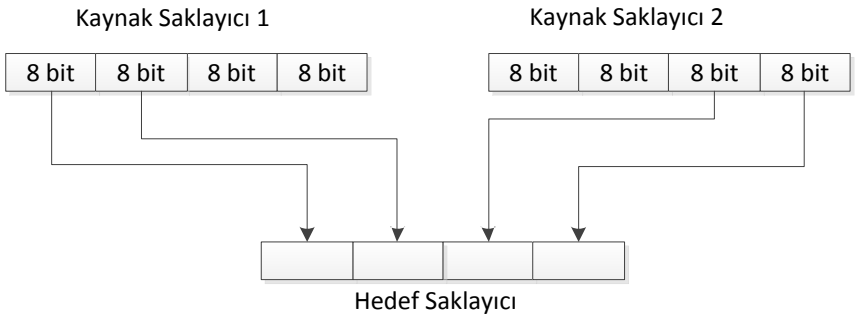
Şekil 4.17 Sbox4s İşlemi Örneği



Şekil 4.18 Mixcol4s İşlemi Örneği

4.7.2.3 Sütun Düzeltme (FixCol)

Şekil 3.1'de verilen AES algoritmasının blok yapısından görüleceği üzere son turda sütun karıştırma işlemi gerçekleştirilmemektedir. Ancak kullandığımız yapıda mixcol4s komutu aynı zamanda satır kaydırma işleminin dolaylı olarak gerçekleştirilmesinde rol almaktadır. Satır kaydırma işleminin, sütun karıştırma işlemi yapmadan gerçekleştirilebilmesi için FixCol komutu tanımlanmıştır. Bu komutun gerçekleştirdiği işlem Şekil 4.19'da gösterilmiştir.



Şekil 4.19 Sütun Düzeltme (FixCol) Komutunun Yapısı

5. TASARLANAN BLOKLAR İLE İŞLEMCİ MİMARİSİNİN OLUŞTURULMASI

Bir önceki kısımda işlemciyi gerçekleştirirken kullanılan bazı temel yapılardan söz edilmişti. Bu kısımda ise 4. bölümde tasarlanan yapılar da kullanarak Harvard mimarisinde bir RISC işlemcinin nasıl oluşturulduğu açıklanacak, işlemcinin komut seti ve mimari yapısı ortaya konulacaktır.

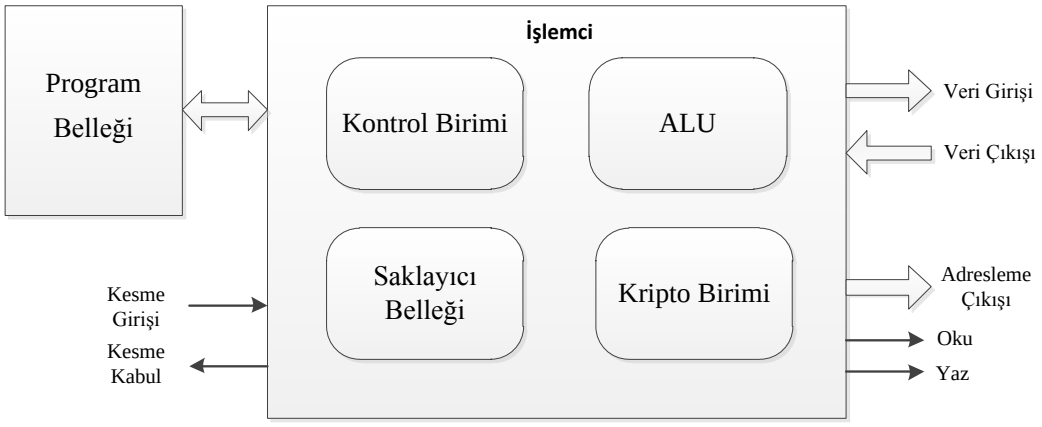
5.1 İşlemcinin Genel Görüntüsü

Tasarlanan işlemci buyrukları harici bir program belleğinden okumaktadır. İşlemci program belleğini adresleme amaçlı olarak toplam 12 adres hattına sahiptir. Bu da 4 kB uzunluğuna kadar buyruk adresleme imkânı sağlamaktadır. Program belleği işlemciye harici olarak bağlandığından yazılacak olan program uzunluğunu göre boyutu küçültülebilmektedir.

İşlemciye çevre donanımlarının bağlanabilmesi ve bu donanımlara erişim sağlanabilmesi açısından genel amaçlı bir giriş/çıkış iskelesi (port) tanımlanmıştır.

Bunun yanında, kesme kaynaklarının işlemciye bağlanabilmesini sağlayan bir adet maskelenebilir kesme girişi de bulunaktadır.

İşlemcinin genel dış görünümü Şekil 5.1'de gösterilmiştir.



Şekil 5.1 İşlemcinin Genel Görünümü

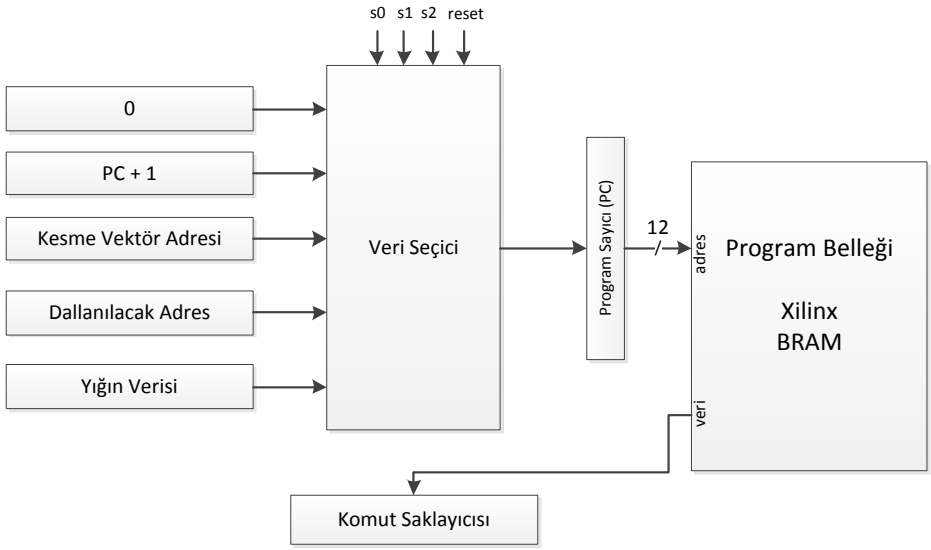
5.2 İşlemcinin Donanım Yapısı ve Senkronizasyonu

İşlemcinin donanım yapısını oluşturan birimler ve bu birimlerin ne şekilde sisteme entegre edileceği hakkında bir önceki kısımda bilgi verilmiştir. Bu kısımda tüm birimlerin oluşturulan mimari içerisinde ne şekilde yerleştirildiği ve hangi görevleri üstlendiği üstünde durulacaktır.

İşlemcinin buyrukları doğru bir sıra ile icra etmesini sağlayan program sayma işlemi kontrolü, sistemin en önemli işlemlerinden biridir. Program sayıcı (Program Counter-PC) saklayıcısı bir sonraki buyruğun adresini tutmaktadır ve program sayıcısının değerinin belirlenmesi buyruk icra edilirken gerçekleştirilmektedir.

5.2.1 Program Sayma İşlemi

Program sayıcı işlemciye sıfırlama (reset) sinyali geldiğinde sıfırlanmakta ve program belleğinin başına işaret etmektedir. Bu giriş ile işlemcinin yazılan programın tekrar başına dönmesi sağlanabilmektedir. Program sayıcı kesme girişi aktif olmadığında ve dallanma komutları dışındaki komutları yürütürken bir artarak devam etmektedir. Bu şekilde program belleğinden sıralı olarak buyruklar okunarak yürütülmektedir. Dallanma komutları icra edilirken, eğer test koşulu varsa ve bu koşul sağlanıyorsa program sayıcıya dallanacağı adres yüklenmektedir. Şartsız dallanma komutları için program sayıcı buyrukta verilen adres ile yüklenmektedir.



Şekil 5.2 Program Okuma Birimi

Şekil 5.2'de program sayıcı kontrolünü sağlayan birimin genel blok diyagramı verilmiştir. Burada yığın verisi olarak gösterilen değer, alt programdan dönülürken gidilecek buyruğun adresidir.

Şekil 5.2 üzerinde gösterilmemesine rağmen, program sayıcı giriş/çıkış işlemleri için diğer komutlardan farklı olarak bir saat darbesi süresince bekletilmektedir. Xilinx'in blok ram çekirdeklerini harici bellek olarak kullanırken, okuma işlemlerinin işlemci ile senkron olması için okuma işlemi yürütülürken ekstra bir saat darbesi için buyruğun değişmemesi gerekmekte olduğundan, program sayıcı bekletilerek aynı buyruğu okuması sağlanmıştır. Senkronizasyonu sağlayacak birim donanımsal olarak gerçeklendiğinden, kullanıcı bu durumdan habersiz olarak okuma işlemini başarılı bir şekilde gerçekleştirebilmektedir.

5.2.1 Yığın Yapısı

Bilindiği üzere yığın, okuma işlemi yapılırken en son yazılan verinin öncelikli olarak okunmasını sağlayan bir yapıdır. Yığın yapıları işlemcilerde de geri dönüş adreslerinin saklanması ve alt programlara dallanırken saklayıcıların yedeklenmesi gibi görevleri gerçekleştirmek üzere kullanılmaktadırlar.

Bizim tasarladığımız işlemcide de yığın yapısı bulunmaktadır ve temel görevi alt programlara dallanırken işlemcinin geri dönüş adresinin saklanmasıdır. Dallanma işlemi gerçekleştirilirken, bir sonraki buyruğun adresi yani program sayıcının bir fazlası yığına yazılmakta ve yığın göstergesi bir artırılmaktadır. Geri dönüş adresinin yanı sıra, bayrakların alt programda değiştirilmesine karşı tedbir olarak elde, sıfır ve negatif bayrakları da yığına yazılmaktadır. Yığına yazılan verinin formatı Şekil 5.3'de gösterilmiştir.

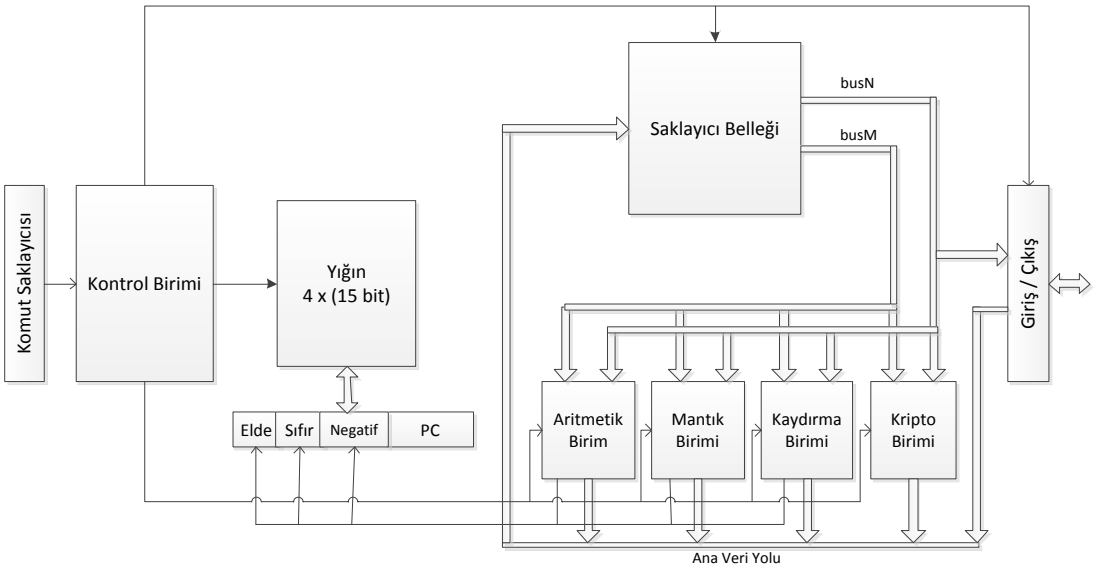
Şekil 5.3 Yığına Yazılan Veri Formatı

5.2.3 Veri Yolu Düzeni

4. bölümde işlemcimizde kullandığımız bazı birimleri tanıtırken bunların hangi veri yolları üzerinden haberleştiğinden de kısaca bahsetmiştik. Bu kısımda işlemci

birimlerinin veri yolu yapıları ile bağlantı düzenleri toplu halde irdelenerek işlemcinin genel çalışma mantığının ortaya konması amaçlanmaktadır.

Oluşturulan yapıda toplam üç adet veri yolu bulunmaktadır. Bunlarda ikisi işleme tabi tutulacak verileri tutmakta olan 32 bitlik yardımcı veri yollarıdır, busN ve busM olarak isimlendirilmişlerdir. Bu iki veri yolu işlem birimlerine giriş olarak gelmektedirler. BusN ve busM veri yollarının dışında, bir adet de ana veri yolu bulunmaktadır. Ana veri yolu yapılan işlemlerin sonucunun yazıldığı veri yoludur. Ana veri yoluna birçok birim bağlıdır ve hangi birimin bu yola erişim sağlayabileceği kontrol birimi tarafından belirlenir. Anlatılan bu veri yolu düzeni ile birlikte işlemcinin genel donanım şeması Şekil 5.4'de verilmiştir.



Şekil 5.4 Veri Yolu Düzeni

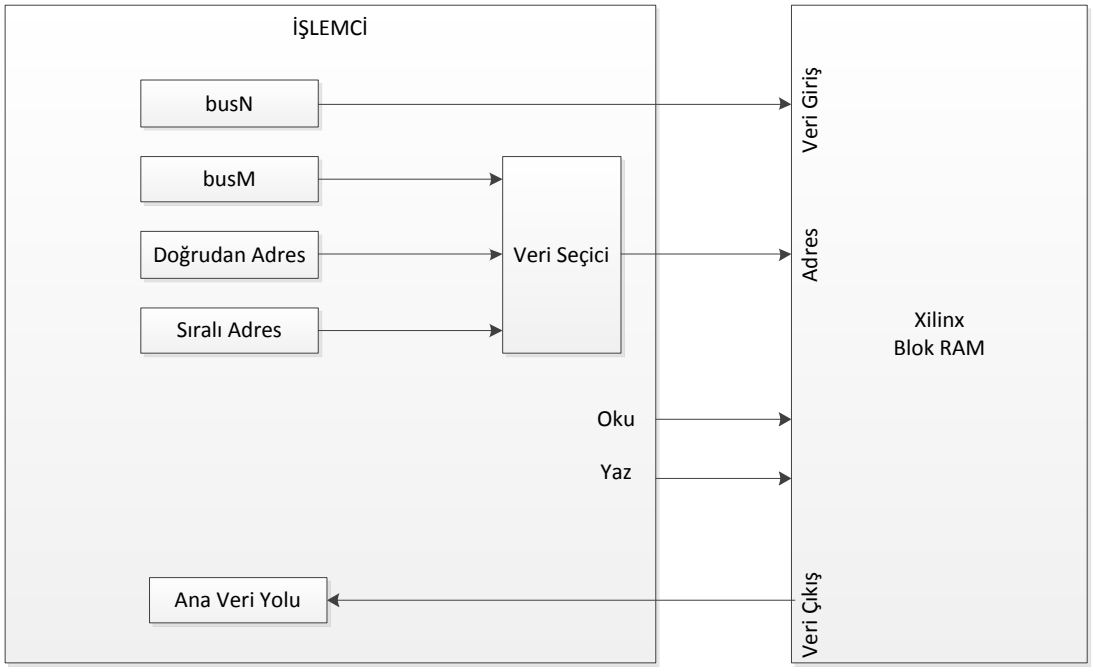
5.2.4 Giriş/Çıkış Organizasyonu

İşlemciye çevre donanımların bağlanabilmesi ve veri giriş-çıkışı yapabilmesi için bir adet iskele bulunmaktadır. Şekil 5.4'de de gösterildiği üzere iskeleye yazılacak veriler busM veri yolu üzerinden sağlanmakta, okunan veriler ise ana veri yoluna aktarılmaktadır. Bu işlemcinin iskelesinin 18 bitlik adresleme hattına sahip olması sebebiyle birçok harici birim adres kod çözücüler kullanılarak işlemciye bağlanabilecektir.

Bu giriş/çıkış iskelesi işlemciye birçok artı özellik kazandırmaktadır. Bu özelliklerin en önemlisi dolaylı işlemciye adresleme yeteneği kazandırmasıdır. Tasarlanan bu

işlemci, buyrukları bir saat darbesinde gerçekleştirmektedir ve dolayısıyla doğrudan adresleme ile çalışmaktadır. Ancak birçok uygulamada saklayıcılarda verinin kendisinin değil adresinin saklanması gerekebilmektedir. Bu tür durumlarda öncelikle saklayıcılardaki adres değerinin alınması ve sonra bu adresteki veriye erişilmesi suretiyle dolaylı adresleme işlemi gerçekleştirilir. Bu iskele sayesinde işlemci saklayıcılarındaki değer adres olarak çıkışa bağlı bir bellek elemanına gönderilebilmektedir. Bu adres gönderme işlemi busM veri yolu üzerinden yapılmaktadır. Dolaylı adresleme imkanının yanında, sıralı bellek gözlerinin tek komut ile iskele üzerinden okunmasını sağlayan özellik de okuma işlemlerinde kolaylık sağlayabilmektedir.

Örnek bir şema oluşturması açısından iskeleye bir blok ram bağlandığı durum için oluşan blok şema Şekil 5.5'de gösterilmiştir.



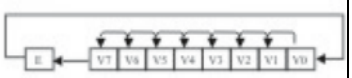
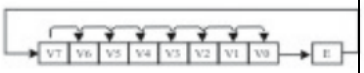
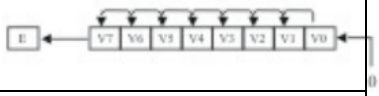
Şekil 5.5 İşlemci ile BRAM'in İskele Üzerinden Bağlantı Şeması

5.3 Komut Kümesi

Bu kısımda işlemcinin komut listesi tanıtılacaktır. Bu işlemciye ilişkin belirlenen bazı komutlar 4. bölümde gösterilmişti. Tablo 4.1, 4.2, 4.3 ve 4.4 de saklayıcı dosyası, aritmetik birim, mantık birimi ve kaydırma birimine ilişkin komutlar

listelenmişti. Kolay takip edilebilmesi açısından, bu komutlar toplu halde Tablo 5.1'de tekrar gösterilmiştir.

Tablo 5.1 ALU ve Saklayıcı Dosyası Komutları

Makine Kodu (32 bit)	Sembolik İfade	İşlem
80kkkknn	MovH rN, <k>	rN = <k> (Üst 16 bit)
90kkkknn	MovL rN, <k>	rN = <k> (Alt 16 bit)
D0ddmmnn	ADD rN, rM, rd	rd = rN + rM
D1ddmmnn	ADDC rN, rM, rd	rd = rN + rM + c
D2ddmmnn	SUB rN, rM, rd	rd = rN - rM
D3ddmmnn	SUBC rN, rM, rd	rd = rN - rM - c
D40000nn	INC rN	rN = rN + 1
D50000nn	DEC rN	rN = rN - 1
D600mmnn	CMP rN, rM	rN ↔ rM
D70000nn	COM rN	rN = rN'
C0dd00nn	Mov rN, rd	rN = rd
C1ddmmnn	Xor rN, rM, rd	rd = rN xor rM
C2ddmmnn	Or rN, rM, rd	rd = rN or rM
C3ddmmnn	And rN, rM, rd	rd = rN and rM
F00000nn	rotL rN	
E10000nn	rotR rN	
E20000nn	Asl rN	
E30000nn	Asr rN	
E40000nn	Lsr rN	
E50000nn	RotLB	Sola Bir Bayt Öteleme
E60000nn	RotRB	Sağa Bir Bayt Öteleme

Program akışının belirli şartlar altında ve ya koşulsuz olarak başka bir adrese dallanmasını sağlayan dallanma buyrukları Tablo 5.2'de listelenmiştir. Ayrıca alt program çağrılarının yapılmasını ve alt programdan dönüş işleminin gerçekleştirilmesini sağlayan komutlara da Tablo 5.2'de yer verilmiştir.

Tablo 5.2 Dallanma Buyrukları

Makine Kodu (32 bit)	Sembolik İfade	İşlem
40000ddd	Bne <dest>	Dallan Eşit Değilse
44000ddd	Beq <dest>	Dallan Eğer Eşitse
48000ddd	Bgt <dest>	Dallan Eğer Büyükse
4C000ddd	Blt <dest>	Dallan Eğer Küçükse
50000ddd	Bge <dest>	Dallan Eğer Büyük Ve Ya Eşitse
54000ddd	Ble <dest>	Dallan Eğer Küçük Ve Ya Eşitse
00000ddd	Call <dest>	Alt Program Çağrısı
10000ddd	Goto <dest>	Koşulsuz Dallanma
20000000	Ret	Alt Programdan Dönüş
24000000	Reti	Kesme Programından Dönüş

Kesme girişlerinin aktif edilmesi ve ya engellenmesi için Tablo 5.3'de gösterilen komutlar kullanılmaktadır.

Tablo 5.3 Kesme İşlemleri

Makine Kodu (32 bit)	Sembolik İfade	İşlem
28000000	di	Kesmeleri Engelle
2C000000	ei	Kesmelere İzin Ver

İskele üzerinden okuma yazma işlemlerinin gerçekleştirilmesi için kullanılabilen komutlar da Tablo 5.4'de tanıtılmıştır. "Mov (<k>),rN" ve "Mov rN, (<k>)" komutları için gösterilen makine kodları ikili düzendedir. Diğer komutların verilen makine kodu karşılıkları ise on altılık tabandadır.

Tablo 5.4 İskele Okuma/Yazma Komutları

Makine Kodu (32 bit)	Sembolik İfade	İşlem
3000mmnn	Mov (rM), rN	İskele Dolaylı Adrese Yazma
340000nn	Mov (++), rN	İskele Sıralı Adrese Yazma
00 11 10 kk kkkk kkkk kkkk	Mov (<k>),rN	İskele Doğrudan Adrese Yazma
F000mmnn	Mov rN, (rM)	İskele Dolaylı Adresden Okuma
F40000nn	Mov rN, (++)	İskele Sıralı Adresden Okuma
11 11 10 kk kk kkkk kkkk	Mov rN,(<k>)	İskele Doğrudan Adresi Okuma

Tablo 5.5'de ise AES algoritmasını gerçeklemek üzere komut setine eklenen özel buyruklar gösterilmiştir.

Tablo 5.4 AES Komutları

Makine Kodu (32 bit)	Sembolik İfade	İşlem
D8ddmmnn	Sbox4x rN,rM,rD	Bayt Değiştirme
D9ddmmnn	Mixcol4s rN,rM,rD	Sütun Karıştırma
DAddmmnn	isbox4s rN,rM,rD	Ters Bayt Değiştirme
DBddmmnn	imixcol4s rN,rM,rD	Ters Sütun Karıştırma
DCddmmnn	RotWord rN,rM,rD	(rN = rM), rd = sbox4r(rN)
DDddmmnn	FixCol rN,rM,rD	Sütun Düzeltme

6. AES ALGORİTMASININ TASARLANAN İŞLEMCİ İLE GERÇEKLEŞTİRİLMESİ

Tasarlanan işlemci ile verilen komut kümesinin doğru bir şekilde icra edilebildiğinin test edilmesi amacıyla AES algoritmasını gerçekleyecek olan program yazılarak işlemci üzerinde çalıştırılmıştır.

Program öncelikle sembolik dil kullanılarak oluşturulmuş, sonrasında buyruklar makine kodu karşılıklarına çevrilerek işlemcinin program belleğine yüklenmiştir.

AES algoritmasını işlemcinin komut setinde bulunan buyruklar ile gerçekleyen program Tablo 6.1'de verilmiştir.

Tablo 6.1 AES Algoritması İçin Yazılan Program Kodu

Satır	Etiket	Komut	Açıklama	Makine Kodu
1	basla	MOVH R0, 3243h	Sifrelenecek veriyi saklayıcılara yazıyoruz.	80324300
2		MOVL R0, F6A5h		90f6a800
3		MOVH R1, 885Ah		80885a01
4		MOVL R1, 308Dh		90308d01
5		MOVH R2, 3131h		80313102
6		MOVL R2, 98A2h		9098a202
7		MOVH R3, E037h		80e03703
8		MOVL R3, 0734h		90073403
9		MOVH R8, 0000h	Döngü Sayacı = 9	80000008
10		MOVL R8, 0009h		90000908
11	İlk_tur	MOV R10, <00>	İskeleden 00 adresini oku	f800000a
12		XOR R0,R10,R0		c100000a
13		MOV R10,(++)	İskeleden bir soraki veriyi al	f400000a

14		XOR R1,R10,R1		c101010a
15		MOV R10,(++)		f400000a
16		XOR R2,R10,R2		c102020a
17		MOV R10,(++)		f400000a
18		XOR R3,R10,R3		c103030a
19	Dongu			
20		Sbox4s R0,R1,R4	Sbox4s İşlemi	d8040100
21		Sbox4s R1,R2,R5		d8050201
22		Sbox4s R2,R3,R6		d8060302
23		Sbox4s R3,R0,R7		d8070003
24		Mixcol4s R4,R6,R0	Mixcol4s İşlemi	d9000604
25		Mixcol4s R5,R7,R1		d9010705
26		Mixcol4s R6,R4,R2		d9020406
27		Mixcol4s R7,R5,R3		d9030507
28		MOV R10,(++)		f400000a
29		XOR R0,R10,R0		c100000a
30		MOV R10,(++)		f400000a
31		XOR R1,R10,R1		c101010a
32		MOV R10,(++)		f400000a
33		XOR R2,R10,R2		c102020a
34		MOV R10,(++)		f400000a
35		XOR R3,R10,R3		c103030a
36		DEC R8	Sayacı azalt	d5000008
37		CMP R8,#00	0 olmuş mu?	d6000809
38		BNE Dongu	Olmamışsa bir sonraki tura geç	40000012
39		Sbox4s R0,R1,R4		d8040100
40		Sbox4s R1,R2,R5		d8050201
41		Sbox4s R2,R3,R6		d8060302
42		Sbox4s R3,R0,R7		d8070003
43		Fixcol R4,R6,R0		dd000604
44		Fixcol R5,R7,R1		dd010705
45		Fixcol R6,R4,R2		dd020406
46		Fixcol R7,R5,R3		dd030507
47		MOV R10,(++)		f400000a

48		XOR R0,R10,R0		c100000a
49		MOV R10,(++)		f400000a
50		XOR R1,R10,R1		c101010a
51		MOV R10,(++)		f400000a
52		XOR R2,R10,R2		c102020a
53		MOV R10,(++)		f400000a
54		XOR R3,R10,R3		c103030a
55	SON	MOV R0,R0	İşlem Yok (NOP)	c0000000
56		BRA SON		10000035

Verilen programda 1-10 satırlarındaki komutlar ile şifrelenecek veriyi ve döngü sayacını saklayıcı dosyasına yazmaktadır. 11-20 satırlarında iskeleye bağlı BRAM'den şifre okunarak ilk turdaki özel veya işlemi gerçekleştirilmektedir. 20-38 satırlarında AES algoritmasının bir turu gerçekleştirilmektedir. Görüldüğü üzere burada sbox4s ve mixcol4s komutları işlemcinin diğer komutları ile birlikte kullanılmaktadır. Bu blok kullanılan sayaç değişkeni ile 9 kere tekrarlanmaktadır. 39-54 satırlarında da AES algoritmasının son turu gerçekleştirilmektedir. Burada sütun karıştırma işlemi gerçekleştirilmeyeceğinden FixCol komutu kullanılmıştır.

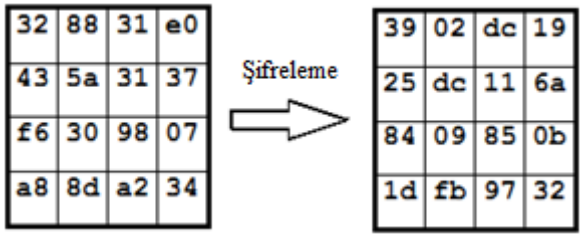
Programın işlemcide çalıştırılması için makine koduna çevrilen program, program belleği olarak kullanılan BRAM'in içine yazılmıştır. Blok ram'e bu verilerin yazılması için ".coe" uzantılı BRAM başlatma dosyası (Bram initialization file) oluşturulmuştur. Bu şekilde yazılan kodlar kolayca program belleğine yerleştirilebilmektedir. Makine kodlarının belleğe yazılmasını sağlayacak bir başlatma dosyasının içeriği Şekil 6.1'de gösterilmiştir.

```
memory_initialization_radix=16;
memory_initialization_vector=
80324300,90f6a800,80885a01,90308d01,80313102,9098a202,80e03703,90073403,
80000008,90000908,f800000a,c100000a,f400000a,c101010a,f400000a,c102020a,
f400000a,c103030a,d8040100,d8050201,d8060302,d8070003,d9000604,d9010705,
d9020406,d9030507,f400000a,c100000a,f400000a,c101010a,f400000a,c102020a,
f400000a,c103030a,d5000008,d6000809,40000012,d8040100,d8050201,d8060302,
d8070003,dd000604,dd010705,dd020406,dd030507,f400000a,c100000a,f400000a,
c101010a,f400000a,c102020a,f400000a,c103030a,c0000000,10000035;
```

Şekil 6.1 Programın BRAM Başlatma Dosyasına Yazılması

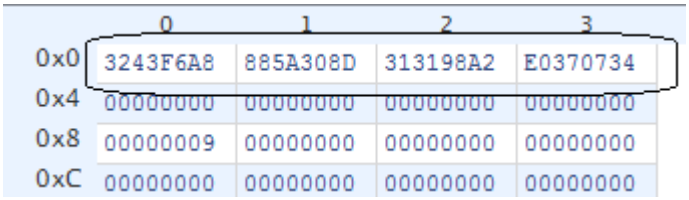
Benzer yöntem şifrenin iskeleye bağlı BRAM'e yazılması için de kullanılmıştır

Verilen bu programda belleğe yazılan veri ve şifreleme sonucunda elde edilmesi gereken doğru şifrelenmiş veri Şekil 6.2'de gösterilmiştir.

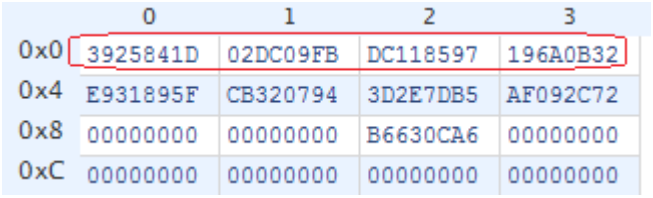


Şekil 6.2 Örnek Bir Şifreleme [8]

Şekil 6.2'de verilen giriş matrisi saklayıcı dosyasına yazılarak program çalıştırıldığında elde edilen sonuç Şekil 6.4'de gösterilmiştir. Veri şifrelenmeden önce bellekteki görüntüsü Şekil 6.3'de verilmiştir.



Şekil 6.3 Şifreleme Öncesi Saklayıcı Belleği Görünümü



Şekil 6.4 Şifreleme Sonrası Saklayıcı Belleği Görünümü

Şekil 6.4'de görüldüğü üzere elde edilen şifrelenmiş veri Şekil 6.2'de verilen veri ile aynıdır. Dolayısıyla, işlemcimizin komut seti kullanılarak yazılan program ile AES algoritmasını başarılı bir şekilde gerçekleştirebilmekteyiz.

7. SONUÇLAR VE TARTIŞMA

Çalışmanın başında AES algoritması ve işlemcilerin donanımsal yapıları üzerine araştırma yapılmıştır. İşlemcilerin donanım yapıları hakkında elde edilen bilgiler kullanılarak ve örnek işlemci yapıları incelenerek genel amaçlı kullanıma hizmet edebilecek, temel aritmetik, mantıksal, giriş-çıkış ve bellek aktarım komutlarını gerçekleştirebilen bir işlemci VHDL kullanılarak gerçekleştirilmiştir. Gerçekleştirilen işlemci için komut seti oluşturulmuş ve oluşturulan komutlar ile yazılan deneme programları işlemci üzerinde başarılı bir şekilde koşturulmuştur.

Genel amaçlı işlemci tamamlandıktan sonra, AES algoritmasını gerçeklememize yardımcı olacak özel komutlar için var olan donanım yapılarının kaynak taraması yapılmıştır. Kullandığımız komutlar bu araştırma sonucunda, işlemcimizin yapısına en uygun olan, etkin ve kolay bir şekilde algoritmanın programlanmasını sağlayacak komutlar olarak seçilmiştir. AES için eklenen komutlar ile satır kaydırma işlemine gerek duyulmadan şifreleme ve şifre çözme işlemleri yapılabilmektedir. Bu da dördüncü kısımda anlatılan satır-sütun dönüştürme problemini ortadan kaldırabilmemizi sağlamaktadır.

AES algoritmasını gerçekleştirmek için oluşturulan komutlar işlemcinin komut listesinde bulunan boşluklara eklenmiş, yeni donanımlar da işlemci ile bütünleştirilmiştir. Oluşturulan bu işlemcinin genel amaçlı ve AES'e özel komutları birlikte kullanılarak örnek bir şifreleme işlemi için AES algoritmasını gerçekleştirecek program kodu yazılmış ve işlemci üzerinde koşturulmuştur. Yazılan program ile oluşturduğumuz uygulamaya özel işlemci üzerinde AES algoritmasını başarılı bir şekilde gerçekleştirebilmiş olduk. Yapılan sentezleme işlemi sonucunda elde edilen raporlar ile işlemcinin çalışma frekansının 78.812 MHz'e kadar çıkabileceği görülmüştür.

Oluşturulan bu işlemci, esnek ve basit bir yapıya sahip olmasından ve komut setinde başka komutlar için uygun boşluklar yer almasından dolayı geliştirilmeye açıktır. Farklı şifreleme algoritmalarını gerçekleyecek komutlar da tasarlanıp eklenerek işlemcinin kriptoloji alanı için işlevselliği artırılabilir.

KAYNAKLAR

[1] Tillich S. and Großschädl J., Instruction set extensions for efficient AES implementation on 32-bit processors. In CHES 2006, volume 4249 of LNCS, pages 270–284. Springer, 2006.

[2] Tillich S. and Großschädl J., Instruction Set Extensions for Efficient AES Implementation on 32-bit Processors. In L. Goubin and M. Matsui, editors, Workshop on Crypto-graphic Hardware and Embedded Systems CHES 2006, volume LNCS 4249, pages 270–284, Yokohama, Japan, October 10–13 2006. Springer-Verlag

[3] Moore, G., 1965. "Cramming more components onto integrated circuits", *Electronics* 38 (8). Retrieved 2009-12-23.

[4] Mano, M.M., 2007. Bilgisayar Sistemleri Mimarisi, Literatür Yayıncılık, İstanbul

[5] Bruce J., *The RISC-16 Instruction-Set Architecture*, University of Maryland, School of Engineering

[6] Dandamudi S., 2003. Fundamentals of Computer Organization and Design, Springer-Verlag, Heidelberg

[7] Buzluca F., "*Bilgisayar Mimarisi Ders Notları*", Bilgisayar Mühendisliği Bölümü, İTÜ Bilgisayar ve Bilişim Fakültesi

[8] FIPS 197, 2001. Advanced Encryption Standard, National Institute of Standarts and Technology.

[9] Tiryakioğlu O., 2008. Güvenli Telekonferans Sisteminin FPGA Üzerinde Gerçeklenmesi, *Lisans Tezi*, İTÜ Elektrik-Elektronik Fakültesi, İstanbul

[10] Tillich S., 2008, Instruction Set Extensions for Support of Cryptography on Embedded Systems, *PhD Thesis*, Graz University of Technology, Austria

[11] Pablo S., Cebrián J.A. and Rey A.B et al., 2006. A very simple 8 Bit RISC Processor for FPGA, *In FPGA World*, Stockholm, Sweden, May 27

ÖZGEÇMİŞ

22.10.1990 yılında İstanbul'un Fatih ilçesinde dünyaya gelen Onur Şahin 1996 yılında ilköğrenim hayatına adım attı. İlk ve ortaöğrenimini 60. Yıl Ataköy İlköğretim Okulu'nda tamamladıktan sonra 2004 yılında Ataköy Cumhuriyet Lisesi'nde lise eğitimine başladı. 2007 yılında bu liseden birincilikle mezun olduktan sonra İTÜ Kontrol Mühendisliği bölümünü kazandı. 2009 yılında Elektronik Mühendisliği bölümüne geçiş yaptı ve 2010 yılında da Bilgisayar Mühendisliği ile çift anadal yapmaya başladı. Temel ilgi alanını ise sayısal donanım tasarımı ve FPGA'lar oluşturmaktadır.