İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI DES KRİPTO ÇÖZÜCÜ SİSTEMİ

YÜKSEK LİSANS TEZİ

Müh. Bora EMİROĞLU

Anabilim Dalı : Disiplinler Arası Programlar
Programı : Savunma Teknolojileri

HAZİRAN 2005

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI DES KRİPTO ÇÖZÜCÜ SİSTEMİ

YÜKSEK LİSANS TEZİ

Müh. Bora EMİROĞLU
514011040

Tezin Enstitüye Verildiği Tarih : 9 Mayıs 2005
Tezin Savunulduğu Tarih : 2 Haziran 2005

Tez Danışmanı : Doç.Dr. A. Coşkun SÖNMEZ

Diğer Jüri Üyeleri : Prof.Dr. Eşref ADALI

Prof.Dr. Serhat ŞEKER

HAZİRAN 2005

ÖNSÖZ

Kriptoloji, sayısal enformasyon verilerini gizlemek için en temel mekanizmadır. Son yıllarda, enformasyon verilerinin yoğunluğunun artmasıyla güvenlik tehditlerine karşı hızlı ve güvenilir kriptografik algoritmalar geliştirilmiştir. Güvenlik uygulamalarında başarımı artırmak için yalnızca algoritmaların güvenilirlikleri yeterli olmayıp, bu algoritmaların yeterince hızlı olması da beklenmektedir.

Bu istekleri karşılayabilmek amacıyla kriptolojide geliştirilen veya üzerinde çalışılan çok sayıda konu bulunmaktadır. Tez kapsamında, bu konular içerisinde yer alan ve kriptolojide temel olarak nitelendirilen belli başlıları açıklanmaya çalışılmış, tekrardan programlanabilir (FPGA) bir üründe, Xilinx firmasına ait ISE-similasyon programıyla bir gerçekleme yapılmıştır.

Tez çalışmasının içeriğinin belirlenmesinde, geliştirilmesinde yardımlarını esirgemeyen danışman hocam sayın Doç. Dr. A.Coşkun Sönmez'e, ve TÜBİTAK-UEKAE (Ulusal Elektronik ve Kriptoloji Araştırma Enstitüsü)' de çalışma arkadaşlarım sayın Levent Ordu, Yener Ülker'e ilgi ve desteklerinden dolayı diğer tüm çalışma arkadaşlarıma, aileme ve gösterdiği hoşgörü ve destekten ötürü sevgili arkadaşım Şeyma Konya' ya teşekkürü bir borç bilirim.

Mayıs 2005 Müh. Bora Emiroğlu

İÇİNDEKİLER

KISALTMALAR

M	: Açık Metin
C	: Kapalı Metin
E	: Kapama İşlemi (Şifreleme)
D	: Açma İşlemi (Şifreyi çözme)
FPGA	: Field Programmable Gate Array
VHDL	: VHSIC Hardware Description Language
PAL	: Programable Array Logic
DES	: Data Encryption Standard
RSA	: Rivest, Shamir ve Adleman isimlerinin ilk harfleri
MD4	: Message Digest 4
CLB	: Configurable Logic Block
IOB	: Input Output Block
LUT	: Look Up Table
IP	: Initial Permutation
ROM	: Read Only Memory
RAM	: Random Acces Memory
3DES	: Triple DES

TABLO LİSTESİ

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

FPGA TABANLI DES KRİPTO ÇÖZÜCÜ SİSTEMİ

ÖZET

Haberleşme ve bilgisayar teknolojilerindeki gelişmelerle birlikte kriptografik uygulamalar ticari alanlarda da önem kazanmıştır. İnternet bankacılığı, e-posta hizmetleri, cep telefonları, uydu haberleşmeleri güvenli haberleşme hizmetlerinin beklendiği başlıca alanlardır. Kriptografi, sayısal enformasyon verisini korumak için kullanılan en temel mekanizmadır. Son yıllarda enformasyon verilerinin yoğunluğunun artmasıyla, güvenlik tehditlerine karşı koymak için güvenli ve hızlı kriptografik algoritmalar geliştirilmiştir.

Kriptografik algoritmaların yeniden programlanabilen donanımlar üzerinde gerçeklenmesi, yazılım ve VLSI platformlarına göre tasarlamaktan daha çok kolaylık sağlar. Yeniden programlanabilen donanımlar (FPGA) üzerinde gerçekleştirilen tasarımlar VLSI uygulaması kadar hızlı, ve bir yazılım uygulaması kadar da esnek olabilmektedir. VLSI uygulamaları çok hızlı çalışma kapasiteli ancak uygulamanın gerçekleşebilmesi için davranışsal tanımlamadan başlayıp devrenin bir entegre olana kadar tüm süre boyunca uzun bir zamana ihtiyaç vardır. Davranışsal tasarımdan sonra pahalı ve zaman alan bir fabrikasyon prosesi izlenmelidir. Tasarımın yazılım olarak gerçeklemeleri ise esneklik açısından kolay kullanılabilir ancak zaman faktörünün kritik olduğu durumlarda yüksek hızda çalışmadığı için uygun değildir. Diğer taraftan, VLSI tasarımlarındaki gibi fabrikasyon işlemleri olmadığından yeniden programlanabilme ve çoklu-yapılar üzerinde birçok kez deneme yapılabilmesi veya birçok revizyonun aynı yapı üzerinde denemeye olanak sağladığından FPGA uygulamaları büyük başarı gösterirler.

Bu çalışmada, özellikle yeniden programlanabilir donanım platformu için etkili ve kompakt DES yapısı sunulmuştur. Tasarım Spartan3s400-FPGA entegresi üzerinde gerçekleştirilmiştir. Tasarımda, tüm sekiz DES S-box ları hesaplamada paralel işleme yoluna gidilmiş ve bu yolla da algoritma hızına etki edecek kritik yollarda azalma sağlanmıştır. DES tasarımı 6624 Mbit/sn veri şifreleme/şifre çözme hızına ulaşmış ve entegre üzerinde 2207 tane CLB yer kaplamıştır. Burada elde edilen sonuçlar literatürde yayınlanmış diğer tasarımlarla yarışabilir düzeydedir.

FPGA BASED DES CRYPTO SYSTEM

SUMMARY

As the developments made in communication and computer technologies, cryptographic applications become more important in commercial markets. Internet banking, e-mail services, cellular phones, satellite communications are some of the areas that secure data transmission is expected.
Cryptography is the main mechanism to secure digital information data. In recent years due to the heavy increase in the volume of information data, secure and rapid cryptographic algorithms were developed to combat security threats.

Implementing cryptographic algorithms on reconfigurable hardware (FPGA) provides major benefits over VLSI (very large scale integrated circuits) and software platforms since they offer high speed similar to VLSI and high flexibility similar to software. VLSI implementations are fast but must be designed all the way from behavioral description to the physical layout. They have to follow an expensive and time consuming fabrication process. Software implementations offer high flexibility but they are not fast enough for the applications where time factor is vital. On the other hand, reconfigurable devices are attractive since the time and costs of VLSI design and fabrication can be reduced. Moreover, they offer high potential for reprogramming and experimenting on multiple architectures or several revisions of the same architecture.

In this study we present an efficient and compact DES architecture especially designed for reconfigurable hardware platforms. The design was implemented on a Spartan3s400 FPGA device. As a strategy to reduce the associated design critical path, we utilized a parallel structure that allowed us to compute all the eight DES S-boxes simultaneously. DES design achieved a data encryption/decryption rate of 6624 Mbits/s and occupying 2207 CLB slices. These result are quite competitive when compared with other reported DES implementations.

1. GİRİŞ

1.1. Giriş ve Çalışmanın Amacı

İnsanlık tarih boyunca yeni bilgilere erişmek için çalışırken, kendince önemli bulduklarını da belli kimselerle paylaşmak, diğerlerinden ise saklamak istemiştir. Kriptoloji (şifreleme) biliminin ortaya çıkması da bu paylaşma isteğinin altında yatmaktadır. Gerçektende böyle bir isteğin olmaması durumunda bilginin düşünceden somut dünyaya geçmesi beklenemez.

Kriptoloji, sayısal enformasyon verilerini gizlemek için en temel mekanizmadır. Bilinen en eski şifreleme yöntemlerinden olan Sezar şifrelemeden günümüze kadar uzanan çizgide kriptoloji bilimi kapalı kapılar arkasında yapılan çalışmalar olarak daha çok askeri ve diplomatik alanlarda kullanılmıştır. Bununla birlikte haberleşme ve bilgisayar teknolojilerindeki gelişmelerin etkisiyle gündelik hayatın bir parçası olan çoğu işlemlerin sanal dünyada gerçeklenebilir hale gelmesi kriptografik uygulamaların ticari alanlarda da yaygınlaşmasını sağlamıştır. Askeri uygulamalar, internet bankacılığı, e-posta hizmetleri, yerel alan sistemleri, uydu haberleşmeleri güvenli haberleşme hizmetlerinin beklendiği başlıca alanlardandır.

Kriptolojide verinin güvenliğinin sağlanmasında dört önemli özellik göze çarpmaktadır. Bunlar veri bütünlüğü, kimlik doğrulama, gizlilik ve reddedememe olarak sıralanmaktadır. Veri bütünlüğüyle gerçeklenmek istenen, göndericinin yolladığı bilginin herhangi bir değişikliğe uğramadan alıcı tarafa varmasıdır. Kimlik doğrulama, gönderici ve alıcının birbirlerinin kimliklerinden emin olmaları; gizlilik, gönderilen verinin üçüncü şahıslar tarafından anlaşılamaması; reddedememe ise gönderilen verinin sahibi tarafından inkar edilememesidir. Bu dört önemli özellikten

biri veya bir kaçının sağlanmaması durumunda söz konusu kripto sisteminin güvenliğinden söz edilemez.

Kriptoloji yukarıda tanımlı özellikleri sağlaması açısından kendi içerisinde çeşitli alt alanlara ayrılmıştır. Verinin gizliliğinin sağlanması çoğu zaman simetrik şifreleme algoritmalarıyla; anahtar değişimi, kimlik doğrulama gibi işlemler asimetrik şifreleme algoritmalarıyla, veri bütünlüğü ve reddedememe işlemleri hash fonksiyonları, sayısal imza algoritmaları ve sertifikasyon işlemleriyle sağlanmaktadır. Farklı kriptografik yapıların kullanılmalarının nedenleri olarak işlemlerin doğasından gelen farklılıklar, algoritmaların gerçeklenebilirlikleri, çalışma hızları gibi çeşitli özellikler gösterilebilir.

Son yıllarda enformasyon verilerinin yoğunluğunun artmasıyla güvenlik tehditlerine karşı hızlı ve güvenilir kriptografik algoritmalar geliştirilmiştir.
Güvenlik uygulamalarında başarımı artırmak için yalnızca algoritmaların güvenilirlikleri yeterli olmayıp, bu algoritmaların yeterince hızlı olması da beklenmektedir.

Bu tez kapsamında, literatürde çokça bilinen DES algoritmasının etkin ve kompakt tasarımı üzerinde durulmuştur. DES algoritması VHDL dili kullanılarak yeniden programlanabilir ürünler üzerinde (FPGA) üzerinde gerçeklenmiştir.
Elde edilen sonuçlar literatürde karşımıza gelebilecek sonuçlarla yarışabilir bir düzeyde olmuştur. Yapılan çalışmaların sonucu olarak, kriptografi içerisinde Brute Force Attack (Kaba Güç Atak Yöntemi) olarak bilinen yöntem ile DES algoritmasının en kısa sürede şifrelemede kullandığı anahtar tespit edilebilecektir.

Tezin 2. bölümünde tasarlanan algoritmanın gerçekleneceği Spartan3 serisi FPGA için önemli bilgiler verilmiş, 3. bölümde tasarım yöntemi olarak kullanılan VHDL dili tanımlanmış, 4. ve 5. bölümlerde literatürde karşımıza çıkan şifreleme algoritmaları çeşitleri incelenmiş, 6. bölümde DES algoritmasının ayrıntısına daha çok girilip 7. ve son bölümde bu yöntemle gerçeklenen DES algoritmalarının karşılaştırılması yapılmıştır.

2. SAYISAL ÜRÜNLER VE FPGA

2.1 Sayısal Ürünlerin Seçimi

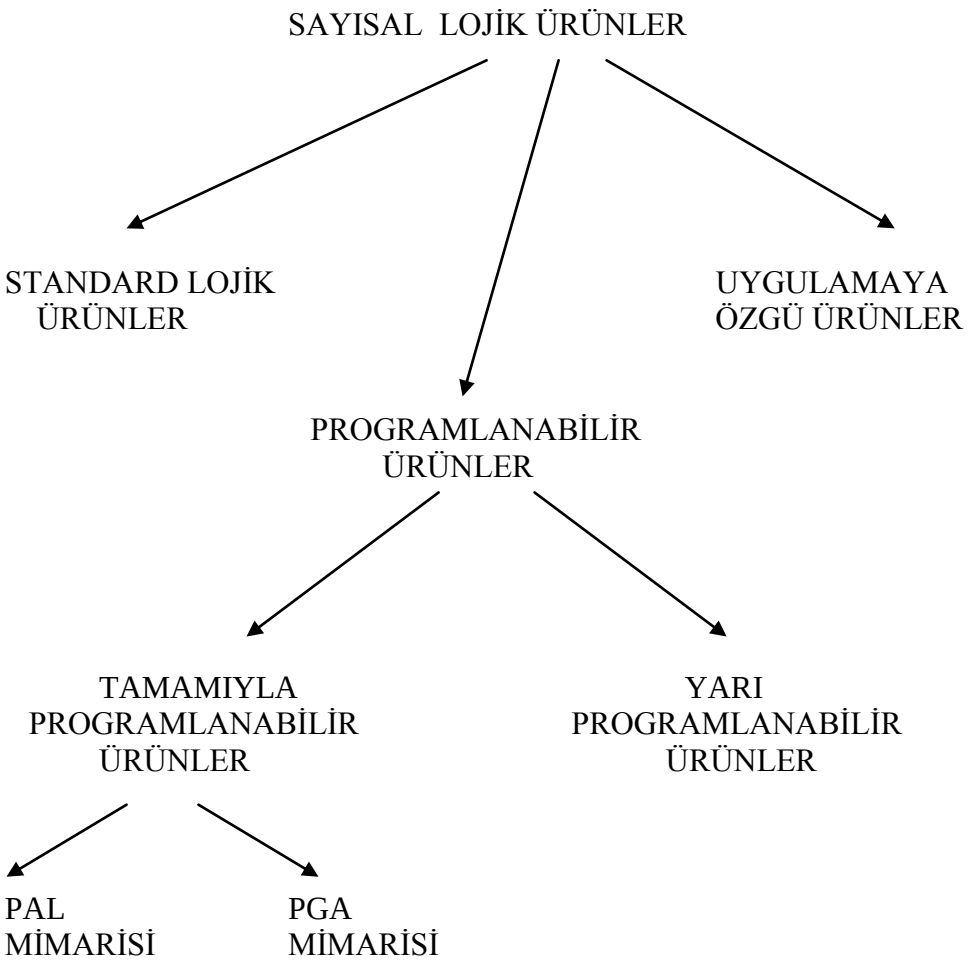
Son yıllarda iletişim teknolojisi, bilgisayar teknolojisi ve diğer teknolojilerde görülen büyük ilerleme, bu alanlarda kullanılan sayısal sistemlerin artması ve mevcut analog sistemlerin yerine sayısal karşılıklarının geçmesi ile mümkün olmuştur. Günümüzde bir sayısal sistem, çok değişik ürün gurupları ve bunlara ilişkin yöntemler ile tasarlanabilmektedir. Tasarımcı, kullanılacak ürünleri ve yöntemleri seçerken, bir takım tasarım kriterlerini göz önünde bulundurmalıdır. Önem seviyesi, tasarımın kullanım alanına göre değişen bu kriterlerden bazıları aşağıda sıralanmıştır:

- Tasarlanan sistemlerin kullanım ömrünün kısalması nedeniyle, sistem tasarımı için mümkün olan en az ön yatırım gereksinimi duyulmalıdır.
- Gerçeklenen ürünlerin mimari yapısı esnek olmalıdır. Yapılacak değişiklikler mümkün olan en az donanım ve yazılım değişimi ile karşılanabilmelidir. Yani sistem esnek yapılmalıdır.
- Teknolojinin hızlı bir biçimde ilerlemesi tasarlanan devrelerin kullanım ömrünü azaltmakta ve sistemlerin tasarım süresini kısaltmaktadır. Bu nedenle tasarım için kullanılacak yöntem mümkün olan en hızlı tasarım yöntemi olmalıdır.
- Bir tasarımın, eşdeğer tasarıma göre kıyaslama kriterlerinden en önemlisi birim zamanda harcanan güç tüketimidir. Güç tüketimi mümkün olduğunca az olmalıdır.
- Fiziksel boyut her türlü tasarım için önemli bir kavramdır. Tasarımı gerçeklenen devrenin fiziksel alanı mümkün olduğunca küçük olmalıdır.

- Tasarımı gerçeklenen devre optimum elemandan oluşmalıdır.
- Tasarımın en önemli başarı kriterlerinden biriside maliyettir. Tasarlanan devre minimum maliyette üretilebilmelidir.
- Tasarımı gerçeklenen devrenin kopyalanma bakımından güvenli olması gerekmektedir.

2.2. Sayısal Ürünlerin Sınıflandırılması

Sayısal ürünler genel olarak üç ana alanda sınıflandırılabilirler. Şekil 2.1 sayısal ürünlerin sınıflandırma yapısını gösterilmektedir.



Şekil 2.1 Sayısal Lojik Ürünlerin Sınıflandırılması

2.2.1. Standard Lojik Ürünler :

Bu yapılarda gerçeklenen ürün belirli bir amaç için tasarlanmıştır. Bu nedenle üretilen herhangi bir ürünün kısıtlı kullanım alanı mevcuttur. Ancak amaca uygun kullanımlarda kullanıcıya bazı avantajlar sağlar. Aynı fonksiyonu gerçeklemek için kullanılabilecek herhangi bir programlanabilir elemana göre, fiyat bazında önemli avantajlar sağlar. Ancak tasarımın esnekliğini zayıflatır. Tasarımda yapılacak değişimler genel olarak devrenin donanımında değişiklik yapmaya zorlar. Bunun yanı sıra gerçeklenmesi düşünülen bir fonksiyonu tek bir standard ürünle gerçeklemek genelde mümkün olmaz. Bu da tasarım donanımının büyümesine neden olur.

Özellikle lojik devre tasarımın ilk başlangıç dönemlerinde, entegre yapım teknolojisinin yetersizliğinden dolayı, programlanabilir elemanlar yeterli seviyede değildi ve fiyatları çok yüksekti. Bu başlangıç dönemlerinde tasarımı yapılan sistemler de çok güçlü fonksiyonlar olmadığından dolayı bu ürünler gereksinimleri karşılıyordu. Günümüzde bu katagori elemanları, kullanımı fonksiyonların karmaşıklaşması, yapım teknolojisinin hızlanması ve maliyetlerin düşmesi nedeniyle, yerini büyük oranda programlanabilir elemanlara bırakmıştır.

Standard ürünlerin kullanımının en büyük sakıncalarından biride tasarımın çok kolay bir biçimde kopyalanabilmesidir.

2.2.2. Uygulamaya Özgü Ürünler (ASIC)

ASIC ürünler genel olarak tasarıma özgü çözüme yönelik olarak tasarlanırlar. Standart tümdevreleri, baskılı devre kartları üzerinde birbirine bağlayarak sistemi oluşturmak yerine, komple sistemi yarı iletken üzerinde tasarlayıp, mümkünse tek tümdevreye yerleştirmek ana düşüncesi ASIC tasarımın ortaya çıkmasını sağlamıştır.

Tamamıyla özel ürünlerde tasarım, transistör seviyesinden başlayarak tasarlanır. Bu nedenle tasarım süresi ve dolayısıyla tasarım maliyeti en yüksek olan yöntemdir. Aynı zamanda ASIC kullanılarak tasarımı gerçeklenen devrelerin test süresi, diğer ürünlerin kullanılarak tasarlanan devrelere göre genelde daha fazla zaman alır. Bu da devre için harcanan gerçeklenme süresini artırır.

- ASIC tasarım yapılarak gerçeklenen devrelerin, istemdışı kopyalanma işlemi diğer tür yöntemler kullanılarak yapılan tasarımlara göre çok daha büyük oranlarda zorluklar taşır.

- ASIC tasarım yöntemiyle tasarlanan sistemler güvenlik olarak da diğer sistemlere göre daha üstündür. Devrenin tasarımının istem dışı olarak belirlenmesi çok daha zordur. Bu da tasarıma benzer özellikte bir başka sistemin, bu sistem üzerinden yapılması engellenmiş olur.

- ASIC tasarım, genel olarak devre boyutlarında büyük küçülmeler getirir.Bu tüm tasarım gücünün tasarımcı elinde olmasından kaynaklanır. Örneğin dışarı konulacak bir direncin sisteme getireceği boyut ile bu direncin çip içerisine konuşlandırılması durumundaki boyut farkı çok küçük bir örnektir. Ayrıca genel olarak yapılması düşünülen her tasarıma özgü tek bir ürün bulmak genelde mümkün değildir. Bu nedenle tasarımın gerçeklenmesi için birçok standard elemanın kullanılması zorunluluğunu getirir. Eleman sayısının fazlalığı sistemin işçilik maliyetini artırır.

- ASIC chip tasarımı işlemi, tasarlanacak sistemin sayısının az olması durumunda maliyet olarak, standard ürünlerin kullanılması durumuna göre daha pahalı bir bir çözümdür.

- ASIC tasarımı tamamiyle bilgisayar destekli yapıldıgından (CAD), aynı karmaşıklıkta bir devrenin standart katalok ürünleri ile tasarlanmasından çok daha kısa süre alır. Donanım Tanımlama dili (HDL) gibi yüksek seviyeli bir dil ile devrenin durum denklemleri, lojik fonksiyonları veya şematigini kagıt üzerinde belirleme zorunluluğu olmadan tasarım girişi yapılabilmektedir.

- Uzun geliştirme süresi ve yüksek geliştirme maliyeti ASIC ürünlerin en caydırıcı yönü olup, bu maliyeti düşürebilmek için yüksek miktarlarda üretim yapmak gerekir. Ayrıca tasarım mühendislerinin yanı sıra ASIC chip tasarımcılarına ihtiyaç duyulur. Bu da önemli bir maliyet artımı getirir.

2.2.3. Programlanabilir Ürünler

Programlanabilir ürünler genel olarak iki bölümde incelenirler.

2.2.3.1. Tamamıyla Programlanabilir Ürünler

Tamamıyla programlanabilir ürünler, günümüz tasarım yapılarında en çok kullanılan ürün pazarındandır. Tamamıyla programlanabilir ürünler mimari bakımdan iki ana yapıda düşünülebilir.

Bunlardan birincisi PAL mimarisidir.Bu mimaride, boole fonksiyonları çarpımların toplamı biçiminde, iki seviyeli VE-VEYA yapısındaki bir matris ile gerçeklenir.Giriş çıkış bacakları ve geri besleme yolları ve matrisinin kapılarının girişleridir. PAL mimarisinde her fonksiyon iki seviyeli bir gecikme ile gerçeklenir. Bu tasarımın hızı bakımından önemli bir değer olarak görülebilir. Ancak bu tür kullanım kapı sayısı bakımından verimsiz olmaktadır. PAL mimarisi ile tasarlanmış bazı önemli yapılar aşağıda belirtilmiştir.

- PROM yapıları (programmable Read Only Memory)

PROM yapıları, sadece bir kez programlanabilir yapılardır. Genel olarak, bir kez programlanabilme mantığı ile çalışan ürünlere OTP (one time programmable) adı verilir. PROM yapılarında ve matrisi sabit, VEYA matrisi ise programlanabildiği yapılardır. Bu yapılarda programlanabilme, bipolar sigortalar yardımıyla gerçekleşir.

- PAL (programmable Array Logic)

PAL'ler VE matrisinin programlanıp VEYA matrisinin sabit olduğu sigorta ile programlama mantığını kullanan mimaridir.

- PLA (programmable Logic Array)

PLA mimarisi, hem VE matrisinin hem de VEYA matrisinin programlanabildiği bir mimari yapıdır.

- EPLD (erasable programmable Logic device)

Genel olarak PAL mimarisinin gelişmiş yapısı olarak düşünülebilir. CMOS procesi ile üretilmiştir. Programlama işlemi UV ışını kullanımı ile birçok defa yapılabilir.

- EEPLD (electricaly erasable programmable Logic device)

EPLD yapısına benzer. Ancak silinme işlemi elektrik kullanılarak gerçekleştirilebilir.

Tamamıyla programlanabilir ürünlerin ikinci tür mimari yapısı ise PGA mimarisidir. Bu mimaride yapının merkezine mantık blokları yerleştirilmiş bunların etrafında ise fiziksel dünya ile bağlantıların kurulması amacına yönelik olarak giriş-çıkış blokları konulmuştur. Blok içi yada bloklar arası çeşitli bağlantılar kullanıcı tarafından oluşturulabilecek biçimde tasarlanmıştır.

PGA mimarisi çalışma mantığında, gerçeklenme için aktarılan bir 'boole' fonksiyonu, tek mantık bloğunun gerçekleyebileceği parçalara ayrılır. Bu parçaların birleştirilmesi ile gerçeklenmesi beklenilen fonksiyona ulaşılır. Burada fonksiyonun optimum biçimde verilmiş olması çok önemlidir. Çünkü devre gecikmeleri ve devrenin hızı bu fonksiyonun yapısına ve karmaşıklığına göre belirlenen parametrelerdir.

PGA'lere gerçeklenen devrenin yerleştirilmesi iki şekilde yapılır. Bunlar aşağıda açıklanmıştır:

– Ara bağlantıları SRAM hücreli olan yapılarda, devreye gerilimin verilmediği zamanlarda, kullanılan PGA bağlantıları boştadır. Gerilim uygulandığında ise, PGA bir konfigürasyon belleğinden, devrenin bağlantı parametrelerini alır ve PGA'i buna göre koşullandırır. Bu tür yapılarda dış donanımda, devrenin bağlantı parametrelerinin saklandığı bir bellek olmalıdır. PGA içerisinde de bu bellekten verileri alıp bağlantıları gerçekleyebilecek özellikte lojik devre olmalıdır.

SRAM hücreli PGA yapılarda devre kolaylıkla kopyalanabilir. Ancak programlama sayısı, belleğin programlanabilme sayısı kadardır. Buda tasarımda yapılacak değişikliklerin, çoğu zaman hafızada yapılacak değişikliklerle gerçeklenebilmesini sağlar.

- Ara bağlantıları 'antifuse' temelli olan yapılarda, PGA bir kez programlanır ve program parametrelerine göre bağlantılar gerçekleştirilir.Bu tür yapılarda dışarıda gerçekleme parametrelerini tutan bir belleğe ihtiyaç duyulmaz. Bu kopyalanma zorluğu sağlar. Ancak tasarımdaki PGA ile ilgili değişimler, yeni bir PGA'in programlanması zorunluluğuna neden olur.

Genel olarak proje geliştirme aşamalarında SRAM hücreli PGA'ler, Seri üretimlerde ise 'antifuse' temelli PGA'ler kullanılır.

2.2.3.2. Yarı programlanabilir ürünler

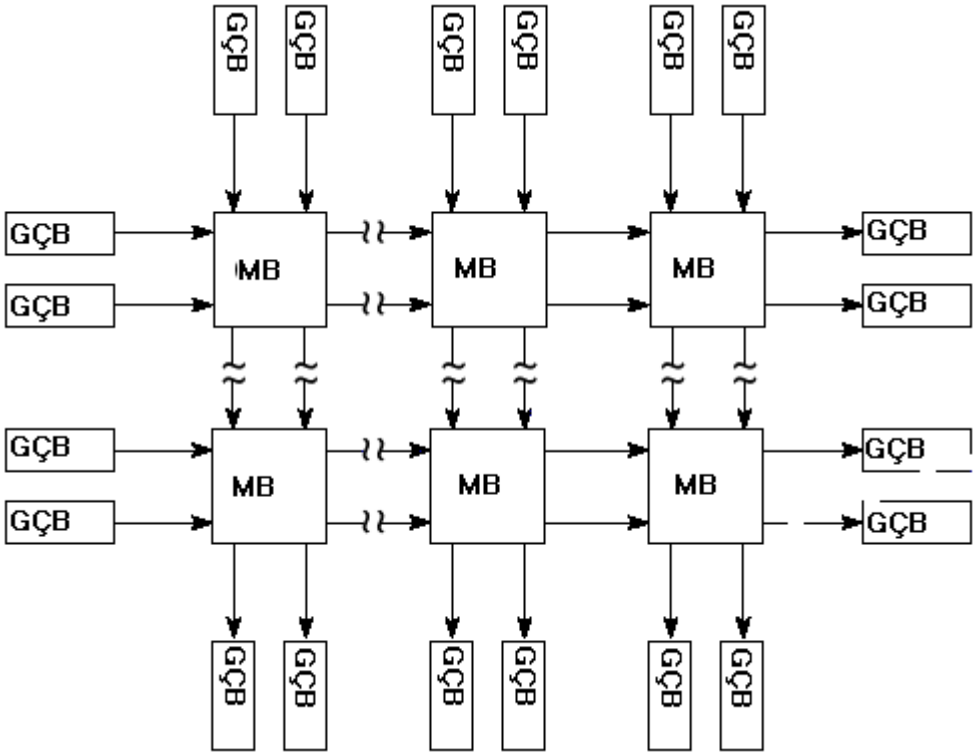
Yarı programlanabilir ürünler temel olarak tamamıyla programlanabilir ürünlerin yapısına benzerlik gösterir. Tek fark bu ürünlerde belli bölgelere kullanıcı isteklerine göre sabit tasarımlar yerleştirilmiştir. Bu bölgelerin giriş çıkış bağlantıları belirlenmiştir. Bu sayede kullanıcı sabit tasarım için ayrı bir donanım birimine ihtiyaç duymaz. Aynı zamanda tasarlanan devrenin güvenliği de artırılmış olur. Buna örnek olarak mikro işlemcinin ve bir tam programlanabilir PGA'in kullanılacağı bir

devrede, içerisinde mikro işlemci tasarımının yerleştirildiği bir birim bulunan yarı programlanabilir ürün kullanılabilir.

2.3. FPGA (Field Programmable Gate Arrays)

FPGA'ler yukarıda bahsi geçen tamamıyla programlanabilir lojik ürünler katagorisine giren yapılardır. FPGA'lerin elektronik pazarındaki payı özellikle yüksek kapı miktarlarına ulaşım ve kompleks fonksiyonların bu yapı içerisinde kolaylıkla gerçekleştirilebilmesi sayesinde hızlı bir biçimde artmaktadır. Ayrıca FPGA'ler kullanılarak tasarım süre bakımından da önemli avantajlar getirir.

Genel olarak FPGA, merkezinde MB'ler (mantık blokları) ve çıkışlarında ise GÇB'ler (giriş çıkış blokları) içeren bir yapıdadır. Şekil 2.2 bu yapıyı simgelemektedir.



Şekil 2.2 MB ve GÇB 'lerin Konumları

MB' ler, FPGA'in 'Boole' fonksiyonlarını gerçekleyen parçalarıdır. Her FPGA üreten firma bu blokları kendi teorilerine göre farklı yapılarda üretir. Örnek olarak XILINX firmasının, Spartan3s400 [2] serisi FPGA'lerinin MB yapıları aşağıda açıklanmaktadır.

2.3.1. Spartan3 FPGA Ailesi

2.3.1.1. Giriş:

Spartan-3 [1] FPGA ailesi yüksek yoğunluklu ve maliyetin önemli olduğu elektronik uygulamalarda tercih edilen bir entegredir. Bu ailenin 8 üyesinin sahip olduğu kapı sayısı 50,000 ile 5,000,000 arasında değişir. Spartan3 ailesi, Spartan-II-E ailesinin sahip olduğu lojik yapıların, RAM kapasitesinin ve I/O sayısının arttırılmasıyla oluşturulmuş bir yapıdır. Aşağıdaki tabloda Spartan ailesine ilişkin özellikler verilmiştir;

Tablo 2.1 Spartan Ailesi Teknik Özellikleri

Device	System Gates	Logic Cells	CLB Array (One CLB = Four Slices)			Distributed RAM (bits[1])	Block RAM (bits[1])	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs						
XC3S50	50K	1,728	16	12	192	12K	72K	4	2	124	56
XC3S200	200K	4,320	24	20	480	30K	216K	12	4	173	76
XC3S400	400K	8,064	32	28	896	56K	288K	16	4	264	116
XC3S1000	1M	17,280	48	40	1,920	120K	432K	24	4	391	175
XC3S1500	1.5M	29,952	64	52	3,328	208K	576K	32	4	487	221
XC3S2000	2M	46,080	80	64	5,120	320K	720K	40	4	565	270
XC3S4000	4M	62,208	96	72	6,912	432K	1,728K	96	4	712	312
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	784	344

Özellikler:

- 90nm proses teknolojisi
- Tüketici tabanlı uygulamalar için yüksek performans/uyun fiyat özelliği

-74880 adet lojik hücre (slice)
-326 MHz sistem saat hızı
-3 farklı gerilim seviyesi (core için 1.2V, I/O lar için 1.2V ‘dan 3.3V’ a ve ekstra özellikler için 2.5 V)

- I/O seçimli sinyalizasyon
  -784 I/O pini
  -I/O başına 622 Mb/sn veri transfer hızı
  -Double Data Rate (DDR) desteği
  -1.14’V dan 3.45V’a kadar sinyal seviyesi yelpazesi

- Lojik Kaynaklar
  -Geniş veri seçiciler (MUX)
  -Yerleşik 18x18 ‘ lik çarpıcı
  -Kayan yazmaç (Shift Register) yapılı birçok lojik hücre

- Hiyerarşik RAM yapısı
  -1872 Kbit e kadar toplam blok RAM
  -520Kbit e kadar dağınık RAM

- Dijital Saat Yönetimi (DCM)
  -Saat eğimi (Clock Skew) önlemi
  -Frekans Sentezleme
- 8 tane saat hattı
- PCI veya diğer core lar ile kolay bağlantı.

2.3.1.2. Spartan-3 Mimarisi

Spartan3 ailesi programlanabilir 5 ana fonksiyondan oluşur:

- Konfigüre edilebilir Lojik Bloklar [1] (CLB), RAM tabanlı lojik ve hafıza ürünleri olarak gerçeklenebilecek veya flip-flop veya latch olarak kullanılabilecek LUT (Look up Table) ları içerir.

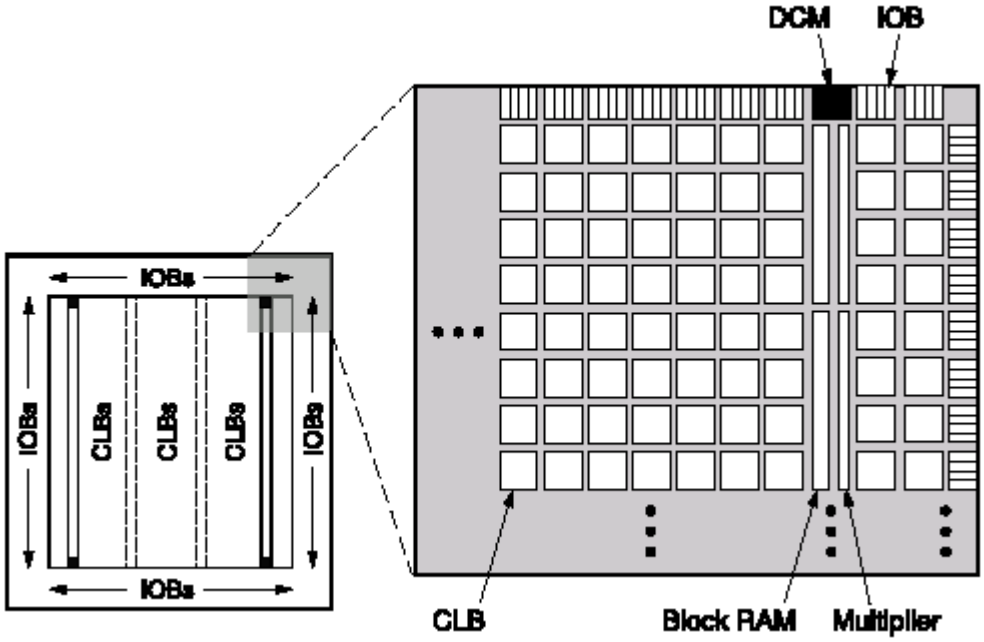
- Giriş/Çıkış Blokları (I/OB), I/O pinleri ile içerdeki lojik arasında veri akışını sağlar. Her I/OB çift yönlü veri akışını destekler ve buna ek olarak da 3. duruma da geçebilir. 24 farklı işaret standardı ve buna ek 7 adet yüksek hızlı farksal standartı da Tablo 2.2 de gösterildiği gibi destekler. DDR destekli yazmaçlar da bulunmaktadır.

Tablo 2.2 Spartan3 Ailesinin Desteklediği Sinyal Türleri

Standard Category	Description	V_{CCO} (V)	Class	Symbol	DCI Option
Single-Ended					
GTL	Gunning Transceiver Logic	N/A	Terminated	GTL	Yes
			Plus	GTLP	Yes
HSTL	High-Speed Transceiver Logic	1.5	I	HSTL_I	Yes
			III	HSTL_III	Yes
		1.8	I	HSTL_I_18	Yes
			II	HSTL_II_18	Yes
			III	HSTL_III_18	Yes
LVCMOS	Low-Voltage CMOS	1.2	N/A	LVCMOS12	No
		1.5	N/A	LVCMOS15	Yes
		1.8	N/A	LVCMOS18	Yes
		2.5	N/A	LVCMOS25	Yes
		3.3	N/A	LVCMOS33	Yes
LVTTL	Low-Voltage Transistor-Transistor Logic	3.3	N/A	LVTTL	No
PCI	Peripheral Component Interconnect	3.0	33 MHz	PCI33_3	No
SSTL	Stub Series Terminated Logic	1.8	N/A	SSTL18_I	Yes
		2.5	I	SSTL2_I	Yes
			II	SSTL2_II	Yes
Differential					
LDT (ULVDS)	Lightning Data Transport (HyperTransport™)	2.5	N/A	LDT_25	No
LVDS	Low-Voltage Differential Signaling		Standard	LVDS_25	Yes
			Bus	BLVDS_25	No
			Extended Mode	LVDSEXT_25	Yes
LVPECL	Low-Voltage Positive Emitter-Coupled Logic	2.5	N/A	LVPECL_25	No
RSDS	Reduced-Swing Differential Signaling	2.5	N/A	RSDS_25	No

- Blok-RAM ler, 18Kbit boyunda ve çift-giriş/çıkış (dual port) özelliği ile veri depolar.
- Çarpma işlemi yapan blok,18 bitlik iki ikili (binary) sayıyı çarpma özelliğine sahiptir.
- Dijital Saat Yönetimi (DCM) bloğu, tamemen dijital çözümü olarak saat işaretini böler, çarpar, geciktirir ve fazını kaydırır.

Mimari içinde yer alan böümler aşağıdaki şekilde görülmektedir:



Şekil 2.3 Spartan3 FPGA İç Yapısı

Görüldüğü üzere I/OB ler CLB lerin etrafını sarmaktadır. XC3S50 [1] entegresi 1 tane blok RAM' e sahiptir. XC3S200 [1] den XC3S200 [1]0 e kadar olan entegrelerde 2 tane blok RAM vardır. XC3S4000 [1] ve XC3S5000 [1] entegreleri ise 4 tane RAM kolonuna sahiptir. DCM ler RAM bloklarınını çıkış taraflarına konumlandırılmıştır.

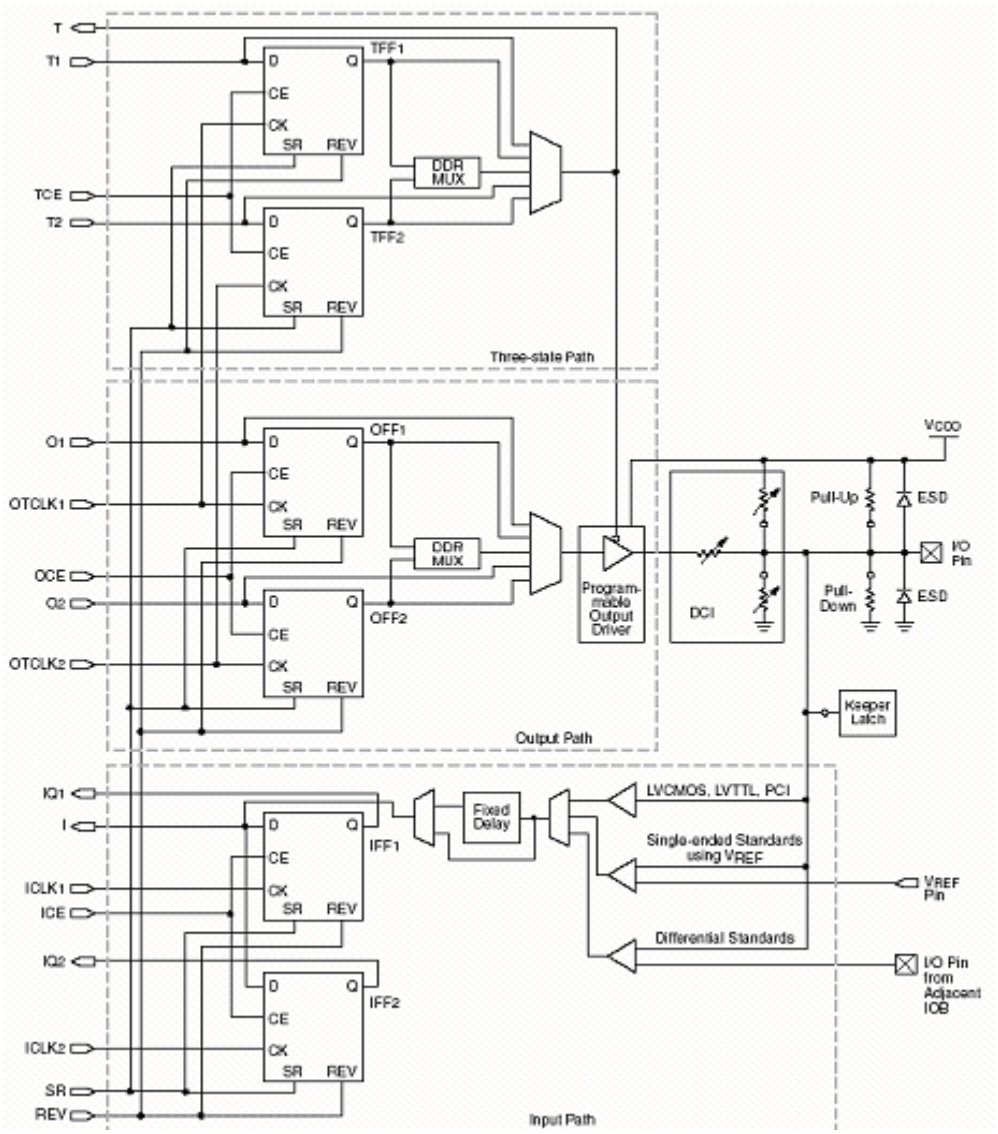
2.3.1.3. Konfigürasyon:

Spartan3-FPGA ailesinin, konfigürasyon verileri statik hafıza hücresine (PROM) yüklenerek tüm fonksiyonel elemanları kontrol edilebilinir. Devreye güç verilmeden önce konfigürasyon verileri dışarda uçucu olmayan bir bellekte tutulur, devreye enerji verildikten sonra bu konfigürasyon verileri FPGA‘ e yazılarak çeşitli modlarda FPGA' in çalışmasını sağlar.

2.3.1.4. IOB'lere Genel Bakış:

Giriş/Çıkış Blokları (IOB), FPGA içersindeki yapı ile I/O pinleri arasıdaki çift yönlü programlanabilir arabağlantı sağlar.

Aşağıdaki şekilde IOB' lerin iç yapısını göstermektedir.



Şekil 2.4 Spartan3 FPGA IOB Yapısı

Şekilde de görüldüğü üzere IOB ler içinde 3 ana sinyal yolu bulunmaktadır; bunlar çıkış yolu, giriş yolu ve 3. durum yoludur. Tüm yolların kendilerine ait saklama elemanları bulunmaktadır.

Bu 3 ana sinyal yolunu daha ayrıntılı inceleyecek olursak:

- Giriş yolu, entegrenin dışardaki bacaklarına bağlıdır ve veriyi bu yol üzerinden taşır. Veya opsiyonel olarak I hattı üzerinde yer alan programalanabilir gecikme elemanından da geçirebilir. Gecikme elemanından sonra IQ1 ve IQ2 hatlarına bir çift saklama elemanından sonrada alternatif yollar bulunmaktadır. IOB çıkışları olan I, IQ1, ve IQ2, FPGA in içindeki lojik yapıya doğru giden hattır.

- O1 ve O2 hatlarıyla başlayan çıkış yolları, FPGA' in içindeki yapıdan verileri taşıyarak veri seçici (multiplexer) ve 3 durumlu sürücü üzerinden IOB bağlantısına götürür. Bu direkt hatta ek olarak, yapıda yer alan veri seçici ile saklama elemanıda ek olarak sistemde düşünülebilir.

- Çıkış sürücüsü 3. durumda olması gerektiğinde 3.durum olarak bu yol yüksek empedans durumuna geçer. T1 ve T2 hatları, FPGA' in içinde yer alan devreden veri seçici ile çıkış sürücüsüne veri taşır.

- IOB ' ye giren tüm sinyallerin,buna saklama elemanlarıda dahil, evirme (invert) özelliğide mevcuttur.

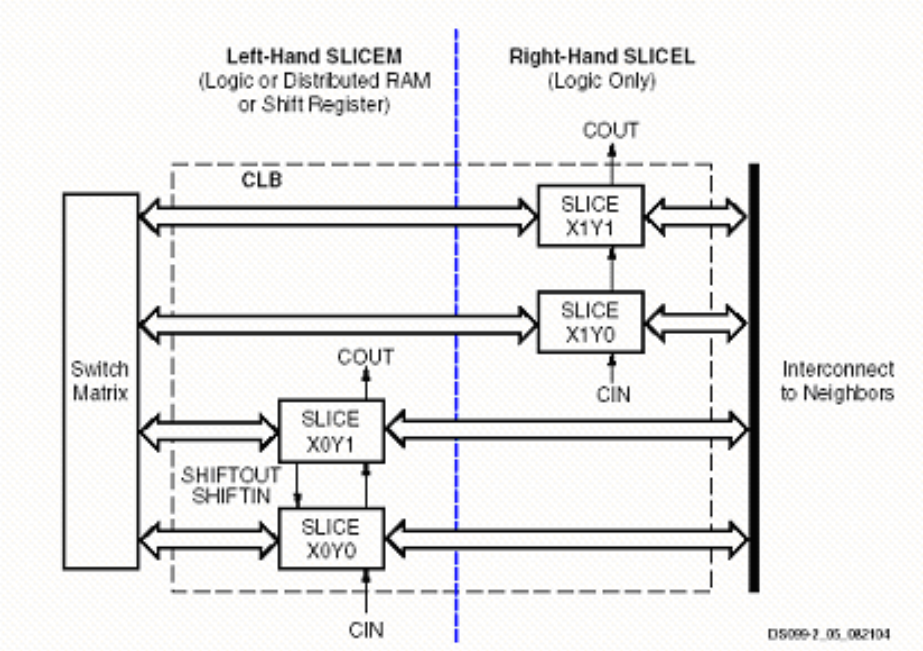
IOB içindeki Saklama Elemanları:

Her IOB de 3 adet saklamam elemanı mevcuttur. Her bir saklama elemanı istendiği zaman kenar tetiklemeli bir D-tipi flip-flop istendiğinde de seviye sezmeli bir latch olarak konfigüre edilebilinir.
Bu saklama elemanları (çıkış yolu veya 3. durum yolu saklayıcıları) özel bir veri seçici ile birlikte kullanılması durumunda DDR (Double Data Rate) iletim özelliğine de sahip olabilir.

Bunu gerçekleştirmek için önce veri, saat işaretinin yükselen kenarına senkronize edilir daha sonra da saat darbesinin hem yükselen hemde düşen kenarına denk getirilerek verinin değişmesi sağlanır.

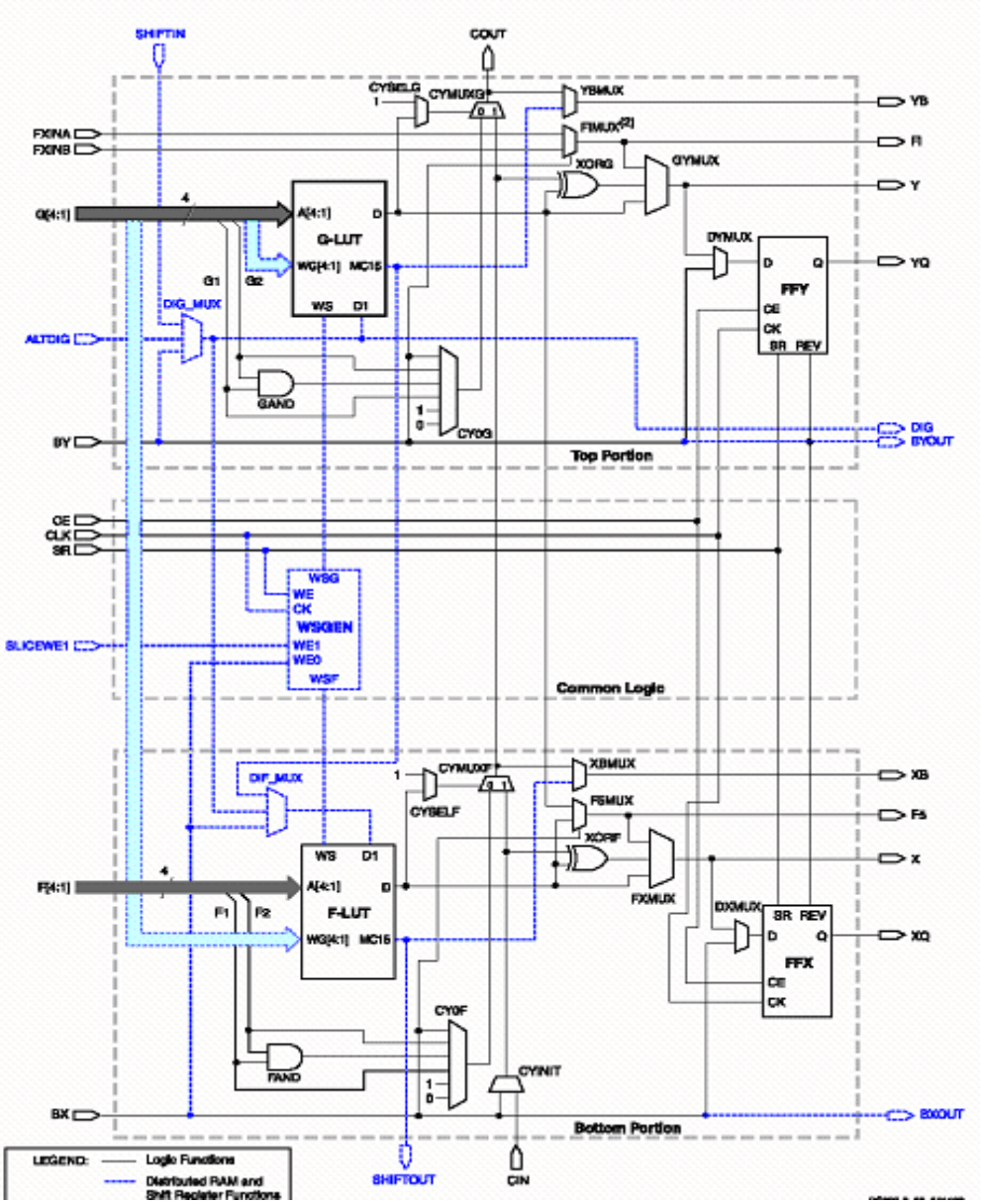
2.3.1.5. CLB (Configurable Logic Block) Yapısı:

CLB ler kombinazonsal ve senkron devre gerçeklemek için gerekli lojik kaynağı oluşturur. Her CLB, Şekil 2.5‘ de görüldüğü gibi 4 tane içerden bağlı slice adı verilen dilimlerden oluşmuştur.



Şekil 2.5 Spartan3 FPGA CLB Yapısı

Bu slice lar çiftler halinde gruplanmışlardır. Bu 4 slice içerisinde yer alan elemanlar herbirinde özdeştir; 2 tane lojik fonksiyon üreteçleri, 2 tane saklama elemanı, esnek özellikli veri seçiciler, elde lojiği ve aritmetik kapılardır. Bunlar Şekil 2.6’ da gösterilmiştir.



Şekil 2.6 Spartan3 FPGA Slice Yapısı

Hem sol hemde sağ slice çift elemanları kullanarak lojik, artimetik ve ROM fonksiyonlarını yerine getirir. Bundan başka sol tarafda yer alan slice çifti 2 ek özelliğe daha sahiptir. Bu slice lar dağıtılmış yapı içindeki RAM leri kullanarak verileri saklar ve 16 bit yazmaçları ile verileri kaydırma işleminide gerçekleştirebilir. RAM tabanlı fonksiyon üreteçleri (aynı zamanda bu üreteçlere LUT-Look up Table da denir) lojik fonksiyonları gerçeklemek için gerekli yapıdır. Ayrıca sol da yer alan slice ın içindeki LUT lar konfigüre edilerek dağıtılmış RAM veya 16 bitlik kaydırmalı yazmaç elde edilir.

D tipi flip-flop veya Latch olarak programlanabilen saklama elemanları, veriyi saat işaretine senkronize etmek ve diğer işlemler için kullanılır.
Esnek kullanımlı veri seçiciler LUT ları birbirine bağlayarak daha kompleks lojik işlemler gerçeklemeyi sağlar. Her slice da bu veri seçicilerden 2 tane bulunur.

Fonksiyon Üreteçleri:

Slice da yer alan her iki LUT (F ve G) 4 lojik girişe (A1-A4) ve 1 lojik çıkışa (D) sahiptir. Bu girişler ile herhangi 4 değişkenli bir Boole fonksiyonu LUT ların içine yerleştirilebilinir. Bundan başka slice içinde yer alan veri seçiciler ile aynı CLB içinde veya farklı CLB ler arasında birleşim gerçekleştirilerek lojik fonksiyonları daha fazla girişli hale getirebiliriz.
Slice da yer alan LUT lar lojik fonksiyonları gerçekleştirmenin haricinde ROM olarak da kullanılabilinir.
Sol slice çiftinde CLB içinde yer alan LUT ların fazladan 2 özelliği bulunmaktadır.

Bunlardan biricisi, bu CLB dağıtılmış RAM olarak programlanabilir. Bu tarz hafıza birimi veri yolu boyunca veri tamponlama özelliğine sahiptir. Sol tarafdaki bir LUT 16 bit saklayabilir. Sol tarafda yer alan LUT lardan bir çoğu birbirleriyle bağlanarak daha fazla yoğunlukta veri saklayabilir. Çift port opsiyonu da iki LUT u birbirinie bağlayarak, hafıza birimine erişimi 2 bağımsız hattan yapılmasına olanak verir.

İkinci özellik, sol tarafda yer alan LUT' u bir kaydırmalı saklayıcı olarak programlamak mümkün olacaktır. Bu yolla, her LUT seri veriyi 1 den 16 saat darbesine kadar geciktirebilecektir. Tek CLB içinde yer alan sol yandaki 4 LUT birleştirilerek 64 saat darbesi boyunca kaydırma işlemi gerçekleştirilebilecektir.

2.3.1.6. Blok RAM' ler:

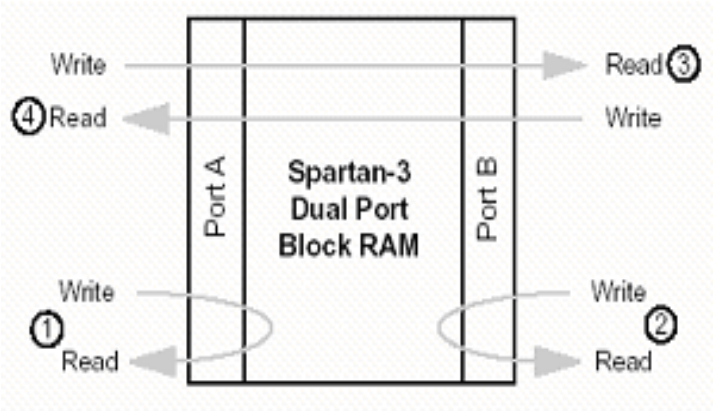
Tüm Spartan3 ailesi senkron 18Kbitlik bloklar halinde konfigüre edilebilir blok RAM yapılarına sahiptir. Blok RAM yapıları daha önceden açıklanan dağınık Ram yapılarına göre daha yüksek veri saklama özelliğine sahiptir.

Her blok RAM' in derinliği ve genişliği konfigüre edilebilir. Bundan başka birden çok blok da birleştirilerek daha büyük RAM boyutları elde edilebilir.

Blok RAM 'in İç Yapısı:

Blok RAM çift yönlü yapıya sahiptir. A ve B isimli iki eş veri portu birbirinden bağımsız olarak RAM bloğuna erişim imkanı sağlar. Bu RAM' in maksimum kapasitesi 18.432 bit (veya 16384 bit parite kullanılmadığında) olabilir.
Her portun kendine özel veri, kontrol ve senkron yazma/okuma işlemleri için saat hattına sahiptir.

Aşağıdaki şekilde görüldüğü üzere 4 temel veri yolu bulunmaktadır:



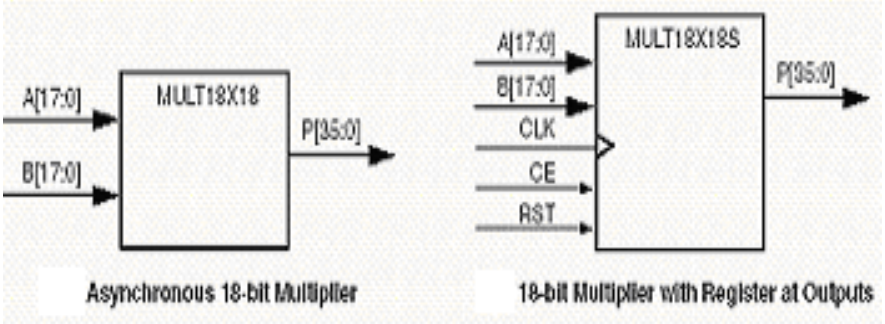
Şekil 2.7 Spartan3 FPGA Blok RAM Yapısı

(1) Port A ya yazma ve okuma işlemleri için,
(2) Port B ye yazma ve okuma işlmeleri için,
(3) Port A' dan Port B' ye veri transferi için,
(4) Prot b'den Port A' ya veri tranferi için kullanılır.

2.3.1.7. Spartan3 Çarpma Bloğu :

Tüm Spartan3 entegrelerinde içersine gömülü bir çarpma devresi yer almaktadır. Bu devre 2 tane 18 bitlik girişi bulunan ve sonuç olarak da 36 bitlik sonuç üretebilen bir yapıdadır.
Çarpma devresinin girişleri 2 ye tümleyen olarak verileri değerlendirir (18 bit işaretli veya 18 bit işaretsiz olarak). Kırmık üzerinde (Spartan3) her çarpma bloğu bir blok RAM' e karşılık düşer.

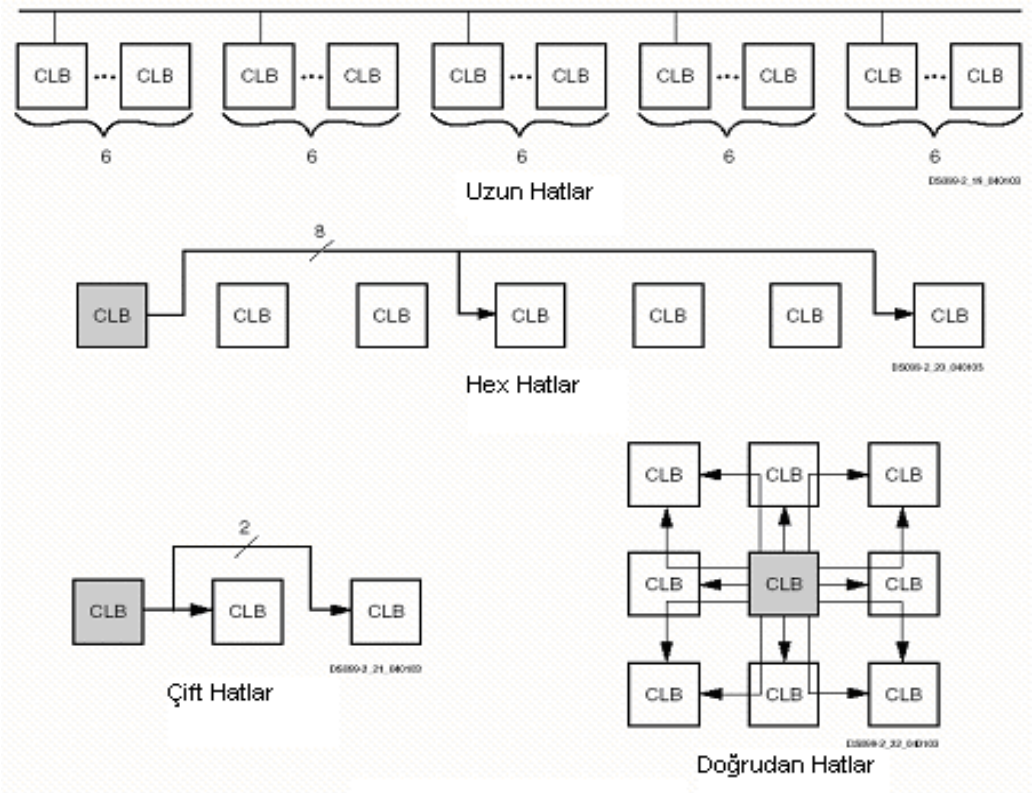
Oluşturulabilecek çarmpıcılar 2 şekilde olabilir. Bunlardan birincisi asenkron olan çarpıcı MULT18x18 (Şekil 2.8) ve diğeride çıkışında bir yazmaç olan çarpıcı MULT18x18S (Şekil 2.8)' dir.



Şekil 2.8 Spartan3 FPGA Çarpma Bloğu

2.3.1.8. CLB-ler Arası Bağlantılar :

Arabağlantılar, devre içindeki sinyalleri Spartan3 içindeki fonksiyonel elemanlar arasından geçiren yapılardır. 4 farklı tip arabağlantı vardır. Bunlar Uzun hatlar, Hex hatlar, Çift hatlar ve Doğrudan hatlar olarak isimlendirilir.
Bu hatların görünümleri aşağıdaki şekilde yer almaktadır :



Şekil 2.9 Spartan3 FPGA Arabağlantı Hatları

3. VHDL DONANIM TASARIM DİLİ

3.1. Giriş

Lojik devre tasarımı başlangıç olarak logic kapılar seviyesinde yapıldı. Ancak tasarımların karmaşıklaşması ve bunların lojik kapılar kullanılarak gerçeklenme zorluğu nedeniyle çeşitli yazılım dilleri tasarlanmaya başlandı. Bu dillere genel olarak HDL (hardware description languages) adı verilir. Bu şekilde tasarımcılara sunulan birçok dil mevcuttur. Bu dillerden en çok kullanılanı VHDL yazılım dilidir. Bu ya zılım dili kullanılarak her türlü logic sistemler yazılım dilinin kurallarına uygun olarak tasarlanabilir. Tasarlanan sistem istenilen işlevi tam anlamıyla yerine getirdiğini yada getirmediğini anlamak için yazılım programı içerisinde belirli test bölümleri oluşturulmuştur. Bu sayede sistem hardware yapısı kurulmadan bilgisayar ortamında test edilebilir. Bu da tasarım süresinin azalmasını sağlar.

3.2. VHDL Dili ve Donanım Tasarımı Karşılaştırılması

VHDL dili yada diğer HDL dilleri kullanılarak tasarım yapılmasının standart tasarım yöntemine göre bazı önemli üstünlükleri vardır. Bu üstünlükler yapılan tasarıma özgü olabildiği gibi genel olarak ifade etmekte mümkündür. Aşağıda bazı önemli noktalar ifade edilmiştir.

- **Tasarım süresi :**

Teknolojinin hızla gelişimi beraberinde kullanılan bir devrenin kullanım ömrü azalmaktadır. Bu da devrenin tasarım zamanının kısıtlanması anlamına gelir. Bu

noktada devrenin fiziksel olarak kapladığı alanın yada optimum tasarım olmasının ötesinde tasarım süresinin kısalığı ön plana çıkar. VHDL dili yada diğer HDL dili kullanılarak tasarım yapılması özellikle bu noktada tasarımcıya önemli artılar getirir. HDL dilleri doğrudan hardware tasarımına göre tasarım süresi bazında çok daha kısa sürede gerçeklenebilir. Ayrıca gerçeklenen devrenin kontrolünün yapılması anlamında da önemli üstünlükler sunar.

- **Tasarım Esnekliği :**

Teknolojideki değişimle birlikte kullanılan elemanların da yapıları da değişmektedir. Bu yapı değişiklerinin daha önce yapılmış tasarımlarda da çalışabilmesi için kullanılan tasarım ortamının buna uygun olabilmesi gerekir. Bu noktada HDL dilleri önemli üstünlükler sağlar . HDL dilleri genel olarak fonksiyon bağımlı yapılardır. Dönüştürücü proğramlar yardımıyla hardware tasarım şekline dönüştürülürler. Teknoloji değişimleri durumlarında sadece bu dönüştürücü programların bu yapıya uygun hale getirilmiş olması yeterli olacaktır.

- **Tasarım Kolaylığı :**

Genel olarak HDL dili kullanarak yapılan tasarım işlemlerinde hardware bilgisi gereksinimi, klasik donanım tasarımına göre daha az bir şekilde ihtiyaç duyulur.

- **Yenileme kolaylığı :**

Teknolojideki hızlı değişim beraberinde kulanılan tasarımlarda da hızlı değişimler getirir. Bu nedenle yapılacak değişimlerin hızlı bir şekilde gerçekleştirilebilmesi gerekmektedir. Bu noktada HDL dilleri doğrudan hardware tasarımına göre önemli bir üstünlük sunar. HDL dili ile yapılan tasarımlarda değişim işlemi tasarımcının yazılım gücü ile sınırlıdır.

3.3. VHDL DİLİ MİMARİ YAPILARI

VHDL dili 3 mimari yapıdan oluşur .

3.3.1. Davranışsal Mimari

Davranışsal mimaride sistemin yapması gereken işlemler ‘process’ ler kullanılarak, bir bilgisayar yazılımı biçiminde bir tasarım yolu izlenir. Ancak tasarımın nasıl gerçekleneceği konusunda bilgi verilmez. Bu nedenle genelde program da yapılan küçük değişimler büyük farklara neden olabilir. Bunun için yazım yapılırken dikkat edilmeli ve yazımda sürekli olarak hardware düşünülmelidir.

Davranışsal mimaride process ile birlikte duyarlılık ifade eden parametreler parantez içinde yazılır. Bu sayede process içinde bulunan bu duyarlılık ifade eden parametrelerden herhangi birinde bir değişim olması durumunda process yukarıdan aşağı dogru gerçekleştirilir. Bir mimaride tek process olmak zorunluluğu yoktur. Bu tür yapılarda tüm processler aynı anda gerçeklenir ancak yukarıda da bahsedildiği gibi process içinde ki işlemler yukarıdan aşağı doğru gerçeklenir.

3.3.2. Veri Akışı Mimarisi

Veri akışı mimarisinde devre tasarımı; karşılaştırıcılar, toplayıcılar, kod cözücüler ve basit logic kapılar kullanılarak tasarlanır. Bu yapıda kullanılan satırlar tamamen eşzamanlı olarak çalışır. Bu tasarım yapısında duyarlılık ifade edilen sinyaller yazılan satırlarda ki eşitliklerin sağ tarafında kalanlardır.

Bu tür mimaride davranışsal mimariye göre dışarıdan bakıldığında yapılacak işlemin anlaşılırlılığı daha azdır. Aynı zamanda yapılacak işlemin gücü büyüdükçe her fonksiyonu standard elemanlarla göstermek, yazılım karmaşıklılığını artıracak ve sonuca ulaşmak zorlaşacaktır. Bu nedenle büyük işlemli tasarımlarda, tasarımı standard elemanlarla ifade etme zorluğu nedeniyle, genelde kullanıcı davranışsal mimariyi tercih eder. Ancak davranışsal mimariye göre daha kontrollüdür.

3.3.3. Yapısal Mimari

Yapısal mimari temel olarak tamamen kullanıcının kontrolündeki bir tasarım biçimidir. Bu tasarımda tüm bağlantılar yazılımcı tarafından tanımlanır ve işaret isimleri verilerek belirlenir. Ayrıca kullanılacak elemanlar yazılımcı tarafından 'component' olarak ifade edilir ve tasarımda kullanılır.

Bu tasarım biçimi bir şekilde yapılacak hardware çiziminin yazılarak yapılması olarak görülebilir. Bu nedenle sistem karmaşıklığı arttıkça bu tasarımın kullanımı çok zorlaşacaktır. Küçük projelerde yada genelde aynı yapıların tekrar ettiği sistemlerde kullanılabilir. En büyük avantajı tasarımın kaplayacağı alan kullanıcı kontrolü altında olmasıdır.

3.4. VHDL Dili Temelleri ve Yazılım Özellikleri

Bu bölümde VHDL dilinin bazı önemli özellikleri ve temel kullanım kuralları gösterilecektir.

3.4.1. Yapısal ve Davranışsal Tanımlamalar

VHDL dilinde component tanımlamaları design entity birimleri içerisinde gerçekleştirilir. Bir component gerçeklemesi iki bölümden oluşur. Birinci bölümde component'in entity'si bulunur. Bu bölümde component'in hangi giriş çıkış uçlarından oluşacağı bildirimi yapılır. Örnek olarak bir yarı_toplayıcı' nın entity yapısı şu şekilde ifade edilir.

```
entity yarı_toplayıcı is
port (
        giriş1 : in Bit ;
        giriş2 : in Bit ;
        toplam : out Bit ;
        elde : out Bit  );
end yarı_toplayıcı ;
```

Burada koyu olarak yazılı ifadeler VHDL dili kullanım kelimeleridir. VHDL dili bu yapı ile kendi yapısı içerisinde Şekil 3.1 ‘i oluşturur.

Şekil 3.1 Yarı Toplayıcı Bloğu

İkinci bölümde ise component’in yapısı ifade edilir. Örnek yarı toplayıcıya ilişkin davranışsal tanımlama bölümü aşağıdaki gibi yapılabilir.

```
architecture  davranışsal_tanımlama of yarı_toplayıcı is
begin
process
        toplam <= giriş1 xor giriş2 after 10 Ns ;
        elde <= giriş1 and giriş2 after 10 Ns ;
wait on giriş1 , giriş2 ;
end process ;
end davranışsal_tanımlama ;
```

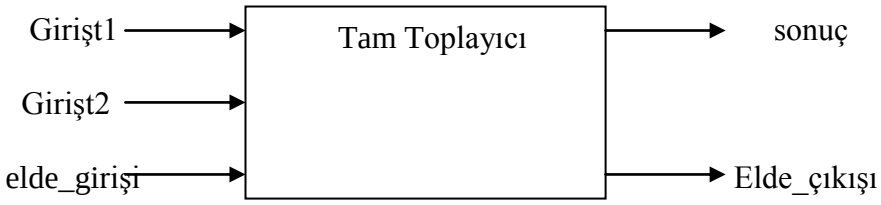
Bu yapı içerisinde kullanılan **after** ifadeleri, gerçeklenmelerde herhangi bir etki yapmaz. Sadece simülasyonlarda gecikme amacıyla kullanılırlar.

Architecture bölümününde gerçeklenmesi ile yarı toplayıcı bir component olarak oluşturulmuştur. Bu component’ lerin başka bir yapı tarafından kullanılmaları durumunda ifade edilen **port** yapıları yeni sitem için, sistem içi işaretleri gösterir. VHDL dilinde **signal** olarak ifade edilirler. Aşağıda bununla ilgili olarak bir tam toplayıcı örneği verilmiştir.

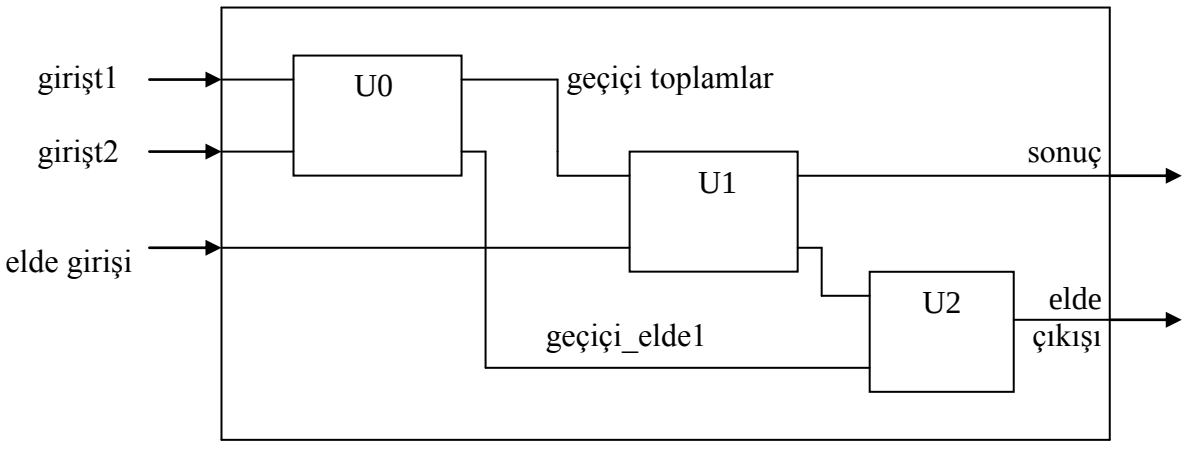
```
entity tam_toplayıcı is
port (
        girişt1 : in bit ; girişt2 : in bit ; elde_girişi : in bit ;
        sonuç : out bit ; elde_çıkışı : out bit ) ;
end tam_toplayıcı ;

architecture yapı of tam_toplayıcı is

        signal geçiçi_toplam : Bit ;
        signal geçiçi_elde1 : Bit ;
        signal geçiçi_elde2 : Bit ;

component yarı_toplayıcı
port (
        giriş1 : in Bit ; giriş2 : in Bit ;
        toplam : out Bit ; elde : out Bit ) ;
end component ;

component or_kapısı
port (
        g1 : Bit ; g2 : Bit ; Ç1 : out Bit ) ;
end component ;

begin

U0 : yarı_toplayıcı
port map (
        giriş1 => girişt1 , giriş2 => girişt2 ,
        toplam => geçiçi_toplam, elde => geçiçi_elde1 ) ;

U1 : yarı_toplayıcı
port map (
        giriş1 => geçiçi_toplam , giriş2 => elde_girişi ,
        toplam => sonuç , elde => geçiçi_elde2 ) ;
```

```
U2: or_kapısı
port map (
        g1 => geçiçi_elde1 , g2 => geçiçi_elde2,
        ç1 => elde_çıkışı ) ;

end  yapı ;
```

Yukarıdaki örnekte de görüldüğü gibi kullanılan her component bir tanımlayıcı ile etiketlenmiştir. Bu etiketten sonra kullanılan component'lerin yapı içerisindeki işaret yerleri ve bağlantıları ifade edilir. Tüm bu tanımlamalar sonucunda VHDL dili şekil 3.2 tanım ile oluşturulan tanımlamayı Şekil 3.3 teki yapı biçiminde gerçekleyecektir.



Şekil 3.2 Tam Toplayıcı Bloğu



Şekil 3.3 Yarı Toplayıcılardan Tam Toplayıcı Yapısı Oluşturma

3.4.2. Veri Türleri ve Nesneler

VHDL dilinde değer tutan her şey bir nesne olarak görülür. Her nesne de kendinin yapısını tanımlayan bir türe sahiptir.

3.4.2.1. Veri Türleri

VHDL dilinde, sıkça kullanılacağı düşünülen belirli data türleri yazılım içerisinde oluşturulmuştur. Örneklerde görülen 'bit' deyimi bahsedilen türlerden biridir. Ancak VHDL'de bu konu ile ilgili olarak esnek bir yapı oluşturulmuştur. Kullanıcı kendi veri türünü oluşturabilir. Aşağıda bazı veri türü tanımlama örnekleri mevcuttur. Bu örnekler genel olarak integer yapısında olan tür tanımlamalarını gösteren örneklerdir.

type x **is range** –10 **to** 10 ;
type y **is range** 40 **downto** 30 ;
type z **is range** –1E10 **to** 1E10 ;

İnteger tür tanımlamalarının yanı sıra kayan nokta tür tanımlamaları da yapılabilir. Bu örneklerde sınır tam sayılardan oluşmaz. Aşağıda bununla ilgili bir örnek mevcuttur.

type k **is range** –2+ 0.1E-20 **to** 2 + 0.1 E 20 **;**

Bu iki farklı tür tanımlamaları dışında kalan tüm tanımlamalar da farklı tür tanımlamaları olarak düşünülür. Bu anlamda, tür tanımlamaları 3 farklı grupta düşünülebilir.

3. grup tür tanımlamalarına en iyi örnek olarak 3-seviyeli logik tasarım düşünüldüğünde, ortaya çıkacak 3 türü tanımlama sorunu cevap verir. Burada 3_seviyeli parametresi, '0', '1', 'Z' değerlerinden oluşturulacak şekilde tanımlanması durumunda 3 seviyeli yapı tanımlanmış olur. Bununla ilgili örnekler aşağıda verilmiştir.

type 3_seviye **is** (‘0’ , ’1’ , ’Z’) ;
type mantık **is** (False,True) ;

3.4.2.2. Nesneler

VHDL’de 3 farklı nesne tanımlamaları bulunmaktadır. Bunlar; signals (işaretler), variables (değişkenler) ve constants (sabitler). Tüm bu tanımlamaların farklı özellikleri bulunmaktadır.

Signal’ ler donanımsal bağlantıyı ifade eden nesne türleridir. Değişimleri bağlı oldukları işaretlerin değişimleri ile gerçekleşir. Doğrudan değerinin değişimine yönelik işlem yapılamaz. Variable’ lar signal’lardan farklıdır. Değerinin değişimi için işlem yapılmalıdır. İşlem sonucunda elde edilen değişim sonucu, bir sonraki yazıma kadar kalır. Son tanımlama biçimi ise constant’ lardır. Bunlar bu iki tanımlamadan farklı olarak bir kez değer alırlar ve bu değeri sürekli olarak korurlar. Bu anlamda bir ROM gibi düşünülebilir. Aşağıda tanımlamalarla ilgili örnekler sunulmuştur.

constant x : Integer := 16#FFFF# ;
variable y : Boolean ;
signal z : Bit ;

3.4.3. Interface (Arabirim) Listeleri

İnterface listeleri VHDL dili içerisinde 4 farklı yapıda kullanılır. Bunlar entity tanımlamaları, yerel component tanımlamaları, block ifadeleri ve altprogram tanımlamalarıdır. Tüm yapılarda interface’i oluşturan elemanlar parantez içerisinde yer alacaktır. Her interface elemanı

- nesne sınıfı (**signal, constant** yada **variable**)
- verinin akış yönü (**in , out , inout** yada **buffer**)
- veri türü (Bit vb...)

gibi vasıflarla birlikte tanımlanır.

Her bir entity tanımlaması, component tanımlaması ve blok tanımlaması için , port ve generic'ler için olmak üzere, iki interface listesi vardır. Alt program tanımlamaları ise alt program parametrelerini gösteren bir adet interface listesi bulunur. Tablo 3.1'de port, generic ve parametre türleri için izin verilen nesne sınıfları ve modları verilmiştir.

Tablo 3.1 VHDL Interface (Arabirim) Listeleri

Entity , component ve block'lardaki interface listesi		
Interface nesnesi	nesne sınıfı	izin verilen mod
port	signal	in out inout buffer
generic	constant	in

altprogram'lardaki interface listesi		
Interface nesnesi	nesne sınıfı	izin verilen mod
parametre	signal	in out inout
parametre	constant	in
parametre	variable	in out inout

3.5. VHDL Dili Ana Yapıları

VHDL dilinde, entity tanımlamaları, architecture yapısı, package tanımlamaları, package yapısı ve altprogramlar gibi önemli ana yapılar bulunur.

3.5.1. Entity tanımlamaları

Entity tanımlamaları her tasarım içerisinde mutlaka bulunması gereken bir yapıdır. Genel yazım kuralı aşağıdaki gibidir:

entity kimlik **is**
generic interface listesi ;
port interface listesi ;
Tanımlamalar

begin
İfadeler
end kimlik ;

Entity tanımlamaları içerisinde önemli tanımlamalar yapılır. Bu tanımlamalar tür tanımlamaları, alt-tür tanımlamaları, sabitlerin tanımlamaları, dosya tanımlamaları, altprogram tanımlamaları ve "use" ifadeleri gibi tanımlamalardır.

3.5.2. VHDL Architecture (Mimari) Yapıları

Architecture yapıları genel yazım biçimi aşağıdaki gibidir.

architecture kimlik **of** kimlik_hedefi **is**
tanımlamalar
begin
ifadeler
end kimlik ;

Bu yapı içerisinde ifade edilen tanımlamalar bölümü kapsamında temel tanımlamalar, signal tanımlamaları, alt program yapısı, nitelik tanımlamaları ve özellikleri, component tanımlamaları gibi tanımlamalar mevcuttur.

3.5.3. Alt Programlar

Alt programlar, VHDL'in önemli yapılarındandır. Alt programlar program içerisinden tekraren çağrılabilirler. VHDL dili, procedure ve function olmak üzere iki tür altprogram yapısını destekler. Alt programlar dönüş değeri almazlar. Function'lar ise tanımı yapılacak bir değer ile döner. Altprogram tanımlamaları sadece interface bilgisini içerir. Alt program yapısı ise interface bilgisi, yerel tanımlamalar ve ifadeleri içerir. Alt program tanımlaması ile altprogram yapısı arasındaki fark; entity bildirimi ve archtecture yapısı arasındaki fark gibidir. Alt program bildirimleri aşağıdaki gibi yapılır.

procedure kimlik interfece_listesi

function kimlik interface_listesi **return** dönüş_türü

Ancak bu yapılarda, interface listelerinin olma zorunluluğu bulunmamaktadır. Yani alt programlar parametresiz olabilir. Bununla ilgili olarak bazı örnekler verilmiştir.

procedure A ;
function B **return** byte ;
procedure C (x : **inout** Integer) ;
function D (x : Integer) **return** byte ;
procedure C (**variable** x : **inout** Integer) ;
function D (**constant** x : **in** Integer) **return** byte ;

Alt program tanımlamaları aşağıda ifade edildiği gibi yapılabilir. Function ve procedure'lar için aynı yapı geçerlidir. Ayrıca function ile ilgili bir örnek mevcuttur.

altprogram özelliği **is**
bildirimler
begin
    ifadeler
end kimlik ;
function deneme (X , Y : byte) **return** byte
begin
    if (X > 2)
    return X ;
else
    return Y ;
end if ;
end deneme ;

3.5.4 Paket ve 'Use' Yapıları :

Tüm yazılım sistemlerinde olduğu gibi birden çok yerde kullanılan elemanları ortak olarak kullanmak için belirli yapılara imkan verilmiştir. Vhdl dilinde ortak kullanım oluşturma yapısını paketler meydana getirir.

Paket bildirimleri :

```
package kimlik is
        tanımlamalar
end kimlik ;
```

şeklindedir. Bununla ilgili bir örnek verilmiştir.

```
Package logic is

type 3_seviye is ('0','1','Z') ;
constant bilinmeyen_deger : 3_seviye := '0' ;
function tümleme (
        Giriş : 3_seviye )
return  3_seviye ;

end logic ;
```

Bu örnekte içerisinde fonksiyon tanımı bulunan bir paket bildirim yapısı verilmiştir. Paketlerin genel yapıları ise

```
package body kimlik is
        bildirimler
end kimlik ;
```

şeklinde gösterilebilir.

Verilen örneğin paket yapı yazılımı ise aşağıdaki gibi olabilir.

```
package body logic is
function tümle (
        giriş : 3_seviye )
return 3_seviye is
begin
case giriş is
when ‘0’ =>
        return ‘1’ ;
when ‘1’ =>
        return ‘0’ ;
when ‘Z’ =>
        return ‘Z’ ;
end case ;
end tümle ;
end logic ;
```

Paket yapılarının kullanımları use ifadeleri ile gerçekleştirilir. Use yazımından sonra kullanılacak paketin adı yazılır. Hemen ardından nokta ve sonrasında ise paket içerisinde kullanılacak tanım yazılır. Başka kullanılacak yapılar varsa virgülden sonra aynı mantık içerisinde yazılarak yapılır. Paket içerindeki tüm tanımlamalar kullanılacak ise bu durumda paketin adından sonra nokta ve sonrasında ‘all’ ifadesi yazılarak, tüm tanımlamalar alınmış olacaktır. Bununla ilgili olarak yukarıdaki örnekte gerçeklenen logic paketini kullanan bir kod yazılabilir.

```
use  logic.all ; logic paketi içerindeki tüm tanımlamalar alındı.
entity tümle is
port (
        X : in 3_seviye ; Y : out 3_seviye) ;
end tümle  ;

architecture tümle_yapısı of tümle is
begin
```

```
process
begin
        Y <= tümle (X) 10 ns ;
wait on X ;
end process ;
end tümle_yapısı ;
```

4. ŞİFRELEME ALGORİTMALARI

4.1 Asimetrik Şifreleme Algoritmaları

Kriptoloji yukarıda tanımlı özellikleri sağlaması açısından kendi içerisinde çeşitli alt alanlara ayrılmıştır. Verinin gizliliğinin sağlanması çoğu zaman simetrik şifreleme algoritmalarıyla; anahtar değişimi, kimlik doğrulama gibi işlemler asimetrik şifreleme algoritmalarıyla, veri bütünlüğü ve reddedememe işlemleri hash fonksiyonları, sayısal imza algoritmaları ve sertifikasyon işlemleriyle sağlanmaktadır. Farklı kriptografik yapıların kullanılmalarının nedenleri olarak işlemlerin doğasından gelen farklılıklar, algoritmaların gerçeklenebilirlikleri, çalışma hızları gibi çeşitli özellikler gösterilebilir.

Anahtar yapısına ve kullanımına bağlı olarak kriptografik algoritmalar simetrik ve asimetrik şifreleme algoritmaları olmak üzere ikiye ayrılmaktadır.

Bundan sonraki 2 bölümde asimetrik ve simetrik algoritma tipleri ayrıntılı biçimde işlenmiştir. Uygulama için kullanılan DES algoritmasıda açıklanmış ve daha sönraki bölümlerde donanım açısındna tekrar incelenmiştir.

4.1.1 Asimetrik Algoritmaların Yapısı

Açık-anahtar şifreleme algoritmaları olarak da geçen asimetrik algoritmaların en önemli özelliği kapama ve açma işlemleri için farklı iki anahtara ihtiyaç duymasıdır. Açık-anahtar şifreleme algoritması denmesinin sebebi kapama anahtarının herkesçe bilinmesinde bir sakınca olmamasıdır. Kapama anahtarından açma anahtarının elde edilmesi günümüz işlem gücü göz önüne alındığında oldukça büyük bir zaman dilimi gerektirmektedir.

Bu tip kriptosistemlerde kapama anahtarı genellikle açık anahtar, açma anahtarı ise özel anahtar olarak adlandırılır. Özel anahtara gizli anahtar da denebilmekle birlikte simetrik algoritmalarda kullanılan gizli anahtarlarla

karşılaştırılmaması açısından özel anahtar adının kullanılması daha doğru olacaktır. Açık anahtar K_1 ve özel anahtar K_2 kullanılarak yapılan kapama ve açma işlemleri aşağıda gösterilmektedir.

$$E_{K1}(M) = C$$
$$D_{K2}(C) = M$$

Mesajların özel anahtarla kapatılıp, açık anahtarla açılması durumu sayısal imzalar bölümünde incelenmiştir.

Bu tez kapsamında örnek olarak incelenen asimetrik şifreleme algoritmaları RSA ve El-Gamal algoritmalarıdır. Bu iki algoritmayla asimetrik algoritmaların yapısı hakkında genel bir bilgi verilmesi amaçlanmıştır. RSA çarpanlara ayırma, El-Gamal ise logaritmik işlemlerin zorluğu ilkesinden hareketle tasarlanmıştır.

Asimetrik algoritmalar dar-veya, kaydırma, toplama gibi basit işlemler içeren simetrik algoritmalara göre karmaşık (büyük sayılarda tanımlı işlemler) işlemler içermesinden dolayı çok daha yavaştır. Bu sebepten dolayı büyük veri dosyalarının şifreleme yerine, daha küçük verilerle kimlik doğrulama, anahtar dağıtımı ve imzalama işlemlerinde kullanılmaktadır.

4.1.1.1. Çarpanlara Ayırma Problemi

$Z_n = \{0, 1, ..., n-1\}$ kümesi üzerinde modülo n'ye göre toplama ve çarpma işlemleri tanımlı olmak üzere n, p ve q gibi iki ayrı asal sayının çarpımı olsun. Böyle bir n için $\Phi(n) = (p-1) * (q-1)$ tanımlansın. RSA kriptosisteminde şifreleme ve şifreyi açma işlemlerinde kullanılacak a, b sayıları mod $\Phi(n)$'e göre birbirlerinin tersi olsunlar.

$$ab \equiv 1 \pmod{\Phi(n)}$$
$$ab = t\,\Phi(n) + 1$$

n'ye göre asal sayıların kümesi Z_n^* olmak üzere $x \in Z_n^*$ sayısı için aşağıdaki işlemler yapılabilir.

$(x^b)^a \equiv x^{t\,\Phi(n)+1} \pmod n$

$(x^b)^a \equiv x^{t\,\Phi(n)}\, x \pmod n$

$(x^b)^a \equiv 1^t\, x \pmod n$

$(x^b)^a \equiv x \pmod n$

Görüldüğü gibi $e_k(x) = x^b \bmod n$ tek yönlü şifreleme fonksiyonunda a'nın bilinebilmesi için herkesçe bilinen n değerinden p ve q asal sayılarının elde edilmesi gerekmektedir. Ancak bu durumda $\Phi(n) = (p-1)(q-1)$ bulunarak şifreyi açmada kullanılan a gizli anahtarına ulaşılır, n sayısının çok büyük olması çarpanlara ayrılması zorluğunu karşımıza çıkarmaktadır.

4.1.1.2. Sonlu Logaritmik Problem

p asal sayısı alınarak oluşturulan Z_p kümesi, üzerinde tanımlı modülo p toplama ve çarpma işlemleriyle birlikte bir alandır. $\alpha \in Z_p$ ilkel bir eleman($\alpha^{(p-1)} \equiv 1 \bmod p$) olmak üzere $\beta \in Z_p^*$ olsun. $0 \leq a \leq p-2$ arasındaki a değerinin aşağıdaki eşitlikten bulunabilme zorluğuna

$\alpha^a \equiv \beta \pmod p$

$a = \log_\alpha \beta$ sonlu logaritmik problem adı verilir. p'nin asal sayı olarak seçilmesiyle oluşturulan Z_p alanı kullanılması durumu El-Gamal kripto sisteminde karşımıza çıkmaktadır. Eliptik eğri kriptosisteminde ise Z_p üzerinde bir eliptik eğri tanımlanarak oluşturulan grup sonlu logaritmik problemde kullanılır.

f(x) Z_p'de çözümü olmayan (indirgenemez) bir polinom olmak üzere $|Z_p / f(x)|$ de bir alandır ve eleman sayısı p^n 'dir (deg f(x) = n). Bu alan üzerinde toplama (+) ve çarpma (*) işlemleri polinomlar için tanımlı toplama ve çarpma işlemleriyle aynıdır. Polinom katsayıları mod p'ye göre belirlenir, çarpma işlemi sonucunda n'den daha büyük dereceli bir polinom ortaya çıkarsa f(x)'e göre kalanı sonuç olarak alınır. Eliptik eğri kripto sisteminde f(x) olarak seçilen eğrinin denklemi kullanılır.

4.2. Asimetrik Şifreleme Algoritmaları Örnekleri

4.2.1. RSA Algoritması

Genel Yapısı ve Özellikleri

RSA asimetrik algoritmalar içerisinde anlaşılması ve gerçeklenmesi en kolay olan algoritmadır. Rivest, Shamir ve Adleman tarafından 1977 yılında sunulmuştur. Tasarımından günümüze kadar geçen zaman dilimi içerisinde üzerinde oldukça çok sayıda kriptoanalitik çalışmalar yapılmakla birlikte algoritmanın güvenliği hakkında kesin bir kanı yoktur. Bu da algoritma hakkındaki olumlu düşünceleri kuvvetlendirmektedir.

RSA, büyük sayıların çarpanlara ayrılabilmesi zorluğu ilkesinden yola çıkılarak tasarlanan bir algoritmadır. Burada büyük sayılar olarak 512, 1024 veya 2048 bitlik sayılar kastedilmiştir. Bu sayılar üzerinde kullanılan işlemler çeşitli algoritmalarda tanımlanmakla birlikte klasik işlemlerle gerçeklenmesi durumunda RSA'in başarımı önemli derecede düşmektedir.

Donanım olarak gerçeklenen RSA, DES'ten 1000 kat daha yavaştır. RSA'i gerçeklemek amacıyla tasarlanan entegreler 512 bit mod işlemleriyle 64kb/s hızla şifreleme yapabilmektedir. 1Mb/s hızda çalışan RSA entegrelerin tasarlanması da düşünülmektedir. RSA algoritması akıllı kartlarda da gerçeklenmiştir. Bu uygulamalarda başarım çok daha düşük olmakla birlikte ek kripto işlemcileri kullanılarak başarımı artırma yoluna gidilmektedir.

Algoritma Tanımı

1) Giriş: açık metin X değeri

2) Çıkış: kapalı metin Y değeri

3) p ve q büyük asal sayı değerleri belirlenerek $n = pq$ ve $\phi(n) = (p-1)(q-1)$ değerleri hesaplanır.

4) ab = 1 (mod ϕ(n)) olacak şekilde a ve b değerleri belirlenir. Burada a ve b ise özel ve açık anahtar çiftleridir. n ve b değerleri açık, p, q ve a değerleri ise gizli tutulmaktadır.

5) $Y = X^b \bmod n$ değeri kapama, $X = Y^a \bmod n$ değeri ise açmada uygulanan işlemlerdir.

Algoritmanın Güvenliği

RSA algoritmasının güvenliği yukarıda da anlatıldığı gibi tamamen büyük sayıların çarpanlara ayrılmasının zorluğuna bağlıdır. Bununla birlikte b ve Y kullanılarak X'in bulunabilmesi için n' nin çarpanlara ayrılması gerektiği hiç bir zaman doğrulanmamıştır. Çok daha değişik yolllar kullanılarak RSA'in kriptoanalizinin yapılabilmesi olası bir durumdur. Böyle bir yolun bulunması büyük sayıların çarpanlarına ayrılabilmesi işlemini de kolaylaştıracaktır.

RSA'e yapılacak olası ataklardan biri de $\phi(n) = (p-1)(q-1)$ değerinin tahmin edilmesidir. Bu atak n'nin çarpanlara ayrılma işleminden daha kolay değildir.

4.2.2. El-Gamal Algoritması

Genel Yapısı ve Özellikleri

El-Gamal açık anahtar şifreleme algoritması özellikle sayısal imzalarda kullanılmak üzere tasarlanmış olup ilk olarak 1985 yılında bir makalede sunulmuştur. Sonlu kümelerde logaritmik işlemlerin zorluğu ilkesinden hareketle tasarlanmıştır.

Algoritma Tanımı

1) Giriş: açık metin X değeri

2) Çıkış: kapalı metin Y değeri

3) p büyük (512 , 1024 bit) bir asal sayı olmak üzere K = { p, α, a, β } kümesi tanımlanır. Burada "p, α ve β" değerleri açık, "a" değeri ise gizlidir. "β" değeri ise aşağıda verilmiştir.

$$\beta = \alpha^a \bmod p$$

4) k gizli rastgele bir sayı olmak üzere, ($k \in Z_{p-1}$)

$$Y_1 = \alpha^k \bmod p$$
$$Y_2 = x\beta^k \bmod p$$

değerleri hesaplanarak karşı tarafa $e_K (X, k) = (Y_1, Y_2)$ kapalı değeri yollanır.

5) Açma işleminde alıcı taraf

$$X = d_K (Y_1, Y_2) = Y_2 (Y_1^a)^{-1} \bmod p$$

işlemini kullanarak açık metine ulaşır.

Algoritmanın Güvenliği

El-Gamal kriptosisteminin deterministik olmaması, kapalı ve açık metnin rastgele bir k değerine bağlı olması aynı açık metnin birden çok karşılığının olabilmesini sağlamaktadır. Ayrıca kapalı mesajın genişletilmesi, açık mesajın iki katı olması sistemin zayıflığındandır.

El-Gamal şifreleme sisteminin güvenliği açısından p uzunluğunun 768 bit olmasının (1996 yılı için) yeterli olduğu düşünülmekteydi. Günümüzde ve ilerleyen zaman içerisinde sistemin güvenliği açısından p'nin 1024 bit veya daha büyük değerler alınması doğru olacaktır.

4.2.3. Eliptik Eğri Kriptosistemi

Eliptik Eğriler

p>3 ve asal olmak üzere $y^2 = x^3 + ax + b$ eğrisi Z_p üzerinde $(x,y) \in Z_p \times Z_p$ çözümleri kümesidir.

$$y^2 = x^3 + ax + b \qquad (\text{mod } p)$$

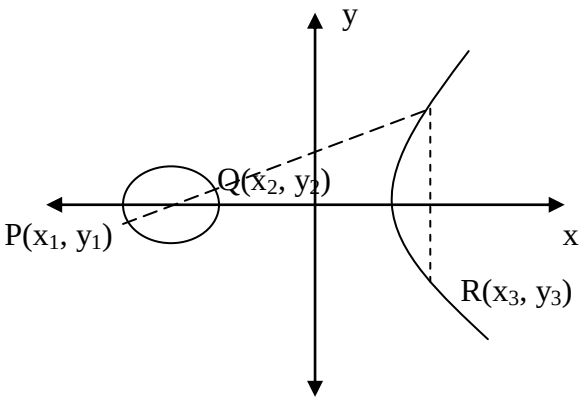
Burada $a,b \in Z_p$ olmak üzere $4a^3 + 27b^2 \neq 0 \pmod p$ sabitleri ve sonsuzdaki nokta olarak da Ό tanımlanır.

E eliptik eğrisi üzerindeki noktalar için tanımlanan uygun işlemlerle bir Abel grubuna dönüştürülebilir. Bütün işlemler Z_p'de gerçeklenmek üzere $P(x_1, y_1)$ ve $Q(x_2, y_2)$ E üzerinde iki nokta olsun. $x_2 = x_1$ ve y2 = y1 olması durumunda P + Q = Ό (sonsuzda nokta), aksi halde $P + Q = (x_3, y_3)$ değerini alır. Burada x_3 ve y_3 değerleri aşağıdaki eşitliklerden bulunabilir.

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

$$\lambda = (y_2 - y_1) / (x_2 - x_1) \qquad (P \neq Q)$$
$$\lambda = (3x_1^2 + a) / (2y_1) \qquad (P = Q)$$

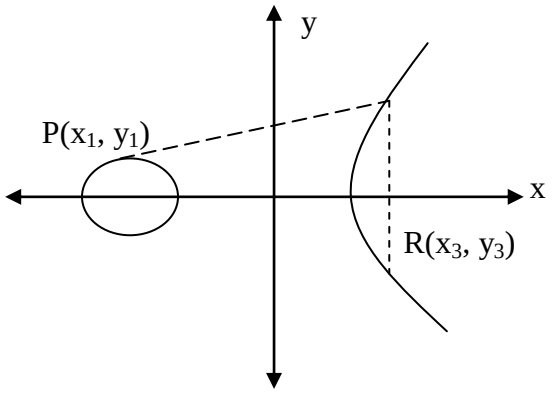
P + Ό = Ό + P = P olmak üzere E eğrisi birim elemanı Ό olan bir Abel gruptur. $\forall$ x,y $\in$ E elemanının bu grup üzerinde tanımlı olan tersi -(x, y), dolayısıyla (x, -y)'dir. Eliptik eğriler üzerinde bulunan $P(x_1, y_1)$ ve $Q(x_2, y_2)$ noktaları için tanımlanan toplama işlemi geometrik olarak Şekil 4.1'de gösterilmiştir.



Şekil 4.1 Eliptik Eğri üzerinde farklı iki noktanın toplamı

Burada P + Q değerinin bulunması için P ve Q noktaları arasında bir doğru çizilir. Bu doğrunun eğriyi kestiği noktanın x eksenine göre simetriği R = P + Q değeridir.

$x_1 = x_2$ ve $y_1 = y_2$ olması durumunda P noktasından çizilen tanjantın eğriyi kestiği noktanın x eksenine göre simetriği toplama işlemi sonucudur. Şekil 3.2'de eğri üzerinde toplama işlemi gösterilmiştir.



Şekil 4.2 Eliptik Eğri üzerinde ikiyle çarpma

Eliptik Eğri Kriptosistemi Yapısı

Eliptik eğri kripto sistemler diğer asimetrik kriptosistemlere göre daha kısa anahtar uzunluklarıyla benzer şifreleme gücüne sahip olması nedeniyle daha kolay gerçeklenebilir. Özellikle akıllı kartlar gibi düşük işlem gücüne sahip sistemler için tercih sebebi olmaktadır. Menezes – Vanstone eliptik eğri kriptosistemi yapısı aşağıda verilmiştir.

E, Z_p (p>3) üzerinde tanımlı bir eğri olmak üzere P = Z_p x Z_p açık metinler kümesi C = E x Z_p x Z_p kapalı metinler kümesi, K = { (E, α, a, β) : $\beta = a\alpha$ } ve $\alpha \in$ E olsun. Burada α ve β herkesçe bilinmekte a ise gizli tutulmaktadır. Rastgele bir $k \in Z_{|H|}$ (H, E içerisinde çevrimli bir grup)ve x =$(x_1, x_2) \in Z_p^* \times Z_p^*$ için

$e_K(x, k) = (y0, y1, y2)$

$y_0 = k\alpha$

$(c1, c2) = k\beta$

$y_1 = c_1 x_1 \quad \text{mod } p$

$y_2 = c_2 x_2 \quad \text{mod } p$

Kapalı metin y = (y0, y1, y2) değeri için açma işlemi ise aşağıdaki gibidir.

$d_K(y) = (y_1 c_1 \text{ mod } p, y_2 c_2 \text{ mod } p)$, a $y_0 = (c1, c2)$

4.3 HASH Fonksiyonları

Fonksiyonların Genel Yapısı

Hash işlemi değişken uzunluktaki verilerden (Kb, Mb gibi) çok daha küçük (128-160 bit) ve sabit uzunlukta değerlerin üretilmesidir. Hash fonksiyonları ilk olarak veritabanlarında kayıtların aranmasında kullanılmıştır. 1976'lardan sonra kriptolojide yaşanan gelişmelere bağlı olarak açık anahtar şifreleme sistemlerinin, sayısal imzaların ve kimlik doğrulama işlemlerinin ortaya çıkmasıyla hash fonksiyonlarının da önemi artmış, üzerinde yapılan çalışmalar yoğunlaşmıştır.

Hash fonksiyonları blok şifreleme tabanlı ve blok şifreleme tabanlı olmayanlar olarak iki grup altında incelenmektedir. Blok şifreleme tabanlı hash fonksiyonları tek-yönlü fonksiyon olarak blok şifreleme algoritmaları kullanırken, blok şifreleme tabanlı olmayan hash fonksiyonları matematiğin pek çok alanında yer alan tek-yönlü fonksiyonları kullanırlar.

Genel anlamda hash fonksiyonlarının güvenilir olabilmesi için Jueneman[2] tarafından bazı özellikler sıralanmıştır. Öncelikle hash fonksiyonlarının herhangi bir kriptografik donanım kullanmadan kolay bir şekilde bilgisayarlarda gerçeklenebilmesi gerekmektedir. Hash sonucu mümkün olan tüm permutasyonlara, veri ekleme ve silme işlemlerine duyarlı olabilmelidir. Farklı iki metin sıkıştırıldığında hash değerlerinin aynı olma olasılığı düzenli bir dağılım içerisinde yer almalıdır. Hash fonksiyonu sonucu olası ataklara karşı duyarlı olmalıdır. Günümüzde 128 bit yeterli bir büyüklüktür. Ayrıca hash fonksiyonunun ters fonksiyonu olmamalı, alt ve bağımsız parçalara ayrılmamalıdır.

Hash fonksiyonları kriptolojide sayısal imzalarda ve kimlik doğrulama işlemlerinde önemli bir yer tutmaktadır. Örnek olarak gösterilebilecek Hash algoritmaları MD4, MD5 ve SHA-1 hash algoritmalarıdır.

5. SİMETRİK ŞİFRELEME ALGORİTMALARI

5.1. Simetrik Algoritmaların Yapısı

Anahtar yapısına ve kullanımına bağlı olarak kriptografik algoritmalar simetrik ve asimetrik şifreleme algoritmaları olmak üzere ikiye ayrılmaktadır. Simetrik algoritmalarda kapama ve açma anahtarları birbirlerinden elde edilebilmekte veya aynı anahtar her iki işlemde de kullanılabilmektedir.

Simetrik algoritmalar çok farklı şekillerde adlandırılmaktadır. Geleneksel algoritmalar, gizli-anahtar algoritmaları, tekil-anahtar algoritmaları bunlardan bazılarıdır. Farklı şekillerde adlandırılmalarının nedeni bu tip algoritmaların ortak özelliklerinden kaynaklanmaktadır. Simetrik algoritmalarda taraflar (gönderici ve alıcı) gizli haberleşmeye başlamadan önce şifreleme (kapama ve açma) işlemlerinde kullanılacak bir anahtara ihtiyaç duyarlar. Bu anahtar güvenli olduğu kabul edilen bir yol üzerinden taraflara dağıtılır. Haberleşmenin gizliliği öncelikle anahtarın gizliliğine bağlıdır.

Simetrik algoritmalarda kapama (E) ve açma (D) işlemleri K anahtarı kullanılarak M açık, C kapalı metni üzerinde (5.1a) ve (5.1b) de gösterilmiştir.

$$E_K(M) = C \quad (5.1a)$$

$$D_K(C) = M \quad (5.1b)$$

Simetrik algoritmalar kendi içerisinde dizi ve blok şifreleme algoritmaları olarak ikiye ayrılmaktadır. Dizi şifreleme algoritmaları açık metindeki bit veya sekizliler üzerinde, blok şifreleme algoritmaları ise sabit uzunluktaki bloklar üzerinde işlem yaparlar. Günümüzde kullanılan donanım özellikleri göz önüne alındığında blok uzunluğunun 64 bit alınması şu an için yeterli olmaktadır. Bu değer

kriptoanaliz açısından yeterli derecede büyük, gerçeklenebilirlik ve başarım açısından ise yeterli derecede küçüktür.

Genel anlamda simetrik algoritmaların çok farklı isteklere yanıt verebilmesi gerekir. Değişik uygulama alanlarında, değişik platformlarda kolay gerçeklenebilir olması istenen özelliklerden bazılarıdır. Dosya şifreleme, rastgele sayı üretme, paket şifreleme (ATM'deki 48 sekizli gibi) belli değişikliklerle hash fonksiyonlarına dönüştürülebilme bu uygulama alanlarından bazılarıdır. Aynı şekilde VLSI tasarımlarda; büyük, orta ve küçük ölçekli mikroişlemcilerde farklı seviyelerdeki güvenlik sınıflandırılmalarını sağlayacak şekilde kullanılabilmesi, yazılım veya donanım olarak kolay gerçeklenebilir olması, olası hatalara yol açmaması gerekmektedir.

Bu tez çalışması kapsamında gerçeklenen uygulamada simetrik algoritmalardan blok şifreleme algoritmaları kullanılmış olmakla birlikte dizi şifreleme algoritmalarına da bu bölüm içerisinde değinilmiştir.

5.2. Blok Şifreleme Algoritmaları

5.2.1. Blowfish Algoritması

Genel Yapısı ve Özellikleri

Blowfish büyük ölçekli mikroişlemcili sistemlerde gerçeklenmek üzere tasarlanmış değişken anahtar uzunluklu 64 bitlik bir blok şifreleme algoritmasıdır. Veriler 32 bitlik mikroişlemcilerde sekizli başına 26 saat çevrimi hızla şifrelenebilmektedir. Algoritma 5K'lık bir bellek alanına ihtiyaç duymakta, 32 bitlik işlenenler üzerinde toplama, dar-veya, ve tablo okuma gibi kolay gerçeklenebilen işlemler kullanmaktadır. Anahtar uzunluğu 32 bitten 448 bite kadar genişletilebilmektedir.

Blowfish daha çok anahtarın sık değişmediği haberleşme hatları veya dosya şifreleme uygulamalarında kullanılmak üzere tasarlanmıştır. Sık sık anahtar değişimine ihtiyaç duyan paket anahtarlama veya hash işlemleri için uygun değildir.

Bununla birlikte büyük cep bellek alanlarına sahip Pentium, PowerPC gibi 32 bitlik mikroişlemcilerde DES'ten çok daha hızlı çalışmaktadır.

Algoritmanın Güvenliği

Serge Vaudenay[1] Blowfish algoritmasını bilinen S-kutuları ve r sarmal için incelemesi sonucunda diferansiyel saldırıyla P dizisinin 2^{8r+1} bilinen açık metinle çıkarılabileceğini göstermiştir. Belli zayıf anahtarlar için kötü S-kutuları üretme olasılığı $1/2^{14}$ dür. Bu durumda aynı saldırıyla 2^{4r+1} bilinen açık metinle P dizisi elde edilebilir. Bu saldırı yöntemi 16 sarmallık Blowfish için tamamen etkisiz olmaktadır.

Verilen bir S-kutusu için iki girişin aynı olmasını sağlayan anahtarlar zayıf anahtarlardır. Anahtar uzunluğunu genişletmeden zayıf anahtarların aranması olanaksızıdır. Bununla birlikte bulunması zor gözükse de zayıf anahtarların bilinmesi oldukça önemlidir.

5.2.2. IDEA Algoritması

Genel Yapısı ve Özellikleri

IDEA (International Data Encryption Standard) algoritması ilk olarak Xuejia Lai ve James Massey tarafından PES (Proposed Encryption Standard) adı altında 1990 yılında sunulmuştur. Bir sonraki yıl Biham ve Shamir'in diferansiyel atakları geliştirmeleri üzerine yazarlar bu tür ataklara karşı algoritmayı geliştirmişler ve algoritmaya IPES (Improved Proposed Encryption Standard) adını vermişlerdir. 1992 yılında IPES, IDEA adını almıştır.

Algoritmanın tasarımında yatan temel düşünce değişik cebir gruplarından işlemlerin bir arada kullanılmasıdır. Dar-veya, modulo 2^16 toplama ve modulo 2^16+1 çarpma işlemleri olmak üzere hem yazılım hem de donanım olarak kolay gerçeklenebilen 3 farklı cebir grubu işlemlerde kullanılmaktadır.

IDEA'nın şu anki yazılım uygulamaları DES'in iki katı hızda çalışmaktadır. IDEA, 33MHz i 386'da 880kb/s, 66MHz i 486'da 2400kb/s hızla şifreleme

yapabilmektedir. IDEA'nın daha hızlı olmasını engelleyen en önemli etken algoritmada kullanılan çarpma işlemleridir.

Will Meier[1] IDEA'nın üç farklı işlemini incelemiş, uyumlu olmamalarına rağmen belli durumlarda sadeleştirmeler yapılabileceğini göstermiştir. 2 sarmallı bir IDEA algoritması için etkili olmakla birlikte normal halde (8 sarmallı) algoritma açısından sakıncalı bir durum ortaya çıkmamıştır.

John Daemen[1] IDEA için belli bir grup zayıf anahtar bulmakla birlikte bu anahtarlar DES'in zayıf anahtarlarından farklı olarak bilinen açık metin saldırısında kolaylıkla bulunabilmektedir. Bununla birlikte bu şekilde bir anahtar üretme olasılığı 1/296'dır. IDEA'nın zayıf anahtarlı kullanılamayacak şekilde değiştirilebilmesi için her bir alt anahtarın 0xdae değeriyle dar-veya işlemine sokulması yeterlidir.

Algoritmanın Güvenliği

IDEA'nın anahtar uzunluğu 128 bit olup DES'in iki katıdır. Anahtar uzayının tamamının taranarak anahtarın bulunmak istenmesi durumunda 2^128'lik bir deneme yapılması gerekmektedir. Bu da algoritmanın daha farklı yöntemler kullanılarak analizinin yapılmasının gerektiğini göstermektedir.

Will Meier[1] IDEA'nın üç farklı işlemini incelemiş, uyumlu olmamalarına rağmen belli durumlarda sadeleştirmeler yapılabileceğini göstermiştir. 2 sarmallı bir IDEA algoritması için etkili olmakla birlikte normal halde (8 sarmallı) algoritma açısından sakıncalı bir durum ortaya çıkmamıştır.

John Daemen[1] IDEA için belli bir grup zayıf anahtar bulmakla birlikte bu anahtarlar DES'in zayıf anahtarlarından farklı olarak bilinen açık metin saldırısında kolaylıkla bulunabilmektedir. Bununla birlikte bu şekilde bir anahtar üretme olasılığı 1/296'dır. IDEA'nın zayıf anahtarlı kullanılamayacak şekilde değiştirilebilmesi için her bir alt anahtarın 0xdae değeriyle dar-veya işlemine sokulması yeterlidir.

5.2.3. DES (Digital Encryption Standard) Algoritması

Genel Yapısı ve Özellikleri

Mayıs 1973'de Amerikan Ulusal Standartlar Kurumu ülke genelinde gizlilik içermeyen alanlarda kullanılmak üzere bir algoritma isteğinde bulunması üzerine IBM tarafından LUCIFER algoritmasının değiştirilmiş bir hali olan DES tasarlanmıştır. Mart 1975'te tasarlanan DES uzunca bir müddet süren tartışmalardan sonra Ocak 1977'de gizlilik içermeyen uygulamalarda kullanılmak üzere bir standart olarak kabul edilmiştir.

DES karmaşık gözükmekle birlikte yazılım veya donanım olarak kolaylıkla gerçeklenebilir. Algoritmada aritmetik işlem olarak sadece dar-veya kullanılmaktadır. Genişletme fonksiyonu E, S-kutuları, IP ve IP^{-1} permutasyonları, K_1, K_2, ..., K_{16} alt anahtarları tablo okuma işlemleriyle (yazılım gerçeklemelerinde) veya ara bağlantılar ve dar-veya kapılarıyla (donanım gerçeklemelerinde) oluşturulabilir. CRYPTO 92'de 250MHz saat hızıyla $300'lık maliyetle 1Gb/s hızında veri şifreleyen bir entegre DEC tarafından açıklanmıştır. Günümüzde donanım maliyetlerinin düşmesi, saat hızlarının artması çok daha hızlı entegrelerin tasarlanabilmesini sağlamaktadır.

Algoritma Tanımı

DES algoritması, 64 bitlik açık metini 56 bitlik anahtar değerini kullanarak 64 bitlik kapalı metine dönüştürür. Şifreleme işlemi 16 sarmaldan oluşmuştur. Her bir sarmalda tanımlı bir F fonksiyonu ve dar-veya işlemleri yer almaktadır. Sarmallarda kullanılacak 48 bitlik alt anahtarlar giriş olarak alınan 56 bitlik anahtardan üretilirler.

Şifreleme

64 bitlik X açık metini 56 bitlik K anahtarı kullanılarak 64 bitlik kapalı metine dönüştürülür.

1) Giriş olarak alınan X değeri bir IP permutasyonundan geçirilerek X_0 değeri oluşturulur. Burada yüksek anlamlı 32 bit R_0, düşük anlamlı 32 bit ise L_0 değerleridir.

$X_0 = IP(X) = R_0\, L_0$

2) For i=1 to 16 do

$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

değerleri hesaplanır. Burada kullanılan f fonksiyonu aşağıda açıklanmıştır. K_1, K_2, ..., K_{16} değerleri ise 56 bitlik K anahtarında üretilen 48'er bitlik alt anahtar kümesidir.

3) IP^{-1} ters permutasyonu $R_{16}\, L_{16}$ değerine uygulanarak Y kapalı metni elde edilir.

$Y = IP^{-1}(R_{16}\, L_{16})$

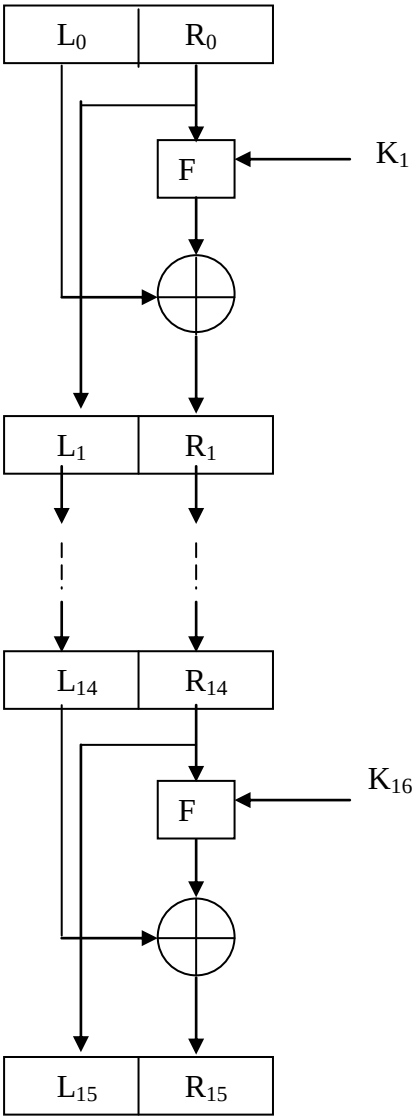
IP ve IP^{-1} permutasyonları Tablo 5.1 ve Tablo 5.2'de, algoritmaya ilişkin blok diyagramı ise Şekil 5.1'de verilmiştir.

Tablo 5.1 DES Algoritması IP permutasyonu

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tablo 5.2 DES Algoritması IP^{-1} permutasyonu

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25



Şekil 5.1 DES Algoritması

Algoritmada kullanılan $F(R_{i-1}, K_i)$ 32 bitlik R_{i-1} ve 48 bitlik K_i değerlerini kullanarak 32 bitlik bir sonuç üretir. Burada

1) R_{i-1} değeri sabit bir genişletme fonksiyonu $E(R_{i-1})$ ile 48 bite çıkartılır.

2) $E(R_{i-1}) \oplus K_i$ değeri hesaplanarak 8 adet 6 bitlik B_i ($1 \leq i \leq 8$) değerleri oluşturulur ($B_1\ B_2\ ...\ B_8$).

3) 6 bitlik B_i değerleri S-kutularından geçirilerek 4 bite indirilip C_i değerleri elde edilir. B_i'nin ilk ve son bitleri satırı, ortadaki 4 bit ise sutunu belirtir. Satır ve sutunun işaret ettiği değer S-kutusu çıktısıdır.

4) Son olarak $C = C_1\ C_2\ ...\ C_8$ değerleri sabit bir P permutasyonundan geçirilir. P(C) değeri aynı zamanda $F(R_{i-1},K_i)$' nin de değeridir.

$E(R_{i-1})$ genişletme tablosu ve P permutasyonu Tablo 5.3 ve Tablo 5.4' de verilmiştir.

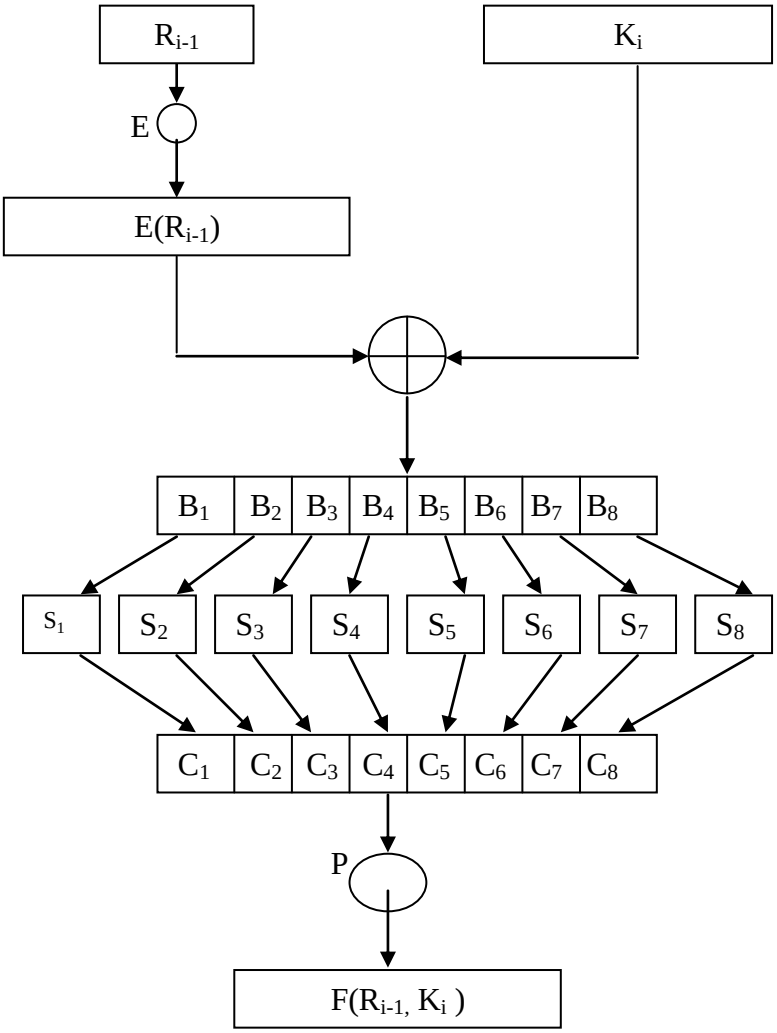
Tablo 5.3 DES Algoritması $E(R_{i-1})$ genişletme tablosu

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Tablo 5.4 DES Algoritması P permutasyonu

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Algoritmada kullanılan fonksiyonun yapısı ve S-kutuları ise, Şekil 5.2 ve Tablo 5.5 – Tablo 5.12'de verilmiştir.



Şekil 5.2 DES algoritması $F(R_{i-1}, K_i)$ fonksiyonu

Tablo 5.5 DES Algoritması S_1 kutusu

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tablo 5.6 DES Algoritması S_2 kutusu

15	1	8	14	6	11	3	4	9	7	3	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Tablo 5.7 DES Algoritması S_3 kutusu

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Tablo 5.8 DES Algoritması S_4 kutusu

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tablo 5.9 DES Algoritması S_5 kutusu

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Tablo 5.10 DES Algoritması S_6 kutusu

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Tablo 5.11 DES Algoritması S_7 kutusu

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Tablo 5.12 DES Algoritması S_8 kutusu

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Alt Anahtarların Üretilmesi:

K anahtarı 56 biti veri, 8 biti hata bulma olmak üzere 64 bitten oluşmaktadır.

1) Verilen bir K anahtar değerinden 8 bitlik hata bulma kısmı atılarak kalan 56 bitlik veri sabit bit PC-1(K) permutasyonuna sokulur. PC-1(K) = C_0 D_0 olarak 28 bitlik iki grup halinde yazılabilir.

2) For i=1 to 16 do

$C_i = LS_i (C_{i-1})$

$D_i = LS_i (D_{i-1})$

$K_i = PC\text{-}2(C_i D_i)$

Burada LS_i i değerine bağlı olarak sola dönmeyi göstermektedir. i = 1, 2, 9 veya 16 olması durumunda bir bit, aksi halde iki bit dönme gerçeklenir. Burada PC-2 diğer bir sabit permutasyondur. PC-1 ve PC-2 permutasyon tabloları Tablo 5.13 ve Tablo 5.14'de verilmiştir.

Tablo 5.13 DES Algoritması PC-1 permutasyonu

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Tablo 5.14 DES Algoritması PC-2 permutasyonu

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Algoritmanın Güvenliği

DES üzerinde yapılan eleştirilerin başında anahtar uzunluğunun 56 bit olması gelmektedir. Bilinen açık metin atakları için düşünülen özel donanım çalışmalarında Diffie ve Hellman 1977 yılında saniyede 10^6 anahtar deneyebilen bir VLSI entegresinin anahtar uzayının tamamını bir günde deneyebileceğini, böyle bir donanımın $20000000 civarında olabileceğini açıklamışlardır. CRYPTO 93'de Michael Wiener yine bir anahtar arama makinesinin detaylı bir açıklamasını vermiştir. Makina, vektörel yapıda olup saniyede $5*10^7$ anahtar deneyebilmekte, gününün teknolojisiyle $10.5 maliyetle 5760 entegreden oluşan bir çerçevesi $100000 tutmakta ve DES anahtarını 1,5 günde bulabilmektedir. 10 çerçeve kullanan bir makina ise yaklaşık olarak $1000000 olup arama zamanını 3,5 saate indirmektedir.

5.2.4. Triple DES

Genel Yapısı ve Özellikleri

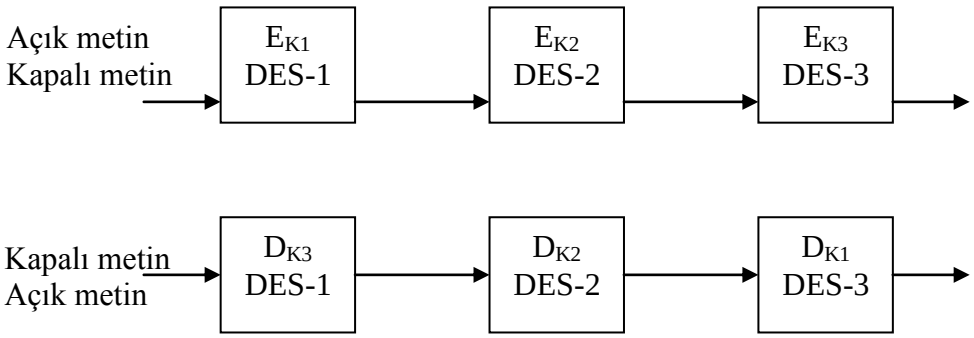
DES'in oldukça yaygın bir algoritma olması üzerinde çok fazla sayıda kriptoanaliz çalışmalarının yapılmasına sebep olmuştur. 1975 yılında ortaya çıkartılan ve 1977 yılında bir standart olarak sunulan DES'in güvenliği halen tartışılmaktadır. Ortaya çıkması sırasında NSA kurumu ile yapılan ortak çalışmalar ve bu kurumun önerilerinin dikkate alınarak ilk olarak 128 bit olarak düşünülen

anahtar boyunun 56 bite düşürülmesi, S kutularının NSA'in isteği doğrultusunda eklenmeleri kafalarda soru işareti bırakan noktalardır.

DES hakkında ileri sürülen bu düşüncelerin yanısıra geçen zaman içerisindeki teknolojik gelişmelerde algoritmanın doğal olarak zayıflamasına neden olmaktadır. Bu tür nedenlerden dolayı bazı kuruluşlar 90'lı yıllara doğru DES yerine DES algoritmasının arka arkaya üç kez uygulanmış hali olan Triple DES'i kullanmaya başlamıştır.

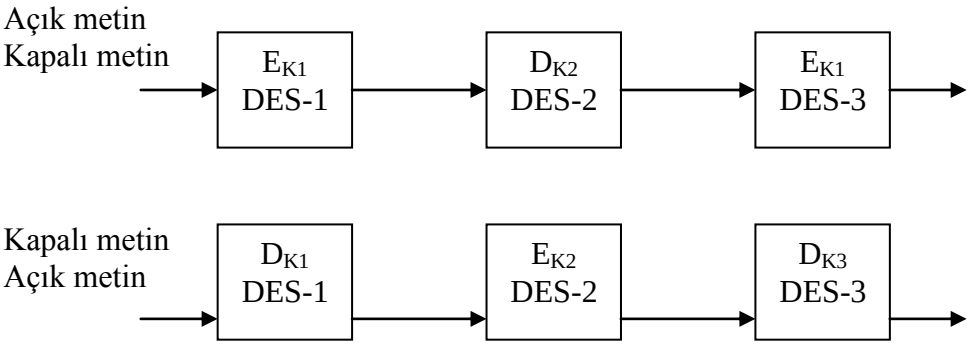
Algoritma Tanımı

Triple DES'in uygulamalarında çeşitli yöntemler ortaya atılmıştır. Bunlardan biri her bir DES adımında ayrı k_1, k_2, k_3 anahtarları kullanarak metnin kapatılması ve k_3, k_2, k_1 anahtar sırasıyla da açılmasıdır. (Şekil 5.3)



Şekil 5.3 Triple DES kullanımı

Diğer bir yöntemde k_1 anahtarıyla kapanan açık metinin kapalısı k_2 anahtarıyla açılmakta ve sonuç yeniden k_3 anahtarıyla kapatılmaktadır. Kapalı gelen metin ise k_1 anahtarıyla açıldıktan sonra k_2 anahtarıyla kapatılıp yeniden k_3 anahtarıyla açılmaktadır. (Şekil 5.4)



Şekil 5.4 Triple DES kullanımı

6. UYGULAMA ALGORİTMASI - DES

DES algoritması için daha önceki bölümlerde anlatılan kısımları daha ayrıntılı incelersek;

- **Blok Şifreleme:**

Blok şifreleyiciler, açık metni (plain text), blok adı verilen dizilere parçalayarak, her seferinde bir bloğu şifreleyen bir yapıda çalışır. Dizi şifreleyiciler (stream cipher) ise bit bit şifreleme işlemini gerçekleştirmektedirler.

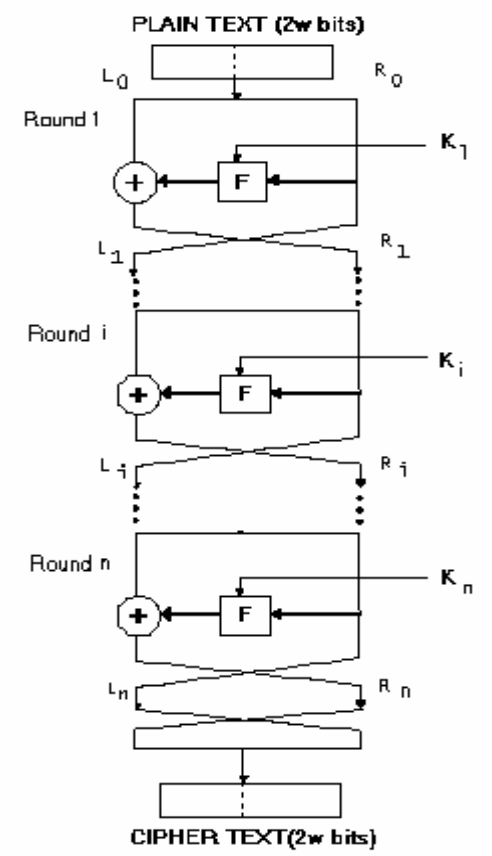
Matematiksel olarak n-bitlik blok şifreliyicilerin fonksiyonu şu şekildedir:

$E:V_n * K -> V_n$

Burada V alfabe kümesini , K herbir anahtar kümesini ve E(P,K) ise $E_k(P)$ şeklinde yazılarak V_n' den V_n 'e şifreleme fonksiyonu göstermektedir.
Bu işlemin tersi olan şifre çözme işlemi $D_k(P)$. $C = E_k(P)$ şeklinde yazılarak Açık metnin (Plain Text) P nin K anahtarı altında şifresinin çözülmesi işlemini göstermektedir.

- **Feistel Şifrelemesi**

Feistel yapısı 1971 yılında bunulmuştur. Bu yapı matematiksel olarak permutasyon ve yer değiştirme ilkelerini kullanarak şifreleme gerçekleştirir. Bu yapıda çalışan şifreleme sisteminin daha güçlü bir yapıda olduğu savunulmuştur. Şekil 6.1 Feistel Şifreleme sisteminin yapısını göstermektedir:



Şekil 6.1 Feistel Şifreleme Yapısı

- **Feistel Şifrelemesi Genel Yapısı :**

Feistel şifrelemesinde, şifrelenecek metin eşit iki parçaya ayrılır. Bu iki parça n kez çevrimden geçirilerek şifreli metin elde edilir. Tüm çevrimler aynı yapıya sahiptirler. Her çevrimde, ”F” çevrim fonksiyonunun 2’ ye bölünen parçaya alt-anahtar kullanılarak uygulanmasına değiştirme işlemi (substitution) denir. Bundan sonra elde edilen çıktı diğer yarı-parçaya ex-or lanır. Bundan sonra bu iki parçanın yer değiştirme işlemide permutasyonu sağlar.

Çevrim fonksiyonu olan “F” fonksiyonu tüm çevrimlerde aynı yapıda olmasına rağmen, farklı alt anahtarların gelmesi nedeniyle farklı parametreler ex-or lanmak üzere gönderilir.

Feistel çevriminin ilginç bir özelliği tek yönlü bir fonksiyon yapısında olmasıdır. Bu Feistel yapısının doğasından dolayı kaynaklanır. Ve bu konu Feistel Şifre çözme de ayrıntılı olarak tartışılacaktır.

En başta Feistel yapısında hem şifreleme hemde şifre çözme sırasında kullanılan algoritma aynıdır. Ancak şifre çözme sırasında kullanılan alt-anahtarlar ters sırada yapıya girilir. Bu şu anlama gelir; hem şifre çözme hemde şifreleme için 2 farklı algoritma gerçeklenmesine gerek yoktur. Bu DES algortimasının gerçeklenmesi işlemini kolaylaştıracaktır.

Görüldüğü üzere, şifreleme, açık metni (plain text) 2 ye parçalayarak n adet lik bir çevrimde dönüşümden geçirme işlemidir.

Şifrelemenin i. iterasyonu şu şekildedir;

$$LE_i = RE_{i-1}$$
$$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$$

(LE_i = i. iterasyonun sol yarısı
RE_i = i. iterasyonun sağ yarısı ve
F = alt anahtar K_i ve RE_{i-1} parametreleriyle çevrim fonksiyonunu gösterir.)

- **Feistel Şifre Çözme :**

Şifreme Çözme işlemide aynı prosesi kullanır. Burada şifreli metin prosesin giriş parametresi ve alt anahtarlarda ters sırada girilmesi gerekmektedir. Bu ne demek:
K_n alt-anahtarı 1. çevrimde ve K_{n-1} alt-anahtarıda 2. çevrim için kullanılarak proses devam ettirilir.
Daha önce de söylediğimiz gibi, çevrim fonksiyonu tek-yönlü bir fonksiyon olabilir. Ve sonra şifre çözme işlemi çok ilginç bir şekil alacaktır. Şimdi DES algoritmasının bir çevrimini düşünelim. Şifrelenecek mesaj 2‘ ye bölünür. 2’ye bölünen parçanın sağ yarı bölümü şifrelenmeden bırakılır ve sol yarının yeni değeri olarak bırakılır. 2’ ye bölünen sağ yarı bölüm aynı zamanda çevrim fonksiyonuna sokulur ve elde

edilen sonuç “I” sol yarı ile ex-or lanır ve sağ yarı yazmacın içine yeni oluşan veri olarak koyulur.
Ex-or işleminin tersi alınabilir bir fonksiyon olduğu göz önünde bulundurulur.
Yani ;

$$P \oplus Q = C \text{ ise } C \oplus Q = P \text{ ‘ dir.}$$

Şifre çözme işlemi boyunca sol yarıda bulunan ve şifrelenmeyen veri yer değiştirilerek sağ yarı olarak yapıda tutulur. Şifrelenen sağ yarı, sol yarı olur. Aynı alt-anahtarla çevrim fonksiyonundan geçirildiğinde şifrelenmeyen sağ yarı veride herhangi bir değer değişikliği söz konusu olmaz (ex-or işlemi nin tersliği). Yani bu Ex-or landığında şifrelenmemiş sol yarıyı veri. Böylece metin şifrelenmiş olur.

- **Feistel Yapısının Gücü:**

Feistel Şifrelemenin gücü bir kaç tasarım parametresine bağlı olarak değişir. Bu parametrelerin değişimi istenilen güvenlik düzeyine, prosesin hızına ve tasarım için gerekli alana bağlı olarak bir değişir.

Bu parametreler;

- Blok Uzunluğu
- Anahtar Boyu
- Çevrim Sayısı
- Alt-Anahtar Üretimi
- Çevrim Fonksiyonun Niteliği ‘dir.

Blok Uzunluğu:

Blok uzunluğunun artması daha yüksek güvenlik demektir. Ancak bu aynı zamanda şifreleme/şifre çözme hızı ve daha fazla saklama alanı demektir. Genel blok uzunluğu 64 bit olarak kullanılır.

Anahtar Boyu:

Daha büyük anahtar boyu brute force (kaba güç) atakları için bir zorluk çıkartır. Şuanda 128 bit uzunluğundaki anahtar boyutları kullanılmaktadır.

Çevrim Sayısı:

Şifreleme gücü çevrim sayısının arttırılmasıyla artar. DES de tipik olarak 16 çevrim kullanılır.

Alt Anahtar Üretimi:

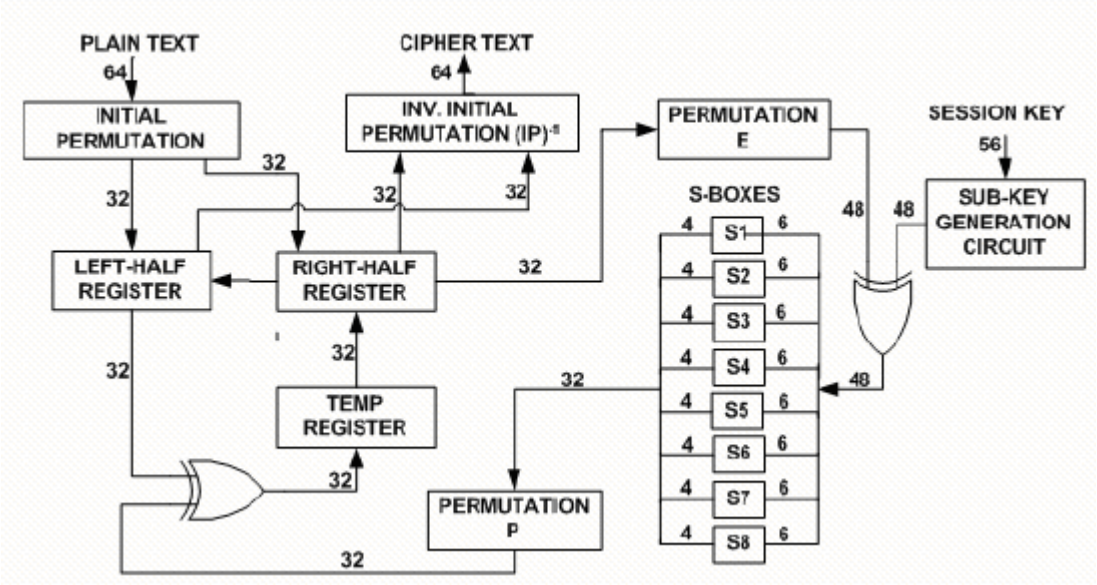
Alt-anahtar üretimi için gerekli algoritma karmaşıklaştırıldıkça, DES kriptoanalizi için daha fazla zorluk söz konusu olacaktır.

Çevrim Fonksiyonunun Niteliği:

Çevrim fonksiyonundaki karmaşıklık daha güçlü şifreleme demektir. Fakat tercih edilen foksiyonun bire-bir olmasına dikkat etmek gerekir. Bundan başka Feistel yapısı çevrim fonksiyonu olarak tek yönlü fonksiyon seçmemize olanak sağlar. Bu da kriptoanalizi zorlaştırır.
Feistel yapısının ilk olarak kullanıldığı algoritmalardan bi tanesi DES algoritmasıdır. Bu algoritma 20 seneyi aşan bir süredir kullanılmaktadır. Bundan sonraki bölümde DES tasarımı ve fonksiyonu üzerinde duracağız.

6.1. DES Donanım Gerçeklemesi

Şekilde 6.2 ‘de bir blok verinin şifrelenmesi ve şifresinin çözülmesi için gerekli algoritma akışı gösterilmiştir :



Şekil 6.2 DES Algoritması

Şifreleme işlemi 64 bit açık metnin sabit bir şablonda karıştırılması işlemi olan IP (başlangıç permutasyonu) ile başlar. IP nin sonucu 2 adet 32 bitlik yazmaçlara gönderilir. Bunlara sağ yarı yazmaç ve sol yarı register (yazmaç) denir. Bu yazmaçlar 16 iterasyon sonucunda ortaya çıkan ara sonuçları tutmaya yarar. Sağ yarı yazmacın içeriği permutasyondan geçirilir (E) ve bir Ex-or birimine alt anahtarlarla işleme girmesi için her iterasyonda gönderilir. Şekil 6.2' de görüldüğü üzere bazı bit ler 2 kez seçilerek 32 bit yazmaçların 48 bit e genişleyebilmesi sağlanır.

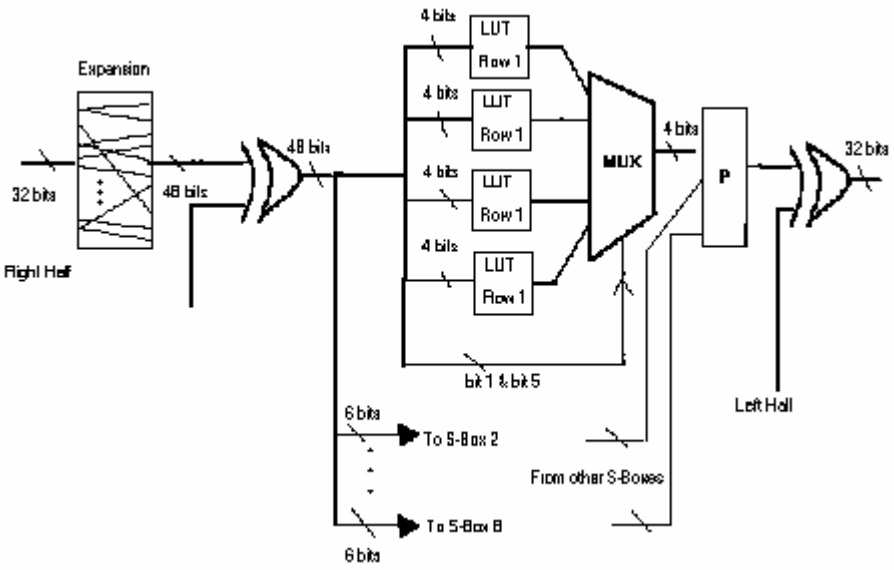
Ex-or bloğunun 48 bitlik çıktısı 8 gruba ayrılır (her grupta 6 bit olacak şekilde) ve bu ayrım 8 tane yer değiştirme belleğini adreslemek için kullanılır (S-box). Bu S-box çıktılarına P permutasyonu uygulanır ve elde edilen çıktı da sol yarı yazmacındaki içerikle ex-or lama ya tabi tutulur ve ilk iterasyonu sonuçlandırmak için Ex-or çıkışı bir geçici yazmaca yazılır.

Diğer saat darbesinde, bu geçici yazmacın içeriği sağ yarı yazmacın içine yazılır ve sağ yazmacdaki önceki veri de sol yarı yazmaca yazılır. Bu işlem 16 kez DES çevrimi boyunca tekrarlanır. 16 iterasyondan sonra sağ ve sol yarı yazmacın içerikleri sonuç permutasyonundan geçirilerek (IP-1) bu ilk permutasyonun tersidir-64 bitlik şifrelenmiş veri elde edilir.

DES algoritması donanım üzerinde gerçeklenirken uygulanan yapıların lojik yapılarını genel olarak anlatmak gerekirse;

- DES algoritmasında çokca kullanılan permutasyon işlemi donanım ile gerçeklemek zor değildir. Permutasyon işlemi donanım içinde kapı girişlerindeki hatların yönlendirilmesiyle kolayca gerçeklenebilir.
- Genişletme işleminde girişte yer alan bazı bitlerin tekrarlanması (16 kez) söz konusudur. Burada genişletme sonucunun saklanacağı yazmaca tekrarlanan bitlerin hatlarını yönlendirmek gerekecektir.
- DES algoritmasının kalbi olan S-box lar, 6 girişin Boole fonksiyonu olarak gerçeklenmesiyle elde edilmiştir. Yapıdaki tablolar zaten sabit ve heryerde bulunabilecek şekildedir. Bu yapı en temel lojik kapılar kullanılarak hızlıca gerçeklenmiştir. Hız amaçlı tasarlanan sistemşimizde DES algortiması için S-box ların her kolonu için LUT lar kullanılmış ve bundan sonra 4 kolondan istenilenin seçilmesi için Spartan içinde yer alan veri seçici (MUX) kullanılmıştır. S-box lar için kullanılan LUT lar sistemde ROM şeklinde kullanılmaktadır.

Şekilde 6.3 ‘ de DES çevrimi için uyarlanan yapının ana yapısı yer almaktadır :



Şekil 6.3 DES Çevriminin Donanım Gerçeklemesi

Hazırlanan yapıda anahtar üretme işlemi 2 tane permutasyon ve dairesel olarak bir kaydırma işlemiyle gerçeklenmiştir. Bu dairesel kaydırma işlemi Spartan-3 de yer alan LUT larda çok hızlı bir şekilde gerçeklenmiştir.
Tasarımın hızın gerekli olduğu bu yapılarda mümkün olduğunca LUT yapıları içinde yer alan kısımlar kullanılmıştır. Tüm işlemler parça parça LUT içine yerleştirilmiş ve diğer LUT lara kaydırma işlemine mümkün olduğunca az gidilmiştir. Donanımın bir diğer bölümü çevrim işleminin 16 kez tekrarlanması ve bunun kontrolünün nasıl sağlanacağıdır.

Bunun için 2 seçenek karşımıza çıkmaktadır :

Birincisi, bir çevrim için devreyi hazırlamak ve aynı devre kullanılarak bir kontrol mekanizması hazırlayarak çevrim işleminin 16 kez tekrarlanmasını sağlamaktır. Bu yöntem kullanılan lojik miktarının azalmasına yardımcı olur ama sistemin pipeline şekilinde çalışmasına engel olur.

İkinci seçenek ise çevrim için oluşturulan yapının 16 kez tekrarlanması yöntemine yönelmektir. Bu yapıda ise pipeline şeklinde çalışma imkanı olacaktır. Çalışma hızının ön planda olduğu tasarımlar için 2. seçenek kullanılması gerekmektedir. Bu tasarımda da aynı yöntem izlenmiştir. Hız veya kaplanan lojik ön planda tutularak çok çeşitli tasarım yapmak mümkün olacaktır.

6.2. DES' in Kripto Analizi:

Şifrelenecek metni veya şifreleme anahatarını bulmaya çalışma işlemine kripto analiz denir. Kripto Analizde hedef doğru anahtara ulaşmak tüm anahtar uzayını taramaktır.

- **Kaba Güç Atak (Brute Force) Yöntemi:**

DES algoritmasının anahtar boyu efektif olarak 56 bittir. Buna göre anahtar uzayı 2^{56}, yani yaklaşık olarak olası anahtar sayısı $7.2x\ 10^{16}$ dır. Bu ilk bakışda çok yüksek bir sayı gibi görünmektedir. Ancak bilgisayar teknolojisindeki gelişmeler ışığında bu

sayıdaki anahtar makul sürelerde bulunabilmektedir. DES kaba güç atak yöntemi için uygun bir algoritmadır.
1977 yıllarında Diffie Hellman [15] DES algoritmasının paralel bir yapı kullanılarak 10 saatte kırılabileceğinden bahsetmiştir.
1998 yılında DES Cracker [15] isimli bir program donanımı ile DES algoritmasını $250.000 maliyetle kırmıştır. Kırma süresi 56 saattir.

DES algoritmasını daha sağlam bir yapıda olması için arka arkaya aynı yapıdan kullanılarak güvenilirliği arttırılabilir.
Triple-DES (3DES) algoritması 168 bit anahtar kullanır ve kaba güç atak yöntemi ile kırılmaya daha dayanıklıdır.
Bundan başka tek-DES algoritması kullanılması durumunda algoritmada kullanılanacak olan anahtarlarında zayıf anahtar olmamasına dikkat edilmelidir.

7. SONUÇLAR VE TARTIŞMA

Haberleşme ve bilgisayar teknolojilerindeki gelişmelerle birlikte veri güvenliği gün geçtikçe önem kazanan çalışma alanlarından biri olmaktadır. Askeri uygulamalar, internet bankacılığı, e-posta hizmetleri, yerel alan sistemleri, uydu haberleşmeleri güvenli haberleşme hizmetlerinin beklendiği başlıca alanlardandır.

Son yıllarda, enformasyon verilerinin yoğunluğunun artmasıyla güvenlik tehditlerine karşı hızlı ve güvenilir kriptografik algoritmalar geliştirilmiştir.
Güvenlik uygulamalarında başarımı artırmak için yalnızca algoritmaların güvenilirlikleri yeterli olmayıp bu algoritmaların yeterince hızlı olması da beklenmektedir.

Algoritmaların geliştirilmesinden sonra bu algoritmalar üzerinde birçok atak yöntemleri denenerek bu algoritmaların güvenilirlikleri test edilmektedir. Atak yöntemlerinden Kaba Güç Atak Yöntemi' nde, tüm anahtar uzayı boyunca algoritmanın olası tüm anahtarlarını deneyerek anlamlı bir çıkış elde edilmeye çalışılır. Bu amaca ulaşabilmek için tasarlanan yada incelenen algoritmanın mümkün olduğunca yüksek hızda çalışması gerekmektedir. Böylece anlamlı çıkışlar makul sürelerde elde edilebilinir.

Bu çalışmada etkili ve kompakt DES gerçeklenmesi işlemini tekrardan programlanabilir ürünler üzerinde tasarlanması konusu gösterilmiştir. Yeniden programlanabilir ürünler üzerinde gerçekleme tasarımda esneklik sağladığı ve yüksek çalışma hızlarına kadar çıkabilmesi nedeniyle tercih sebebi olmuştur. VLSI ve FPGA gerçeklenmeleri tasarım stratejilerine, tasarım kaynaklarına, algoritmada

ve tasarım seviyesinde yapılan optmizasyonlara bağlı olarak yüksek veri şifreleme hızlarına ulaşmaktadırlar. Tasarımda tüm lojik yollar kısaltılmaya çalışılmış, tasarımda yavaşlamaya neden olacak tüm bölümler hızlı cevap verebilecek şekle getirilmiştir. Çalışma hızını doğrudan etkileyen paralel çalışma mantığı tasarım üzerinde gerçeklenmiştir.

Tasarlanan algoritma paralel çalıştığından, ilk 16 saat darbesi boyunca sistem herhangi bir şifreli mesaj vermezken, 17. ve daha sonra tüm saat darbeleri boyunca şifreleme yapabilmektedir. Böylece sistem paralel çalışmayla 16 çevrim süren kayan anahtar üretme prosedüründe zaman kaybetmemiş olur. Bu durum doğrudan veri şifreleme/şifre çözme hızını (Throughput) otomatik olarak 16 kat artırmıştır.

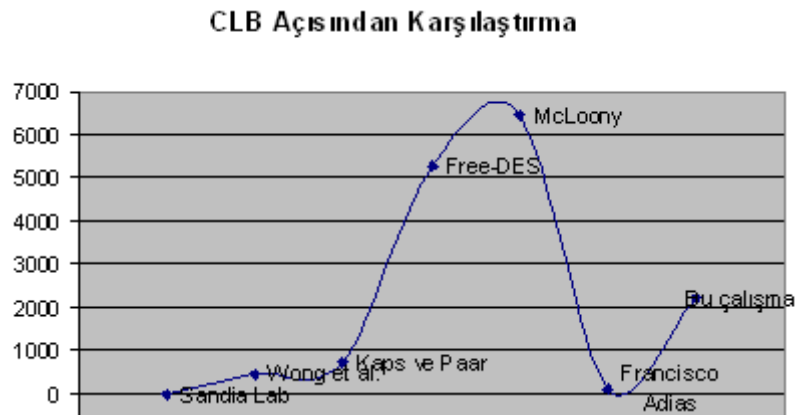
Tablo.8.1' den görüldüğü üzere tasarımımızın ulaştığı hız diğer tasarımlardan daha fazladır. Buna bağlı olarak T/A oranı da şifreleme hızı yüksek olduğundan diğer tasarımalardan daha optimumdur. Ancak paralel çalışma mantığı nedeniyle DES çevrimler tek bir çevrim olarak gerçeklenmemiştir. Hız açısından yüksek çalışma frekansı elde etmek için FPGA üzerinde kaplanan alan miktarı artmıştır.

Tablo 7.1. DES Algoritmaları Karşılaştırma Tablosu:

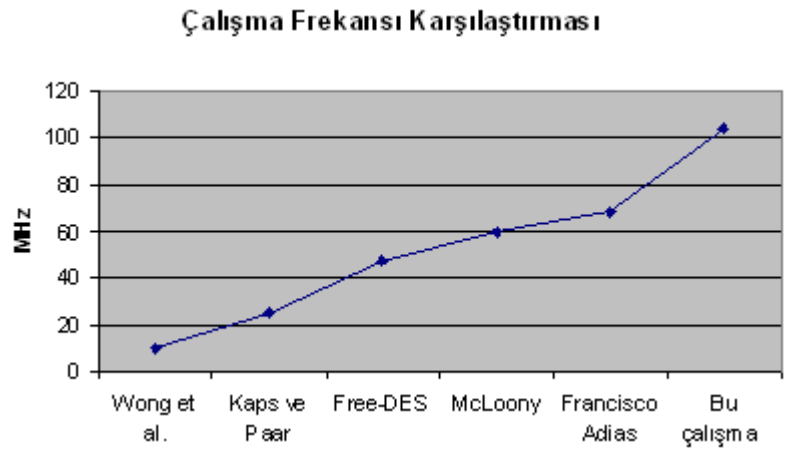
Tasarım	Kullanılan Entegre	CLB Sayısı (A)	Çalışma Frekansı (MHz)	Şifreleme Hızı (Mbit/sn) (T)	T/A Oranı
Wong et al. [10]	XC4020E	438	10	26.7	0.06
Kaps ve Paar [11]	XC4028EX	741	25.18	402.7	0.54
Free-DES [12]	XCV400	5263	47.7	3052	0.57
McLoony, McCanny[13]	XCV1000	6446	59.5	3808	0.59
Sandia Laboratuarları [8]	ASIC	-	-	9280	-
Francisco,Adias [9]	XCV400e	117	68.05	274	2.34
Bu Çalışma	**XCSpartan3s**	**2207**	**103.55**	**6627**	**3.002**

Karşılaştırma da karşımıza çıkan Sandia laboratuarının[8] ASIC gerçeklemesi en yüksek şifreleme hızına ulaşmıştır. ASIC tasarımın hız açısından FPGA' den daha yüksek performans gösterebileceği daha önceki bölümlerde ayrıntılı olarak incelenmiştir. T/A oranı açısından elde ettiğimiz sonuca en yakın değer Francisco, Adias[9] tarafından ulaşılmıştır. Bu grupta parallel çalışma mantığı kullanmış ve diğer gerçeklemelerden daha yüksek çalışma frekansı elde etmiştir. Ancak tasarlanan algoritmanın veri şifreleme hızı bizim sonuçlarımıza göre daha düşük kalmıştır. Buna karşın CLB bakımından devrenin FPGA üzerinde kapladığı alan bir hayli düşüktür. Ancak bu nedenle de 100 MHz gibi bir çalışma frekansının üstüne çıkamamışlardır.

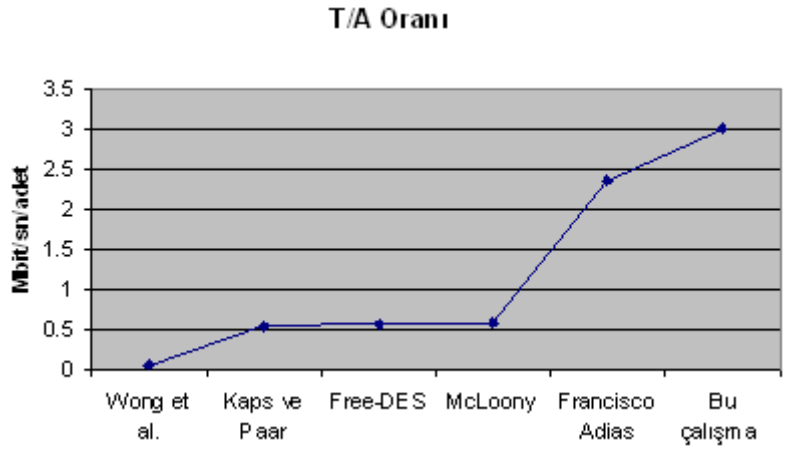
Grafik olarak karşılaştırmalı bir tablo verirsek;



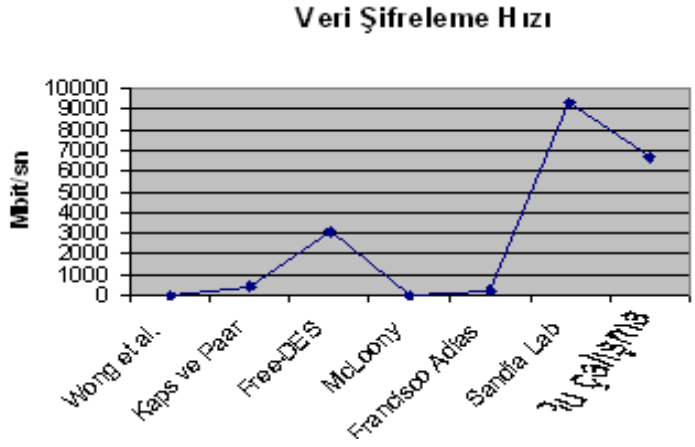
Şekil 7.1 CLB Açısından Karşılaştırma



Şekil 7.2 Çalışma Frekansı Açısından Karşılaştırma



Şekil 7.3 T/A Oranı Açısından Karşılaştırma



Şekil 7.4 Veri Şifreleme Açısından Karşılaştırma

Tasarımın yapısı daha iyi veri şifreleme performansına ulaşabilecek şekilde geliştirmek mümkün olabilir. Bunun için daha yüksek teknolojili bir FPGA kullanılarak ve de devrede indirgemeler yapılarak gerçekleştirilir.

KAYNAKLAR

[1] XILINX The Programmable Logic Data Book (2001-2003)

[2] CYPRESS Programmable Logic Data Book (1994-1995) (PLDs , CPLDs , FPGAs , Tools)

[3] Perry, D. L. VHDL, McGraw-Hill International Editions, (1991)

[4] HSU, Y.C., TSAI, K. F. LIU, J. T., LIN and E. S. , VHDL Modelling for Digital Design Synthesis, Kluwer Academic Publishers, (1995) .

[5] Shneier, B., John Wiley & Sons Inc., New York., Applied Cryptography. (1996)

[6] Vikram Pasham, Steve Trimberger., High Speed DES Encryptor/Decryptor (1999)

[7] Publication 46., FIPS (Federal Information and Processing Standards)

[8] Wilcox, D., Pierson, L., Robertson, P., Witzke, E.L., Gass, K., A DES ASIC suitable for network encryption at 10 Gbs and beyond. In: CHESS 99, LNCS 1717 (1999)

[9] Nazar A. Saqib, Francisco Rodriguez-Henriquez, and Arturo Diaz-Perez., A Compact and Efficient FPGA Implementation of the DES Algorithm, (2000)

[10] Wong, K., Wark, M., Dawson, E., A Single-Chip FPGA Implementation of the Data Encryption Standard (des) Algorithm. In: IEEE Globecom Communication Conf., Sydney, Australia (1998)

[11] Kaps, J., Paar, C., Fast DES implementations for FPGA's and its application to a Universal key-search machine. In: Proc. 5th Annual Workshop on selected areas in cryptography-Sac' 98, Ontario, Canada, Springer-Verlag, (1998)

[12] Core(2000), F.D., URL: http://www.free-ip.com/DES/. (2000)

[13] McLoone, M., McCanny, J., High-performance FPGA implementation of DES using a novel method for implementing the key schedule. IEEE Proc.: Circuits, Devices & Systems (2003)

[14] Patterson, C.: High Performance DES Encryption in Virtex FPGAs using Jbits. In: Field-programmable custom computing machines, FCCM' 00, Napa Valley, CA, USA, IEEE Comput. Soc., CA, USA, (2000)

[15] S V Raghavan, Vinay Kumar S., CS650 Cryptology and Network Security – DES , (1998)

EKLER

XILINX ISE Programı

Xilinx firmasına ait ISE programıyla tasarımlar gerçekleştirilmiştir. Tasarım VHDL veya şematik olarak ISE programıyla hazırlandıktan sonra, ISE programıyla birlikte gelen Modelsim programı hazırlanan tasarım için similasyon yapılmasını sağlar. Similasyon sonucunda (hem davranışsal hemde lojik eleman gecikmeli similasyon yapılabilir) tasarım doğru çalıştıktan sonra tasarım kullanılacak devre üzerine yerleştirilmesi ve devre içinde bulunan hatların birbirleriyle bağlanma işlemi ISE programı tarafından gerçekleştirilir.

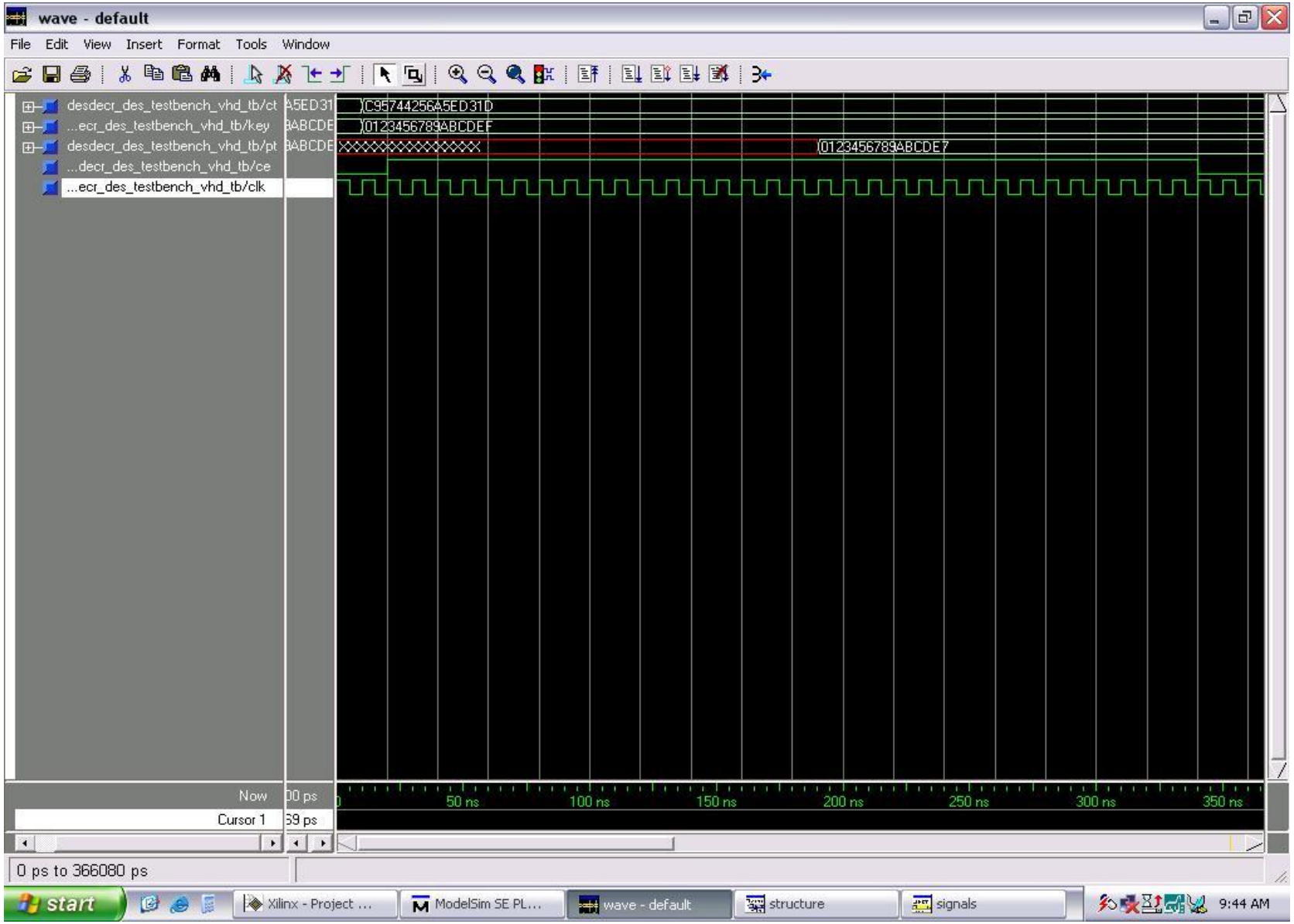
Eğer devrede de yer alan lojik yapı devreye uygunsa, FPGA entegresinin programlanacağı PROM ‘un içinde yer alacak programlama dosyası (.mcs uzantılı bir dosya) oluşturulur. Bu programlama dosyası da ISE tarafından oluşturulur ve bir RS-232’ den yükleme kablosu ile tasarım bilgisayarından FPGA entegresine program yüklenir.

ISE programı tasarım sonunda birçok dosya yaratmaktadır. Bunlar programlama dosyasının haricinde, devrenin maksimum çalışma frekansını ve devrenin kapladığı alanı gösteren dosyalarıda bulunmaktadır. Bu dosyaların önemli bölümleri ”printscreen” yapılarak ekte verilmiştir.

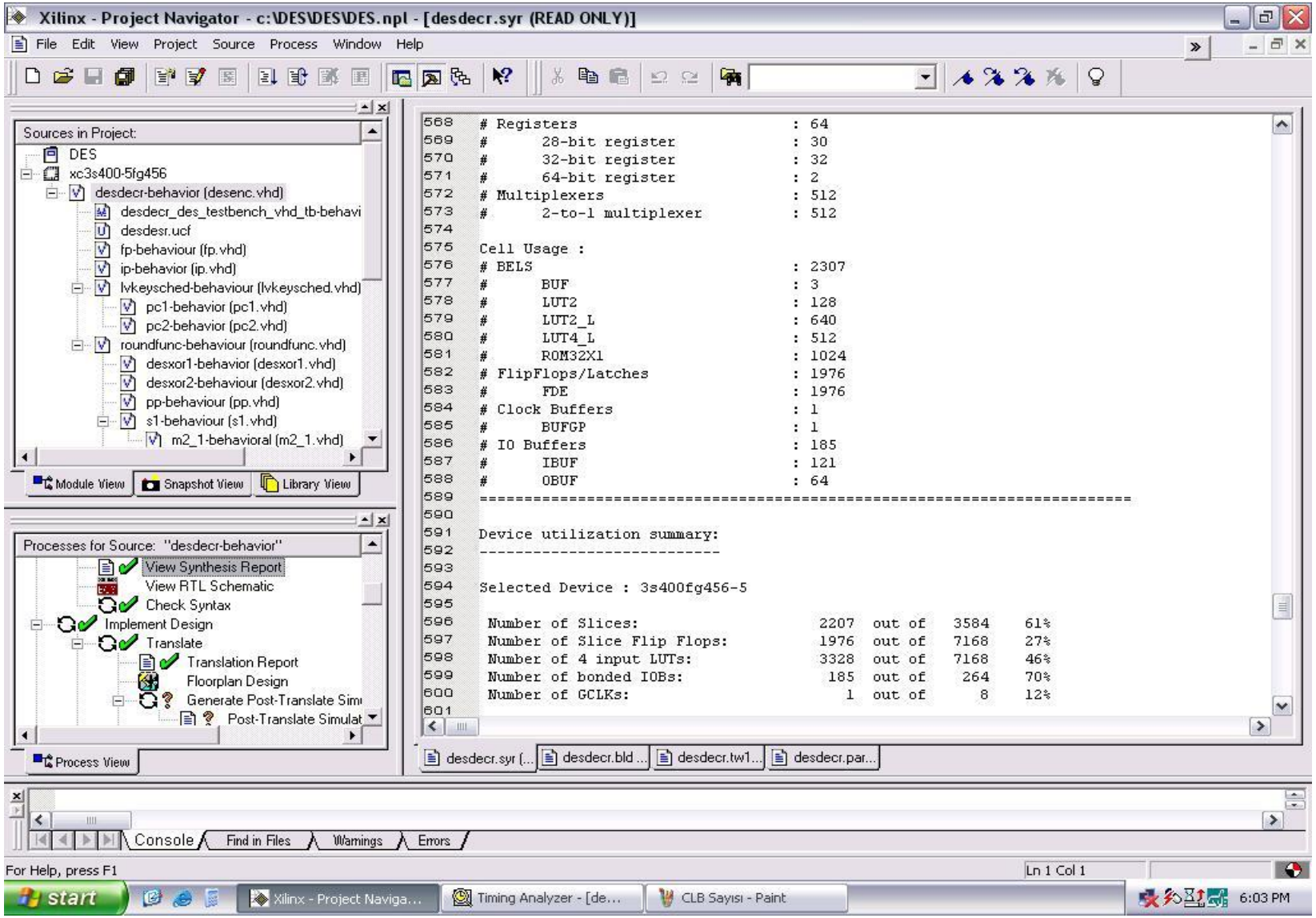
Şekil 8.1 de similasyon programında devrenin verilen bir anahtar ve açık metin ile oluşturduğu şifreli metni görmekteyiz. Burada dikkat edilecek husus ilk 16 saat darbesi boyunca algritmanın bir çıkış üretmemesi, ancak bundan sonra paralel çalışma mekanızması sayesinde her saat işaretinin yükselen kenarında bir şifreli mesaj üretilecektir.

Şekil8.2 de devrenin CLB açısından kapladığını alanı göstermektedir.

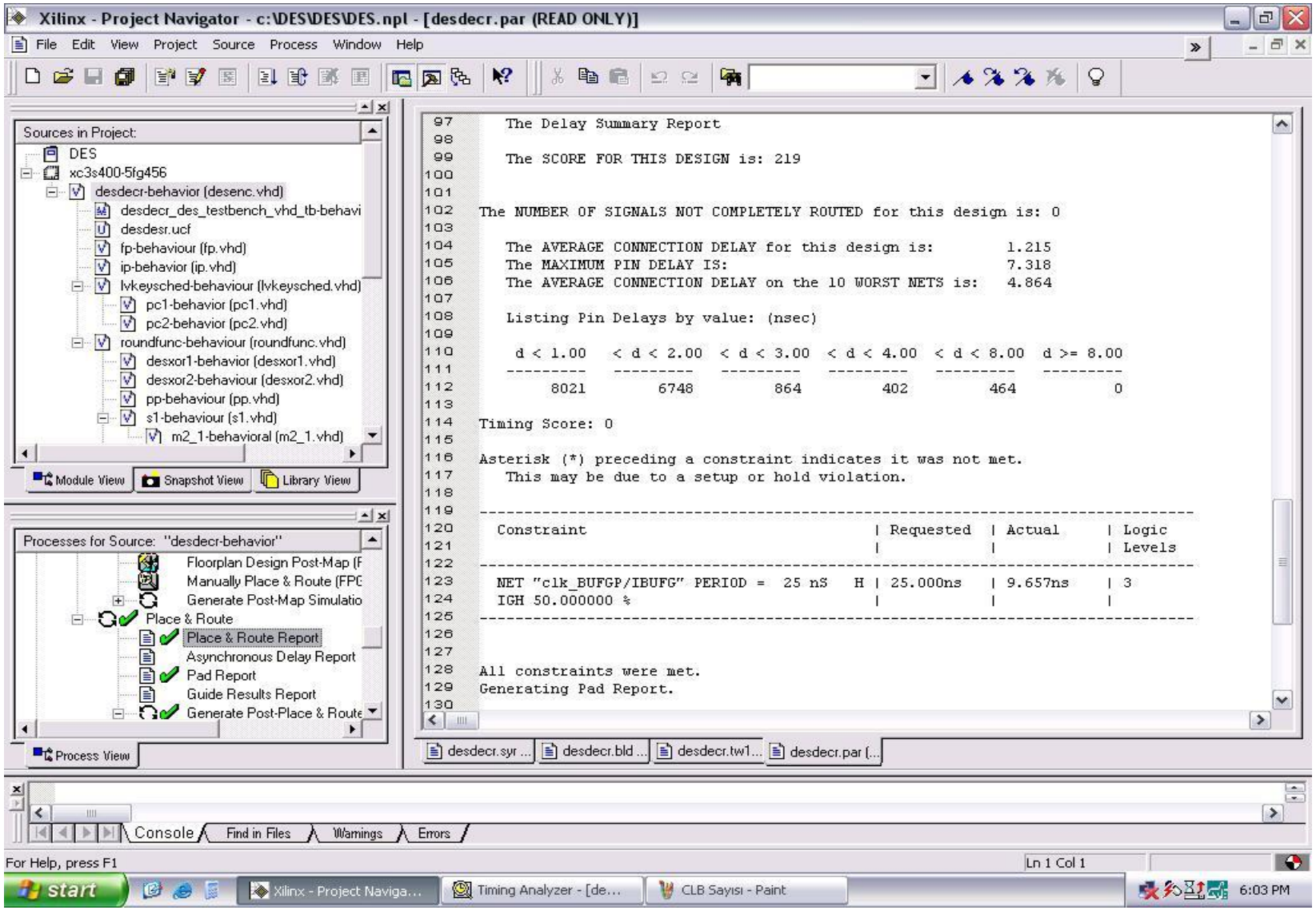
Şekil8.3 de ise maksimum çalışma frekansı dosyasının similasyon programında gösterimi yer almaktadır.



Şekil 8.1 Veri Şifreleme Noktası



Şekil 8.2 Kullanılan CLB Miktarı Görünümü



Şekil 8.3 DES Tasarımı Çalışma Frekansı

ÖZGEÇMİŞ

Bora Emiroğlu, 1978 yılında İstanbul'da doğdu. Orta ve lise öğrenimini Özel Akasya Koleji'ini birincilikle 1996 yılında tamamladı. Aynı yıl içerisinde İTÜ Elektrik-Elektronik Fakültesi, Elektronik ve Haberleşme Mühendisliği Bölümü'ne girmeye hak kazandı. 1998 yılında Kontrol ve Bilgisayar Mühendisliği Bölümü'nde ÇAP programına katıldı. Temmuz 2000'de Elektronik ve Haberleşmeyi ve Temmuz 2002'de de Kontrol ve Bilgisayar Mühendisliği bölümünü tamamlayarak mezun oldu.

Eylül 2001'de Savunma Teknolojileri yüksek lisans programına kayıt oldu ve Tübitak-UEKAE'de (Ulusal Elektronik ve Kriptoloji Araştırma Enstitüsü'nde) araştırmacı olarak çalışmaya başladı. Halen Tübitak-UEKAE' de çalışmaya devam etmektedir.

Başlıca ilgi alanları kriptografik donanım (FPGA tasarımları) gerçeklemeleri ve kriptografik cihazların güvenlik özellikleridir.