İÇİNDEKİLER

1. GİRİŞ

1.1 Giriş ve çalışmanın amacı

Gelişen teknolojiyle internetin kullanımı her geçen gün daha da yaygınlaşmakta olup, internette iletilen veri paketleri birçok dışarıya açık ağlardan geçmektedir. Gizli ve başkaları tarafından ulaşılması istenilmeyen bilgilerin internette bir yerden bir yere iletilmesi ciddi anlamda güvenlik kaygısını da beraberinde getirmektedir. Bu tür bilgileri korumak mümkün olmadıkça, gizli ve şahsi yazışmalarda bulunmak da mümkün olmayacaktır. Bilgi güvenliği, başkası tarafından dinlenme, bilginin değiştirilmesi, kimlik taklidi gibi tehditlerin ortadan kaldırılması ile sağlanır ve bu amaç için geliştirilen temel bilim kriptografidir. Kriptografi, bilgi güvenliğini sağlayan temel bilim dalıdır. Güvenilirlik, veri bütünlüğü, kimlik doğrulama gibi bilgi güvenliği konularıyla ilgilenen matematiksel yöntemler üzerine yapılan çalışmalar, kriptografinin önemli konularıdır.

Kriptografi genel olarak şu ana konularla ilgilenir:

Gizlilik: Bilgi istenmeyen kişiler tarafından anlaşılamamalıdır.

Bütünlük: Bir iletinin alıcısı bu iletinin iletim sırasında değişikliğe uğrayıp uğramadığını öğrenmek isteyebilir. Dışarıdan biri, doğru iletinin yerine yanlış bir ileti koyma şansına erişmemelidir. Saklanan veya iletilmek istenen bilgi farkına varılmadan değiştirilememelidir.

Reddedilemezlik: Bilgiyi oluşturan ya da gönderen, daha sonra bilgiyi kendisinin oluşturduğunu veya gönderdiğini inkâr edememelidir. Bir gönderici daha sonrasında bir ileti göndermiş olduğunu yanlışlıkla reddetmemelidir.

Kimlik belirleme: Gönderen ve alıcı, birbirlerinin kimliklerini doğrulayabilmelidir. Dışarıdan biri, bir başkasının kimliğine bürünme şansına erişmemelidir.

Bir kriptografik sistem, bilgi güvenliğini sağlamak için bir araya getirilmiş birçok küçük yöntemler bütünlüğü olarak görülebilir. Bu yöntemlerin içinde belirli bir algoritma kullanılarak anahtar üretme yöntemi yer alır. Bir algoritmaya dayanılarak üretilen anahtar ile iletilmek istenen mesaj şifrelenir ve alıcı tarafa gönderilir. Alıcı tarafta ise yine bir şifre kullanılarak şifrelenmiş metin çözülür ve gönderilmesi istenen mesaj elde edilir. Gönderen tarafın mesajı şifreleme işlemine şifreleme (encryption), alıcı tarafta yapılan şifre çözme işlemine ise şifre çözme (decription) denir ve şekil 1.1 ile gösterilmiştir.

Şekil 1.1: Şifreleme ve şifreyi çözme işlemleri diyagramı

Şifreleme teknikleri, blok şifreleme ve dizi şifreleme olarak ikiye ayrılır. Dizi şifrelemede bir anda mesajın bir biti şifrelenirken, blok şifrelemede bir anda birden fazla bitten oluşan bir parçası, blok halinde şifrelenir.

Dizi şifreleme de kendi içinde senkronize ve kendiyle senkronize olmak üzere ikiye ayrılır. Senkronize şifrelemede şifre algoritmasında anahtar ve başlangıç vektörü kullanılırken, kendiyle senkronize şifrelemede ise bunlara ek olarak daha önceden şifrelenmiş şifreli mesaj bitleri de kullanılmaktadır.

Sahada programlanabilir kapı dizileri (FPGA) lojik bloklardan ve flip-flop'lardan oluşan ve bunların arasında elektriksel programlanabilir bağlantılar bulunduran iki boyutlu bir dizidir. Aralardaki bağlantılar, elektriksel programlanabilir anahtarlardır. FPGA, kullanıcının, hem aradaki bağlantıları programlayabildiği, hem de içinde bulunan her bir lojik bloğu ayrı ayrı programlayabildiği bir tümdevre elemanıdır.

Mosquito algoritması, şifreleme yöntemlerinden biri olan "kendiyle senkronize dizi şifreleme" yöntemine örnek bir algoritmadır. Bu algoritmada, anahtar olan K ve başlangıç vektörü olan IV (initialization vector) kullanılarak yedi aşamada bir bit anahtar üretilmektedir. Bu işlemler için kullanılmak üzere, devrenin girişini her defasında farklı kılacak bir ötelemeli kaydedici, giriş ile aldığı bilgiyi güncelleyebilmesi için bir koşullu tamamlayan ötelemeli kaydedici ve yedi aşamanın her biri için yedi adet kombinezonsal devre tasarlanmıştır.

Projede, Mosquito algoritmasının çok yüksek seviyeli donanım tanımlama dilleri (VHDL) kullanılarak tanımlanması ve FPGA üzerinde gerçeklenmesinin aşamaları anlatılacaktır.

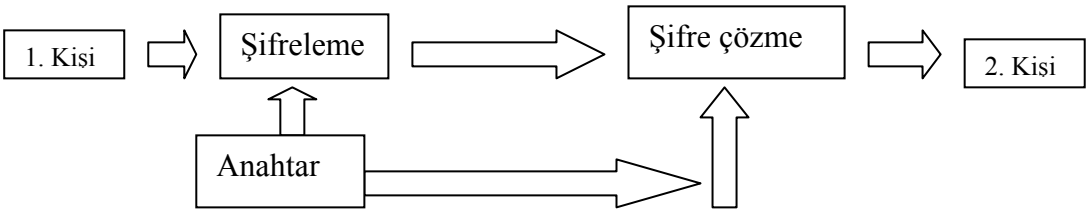
Projenin ikinci bölümünde kriptografi ve bir kriptografik sistem türü olan dizi şifreleme hakkında detaylı bilgi verilmiştir. Üçüncü bölümde algoritmanın gerçeklenmesinde kullanılacak FPGA'lar hakkında daha geniş bilgi verilmiş, yapıları ve programlanmaları anlatılmıştır. Projenin dördüncü bölümünde ise Mosquito algoritması, algoritmanın gerçekleme adımları ve sonuçları verilmiştir.

2. KRİPTOGRAFİ VE DİZİ ŞİFRELEME

2.1 Kriptografi

Kriptografi, okunabilir durumdaki bir bilginin istenmeyen taraflarca okunamayacak bir hale dönüştürülmesinde kullanılan tekniklerin tümü olarak görülebilir. Bu teknikler, bir bilginin iletimi esnasında karşılaşılabilecek aktif ya da pasif ataklardan bilgiyi, dolayısıyla bilgi ile beraber bilginin göndericisi ve alıcısını da koruma amacı güderler.

Bir başka deyişle kriptografi, iki kişinin, güvenli olmayan bir kanal üzerinden üçüncü bir kişinin anlayamayacağı bir şekilde haberleşebilmesi amacıyla geliştirilen tekniklerdir. Burada güvenli olmayan kanal telefon ya da şebeke hattı olabilir. Gönderilmek istenen mesaj, önceden belirlenmiş bir anahtar ile şifrelenerek kanal üzerinden gönderilir ve alıcı tarafta aynı anahtar ile şifre çözülür. İki tarafın da ihtiyaç duyduğu anahtar ise ayrı olarak güvenli bir kanal üzerinden gönderilmelidir. Bu gönderim ve şifreleme işlemleri Şekil 2.1'de gösterilmiştir [2].



Şekil 2.1: Haberleşme kanalı

Kullanılan anahtarın K, şifreleme fonksiyonunun e(x,K), şifre çözme fonksiyonunun ise d(y,K) olduğu düşünülürse şifrelenmiş metin;

$$Y = Y1\ Y2\ Y3\ \ldots..\ Yn = e(X1,K)\ e(X2,K)\ e(X3,K)\ \ldots.\ e(Xn,K) \qquad (2.1)$$

şifresi çözülmüş metin ise;

$$X = X1\ X2\ X3\ \ldots..\ Xn = d(Y1,K)\ d(Y2,K)\ d(Y3,K)\ \ldots.\ d(Yn,K) \qquad (2.2)$$

şeklinde elde edilir.

Kullanılan anahtarlar belli bir algoritmaya göre oluşturulur. Kullanılan algoritmanın değerlendirilmesindeki önemli etkenler; güvenlik seviyesi, fonksiyonellik, işlem metodu, performans ve gerçekleme kolaylığıdır [3].

Algoritmalar şifreleme ve çözme işlemlerini kontrol etmek için bir anahtar kullanırlar. Anahtar temelli algoritmaların iki çeşidi vardır; simetrik (gizli-anahtar) ve asimetrik (açık-anahtar) algoritmaları. Simetrik algoritmalar şifreleme ve çözme işlemleri için aynı anahtarı kullanırken, asimetrik algoritmalar şifreleme ve çözme için farklı anahtar kullanırlar. Çözme anahtarı şifreleme anahtarından elde edilemez.
Simetrik algoritmalar dizi şifreleme (stream cipher) ve blok şifreleme (block cipher) olarak ikiye ayrılır. Dizi şifrelemede belli bir anda bir bitlik metin şifrelenebilirken, blok şifrelemede ise pek çok bit alınıp (tipik olarak 64 bit) bunlar tek bir birim olarak şifrelenirler [7].

2.2 Dizi Şifreleme

Dizi şifreleme, mesajdaki her harfin özel bir algoritma kullanılarak anahtardan üretilen bir anahtar dizisi ile sırayla şifrelenmesidir [3].

Dizi şifrelemede mesaj metni $X_1 X_2 X_3 \ldots\ldots X_n$ ve şifre dizisi $Z_1 Z_2 Z_3 \ldots\ldots Z_n$ iken $1 \leq i \leq n$ için X_1 biti Z_1 ile, X_2 biti Z_2 ile şifrelenir.

En genel anahtar üretme yönteminde anahtar, belirli bir algoritma ile oluşturulur. Algoritma sonucunda belirli uzunluktaki K anahtarından bir anahtar dizisi oluşturulur. Bu sayede bit sayısı olarak daha kısa bir şifre (K), daha uzun ve güvenli bir şifre dizisine dönüşür. Bu dönüşüm sonucunda (2.3) denkleminde görülen n bitlik Z şifre anahtar elde edilir.

$$G(K) = Z1\ Z2\ Z3\ \ldots..\ Zn \tag{2.3}$$

Birçok şifreleme algoritması, bir bit daha önce üretilen bitlere bağlı olacak şekilde tasarlanır. Burada amaç, şifrelemenin daha güvenli olması için, periyodu daha büyük anahtar dizileri elde etmektir.

Şifrelenecek metnin bitlerinin ve şifre bitlerinin 1 ya da 0 olduğu ve metin biti ve şifre arasında yapılan işlemin modül 2 ye göre toplama işlemi olduğu düşünüldüğünde (2.4) ve (2.5) denklemleri gereği bit düzeyinde uygulanan işlemin "Ayrıcalıklı Veya" (XOR) işlemi olacağı açıkça görülür [1].

$$e(x) = x + z \tag{2.4}$$

$$d(y) = y + z \tag{2.5}$$

Bir anahtardan bir algoritmaya dayanarak bir anahtar dizisi oluşturmak ve bu dizinin bitleri ile mesaj metninin bitleri ayrıcalıklı veya işlemine tabi tutularak yapılan bir şifreleme yöntemi olan dizi şifreleme iki türe ayrılır. Bu türlerden birincisi olan senkronize dizi şifrelemede üretilecek anahtar sadece anahtar K'ya ve başlangıç vektörü (initialization vector) olan IV'ye bağlıdır. Diğer tür olan kendiyle senkronize dizi şifrelemede (Self-synchronizing stream encryption) ise üretilecek anahtar, K ve IV'nin yanı sıra önceden üretilmiş anahtar ile şifrelenmiş metine de bağlıdır [7]. Bölüm 4'te anlatılacak olan Mosquito algoritması, kendiyle senkronize dizi şifreleme algoritmasına bir örnek niteliğindedir.

Bir t anında şifrelenecek mesaj m^t, üretilmiş şifre z^t iken şifrelenmiş metin denklem (2.6)'dan uyarınca c^t olsun.

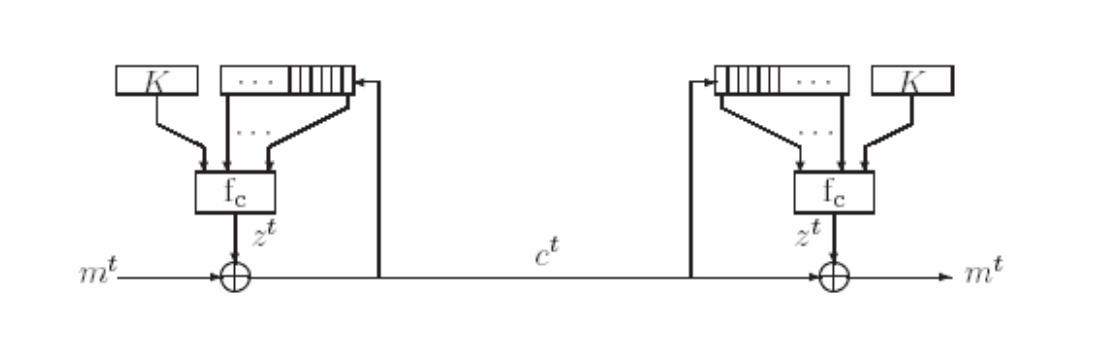
$$c^t = m^t + z^t . \tag{2.6}$$

Kendiyle senkronize dizi şifrelemenin en önemli özelliği, denklem 2.7 den de görüldüğü gibi, üretilen anahtar biti olan z'nin, anahtar K'nın yanı sıra daha önceden şifrelenmiş mesaj bitleri olan $c^{t-1} \ldots c^{t-nm}$, ye bağlı bir fonksiyon ile üretiliyor olmasıdır [1].

$$z^t = f_c[K](c^{t-nm} \ldots c^{t-1}) . \tag{2.7}$$

Bir başka deyişle algoritma ile üretilecek anahtar, yine o algoritma ile üretilmiş anahtar ile şifrelenmiş metin ile üretilmektedir. Kendiyle deyimi bu sebepten ötürüdür.

Eşitlik 2.7'deki n_m, anahtar bitinin üretilmesi için gerekli olan şifrelenmiş metnin bit sayısıdır. Kendiyle senkronize dizi şifrelemenin blok diyagramı olan Şekil 2.2'den de görüleceği gibi anahtar, anahtar (K) ve daha önceki anlarda şifrelenmiş metnin bir fonksiyonudur.



Şekil 2.2: Kendiyle senkronize dizi şifreleme blok diyagramı

Eşitlik 2.7'den görüleceği gibi bir t anında z'nin üretilebilmesi için daha önceki anlarda elde edilmiş n_m adet c'ye ihtiyaç vardır. Fakat şifreleme işlemine başlanıldığında n_m adet c elde edilmiş değildir ve başlangıçta kullanılmak üzere, n_m uzunluğunda bir bit dizisi gerekir. Bu problemi karşılamak için, şifrelenmiş metin gibi davranan bir "başlangıç vektörü" (IV) denilen, n_m uzunluğunda bir bit dizisi kullanılır. Böylece şifreleme işlemi için gerekli olan n_m adet c sağlanmış olur. IV, sadece başlangıç ihtiyacını karşılamak için kullanılır. n_m adet şifre üretildiğinde ise IV'ye olan ihtiyaç biter ve üretilen anahtarlar ile şifrelenmiş mesaj (c) kullanılmaya başlanır [1]. Şifreleme ve şifre çözme işlemleri için IV aynı anahtar gibi, her iki tarafta bulunmalıdır fakat şifreden farklı olarak gizli tutulmasına ihtiyaç yoktur.

En genel haliyle bir mesaj m'nin K anahtarı ve IV başlangıç vektörü ile şifrelenmesi aşağıdaki 6 adım ile anlatılabilir:[7]

1. Ayrıştırma: Bu aşamada mesaj m, $m_1m_2m_3....m_n$ formuna ayrıştırılır.
2. Başlangıç: Şifre üretecine anahtar K ve başlangıç vektörü IV verilir. Sayıcı 0'a ayarlanır
3. Anahtar üretme: anahtar üreteci $z_1z_2z_3....z_n$ anahtar bloğunu üretir.
4. Anahtar bloğu ile mesaj şifrelenerek her m_i ve z_i için bir c_i elde edilir.
5. Döngü: i<n ise 3. adıma dönülür ve i = n oluncaya yani tüm mesaj şifreleninceye kadar adımlar tekrarlanır.
6. Çıkış: Şifrelenmiş metin olan $c_1c_2c_3....c_n$ elde edilir.

3. SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ

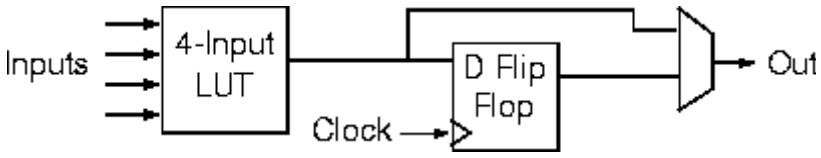
3.1 Sahada Programlanabilir Kapı Dizileri Hakkında Genel Bilgi

Sahada Programlanabilir Kapı Dizileri (Field Programmable Gate Array, FPGA) yaygın olarak kullanılan, geniş uygulama alanlarına sahip programlanabilir tümdevrelerdir. FPGA kullanımıyla, temel lojik kapıların (AND, OR) ya da çözücü gibi daha karmaşık yapıdaki devre elemanlarının fonksiyonelliği artırılmaktadır.

FPGA'lar, programlanabilir lojik bloklar ve ara bağlantılardan oluşur. Lojik blokların ve ara bağlantıların imalat aşamasından sonra tasarımcı tarafından programlanabilmesi sebebiyle sahada programlanabilir adını taşımaktadır [5]. Tasarımcının ihtiyacı olan lojik fonksiyonları gerçekleştirebilmesi amacıyla sahada programlanabilir olarak üretilmiştir. Kullanıcının tasarladığı lojik devreye göre, tümdevre üreticisi tarafından sağlanan bir yazılım sayesinde lojik bloklar ve aralarındaki bağlantılar programlanır [4]. Tasarım sırasında kullanıcıya sağladığı esneklik, düşük maliyet ve hızlı ilk üretme özelliği ile FPGA'lar sayısal tasarım ortamlarının vazgeçilmez yapıları haline gelmiştir. Uygulama alanlarından bazıları sayısal işaret işleme, kriptografi, uzay, savunma ve tıbbi görüntülemedir [5].

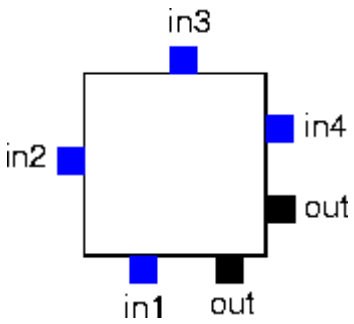
3.2 Sahada Programlanabilir kapı dizilerinin yapısı

Tipik FPGA lojik bloğu Şekil 3.1'de görüldüğü gibi 4 giriş (Look up table, LUT) ve bir flip-flop'tan oluşur. FPGA lojik bloğunun giriş-çıkışları ise Şekil 3.2'de gösterilmektedir.



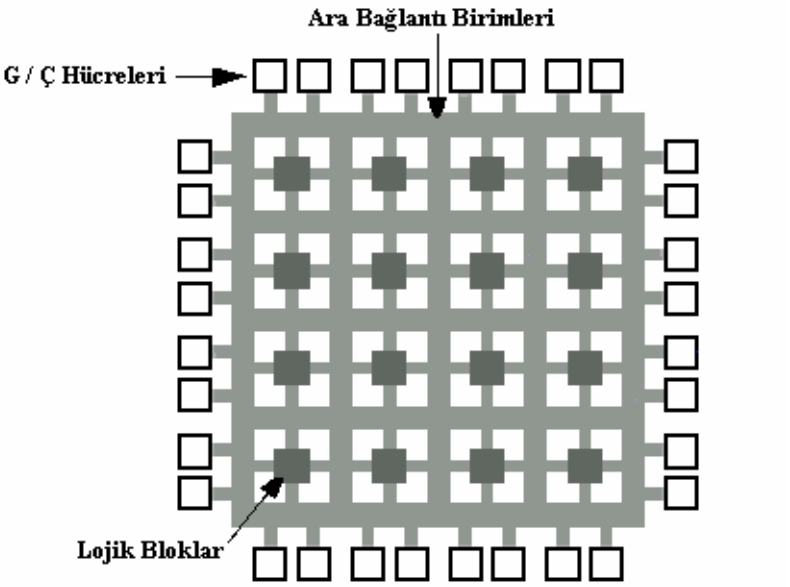
Şekil 3.1: FPGA lojik bloğu

FPGA, düzenlenebilir lojik bloklar (configurable logic blocks, CLB), giriş / çıkış blokları (Input / Output Blocks, IOB) ve ara bağlantılar olmak üzere üç tane önemli düzenlenebilir elemana sahiptir [3]. FPGA'ların programlanması ara bağlantıların nasıl olacağını belirleme ve FPGA bellek hücrelerinin programlanması işlemidir.



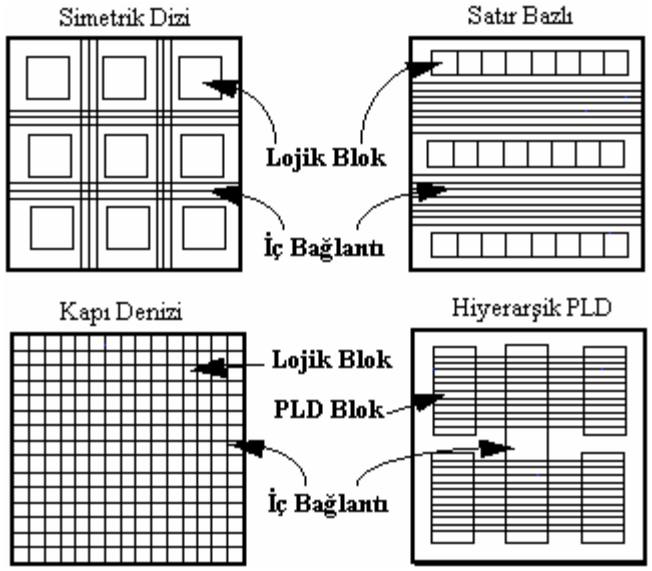
Şekil 3.2: FPGA bloğu (giriş- çıkış)

Şekil 3.3'te FPGA da bulunan lojik bloklar ve ara bağlantı birimleri görülmektedir.[3]



Şekil 3.3: FPGA genel yapısı

FPGA'lar bağlantı çeşitlerine göre simetrik dizi, sıra tabanlı, hiyerarşik PLD ve kapı denizi olmak üzere dörde ayrılırlar. (Bkz. Şekil 3.4). [6]



Şekil 3.4: Bağlantı şekillerine göre FPGA'lar

3.3 Sahada Programlanabilir kapı dizilerinin programlanması

FPGA'lar genellikle yüksek seviyeli donanım tanımlama dilleri (Hardware Description Language, HDL) kullanılarak programlanır. Yazılan program derlenerek benzetimi yapılır. Benzetim aşamasında beklenilen sonuç alındıktan sonra sentezleme aşamasına geçilir. Sentezleyici, kullanıcının belirtmiş olduğu FPGA elemanına göre gerekli bağlantıları çıkartır ve kullanıcıya bu bağlantıları liste halinde sunar. Yerleştirme ve yollandırma sonrası istenilen sonuçlar elde edildikten sonra FPGA'nın programlanması aşamasında kullanılacak bit dizisi, üreticinin sağladığı yazılımla elde edilir. FPGA'nın uygun donanım kullanılarak programlanmasıyla tasarım tamamlanır [3-4, 6]. En genel anlamda bir tasarım süreci ve tasarlanan fonksiyonun bir FPGA'ya yüklenmesi beş başlık altında toplanabilir:

1. Sistem tasarımı: Bu aşamada tasarımcı FPGA'ya uygulanacak olan fonksiyona ve bu fonksiyonun sistemin geri kalan kısmıyla nasıl bütünleneceğine edileceğine karar verir.
2. Giriş-çıkış tümleştirimi: FPGA'nın giriş ve çıkışlarının sisteme yani devre tümleştirme aşamasıdır.
3. Tasarımı tanımlama: Bu aşamada tasarımcı, tasarladığı fonksiyonu şemalar ya da HDL kullanarak tanımlar.
4. Sentezleme: Bu aşamada en iyileme, lojik blokların yerleşimi ve ara bağlantıların belirlenmesi işlemleri yapılır.
5. Tasarımı doğrulama: Bu aşama test aşamasıdır. Elde edilen sonuçların benzetim sonuçlarıyla karşılaştırılması bu aşamada yapılır.

4. MOSQUITO ALGORİTMASI VE GERÇEKLENMESİ

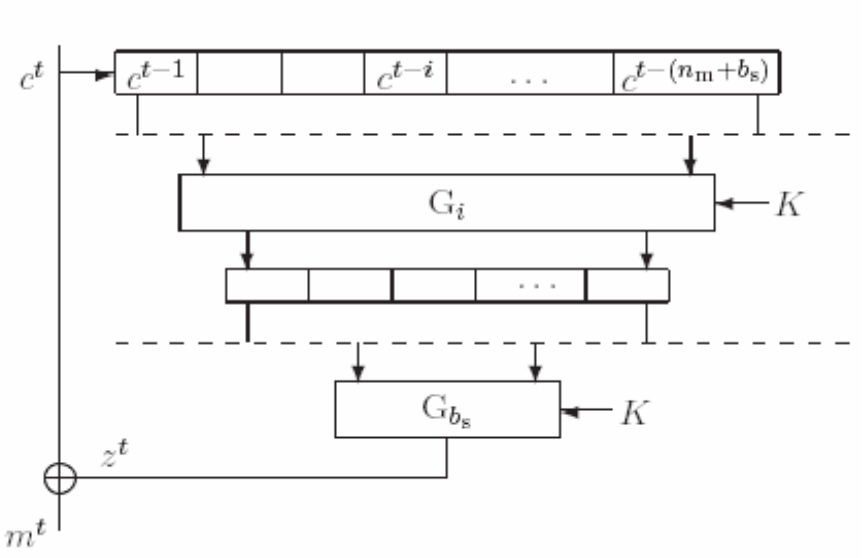
4.1 Mosquito Algoritması

Kendiyle senkronize dizi şifreleme tipine bir örnek olan Mosquito algoritmasında, koşullu tamamlayan ötelemeli kaydedici (Conditional complementing shift register, CCSR) kullanılmaktadır. İş hattı kullanılarak gerçeklenen yedi aşama vardır. Bu yüzden bu bölümde önce CCSR ve ardışık düzen hakkında genel bilgi verilecektir. Daha sonra Mosquito algoritmasında kullanılan CCSR ve uygulanan ardışık düzen anlatılacaktır.

4.1.1 İş Hattı (Pipelining)

Anahtar, dayanıklı olması amacıyla birçok aşamadan geçirilerek üretilmek istenir. Bu aşamalar Şekil 4.1' de görüldüğü gibi, G_i ile gösterilen b_s adet aşamadır.

Donanımsal olarak, her aşama bir kombinezonsal devre ve ara sonucu saklayan bir bellekten oluşur. Devrenin b_s aşamadan oluşması, ilk aşama ile son aşama arasında b_s zaman farkı oluşmasına neden olur. Bu zaman farkı dolayısı ile üretilen z^t, Bölüm 2'de bahsedildiği gibi c^{t-nm}… c^{t-1} 'e bağlı değil de c^{t-nm}… $c^{t-(bs+1)}$ 'e bağlı hale gelir. Ardışık zaman nedeniyle n_m uzunluğundaki giriş hafıza belleği, $n_m + b_s$ uzunluğuna çıkar. Fakat bu artış, üretilen z'nin fazla olan b_s adet bitten bağımsız olacağı nedeniyle giriş uzunluğunu değiştirmez. b_s, şifre fonksiyonunun gecikmesi olarak da bilinir [1].



Şekil 4.1: Şifre oluşturuluş aşamaları

Bu iş hattının ilk aşamasında giriş, ötelemeli kaydedicide bulunan $n_m - b_s$ bitlik şifrelenmiş metindir. Böylece üretilecek olan z'nin, sadece bu $n_m - b_s$ bitlik diziye ve anahtar olan K'ya bağlı olması garantilenmiş olur. Ötelemeli kaydedicinin n_m uzunluğunda bir iç bellekli bir sonlu durum makinesi ile değiştirilmesi, devrenin yayılma özelliklerini geliştirecektir. q bir durum ve G durum güncelleme fonksiyonu olarak düşünüldüğünde denklem 4.1'den de görüleceği gibi bir t+1 anındaki durum, t anındaki duruma ve yine t anındaki şifrelenmiş mesaj bitine bağlı olacaktır.

$$qt+1 = G\ (qt, ct) \qquad (4.1)$$

Durum q, bağlı olduğu şifre mesaj biti sayısına eşit iç belleklerle ilişkilendirilmiş parçalara ayrılırsa, j adet şifrelenmiş metin bitine bağlı bir durum elemanı q^j ile gösterilir. q_i^j ise q^j'nin i

inci parçasıdır. Burada c, hiçbir bite bağlı olmayan, yani q durumunun j=0 indeksli parça olarak düşünülebilir. Sonuç olarak bir t anındaki q_i^j daha büyük j indeksli durum elemanlarından bağımsız, daha küçük j indeksli durum elemanlarına bağımlıdır. q durumu, 1 ila n_m arasındaki uzunluklarda belleklere sahip alt durumlara ayrılmıştır ve her durum parçası için durum güncelleme fonksiyonu, $j \leq n_m$ için denklem 4.2'deki halini alır.

$$q^{(j)\ t+1} = G\,[K]_i^{(j)}\,(c^t, q^{(1)t}, \ldots., q^{(j-1)\,t}) \qquad (4.2)$$

4.1.2 Koşullu tamamlayan ötelemeli kaydedici (CCSR)

Koşullu tamamlayan ötelemeli kaydedici, sonlu iç bellekli sonlu durum makineleri için geliştirilmiş bir çözümdür. Bu bellek ile durum güncelleme fonksiyonu denklem 4.3'teki halini alır.

$$q^{(j)\ t+1} = q^{(j)\ t+1} + E\,[K]^{(j)}\,(q^{(j-2)t}, \ldots., q^{(1)\,t}, c^t) \qquad (4.3)$$

Burada t+1 anındaki q^j, $q^{j-1'}$ in eski değerinin ve bir Boole fonksiyonunun toplamına eşit olduğundan belleğe koşullu tamamlayan ötelemeli kaydedici ismi verilmiştir.

4.1.3 Mosquito CCSR

Mosquito, anahtar uzunluğu 96, bellek uzunluğu 105 ve fonksiyon gecikmesi 9 olan bir kendiyle senkronize dizi şifreleme fonksiyonudur. Mosquito algoritmasında IV 105 bit uzunluğunda olup, üretilecek olan şifre z'nin genel fonksiyonu denklem 4.4'teki gibidir.

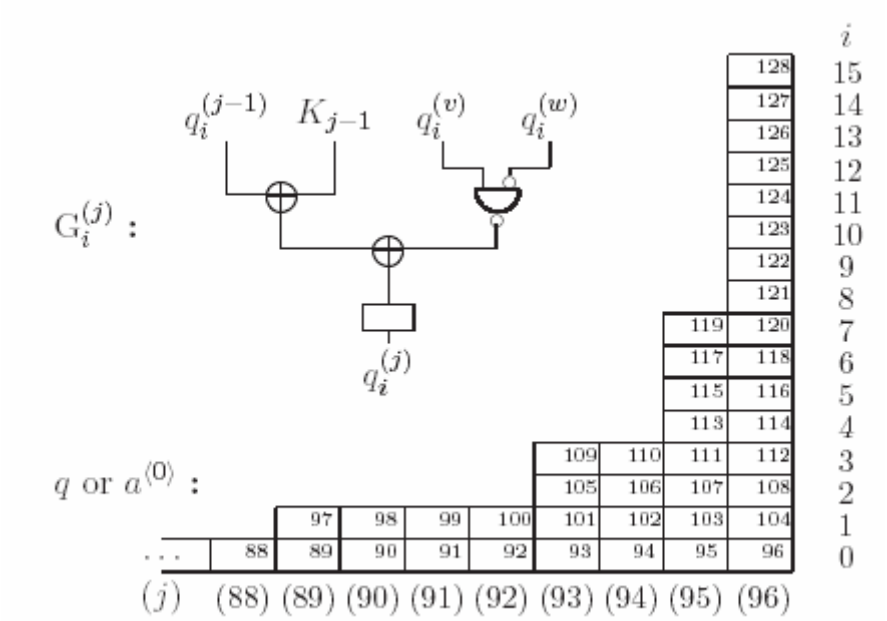
$$z^t = f_c\,[K]\,(c^{t-105} \ldots\ldots c^{t-10}) \qquad (4.4)$$

Mosquito CCSR'de 96 bit uzunluğunda iç bellek olup, tüm CCSR'nin içeriği 128 bite genişletilmiştir. Bu genişleme, her iç belleğin j = 89'dan itibaren 2, j = 93'ten itibaren 4, j = 95'te 8 ve son olarak j = 96'da 16 bitlik bir beleğe genişletilmesiyle elde edilir. Bu durum Şekil 4.1'de gösterilmektedir.

Basitlik, alan ve kapı gecikmeleri açısından verimli bir devre olması açısından durum güncelleme fonksiyonu basit bir Boole fonksiyonu olarak düşünülmüştür ve Şekil 4.1'de kombinezonsal devre karşılığı görülen ve denklem 4.5'te verilen fonksiyondur.

Denklem 4.4'teki v ve w değerleri j-1'den küçük değerlerdir. Bu v ve w'nın alacağı değerler i ve j değerlerine bağlı olup Tablo 4.1'e göre hesaplanmaktadırlar. j≤4 için v ve w değerleri 0 alınır. Tablo 1'den hesaplanan q_i^j'nin mevcut bitlere karşılık gelmediği durumlarda ise var olan bir karşılık bulunana kadar i indeksi 2'nin en büyük kuvvetleri kadar azaltılır. Örneğin, q_{14}^{93} Şekil 4.1'den de görüldüğü gibi

$$G[K]_i^{(j)}\,(q,c) = q_i^{(j-1)} + K_{j-1} + (q_i^{(v)})(\,q_i^{(w)} + 1) + 1 \qquad (4.5)$$



Şekil 4.2: Mosquito CCSR'ın genişleme yapısı ve durum güncelleme fonksiyonu.

Tablo 4.1: 4<j<96 indeksli G_i^j ve G_0^{96} için v ve w değerleri

	v	w
$(i+j) \bmod 3 = 0$	$j-4+(i \bmod 2)$	$j-2$
$(i+j) \bmod 3 = 1$	$j-6+(i \bmod 2)$	$j-2$
$(i+j) \bmod 6 = 2$	$j-5+(i \bmod 2)$	0
$(i+j) \bmod 6 = 5$	0	$j-2$

bulunmamaktadır ve dolayısıyla i indeksi önce 14 ten küçük 2'nin en büyük kuvveti olan 8 azaltılır ve daha sonra da 6 ten küçük 2'nin en büyük kuvveti olan 4 azaltılarak var olan bir hücre olan q_2^{93} 'e ulaşılır ve durum güncelleme fonksiyonunda o kullanılır.

i>0 için G_i^{96} olan 15 bit ise farklı bir denklem ile hesaplanır ve bu denklem 4.6'te verilmiştir.

$$G[K]_i^{96}(q,c) = qi^{95}\,(q0^{(95-i)} + 1) + qi^{94}\,(qi^{(94-i)} + 1) \qquad (4.6)$$

4.1.4 Mosquito iş hattı aşamaları

Mosquito iş hattında yedi aşama vardır. İlk beş aşama 53'er bitlik belleklerden oluşmaktadır. Birinci aşamada CCSR' nin 128 biti kullanılarak 53'bitlik bir bellek doldurulur. Bu işlemde denklem 4.7'daki fonksiyon kullanılır.

$$G^{j}_{4i \bmod 53}{}^{(1)} = \alpha_{128\text{-}i}{}^{(0)} + \alpha_{i+18}{}^{(0)} + \alpha_{113\text{-}i}{}^{(0)}\,(\alpha_{i+1}{}^{(0)} + 1) + 1 \qquad (4.7)$$

2 ila 5'inci aşamalardaki 53'er bitlik bellek için fonksiyonlar ise denklem 4.8'de verilmiştir.

$$G^{j}_{4i \bmod 53}{}^{(j)} = \alpha_i^{(j)} + \alpha_{i+3}{}^{(j)} + \alpha_{i+1}{}^{(j)} (\alpha_{i+2}{}^{(j)} + 1) + 1 \qquad (4.8)$$

Her iki fonksiyon için de sağ taraftaki terimlerden herhangi birinin indeksinin 52'den büyük olduğu durumda bu terim 0 alınır. Altıncı aşamadaki bellek ise 12 bit uzunluğundadır ve güncelleme fonksiyonu denklem 4.9'deki gibidir.

$$Gi\ (6) = \alpha 4i(5) + \alpha 4i+3(5) + \alpha 4i+1(5)\ (\ \alpha 4i+2(5) + 1) + 1 \qquad (4.9)$$

Son aşama olan 7'inci aşamada ise 3 bitlik bir bellek vardır ve fonksiyonu 4.10'da verilmiştir.

$$G_i{}^{(7)} = \alpha_{4i}{}^{(6)} + \alpha_{4i+1}{}^{(6)} + \alpha_{4i+2}{}^{(6)} + \alpha_{4i+3}{}^{(6)} \qquad (4.10)$$

Algoritma sonucu olan şifre biti z ise bu son aşamadaki 3 bitlik belleğin bitlerine XOR işlemi uygulanması ile elde edilir (4.10).

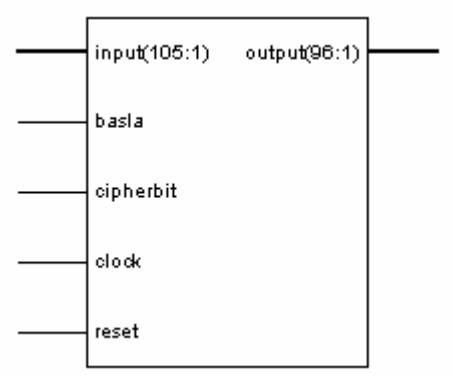
$$z = a_0{}^7 + a_1{}^7 + a_1{}^7 \qquad (4.10)$$

4.2 Gerçekleme Adımları

Mosquito algoritmasını gerçekleyen devrenin mesaj girişi, IV girişi, başlama, Anahtar girişi (K), ilk duruma getirme ve saat girişi olmak üzere olmak üzere toplam altı girişi, ve şifrelenmiş mesaj olan bir adet çıkışı vardır. Mesaj girişi ve Anahtar girişi 96, IV girişi 105 diğer girişler ve çıkış 1 bittir. Devrenin IV girişi bir kaydırmalı belleğe verilmekte bu kaydırmalı bellekten alınan 96 bitlik çıkış CCSR'nin girişini oluşturmaktadır. CCSR'nin 128 bit olan çıkışının ardından ise 7 adet kombinezonsal devre ve bu devrelerin içeriğini tutan 7 kaydedici gelmektedir. Son kaydedicinin 3 bitinin xor işlemiyle şifre olan c oluşturulmakta, bu şifre c, mesaj girişi m ile şifrelenerek ötelemeli kaydediciye geri besleme yapılmaktadır. Tüm bu aşamalar aşağıda detaylandırılacaktır.

4.2.1 Kaydırmalı Bellek

CCSR'nin 96 bitlik girişi, 105 bitlik IV başlangıç vektörünün en yüksek anlamlı 96 bitidir. Algoritmada bulunan 9 katman sebebiyle ilk şifrenin üretilmesi başlangıçtan 9 saat darbesi sonra olmaktadır. Bu 9 saat darbesi boyunca CCSR'nin girişi olan 96 bit, 105 bitlik IV'nin her saat darbesinde bir kez ötelemeli bellekte kaydırılmasıyla elde edilen yeni 105 bitlik vektörün en yüksek anlamlı 96 bitidir. 9 saat darbesi sonucunda elde edilen anahtar dizisi (z) ile devrenin diğer girişi olan mesaj (m) şifrelenir ve bu şifrelenmiş mesaj biti (c), IV'nin kaydırıldığı kaydırmalı belleğe, kaydırma sonucu CCSR'nin girişine verilecek 96 bitlik kısmın en düşük anlamlı hanesine yazılmak üzere geri besleme şeklinde bağlanmaktadır. Bu kaydırmalı belleğin gerçeklenmesiyle elde edilen devrenin blok şeması Şekil 4.3'teki gibidir.

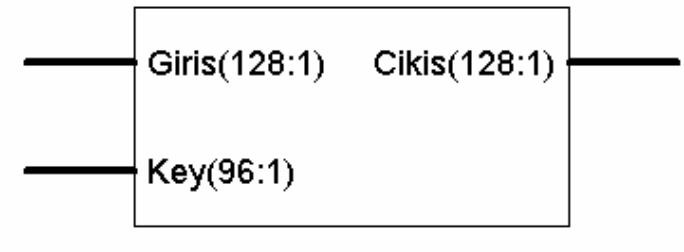


Şekil 4.3: Kaydırmalı bellek

4.2.2 CCSR

CCSR, girişine gelen 96 biti, 128 bite genişletmekte ve bu 128 biti Bölüm 4.1'de anlatılan güncelleme fonksiyonları gereğince değiştirerek 128 bitlik çıkışına vermekte ve girişine kaydırmalı bellekten gelen yeni 96 biti almaktadır.

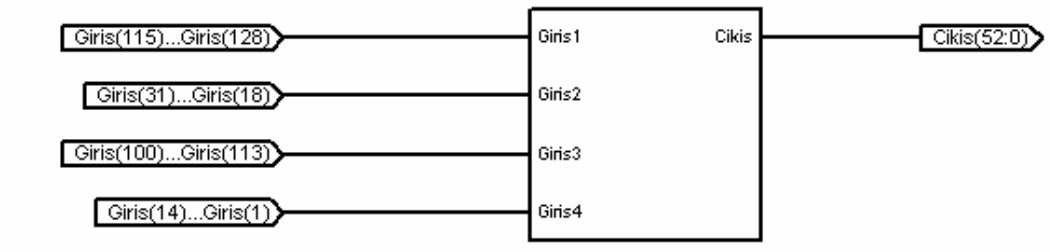
CCSR'nin gerçeklenmesine 128 hanenin her biri için güncellemeyi sağlayacak alt devreler oluşturulmuş, girişleri ve çıkışları ayrı ayrı belirlenmiştir. Güncelleme işlemlerinde Anahtar (K) kullanılmıştır. Devrenin gerçekleme ile elde edilen şematiği Şekil 4.4'teki gibidir.



Şekil 4.4: CCSR blok diyagramı

4.2.3 Katmanlar

Algoritmada CCSR'nin çıkışı olan 128 bit birbirine kaskod 7 katmandan geçerek şifre bitini oluşturmaktadır. Bu katmanlardan ilki 128 giriş- 53 çıkış'lı, bundan sonra gelen dördü 53 giriş – 53 çıkışlı, daha sonraki katman 53 giriş- 12 çıkış'lı ve son katman da 12 giriş 3 çıkışlıdır. Son katmanın üç biti xor işlemi yapılarak şifre elde edilmektedir. Örnek olarak birinci katman Şekil 4.5'te verilmiştir.

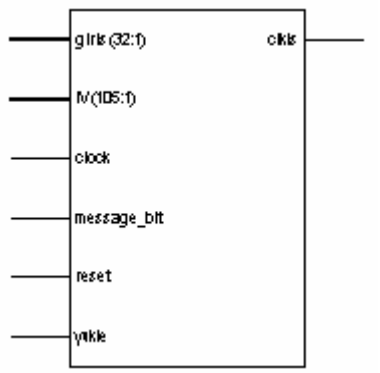


Şekil 4.5: 128 giriş- 53 çıkış'lı katman

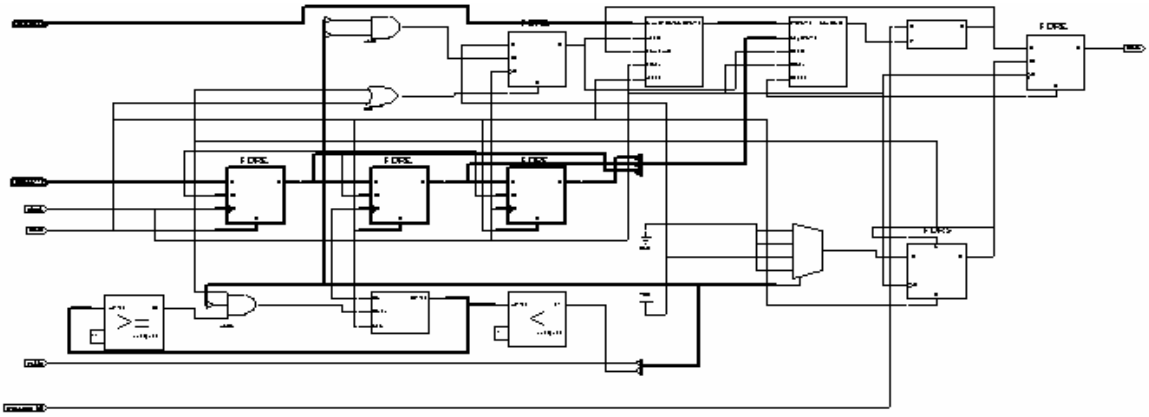
4.2.4 Devrelerin birleştirilmesi

Yukarıda detaylarıyla verilen devrelerin birleştirilmesi ile elde edilen ana devrenin 96 bitlik anahtar (K), 105 bitlik başlangıç vektörü (IV) ve birer bitlik mesaj (m), başlangıç ve ilk

duruma getirme girişleri ve bir bitlik çıkışı şekil 4.6'da görülmektedir. Ana devrenin girişleri kullanılan FPGA'nın bacak sayısını aştığından devrenin anahtar girişi devreye 32'şer bitlik 3 aşamada verilmektedir. Ana devrenin 32 bitlik anahtar girişine her bir saat darbesinde anahtarın 32 biti verilmekte ve üç saat darbesi sonucunda 96 bitlik giriş devreye verilmiş olmaktadır. Anahtar girişi en yüksek anlamlı 32 bitten başlanarak yapılmaktadır. Şekil 4.7'de devrenin daha açık hali görülmektedir.



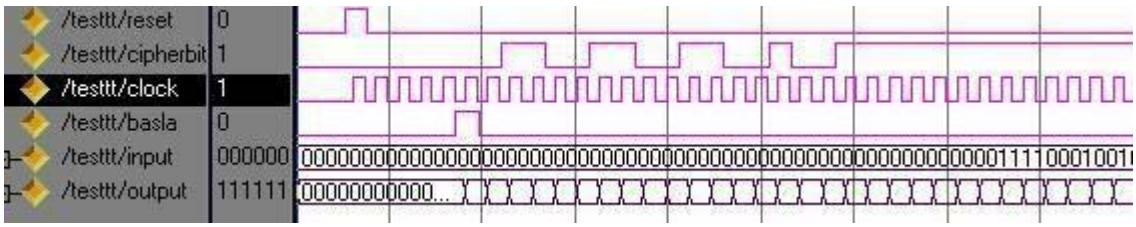
Şekil 4.6: Ana devrenin şematiği (1)



Şekil 4.7: Ana devrenin blok şematiği (2)

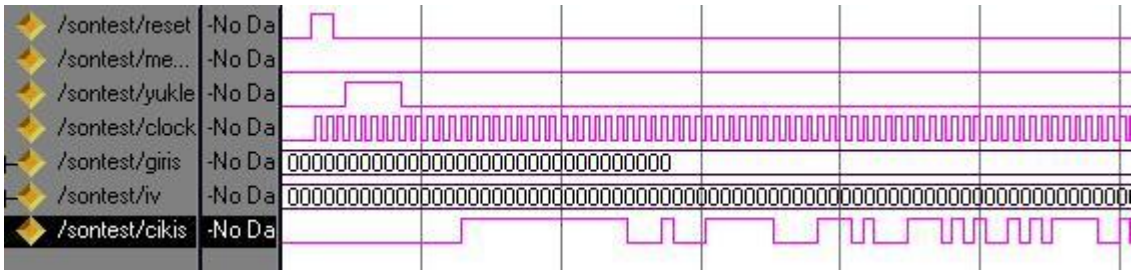
4.3 Gerçekleme Sonuçları

Kaydırmalı belleğin benzetiminde elde edilen sonuç Şekil 4.8'deki gibidir. İlk 9 saat darbesi boyunca şifre üretilmediğinden ve dolayısı ile belleğe gelen geri besleme biti tanımsız olacağından bu sürede kaydırma sonucu boş kalan haneye 0 yazılmış, bu süre sonunda bu haneye üretilen şifrelenmiş mesaj biti yazılmaya başlanmıştır. Şekil 4.8'den de görüldüğü gibi her saat darbesiyle birlikte çıkış değişmektedir. Benzetimde devreye önce ilk duruma getirme işareti verilmiş daha sonra da başlama işareti verilmiştir. Bu işaretler gelmeden önce devrenin çıkışı 0 olacak şekilde ayarlanmıştır.



Şekil 4.8: Kaydırmalı bellek benzetim sonucu

Ana devrelerin benzetimi sonucunda elde edilen sonuç şekil 4.9'deki gibidir. Devrenin IV, anahtar ve mesaj girişleri 0 verilmiş, önce ilk duruma getirme işareti verilerek devre başlamaya hazırlanmış daha sonra başlama işareti verilmiş (yükle) ve devreye üç saat darbesi boyunca anahtar bilgisi yüklenmiş, daha sonraki 9 saat darbesi sonunda da şifre üretilmeye ve mesaj şifrelenmeye başlanmıştır. Şekilde görülen çıkış mesajın şifrelenmiş halidir.



Şekil 4.9: Ana devrenin benzetim sonuçları

5. SONUÇLAR

Bu çalışmada Mosquito dizi şifreleme algoritmasına yer verilmiş, bu algoritmada kullanılan bloklar VHDL donanım tanımlama dili kullanılarak tasarlanmış ve Xilinx programı kullanılarak, xcv1000e kodlu FPGA üzerine sanal olarak gerçeklenmiştir. Algoritmada kullanılan fonksiyonların basit fonksiyonlar olması sebebiyle FPGA'nın az bir alanı kullanılarak algoritma gerçekleştirilmiştir. Yerleştirme ve yollandırma sonucunda devrenin minimum periyodunun 8.321ns olduğu yani maksimum frekansının 120.178MHz olduğu görülmüştür. Benzetim sonrasında elde edilmiş sonuçlar Tablo 5.1'de verilmiştir.

Tablo 5.1: FPGA'da kullanılan alan

Lojik Birimler	**Mevcut Sayı**	**Kullanılan Sayı**	**Yüzde**
Dilimler	12288	395	%3
Dilimlerdeki Flip-Floplar	24576	700	%2
Dört Girişli LUT'lar	24576	542	%2
IOB'ler	158	140	%88
Saat	4	1	%25

KAYNAKLAR

[1] **Daemen, J.,** 2005. The self-synchronizing stream cipher Mosquito: eStream documentation, version 2, STMİcroelectronics Belgium

[2] **Stinson, D.R.,** 2002. Cryptography, Chapman & Hall/CRC Press Company, United States of America

[3] **Acar, S.,** 2005. Eliptik Eğri Kriptografisinde Skaler çarpma bloğunun VHDL ile Tasarımı, *Yüksek Lisans Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul

[4] **Topçu, İ.H.,** 2002. Sahada Programlanabilir Kapı Dizileri Kullanılarak Sayısal Tasarım Kartı Gerçeklenmesi, *Yüksek Lisans Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul

[5] **Coffman, K.,** 2001. Real World FPGA Design With Verilog, Prentice-Hall, Inc, New Jersey

[6] **Berna, A.,** 1998. Sahada Programlanabilir Kapı Dizileri ile Lojik Devre Tasarımı ve VHDL Kullanılarak Bazı Devrelerin Gerçekleştirilmesi, *Yüksek Lisans Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul

[7] **Dent, A.,** 2005. User's Guide to Cryptography and Standarts, Artech House, London