1. GİRİŞ

1.1. Giriş ve Çalışmanın Amacı

İnsanların birbirlerine olan güvenlerinin iyice azaldığı günümüz dünyasında, veri iletişiminde güvenliğin sağlanması en önemli konulardan biri haline gelmiştir. Gün geçtikçe çok büyük miktarlardaki verilerin bilgisayarlar tarafından işlenmesi, saklanması ve elektronik haberleşme kanalları üzerinden bir yerden diğer bir yere iletilmesi gündelik hayatın sıradan işlerinden biri haline gelmektedir. Ancak verilerin iletimi sırasında kullanılan haberleşme kanallarının herkesin kullanımına ya da erişimine açık olması, sözkonusu verilerin yetkili olmayan (üçüncü) şahıslar tarafından değiştirilmesi, yok edilmesi ve içeriğine ulaşılması problemini gündeme getirmektedir. Bu noktada, mesajların herkesin erişimine açık elektronik haberleşme kanallarından iletilebilmesi için bir takım dönüşümler sonucunda değişikliğe uğratılarak üçüncü şahıslar için anlaşılamaz bir hale getirilmesi gerekmektedir. Bu amaçla yapılan tüm işlemlere birden Kriptografi ya da Şifreleme adı verilir [1]. Diğer bir tanımla Kriptografi, en az iki kişinin güvenli olmayan bir kanal üzerinden üçüncü bir şahsa bilgi sızdırmadan haberleşmesini sağlamak amacıyla matematiksel tekniklerin geliştirilmesidir [2].

Kriptografik teknikler iki grupta incelenir; simetrik anahtarlama kriptografisi (symmetric key encryption) ve açık (asimetrik) anahtarlama kriptografisi (public key encryption). Açık anahtarlama kriptografisinde, vericideki şifreleme anahtarı ile alıcıdaki şifre çözme anahtarı birbirinden farklıdır. Aksine, simetrik anahtarlama kriptografisinde şifreleme ve şifre çözme anahtarları aynıdır. Simetrik anahtar kriptografisi iki sınıfa ayrılır: Blok şifreleme (block ciphers) ve dizi şifreleme (stream ciphers). Blok şifrelemede veri bloklar halinde şifrelenir ve blok genişliğini daha önceden algoritma tasarımcısı belirler. Dizi şifreleyicide ise blok uzunluğunu algoritmanın kullanıcısı belirler. Bu esneklik başlangıç vektörü denen bir diğer parametreye ihtiyaç duyar. Dizi şifreleme, mesajdaki her harfin özel bir algoritma kullanılarak, anahtardan üretilen bir anahtar dizisi ile sırayla şifrelenmesidir. Böylece alfabedeki bir harf her defasında farklı bir şifre ile şifrelenecek ve dolayısıyla farklı sembollere dönüştürülecektir [3].

Bu çalışmada, bir dizi şifreleme algoritması olan Trivium Dizi Şifreleme Algoritması' nın Sahada Programlanabilir Kapı Dizileri (FPGA - Field Programmable Gate Array) üzerinde gerçeklenmesi hedeflenmiştir. Trivium, dizi şifreleme konusunda Avrupa Birliği Standardının seçilmesi amacıyla düzenlenen ECRYPT Dizi Şifreleme Projesi' ne aday olan algoritmalardan bir tanesidir. Trivium, donanım bazlı çalışan senkron bir dizi şifreleme algoritmasıdır. Donanım bazlı olması sebebiyle bir donanım betimleme dili olan VHDL (Very High Speed Integrated Circuit – Hardware Description Language) kullanılarak FPGA üzerinde gerçeklenmesi gerekmektedir. Bu sebeple projenin gerçeklenebilmesi için öncelikle VHDL dili öğrenilmiş, sonra Xilinx ISE adlı programda algoritma yazılmış, daha sonra ModelSim adlı programda simülasyonlar gerçekleştirilmiş ve son olarak FPGA' e yüklenerek algoritmanın çalışması sağlanmıştır.

Boole cebrine göre “0” ın yanlışı, “1” in doğruyu temsil ettiğini kabul edersek, modülo-2 sisteminde toplama (+) işleminin karşılığı EXOR (exclusive-or - ayrıcalıklı-veya) ve çarpma (.) işleminin karşılığı VE (AND) lojik işlemleridir [4]. Bu çalışmada (+) EXOR işlemini, (.) ise AND işlemini temsil etmektedir.

Projenin son aşaması olan algoritmanın Sahada Programlanabilir Kapı Dizileri (FPGA) üzerinde gerçeklenmesi için öncelikle FPGA tanıtılmalıdır. FPGA, Programlanabilir Lojik Elemanların (Programmable Logic Devices, PLD) gelişmiş bir tipi olarak düşünülebilir. PLD elemanları VEYA (OR) kapı dizilerine bağlanmış VE (AND) kapı dizilerini içerir. Örneğin, Programlanabilir Lojik Dizi (Programmable Array Logic, PAL), programlanabilir VE uzayı ve sabit VEYA uzayından oluşan PLD’ dir. PLD’ lerin en büyük eksikliği, bir fonksiyonu çarpımlar toplamı biçiminde gerçeklediklerinden, yüksek çarpım terimleri içeren fonksiyonları gerçekleyememeleridir. FPGA’ lar ise programlanabilir lojik bloklar ve ara bağlantılardan oluşur. Kullanıcının tasarladığı lojik devreye göre, tümdevre üreticisi tarafından sağlanan bir yazılım sayesinde lojik bloklar ve aralarındaki bağlantılar programlanır. Tasarım sırasında kullanıcıya sağladığı esneklik, düşük maliyet ve hızlı ilk üretme özelliği ile FPGA’ lar sayısal tasarım ortamlarının vazgeçilmez yapıları haline gelmiştir [2].

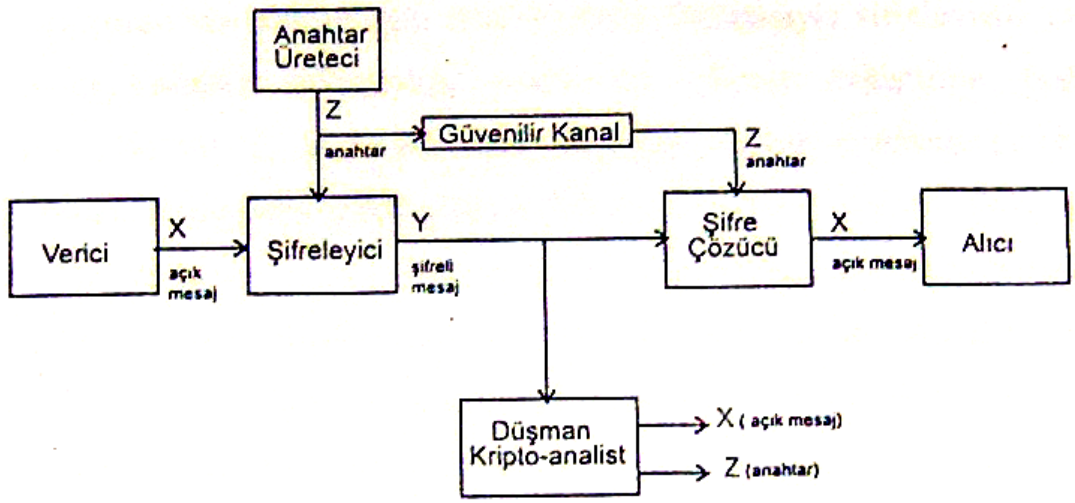
Sunulan bu çalışmada Trivium Dizi Şifreleme Algoritması’ nın FPGA üzerinde gerçeklenmesinin yanında asıl amaç, şifreleme konusunda geniş bilgi edinmek, donanım betimleme dillerinden biri olan VHDL öğrenilerek şifreleme algoritmalarını yazabilecek duruma gelmek, algoritmaların yazımı ve denenmesi için Xilinx ISE ve ModelSim programlarını detaylı bir şekilde öğrenmek ve kullanmak, ayrıca FPGA’ lar hakkında bilgi edinerek bu dizileri kullanmak ve algoritmanın işlevsel hale dönüşmesini sağlamaktır. Bunların yanında Kriptografi alanında Avrupa Birliği standardı olabilecek bir algoritmanın gerçeklenmiş olması ve bu algoritma hakkında yorum yapabilecek sayılı insanlardan biri haline gelmek diğer bir hedeftir.

Bu amaçlar doğrultusunda hazırlanan bitirme ödevinde 1. bölümde konuya genel bir giriş yapılmış ve Kriptografi, Dizi Şifreleme, FPGA ve Trivium Dizi Şifreleme Algoritması kısaca tanıtılarak çalışmanın amacı belirtilmiştir. 2. bölümde ise Dizi Şifreleme sistemleri detaylı olarak ele alınmış, yapısı ve özellikleri anlatılmıştır. 3. bölümde FPGA yapısı, mimarisi ve programlanması hakkında bilgiler verilmiştir. 4. bölümde ise ödevin asıl konusu olan Trivium Dizi Şifreleme Algoritması ve algoritmanın gerçekleme adımları, algoritmanın test edilmesi ve FPGA üzerinde gerçekleme anlatılmıştır. Son bölüm olan 5. bölümde ise tezin genelinde elde edilen sonuçlardan bahsedilmiştir.

2. DİZİ ŞİFRELEME SİSTEMLERİ

Günümüzde şifreleme sistemleri, şifreleme anahtarının gizli tutukluğu "gizli-anahtarlı" ve şifreleme anahtarının açık olduğu "açık-anahtarlı" sistemler olmak üzere ikiye ayrılırlar. Bu sistemler sırasıyla "simetrik" ve "anti-simetrik" şifreleme sistemleri olarak da bilinirler. Açık-anahtarlı ya da anti-simetrik şifreleme sistemlerine örnek RSA algoritmasının kullanıldığı sistemlerdir. Bu tür sistemlerde şifreleme ve şifre çözme işlemleri, gizli-anahtarlı sistemlerin aksine, farklı anahtarlar kullanılarak gerçekleştirilir. Bu nedenle açık-anahtarlı sistemler simetrik olmayan bir yapıya sahiptir.

Simetrik yapılı gizli-anahtarlı şifreleme sistemlerinde şifreleme ve şifre çözme anahtarları aynıdır. Şekil 2.1 'de gizli-anahtarlı sistemin genel yapısı görülmektedir:



Şekil 2.1. Gizli-anahtarlı şifreleme sisteminin genel görünüşü

Sistemde *E* şifreleme, *D* de şifre çözme dönüşümlerini göstermektedirler. Şifreleme ve şifre çözme işlemleri aynı 2 anahtarı kullanılarak gerçekleştirilir:

$$Y = E_Z(X) \qquad \text{(Şifreleme)} \qquad (2.1)$$

$$X = D_Z(Y) \qquad \text{(Şifre çözme)} \qquad (2.2)$$

Düşman kripto-analistin gizli anahtara erişimi güvenilir bir kanalla engellendiği taktirde sistem oldukça güvenilir olabilmektedir.

Simetrik sistemler de "blok şifreleme" ve "dizi şifreleme" sistemleri olarak ikiye ayrılırlar. "Blok şifreleme" sistemlerinde mesaj sabit uzunluklu bloklara bölünür ve her bir blok diğerlerinden bağımsız bir şekilde şifrelenir. Bu şekilde her bir bloğa karşılık aynı alfabeden aynı boyutta başka bir blok karşılık düşürülür. Bu nedenle blok şifreleme sistemleri basit

yerine koyma sistemleridir. Eğer anahtarı bilmiyorsa, düşman kripto-analist eline geçirdiği şifreli mesajı çözmek için şifrelemede kullanılmış olması olası tüm anahtarları denemek zorundadır (Kaba kuvvet yaklaşımı). Bu nedenle bu tür sistemlerde anahtar kümesi olabildiğince geniş tutulmalıdır [1].

"Dizi şifreleme" sistemleri ise, her bir mesaj birimini, zamanla değişen bir fonksiyon kullanarak şifreler. Dizi şifreleme sisteminin iç yapısı zamana bağlı olarak durum değiştiren bir makine gibi düşünülebilir. Dolayısıyla şifrelemede kullanılan fonksiyonun zamana bağımlılığı makinenin durum değiştirme bağıntısıyla tanımlanır. Bu tür bir sistemde mesaj biriminin büyük olmasına gerek yoktur. Aksine mesaj biriminin olabildiğince küçük tutulması istenir. Dizi şifreleme sistemlerinde kullanılan mesaj birimleri genelde Latin alfabesinden bir karakter ya da tek dijitlik sayılardır. Ancak bu çalışmada mesaj birimi olarak tek dijitlik sayılar ele alınacak, kimi özel incelemeler gerektiren durumlarda ise bu tek dijitlik sayılar {0, 1} ikili sayıları olacaktır. Bu nedenle bu sayılardan söz ederken sayı ya da dijit yerine kimi zaman da bit kullanılabilecektir.

Bir dizi şifreleme sisteminde mesaj dijitleri tek tek şifrelendiğinden mesajlar genelde bir dizi şeklinde düşünülür. Mesaj dizisinin her bir dijiti şifrelendikten sonra sistem belirli bir kural uyarınca durum değiştirir. Sistemin durum değiştirmesi şifreleme anahtarının da değişmesine neden olur. Bu nedenle dizi şifreleme sistemlerinde anahtar da mesajla aynı boyutta bir dizi şeklindedir. Bu anahtar dizisinin yapısı şifreleme sisteminin güvenilirliği konusunda oldukça büyük bir önem taşımaktadır.

Tipik bir dizi şifreleme sistemi "kayar anahtar üreteci" (KAÜ) olarak adlandırılan çıkışında sanki-rasgele (pseudorandom) diziler üreten bir sonlu-durumlu makineden oluşmaktadır. Mesajın ikili bir sayı dizisi olduğu düşünülürse sonlu durumlu makinenin çıkışındaki dijitler de ikili sayı dizisi olmalıdır. Bu dizinin bitleri ardışıl olarak açık mesaj (plaintext) dizisindeki bitlerle modülo-2 toplanır (yani EXOR işlemine girer). Sonuçta çıkan ikili dizi de şifreli metin (ciphertext) dizisi olarak adlandırılır. Anahtar dizisi de denilen KAÜ 'nin çıkışındaki dizinin rasgele görünüşlü olmasındaki amaç, açık mesaj dizisindeki bitlerle şifreli mesaj bitleri arasındaki karşılıklı informasyonun sıfıra inmesini sağlamaktır. Aslında bu tüm kripto sistemlerinin ana amacıdır. Gerçekten bir dizi tamamen rasgele bir başka bir diziyle toplanırsa sonuçta çıkan diziden istatiksel olarak bağımsız olacağı aşikardır.

Bu nedenle dizi şifreleme sistemlerinin tasarım aşamasında gerçeklenmesi istenen temel özellik olabildiğince rasgele görünüme sahip dizilerin deterministik yollarla bir alt sistem tarafından üretilmesini sağlamaktır. Bilindiği üzere tamamen deterministik yollarla gerçek rasgele dizilerin üretilmesi olanaklı değildir. Bu yüzden amaç KAÜ 'nin olabildiğince rasgele görünüme sahip bit dizileri üretmesini sağlamak olmalıdır.

Bu durumda bir dizinin rasgele olmasının ne demek olduğu ve kriptografik sağlamlık açısından ne tür kıstasların değerlendirilmeye alınması gerektiği konusu biraz açılmalıdır. Genel olarak, eğer bir dizi içerisinde yapısal bir düzenlilik bulunmuyorsa, dizi elemanları hakkında, önceki terimlere bakarak hiç bir öngörü ya da dizi üzerinde herhangi bir tanımlama yapılamıyorsa, o dizinin rasgeleliğinden söz edilebilir. Burada tanımlama sözüyle görece az sayıda teriminden dizinin yeniden üretilebilmesini sağlayacak bir kural ya da bir bağıntı kastedilmektedir. Bir dizinin görece basit bir bağıntıyı sağlaması demek o dizinin gerçek rasgele bir dizi olmadığını göstereceğinden, gerçek rasgele bir dizinin yeniden üretilebilmesi için basit bir bağıntı bulmak imkansızdır. Diğer yanda deterministik yollarla üretilen sanki-rasgele dizilerin gerçek rasgele diziler kadar güçlü olacağını düşünmek yanlış olur. Dolayısıyla böyle bir dizinin göreceli olarak daha basit bir bağıntıyı sağlayacağı açıktır. Bu durumda KAÜ çıkışındaki dizinin, basit olması olası böyle bir bağıntıyı sağlaması sistemin güvenilirliğini önemli ölçüde sarsacaktır. O halde yapılması gereken sistem

tasarlanırken bu bağıntının yeterince karmaşık olmasını sağlamak, böylece eldeki hesaplama olanaklarıyla bulunmasını imkansız bir hale getirmek olacaktır.

Kayar anahtarın küçük bir parçasından yeniden üretilebilmesinin neden bu kadar sakıncalı olduğu sorusu akla gelebilir. Eğer düşman kripto-analist eline geçirdiği küçük bir anahtar parçasıyla dizinin geri kalan kısmını kolayca bulabiliyorsa şifreyi kırmış olacaktır. Çünkü dizi şifreleme sistemlerinin tüm sağlamlığı gizli anahtarın iyi saklanmasına bağlıdır.

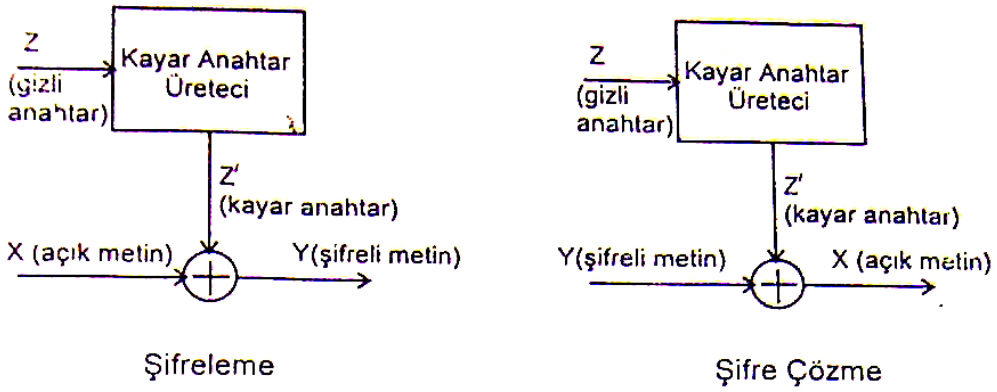
Dizi şifreleme sistemlerine örnek olarak çok bilinen ve yaygın kullanım alanı bulan "ikili toplamsal dizi şifreleme" sistemidir. Bu sistemin basit blok şeması Şekil 2.2 'de görülmektedir. İkili toplamsal dizi şifrelemede gerçek gizli anahtar, kayar anahtardan farklıdır. *K* bitlik gizli anahtar *Z*, KAÜ 'ni kontrol ederek asıl anahtar dizisi $Z_1', Z_2', \ldots, Z_N'$ nin üretilmesini sağlar. Burada *Z'* anahtar dizisini üreten *K* bitlik gerçek gizli anahtar *Z* ye tohum (seed) denir. Gerçek gizli anahtar (*Z*) hiçbir zaman şifreleme işleminde doğrudan kullanılmaz. Gizli anahtarın uzunluğu kayar anahtarın uzunluğu yanında oldukça küçüktür *(K « N).* Şifreli-metin bitleri, ikili açık-metin bitlerinin kayar anahtar dizisiyle terim terim modülo-2 toplanması sonucunda elde edilirler:

$$Y_n = X_n + Z'_n \qquad n = 1,2,\ldots,N \qquad \text{(Şifreleme)} \qquad (2.3)$$

Burada X_n açık-metinin *n.* Bitini Y_n de şifreli-metinin *n.* bitini göstermektedir. Modülo-2 toplama ve çıkarma işlemleri eşdeğer olduğundan,

$$X_n = Y_n + Z'_n \qquad n = 1,2,\ldots,N \qquad \text{(Şifre Çözme)} \qquad (2.4)$$

şifreleme ve şifre çözme işlemleri eşdeğer cihazlarla gerçekleştirilirler.



Şekil 2.2. İkili toplamsal dizi şifreleme sistemi

Bu sistemde bir açık-metin biti yalnızca tek bir şifreli-metin bitini etkileyecektir. Bu çok iyi bir durum değildir. Çünkü şifreleme uygulamaları açısından açık-metinde yer alan bir bitin şifreli-metinde yer alan birçok biti etkilemesi, bir diğer deyişle "yayılımın" (diffusion) fazla olması gerekli bir koşuldur. Ancak diğer yandan her bir gizli anahtar biti şifreli-metindeki birçok biti etkileyeceğinden "anahtar yayılımı" iyi olur. Bu şekilde sistem dengelenmiş olur [1].

3. FPGA YAPISI

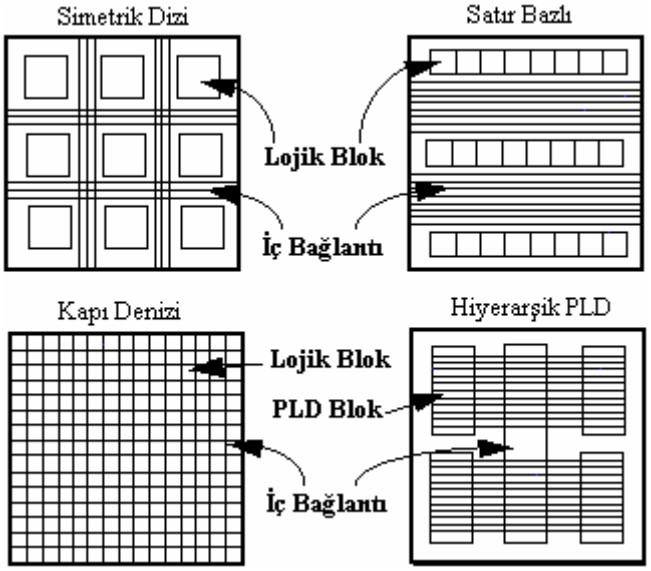
Sahada Programlanabilir Kapı Dizileri (Field Programmable Gate Array, FPGA) yaygın olarak kullanılan programlanabilir devre elemanlarıdır. Programlanabilir devre elemanları geniş uygulama alanları sağlayabilmek için genel amaçlı tümdevreler olarak tasarlanmışlardır. Geniş kullanım alanları bulunan ilk programlanabilir devre elemanı, programlanabilir salt oku bellek (Programmable Read Only Memory, PROM) elemanıdır. PROM' ların, bir kez programlanabilen elemanlar olarak iki sürümü vardır. Maske Programlanabilir Tümdevre, sadece üretici tarafından programlanabilir, Sahada Programlanabilir Tümdevre ise son kullanıcı tarafından programlanabilir. Sahada programlanabilir PROM' un, silinebilir programlanabilir salt oku bellek (Erasable Programmable Read Only Memory, EPROM) ve elektriksel olarak silinebilir ve programlanabilir salt oku bellek (Electrically Erasable Programmable Read Only Memory, EEPROM) olarak iki tipi geliştirilmiştir. EEPROM defalarca silinebilme ve programlanabilme avantajına sahiptir [2].

Bu alandaki diğer bir aşama ise Programlanabilir Lojik Elemanların (Programmable Logic Devices, PLD) geliştirilmesidir. Bu elemanlar lojik devre gerçeklemek amacıyla yapılmıştır. PLD elemanları VEYA (OR) kapı dizilerine bağlanmış VE (AND) kapı dizilerini içerir. Örneğin, Programlanabilir Lojik Dizi (Programmable Array Logic, PAL), programlanabilir VE uzayı ve sabit VEYA uzayından oluşan PLD' dir. PAL' lerin maske ve sahada programlanabilir sürümleri vardır. Küçük lojik devreleri gerçeklemek amacıyla tasarlanmışlardır.

PLD' lerin en büyük eksikliği, bir fonksiyonu çarpımlar toplamı biçiminde gerçeklediklerinden, yüksek çarpım terimleri içeren fonksiyonları gerçekleyememeleridir. PLD' lerin performansını iyileştirmek, kullanılan silisyum alanını azaltmak ve güvenilirliği artırmak amacıyla Karmaşık PLD' ler (Complex PLD, CPLD) üretilmiştir. Genel bir CPLD' nin içerisinde lojik bloklar bulunur ve bu bloklar küçük PLD yapılarında olup birbirleri ile programlanabilir ara bağlantı matrisi kullanarak haberleşirler. Bu şekilde silisyum alanı etkin bir biçimde kullanılmakta ve PLD' lere oranla daha fazla fonksiyon CPLD' lerle gerçeklenebilmektedir.

1985'te Xilinx Inc. Sahada Programlanabilir Kapı Dizilerini (FPGA) piyasaya sürmüştür. FPGA' lar, programlanabilir lojik bloklar ve ara bağlantılardan oluşur. Kullanıcının tasarladığı lojik devreye göre, tümdevre üreticisi tarafından sağlanan bir yazılım sayesinde lojik bloklar ve aralarındaki bağlantılar programlanır. Tasarım sırasında kullanıcıya sağladığı esneklik, düşük maliyet ve hızlı ilk üretme özelliği ile FPGA' lar sayısal tasarım ortamlarının vazgeçilmez yapıları haline gelmiştir [2].

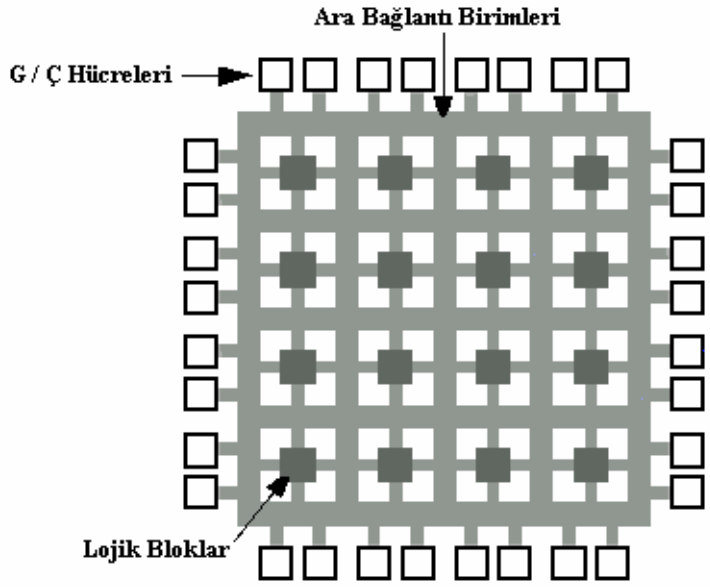
FPGA mimarileri, bağlantı kanallarının yapısına göre: simetrik dizi (symmetrical array), satır-bazlı (row-based array), hiyerarşik PLD (hierarchical PLD) ve kapı denizi (sea of gate) mimarisi olmak üzere dört ana gruba ayrılır (Şekil 3.1).



Şekil 3.1: FPGA sınıfları

Tüm bu FPGA' larda ara bağlantılar ve bunların nasıl programlanacağı farklıdır. Halen kullanılmakta olan dört programlama teknolojisi bulunmaktadır. Bunlar, Statik Rastgele Erişimli Bellek (Random Access Memory, RAM) hücreleri, anti-sigorta ("anti-fuse"), EPROM transistorları ve EEPROM transistorlarıdır. Uygulamaya bağlı olarak bir FPGA teknolojisi tercih edilir.

FPGA üç tane önemli düzenlenebilir elemana sahiptir: Düzenlenebilir Lojik Bloklar (Configurable Logic Blocks, CLB), Giriş / Çıkış blokları (Input / Output Blocks, IOB) ve Ara Bağlantılar (Şekil 3.2). CLB' ler kullanıcı lojiğini oluşturan fonksiyonel elemanlardır. IOB' ler FPGA' nın bacakları ile iç işaretler arasında arayüz oluşturur. Programlanabilir ara bağlantı birimleri ise CLB ve IOB' lerin giriş ve çıkışlarını birleştirmek için uygun hatlar üzerinden yolları belirler. İstenilen düzenleme, lojik fonksiyonların ve ara bağlantıların nasıl gerçekleneceğini belirleyen iç statik bellek hücrelerinin programlanmasıyla sağlanır.



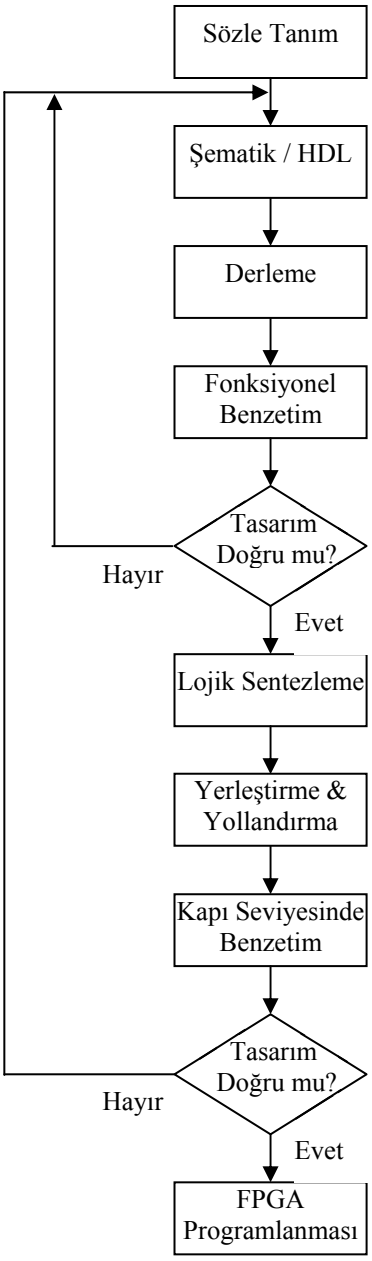
Şekil 3.2: Genel FPGA mimarisi

CLB' ler, çok karışık veya TÜVE (NAND) kapısı kadar basit olabilir. CLB' lerin mimarisi genellikle doğruluk tablosu (Look-Up Table, LUT) veya Çoğullayıcı (Multiplexer, MUX) yapısından oluşur. Bazılarında ise ardışıl devrelerin de gerçeklenebilmesi amacıyla flip-flop' lar kullanılmıştır**.**

EPROM programlama teknolojisi, EPROM belleklerinde kullanılan yönteme benzemektedir. Bu tip FPGA' ların da tekrar programlanabilmeleri için devreden sökülerek ultraviyole ışınlarına tutulması gerekmektedir. EEPROM teknolojisinin EPROM teknolojisine göre üstünlüğü FPGA' nın devreden sökülmeden elektriksel olarak silinebilmesidir. Ancak, bu tip FPGA' ların silisyum alanları EPROM teknolojisini kullananlara göre iki kat daha fazladır. Günümüzde en çok kullanılan programlama teknolojileri SRAM ve anti-sigorta programlamadır. Her iki teknolojinin de birbirlerine göre üstünlükleri ve zayıf yönleri bulunmaktadır.

FPGA' ların programlanması aşamasında ilk olarak tasarlanacak devrenin sözle tanımı verilir. Daha sonra şematik olarak veya yüksek seviyeli donanım tanımlama dilleri (Hardware Description Language, HDL) kullanılarak tasarım yapılır. Tasarım her ne şekilde olursa olsun derleme işleminden sonra devreye ait standart bağlantı listesi (netlist) oluşturulur. Yapılan tasarımın devreye ait istenen özellikleri yerine getirip getirmediği fonksiyonel benzetim (functional simulation) yapılarak test edilir. Benzetim sonucuna göre gerekirse, tasarımda değişiklikler yapılarak istenen sonuç elde edilene kadar bu şekilde iterasyona devam edilir. İstenilen sonuç elde edildikten sonra lojik sentezleme adımına geçilir. Bu aşamada üretilecek devre gerçeklenirken kullanılacak FPGA seçilir. Buna göre lojik sentezleyicinin, kullanılacak FPGA' yı desteklemesi gerekmektedir. Sentezleme işleminde ayrıca varsa, devreye ait lojik kısıtlamalar (Giriş / Çıkış bacakları, zamanlama, yerleştirme, saat frekansı, kritik yollar gibi) kullanıcı kısıtlama dosyası (user constraints file, Xilinx) ile birlikte verilebilir. Lojik sentezleyiciler, istenilen fonksiyonların en iyi şekilde gerçeklenebilmesi için gerekli lojik indirgemeleri (logic optimization) de yaptıktan sonra elde edilen lojik fonksiyonların FPGA içerisindeki lojik bloklarla eşleştirilmesi işlemi yapılarak (technology mapping) kapı seviyesinde bir bağlantı listesi oluşturulur. Teknoloji eşleştirilmesi sırasında, kullanıcı kısıtlama dosyası da kullanılarak zamanlama gereksinimi karşılanmak amacıyla gerekirse daha fazla lojik eleman kullanılabilir. Sentezleme sonrasında, yerleştirme ve yollandırma (placement and routing) işlemleri yapılır. Bu adımda, devre fonksiyonları ile eşleştirilmiş lojik bloklar FPGA içerisinde uygun yerlere yerleştirilir ve bu bloklar arasındaki bağlantılar oluşturulur. Lojik yolların daha kısa olması amacıyla birbirleriyle ilişkili CLB' ler yakın yerleştirilir. Yollandırmada ise bağlantılar uygun şekilde seçilir. Örneğin, tasarımın bir çok alanında bir işarete ihtiyaç varsa en küçük gecikmeyi sağlamak amacıyla uzun bir yol kullanılır. Bu aşamadan sonra kapı seviyesinde benzetimin gerçekleştirilmesi uygun olacaktır. Çünkü artık bütün CLB' lere (LUT veya Çoğullayıcılar ve flip-flop' lara) ve yollandırma bağlantılarına ait gecikmeler gerçeğe çok yakın olarak elde edilmiştir. Bu gecikmeler de eklenerek devrenin benzetimi yapıldığında, zamanlama ve hız açısından kritik yollar saptanabilir. Yerleştirme ve yollandırma sonrası benzetimlerde 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. Şekil 3.3'te yüksek seviyeli tasarım sürecine ilişkin akış diyagramı verilmiştir [2].

Trivium Dizi Şifreleme Algoritmasının FPGA üzerinde gerçeklenmesi işleminde kullanılan FPGA, Xilinx firmasına ait olup Virtex-E (XCV1000E)' dir. Virtex-E yapısı da diğer FPGA' lere benzer olarak CLB, programlanabilir Ara Bağlantılar ve Giriş Çıkış Bloklarından oluşur. CLB' ler lojik fonksiyonları gerçeklerken, Giriş Çıkış Blokları FPGA birimiyle CLB' ler arasındaki ara yüzü oluştururlar [5].



Şekil 3.3: Tasarım sürecine ilişkin akış diyagramı

4. TRIVIUM ALGORİTMASI VE GERÇEKLENMESİ

4.1. Trivium Dizi Şifreleme Algoritması

Trivium Dizi Şifreleme Algoritması donanım bazlı senkron bir dizi şifreleme sistemidir. Senkron olmasının sebebi, anahtar dizisinin açık metinden bağımsız olarak üretilmesidir. Trivium Dizi Şifreleme Algoritması' nın üretilme amacı hız ve alan arasında esnek bir optimizasyon sağlamaktır. Trivium, güvenlik, hız ve esneklikten fedakarlık etmeden bir dizi şifreleme sisteminin ne kadar basitleştirilebileceğini araştırmak için tasarlanmıştır [6].

Trivium Dizi Şifreleme Algoritması, 80-bit gizli anahtar (tohum) ve 80-bit ilk değer (IV) ile 2^{64} bite kadar anahtar dizisi üretebilen senkron bir dizi şifreleme sistemidir. Trivium, diğer dizi şifreleme sistemlerinde olduğu gibi iki aşamadan oluşur: ilk olarak gizli anahtar (K) ve ilk değer (IV) kullanılarak şifreleyici başlangıç durumuna getirilir, daha sonra durum tekrar tekrar güncellenir ve anahtar bitlerini üretmek için kullanılır. Tablo 1' de Trivium parametreleri verilmiştir:

Tablo 4.1. Trivium Parametreleri

Parametreler	Uzunluğu
Anahtar Uzunluğu	80 bit
İlk Değer Uzunluğu	80 bit
İç Durum Uzunluğu	288 bit

Trivium Dizi Şifreleme Algoritmasında 288-bitlik iç durum (internal state) $s_1, s_2, \ldots, s_{288}$ ile gösterilmiştir. Anahtar dizisi üretilme işlemi 15 özel bitin ayrılması, hepsinin 3 bitin güncellenmesinde kullanılması ve son olarak 1 bit anahtar dizisi z_i' nin hesaplanması işlemlerini içerir. Daha sonra durum bitleri kaydırılarak döndürülür ve istenen $N \leq 2^{64}$ bitlik anahtar dizisi üretilene kadar işlem tekrarlanır. Algoritmanın matematiksel tanımı aşağıda verilmiştir:

for i = 1 to N do

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \,.\, s_{92} + s_{171}$

$t_2 \leftarrow t_2 + s_{175} \,.\, s_{176} + s_{264}$

$t_3 \leftarrow t_3 + s_{286} \,.\, s_{287} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

Modülo-2' ye göre yazılan yukarıdaki algoritmada toplama (+) işlemi EXOR ve çarpma (.) işlemi AND lojik işlemlerini göstermektedir. Ayrıca s_1 en düşük anlamlı biti, s_{288} ise en yüksek anlamlı biti göstermektedir. Anahtar üretilme işlemi Şekil 4.1.' de gösterilmiştir.

Trivium Dizi Şifreleme Algoritması' nda anahtar üretme işleminden önce durumların başlangıç durumuna getirilmesi gerekmektedir. 288-bitlik iç durum 80-bitlik gizli anahtarın, 80-bitlik ilk değerin (IV) yüklenmesi ve s_{286}, s_{287} ve s_{288} hariç diğer bitlerin '0' atanması ile başlangıç konumuna getirilmiş olur. Daha sonra 288-bitlik iç durum 4 tam tur boyunca anahtar bitinin üretilmesi hariç yukarıda anlatıldığı gibi kaydırılarak döndürülür. Başlangıç koşuluna getirilme ve 4 tur döndürme aşağıdaki gibi matematiksel olarak ifade edilebilir:

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0)$

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$

for i = 1 to 4 . 288 do

$t_1 \leftarrow s_{66} + s_{91} \,.\, s_{92} + s_{93} + s_{171}$

$t_2 \leftarrow s_{162} + s_{175} \,.\, s_{176} + s_{177} + s_{264}$

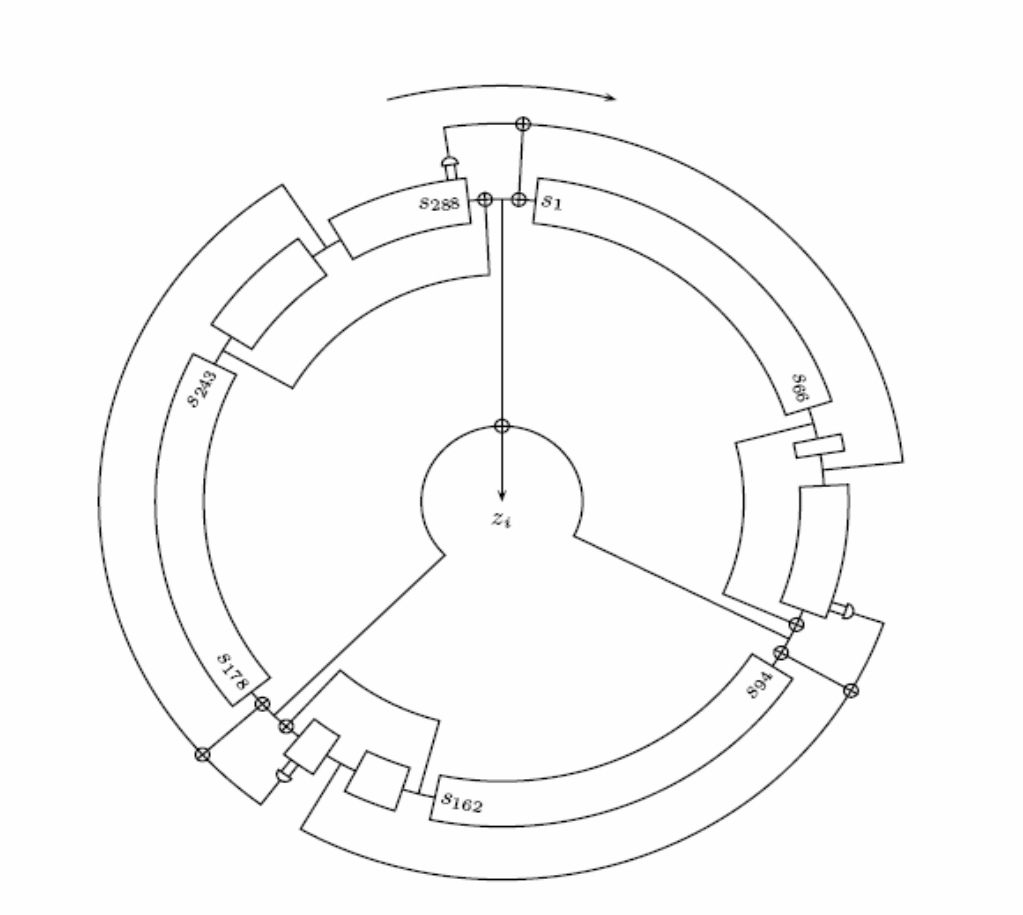
$t_3 \leftarrow s_{243} + s_{286} \,.\, s_{287} + s_{288} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for



Şekil 4.1. Trivium ve Anahtar Üretilme İşlemi

4.2. Gerçekleme Adımları

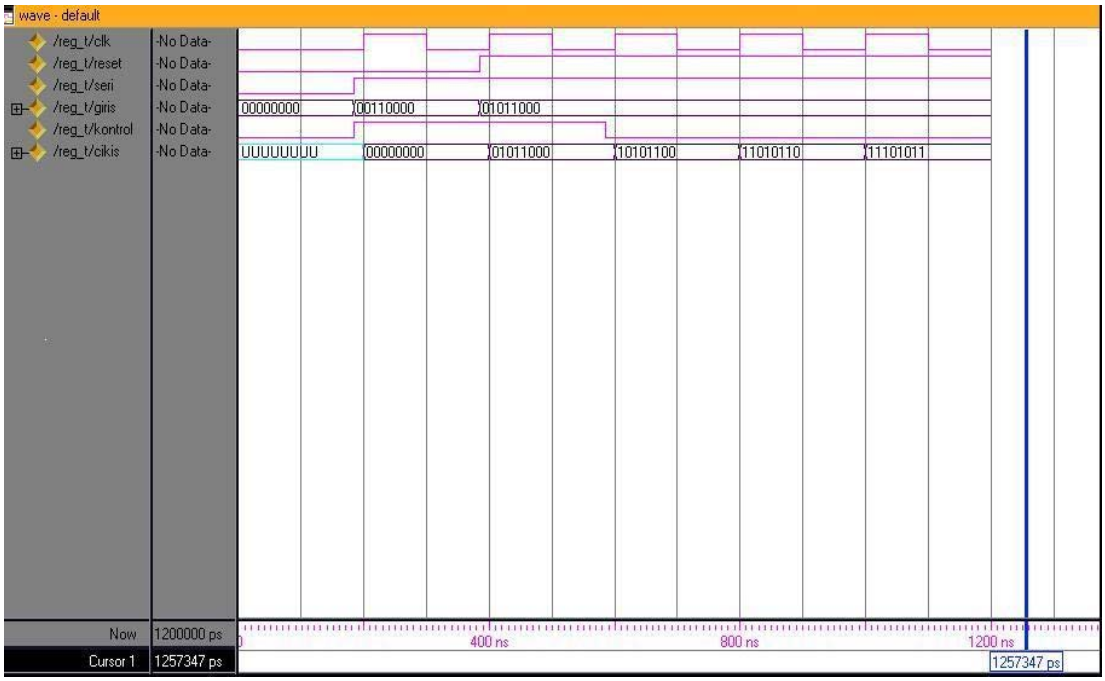
Trivium Dizi Şifreleme Algoritmasının FPGA üzerinde gerçeklenmesi ödevinin ilk aşaması olarak VHDL donanım betimleme dilinin öğrenilmesi gelmektedir. Bilinen en güncel donanım betimleme dillerinden biri olan VHDL' in öğrenilmesinde [7] no' lu kaynaktan yararlanılmıştır. Özellikle döngü içeren durumlar ve bileşen (component) olarak gösterme işlemleri üzerinde durulmuştur. Ayrıca internetten bulunan VHDL dili ile yazılmış örnekler incelenerek Trivium ile benzerlik gösteren algoritmalar hakkında bilgi edinilmiştir. Böylelikle Trivium algoritmasını donanım dili ile yazabilecek duruma gelinmiştir.

Çalışmadaki ikinci aşama olarak VHDL dilinin kullanılacağı Xilinx ISE adlı program hakkında bilgi edinilmiştir. Xilinx ISE programı bilgisayara kurularak programın özellikleri incelenmiştir. Ayrıca simülasyonların yapılacağı ModelSim adlı program da bilgisayara kurularak bu program hakkında da bilgi edinilmiştir. Böylece algoritmanın yazım aşamasına hazır hale gelinmiştir.

Algoritmanın gerçeklenmesinde ilk adım olarak N bitlik bir ötelemeli kaydedici (shift register) tasarlanması hedeflenmiştir. 288-bitlik iç durum için bir kaydedici kullanılması uygun görülmüştür. Bu amaçla 'reg_triv' adlı VHDL kodu yazılmıştır. Bu kodda bir adet sıfırlama (reset) girişi ve bir adet de kontrol girişi kullanılmıştır. Kontrol girişinin '1' değeri

için paralel yükleme, ‘0’ değeri için ise sağa öteleme yapması sağlanmıştır. Simülasyonun kolay anlaşılabilir olması amacıyla N sayısına ilk değer olarak 8 değeri verilmiştir, yani 8 bitlik bir ötelemeli kaydedici tasarlanmıştır. Şekil 4.2.’ de ModelSim’ de elde edilen simülasyon sonucu verilmiştir. Simülasyon sonucundan da görüldüğü gibi sıfırlama girişinin ve kontrol girişinin doğru çalıştığı ve kaydedicinin ötelemeyi gerçekleştirdiği gözlenmiştir.

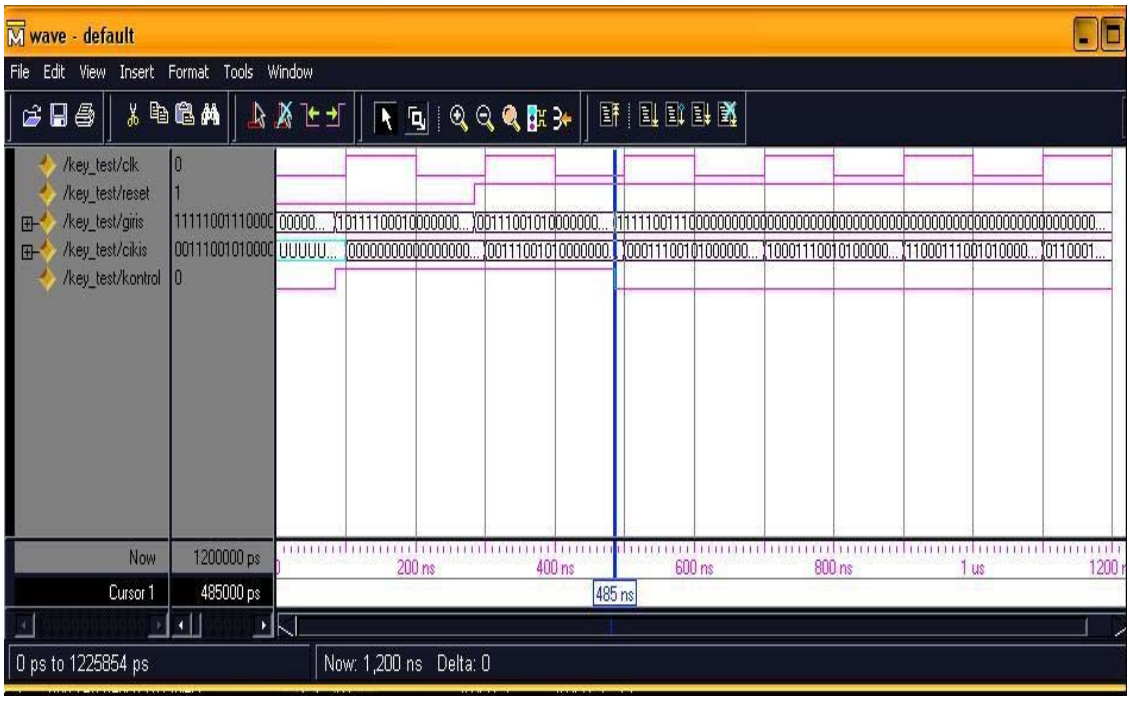
Gerçeklemenin ikinci adımı olarak gizli anahtar ve ilk değer kullanılarak şifreleyicinin başlangıç konumuna getirilmesine ilişkin VHDL kodunun yazılması amaçlanmıştır. Bu aşamada, daha önceden yazılmış olan ötelemeli kaydedici bileşen olarak kullanılarak tasarım yapılması düşünülmüş fakat bileşen olarak başka bir dosyanın çağırılması devrenin saat frekansını düşüreceğinden



Şekil 4.2. Ötelemeli Kaydedici Simülasyon Sonucu

vazgeçilmiştir. Bileşen kullanmak yerine ‘key_IV_setup’ adlı kodda 288-bitlik geçici bir kaydedici (temp) tanımlanmıştır. Kaydedicide olduğu gibi yine sıfırlama (reset) ve kontrol girişleri kullanılmıştır. Ayrıca gizli anahtar ve ilk değer 160-bitlik ‘giris’ adlı giriş vektörüne aktarılmıştır. Kontrol girişi ‘1’ için giriş adlı vektör geçici kaydediciye aktarılmış ve başlangıç konumuna getirilmiştir. Kontrol ‘0’ için ise sağa kaydırarak 4 tam tur döndürme işlemi gerçekleştirilmiştir. Son olarak da geçici kaydedicinin içeriği ‘cikis’ adlı çıkış vektörüne aktarılmıştır. Şekil 4.3.’ te ModelSim programında elde edilmiş simülasyon sonucu görülmektedir. Şekilden de görüldüğü gibi ‘reset’ girişi ‘0’ için çıkış sıfırlanmakta, ‘kontrol’ girişi ‘1’ için anahtar ve ilk değer çıkışa yüklenmekte, 485. ns’ den itibaren ise ‘kontrol’ girişi ‘0’ için sağa öteleme işlemi gerçekleştirilmektedir. Çıkış vektörünün içeriğinin 3 saat darbesi için sağa ötelendiği yani kodun doğru çalıştığı gözlenmektedir.

Gerçeklemenin üçüncü adımında ise Trivium Dizi Şifreleme Algoritmasının tamamının VHDL kodunun yazılması amaçlanmıştır. Bu amaçla anahtar bitinin üretilme kısmının kodu yazılmış ve başlangıç konumuna getirme programı ile birleştirilmiştir. Böylelikle tüm algoritmanın kodu elde edilmiş ve ‘bitirme’ adı altında kaydedilmiştir. Bitirme programında başlangıç konumuna getirme programına ek olarak bir de üretilen anahtar bitlerinin görüldüğü ‘key_cikis’ adlı bir çıkış vardır. Trivium algoritması tarafından üretilen anahtar bitleri



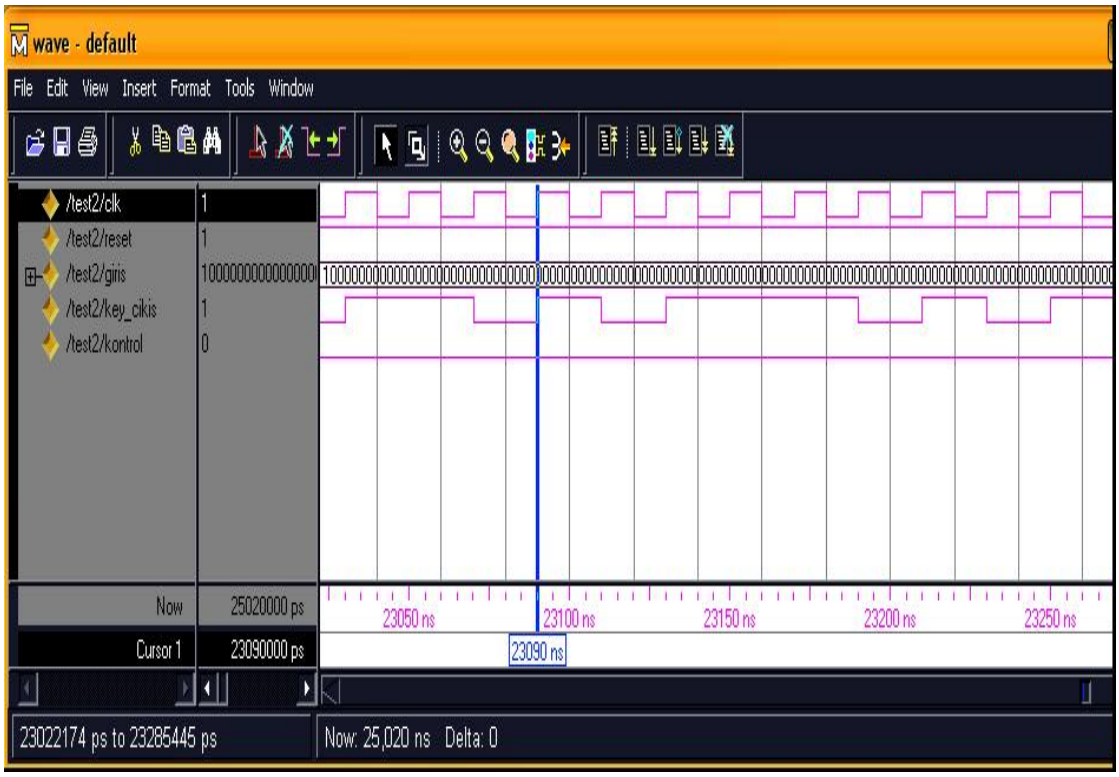
Şekil 4.3. Başlangıç Konumuna Getirme Programı Simülasyon Sonucu

'key_cikis' çıkışında görülmektedir. Şekil 4.4.' te ModelSim programında elde edilen davranışsal simülasyon (behavioral simulation) sonuçları verilmiştir. Şekilde görülen 23090. ns' de başlangıç konumuna getirilmiş olan kaydedicinin sağa kaydırılarak 4 tam tur döndürülmesinden hemen sonra üretilen ilk anahtar biti görülmektedir. Yani 50. ns' de 'kontrol' girişinin '0' olması ile sağa kaydırma işlemi başlamış, 4 * 288 = 1152 saat darbesi sonra ise anahtar dizisinin bitleri üretilmeye başlanmıştır. Saat darbesinin periyodu 20 ns olduğundan ilk gerçek anahtar bitinin üretilmesi için 1152 * 20 = 23040 ve 23040 + 50 = 23090 ns beklenmelidir. Bu andan itibaren $N \leq 2^{64}$ olmak üzere kullanıcı isteğine bağlı olarak N tane anahtar biti elde edilebilir.

4.3. Algoritmanın Test Edilmesi

Algoritmanın test edilmesi aşaması iki ayrı yöntemle gerçekleştirilmiştir. İlk olarak 'bitirme' adlı VHDL kodu 4 adım için elle test edilmiştir. Bunun için algoritmada anahtar bitinin üretilmesinde kullanılan 15 özel bite rastgele değerler verilmiştir. Daha sonra algoritmanın 4 adım boyunca ürettiği anahtar bitleri elle hesaplanmıştır. Elde edilen sonuçlar ModelSim programında elde edilen davranışsal simülasyon sonuçları ile karşılaştırılmıştır. Sonuçların aynı olmasıyla programın 4 adım için doğru çalıştığı ortaya çıkmıştır.

Algoritmanın test edilmesinde ikinci adım olarak üretilen tüm anahtar bitlerinin kontrol edilmesi yani yazılan 'bitirme' kodunun tamamen doğru çalıştığının ispatlanması hedeflenmiştir. Bu amaçla algoritma MATLAB programında da yazılmıştır. Bu programda da 'bitirme' kodu 4 * 288 = 1152 adım boyunca çalıştırılmış ve anahtar bitlerinin üretilmesi sağlanmıştır.



Şekil 4.4. Bitirme Programının Simülasyon Sonucu

Anahtar bitleri 100 adım boyunca üretilmiş ve ModelSim programında üretilen anahtar bitleriyle karşılaştırılmıştır. Sonuç olarak ikisinin de aynı bitleri ürettiği gözlenmiş yani VHDL kodunun doğru çalıştığı ispatlanmıştır. Yazılan bütün kodlar ve simülasyon sonuçları ile üretilen anahtar bitleri ek olarak verilen CD de bulunmaktadır.

4.4. FPGA Üzerinde Gerçekleme

FPGA üzerinde gerçekleme işlemi, 'bitirme' adlı VHDL kodunun sentezlenmesiyle başlamaktadır. Sentezleme işleminin gerçekleştirilebilmesi için kodun kullanılacak olan FPGA' e uygun olarak yazılmış olması gerekmektedir. Bu çalışmada kullanılan FPGA, Xilinx firmasına ait olup Virtex-E (XCV1000E)' dir. Kullanılan FPGA' in Giriş / Çıkış sayısı 162' dir. 'bitirme' adlı kodda giriş vektörünün uzunluğu 160 bit olduğundan kod sentezlenemez. Bu sorunu çözmek için giriş vektörü 80 bite düşürülmüş ve önce IV sonra da tohum (gizli anahtar) alınmıştır. Bu düzeltmeden sonra devre sentezlenmiştir. Elde edilen sentezleme raporuna göre alan ve zaman bilgileri aşağıdadır:

Selected Device : v1000ehq240-8

Number of Slices:	167	out of	12288	1%
Number of Slice Flip Flops:	288	out of	24576	1%
Number of 4 input LUTs:	168	out of	24576	0%
Number of bonded IOBs:	84	out of	162	51%
Number of GCLKs:	1	out of	4	25%

Timing Summary:

Speed Grade: -8

Minimum period: 3.480ns (Maximum Frequency: **287.356MHz**)

Minimum input arrival time before clock: 9.456ns

Maximum output required time after clock: 8.531ns

Maximum combinational path delay: No path found

Devrenin sentezlenmesinden sonra yerleştirme ve yollandırma (place and route) işlemi gerçekleştirilmiştir. Yaklaşık 3 saat kadar süren yerleştirme yollandırma ile simülasyonu sonucunda anahtar çıkışında sürekli kısa süreli inişler çıkışlar görülmüştür. Bu sebeple kodda değişiklikler yapılmış ve 'bitirme' kodunun son halinin olduğu 'tarik' adlı proje oluşturulmuştur. Bu kod sentezlendiğinde alan bilgileri değişmemiş fakat zaman bilgileri değişmiş, saat frekansı düşmüştür. Ancak iniş çıkışlar engellenmiştir. Sentez sonuçları aşağıdadır:

Speed Grade: -8

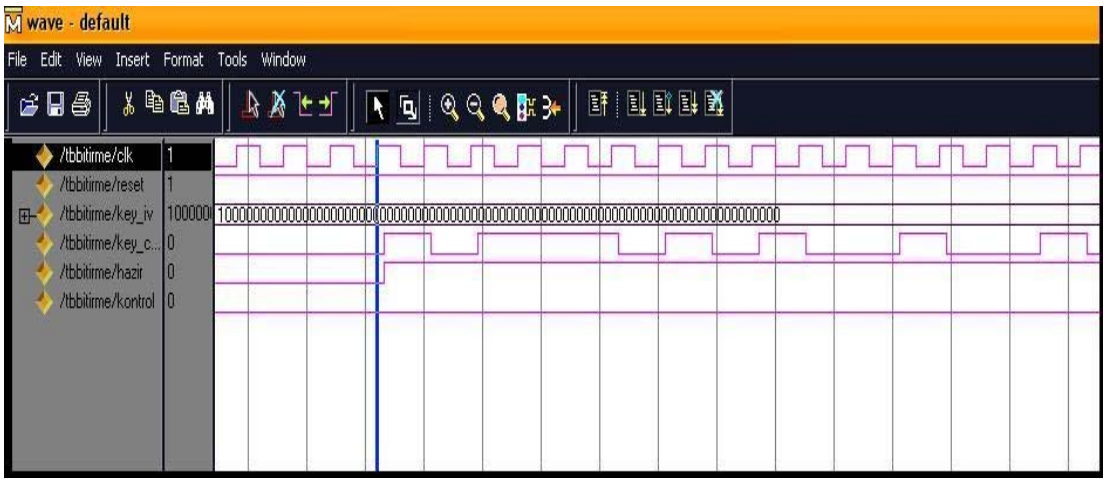
Minimum period: 5.238ns (Maximum Frequency: **190.913MHz**)

Minimum input arrival time before clock: 10.424ns

Maximum output required time after clock: 5.783ns

Maximum combinational path delay: No path found

Yerleştirme ve yollandırma simülasyonu sonuçları yani FPGA üzerinde gerçekleme işleminin simülasyon sonucu Şekil 4.5' te verilmiştir. Bu simülasyon sonucu üretilen anahtar 'hazır' çıkışının '1' olduğu anda başlamaktadır ve daha önce üretilen anahtar ile aynıdır. Sonuç olarak FPGA üzerinde gerçekleme işlemi başarı ile sonlandırılmıştır.



Şekil 4.5. FPGA üzerinde Gerçekleme Sonucu Simülasyon

5. SONUÇLAR

Trivium Dizi Şifreleme Algoritması' nın FPGA üzerinde gerçeklenmesi bitirme ödevinde sırasıyla çalışmanın amacı, dizi şifreleme sistemleri, FPGA yapısı ve programlanması ile Trivium Algoritması ve gerçeklenmesi geniş bir biçimde anlatılmıştır. Yapılan bu ödevde, dizi şifreleme konusunda Avrupa Birliği Standardının seçilmesi amacıyla düzenlenen ECRYPT Dizi Şifreleme Projesi' ne aday olan algoritmalardan bir tanesi olan Trivium Algoritmasının gerçeklenmesinin yanında şifreleme konusunda geniş bilgi edinilmiş, donanım betimleme dillerinden biri olan VHDL öğrenilerek şifreleme algoritmalarını yazabilecek duruma gelinmiş, algoritmaların yazımı ve denenmesi için Xilinx ISE ve ModelSim programları detaylı bir şekilde öğrenilmiş ve kullanılmış, ayrıca FPGA' lar hakkında bilgi edinilerek bu diziler kullanılmış ve algoritmanın işlevsel hale dönüşmesi sağlanmıştır. Ayrıca Kriptografi alanında Avrupa Birliği standardı olabilecek bir algoritmanın gerçeklenmiş olması ve bu algoritma hakkında yorum yapabilecek sayılı insanlardan biri haline gelmiş olmak başarılan diğer bir hedeftir.

Kriptografi ve özellikle dizi şifreleme alanında Türkiye' de yapılan çalışmaların oldukça az olması ve Türkçe kaynak bulmanın zor olması sebebiyle yapılan bu çalışma, gelecekte bu konuda yapılacak olan diğer çalışmalara bir örnek olacak niteliktedir. Konuya genel bir bakışın bulunması, gerekli temel tanımlamaların yapılmış olması ve dizi şifreleme konusunda Trivium Algoritması' nın basit ve anlaşılır bir örnek olması Kriptografi konusuna ilgi duyanlar için bu çalışmayı güzel bir kaynak haline getirmiştir.

KAYNAKLAR

[1] **Savaş, E.,** 1994. Dizi Şifreleme Sistemleri ve Doğrusal Karmaşıklık, *Yüksek Lisans Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

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

[3] **Apohan, A.M.,** 1993. Kriptoloji, *Yüksek Lisans Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

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

[5] http://www.xilinx.com

[6] **De Canniere, C.,** 2005. Trivium Specifications , Belgium.

[7] **Hsu, Y.C., Tsai, K.F., Liu J.T. and Lin, E.S.,** 1995. VHDL Modeling For Digital Synthesis, Kluwer Academic Publishers, Riverside.

EKLER

EK-A : Uygulamanın kaynak kodları (CD'ye kaydedilmiştir).

ÖZGEÇMİŞ

TARIK KAPLAN
0 536 351 92 91 - 0 544 247 16 96
tarikkaplan2000@yahoo.co.uk kaplant@itu.edu.tr
Bahçeköy Öğrenci Yurdu, Hacıosman Bayırı Sarıyer - İSTANBUL

EĞİTİM

2001-2005 **Elektronik Mühendisliği, İstanbul Teknik Üniversitesi**
1 yıl **İngilizce Hazırlık** Eğitimi
Bitirme Projesi: **Kriptografi**: "Trivium" Dizi Şifreleme Algoritmasının VHDL ile FPGA Üzerinde Gerçeklenmesi
ORT **3.50 / 4.00**

1995-2001 **M.N. Çakallıklı Anadolu Lisesi**
7 yıl (Ortaokul + Lise)
ORT 4.90 / 5.00

İŞ DENEYİMİ

2005	**Innova Bilişim Çözümleri A.Ş.** Network and Security'ye Giriş	İstanbul	Yaz Stajı
2004	**Telkom Bilişim** Yaz Stajı Network Temelleri	Antalya	
2003	**TRT Bölge Vericiler Müd.** Yaz Stajı	Antalya	
2003-2004	**İTÜ Bilgi-İşlem Daire Başkanlığı** Asistan Öğrenci (Yarı Zamanlı)	İstanbul	

DİLLER

- İngilizce (Akıcı)
- Türkçe (Anadil)
- Almanca (Başlangıç)

BİLGİSAYAR BİLGİLERİ

- Windows İşletim Sistemleri, MS Office Programları
- C dili ile programlama
- Elektronik Mühendisliği Programları: VHDL, MWOffice, Pspice and MicroSim, Orcad Spice, Circuit Maker, Xilinx and ModelSim
- Mayasoft Bilişim'de **Cisco CCNA** kursu
- Symantec Internet Security ürünleri ile ilgili

.

KİŞİSEL

- Doğum Tarihi ve Yeri : 21/08/1983, Antalya
- Medeni Hali : Bekar
- Ehliyet : B sınıfı