1. GİRİŞ

Bu çalışmada, steganografi sisteminin FPGA üzerinde tasarımı ve gerçeklenmesi sağlanmıştır. Eski Yunancada gizlenmiş yazı anlamına gelen steganografi, bilginin görünürlüğünü gizleme bilimine verilen isimdir. Günümüzde karşılaşılan en büyük yanlış anlama steganografinin şifreleme ile karıştırılmasıdır. Veriyi gizleme sanatı olarak bilinen bu bilimin şifrelemeye göre en büyük üstünlüğü bilgiyi gören bir kimsenin gördüğü şeyin içinde önemli bir bilgi olduğunu fark edemiyor olmasıdır, böylece içinde bir bilgi aramaz oysa bir şifreli mesaj, çözmesi zor olsa bile, gizemi dolayısıyla ilgi çeker. Çünkü bir bilginin gizlendiği bellidir [3]. Günümüzde steganografi bilimi sayesinde ses, video, resim dosyalarına, disketlere ve haberleşme kanallarına istenilen veri gizlenebilmektedir [1].

Çalışmada steganografinin resim alanındaki uygulamaları donanımsal olarak gerçeklenmiştir. Gerçekleme aşamasına geçmeden önce konu araştırması yapılmış, buna göre resimlerin çeşitli noktalarda, ışık yoğunlukları ile temsil edilen sayı dizileri olduğu öğrenilmiştir. Bu sayı dizileri esasında yalnızca 0 veya 1'lerden yani bitlerden oluşan yapılar olduğundan veri gizlenirken bu durumdan yararlanılmıştır.

Steganografide resmin içine veri gizlerken en çok kullanılan yöntem resmi oluşturan piksellerin en anlamsız bitlerin kullanıldığı uygulamalarıdır. Bunun nedeni en düşük anlamlı bitte yapılan değişiklikleri insan gözünün fark edememesidir. Yani en anlamsız bitlere saklanan veriyi içeren resimle, esas resim yan yana konulduğunda gözle görülür hiçbir fark yoktur. Bu durumdan yola çıkılarak steganografide resmin içine veri gizlenirken çeşitli yöntemler geliştirilmiştir [1].

Bu çalışmada bu yöntemlerden üç tanesi kullanılmıştır. Bunlardan birincisinde bir resmi oluşturan tüm piksellerin son bitleri değiştirilmiş ve bu değişimin bir fark yaratmadığı anlaşılmış, ikincisinde ilk pikselden başlanarak veri gizlenmiş daha sonra da aynı şekilde çözümlenmesi yapılmıştır. Son yöntemde ise verinin hangi piksellere gizleneceğini belirten bir steganografik anahtar kullanılarak veri gizlenmiş ve tekrar elde edilmiştir.

2. STEGANOGRAFİ

Steganografi veriyi gizlemenin ve zararsız taşıyıcılarla veriyi taşımanın sanatıdır. Kelime anlamı olarak kökleri στεγαυος ve γραΦειν 'ye ait olmakta ve Yunan alfabesinden gelmekte olup kaplanmış yazı anlamına gelmektedir. Yunanca steganos sözcüğü gizli, saklı ve grafi sözcüğü çizim ya da yazım demektir. Bu sanat var olan veriyi gizlemede birçok gizli haberleşme tekniği kullanmaktadır. Steganografi eski bir el sanatı olmasına rağmen günümüzde bilgisayar teknolojisiyle yeni bir içerik kazanmıştır. Bilgisayar tabanlı steganografi teknikleriyle veriyi yeni gizleme teknikleri geliştirilmiştir [1].

Bilgiler metin formunda, sayısal 1 ve 0 olarak ya da başka çeşitlerde iletime geçerken verinin kime ait olduğunu belirten bir çeşit parmak izi bırakırlar. Steganografi bir çeşit kriptoloji gibi düşünülebilir. Her ikisi de haberleşme sırasında kaydedilmiş veriye bilgi ekleyerek çalışırlar. Kriptoloji teknikleri bilgiyi belirli algoritmalara dayanarak şifreleyip güvenli bir örtü yaratmayı amaçlar. Steganografi ise kriptolojiden farklı olarak veriyi örterek gizlemeyi sağlar. Kriptolojide şifreli metin olarak adlandırdığımız örtülü yapı dikkat çekebilirken steganografide kendini gizlediğinden dikkat çekmemeyi sağlar. Bu da verinin güvenli bir şekilde taşınması açısından önemli ve yararlı bir durum oluşturmaktadır [7].

2.1 Steganografi Tarihçesi

Geçmişten bugüne kadar bu alanda veriyi gizleme amacıyla birçok değişik teknik kullanılmış ve geliştirilmiştir. Steganografinin geçmişte bilinen ilk örneği olarak kabul edilen hikâye Heredotus'a aittir. Eski Yunanistan'da Herodot 486-425 B.C. tarihleri arasında Histiaeus adlı çok güvendiği bir kölesinin kafasını kazıtmış ve dövme yaptırmıştır. Dövmede yer alan bu gizli bilgi kölenin saçları uzadığında gözükmemektedir. Böylece Persler fark etmeden bilgi iletimi sağlanmıştır. Eski çağlarda kullanılan bir başka yöntem ise 1. ve 2. Dünya Savaşı'nda kullanılan mektup aracılığıyla görünmez mürekkeplerle satır aralarında veriyi taşımaktır. Bu görünmez mürekkep içinde süt, meyve suyu gibi bileşenler içermekte olduğundan

veri eline ulaşan kişi bu mektubu ısıttığı zaman veriyi okuyabilecek hale gelmiş olmaktadır. Daha sonra Almanlar 2. Dünya Savaşı'nda birbirlerine yolladıkları sıradan bir metnin içindeki her sözcüğün yalnızca baştan ikinci harflerine veriyi gizlemişler, bu ikinci harfleri yan yana getirince oluşan cümleyle iletililerini iletmişlerdir. Aşağıda bir Alman casusun 2.Dünya Savaşı'nda gönderdiği bir metin örneği yer almaktadır:

"Apparently neutral's protest is throughly discounted and ignored. Isman hard it Blockade issue affects pretext for embargo on byproducts,ejecting suets and vegetable oils."

Her kelimenin ikinci harflerini aldığımızda oluşan mesaj şu şekildedir:

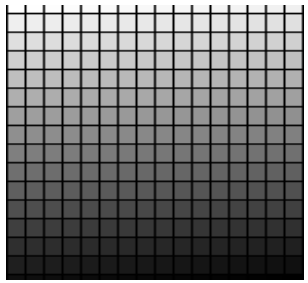
" Pershing sails from NY June 1."

Bu gizli verileri çözümleme yolları geliştikçe veriyi saklama yolları da zamanla gelişti. Almanların geliştirdiği microdot tekniğiyle veri normal boyutta sözcükler içeren bir metnin içine çok ufak yazılarak saklandı. 1999 yılında ise New York'taki Mount Sinai School of Medicine veriyi DNA içinde organik bazların yerini değiştirerek saklamayı başardı. Böylece bir milyon veri yalnızca bir gram DNA içinde saklanabilir oldu. Bilgisayar teknolojisi geliştikçe işletim sistemlerinde veri gizlenmeye ve taşınmaya başlandı. Windows 95 FAT 16 sisteminde 1 kilobaytlık veri bile işlem görse 32 kilobaytlık yer ayırdığını fark eden uzmanlar bu kullanılmayan alanla veriyi gömmeye başladı. TCP/IP paketlerinin kullanılmayan yerlerine de bilgi gizlendikten sonra steganografi günümüzde çok büyük bir gelişme göstermiştir. Bilgisayar teknolojisiyle ve internetle yeni bir hayat bulan steganografi günümüzde yalnızca metin değil, ses ve video dosyalarına sayısal formda 1 ve 0 olarak kodlanarak uygulama alanı bulmuştur [1].

3. SAYISAL RESİMLER

Resim, piksel adı verilen çeşitli noktalarda, ışık yoğunlukları ile temsil edilen sayı dizisidir. Genel olarak bir resim 640 x 480 pikseldir. Böyle resimler 2^{24} yani yaklaşık 300.000 pikselden oluşabilirler. Pikseller genelde 24 bit ya da 8 bit şeklinde depolanırlar. 24 bit piksel yaklaşık olarak 16.777.216 tane olası renk kombinasyonu içerir ve sıkıştırılmamış resim dosyaları çok büyük olabilirler. Bütün renk çeşitleri üç ana renkten türetilmiştir. Bunlar kırmızı, yeşil ve mavidir. Genel olarak 24 bit resimlerde her ana renk bir bayt yani 8 bit ile temsil edilir. Her bayt, rengin yoğunluğunu temsil eder ve 0 ile 255 arası değerler alır. En koyu yoğunluk değeri 0 iken en açığı ise 255'tir. Örneğin, renkler 6 basamaklı 16'lı sayılar ile temsil edilebilir. Burada her üçlü gurup sayı kırmızının, yeşilin ve mavinin yoğunluğunu temsil etmektedir [1].

8 bit resimler 256 renkli ya da siyah-beyaz resimler olabilir. 8 bit resimler 256 rengi temsil eden bir renk indeksi kullanırlar. GIF gibi 8 bitlik resimlerde her piksel bir bayt ile temsil edilir ve her piksel sadece renk paletinde tek bir rengi işaret etmektedir. Piksel değerleri (0 ile 255 arası bir değer) renk paletinde rengin pozisyonuna karşılık gelir. Ne zaman 8 bitlik bir resim görüntülense, yazılım belirtilen rengi ekranda, belirtilen pozisyonda boyar. Gerçek siyah-beyaz resimlerde ise, paletler bulunmaz ve 0 dan 255 e kadar olan değerler rengin ve ışığın yoğunluğunu temsil eder.



Şekil 3.1 Gri Ölçek Paleti [6]

Şekil 3.1'de siyah-beyaz renklerin değerlerindeki kademeli değişim gözükmektedir. Siyah-beyaz renkli resimlerde renk tonu bayttan bayta kademeli olarak değişir. Üsteki 256 renk tonu barındırmaktadır. Bazı resimler ise 4 bit'tir ve bu resimlerde sadece 16 tane gri tonu bulunur. Bu tip resimler bu nedenle çok az renk çeşidi sunar. Piksel gösteriminin getirdiği özelliklerden bir tanesi dosya büyüklüğünü arttırmasıdır. Örneğin, 1024 x 768 pikselden oluşan 24 bit'lik bir resimde 768.432 piksel bulunur ve bu yaklaşık olarak 2MB civarında bir dosya yaratır. Dosya sıkıştırması böyle bir dosyanın taşınmasında faydalı olur [1].

3.1 Sayısal Resim Dosyalarının Sıkıştırılması

Sayısal resim dosyalarını sıkıştırma ile iki çeşit yapı oluşur. Bu yapılar kayıpsız ve kayıplı resim yapılarıdır. Her iki tür de depolamada farklı sonuçlar verir. Kayıpsız sıkıştırma tipik GIF ve 8 bit bitmap dosyaları olarak kaydedilir. Diğer yandan kayıplı sıkıştırma dosyayı kaydederken, dosyanın özgünlüğünü koruyamamaktadır. Bu yöntemde resimler JPEG olarak kaydedilir. JPEG dosyası 24bit bitmap dosyasının renklerine yaklaşırken, sıkıştırmadan dolayı bazı veri kayıplarına neden olur. GIF ve JPEG dosyaları internette en yaygın kullanılan dosya formatlarıdır. Başka bir yaygın olan dosya türü ise bitmap formatıdır. Sıkıştırılmamış bitmap resimler 24 bittir [1].

3.2 Sayısal Resimlerde Bilgi Saklama

Veri, resimlerde birçok farklı yol ile sağlanabilir. Bilgi saklamak için, basit bilgi eklenmesi resmin her bir bitinin içine şifrelenerek sağlanabilir ya da dikkat çekmeyecek yoğun alanları seçerek mesajı gömme işlemi yapılır. Bir veri resmin içinde gelişigüzel şekilde dağıtılabilir ve ya resim içerinde birçok defa tekrarlanabilir [1].

En önemsiz bit uygulamaları (maskeleme ve filtreleme) ve daha karmaşık resim işleme algoritmaları ve dönüşümleri kullanmak sayısal resimlerde bilgi saklamanın bazı yaygın uygulamalarıdır. Bu teknikler her türlü formattaki dosyaya farklı başarı oranlarıyla uygulanabilir [2].

Çoğu resim tabanlı steganografi yöntemleri 256 gri renk tonu olan resimlerde en iyi şekilde uygulanır. Baytlar arasındaki kademeli yoğunluk değişimi bilgi saklandıktan sonra çok az görsel bozukluk oluşturur. Siyah-beyaz resimler bazı steganografi

yazılımlarında en iyi sonucu verse de, kurnaz renk çeşitleri de gayet etkili olabilir. Resim içine bilgi gizleme yapılır iken, resmin yapısı ile palet çok iyi düşünülmelidir.

Her birinin bilgi şifrelemesini kolaylaştırdığı ve zorlaştırdığı karakteristikleri vardır [1].

Şekil 3.2 Veri Gizlenmemiş Yabani Tavşan Resmi [4]

Şekil 3.2'de belirli bir algoritmaya dayalı veri gizleme yöntemi kullanılmadan önce yani resmin içine veri gizlenmemiş haliyle yabani tavşan resmi yer almaktadır [5].

Şekil 3.3 İçine Veri Gizlenmiş Resim [4]

Şekil 3.3'te esas resmin içine bir algoritmaya bağlı kalınarak veri gizlenmiştir. Buradan da görüldüğü gibi resimlerde gözle görülür bir fark yoktur [4].

Resimlerde steganografi uygulamalarında verilerin güvenli iletimi için resim seçimi de önemlidir. Örneğin içinde birçok varlık olan resimler daha kalabalık olduğu için bilgi saklama adına daha iyi bir örtü oluşturmaktadır ve kalabalık alanları fazla olan resimler bilgi şifreleme adına akılcı seçeneklerdir; çünkü içerisindeki değişimler gömülü mesajı daha az görünür yapar. Bu tür resimlere gömülen veriler kolayca belli olmamaktadır. Bu nedenle bir resim daha yakın renkler ve daha az doku içeriyorsa, kalabalık ve daha renkli bir resim veri gizlemede tercih sebebi olmalıdır [1].

3.2.1 En Düşük Anlamlı Bite Bilgi Saklama

Resim içerisinde veri gizlerken en çok kullanılan yöntem resmi oluşturan piksellerin en düşük anlamlı bitine veriyi gömmektir. Bu yöntemin bu kadar tercih edilmesinin sebebi en son bitte yapılan değişikliğin gözle görülür bir fark yaratmamasındandır[3].

Eğer veri gizlemek istediğimiz resim 24 bitlik ise bu resmin her baytının en önemsiz bitine veri gizlemek için, en fazla piksel basına 3 bit kullanılabilir. 1024 x 768'lik bir resim 2.259.296 bitlik bir gizle potansiyeli barındırmaktadır. Eğer veri gizlemeden önce sıkıştırılırsa, büyük miktarda veri gizlenebilir. İnsan gözüyle bakıldığında orijinal resimdekinden olan farkı belli olmaz [1].

Örneğin, 'A' harfi 3 piksel içine saklanabilir (sıkıştırma olmadan). 3 pikselin veri gizlenmemiş haldeki verisi:

00100111 11101001 11001000

00100111 11001000 11101001

11001000 00100111 11101001

İkilik tabanda 'A' nın değeri 10000011 dir. 'A' nın iki tabanındaki değerini 3 piksele girdiğimizde sonuç:

00100111 1110100*1* 11001000

00100110 11001000 11101000

11001000 00100111 11101001

Altı çizilen bitler 8 baytlık kullanımda değişen 3 biti göstermektedir [1].

8 bitlik resimlerde ise sadece son biti değiştirilerek bu yöntem uygulanır. Steganografiyi resim piksellerinin düşük değerlerine uygulamak resmin kompozisyonuna bağlıdır. Yüksek sıklıkta alanlar içeren resimler üzerinde daha fazla yönlendirme yapılabilir. En düşük bite veriyi gizleme yöntemi bilgi saklamak için hızlı ve basit bir yoldur. Fakat resmin işlenmesinden ve kayıplı sıkıştırmadan kaynaklanan küçük değişiklere karşı duyarlılığı da vardır. Bu nedenle steganografinin resim alanındaki uygulamalarında kayıplı bir format olan JPEG formatından daha çok kayıpsız bir resim formatı olan bitmap tercih edilir [2].

4. STEGANOGRAFİ UYGULAMASI

Günümüzde resim alanında yapılan steganografi uygulamalarında çeşitli algoritmalar ve yöntemler kullanılmaktadır. Bunlardan en yaygın olanı resmi oluşturan piksellerin en düşük anlamlı bitinde yapılan değişiklikleri kapsar. Bu çalışmam kapsamında bu yöntemlerden üç tanesi yer almaktadır.

4.1 Uygulamada Kullandığım Yöntemler

Çalışmada resmi oluşturulan piksellerin son bitinde yapılan değişikliklerle üç yöntem gerçeklenmiştir. Gerçeklenme aşamasından sonra elde edilen resimlerle esas resimlerle kıyaslandığında gerçekten gözle görülür bir fark olmadığı ortaya çıkmıştır. Çalışma esnasında en yüksek verim alınabilmesi için resimler için kayıpsız bir sıkıştırma algoritması olan bitmap formatı seçilmiştir.

4.1.1 En Düşük Anlamlı Bitlerin Tümünün Değiştirilmesi Yöntemi

Bu yöntemle steganografinin kapsamında anlatılanları gerçekleyerek gerçekten kullanılan ve değişen resimler arasında fark olmadığını görülmesi sağlanmıştır. Öncelikle resimler piksellerden oluştuğu ve matris halinde bu pikseller dizildiği için bir metin dosyasına aktarım işlemi yapılması gerekiyordu. Bunun için wnbrowse editörü ile şekil 4.1'deki resmin hex formatında görülmesi sağlanmıştır.

Şekil 4.1 F15 Uçak Resmi[4]

Oluşan 205 x 138' lik içinde sadece hex formatında pikseller içeren bir yapı oluşmuştur. Şekil 4.2'de bu matrisin sadece ilk 10 x 10'luk kısmı alınmıştır. Yapılan işlemlerin anlatımına bu örnek parça üzerinden gideceğim.

42	4D	76	30	05	00	00	00	00	00
00	00	9A	01	00	00	14	01	00	00
00	00	00	00	00	00	C4	0E	00	00
00	00	00	00	00	00	C0	86	48	C4
86	49	C1	84	46	C2	85	49	C1	85
47	C1	86	4B	C1	84	46	C4	84	46
C1	86	46	C3	86	45	C0	86	48	C2
89	44	C4	87	43	C3	86	45	C3	85
49	C4	87	45	C2	88	47	C5	8A	49
C2	86	4A	C4	84	46	C2	87	47	C0
82	45	BF	81	46	BD	85	47	C2	87

Şekil 4.2 Esas Resmin İlk 10x10'lık kısmının Hex Formatında Görünümü

Burada 8 bitlik yapılardan yani baytlardan oluşmuş bir matris görmekteyiz. Bu matriste her satır ve sütun 10 bayttan oluşmaktadır. Yani 00 ile FF arasında değerler alan ikili hex yapılarından her satır ve sütunda 10'ar tane vardır. Örneğin ilk satırın ilk elemanı 42'dir. Burada 4 dört bitten, 2 dört bitten oluşan 42 olarak yan yana geldiklerinde 8 bit yani bir bayt eden yapı vardır.

8 bitlik bu yapıya resimde piksel adı verilmektedir. Resmin boyutu değiştikçe matris boyutu da dolayısıyla resmi oluşturan piksel sayısı da değişmektedir [2].

Steganografideki en yaygın yöntem olan en düşük anlamlı bitin değiştirilerek uygulanmasını burada matriste gerçeklenmesi sağlanmıştır. Yöntemde belirtilen yapıya göre bir resmi oluşturan piksellerin her birinin son bitlerini değiştirilirse yani 0 ise 1, 1 ise 0 yapılırsa yeni oluşan resimle eskisi kıyaslanırsa arada gözle görülür bir fakın olmayacağıdır [7].

Şekil 4.2'deki kaynak resmin ilk 10 x 10'luk kısmının piksellerine bakacak olursak her pikselin son biti 1 ise 0, 0 ise 1 yapılarak yöntem gerçeklenmiştir. Bunun için her hex yapısı ikilik düzende düşünülmüştür. Örneğin ilk baytı ele alalım. 42 ikilik düzende 0100 0010'dir. Yazılan algoritmayla bu 8 bitin sadece son biti lojik değil kapısına sokularak 0100 0011 yani 43 elde edilmiştir. Bu işlemin resimdeki her piksele uygulanması sağlanmıştır ve şekil 4.3'deki matris elde edilmiştir.

43	4C	77	31	04	01	01	01	01	01
01	01	9B	00	01	01	15	00	01	01
01	01	01	01	01	01	C5	0F	01	01
01	01	01	01	01	01	C1	87	49	C5
87	48	C0	85	47	C3	84	48	C0	84
46	C0	87	4A	C0	85	47	C5	85	47
C0	87	47	C2	87	44	C1	87	49	C3
88	45	C5	86	42	C2	87	44	C2	84
48	C5	86	44	C3	89	46	C4	8B	48
C3	87	4B	C5	85	47	C3	86	46	C1
83	44	BE	80	47	BC	84	46	C3	86

Şekil 4.3 Son Bitleri Değişmiş Resmin İlk 10 x 10'luk Kısmının Görünümü

Sadece son bit değiştirilmesi yazılan VHDL koduyla elde edilmiş ve hedef adında bir metin dosyasına yöntem dâhilinde yazdırılmıştır. Bundan sonra gereken işlem 205 x 138 formatındaki piksellerden resim elde etmektir. Ancak bu şekilde bitleri değiştirilmeden önceki resimle farkının olup olmadığı karşılaştırabilmektedir [2].

Bu aşamada Matlab programından yararlanılmıştır. Her pikselin son bitleri değişmiş halinde elimizde bulunan hedef dosyası Matlab programında yazılmış olan kod yapısıyla bitmap formatında resme dönüştürülmüştür. Sonuçta tüm piksellerin son bitleri değişmiş haliyle elde edilen resim şekil 4.4'deki gibidir.

Şekil 4.4 F15 Uçak Resmi

Bu iki resimde burada da görüldüğü gibi gözle görülür hiçbir fark yoktur. Bu yöntemle bir resmi oluşturan piksellerin son bitleri lojik olarak değil kapısına girdiğinde yeni oluşan resmin eskisinden farkının insan gözünün fark edemeyecek kadar az olduğunu görmüş olduk. Bundan sonra gerçeklenen iki yöntem bu durumun kullanılmasıyla elde edilen yöntemler olmuştur.

4.1.2 Bir Veriyi En Düşük Anlamlı Bitlere Gömme ve Elde Etme Yöntemi

Bir resmi oluşturan tüm piksellerin son bitlerinin değişimi resimde fark edilir bir ayrım yaratmadığından ilerleyen steganografik çalışmalarda bu en düşük anlamlı bitlerle veri gömmeye başlanmıştır. Bu konuda çeşitli algoritmalar geliştirilmiştir [3]. Bu çalışmada resmin içine veri gizlenmiş ve daha sonra bu resim iletilmiştir. Resmi alan kullanıcı resmin içine gömüldüğü gibi veriyi çıkartmış ve böylece haberleşme sağlanmıştır. Bu yöntemde veriyi gömme işlemi 4.1 algoritmasında belirtildiği gibidir. Bu algoritma

for $i = 1,...,l(c)$ do (4.1)

$s_i \leftarrow c_i$

$s_{ji} \leftarrow c_{ji}$ □ m_i

end for

$l(c)$ = esas resmi oluşturan baytların uzunluğu

s_i = steganografik anahtarın bitlerinin gösterimi

c_i = esas resmin bitlerinin gösterimi

m_i = esas resme gizlenecek verinin bitlerinin gösterimi

ji = esas resmin hangi bitlerine gömülüm yapılacağının gösterimi

Bu yöntemde önceki yöntem gibi resmin wnbrowse editörü gibi bir araç ile hex formatında gösterimi sağlanır ve bir metin dosyasına kaydedilir. Elde edilen bu esas resim *cover* olarak ifade edilir. İkilik düzende düşündüğümüzde her bir bayt 8 bitten oluştuğundan esas resmin bitleri *ci* olarak ifade edilmektedir. *i* indisi kaçıncı bit olduğunu temsil etmektedir. Steganografik tanım olarak steganografik anahtar ifadesi *si* olarak belirtilmiştir. Resmin içine gömülmek istenen verinin (message) bitlerini ifade etmek için *mi* kullanılmıştır. Bu algoritmaya göre veri gizlenirken resmi oluşturan ilk baytın son bitinde değişiklik yapılarak veri gömülmeye başlanır. Yani veriyi gömme işlemi ilk baytın en düşük anlamlı bitinden başlanarak sırayla hangi baytın en düşük anlamlı bitine kadar gidiyorsa yapılır. Bu çalışmada, veri gizlenirken alfabedeki harflerin ikilik düzende karşılıkları şeklinde bir dönüşüm kullanılması gerekmektedir. Bunu sağlamak için ascii karakter kodlamasından yararlanılmıştır [2]. Örneğin, A harfinin ascii karakter kodlamasında karşılığı hex olarak 41'dir ve 8 bit olarak 0100 0001 olarak ifade edilmektedir. O halde veri gizlenirken A harfi yerine 0100 0001 şekli gizlenmelidir. Gizlemek istediğimiz bilginin ascii karakter kodlamasında karşılıklarını görebilmek adına şekil 4.5'teki ascii karakter tablosundan yararlanabilmektedir. Karşı taraf veriyi elde ettiğinde ikilik düzenin karşılığını alfabede bulunan 29 harfin karşılığı olarak elde etmesi içinse Matlab programında yazılan kod parçası kullanılmıştır. Örnek bir uygulama olarak olarak *itu* sözcüğünü gizlemeye çalıştığımızda ilk olarak yapılan işlem *itu* kelimesinin ascii karakter kodlamasında karşılığını bulmak olacaktır.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Şekil 4.5 Ascii Karakter Tablosu [6]

Şekil 4.5'teki tablodan *itu* kelimesinin ascii karakter kodlamasındaki karşılığı bulunur. Buna göre i = 69, t = 74, u = 75 şeklinde belirlenir. Burada 69,74 ve 75 şeklinde gördüğümüz sayılar hex formatında yapılardır. Yani 16'lık düzendedir. Veriyi resmin son bitlerine gizlemek için ikilik düzendeki ifadesine ihtiyacımız var. Bu nedenle *itu* sözcüğünün ikilik düzendeki karşılıkları i = 0110 1001, t = 0111 0100, u = 0111 0101 şeklinde belirlenir. Bundan sonraki işlemi bu bitleri veriyi gizleyeceğimiz resme ilk baytın son bitlerinden başlayarak sırayla gömmektir. Alfabede bulunan üç harfi gizlemek için her bir harfin 8 bitlik karşılığı olduğundan 24 bite ihtiyaç vardır. Bu da resmimizin ilk baytından başlanarak sırayla 24 baytın son bitlerine veriyi gizlemek anlamına gelmektedir. Kaynak resmimizin ilk 10 x 10'luk piksel gösterimini tekrar düşünerek durumu örnekleyerek açıklamasını yapacağım. F15 uçak resminin ilk 10 x 10'luk kısmındaki ilk bayt, 42 idi. Yani 0100 0010 şeklinde ikilik düzende ifadesi vardır. Gizlemek istediğimiz "itu" sözcüğünün ise ilk karakteri i = 0110 1001 şeklinde ifade edilmişti. Demek ki ilk baytın son bitine yani 42 in son bitine *i* karakterinin ilk biti gömülmelidir. 42 = 0100 00*1*0 olduğundan son bit 0'dır. i = *0*110 1001 olduğundan ilk bit 0'dır. Demek ki kaynak resmin ilk baytında bir değişiklik olmayacaktır. Yine 0 olarak kalacaktır. Bu şekilde

bu durum sırayla devam edecektir. Daha sonraki aşama *i* karakterinin ikinci biti olan 1'i ana resmin ikinci baytının son bitine gömmek olacaktır. Gömmek istenilen 24 bit resmin ilk baytından başlanarak 24 baytın en düşük anlamlı bitlerine yerleştirilecektir.

Burada algoritmada da belirtildiği gibi resmin bitlerini kontrol eden bir yapı yoktur. Yani *i* karakterinin ilk biti 0, aynı zamanda resmin ilk baytının da son biti 0 diyen bir kontrol mekanizmasına gerek yoktur. Saklamak istediğimiz bilgi ne ise onu oluşturan bitleri teker teker piksellerin son bitlerine yazdırmak yeterlidir. [2]

Veriyi gömme işlemi bittikten sonra tekrar Matlab programı kullanarak bu matrisi bitmap formatında resme dönüştürürüz. Yeni oluşan resimle eskisi kıyaslandığında gözle görülür bir fark olmamaktadır.

Bu içinde veri gömülü resim iletilmek istenen kişiye iletildiğinde o kişinin bu bilgiye ulaşması için tekrar o bilgiyi elde etmesi gerekir. Bunun için kullanılan algoritma

for i=1,...,$l(M)$ (4.2)

$m_i \leftarrow LSB(c_{ij})$

end for

$l(M) = l(m_i)$ = verinin uzunluğunun ifadesi

$LSB(c_{ij})$ = esas resmin son bitleri değişmiş olan baytlarının gösterimi

m_i = esas resme gizlenmiş olan verinin bitlerinin gösterimi

İçine veri gizlenmiş olan resim iletilmek istenen kişinin eline geçtiğinde veriyi gönderenden gerekli bilgileri önceden aldığından, veriye ulaşmak için 4.2 algoritmasını kullanması yeterlidir. Bu gerekli bilgi verinin alfabetik olarak ya da bit sayısı olarak uzunluğunu içermektedir.

Veri uzunluğunu bilen kullanıcı, verinin içerdiği bit sayısı kadar matrisi oluşturan baytlardan baştan başlayarak alır. Aldığı bu baytların son bitlerini bir metin dosyasına çıkartır ve ascii karakter kodlamasına göre her baytın karşılığını alfabedeki

harfler olarak elde etmelidir [2]. Bunun için de Matlab programında her harfin ascii karakter kodlamasını içeren bir kod parçası kullanarak bunun elde edilmesi sağlanır.

Günümüzde steganografinin çok fazla gelişme imkânı bulmasıyla buna karşı geliştirilen atak türlerinde de gelişme olmuştur. Bu nedenle böyle bir veri örneğin internet ortamında dolaşırken çok da fazla güvenli olmaz. Çünkü resmin içinde veri var mı diye bakan analiz yapmak isteyen kişi ilk baytın son bitlerinden başlayarak bir çözümleme yapmaya çalışabilir. Bu da verinin güvenliğini önemli ölçüde tehdit eden bir yapı oluşturur. Bu nedenle steganografinin resim alanındaki birçok çalışmasında bu yöntem kullanılsa da saldırı ve analiz tekniklerinin de geliştirilmesiyle veriyi gömme - çıkarma alanında yeni algoritmalar geliştirilmiştir [2,3].

Şekil 4.6 Veri gömülmeden önce

Şekil 4.7 Veri gömüldükten sonra

Görüldüğü gibi *itu* sözcüğü gömülmeden önce şekil 4.6'daki resimle,gömüldükten sonraki halini yani şekil 4.7'deki resmi yan yana koyduğumuzda gözle görülemeyecek kadar değişim oluştuğundan fark edememekteyiz. Buradan bilginin en düşük anlamlı bitlere gömülmesinin steganografi anlamında çok yararlı bir teknik olduğunu yorum olarak belirtebiliriz.

4.1.3 Rastgele Aralık Yöntemi

Bir önceki yöntemin zayıf yani iletilmek istenilen verinin başkaları tarafından ele geçirilme durumunun olmasıydı. Bu ihtimali azaltmak için geliştirilen algoritmalardan birisi "Rastgele Aralık Yöntemi" dir. Bu yöntemin bir önceki yöntemden farkı gizlenecek verinin bitlerinin ilk bayttan başlanarak sırayla en düşük anlamlı bitlere yüklenmeyecek olmasındandır. Rastgele Aralık Yöntemi'nde hangi piksellere verinin gizleyeceği belirli denklemlere bağlıdır [2].

Veriyi gömerken kullanılan algoritma

for i=1,...,$l(c)$ do (4.3)

$s_i \leftarrow c_i$

end for

rastgele k_i değerleri üretilir.

$n \leftarrow k_1$

for i=1,...,$l(m)$ do

$s_n \leftarrow c_n$ □ m_i

$n \leftarrow n+k_i$

end for

$l(c)$ = esas resmi oluşturan baytların uzunluğu

$l(m)$ = gizlenecek veriyi baytların uzunluğu

s_i = steganografik anahtarın bitlerinin gösterimi

c_i = esas resmin bitlerinin gösterimi

m_i = esas resme gizlenecek verinin bitlerinin gösterimi

k = gizleyeceğim veri uzunluğunda belirlenen anahtarın gösterimi

Bu yöntemin en önemli özelliği veri gizlenirken bir steganografik anahtarın kullanılmasıdır. Steganografik anahtar verinin rastgele olarak hangi piksellerin en düşük anlamlı bitlerine gizleneceğini belirtir. Burada önemli olan hem veriyi gönderen hem veriyi alan kişi için steganografik anahtardır. İki taraf da aynı steganografik anahtar olmadan veri iletimini sağlayamazlar. [2]

Çalışma sırasında steganografik anahtar belirleme de çeşitli yöntemler araştırılmıştır. Bu yöntem dâhilinde Matlab programı ile rastgele sayılar üretilmiştir. Bu sayıların steganografik anahtar olmasına karar verilmiştir. Matlab programı kullanılarak bu anahtarı üretilme sebebi bunun gerçekten diğer algoritmalardan daha hızlı ve kolay bir yol olmasıdır.

Rastgele aralık yöntemine dayalı olarak steganografik anahtarın uzunluğu gizlenmek istenen verinin uzunluğu ile aynı olmalıdır. Bu yöntemde veriyi gömme işlemine bir önceki yöntemdeki gibi başlanır [2].

Aynı örnek üzerinden gidecek olursak *itu* sözcüğünü gizlenecek bilgi olarak seçersek, ilk olarak bu kelimenin ascii karakter kodlamasında karşılığı bulunur. Daha önce de belirtildiği gibi *itu* sözcüğünün ascii olarak karşılığı ikilik düzende i = 0110 1001, t = 0111 0100, u = 0111 0101 şeklindedir. Burada bu bitler yine resmin piksellerinin en son bitlerine gömülecektir.

Yalnız bu sefer ilk bayttan başlanarak sırayla gömme işlemi yapmak yerine hangi baytlara verinin gizleneceğini gösteren bir steganografik anahtarın kullanıldığı denklem yer almaktadır [2]. Burada saklamak istediğimiz bilgi toplam 24 bitten oluşmaktadır. Bu da oluşturmamız gereken anahtarın 24 bit uzunluğunda olması gerektiğini belirtir. Bu steganografik anahtar belirlendikten sonra yönteme dayalı olarak veriyi resmin hangi baytlarına saklanıldığını belirtildiği denklem 4.4 kullanılır.

$$j_1=k_1 \tag{4.4}$$

$$j_i=j_{i-1}+k_i, i \geq 2$$

Bu denklemde j_i rastgele belirlenmiş steganografik anahtarı buna bağlı olarak 4.4 denklemini kullanarak resmin hangi baytlarına verinin gizleneceğini belirten bir indistir [2].

Örneğin Matlab programını kullanarak *itu* sözcüğün *i* harfini gizlemek için 8 tane rastgele sayı yani 4.4. denklemindeki k değerlerini belirleyelim. Bu sayılar 1, 3, 12, 56, 9, 10, 18, 22 şeklinde belirlendi. Buradan anlaşıldığı gibi k değerleri sırasıyla rastgele belirlenen 8 değerden oluşmaktadır. Belirlenen bu k değerlerinden yararlanarak veriyi gizleyeceğimiz j değerleri 4.4 denklemine göre belirlenir.

$k_1=1 \rightarrow j_1=1$

$k_2=3 \rightarrow j_2=j_1+k_2=4$

$k_3=12 \rightarrow j_3=j_2+k_3=16$

$k_4=56 \rightarrow j_4=j_3+k_4=72$

$k_5=9 \rightarrow j_5=j_4+k_5=81$

$k_6=10 \rightarrow j_6=j_5+k_6=91$

$k_7=18 \rightarrow j_7=j_6+k_7=109$

$k_8=22 \rightarrow j_8=j_7+k_8=131$

Belirlenen k dolayısıyla j değerleri resmin bayt numaralarını göstermiş oldu. Buradan çıkarılması gereken sonuç *i* harfini gizleme işleminin, resmin 1, 4, 16, 72, 81, 91, 109 ve 131 numaralı baytlarında gerçekleşmesi gerektiğidir. Numarası belirtilmiş baytların en düşük anlamlı bitlerine sırasıyla 0110 1001 bitleri gömülür. Aynı işlemler diğer karakterler için de steganografik anahtar ve verilen denklem yardımıyla hangi baytlara gömme işleminin yapılacağı belirlenerek uygulanır. Gömme işlemi rastgele aralık yöntemine göre gerçeklenmiştir. İçine veri gömülen resim iletilmek istenen yere ulaştığında bu veriyi tekrar elde etmek için de bu yöntem dâhilinde bir algoritma vardır.

Burada unutulmamalıdır ki hem veriyi gönderen hem veriyi alan kişilerde aynı steganografik anahtar bulunmaktadır [2].

Yani örnek olarak *i* karakterini gömmek için belirlenen 8 tane *k* değeri her iki tarafta da vardır. Steganografik anahtarın her iki tarafta da olduğu kabul edilen bu yöntem gereği veriyi resimden çıkartma algoritması

$$n \leftarrow k_1 \qquad (4.5)$$

for $i=1,...,l(m)$ do

$m_i \leftarrow LSB(c_n)$

$n \leftarrow n+k_i$

end for

k_1 = steganografik anahtarın ilk sayısının gösterimi

$l(m)$ = gizlenmiş olan verinin uzunluğu

$LSB(c_n)$ = resimde veri gizlenen en düşük anlamlı bitlerin gösterimi

4.5 denkleminin belirttiği algoritmada ifade edildiği gibi gömülmüş veriyi tekrar elde ederken öncelikle hangi baytlara verinin gizlenmiş olduğunun bulunması gerekir. Bu

nedenle elinde steganografik anahtar değerleri olan kişi denklem 4.4 gereği j_i değerlerini yani hangi baytlara veri gizlendiğini belirler.

Şekil 4.7 Veri gizlenmeden önce

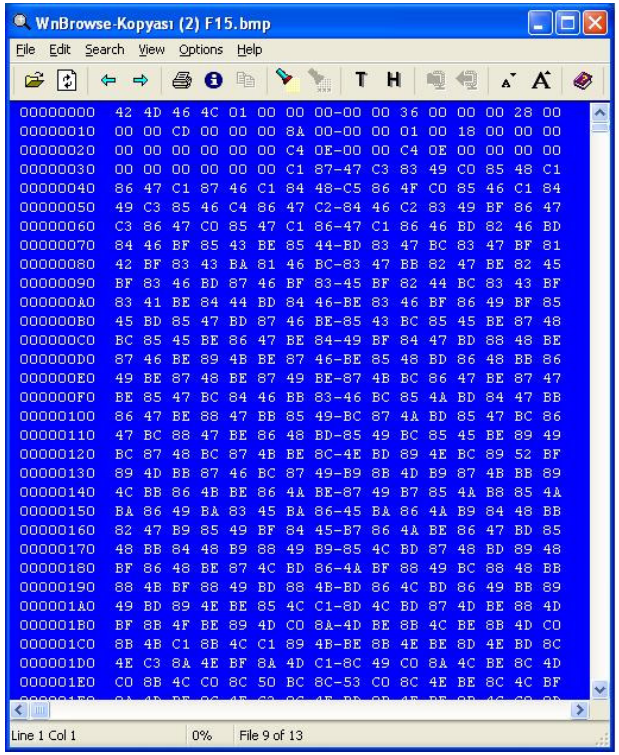
Şekil 4.8 Veri gizlendikten sonra

Daha sonra belirlenen baytların en düşük anlamlı bitleri yan yana getirilerek gizlenmiş olan veri elde edilmiş olur [2]. Burada yapılması gereken son işlem elde ettiğimiz veriyi alfabe düzenine geçirmektir. Bunun için de bir önceki yöntem de olduğu gibi Matlab programı yardımıyla gerekli dönüşüm yapılır ve alfabetik karşılıklar elde edilir. Şekil 4.7 ve Şekil 4.8'de görüldüğü gibi algoritma gerçeklendikten sonra resmin içine veri gömüldüğünde gözle görülür bir fark olmamıştır.

5. ÇALIŞMA KAPSAMINDA KULLANILAN ARAÇLAR

5.1 Wnbrowse Aracı

Wnbrowse programı, çok fonksiyonlu bir yapıya sahip olup ascii, ebcdic ve hex dosya görüntüleyici olarak bilinmektedir. Windows'un her sürümüyle birlikte oldukça uyumlu çalışan bir yapıya sahiptir [7]. Bu çalışmada resimleri hex formatında göstermede yardımcı araç olarak kullanılmıştır. Şekil 5.1' deki gibi bir ara yüze sahiptir. Çalışma sırasında bu ara yüzden metin dosyasına piksel değerleri kopyalanmıştır.



Şekil 5.1 Wnbrowse ara yüzü [8]

5.2 Matlab

Açılımı matris laboratuarı olan Matlab özellikle görsel uygulamalarda kullanılan oldukça geliştirilmiş bir yüksek seviyeli program dilidir [7]. Çalışmam sırasında hex formatından resme dönüşüm için yazılmış olan kod parçasını kullandım. Ayrıca steganografik anahtar üretmede yürütülen tek bir komutla istenilen anahtar değerlerine ulaştığımdan son yöntemi kolaylaştırıcı bir etken oldu. Gerçeklemiş olduğum iki yöntemde de ascii karakter karşılığının bulmak için kullandım.

5.3 Xilinx ISE ve FPGA

FPGA yani alanda programlanabilir kapı dizileri konfigüre edilebilir lojik bloklardan oluşan çok geniş ölçekli sayısal tümleşik devrelerdir [9]. Bu bloklar, tasarımcının yaptığı tasarıma göre işlevini kendi düzenleyebilmektedir. FPGA'lar mikroişlemcilerin adım adım çalışma mantığından farklı olarak işlemleri blok blok gerçekleştiğinden pek çok alanda her geçen gün artan oranda kullanılmaktadır. FPGA'de işlemler parelel olarak işlendiği için aynı anda yazılan programa göre çok fazla modül kullanabilme olanağı sunmaktadır. Ayrıca içinde birçok lojik kapılar ve flip floplar gibi sayısal tasarım için elemanlar bulunduran FPGA, VHDL veya Verilog gibi donanım dilleriyle bu elemanların bağlantılarının tasarıma göre yapılmasına olanak verir. Böylece tek bir FPGA ile istenilen sayısal devre meydana getirilebilir. Ürünün markete ulaşma süresini uygulamaya özel tümleşik devre (ASIC) tasarıma göre büyük ölçüde kısalttığından ve ayrıca kullanımın artmasıyla birlikte maliyetinin de düşmesinden ötürü FPGA'lar özellikle sayısal işaret işleme uygulamalarında, güvenliğin ön plana çıktığı savunma alanında ve daha pek çok alanda kullanılmaktadır. Markette FPGA üreticisi firma sayısı çok olmamakla birlikte her geçen gün sayıları da artmaktadır [7]. Bu çalışmada kullanılan FPGA'nın üretici firması da Xilinx'tir. Çalışmanın tasarım geliştirme sürecinde kullanılan benzetim ve sentez araçları, ayrıca FPGA'da gerçekleme adımında gerekli araçlar ve FPGA'ya yükleme işlemlerinin yapılması için gerekli araçlar Xilinx firmasının sağladığı bilgisayar destekli tasarım (CAD) ortamı Xilinx ISE'de bulunmaktadır [8].

FPGA tasarımının ilk aşaması gerçeklenecek olan algoritmanın davranışsal tanımlamasıdır. Bu istenen işlemlerin VHDL dilinde kodlanması ile gerçekleştirilir. İkinci adımda yazılan kodun istenen şekilde çalıştığını teyit etmek için yapılan davranışsal benzetimdir. Bu amaçla tasarlanan devre alt blok olarak bir test ortamına dâhil edilir. Devreye dışarıdan verilmesi beklenen işaretler veya veriler test ortamında sağlanır. Algoritmaların gerçeklenmesinin bu aşamasında işlenecek olan fotoğrafların piksel bilgileri metin dosyasına yazılmış ve bu dosyalar test ortamına dışarıdan giriş olarak verilmiştir. Test ortamının çıkışı da yine başka bir metin dosyasına yazdırılmıştır. Bu aşamada beklendiği gibi çalıştığı görülen kod sentezlettirilmiştir. Sentezleme sonrasında gerçeklenecek olan devrenin RTL şematiği oluşur. Ayrıca sentez raporunda devrenin kullanılacak olan FPGA üzerinde

kaplayacağı alan ve çalışabileceği en yüksek saat frekansı sonuçları bulunur. Sentez işleminden sonra kütüphanelerden devredeki parçalara karşılık düşen hücrelerin tespiti ve bunların yerlerine yerleştirilmesi sonucu oluşan konfigürasyon dosyasının oluşturulması adımı gelir. Bu adımda yapılacak olan benzetim sentez sonrası benzetim olarak geçer ve devredeki kapı gecikmelerinin de hesaba katıldığı benzetimdir. Bu benzetimde istenen saat frekansında devrenin ne şekilde çalışacağı görülür. Bu adımdan sonra oluşturulan konfigürasyon dosyası FPGA'ya yüklenir.

6. SONUÇ

Bu çalışmada incelenen ve uygulama yapılan üç yöntemle hem steganografinin çalışma prensibi hem de veri iletiminin nasıl olduğu anlaşıldı. Aşama aşama yöntemlerle öğrenilenler ve uygulamayla ilgili yorumlama yapmak çalışmanın getirdikleri açısından önemlidir.

İlk yöntemle resimlerin aslında pikseller yani ikilik düzende bitlerden oluştuğu öğrenildi. Bu yöntem dâhilinde resmi oluşturan 8 bitlik grupların yani baytların en düşük anlamlı biti değiştirildiğinde ve yeni yapı tekrar resim haline getirildiğinde yeni oluşan resimde gözle görülür bir fark olmadığı görülmüştür. En düşük anlamlı bitlerin tümünü değiştirilmesi yöntemi ile steganografinin ana mantığı anlaşılmış oldu. Bir veriyi en düşük anlamlı bitlere gömme ve elde etme yöntemi ile, ilk yöntemde elde edilen bilginin sağladıklarıyla beraber bu sefer resmi oluşturan piksellerin en anlamsız bitine bir veri gömülüp tekrar elde edilme işlemi yapıldı. Yani bu sefer bir yerden bir yere verinin resim aracılığıyla kimse fark etmeden taşınabileceği ve verinin tekrar gömüldüğü gibi çıkarılabileceği görüldü ve anlaşılmış oldu.

Yapılan araştırmalar sonucunda steganografi teknikleri ve bu alanda kullanılan algoritmalar geliştikçe stegoanaliz tekniklerinin de zamanla geliştiği görülmüştür. Bu nedenle bu algoritmanın aslında zayıf bir yanın olduğu belirtilmelidir. Bu zayıf yan veri gömülme işlemi sırasında ilk baytın en düşük anlamlı bitinden başlanarak sırayla verinin gömülme işleminin uygulanmasındandır. Veri başkaları tarafından gelişen stegoanaliz teknikleriyle ele geçebilir. Bu nedenle steganografi alanında kullanılan başka yöntemler de araştırılmış ve rastgele aralık yöntemi öğrenilmiştir. Bu yöntem bir önceki yöntemden daha güvenlidir. Bunun nedeni veri gizlerken güvenliği sağlamak adına steganografik bir anahtar kullanılmasıdır. Kullanılan bu anahtar sayesinde algoritmaya bağlı olarak verinin hangi baytların en anlamsız bitine gizleneceği belirlenir. Bu sayede elinde steganografik anahtar olmayan birinin veriyi ele geçirmesi çok güçtür.

Steganografi konusunda öğrenilenlerin yanında vhdl diliyle programlama konusunda tecrübe edinilmiş olundu. Simulasyonlarla gerçekleme sırasında çıkabilecek sorunların çözümlenmesi ve fpga çalışma mantığı konuları da gerek uygulama sırasında gerek yapılan araştırmalarla öğrenildi. Özetle gerçeklenen konu ne olursa olsun sayısal bir tasarım sırasında çıkabilecek sorunlar, tasarım süreci ve planlaması bu örnek çalışmayla öğrenilmiş oldu.

KAYNAKLAR

[1] **Johnson, N. F., Duric Z. ve Jajodia S.,** 2001. Information Hiding : Steganography and Watermarking - Attacks and Countermeasures, Boston.

[2] **Katzenbeisser, S. ve Petitcolas, Fabien A. P.,** 2000. Information Hiding Techniques for Steganography and Digital Watermarking, London.

[3] **Marvel, L. M.,** *Information Hiding: Steganography and Watermarking,* 2005, Optical and Digital Techniques for Information Security

[4] **Petitcolas, Fabien A. P.,** 2008. The Image Downgrading Problem, http://petitcolas.net/fabien/steganography/image_downgrading/index.html, [Ziyaret Tarihi: 27.03.2008]

[5] *Ascii Tablosu,*

http://www.antrak.org.tr/gazete/062006/cizimler/tolgatastan/ascii_table.jpg

[Ziyaret Tarihi: 28.04.2008]

[6] *Gray Palette,* http://cat.xula.edu/tutorials/imaging/modes/pallettecropped.gif,

[Ziyaret Tarihi: 24.04.2008].

[7] http://en.wikipedia.org, [Ziyaret Tarihi:12.04.2008]

[8] *Wnbrowse*, http://www.ngthomas.co.uk/wnbrowse.html, [Ziyaret Tarihi:15.12.2007].

[9] *Xilinx,* http://www.xilinx.com/company/gettingstarted/index.htm, [Ziyaret Tarihi:01.05.2008].

ÖZGEÇMİŞ

Betül Elci 1985 yılında İzmit'te doğdu. Gölcük Anadolu Lisesi'nde okudu, Konya Meram Anadolu Lisesi'nde orta öğrenimini tamamladı. Kocaeli Körfez Fen Lisesi'nde lise öğrenimini gördü. 2004 yılından beri İstanbul Teknik Üniversitesi, Elektronik Mühendisliği bölümünde okumaktadır.