İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

LEON3 MİKROİŞLEMCİSİ TABANLI SİSTEM GERÇEKLEMESİ

YÜKSEK LİSANS TEZİ

Ahmet Çağrı BAĞBABA

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

MAYIS 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

LEON3 MİKROİŞLEMCİSİ TABANLI SİSTEM GERÇEKLEMESİ

YÜKSEK LİSANS TEZİ

Ahmet Çağrı BAĞBABA
(504131201)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Berna Örs YALÇIN

MAYIS 2015

İTÜ, Fen Bilimleri Enstitüsü'nün 504131201 numaralı Yüksek Lisans Öğrencisi **Ahmet Çağrı BAĞBABA**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"LEON3 MİKROİŞLEMCİSİ TABANLI SİSTEM GERÇEKLEMESİ"** başlıklı tezini aşağıdaki imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Doç. Dr. Berna Örs YALÇIN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Bilge Günsel KALYONCU** İstanbul Teknik Üniversitesi	.....
	Doç. Dr. Berna Örs YALÇIN İstanbul Teknik Üniversitesi	.....
	Yrd. Doç. Dr. Selçuk BAKTIR Bahçeşehir Üniversitesi	.....

Teslim Tarihi : **4 Mayıs 2015**
Savunma Tarihi : **27 Mayıs 2015**

Aileme,

ÖNSÖZ

Mezunu ve mensubu olmaktan her zaman gurur duyduğum İstanbul Teknik Üniversitesi'nde yürüttüğüm tez çalışmam boyunca derin bilgi ve tecrübesiyle çalışmama yön veren değerli hocam Doç. Dr. Berna Örs YALÇIN'a, tüm bilgilerini ve deneyimlerini her an benimle paylaşarak tecrübe kazanmamda büyük pay sahibi olan Anka Mikroelektronik Sistemler'in kurucuları İnan ERDEM'e ve Gökhan IŞIK'a, son olarak tüm hayatım boyunca manevi desteğini hiç bir zaman esirgemeyen ve yanımda olduklarını her daim hissettiğim aileme sonsuz teşekkürlerimi sunarım.

Mayıs 2015

Ahmet Çağrı BAĞBABA
Elektronik Mühendisi

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **vii**
İÇİNDEKİLER ..... **ix**
KISALTMALAR ..... **xi**
ÇİZELGE LİSTESİ ..... **xiii**
ŞEKİL LİSTESİ ..... **xv**
ÖZET ..... **xvii**
SUMMARY ..... **xix**
1. GİRİŞ **1**
1.1 Literatür Araştırması 2
2. LEON3 MİKROİŞLEMCİSİ **5**
2.1 Gaisler Kütüphanesi 6
2.1.1 Kütüphane Organizasyonu 6
2.1.2 Kırmık Üzeri Veri Yolu Yapısı 7
2.2 Gaisler Araçları 7
2.2.1 Gaisler monitor 8
2.2.2 Bare-C çapraz derleyicisi 9
3. KÜÇÜK ŞİFRELEME ALGORİTMASI **11**
4. JPEG STANDARDI **15**
4.1 Ayrık Kosinüs Dönüşümü 16
4.2 Kuantalama 19
4.3 Zig-Zag Tarama 20
4.4 DC Katsayılar İçin Farksal Kodlama 21
4.5 AC Dizi Uzunluğu (Run Length) Kodlama 22
4.6 Huffman Kodlama 22
4.7 JPEG Dosya Yapısı 24
5. DONANIM TASARIMI **27**
5.1 JPEG Kodlayıcının Tasarlanması 27
5.1.1 DCT bloğu 27
5.1.2 Kuantalama bloğu 30
5.1.3 Zig-zag tarama bloğu 31
5.1.4 DC farksal kodlama bloğu 32
5.1.5 AC dizi uzunluğu kodlama bloğu 32
5.1.6 Kod paketleme bloğu 32
5.2 Şifreleme Bloğunun Tasarımı 33
5.3 Donanımın Leon3 Mikroişlemcisine Eklenmesi 34
6. GERÇEKLEME SONUÇLARI **37**

7. SONUÇ 39
KAYNAKLAR..... 41
ÖZGEÇMİŞ 43

KISALTMALAR

MİB : Merkezi İşlem Birimi
VHDL : Very High Speed Integrated Circuit Hardware Description Language
IP : Integrated Peripheral
SoC : System On Chip
CAD : Computer Aided Design
AMBA : Advanced Microcontroller Bus Architecture
APB : Advanced Peripheral Bus
AHB : Advanced High-speed Bus
BCC : Bare-C Cross Compiler
JPEG : Joint Photographic Experts Group
TEA : Tiny Encryption Algorithm
DCT : Discrete Cosine Transform
ITU : International Telecommunication Union

ÇİZELGE LİSTESİ

Sayfa

Çizelge 4.1: Örnek 8x8'lik Giriş Matrisi 18
Çizelge 4.2: Örnek DCT Katsayı Matrisi 18
Çizelge 4.3: Örnek Kuantalama Sonuç Matrisi 20
Çizelge 4.4: Kategori ve Bit Kodlama Tablosu. 23
Çizelge 4.5: AC Katsayılar için Huffman Tablosu. 23
Çizelge 4.6: DC Katsayılar için Huffman Tablosu. 24
Çizelge 5.1: Zig-zag tarama adresleri 32
Çizelge 6.1: Alan kullanım bilgisi. 37
Çizelge 6.2: Görüntülere ait değerler. 37

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1 : Leon3 Yapısı [10]. 5
Şekil 2.2 : Leon3 ve GRLIB [11]..... 7
Şekil 2.3 : GRMON [12]..... 8
Şekil 3.1 : TEA Şifreleme Rutini [15]. 12
Şekil 3.2 : TEA i. döngüsü [15]. 13
Şekil 4.1 : DCT Tabanlı JPEG. 16
Şekil 4.2 : 8x8 DCT 17
Şekil 4.3 : Zig-Zag Yol..... 21
Şekil 4.4 : 8x8'lik blokların DC farksal kodlaması. 21
Şekil 4.5 : Kuantalama tablosu belirleme [18]..... 24
Şekil 5.1 : Birinci bir boyutlu DCT. 29
Şekil 5.2 : İkinci bir boyutlu DCT..... 29
Şekil 5.3 : Ping-pong tampon bellek [20]. 30
Şekil 5.4 : İkili RAM yapısı [20]. 30
Şekil 5.5 : İkili RAM yapısı. 31
Şekil 5.6 : Kod paketleme bloğu. 33
Şekil 5.7 : GRLIB'e donanım eklenmesi. 34
Şekil 5.8 : Bayraklar için C kodu. 35
Şekil 5.9 : Sistemin tüm yapısı. 36
Şekil 5.10 : Tüm sistemin başka bir ifadesi. 36
Şekil 6.1 : Lena. 38
Şekil 6.2 : Melissa. 38

LEON3 MİKROİŞLEMCİSİ TABANLI SİSTEM GERÇEKLEMESİ

ÖZET

Bu tez kapsamında JPEG (Joint Photographic Experts Group) görüntü sıkıştırma standardı kullanılarak görüntü şifreleme yapılmıştır. Özellikle internet iletişimin yaygınlaşmasından sonra bilgisayarlar arasında görüntü aktarımı önem kazanmıştır. Buna paralel olarak depolama problemi ortaya çıkmaya başlamıştır. Bu soruna çözüm olarak ortaya çıkan JPEG, orjinal görüntünün boyutunda önemli miktarda azalma sağlamaktadır. Depolama probleminin yanı sıra güvenlik de aynı şekilde önem arz etmektedir. İletilen verilerin yada görüntülerin istenmeyen kişiler tarafından ele geçirilmesi önemli bir güvenlik problemi olarak görülmektedir. Bu amaçla hem sıkıştırmanın hem şifrelemenin bir arada yapılması oldukça mühim hale gelmiştir. Bu yüksek lisans tezi kapsamında da sıkıştırma oranınından fazla kayıp vermeden görüntünün şifrelenmesi amaçlanmıştır. Sıkıştırma standardı olarak JPEG seçilmiştir. Şifreleme yöntemi olarak Küçük Şifreleme Algoritması (Tiny Encryption Algorithm-TEA) seçilmiştir. Şifreleme işlemi sıkıştırma işlemiyle iç içe yapılmıştır. JPEG formatına dönüştürülmüş bir görüntüyü şifrelemek yerine sıkıştırma bloklarıyla birlikte piksel seviyesinde şifreleme yapılarak güvenlik seviyesinin arttırılması hedeflenmiştir.

JPEG kodlayıcıda tasarlanması gereken bloklar Ayrık Kosinüs Dönüşümü (Discrete Cosine Transform-DCT), kuantalama, zig-zag tarama, DC farksal kodlama, AC dizi uzunluğu kodlama ve kod paketleme bloklarıdır. Şifreleme bloğu olan TEA, zig-zag tarama bloğundan sonra gelmektedir. İlk olarak giriş görüntüsü DCT bloğuna gelmektedir. Bu blok giriş görüntüsünü 8x8'lik matrisler halinde alarak işlem yapar ve çıkışında da 8x8'lik matrisler üretir. Bu matrislerin ilk elemanına DC katsayı, diğer 63 elemanına AC katsayı adı verilmektedir ve çıkış görüntüsü artık frekans tanım bölgesine geçirilmiş olur. Daha sonra kuantalama bloğuyla, matrislerdeki yüksek frekans terimlerinin çoğunlukla sıfırlanması sağlanır. Bu sıfırlanmalar sayesinde sıkıştırma yapılacaktır. Daha sonra gelen zig-zag tarama bloğunda kuantalanmış görüntü verisini içeren 8x8'lik matrisler seri hale getirilir. Verilerin sıralanması zig-zag düzeninde gerçekleşir. Bu şekilde sıralama yapıldığında DCT katsayılar, en düşük frekanslı katsayı olan DC katsayıdan yüksek frekanslı AC katsayılara doğru dizilmiş olur. Kuantalama işlemi de yüksek frekansları sıfırlamak üzere kurulu olduğundan zig-zag yolda ilerlendiğinde sıfırlar ardı ardına gelmektedir. Bir sonraki adımda TEA şifreleme bloğu vardır. Şifreleme için öncelikle sadece tüm 8x8'lik matrislerin sadece DC katsayıları kullanılmıştır. Daha sonra hem DC katsayılar hem de her matrisin 5 adet AC katsayısı kullanılmıştır. Sıkıştırmanın fazla azalmaması amacıyla daha fazla sayıda AC katsayı şifrelenmemiştir. Şifreleme bloğundan sonra DC farksal kodlama bloğu ve AC dizi uzunluğu kodlaması yapılmıştır. Son aşamada ise kod paketleme bloğu kullanılarak sıkıştırma ve şifreleme işlemi bitirilmiştir.

JPEG görüntü şifreleme donanımı tasarlandıktan sonra bir mikroişlemciye çevresel olarak eklenmesi hedeflenmiştir. Bu amaçla Gaisler firmasının geliştirdiği 32 bitlik Leon3 mikroişlemcisi seçilmiştir. Leon3, Sparc V8 mimarisi tabanlı ve çoklu işlemi destekleyen bir mikroişlemcidir. Sentezlenebilir Very High Speed Integrated Circuit Hardware Description Language (VHDL) modelleriyle oluşturulmuştur ve 16 çekirdeğe kadar tasarım yapılabilir. Yüksek konfigürasyon, yüksek performans, enerji verimliliği, basit tasarım entegrasyonu ve yazılım desteği sebebiyle bu mikroişlemci seçilmiştir. İçerisindeki Advanced Microcontroller Bus Architecture (AMBA) veri yolu yapısı, Advanced High-Performance Bus (AHB) ve Advanced Peripheral Bus (APB) olmak üzere ikiye ayrılmaktadır. Bu çalışmada tasarlanan donanım APB veri yoluna eklenmiştir. Bu işlemi yapmak için AMBA protokolüne uygun bir arayüz tasarlanmıştır. Daha sonra gerekli tanımlama işlemleri yapılarak donanım ekleme işlemi tamamlanmıştır. Bunun yanında donanımın mikroişlemci çekirdeğiyle haberleşmesinde kullanılacak olan bayrak tanımlamalarını yönetmek için bir C kodu gerekmiştir. Bu C kodu, Leon3 mikroişlemcisinin çapraz derleyicisi ile derlenmiş ve kullanılmıştır.

Tasarlanan devrenin girişine verilen görüntü boyutu 288x288 olarak belirlenmiştir. Bu görüntü istendiği takdirde değiştirilebilir. Elde edilen şifrelenmiş ve sıkıştırılmış görüntü matrisleri MATLAB ortamında okutulduktan sonra elde edilen çıkış görüntüleri karşılaştırılmış ve Peak Signal to Noise Ratio (PSNR) değerleri verilmiştir. Bunun yanında donanımın Sahada Programlanabilir Kapı Dizisi (FPGA) üzerinde kapladığı yer bilgileri de verilmiştir.

LEON3 MICROPROCESSOR BASED SYSTEM DESIGN

SUMMARY

In this work, image encryption system was designed on a Field Programmable Gate Array (FPGA) by using JPEG (Joint Photographic Experts Group) image compression standard. Transmission of images comes into prominence after the development of internet technology. Also, the storage problem becomes evident while internet technology is developing. Minimization of the number of information carrying units is important for image data compression. Also, since the main target is reducing the memory and decreasing the bandwidth in communication, image compression techniques are very useful. In this area, JPEG is international standard due to the fact that it has high compression ratio and it does not cause of deformation of images. Nowadays, digital images and the security of videos come into prominence in areas such as television broadcast, video conferences, medical imaging.

Security of data transmission is crucial as well. It is necessary to provide security of data or image by sending them in a channel. In this work, Tiny Encryption Algorithm (TEA) was used for encryption. TEA is suitable for embedded systems owing to high performance, ease of implementation, high speed, low energy consumption, low cost, and being lightweight. The TEA design's main aim is to provide minimum memory space and have maximum speed. Moreover, it uses Feistel Encryption type. As a result of this, when plain text is changed 1 bit, this reflects to output, which name is chipper text, as 32 bit. 128 bit key is used for encryption. Modified TEA was used in this work by changing the number of input bits. In standard TEA, input is 64 bit and it is divided left and right side as 32 bit. It was modified as the input of circuit as 72 bit so our left and right sides are 36 bit. However, key length is still 128 bit.

The design of encryption and compression were implemented together in order to increase security level. The method is not like encryption of one image which is in JPEG format. Encryption and compression blocks were implemented as one hardware. Hence, if someone gets encrypted and compressed image, he/she can not obtain original image after decryption. Because the encryption was performed at the pixel level.

To accomplish the JPEG image encryption system, JPEG encoder was designed in the beginning. JPEG encoder design includes Discrete Cosine Transform (DCT), Quantization, Zig-Zag reordering, Diffence Coding for DC coefficients, Run Length Coding for AC coefficients, and Huffman Coding blocks. The main aim of the DCT is to transform the value of pixels from spatial domain to frequency domain. The level of detail in an image is related to the frequencies. High spatial frequencies correspond to high levels of detail, while lower frequencies corresponds to lower levels of detail. The result matrix of DCT consists of 64 DCT coefficients. The next step in the compression process is quantization of the DCT matrix. It is the process of reducing number of

bits needed to store an integer value by reducing the precision of the integer. The quantization process has the key role in the JPEG compression. The quantization cycle has readily apparent effects on an image. The low frequency elements which are close to DC coefficient have been modified. The high frequency elements have been reduced to zero. The human eye is much more sensitive to lower spatial frequencies than to higher frequencies so quantization is necessary for JPEG. This is achieved by dividing values at high amplitude frequencies in the matrix with larger values than the values by which are divided the amplitudes of lower frequencies. Due to rounding in quantization, it is the lossy step of JPEG standard. As a result, insignificant data in matrix is discarded. After doing the DCT transform and quantization over a block of 8x8 values, a new 8x8 block is obtained. This 8x8 block is traversed in zig-zag ordering. The reason for this zig-zag traversing is that we traverse the 8x8 DCT coefficients in the order of increasing the spatial frequencies. After the zig-zag reordering, first six elements of zig-zag vectors were encrypted by using TEA. These six elements have the lowest frequencies of each matrices. Therefore, they have the most effect on the image. First one of these six elements is the DC coefficient of 8x8 blocks. Other five coefficients are AC coefficients of 8x8 blocks. After the encryption, all pixel values are gathered again and Huffman Coding is started. In the Huffman Coding, there are Zero Run Length Encoding for AC coefficients, Difference Coding for the DC Coefficients and Encoding blocks. In the result of Huffman Coding Block, encrypted image bit stream is obtained.

Some special blocks were used in order to design JPEG encoder. For DCT design, it is necessary that all input pixels have to be ready at the same time. In order to provide these inputs, ping-pong buffer was used. In ping-pong buffer, there are 8 registers and when the input is available, it is stored at the first register. Previous value of register is shifted to the next register. Hence, the buffer is sampled for each 8 clock cycle. After the 8 inputs are ready in registers, enable is activated in order to send these inputs to outputs. This block is necessary since inputs of DCT block are serial but 8 pixel values have to be ready at the same time in order to calculate DCT values. During these processes, input registers takes new input pixels. Therefore, there is no need to wait for DCT calculations or new inputs. Whole results were stored by using dual RAM structure. 1-D DCT and 2-D DCT results were stored at this RAMs and they were used alternately. This structure helps the designer about providing continuity of calculations.

After the designing of image encryption and compression blocks on a FPGA, this hardware was added to the Leon3 microprocessor as a peripheral. Leon3 is a 32 bit SPARC processor which is implemented as synthesizable and open source VHDL model. It was primarily developed for space application by European Space Agency (ESA) and it is widely used in embedded systems and applications today. It is also the part of Gaisler Library (GRLIB) which includes IP cores for FPGA and ASIC designs. Full source code is available under the GNU GPL license. The GRLIB is designed in order to connect all IP cores to the bus which is Advanced Microcontroller Bus Architecture (AMBA) Advanced High Speed Bus (AHB)/Advanced Peripheral Bus (APB). In this work, hardware design was added to the APB of Leon3 microprocessor. Interface was designed in order to connect our IP to Leon3 core. Also, it is necessary to have a C program in order to manage flag operation of the hardware. C program was compiled by using cross compiler of Leon3.

288x288 pixels images were used in order to test the system. First, only DC coefficients of 8x8 blocks were encrypted and result images were given. Then, DC coefficients and first five AC coefficients of 8x8 blocks were encrypted and result images were given. These two methods were compared by calculation their Peak Signal to Noise Ratio (PSNR). UART of the Leon3 microprocesssor was used for transferring of image matrices.

1. GİRİŞ

Görüntü sıkıştırma, görüntüyü tanımlayan bilginin azaltılmasıyla ilgilenir [1]. Depolamada hafızayı azaltmak, iletişimde bant genişliğini azaltmak hedef olduğu için veri sıkıştırması bu alanlarda kullanışlı hale gelmektedir [1]. Bu alanda, yüksek sıkıştırma oranı ve görüntüde az bozunuma sebep olduğu için Joint Photographic Expert Group (JPEG) uluslararası standart olarak gözükmektedir. JPEG ağ iletişimi, kablosuz haberleşme, medikal, multimedya ve özellikle gömülü sistemlerde yaygın bir biçimde kullanılmaktadır. Günümüz teknolojisinde, ücretli televizyon yayınları, özel video konferanslar, medikal görüntüleme, endüstriyel ve askeri görüntüleme gibi uygulamalarda sayısal görüntü ve videoların güvenliği büyük bir önem kazanmıştır [2].

Günümüzün dijital dünyasında, internet ağı bilgiyi dönüştürmek ve iletmek için en yaygın kullanılan kanaldır. Bu bilgiler mesaj, görüntü, ses yada video olabilir. Fakat bu kanalın güvenli olmaması bilginin depolanması yada taşınması sırasında istenmeyen kişiler tarafından ele geçirilmesine sebep verebilir. Bu sebeple şifreleme yapmak, bilginin güvenliğini sağlayacak ve sadece istenen kişiler tarafından anlamlı hale getirilmesini sağlayacaktır [3]. Görüntü şifrelemedaki amaç ise görüntüyü ya tamamen anlaşılmaz hale getirmek yada sadece belli bir bölgesini başkaları tarafından anlaşılmaz hale getirmeyi amaçlamaktadır. Bu sebeple görüntü şifreleme, taşınan görüntünün güvenli hale getirilmesi için akla gelen en güvenli yöntem olma özelliğini taşımaktadır.

JPEG görüntü sıkıştırma standardından görüntü şifreleme yapmak ise hem depolama konusunda hem de güvenlik konusunda büyük avantajlar getirmektedir. Tüm JPEG görüntü şifreleme algoritmaları şifrelenecek olan görüntünün formatının bozulmaması, görüntü boyutunun artmaması ve şifreleme algoritmasının yüksek güvenlikli olmasını gözetmelidir [4]. Bu çalışmada da kullanılan yöntemde sıkıştırmanın etkisinin kaybolmaması gözetilerek güvenli bir sıkıştırma yapılması amaçlanmıştır. Şifreleme JPEG kodlama bloklarıyla birleştirilerek piksel seviyesinde yapılmıştır. Yani JPEG formatına dönüştürülmüş bir görüntüyü şifreleme bloğuna sokmak yerine, şifreleme

ve sıkıştırma birlikte gerçeklenerek güvenlik seviyesinin arttırılması amaçlanmıştır. Böylece şifreli görüntünün ele geçirilmesi ve şifrenin çözülmesi durumunda elde edilen görüntü karşı taraf için anlamlı olmayacaktır. Aynı zamanda görüntünün hangi piksellerine ve hangi bitlerine şifreleme yapıldığı da bilinemeyecektir. Bu çalışmanın bir başka amacı da tasarlanan JPEG görüntü şifreleme donanımının seçilen bir mikroişlemciye çevresel olarak eklenmesidir. Mikroişlemci olarak 32 bitlik bir mikroişlemci olan Leon3 seçilmiştir.

Şifreleme işlemi JPEG'e ait zig-zag sıralama işlemlerinden sonra yapılmıştır. Zig-zag sıralama daha sonra da anlatılacağı üzere matrisleri düşük frekanstan yüksek frekansa göre vektör şeklinde sıralar. Bu sayede daha çok bilgi taşıyan pikseller vektörün ilk elemanlarında yer alır. Şifreleme bu elemanlara uygulanmıştır.

Çalışmanın donanım kısmında öncelikle JPEG kodlayıcıyla birlikte şifreleme bloğu tasarlanmıştır ve bu iki donanım birleştirilmiştir. Tasarlanan JPEG görüntü şifreleme donanımı daha sonra 32 bitlik Leon3 mikroişlemcisine çevresel donanım olarak eklenmiştir. Tüm sistem Sahada Programlanabilir Kapı Dizisi (Field Programmable Gate Array-FPGA) üzerinde gerçeklenmiştir. Şifreleme için öncelikle ilgili görüntünün sadece DC katsayıları şifrelenmiş, daha sonra hem DC hem de 5 adet AC katsayılar şifrelenmiştir. İkisi arasındaki farklar ve performans kriterleri çalışma sonunda verilmiştir. Girişte kullanılan görüntülerin boyutu 288x288 olarak belirlenmiştir.

1.1 Literatür Araştırması

Literatürde JPEG görüntü sıkıştırma standardından görüntü şifreleme yapılan birçok çalışma vardır. Bu bölümde bu çalışmalar incelenecektir.

Zaman-Uzamsal kaos tabanlı bir şifreleme yöntemi kullanılan [5]'te DC katsayılar çıkarılmış, böylece Ayrık Kosinüs Dönüşümü (Discrete Cosine Transform-DCT) katsayılarının genel istatistiksel dağılımlarının değişmemesi sağlanmıştır. Şifreleme kısmında uygulanan kaos tabanlı yöntem, DCT bloklarını karıştırmak ve kuantize edilmiş DC katsayıları dağıtmak için kullanılmıştır.

JPEG görüntü şifreleme uygulamasının yapıldığı bir başka çalışma olan [6]'da, tanımlanmış bloklardaki katsayıların dağıtımı kullanılmıştır. Bu yöntem temelde DC

katsayıları ayrıştırır ve bu ayrıştırılmış DC katsayıları ve diğer AC katsayıları karıştırır. DC katsayıların ayrıştırılması işlemi bu katsayıları AC katsayılara benzer hale getirir. Bu sebeple DC katsayılara yapılacak olan ataklar önlenmiş olur. Daha sonra yapılacak olan karıştırma işlemi sayesinde ise şifrelenmiş görüntü kaotik bir hal alır ve anahtar bilgisi olmadan şifre çözme işlemi yapmak çok zorlaşır.

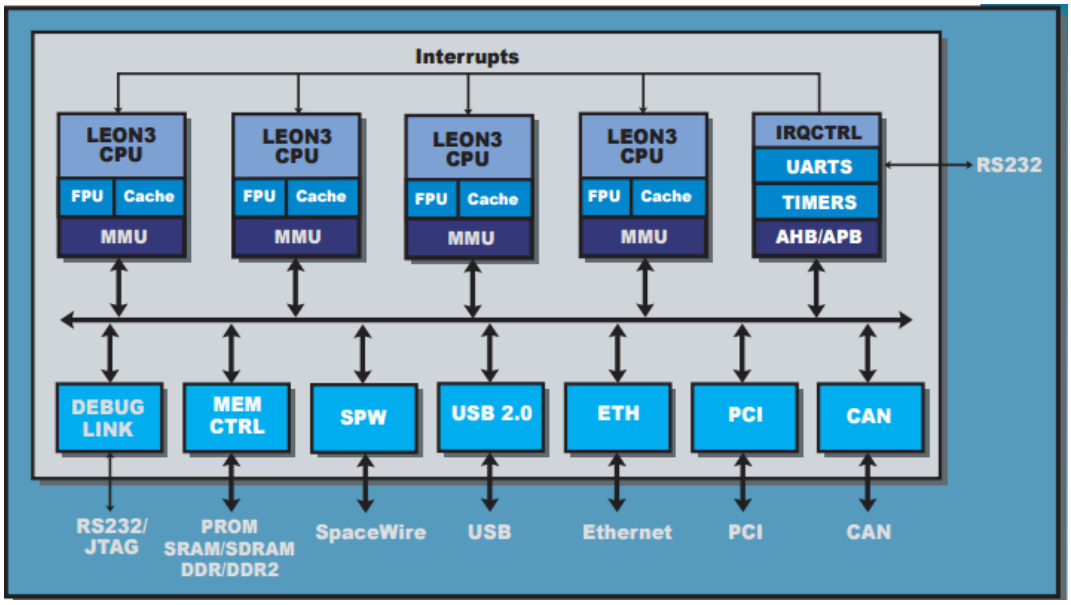
Yalnızca seçilen DCT katsayılarına şifreleme yapılan çalışmalar da vardır. [7]'de seçilen katsayılar yüksek frekanslıdır ve bu katsayılar rastgele sayılarla "özel veya" işlemine sokularak şifreleme yapılmıştır. Rastgele sayı üreteci olarak Lineer Olmayan Geri Kaydırıcı Yazmaç (Non-Linear Shift back Register) kullanılmıştır. Bu çalışmada güvenliğin biraz daha arttırılması amacıyla şifrelenmeyen DCT katsayıların da yeri değiştirilmiştir.

Bit Devirdaim Görüntü Şifreleme (Bit Recirculation Image Encryption) çalışmalardan birisi [8]'dir. Bu işlem yapılırken düşük hesaplama maliyeti, yüksek güvenlik ve bozulmama olmaması hedeflenmiştir. Piksellerin yerlerinin değiştirilmesi için kaotik sistemlerden faydalanılmıştır. Sonuçta seçilen yöntemin giriş görüntülerini kaotik bir hale soktuğu belirlenmiştir.

Sonsuz Seriler Yakınsama Methodunun (Infinite Series Convergence Method) kullanıldığı [9]'da piksel koordinatları kullanılarak şifreleme yapılmıştır. Bu koordinatlar kullanılan şifreleme algoritması ile karıştırılarak görüntü tek boyutlu bir matrise çevrilmiştir. Süreç boyunca üç anahtar kullanılmıştır.

2. LEON3 MİKROİŞLEMCİSİ

Leon3, Sparc V8 mimarisi tabanlı, çoklu işlemi destekleyen, açık kaynak kodlu, 32 bitlik mikroişlemcidir. Tamamen sentezlenebilir Very High Speed Integrated Circuit Hardware Description Language (VHDL) modelleriyle oluşturulmuş bir mikroişlemcidir ve 16 Merkezi İşlem Birimi (MİB) çekirdeğine kadar gerçekleme yapılabilir [10]. Leon3 mikroişlemcisi eğitim ve geliştirme amacı güden GNU GPL lisansı altında dağıtılmaktadır. En büyük avantajı karmaşık sistemleri basitleştirerek tasarımın markete ulaşma süresini ve maliyeti azaltmasıdır. Şekil 2.1'de Leon3 yapısının genel hali verilmiştir. Görüldüğü gibi RS232 gibi birçok haberleşme arayüzünü desteklemektedir. 7 iş hattından (pipeline) oluşmaktadır. Tam sayı birimi, genel amaçlı saklayıcı dosyası, önbellekler ve onları kontrol eden birimler ve kayan nokta birimi işlemci çekirdeği olarak görev yapabilmektedir. Harvard mimarisine göre tasarlanmıştır.



Şekil 2.1: Leon3 Yapısı [10].

Özellikleri şu şekilde incelenebilir:

- **Yüksek Konfigürasyon**: Gaisler Kütüphanesi'ne dahil olduğundan kolayca değiştirilebilmektedir ve kırmık üstü tasarımlar için oldukça uygundur. 1 ve 16

arasında işlemcili tasarım yapılabilir. Veri ve komut önbelleklerinin boyutları her bir MİB için 0k ve 2Mbytes arasında ayarlanabilir. Ayrıca IEEE-754 Kayan Nokta Birimi (Floating Point Unit) ile yüksek performanslara ulaşılabilir.

- **Yüksek Performans**: 0.13 mikron tasarımda 400 MHz saat frekansına ulaşılabilmektedir.
- **Enerji Verimliliği**: Güç kapama modu ve saat işareti kapılama özelliği ile güç tüketimi düşürülmüştür.
- **Basim Tasarım Entegrasyonu**: Gaisler Kütüphanesi'nin tak-oynat özelliği sayesinde geliştirme süresi düşürülmüş ve tasarım esnekliği arttırılmıştır.
- **Yazılım Desteği**: Üzerinde eCos, Linux, Real Time Executive for Multiprocessor Systems (RTEMS) gibi işletim sistemleri çalıştırılabilmektedir.

2.1 Gaisler Kütüphanesi

Gaisler Kütüphanesi (Gaisler Library-GRLIB) [11], yeniden kullanılabilir Integrated Peripheral (IP) çekirdeklerinden oluşan kırmık üstü sistemler (System on Chip-SoC) için tasarlanmış bir kütüphanedir. IP çekirdekleri ortak bir veri yolu etrafında dizilmiş şekildedir ve sentez-benzetim işlemleri için uyumlu yöntemler kullanırlar. Üreticiden bağımsız olan bu kütüphane, farklı Computer Aided Design (CAD) araçları ve farklı hedef teknolojileri desteklemektedir. Tak-oynat özelliği sayesinde IP çekirdekleri kolaylıkla değiştirilebilmekte ve birbirlerine bağlanabilmektedir.

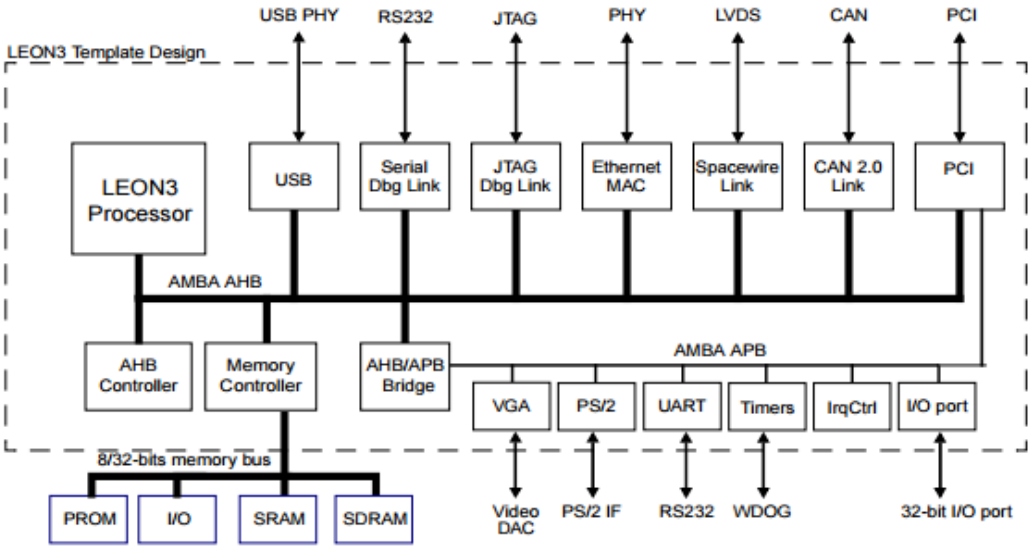
2.1.1 Kütüphane Organizasyonu

Gaisler Kütüphanesi VHDL ile oluşturulmuştur. Oluşturalan her IP'nin kendine özel kütüphane ismi vardır. Her bir VHDL kütüphanesi çeşitli paketler ve arayüzler içermektedir. Benzetim ve sentez komut yazıları (script) evrensel make dosyaları tarafından otomatik olarak oluşturulabilmektedir. Herhangi bir kütüphane veya paket ekleme-çıkarma işleminde daha önceden evrensel olarak tanımlanmış olan dosyaları silmeye gerek yoktur. Yani bir kütüphanenin yada IP'nin değiştirilmesi diğer IP'leri etkilememektedir. Aynı zamanda Modelsim, Ncsim, Aldec, Sonata gibi benzetim

araçları ve Cadence, Synopsys, Mentor, Altera, Xilinx gibi gerçekleme araçları için otomatik olarak komut yazıları oluşturulabilmektedir.

2.1.2 Kırmık Üzeri Veri Yolu Yapısı

Gaisler Kütüphanesi bütün IP'ler bir veri yolunun etrafında olacak şekilde tasarlanmıştır [11]. Bu yapıda veri yolu olarak marketteki etkinliği, iyi belgelendirilmiş olması ve herhangi bir lisans kısıtı olmadan kullanılabilmesi sebebiyle Advanced Microcontroller Bus Architecture (AMBA) 2.0 Advanced High Speed Bus (AHB)/Advanced Peripheral Bus (APB) kullanılmıştır. Şekil 2.2'de Gaisler Kütüphanesiyle tasarlanmış Leon3 yapısı görülmektedir. Görüldüğü gibi yüksek band genişliği gerektiren yapılar AHB'ye bağlanmış, daha yavaş kalması sorun olmayacak IP'ler ise APB'ye bağlanmıştır. Ayrıca AHB ve APB bir köprü üzerinden birleştirilmiştir. Bu tez kapsamında tasarlanan görüntü sıkıştırma ve şifreleme IP'si, APB'ye bağlanmıştır.



Şekil 2.2: Leon3 ve GRLIB [11].

2.2 Gaisler Araçları

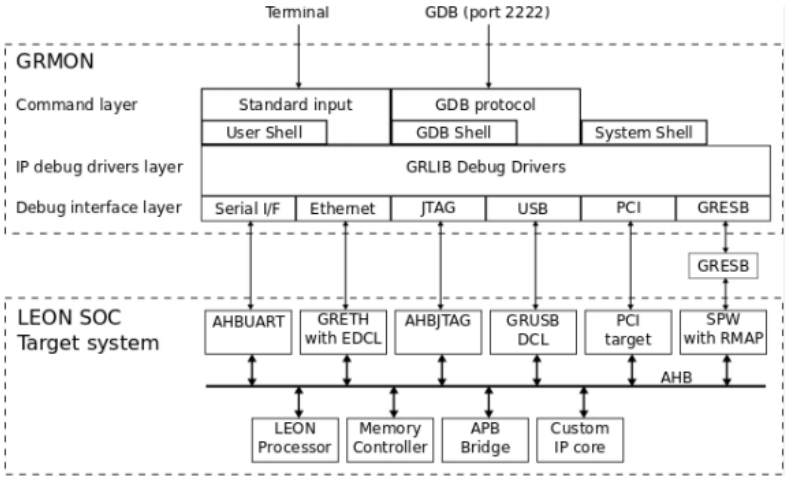
Gaisler Kütüphanesi'ni geliştiren Gaisler firması, tasarımların daha kolay bir şekilde yapılması ve hata ayıklama işlemlerinde bazı kolaylıklar sağlaması açısından çeşitli araçlar geliştirmişlerdir. Bu araçlardan Gaisler Monitor (GRMON) donanımın FPGA'ya yüklenmesi ve üzerinde hata ayıklama işlemlerinin yapılabilmesi açısından

kullanılmıştır. Bare-C Çapraz Derleyicisi ise IP, APB'ye eklendikten sonra yazılan C kodunun derlenmesi için kullanılmıştır.

2.2.1 Gaisler monitor

Gaisler Ekranı (Gaisler Monitor-GRMON), Leon işlemciler ve Gaisler Kütüphanesi tabanlı kırmık üstü sistem tasarımları için kullanılan bir hata ayıklama ekranıdır. Evrensel Seri Veriyolu (Universal Serial Bus-USB), Joint Test Action Group (JTAG), RS232, Çevresel Bileşen Ara Bağlantısı (Peripheral Component Interface-PCI), Ethernet ve SpaceWire hata ayıklama arayüzlerini (Debug Support Unit-DSU) desteklemektedir [12].

Şekil 2.3'te GRMON'un Leon3'e bağlantısı gösterilmiştir. GRMON, hedef donanımdaki adanmış arayüzlerden birini kullanarak bağlantı kurar. Hedef donanımdaki tüm hata ayıklama arayüzleri AHB master'ları olarak algılanır. Bu sebeple hata ayıklama işlemini yapabilmek için herhangi bir yazılıma ihtiyaç duyulmaz. GRMON'un bir diğer işlevi hedef donanımdaki yani Leon3 içeren Gaisler Kütüphanesi sistemindeki bağlı olan donanımları ve bu donanımların hangi adreste yer aldığını tespit edebilmesidir. Bu özellik sayesinde eklenen IP'nin sisteme doğru adresten eklenip eklenmediği görülebilir. Ayrıca işlemci üzerine işletim sistemi yada C/C++ kodu yükleme işlemleri de GRMON aracılığıyla gerçekleştirilir. GRMON hedef sisteme ilk bağlandığında öncelikle sistemde var olan IP'leri tarar. Bu işlem normal durumda AHB veri yolunda 0xFFFFF000 adresinde yer alan tak-oynat bilgisine bakılarak yapılır.



Şekil 2.3: GRMON [12].

2.2.2 Bare-C çapraz derleyicisi

Bare-C Çapraz Derleyicisi (Bare-C Cross Compiler-BCC), Leon3 mikroişlemcileri geliştirilmiş çapraz derleyicidir. C ve C++ uygulamalarının derleme işlemlerini yapar. Kayan nokta işlemlerini, SPARC V8 çarpma ve bölme komutlarını destekler. Aynı zamanda eCos çekirdeğinin derlenmesi için de kullanılabilmektedir [13]. Bu çalışmada da tasarlanan IP'nin bayrak işlemleri için yazılan C kodunun Leon3 için derlenmesi işleminde kullanılmıştır.

3. KÜÇÜK ŞİFRELEME ALGORİTMASI

Bu bölümde Küçük Şifreleme Algoritması (Tiny Encryption Algorithm-TEA) detaylı bir şekilde anlatılacaktır. JPEG görüntü şifreleme işleminde şifreleme bloğu olarak David Wheeler ve Roger Needham tarafından 1994 yılında geliştirilen TEA kullanılmıştır [14]. Günümüzde teknolojinin gelişmesiyle birlikte donanımların kaplayacağı yer önem kazanmıştır. Bu sebeple daha az enerji tüketen ve gerçeklendiğinde daha az yer kaplayacak bir şifreleme yöntemine ihtiyaç duyulmuştur ve TEA geliştirilmiştir. TEA gerçekleme kolaylığı, hızı, düşük enerji tüketmesi, düşük masraflı olması ve güvenli olması sebebiyle gömülü sistem tasarımlarına oldukça uygundur.

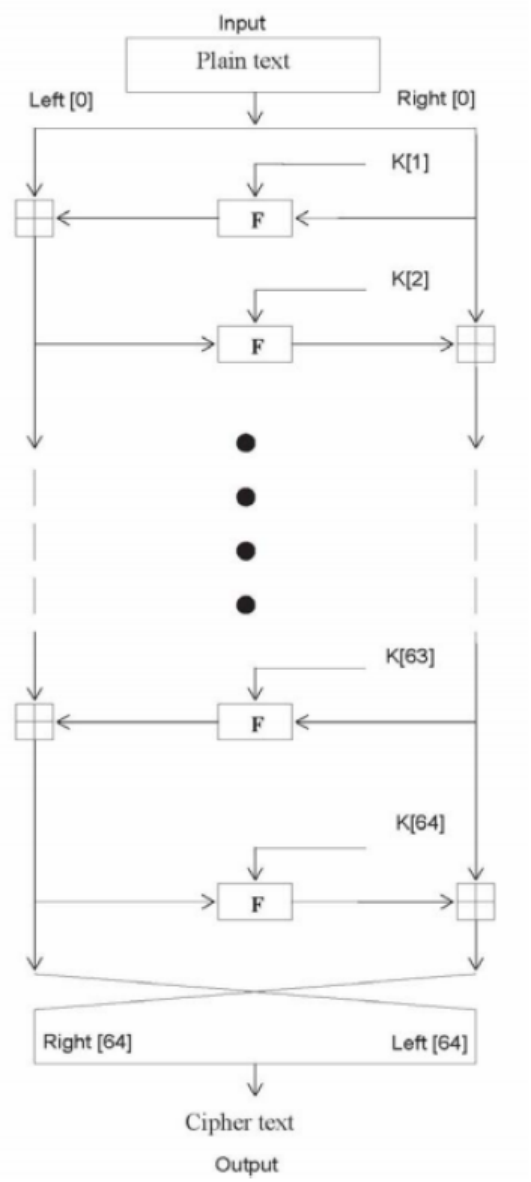
TEA, karışık cebirsel işlemleri kullanan ve Feistel türü şifreleme yapan, aynı zamanda minumum hafıza alanı ve maksimum hız hedeflenerek oluşturulmuş bir şifreleme algoritmasıdır. Feistel türü şifreleme yapması blok şifreleme yöntemine göre oluşturulduğu anlamına da gelmektedir. TEA, farksal şifre analizine (Differential Cryptanalysis) oldukça dirençlidir. Altı tur sonunda tam yayınım sağlamaktadır. Bir başka deyişle, şifrelenecek metinde 1 bit değiştirildiğinde bu değişiklik çıkıştaki şifrelenmiş metine 32 bit olarak yansımaktadır.

Blok şifreleme yapısında olması sebebiyle TEA, girişine uygulanan 64 bit veriyi tek tek şifrelemek yerine 64 biti tek blok olarak kabul ederek sanki tek bir bitmiş gibi şifreler. TEA şifreleme yapısında 128 bit uzunluklu anahtar kullanılmaktadır. Bu anahtar $K[0]$, $K[1]$, $K[2]$ ve $K[3]$ olarak 32 bit uzunluklu anahatarlara bölünerek işleme sokulur [15]. Şekil 3.1'de algoritmanın şifreleme kısmının blok diyagram yapısı görülmektedir. Şifreleme yapısı 64 adet Feistel döngüsünden oluşmaktadır. Şekil 3.1'den de görüldüğü gibi şifrelenmek istenen metin her biri 32'şer bit olmak üzere ikiye bölünerek sağ ve sol olarak yapıya verilir. Her bir döngüde farklı anahtar kullanılır. Girişe uygulanan $Sol[0]$ ve $Sag[0]$ bitleri sırasıyla $K[0]$, $K[1]$, $K[2]$,..., $K[64]$ anahtarları ile şifrelenerek 64 döngü sonunda $Sol[64]$ ve $Sag[64]$ bitleri olarak

şifrelenerek çıkışa verilir. Bu yapıda, her bir döngünün girişi bir önceki döngünün çıkışına bağlıdır.

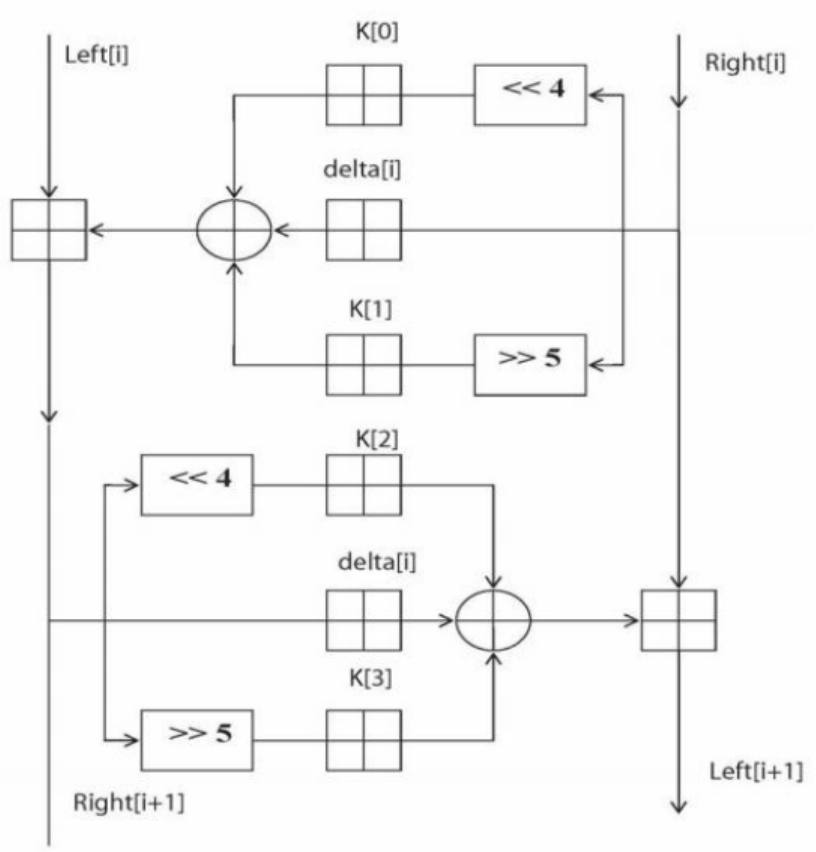
Her bir döngüde kullanılan 64 farklı $K[i]$ anahtarı, delta ismi verilen ve altın oran yardımıyla üretilen bir sabit sayesinde oluşturulur. Bu ifade, 3.1'de verilmiştir.

$$delta = (\sqrt{5} - 1) * 2^{31} = 9E3779B9_h \quad \textbf{(3.1)}$$



Şekil 3.1: TEA Şifreleme Rutini [15].

Şifreleme kısmının iç yapısı ise Şekil 3.2'de gösterilmiştir. Buna göre 64 Feistel döngüsünden oluşan şifreleme, 32 döngüden oluşmaktadır. Yani Şekil 3.2 iki tane Feistel döngüsü içermektedir. Her bir döngüde, döngüye girecek olan metinler özel veya (XOR), toplama ve mantıksal kaydırma işlemlerinden geçmektedir.



Şekil 3.2: TEA i. döngüsü [15].

Şekil 3.2'den de görüldüğü gibi şifreleme döngüsüne sağ taraftan giren şifresiz metine ilk olarak 4 bit sola kaydırma işlemi uygulanır. Daha sonra $K[0]$ anahtarı ile toplama işlemi yapılır. Aynı bilgi $K[1]$ anahtarı için 5 bit sağa kaydırılarak bir kez daha gerçekleştirilir. Aynı zamanda bilginin kendisi de $delta[i]$ sabitiyle toplama işlemine girer. Daha sonra özel veya bloğu girişine gelen bu üç değeri işleme sokar ve çıkış sol koldan gelen veriyle toplanır. Bu toplama işleminin sonucunda döngünün sol taraftan verdiği çıkış elde edilmiş olur. Bir Feistel turu önce sağdan girmiş olan metinle az önce elde edilen sol taraftan çıkan bilgi yine aynı işlemlerden geçirilerek döngünün sağ taraftan verdiği çıkış elde edilmiş olur. Bu işlemlere 32 tur devam edilir ve son turun çıkışında girişe verilen şifresiz metnin şifreli hali elde edilmiş olur.

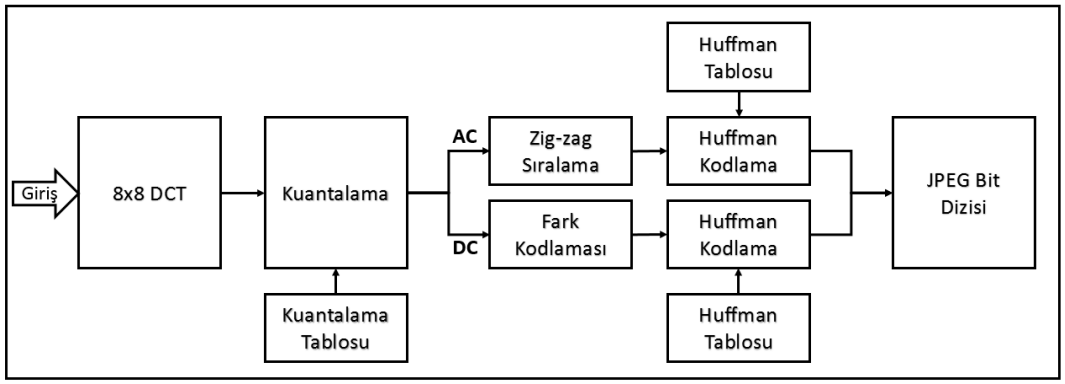
4. JPEG STANDARDI

Bu bölümde JPEG standardı ve temel blok yapıları detaylı bir şekilde anlatılacaktır. JPEG hem gri seviye hem de renkli resimlerin sıkıştırılabilmesi için geliştirilmiş bir sıkıştırma yöntemidir. İlk defa 1992 yılında ITU-T81 önerisi olarak geliştirilmiştir. Genel olarak iki türde gerçeklenmektedir. Bunlardan ilki olan Ayrık Kosinüs Dönüşümü (DCT) tabanlı yöntemde kayıplı sıkıştırma yapılmaktadır. Diğer yöntem tahminsel bir yöntem olmak birlikte kayıpsız sıkıştırma yapmaktadır. DCT tabanlı yöntem, günümüzde kullanılan bir çok uygulama için yeterli olmaktadır. Bu çalışmada da DCT tabanlı JPEG yöntemi gerçeklenmiştir. Aşağıda verilen şekillerde dört tane modu vardır [16].

- **Kayıpsız mod:** Sıkıştırma oranı orjinal resimden az bile olsa, buradaki tek amaç görüntünün her bir pikselinin geri kazanılabilmesidir.
- **Ardışıl mod:** Görüntü soldan sağa ve yukarıdan aşağı olacak şekilde sıkıştırılır.
- **İlerlemeli mod:** Çoklu taramalarla sıkıştırma yapılır.
- **Hiyerarşik mod:** Görüntü birden çok çözünürlükte sıkıştırılır. Böylece en düşük çözünürlüklü resim diğer çözünürlüklere sıkıştırma uygulanmadan önce elde edilir.

Yukarıda bahsedilen son üç mod kayıplı sıkıştırma yapmaktadır. Bunun sebebi içerisinde kuantalama bloğu bulundurmasıdır. Kayıpsız mod ise adından da anlaşılacağı üzere içinde kuantalama bloğu bulundurmadığından görüntü üzerinde herhangi bir kayıp oluşturmaz. Bu çalışmada kullanılan DCT tabanlı yöntem ise ardışıl moda dahildir. Şekil 4.1'de JPEG'e ait tüm bloklar gösterilmiştir.

JPEG standardı sadece sıkıştırılmış bit dizisini üretmektedir. Yani sonuçta üretilen dosya formatıyla ilgili herhangi bir bilgi içermez. Üretilen sonuç görüntüsünün bilgisayar gibi cihazlarda tanımlanabilmesi için IJG (Independent JPEG Group) tarafından JFIF (JPEG File Interchange Format) geliştirilmiştir. Bu standard sonucunda üretilen görüntü, sıkıştırılmış bit dizisinin yanında JPEG dönüşümünde



Şekil 4.1: DCT Tabanlı JPEG.

kullanılan tablo bilgileri de yer alır. Bunlar ilerleyen alt bölümlerde detaylı bir şekilde incelenmiştir.

Şekil 4.1'deki yapının temelini Ayrık Kosinüs Dönüşümü oluşturmaktadır. JPEG kodlayıcı girişine verilen 8x8'lik görüntü blokları ilk olarak DCT bloğuna gelmektedir. DCT bloğu çıkışında görüntü frekans tanım bölgesine çevrilmiş ve her bir elemanın frekans tanım bölgesi katsayıları çıkışa verilmiştir. Bir sonraki aşamada tüm katsayılar kuantalanmaktadır. Kuantalama bloğunda tüm matris elemanları yuvarlanmaktadır. Bu sebeple kayıp oluşmaktadır. Kuantalama bloğundan sonra tüm matris elemanları zig-zag sıralama bloğuna girer ve bu aşamadan sonra AC ve DC katsayılar için ayrı ayrı kodlama işlemi uygulanmaktadır. Kodlama kısmında sıkıştırma işlemi gerçekleştirilmektedir ve gelen tüm katsayılara ilişkin kod kelimeleri atanır. DCT sonucunda elde edilen yüksek frekanslı ve düşük değerli veriler, kuantalama sonucunda sıfır olmaktadır. Kodlama kısmında ise bu sıfırların kodlanmasına ihtiyaç duyulmamakta ve bu sayede sıkıştırma sağlanmaktadır. Kuantalama ve kodlama bloklarında özel tablolar kullanılmaktadır. Bu tablolar değişik uygulamalara göre değişiklik gösterebilirler. Genelde DCT sonucu elde edilen matrisin ilk elemanı olan DC katsayı için ve diğer katsayılar olan AC katsayılar için ayrı kodlama yöntemleri kullanılmaktadır. Bütün bu bloklar aşağıdaki bölümlerde detaylı bir şekilde anlatılmıştır.

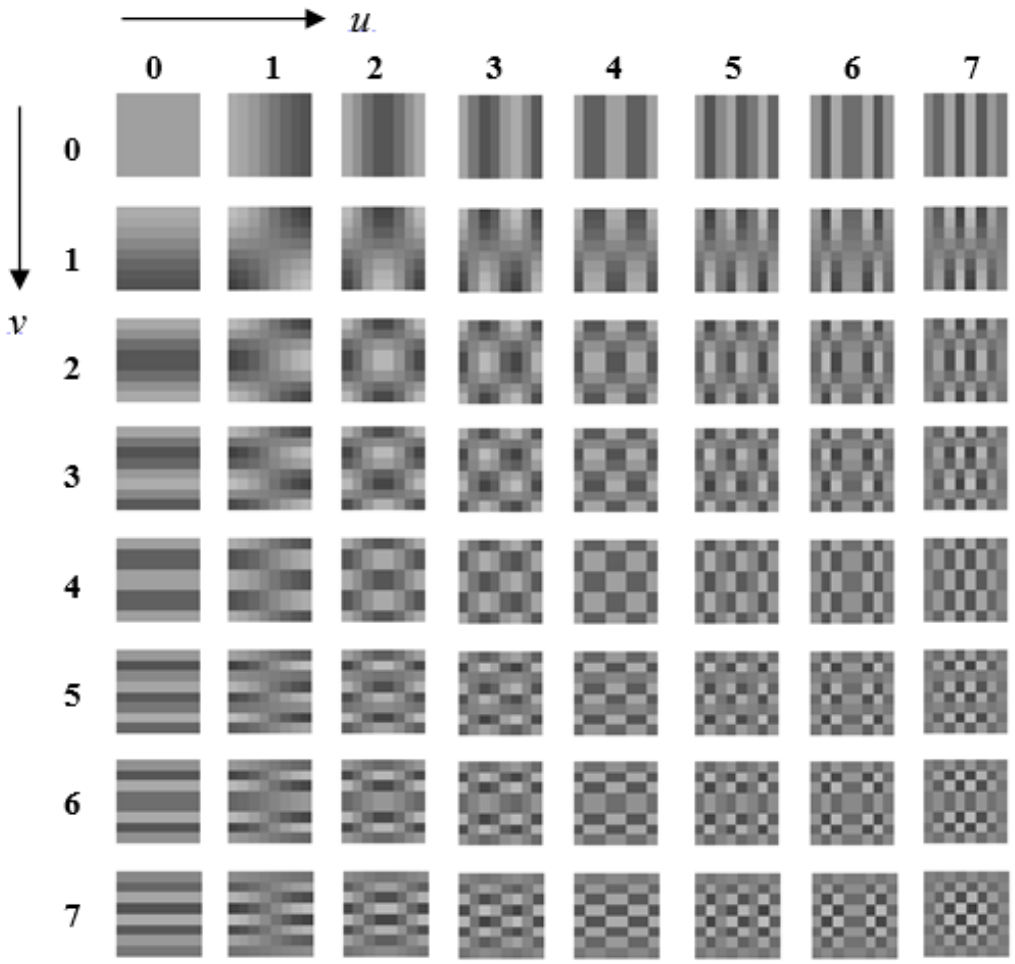
4.1 Ayrık Kosinüs Dönüşümü

JPEG kodlama bloğunun ilk adımı olan ayrık kosinüs dönüşümü, birçok video ve görüntü sıkıştırma standardında kullanılmaktadır. Girişine gelen görüntü verilerini frekans tanım bölgesine çevirmektedir ve bu frekans değerleri görüntünün içindeki

bilgiyi temsil etmektedir. Giriş olarak kullanılan görüntü iki boyutlu olduğundan burada da iki boyutlu DCT bloğu tanıtılacaktır. DCT uygulamak için öncelikle giriş görüntüsü 8x8'lik bloklara bölünür. Bunun sebebi, 8x8'lik blokların hesaplama bakımından fazla karmaşıklık getirmemesi ve daha büyük değerlerin seçilmesinin daha ideal sonuçlar getirmemesidir [17]. Bölünen 8x8'lik bloklar soldan sağa ve yukarıdan aşağı olacak şekilde işleme sokulurlar. İki boyutlu DCT'nin matematiksel ifadesi Denklem 4.1'de verilmiştir. Şekil 4.2'de ise DCT'nin işleniş şekli gösterilmiştir. Çizelge 4.1 ve Çizelge 4.2'de örnek bir 8x8'lik giriş matrisi için DCT sonucu elde edilen çıkış matrisi verilmiştir. Çizelge 4.2'deki ilk elemana DC katsayı, diğer 63 elemana ise AC katsayı ismi verilir.

$$F(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7} f(x,y)\cos[\frac{\Pi(2x+1)u}{16}]\cos[\frac{\Pi(2y+1)v}{16}] \quad \textbf{(4.1a)}$$

$$u = 0...7, v = 0...7 \quad \textbf{(4.1b)}$$



Şekil 4.2: 8x8 DCT

Çizelge 4.1: Örnek 8x8'lik Giriş Matrisi

48	39	40	68	60	38	50	121
149	82	79	101	113	106	27	62
58	63	77	69	124	107	74	125
80	97	74	54	59	71	91	66
18	34	33	46	64	61	32	37
149	108	60	106	116	61	73	92
211	233	159	88	107	158	161	109
212	104	40	44	71	136	113	66

Çizelge 4.2: Örnek DCT Katsayı Matrisi

699.25	43.18	55.25	72.11	24.00	-25.51	11.21	-4.141
-129.78	-71.50	-70.26	-73.35	59.43	-24.02	22.61	-2.05
85.71	30.32	61.78	44.87	14.84	17.35	15.51	-13.19
-40.81	10.17	-17.53	-55.81	30.50	-2.28	-21.00	-1.26
-157.50	-49.39	13.27	-1.78	-8.75	22.47	-8.47	-9.23
92.49	-9.03	45.72	-48.13	-58.51	-9.01	-28.54	10.38
-53.09	-62.97	-3.49	-19.62	56.09	-2.25	-3.28	11.91
-20.54	-55.90	-20.59	-18.19	-26.58	-27.07	8.47	0.31

Denklem 4.1'de verilen ifade Denklem 4.2'deki gibi matris çarpımı olarak da ifade edilmektedir. Bu ifadedeki Z DCT dönüşüm sonucunu, C ise katsayı matrisini ifade etmektedir. C matrisinin katsayıları Çizelge 4.3'teki gibidir. Bu elemanlar 4.4'teki ifadeler kullanılarak hesaplanır. Bu çalışmada kullanılan C matrisi ise Çizelge 4.5'te verildiği gibi kullanılmaktadır. Normalde C matrisi bu değerlerin 65536'ya bölünmüş halidir. Fakat bu tez kapsamında donanım tasarımı yapılacağından C matrisi bu şekilde kullanılmıştır.

$$Z = CXC^T \tag{4.2}$$

$$C = \begin{bmatrix} a & a & a & a & a & a & a & a \\ b & c & f & g & -g & -f & -c & -b \\ d & e & -e & -d & -d & -e & e & d \\ c & -g & -b & -f & f & b & g & -c \\ a & -a & -a & a & a & -a & -a & a \\ f & -b & g & c & -c & -g & b & -f \\ e & -d & d & -e & -e & d & -d & e \\ g & -f & c & -b & b & -c & f & -g \end{bmatrix} \tag{4.3}$$

$$a = \sqrt{\frac{1}{8}}, b = \sqrt{\frac{1}{4}}\cos(\frac{\Pi}{16}), c = \sqrt{\frac{1}{8}}\cos(\frac{3\Pi}{16}), d = \sqrt{\frac{1}{4}}\cos(\frac{\Pi}{8})$$
$$e = \sqrt{\frac{1}{4}}\cos(\frac{3\Pi}{8}), f = \sqrt{\frac{1}{4}}\cos(\frac{5\Pi}{16}), g = \sqrt{\frac{1}{4}}\cos(\frac{7\Pi}{16}) \quad \textbf{(4.4)}$$

$$C = \begin{bmatrix} 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 \\ 32138 & 27246 & 18205 & 6393 & -6393 & -18205 & -27246 & -32138 \\ 30274 & 12540 & -12540 & -30274 & -30274 & -12540 & 12540 & 30274 \\ 27246 & -6393 & -32138 & -18205 & 18205 & 32138 & 6393 & -27246 \\ 23170 & -23170 & -23170 & 23170 & 23170 & -23170 & -23170 & 23170 \\ 18205 & -32138 & 6393 & 27246 & -27246 & -6393 & 32138 & -18205 \\ 12540 & -30274 & 30274 & -12540 & -12540 & 30274 & -30274 & 12540 \\ 6393 & -18205 & 27246 & -32138 & 32138 & -27246 & 18205 & -6393 \end{bmatrix} \quad \textbf{(4.5)}$$

4.2 Kuantalama

JPEG standardına göre DCT bloğundan sonra kuantalama bloğu gelmektedir. Bu blokta frekans tanım bölgesine geçirilmiş olan görüntü matrisindeki yüksek frekans terimlerinin çoğunlukla sıfırlanması hedeflenmektedir. Bu sayede elde edilen sıfırlar bir sonraki blokta kodlanmayacak ve görüntü boyutunda azalma sağlanmış olacaktır. Kuantalama işlemi JPEG standardında anahtar rol oynamaktadır. Bu işlem sırasında orijinal görüntü içindeki yüksek frekanslı pikseller yok edilir. Bunun yapılmasının sebebi insan gözünün görüntü içerisindeki düşük frekanslara karşı daha hassas olmasıdır. Bu sebeple yüksek frekanslı piksellerin yok edilmesi görüntü üzerinde fazla bozulma yapmamaktadır ve sıkıştırma işlemi de bu pikseller üzerinden yapılır.

64 adet DCT katsayısının hepsinin kuantalanması gerekmektedir. Kuantalama işlemi Eşitlik 4.6'da verilmiştir. Bu eşitlikte yer alan Q kuantalama matrisi standardlarda berirlenmiştir ve 4.7'de verilmiştir. Bu eşitlik gri seviyede sıkıştırma yapılacağı durumda geçerlidir. Kuantalama matrisi kullanıcı tarafından değiştirilebilir ve matristeki her elaman 1 ile 255 arasında değişen tamsayılardır. Matristeki katsayıların boyutu görüntü kalitesini ve sıkıştırma oranını etkilemektedir. DCT matrisindeki yüksek frekanslı ve düşük bilgi taşıyan elemanların kuantalama sonucunda sıfırlanması beklenmektedir. Çizelge 4.1'de verilen örnek DCT matrisi için elde edilen kuantalama matrisi Çizelge 4.3'te gösterilmiştir.

$$F_q(u,v) = Round\frac{F(u,v)}{Q(u,v)} \tag{4.6}$$

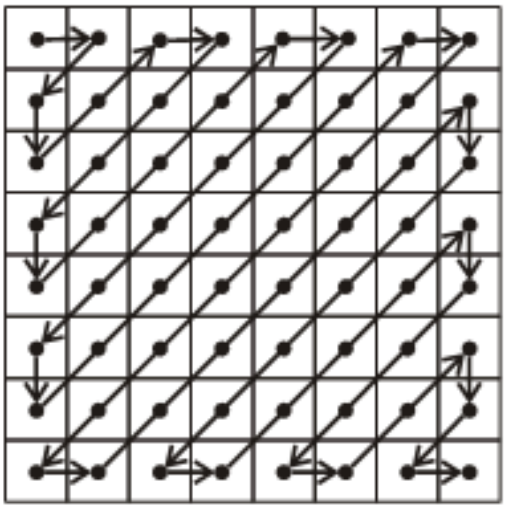
$$C = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{4.7}$$

Çizelge 4.3: Örnek Kuantalama Sonuç Matrisi

44	4	6	5	1	-1	0	0
-11	-6	-5	-4	2	0	0	0
6	2	4	2	0	0	0	0
-3	1	-1	-2	1	0	0	0
-9	-2	0	0	0	0	0	0
4	0	1	-1	-1	0	0	0
-1	-1	0	0	1	0	0	0
0	-1	0	0	0	0	0	0

4.3 Zig-Zag Tarama

DCT ve kuantalama bloklarından sonra zig-zag tarama bloğu gelmektedir. Bu blok sonucunda kuantalanmış 8x8'lik görüntü verisini içeren matris seri hale dönüştürülür. Tüm bu verilerin sıralanması zig-zag sıralamasında yapılır. Bunu yapmanın amacı matris elemanlarını düşük frekanstan yüksek frekansa doğru artacak biçimde dizmektir. Böylece ilk sıradaki eleman DC katsayı olacaktır. Kuantalama bloğunda sıfırlanan elemanlar ise burada yan yana gelecektir. Bu durum daha önce de söylendiği gibi kodlama kısmında kolaylık sağlayacaktır. Zig-Zag sıralamanın nasıl yapıldığı Şekil 4.3'te gösterilmiştir.



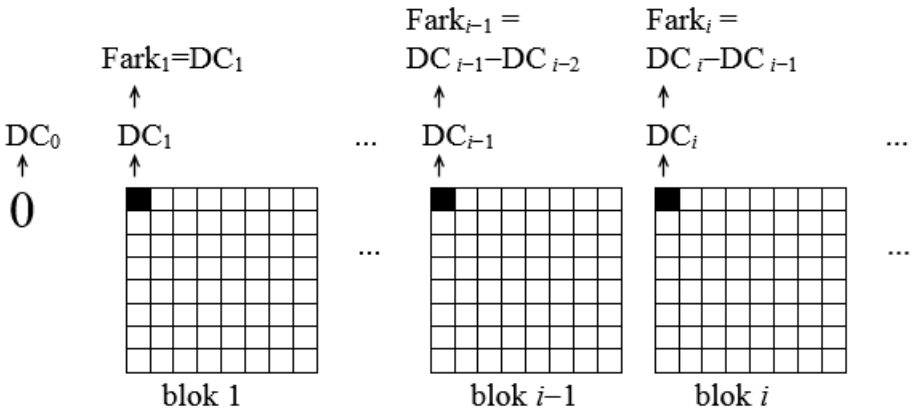
Şekil 4.3: Zig-Zag Yol.

4.4 DC Katsayılar İçin Farksal Kodlama

Kodlama kısmı JPEG standardında sıkıştırmanın esas yapıldığı yerdir. DC katsayıların en çok enerji taşıyan katsayılar olması sebebiyle, genelde AC katsayılardan daha büyük değerdedirler. Ayrıca peşpeşe gelen 8x8'lik blokların DC katsayıları arasında da ilişki vardır. Bu sebeple JPEG kodlamada tüm DC katsayıları kodlamak yerine aralarındaki fark kodlanmaktadır. Matematiksel ifadesi 4.8'de verilmiştir.

$$Fark_i = DC_i - DC_{i-1} \tag{4.8}$$

i=0 için DC katsayı 0 olarak kabul edilerek işleme başlanır. Daha sonra sırayla gelen bloklar için DC farksal kodlama yapılır. Bu işlem Şekil 4.4'te gösterilmiştir. Buradan elde edilen farksal kodlamalar daha sonra Huffman bloğuna gönderilirler.



Şekil 4.4: 8x8'lik blokların DC farksal kodlaması.

4.5 AC Dizi Uzunluğu (Run Length) Kodlama

Kuantalama ve zig-zag bloklarından sonra içinde birçok sıfır içeren dizi elde edilmişti. Bu aşamada her 8x8'lik blokta yer alan 63 adet AC katsayı kodlanır. Örnek olarak 57, 45, 0, 0, 0, 0, 23, 0, -30, -16, 0, 0, 1, 0 .... 0 şeklinde bir dizi varsa AC dizi uzunluğu kodlaması şu şekilde olacaktır: (0,57); (0,45); (4,23); (1,-30); (0,-16); (2,1); EOB.

EOB (End of Block) kodu özel olarak belirlenmiştir. Eğer zig-zag bloğundan gelen vektörün kalan elemanlarının hepsi 0 ise EOB bloğu konularak bütün sıfırların gönderilmesi engellenebilir. Eğer zig-zag bloğundan gelen vektör 0 ile bitmiyorsa, yani son eleman 0 değil ise, EOB koyulmaz.

4.6 Huffman Kodlama

JPEG standardına göre, gerçek değerlerin yerine her birine atanan kodlar saklanmaktadır. Bunun için standard gereği bir tablo belirlenmiştir. Bu tablo Çizelge 4.4'te verilmiştir. Bir önceki bölümde verilen (0,57); (0,45); (4,23); (1,-30); (0,-16); (2,1); (0,0) örneğine tekrar bakılacak olursa burada sadece sağ taraftaki sayılar kodlanacaktır. EOB'yi ifade eden (0,0)'a herhangi bir kod ataması yapılmaz. 57 değerine tablodan bakıldığında 6. kategoride olduğu görülmektedir ve ona ait bit değeri 111001 olarak belirlenmiştir. Bu yüzden 57 değeri 6, 111001 olarak kodlanır. Tüm sayılar tablodan bakılarak kodlanırsa şu elde edilir: (0,6,111001); (0,6,101101); (4,5,10111); (1,5,00001); (0,4,0111); (2,1,1); (0,0).

Parantez içindeki ilk iki değer 0 ile 15 arasında olduklarından bayt olarak temsil edilmektedir. Bir bayt içindeki yüksek anlamlı dört bit 0 sayısını, düşük anlamlı 4 bit ise 0 olmayan değerin kategorisini temsil eder.

Kodlamadaki ilk aşama AC katsayılar için Huffman Kodlama'dır. Örneğin girişe gelen (0,6) 1111000 olarak kodlanır. (4,5) ise 1111111110011001 olarak kodlanır. Bu değerler Çizelge 4.5'te verilmiştir. Bir önceki örnekte verilen değerler için kodlama şu şekilde olacaktır: 1111000 1111001, 111000 101101, 1111111110011000 10111, 11111110110 00001, 1011 0111, 11100 1, 1010.

Çizelge 4.4: Kategori ve Bit Kodlama Tablosu.

Kategori	Değer	Bit Değeri
1	-1,1	0,1
2	-3,-2,2,3	00,01,10,11
3	-7,-6,-5,-4,4,5,6,7	000,001,010,011,100,101,110,111
4	-15,...,-8,8,...,15	0000,...,0111,1000,...,1111
5	-31,...,-16,16,...,31	00000,...,01111,10000,...,11111
6	-63,...,-32,32,...,63	000000,...,011111,100000,...,111111
7	-127,...,-64,64,...,127	0000000,...,0111111,10000000,...,1111111
8	-255,...,-128,128,...,255	...
9	-511,...,-256,256,...,511	...
10	-1023,...,-512,512,...,1023	...
11	-2047,...,-1024,1024,...,2047	...

Çizelge 4.5: AC Katsayılar için Huffman Tablosu.

AC Dizi Uzunluğu/Kategori	Kod Uzunluğu	Kod Kelimesi
0/0	4	1010
...		
0/6	7	1111000
...		
0/10	16	1111111110000011
1/1	4	1100
...		
4/5	16	1111111110011000
...		
15/10	16	1111111111111110

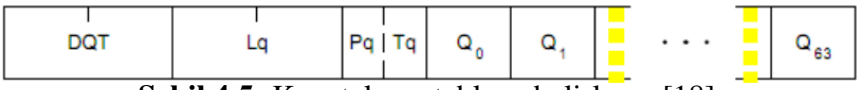
DC katsayıların farklarının da kodlanması gerekmektedir. Daha önce hesaplanan farkın kategori ve bit değeri seçiminin yapılması gerekmektedir. Daha sonra yalnızca kategori değeri dikkate alınarak Huffman kodlaması yapılır. Bunun için Çizelge 4.6'dan faydalanılır. Örnek olarak eğer hesaplanan fark -511 ise, bu değer (9,000000000) şeklinde temsil edilir. 9'un Huffman değeri de 1111110 olduğu için tüm ifade 1111110 000000000 şeklinde temsil edilir. Son olarak daha önce hesaplanan AC katsayıların kodlanması ile DC katsayıların kodlanması birleştirilirse 64 katsayılık bir vektör için hesaplanan kodlama şu şekilde olacaktır: 1111110 000000000, 1111000 1111001, 111000 101101, 1111111110011000 10111, 11111110110 00001, 1011 0111, 11100 1, 1010.

Çizelge 4.6: DC Katsayılar için Huffman Tablosu.

Kategori	Kod Uzunluğu	Kod Kelimesi
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110

4.7 JPEG Dosya Yapısı

JPEG formatındaki görüntülerin bilgisayarlar tarafından tanımlanarak açılması için sadece kodlama yapılması yeterli olmaz. Sıkıştırılmış verinin başına ve sonuna standart gereği eklenmesi gereken bazı özel kodlar vardır. Bu kodlar hangi kodlama ve kuantalama tablolaların kullanıldığını, görüntünün boyutunu, görüntünün gri seviye mi renkli mi olduğu gibi bilgiler içerir [18].



Şekil 4.5: Kuantalama tablosu belirleme [18].

JPEG dosyasında az önce bahsedilen komutların hepsi "FF" ile başlar. Bu sebeple eğer kodlama bloğunun herhangi bir yerinde "FF" değeri gelirse, bunun özel bir kod olarak algılanmaması için arkasına "00" eklenmelidir. Şekil 4.5'te kuantalama tablosu belirleme yapısı verilmiştir. Dosya yapısının en başında DQT ile belirtilen görüntünün başladığını belirten "FF D8" komutu yer alır. Aynı zamanda kuantalama tablosunun başladığını belirtmek için kullanılır. 16 bitten oluşan Lq 4.9 ifadesi ile bulunur. 'n' kac tane kuantalama tablosu kullanıldığını ifade eder. Pq ise 4 bit ile ifade edilen kuantalama tablosunun bit çözünürlüğüdür. Kullanılan temel method için 0 değerini alır ve bu 8 bit çözünürlük anlamına gelir. Tq, tanımlanan tablonun numarasını tanımlar ve 4 bittir. Qi'de ise kuantalama matrisinin 64 değeri zig-zag düzeninde girilir.

$$2+\sum_{t=1}^{n}(65+64xPq(t)) \tag{4.9}$$

Yukarıda anlatılanlar sadece kuantalama tablosu için belirlenmiş özel kodlardır. Kuantalama bölümünden sonra görüntüye ait bazı özellikleri belirleyen çerçeve bilgisi girilmelidir. Bu kısım "FF C0" kodu ile başlar ve arkasına 16 bitlik çerçeve uzunluğu girilir. Çerçeve uzunluğu "$8 + 3.Nf$" ile belirlenir. Nf resmin renk bileşeni sayısını temsil eder. Çerçeve uzunluğundan sonra piksel bit çözünürlüğü girilir. Satır ve sütun sayısı için her biri 16 bit olarak belirlendikten sonra Nf değeri girilir. Daha sonra kullanılacak olan kuantalama tablosuna ait numara girilir. Çerçeve bilgisi de bittikten sonra DC ve AC kod tabloları tanımlanmalıdır. Bunun için başlangıç kodu "FF C4" olarak girilir ve sonrasında tanımlanacak tablodaki eleman sayısı girilir. Bir sonraki aşamada sıkıştırılmış görüntünün kodları başlar. Bunun başlangıç kodu "FF DA" olarak belirlenmelidir. Sıkıştırılmış görüntüye ait kodlar bittikten sonra görüntünün bittiğini gösteren "FF D9" kodu girilmelidir. Böylece tüm JPEG dosyası tanımlanmış olur.

5. DONANIM TASARIMI

Bu çalışmanın tasarım kısmı JPEG kodlayıcının Verilog HDL kullanılarak donanım tasarımının yapılması, şifreleme bloğunun tasarımı ve bütün donanımın FPGA üzerinde gerçeklenen Leon3 mikroişlemcisine eklenmesinden oluşmaktadır. Aşağıdaki bölümlerde JPEG kodlayıcıdaki blokların tasarımında kullanılan yöntemlerin yanı sıra şifreleme bloğunun tasarımı ve Leon3 mikroişlemcisine donanım eklenmesinin nasıl yapılacağı ayrıntılı bir biçimde incelenecektir.

5.1 JPEG Kodlayıcının Tasarlanması

4. bölümde anlatılan JPEG bloklarının her biri FPGA üzerinde gerçeklenmiştir. Tasarım için Virtex 6 ML605 geliştirme kiti kullanılmıştır [19]. Tasarımda işleyişin sürekli olmasına önem verilmiştir.

5.1.1 DCT bloğu

DCT ifadesi daha önce Eşitlik 4.2'de verilmişti. Bu ifadeye göre öncelikle X matrisiyle C matrisinin transpozunun çarpımı hesaplanacak (Z), daha sonra ise elde edilen sonuç soldan C matrisi ile çarpılacaktır. C matrisindeki katsayılar virgüllü sayılar olduğundan hepsinin 65536 katı ile işlemler yapılmıştır. Daha sonra elde edilen sonuçlar kuantalama bloğu öncesinde tekrar 65536'ya bölünerek doğru değerlerin elde edilmesi sağlanmıştır. Bu bölme işlemi için 16 bit sağa kaydırma yapılmıştır. Kullanılan C matrisi 4.5'te verilmişti.

Yukarıda anlatılanlara göre bir boyutlu DCT'ye denk düşen Y'nin hesabı için Eşitlik 5.1 kullanılır. Burada k 0 ile 7 arasındadır. Sütun numarasına göre işaretler sırasıyla + ve - olarak değişmektedir. Dolayısıyla toplam dört adet çarpıcı ve 4 adet ekle-çıkar bloğu ile ara işlemler yapılabilir. En sona koyulacak bir toplayıcı ile ise Z hesaplanabilir. Daha önce de söylendiği üzere Z ilk bir boyutlu DCT sonucudur. Esas istediğimiz DCT sonucunu elde etmek için Z, C ile çarpılarak sonuç (Y) elde edilir. Bu adım da ilk adıma benzemektedir. Yani Z'nin her bir sütunu kullanılarak Y'de karşılık

gelen sütun hesaplanabilir. Fakat bu defa ikinci bir boyutlu dönüşüme geçtiğimizden Z'nin sütunlarına dönüşüm uygulanır. İlk bir boyutlu ve ikinci bir boyutlu DCT'nin hesaplanması Şekil 5.1 ve Şekil 5.2'de gösterilmiştir. Bu yapılarda görülen ekle-çıkar bloklarının yapacağı işlem bir kontrol girişiyle belirlenmiştir.

$$Z_{(k,0)} = 23170.(X_{k0} + X_{k1} + X_{k2} + X_{k3} + X_{k4} + X_{k5} + X_{k6} + X_{k7}) \quad \textbf{(5.1a)}$$

$$Z_{(k,0)} = 32138.(X_{k0} - X_{k7}) + 27246.(X_{k1} - X_{k6}) + 18205.(X_{k2} - X_{k5}) + 6393.(X_{k3} - X_{k4}) \quad \textbf{(5.1b)}$$

$$Z_{(k,0)} = 30274.(X_{k0} + X_{k7}) + 12540.(X_{k1} + X_{k6}) - 12540.(X_{k2} + X_{k5}) - 30274.(X_{k3} + X_{k4}) \quad \textbf{(5.1c)}$$

$$Z_{(k,0)} = 27246.(X_{k0} - X_{k7}) - 6393.(X_{k1} - X_{k6}) - 32138.(X_{k2} - X_{k5}) - 18205.(X_{k3} - X_{k4}) \quad \textbf{(5.1d)}$$

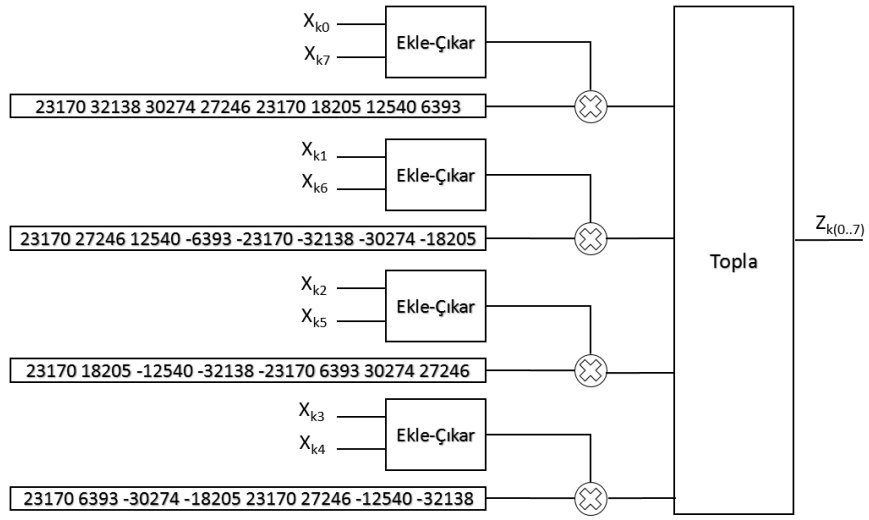
$$Z_{(k,0)} = 23170.(X_{k0} + X_{k7}) - 23170.(X_{k1} + X_{k6}) - 23170.(X_{k2} + X_{k5}) + 23170.(X_{k3} + X_{k4}) \quad \textbf{(5.1e)}$$

$$Z_{(k,0)} = 18205.(X_{k0} - X_{k7}) - 32138.(X_{k1} - X_{k6}) + 6393.(X_{k2} - X_{k5}) + 27246.(X_{k3} - X_{k4}) \quad \textbf{(5.1f)}$$

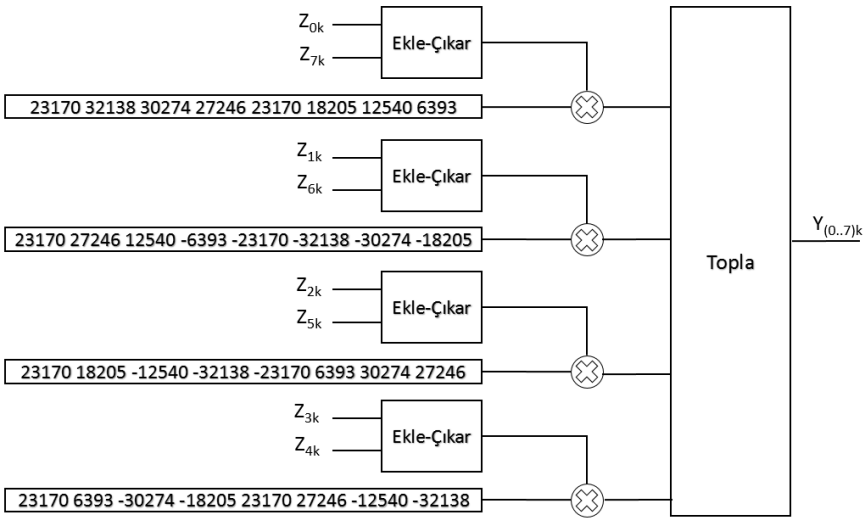
$$Z_{(k,0)} = 12540.(X_{k0} + X_{k7}) - 30274.(X_{k1} + X_{k6}) + 30274.(X_{k2} + X_{k5}) + (X_{k3} + X_{k4}) \quad \textbf{(5.1g)}$$

$$Z_{(k,0)} = 6393.(X_{k0} - X_{k7}) - 18205.(X_{k1} - X_{k6}) + 27246.(X_{k2} - X_{k5}) + 32138.(X_{k3} - X_{k4}) \quad \textbf{(5.1h)}$$

Yükselen kenar tetiklemeli olarak çalışan DCT modülü, asenkron aktif "1" reset girişine sahiptir. Reset'in 0 yapılmasıyla her saat periyodunda bir piksel girişi verilmesi gerekmektedir ve bu girişlerin görüntünün sonuna kadar aralıksız devam etmesi gerekmektedir. 8x8'lik bloklar halindeki giriş, soldan sağa ve yukarıdan aşağıya doğru verilir. DCT standardı gereği öncelikle tüm piksellerden 128 değeri çıkarılarak, piksellerin 0 etrafında salınması sağlanır. Her saat işaretinde bir piksel değeri gelmektedir. Şekil 5.1 ve Şekil 5.2'de kullanılan yapıların girişine 8 piksel değeri de aynı anda gelmiş olmalıdır. Bunu sağlamak amacıyla Şekil 5.3'te verilen ping-pong tampon bellek kullanılmıştır [20].

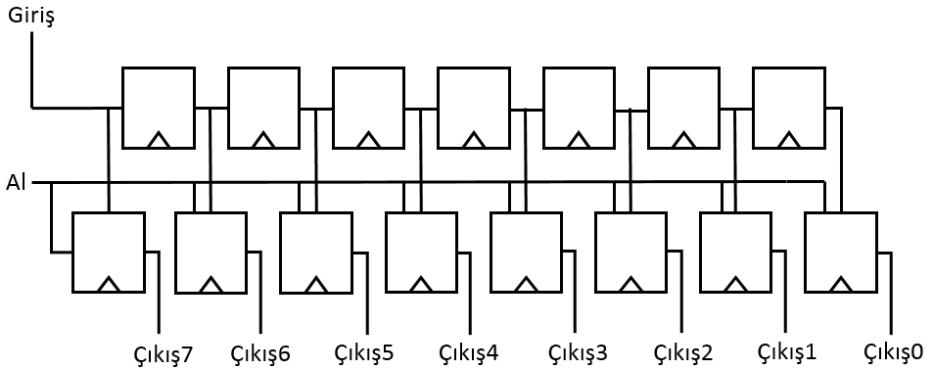


Şekil 5.1: Birinci bir boyutlu DCT.



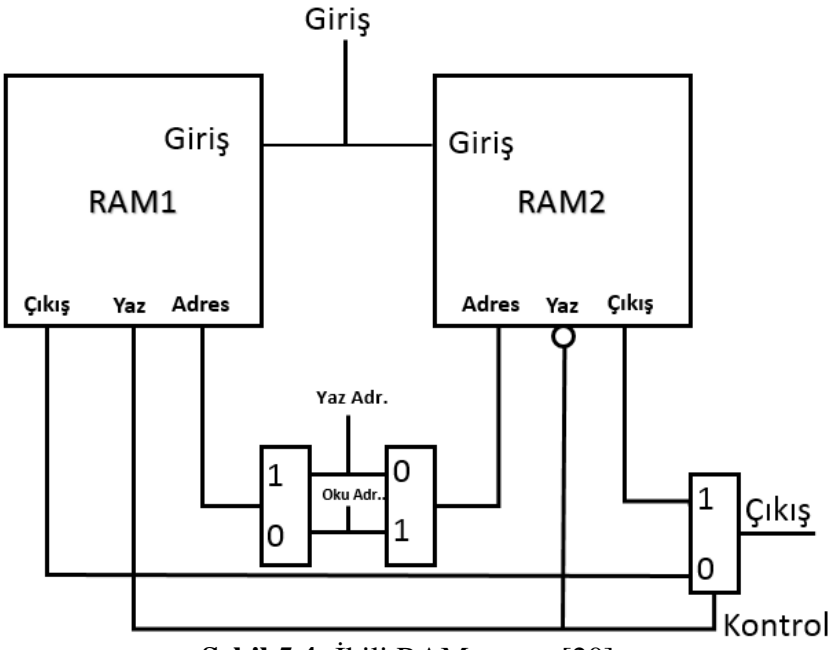
Şekil 5.2: İkinci bir boyutlu DCT.

Şekil 5.3'teki ping-pong tampon belleğe girişler sürekli gelmektedir ve her saat işaretinde kaydırılmak suretiyle ilerlemektedirler. 8 tane giriş geldikten sonra "Al" kontrol işareti aktif olarak girişler örneklenip çıkışa verilmektedir. Bunlar DCT hesaplarında kullanılacak çıkışlardır. Bahsedilen hesaplar devam ederken yeni gelen girişler giriş kaydedicileri tarafından kaydedilmektedir. Bu da DCT hesabında sürekliliği sağlamaktadır. İlk giriş gelmeye başladıktan sonra 14. saat işaretinde bir boyutlu dönüşüm katsayıları çıkışa gelmeye başlamaktadır. Elde edilen bu çıkışla Şekil 5.4'de gösterilen yapı kullanılarak kaydedilmektedir. Kaydedilen bir boyutlu DCT sonuçları, ikinci bir boyutun hesaplanacağı bloğa giriş olarak gitmektedir ve 95. saat periyodunda ilk istenen çıkışlar gelmeye başlar. İkili RAM yapısı az önceki hesapları yaparken bu sırada yapıya giriş gelmeye devam etmektedir. Yeni



Şekil 5.3: Ping-pong tampon bellek [20].

gelen piksellerin birinci boyut DCT sonuçları da diğer bir RAM'e kaydedilir. Bu iki RAM'in dönüşümlü kullanılması çıkıştaki sonuçların sürekli olarak var olmasını sağlar ve böylece hiç bir giriş değeri kaçırılmamaktadır. Görüldüğü üzere ilk RAM'e birinci bir boyutlu DCT sonuçları yazılırken diğer RAM daha önce yazılmış sonuçları ikinci bir boyutlu DCT bloğuna sağlar. Böylece birinci bir boyutlu DCT bloğu hesaplamalarının bitmesini beklemeden kendi RAM'ine yaptığı işlemleri kaydedebilir. İkinci bir boyutlu DCT bloğu için ise sonuçlar kaydedilmez. Bunun yerine "TAMAM" bayrağı kaldırılarak gelen her saat işaretinde bir sonuç sürekli çıkışa verilir.



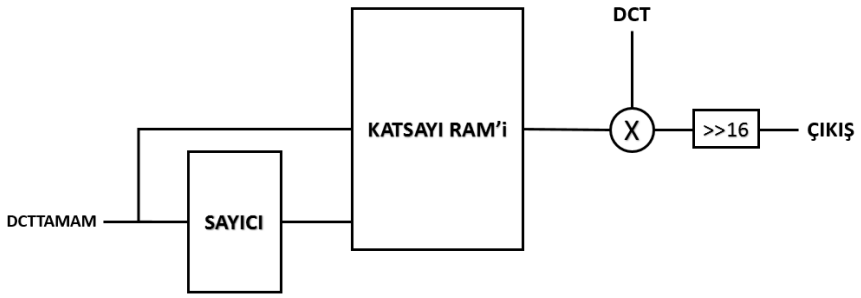
Şekil 5.4: İkili RAM yapısı [20].

5.1.2 Kuantalama bloğu

Daha önce bahsedildiği gibi kuantalama bloğu bölme işlemi gerektirmektedir. Bu bloğun tasarımında gerekli olan bölme işlemleri çarpma işlemine dönüştürülmüştür.

Bunun sebebi FPGA'larda hazır çarpıcı blokları bulunmasıdır. Bölme bloklarının tasarımı ise hem çok yer kaplar hem de zahmetlidir. Bu sebeple Q kuantalama matrisinin tersi olan 1/Q matrisinin 65536 katı bir RAM'de kaydedilmiştir. Bu katsayılar ile DCT sonuçlarının çarpımının, bölümün sonucunun 65536 katı olduğu görülmektedir. Son olarak ise 16 bit sağa kaydırma işlemi ile istenilen sonuçlara ulaşılır.

Kuantalama bloğunun yapısı Şekil 5.5'te görülmektedir. "DCTTAMAM" bayrağı bir önceki blok DCT sonuçlarını vermeye başladığı zaman aktif olmaktadır. Bu işaretin aktif olmasıyla her bir saat işaretiyle birlikte kuantalama bloğunda da işlemler başlamaktadır. Sayıcı bloğu girişten gelen 64 elemanlık bloğun kaçıncı elemanının geldiğini saymaktadır. Aynı zamanda katsayıları tutmaya yarayan RAM'i adreslemektedir. İlk kuantalama sonucu uygun katsayının DCT katsayısıyla çarpılmasıyla elde edilir ve ilk sonucun çıkışa verildiğini gösteren "TAMAM" bayrağı aktif olur. İlk sonuçlar 119. saat işaretiyle gelmektedir.



Şekil 5.5: İkili RAM yapısı.

5.1.3 Zig-zag tarama bloğu

Zig-zag tarama bloğu kuantalama bloğundan gelen verileri ve tamam bayrağını giriş olarak almaktadır. Çıkışına da yine bir "TAMAM" bayrağı ve zig-zag verisini vermektedir. Ayrıca her bir DC katsayı çıkışa verildiğinde DC bayrağı, her blok sonunda ise EOB bayrağı aktif edilmektedir. Ayrıca bir sonraki kodlama bloğunda DC katsayıların farkı gerekeceğinden, bu blokta DC katsayının kendisi yerine bu fark çıkışa verilmektedir. Zig-zag tarama adresleri Çizelge 5.1'de verilmiştir. Kuantalama matrisi girişe sütun sütun gelmektedir ve bu sütunlar taranırken her bir pikselin kaçıncı adrese denk geldiği bir RAM'de kayıt altına alınmıştır. Kuantalanmış girişler bu bloğun girişine geldikçe RAM'den ilgili zigzag adresi çekilir ve ikinci bir RAM'in bu adresine kaydedilir. Aynı DCT bloğunda olduğu gibi adreslerin kayıtlı olduğu

RAM dışında yine iki tane RAM kullanılır. Bu RAM'lerden birine zig-zag düzenindeki veriler yazılırken diğeri çıkışa verilir. Böylece çıkışların sürekliliği sağlanmış olur. Bu RAM yapısı daha önce verilen Şekil 5.4'tekiyle aynıdır. Sonuçta ilk zig-zag tarama verisi 185. saat işaretinde blok çıkışına gelmektedir.

Çizelge 5.1: Zig-zag tarama adresleri

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	48	57	58	62	63

5.1.4 DC farksal kodlama bloğu

DC kodlama bloğu kategoriye göre çıkış veren bir kod çözücü ile gerçeklenmiştir. Fakat kodların uzunlukları değişken olduğu halde çıkıştaki bit sayısının belirlenmesi gerekmektedir. Bu sebeple blok çıkışında kodla birlikte kodun uzunluğu da verilmelidir. Tablodan seçilecek kod kelimesi ve genlik maksimum 20 bit olmalıdır. 5 bit de kodun uzunluğu için ayrılmıştır. Sonuçta 25 bitlik çıkış dışarı verilir. Çıkışın yapısı kod kelimesi, genlik, 5 bitlik uzunluk şeklindedir.

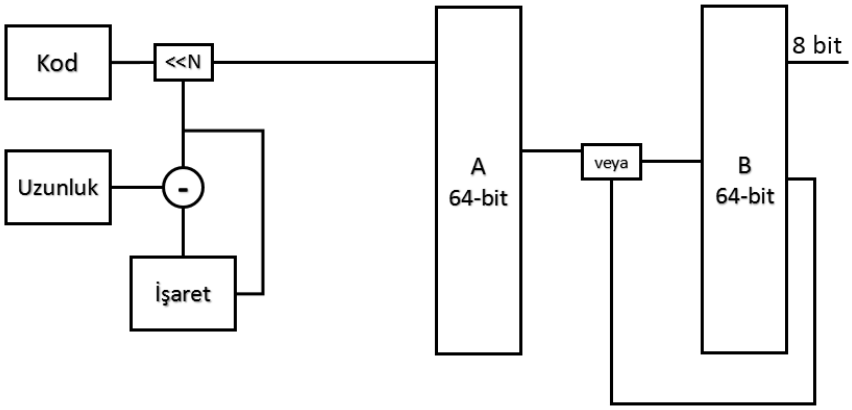
5.1.5 AC dizi uzunluğu kodlama bloğu

AC dizi uzunluğu kodlama bloğu DC fark kodlama bloğuna benzemektedir. Fakat burada kod çözücü koşul olarak kategorinin yanında katsayıdan önce gelen 0 sayısını da dikkate almaktadır. Ayrıca çıkışta kod ve genlik 26 bittir. 5 bit ise bu kodun bit sayısını tanımlar ve kod çözücünün toplamda 31 bitlik çıkışı vardır. Çıkışın yapısı burada da kod kelimesi, genlik, 5 bit uzunluk şeklindedir.

5.1.6 Kod paketleme bloğu

Kodlama bloklarından gelen verilerin uzunlukları sabitlenmiş olsa da bu bloklardan alınması gereken bit sayısı değişiklik göstermektedir. Bu sebeple kod paketleme bloğuna ihtiyaç duyulmuştur. Bu blok, gelen kodlardan ilgili kısmı alarak yan yana dizmek ve birleştirilen bu kodları 8 bitlik paketler halinde çıkışına vermektedir. Bu

işlemleri yapabilmek amacıyla Şekil 5.6'da gösterilen yapı tasarlanmıştır. Bu yapıda 64-bitlik tampon bellek ve bu belleği adreslemek üzere bir işaretçi bulunmaktadır. İlk durumda bellek boştur ve işaretçinin gösterdiği değer 64'tür. Blok girişine kod ve ilgilenilen uzunluk geldiğinde kod, işaretçinin değeri ile kodun uzunluğu arasındaki fark kadar sola kayar ve işaretçinin değeri gelen kodun uzunluğu kadar azalır. Hesaplanan bu değer tampon bellek ile "veya" işlemine girerek girişe gelen kodlar tampon belleğe kaydedilmiş olur. İşaretçinin değeri önceki kayıt işleminde azaltıldığından bir sonraki kod önceki kodun bittiği yerden itibaren kaydedilmiş olur. Soldan sağa doğru 8 bitlik bir alan dolduktan sonra tampon belleğin soldan 8 biti sola kaydırılarak çıkışa verilmeye başlanır. Bu sırada "AL" bayrağı bir periyot boyunca aktif edilir ve işaretçinin değeri 8 arttırılır. Bu işlem kodlar bitene kadar devam eder. Eğer bitişte bellek içerisinde 8 bitten az eleman var ise en az anlamlı bitler 0 ile doldurularak çıkışa verilir. İlk sonuçlar 167. saat işaretinde çıkışa gelmeye başlamaktadır. Böylelikle JPEG kodlayıcı tasarımı tamamlanmış olur.



Şekil 5.6: Kod paketleme bloğu.

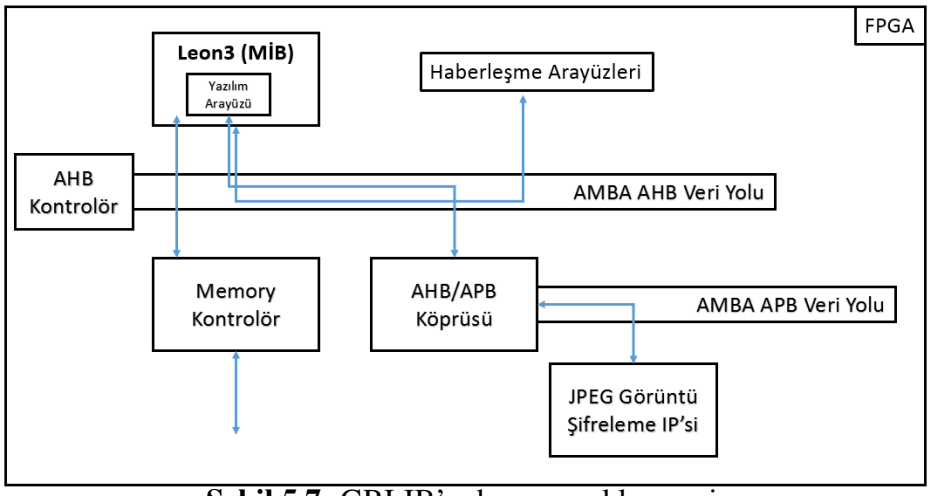
5.2 Şifreleme Bloğunun Tasarımı

Daha önce de anlatıldığı gibi şifreleme bloğu olarak TEA kullanılmıştır. TEA normalde soldan ve sağdan 32'şer bit olmak üzere toplam 64 bitlik girişe ve çıkışa sahiptir. Fakat bu çalışmada giriş ve çıkış bit sayısı değiştirilerek kullanılmıştır. Şifrelemenin zig-zag sıralama bloğundan sonra yapılması bunu gerektirmiştir. Zig-zag sıralama bloğundan çıkan her bir katsayı 12 bittir. Bu sebeple şifreleme bloğunun girişi de 12'nin tam katı olması sebebiyle 72 bit olarak belirlenmiştir. Böylece sağdan ve soldan 36 bit veri girişi olacaktır. Öteleme blokları 4 bit ve 5 bit olarak bırakılmıştır. Bunun yanında toplama ve XOR blokları da yeni bit sayısına göre

ayarlanmıştır. Anahtar uzunluğu 128 bit olarak kullanılmaya devam edilmiştir. Fakat bu sefer 32'şer bitlik 4 anahtar kullanmak yerine anahtarlar 36'şar bit olmuştur. Fakat 128 bit uzunluğundaki anahtarın bütün bitleri anahtar olarak kullanılarak güvenlik sağlanmıştır. Böylece devre girişine 1 DC katsayı, 5 AC katsayı gelebilir.

5.3 Donanımın Leon3 Mikroişlemcisine Eklenmesi

GRLIB, her biri özel arayüzler ve IP'ler içeren çeşitli VHDL kütüphanelerinden oluşmaktadır [21]. Kütüphaneler üreticiye göre IP çekirdeklerini gruplamak için kullanılırlar. GRLIB kütüphanesi kullanıcılar tarafından genişletilebilmektedir. Bu işlem var olan kütüphanelere ek yapılarak ya da sıfırdan kütüphane oluşturularak yapılabilmektedir. Bu çalışmada da tasarlanan JPEG görüntü şifreleme IP'si Leon3 mikroişlemcisinin dahil olduğu GRLIB'e eklenmiştir. Tasarlanan IP'ler AHB yada APB veri yoluna eklenebilir. Bu çalışmada ekleme işlemi APB veri yoluna yapılmıştır. AHB, daha yüksek band genişliği gerektiren IP'ler içindir. Bu çalışmada tasarlanan donanımın APB veri yoluna eklenmesi yeterli görülmüştür. Sistemin genel yapısı Şekil 5.7'de görülmektedir. Görüldüğü üzere JPEG Görüntü Şifreleme IP'si AHB/APB Köprüsü üzerinden AHB veri yoluna, oradan da Leon3 MİB'ye bağlanmaktadır. Burada önemli olan eklenmek istenen IP'nin ABP veri yoluna AMBA protokolüne uygun bir arayüz üzerinden bağlanması gerektiğidir. Bu yüzden bu arayüze ait VHDL kodunun da oluşturulması gerekmektedir. Bu kod protokolün gerektirdiği bayrak ve modül tanımlamaları içermektedir.



Şekil 5.7: GRLIB'e donanım eklenmesi.

Tasarlanan donanımın APB veri yoluna eklenmesi için yapılması gereken işlemler şu şekildedir:

- grlib/lib/opencores/JPEG_IP yoluna eklenmek istenen IP'ye ait yol oluşturulur.
- grlib/lib/opencores/dir.txt dosyasına IP'nin adı yazılır.
- IP.vhd ve IP_amba_interface.vhd oluşturulur.
- grlib/lib/opencores/JPEG_IP yolunda vhdlsyn.txt adında bir dosya oluşturulup tüm VHDL modüllerinin adı yazılır. Verilog modülleri ise vlogsyn.txt içerisine yazılır.
- grlib/lib/grlib/amba içerisindeki devices.vhd dosyasına kendi IP'miz eklenmelidir ve numara atanmalıdır.
- Eklenmek istenen IP (JPEG_IP) en üst modül olan leon3.vhd içerisinde tanımlanmalıdır.

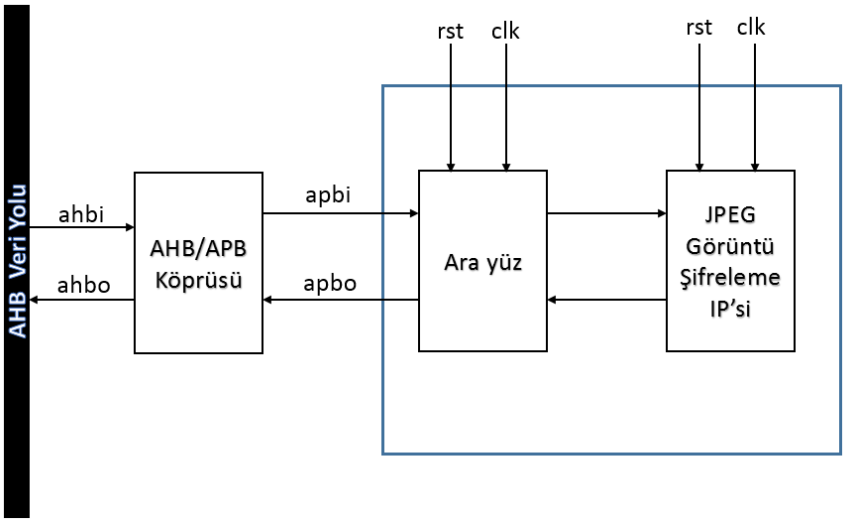
Bu sıralanan işlemler tamamlandıktan sonra leon3mp.vhd kodu sentezlenebilir hale gelir. Bir sonraki aşamada veri alış verişini ve adreslemeleri düzenleyecek olan C kodunun yazılması gerekmektedir. Bu kodun bayrak kısımların yazılışı Şekil 5.8'de verilmiştir.

```
struct JPEG_IP {
volatile int flag_send_data;
volatile int data_transmitted;
volatile int flag_read_data;
volatile int data_received;
};
struct JPEG_IP*IP=(struct JPEG_IP*)0x80000800;

flag_send_data(address 0x8000800)
data_transmitted(address 0x8000804)
flag_read_data(address 0x8000808)
data_received(address 0x800080C)
```

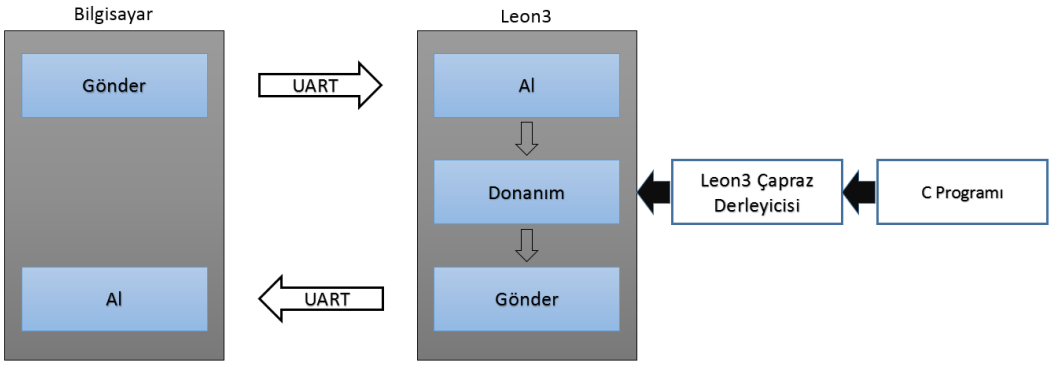
Şekil 5.8: Bayraklar için C kodu.

Sistemin arayüzden gelen ana sinyallerle birlikte gösterimi Şekil 5.9'da verilmiştir. Burada görülen *ahbi*, *ahbo*, *apbi* ve *apbo* sinyalleri AMBA protokolünden gelen seçme sinyalleridir. *i* harfi hangi IP'ye erişeceğini belirtmek için kullanılmaktadır.



Şekil 5.9: Sistemin tüm yapısı.

Sistemin diğer bir anlatımı ise Şekil 5.10'da verilmiştir. GRMON'a bağlanma işlemi ethernet arayüzüyle yapılmıştır.



Şekil 5.10: Tüm sistemin başka bir ifadesi.

6. GERÇEKLEME SONUÇLARI

Tüm donanımın FPGA üzerinde kapladığı alan bilgisi Çizelge 6.1'de verilmiştir. Devrenin maksimum çalışma frekansı 89.208 MHz olarak belirlenmiştir.

Çizelge 6.1: Alan kullanım bilgisi.

Mantık Birimi	**Kullanılan**	**Mevcut**	**Yüzde**
Dilim Saklayıcılar	21730	81920	27
Dilim LUTlar	32906	81920	40
DSP48E	10	320	3
LUT-FF çiftleri	16039	34597	46

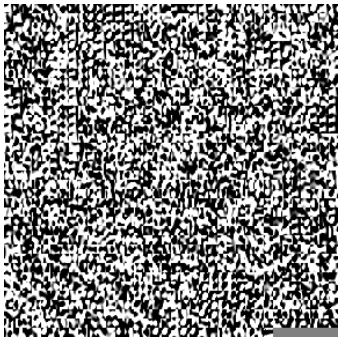
Çizelge 6.2: Görüntülere ait değerler.

LENA	**DC Katsayı Şifreleme**	**DC-AC Katsayı Şifreleme**
PSNR	8.015 dB	7.21 dB
Şifreli Boyut	9 KB	22 KB
MELISSA	**DC Katsayı Şifreleme**	**DC-AC Katsayı Şifreleme**
PSNR	7.56 dB	6.27 dB
Şifreli Boyut	7 KB	21 KB

Daha önce de açıklandığı gibi çalışma kapsamında iki ayrı şifreleme yöntemi uygulanmıştır. Bunlardan ilkinde tüm DCT matrislerindeki bütün DC katsayılar şifrelenmiştir. Diğer yöntemde ise tüm DC katsayıların yanında ilk 5 AC katsayılar da şifrelemeye katılmıştır. Aşağıda verilen şekiller dikkatlice incelendiğinde yalnızca DC katsayıların şifrelendiği yöntemde, orjinal görüntüye (288x288) ait izler görülebilmektedir. Bu sebeple tüm DC katsayılara ek olarak seçilen en düşük frekanslı AC katsayılar da şifrelenmiştir. Bu şekilde yapıldığında görüntünün başarılı bir şekilde şifrelendiği görülebilmektedir. Şifrelenmiş görüntülere ait PSNR değerleri Çizelge 6.2'de verilmiştir. PSNR değerleri beklendiği gibi düşük gelmiştir. Burada ayrıca şifreli JPEG görüntülerin ve orjinal görüntünün boyutları da verilmiştir. Görüldüğü gibi hem DC hem AC katsayıların şifrelendiği durumda sıkıştırma azalmıştır. Fakat yine de yüksek oranda sıkıştırma elde edilmiştir. Gri seviye Lena görüntüsünün orjinal boyutu 83 KB'tır. Şifrelenme uygulanmadan sıkıştırma yapıldığında boyutu 8 KB'a düşmektedir. Bu değerler Melissa görüntüsü için sırasıyla 83 KB ve 6 KB'tır.

(a) Orjinal Lena görüntüsü

(b) DC katsayıların şifrelenmesi

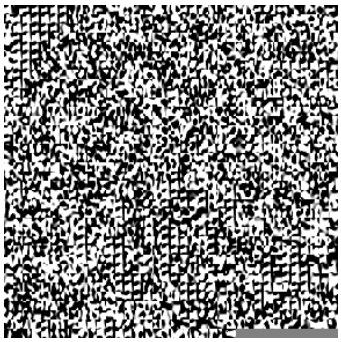


(c) DC-AC katsayıların şifrelenmesi

Şekil 6.1: Lena.

(a) Orjinal Melissa görüntüsü

(b) DC katsayıların şifrelenmesi



(c) DC-AC katsayıların şifrelenmesi

Şekil 6.2: Melissa.

7. SONUÇ

Bu tez kapsamında Leon3 mikroişlemcisi tabanlı JPEG görüntü şifreleme donanımı tasarlanmıştır. Bu amaçla öncelikle JPEG sıkıştırma donanımı tasarlanmış ve TEA şifreleme donanımıyla birleştirilmiştir. Şifreleme için iki yöntem belirlenmiştir. Bunlardan ilki, tüm DCT matrislerinin sadece DC katsayılarının şifrelenmesidir. Diğer yöntem ise tüm DC katsayıların yanında, en düşük frekanslı 5 tane AC katsayıların da şifrelenmesidir. Son aşamada tasarlanan şifreleme ve sıkıştırma donanımı 32 bitlik Leon3 mikroişlemcisine eklenerek sistem tasarımı tamamlanmıştır. Şifrelenen görüntüler incelendiğinde sadece DC katsayıların şifrelenmesinin tüm görüntüyü gizlemek için yeterli olmadığı görülmüştür. İkinci yöntem incelendiğinde ise görüntülerin başarılı bir şekilde gizlenebildiği görülmüştür. Ayrıca tüm AC katsayılar şifrelenmeyerek sıkıştırma oranının fazla azalmaması da sağlanmıştır. Tüm sistem FPGA üzerinde gerçeklenmiştir ve donanım Leon3'e çevresel olarak eklendikten sonra karta bağlanma işlemleri ethernet arayüzü ile yapılmıştır. Sonuçta elde edilen görüntülere ait PSNR değerleri ve donanımın FPGA üzerinde kapladığı alan bilgisi verilmiştir.

Tasarımda güvenliğin sağlanması gözetilmiştir. Bu amaçla görüntü sıkıştırma ve şifreleme iç içe yapılmıştır. Burada yapılan işlem bir görüntüyü önce sıkıştırıp daha sonra şifrelemek değildir. Bunun yerine hem sıkıştırma hem şifreleme blokları ard arda gerçeklenmiş ve şifreleme piksel seviyesinde yapılmıştır. Böylece şifreli görüntünün ele geçirilmesi ve çeşitli kriptanaliz yöntemlerinin uygulanması orjinal görüntünün elde edilmesini sağlamayacaktır. Ayrıca sıkıştırmadan fazla kayıp olmaması amacıyla şifrelenen pikseller az sayıda tutulmaya çalışılmıştır. Daha çok pikselin şifrelenmesinin daha büyük boyutta sonuç görüntüsü vereceği gözlenmiştir.

Gelecek çalışmalarda sıkıştırma standardı aynı kalmak şartıyla farklı şifreleme yöntemlere sisteme uygulanabilir. Aynı zamanda tasarlanan IP'nin bir işlemciye eklenmesiyle elde edilen sistem üzerine araç sürücüsü yazılarak işletim sistemi konulduğu takdirde, sistem birden fazla şifreleme yöntemini barındırabilir hale

gelecektir. Bu durumda farklı şifreleme yöntemleri istenilen görüntüye uygulanabilir hale gelir.

KAYNAKLAR

[1] **Jain, A.** (1981). Image data compression: A review, *Proceedings of the IEEE*, ***69***(3), 349–389.

[2] **B K ShreyamshaKumar ve Chidamber R Patil** (2009). JPEG Image Encryption using Fuzzy PN Sequences, *Signal Image and Video Processing*, ***2010***, 1–9–9.

[3] **Leong, M.**, **Naziri, S. ve Perng, S.** (2013). Image encryption design using FPGA, *Electrical, Electronics and System Engineering (ICEESE), 2013 International Conference on*, s.27–32.

[4] **Yang, B.**, **Zhou, C.Q.**, **Busch, C. ve Niu, X.M.** (2009). Transparent and perceptually enhanced JPEG image encryption, *Digital Signal Processing, 2009 16th International Conference on*, s.1–6.

[5] **Luo, Y.**, **Du, M. ve Liu, D.** (2012). JPEG Image Encryption Algorithm Based on Spatiotemporal Chaos, *Chaos-Fractals Theories and Applications (IWCFTA), 2012 Fifth International Workshop on*, s.191–195.

[6] **Wang, F. ve Bai, S.** (2013). JPEG Image Encryption by Shuffling DCT Coefficients in Defined Block, *Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on*, s.60–63.

[7] **Divya, V.**, **Sudha, S. ve Resmy, V.** (2012). Simple and secure image encryption, *International Journal of Computer Science*, ***9***(6), 286.

[8] **Yen, J.C. ve Guo, J.I.** (1999). A new image encryption algorithm and its VLSI architecture, *Signal Processing Systems, 1999. SiPS 99. 1999 IEEE Workshop on*, s.430–437.

[9] **Balasubramanian, S.** (2005). Image encryption using infinite series convergence, *Systems Engineering, 2005. ICSEng 2005. 18th International Conference on*, s.257–262.

[10] **Gaisler**, `http://www.gaisler.com/doc/leon3_product_sheet.pdf`, alındığı tarih: 08.04.2015.

[11] **Gaisler**, `http://www.gaisler.com/products/grlib/grlib.pdf`, alındığı tarih: 08.04.2015.

[12] **Gaisler**, `http://www.gaisler.com/doc/grmon2.pdf`, alındığı tarih: 08.04.2015.

[13] **Gaisler**, `http://www.gaisler.com/doc/bcc.pdf`, alındığı tarih: 08.04.2015.

[14] **Abdelhalim, M.**, **El-Mahallawy, M.**, **Ayyad, M. ve Elhennawy, A.** (2011). Implementation of a modified lightweight cryptographic TEA algorithm in RFID system, *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, s.509–513.

[15] **Bagbaba, A. ve Ors, B.** (2013). Implementation of a secure Near Field Communication system on a FPGA, *Electrical and Electronics Engineering (ELECO), 2013 8th International Conference on*, s.621–625.

[16] **In, J.**, **Shirani, S. ve Kossentini, F.** (1998). JPEG compliant efficient progressive image coding, *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, cilt 5, s.2633–2636 vol.5.

[17] **Bhaskaran, V. ve Konstantinides, K.**, (1997). Image and Video Compression Standarts: Algorithms and Architectures, Kluver Academic Publishers, Norwell, Massachusetts, s. 74.

[18] **Wallace, G.** (1992). The JPEG still picture compression standard, *IEEE Transactions on Consumer Electronics*, ***38***(1).

[19] **Xilinx**, `http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf`, alındığı tarih: 21.04.2015.

[20] **Agostini, L.**, **Silva, I. ve Bampi, S.** (2001). Pipelined fast 2D DCT architecture for JPEG image compression, *Integrated Circuits and Systems Design, 2001, 14th Symposium on.*, s.226–231.

[21] **AMBA**, `http://www-micro.deis.unibo.it/~magagni/amba_99.pdf`, alındığı tarih: 21.04.2015.

ÖZGEÇMİŞ

Ad Soyad: Ahmet Çağrı BAĞBABA

Doğum Yeri ve Tarihi: Ankara-12.07.1989

Adres: Zekeriyaköy Mah. Gök Sk. Sarıyer Evleri Sitesi Yasemin Apt. Daire 12 Sarıyer/İSTANBUL

E-Posta: bagbaba@itu.edu.tr

Lisans: İstanbul Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği (2013)

Y. Lisans: İstanbul Teknik Üniversitesi Elektronik Mühendisliği (2015)

Mesleki Deneyim ve Ödüller:
Araştırma Görevlisi- İTÜ (12/2013-devam)
ASIC Tasarım ve Doğrulama Mühendisi- Anka Mikroelektronik Sistemler (7/2013-12/2013)

Yayın ve Patent Listesi:

Ahmet Çağrı Bağbaba, Buse Ustaoglu, İnan Erdem, Gökhan Işık, Berna Örs, "Leon3 Tabanlı SoPC Tasarımı ve Uygulama Gerçeklenmesi", GOMSIS2014.
Bagbaba, A.C.; Ors, B.; Erozan, A.T., "Image filtering processor and its applications," Signal Processing and Communications Applications Conference (SIU), 2014 22nd , vol., no., pp.2011,2014, 23-25 April 2014
Bagbaba, A.C.; Ors, B., "Implementation of a secure Near Field Communication system on a FPGA," Electrical and Electronics Engineering (ELECO), 2013 8th International Conference on , vol., no., pp.621,625, 28-30 Nov. 2013

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- **Ahmet Çağrı Bağbaba**, Buse Ustaoglu, İnan Erdem, Gökhan Işık, Berna Örs, "Leon3 Tabanlı SoPC Tasarımı ve Uygulama Gerçeklenmesi", GOMSIS2014.