İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GELİŞMİŞ ŞİFRELEME STANDARDI BLOK ŞİFRELEME ALGORİTMASININ BİR MİKROİŞLEMCİ ÜZERİNDE GERÇEKLEMESİNE YAN KANAL SALDIRISI

YÜKSEK LİSANS TEZİ

Gizem Çisem KULA

Anabilim Dalı : Elektronik ve Haberleşme Mühendisliği

Programı : Elektronik Mühendisliği

Tez Danışmanı: Yrd. Doç. Dr. Sıddıka Berna ÖRS YALÇIN

HAZİRAN 2009

Anneme,

ÖNSÖZ

Tez çalışmalarım sırasında yol gösteren, yardımlarını eksik etmeyen danışman hocam Yrd. Doç. Dr. Sıddıka Berna Örs Yalçın'a teşekkürü borç bilirim.

Ayrıca tüm çalışmalarım boyunca yanımda olan, beni bu çalışmaya teşvik eden ve manevi desteğini üzerimden eksik etmeyen annem Nesrin Tüzel'e teşekkürlerimi sunarım.

Haziran 2009

Gizem Çisem Kula

(Elektronik Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

AES	**:** Advanced Encryption Standard
DES	**:** Data Encryption Standard
NIST	**:** National Institute of Standarts and Technology
FIPS	**:** Federal Information Processing Standards
ASM	**:** Assembly
ACC	**:** Accumulator
R	**:** Register
DPTR	**:** Data Pointer Register
PC	**:** Program Counter
SP	**:** Stack Pointer
CMOS	**:** Complementary Metal Oxide Semiconductor
IC	: Integrated Circuit
RAM	**:** Random Access Memory
ROM	**:** Read Only Memory
EPROM	**:** Erasable Programmable Read Only Memory
ADC	**:** Analog-Digital Converter
VB	**:** Visual Basic
DOMT	**:** Distance Of Mean Test

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

GELİŞMİŞ ŞİFRELEME STANDARDI BLOK ŞİFRELEME ALGORİTMASININ BİR MİKROİŞLEMCİ ÜZERİNDE GERÇEKLEMESİNE YAN KANAL SALDIRISI

ÖZET

Gelişmiş şifreleme standardı (AES) bir veriyi şifrelemek ve şifrelenmiş veriyi çözmek için kullanılan bir simetrik blok şifreleme algoritmasıdır. Algoritma, Kasım 2001'de Amerikan ulusal standartlar ve teknoloji enstitüsü (National Institute of Standards and Technology, NIST) tarafından federal bilgi işleme standardı (FIPS) olarak yayınlanmıştır [1].

Şifreleme algoritmalarında verinin gizliliği esastır. Bu sebeple şifreleme ve şifre çözme işlemleri arasında verinin başkası tarafından elde edilememesi büyük önem taşır. Şifrelemede kullanılan anahtarın bir kısmını veya tamamını elde etmek için algoritmaların gerçeklemelerinin verdiği yan bilgilerden faydalanılarak yapılan saldırılara Yan-Kanal saldırıları denir. Bu saldırılar şifreleme işlemi sırasında gerçekleşen ve dışarıdan gözlemlenebilen fiziksel veya elektriksel bilgilerin ölçümü ile gerçekleştirilir. Yan kanal saldırısı ile anahtar, algoritmanın zayıflığından yararlanma yerine algoritmanın gerçeklemesinin çalışması sırasındaki elektriksel ortamın sağladığı bilgilerden yararlanılarak bulunur. Bu sebeple yan kanal saldırıları için şifreleme yapılan sisteme fizikel erişim gerekmektedir [2]. Yan kanal bilgileri, ses, güç veya elektromanyetik alan bilgisi olabilir. Yan kanal saldırılarının bir çeşiti olan güç tüketimi saldırıları, şifreleme işlemi sırasında harcanan gücün ölçülmesi ve bu güç ile kullanılan şifreleme anahtarı arasında bir ilişki kurma yoluyla yapılır. Güç tüketimi saldırılarında şifreleme işlemini gerçekleştiren elektronik sistemin içerisindeki transistörlerin 0-1 geçişleri sayılarak bu yolla sistemin çektiği akım bilgisi tahmin edilir. Bu bilgi sistemin harcadığı güç bilgisinin benzetimidir. Sonuç olarak şifreleme algoritması gerçeklenirken gerçeklemenin donanım üzerinde tüketeceği güç, verinin güvenliği açısından önemlidir.

Bu tezde AES blok şifreleme algoritması bir mikroişlemci üzerinde gerçeklenecektir. İlk olarak algoritma C programlama dili ile gerçeklenecektir. Ardından bu gerçekleme, seçilen bir mikroişlemci üzerinde uygulanacak ve gerçeklemenin güç tüketim analizi yapılacaktır. Bu analizi yapabilmek için, gerçeklemenin donanımdaki işleyişi bir simulasyon programı yardımı ile elde edilip, bu işleyişten yola çıkarak işleyiş sırasında tüketilen gücün hesabı için ayrı bir simulasyon programı oluşturulacaktır. Bu simulasyon programı, mikroişlemcinin şifreleme işlemi sırasında işlediği komutların analizini ve bu komutların işlenmesi sırasında mikroişlemci üzerinde bulunan saklayıcıların ve mikroişlemci hafızasının durum değişimlerini analiz etmek yöntemiyle sistemin tükettiği gücü tespit edecektir. Güç tüketimi bilgisi tespit edildikten sonra bu bilgiler ışığında sistemin güç tüketimi grafikleri elde edilerek algoritma aşamaları grafikler üzerinde belirlenecektir. Son olarak yan-kanal saldırıları yapılarak gerçeklemenin ne kadar güvenilir olduğu tespit edilecektir.

SIDE-CHANNEL ATTACKS ON REALIZATION OF ADVANCED ENCRYPTION STANDARD BLOCK CIPHER ALGORITHM ON A MICROPROCESSOR

SUMMARY

Advanced encryption standart (AES) is a symmetric block cipher which is used to encrypt data and decrypt an encrypted data. Algorithm is published as federal information processing standart by National Institute of Standards and Technology on November 2001 [1].

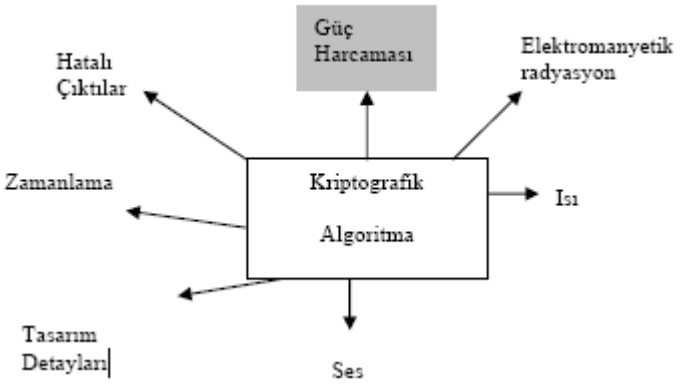
Security is the main factor for encryption algorithms. Therefore, keeping the data secure during encryption is major. Attacks for obtaining the data by obtaining parameters used by encryption is called side-channel attacks. These attacks use measurements of electrical or physhical effects which occur during encryption process [2]. Side-channel information can be sound, power or electromagnetic field information. Power analysis which is a type of side-channel attacks makes use of varying power consumption by the hardware during computation. For power consume attacks, measurement of 0-1 switches of transistors in device is used. Hence, power consumption of a realization of an encryption algorithm matters for the security of information. So, the power consumption during encryption of device is important for the security of information.

In this thesis, AES algrithm will be realized on a 8051 microprocessor. Firstly, algorith will be realized by using C programming language. Then this realization will be implemented on a chosen 8051 microprocessor and power consumption of this realization on chosen hardware will be measured. To analyze power consumption, steps of the process of the realization willl be obtained by a simulation program and then another simulation program will be developed to measure the power consumption of process by using obtained process steps. This second simulation program will process commands that microprocessor processes during encryption and it will watch the register and memory updates to measure the power consumption. Then power consume graphics will be obtainerd and steps of algorithm will be shown on these graphics. Finally, security of this realization will be determined by using side-channel attacks.

1. GİRİŞ

Günümüzde haberleşmedeki veri güvenliği, bir başka deyişle bilginin gönderici taraftan alıcı tarafa güvenli bir şekilde ulaştırılması önemli bir sorun haline gelmiştir. Bilginin güvenliği, bilginin başkası tarafından dinlenmesi, bilginin değiştirilmesi, kimlik taklidi gibi tehditlerin ortadan kaldırılması amacı için geliştirilen bilim dalı kriptolojidir [3]. Kriptoloji, kriptografi ve kriptanaliz olmak üzere iki dala ayrılır. Kriptografi şifreli metin üretme, kriptanaliz şifreyi kırmak için kullanılır. Kriptografi ile geliştirilen şifreleme algoritmaları haberleşmedeki güvenlik sorununa bir çözüm getirmiştir. Ancak haberleşme sisteminde kullanılan şifreleme algoritmasının saldırılarla kırılamaması yani güvenliği sağlayan şifreleme anahtarının herhangi bir yolla elde edilmemesi gerekmektedir.

Kriptografik sisteme yapılan saldırılar aktif ve pasif olmak üzere iki grupta toplanmaktadır. Aktif saldırılarda sistemin yapısına doğrudan müdahele edilerek ölçümler yapılmakta veya oluşturulan hatalardan faydalanılmaktadır. Pasif saldırılar, yan-kanal saldırıları olarak da adlandırılır. Bu saldırılarda algoritmayı gerçekleyen sistemin ürettiği bazı istemsiz çıkışların gizli bilgiyle ilişkili olmasından faydalanılır [4]. Bu fiziksel ve elektriksel çıkışlara yan kanal bilgileri denir. Yan kanal bilgileri zaman bilgisi, güç tüketimi bilgisi, elektromanyetik kaçak bilgisi ve ses bilgisidir (bkz. Şekil 1.1) [5]. Yan kanal saldırılarında, sistemden elde edilen bu bilgiler ile kullanılan şifreleme anahtarı arasında bir ilişki kurmak yoluyla anahtar elde edilir.



Şekil 1.1 : Yan kanal bilgileri

1.1 Tezin Kapsamı

Daha önce bir çok güvenlik sistemlerinde şifreleme standardı olarak kullanılan Veri kodlama standardı (Data Encryption Standard (DES)) güvenlik problemlerine sahiptir [6,7]. Bu sebeple bu standart yerine Amerikan Ulusal Standartlar ve Teknoloji Enstitüsü (National Institute of Standards and Technology (NIST)) tarafından Kasım 2001'de federal bilgi işleme standardı (Federal Information Processing Standards (FIPS)) olarak yayınlanan Gelişmiş kodlama standardı (Advanced Encryption Standard (AES)) kullanılmaya başlanmıştır [1, 6].

Her sistem işleyişi sırasında güç harcar. Yan kanal saldırılarından biri olan güç tüketimi saldırıları, sistemin şifreleme işlemi sırasında harcadığı güçten yola çıkarak gizlenmeye çalışılan veriyi elde etmeye yönelik saldırılardır [5]. Dolayısı ile tasarım yapılırken, tasarımın güvenli bir tasarım olması için, devrenin çalışma esnasında tüketeceği güç göz önünde bulundurulmalıdır. Bu çalışmada AES algoritması bir miktroişlemci üzerine gerçeklenmiş ve bu gerçeklemeye yan kanal saldırısı yapılarak gerçeklemenin güvenilirliği incelenmiştir.

1.2 Tezin Konuya Katkısı

Şimdiye dek çeşitli programlama dilleri ile çeşitli donanımlar üzerine gerçeklenmiş bir çok AES gerçeklemesi bulunmaktadır. Bu çalışmanın amacı mikroişlemci üzerinde bir AES gerçeklemesi yaparak ve bu gerçeklemeye yan kanal saldırıları uygulamaktır.

2. MİKROİŞLEMCİ VE SİMULASYON

2.1 Mikroişlemci

2.1.1 Mikroişlemci nedir?

Mikroişlemci ana işlem biriminin (CPU) fonksiyonlarını tek bir yarı iletken tümleşik devrede (IC) birleştiren programlanabilir bir sayısal elektronik bileşendir [8]. Gerek yaptığı işlmelerin mikrosaniyeler mertebesinde olması gerek içerisinde barındırdığı devrelerin mikron boyutlarında olması sebebiyle bu adı almıştır. Bir veya daha çok mikroişlemci, tipik olarak bilgisayar sisteminde, gömülü sistemde ya da mobil cihazda ana işlem birimi olarak görev yapmaktadır. Mikroişlemciler veri işleme ve sinyal iyileştirmelerinde yaygın olarak kullanılırlar [9].

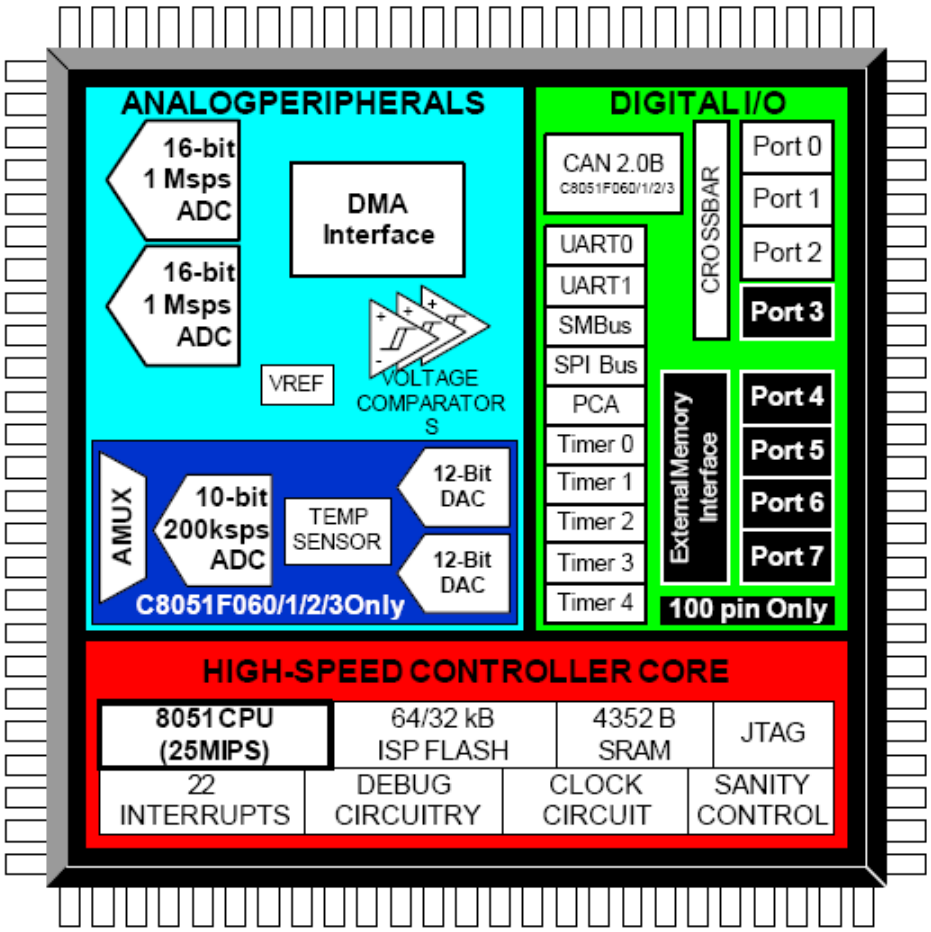
2.1.2 8051 tipi mikroişlemciler

8051 tipi mikroişlemciler ilk olarak Intel tarafından 1980 yılında üretilmiştir. Eski bir ürün olmasına rağmen, hem kendisi, hem de yapısı temel alınarak üretilmiş diğer işlemciler bugün geniş bir kullanım alanına sahiptir [10]. Genel olarak 8051 ailesine ait mikroişlemciler bünyelerinde programlanabilir bağlantı kapıları, silinir programlanabilir salt okunur bellek (EPROM), analogtan sayısala çevirici (ADC) ve rastgele erişimli hafıza (RAM) bulundururlar [9]. Bu mikroişlemcilerin bazı teknik özellikleri aşağıdaki gibidir [10]:

- ACC ve B belleklerine ek olarak 8 adet R saklayıcıları (R0,R1,..,R7).
- 16 bitlik veriye erişimi sağlayan işaretçi (DPTR).
- Program sayacı (PC), yığın göstergesi (SP).
- Üzerinde var olan iç belleklere ek olarak dış bellekler ekleyebilme.
- 128 adet bit düzeyinde değişken.
- RAM, ROM, bazı modellerinde EPROM.

2.1.3 C8051F060

Gerçekleme için kullanılmak üzere seçilen mikroişlemci Silicon Laboratories firmasının ürettiği 8051 tipindeki C8051F060 kodlu mikroişlemcidir. Bu mikroişlemci 16-bitlik veri işlemektedir. 4352 byte büyüklüğünde bir dahili hafızaya ve 59 adet giriş-çıkış bacağına sahiptir. Mikroişlemcinin iç yapısı Şekil 2.1'de görülmektedir [11].



Şekil 2.1 : C8051F060 yapısı

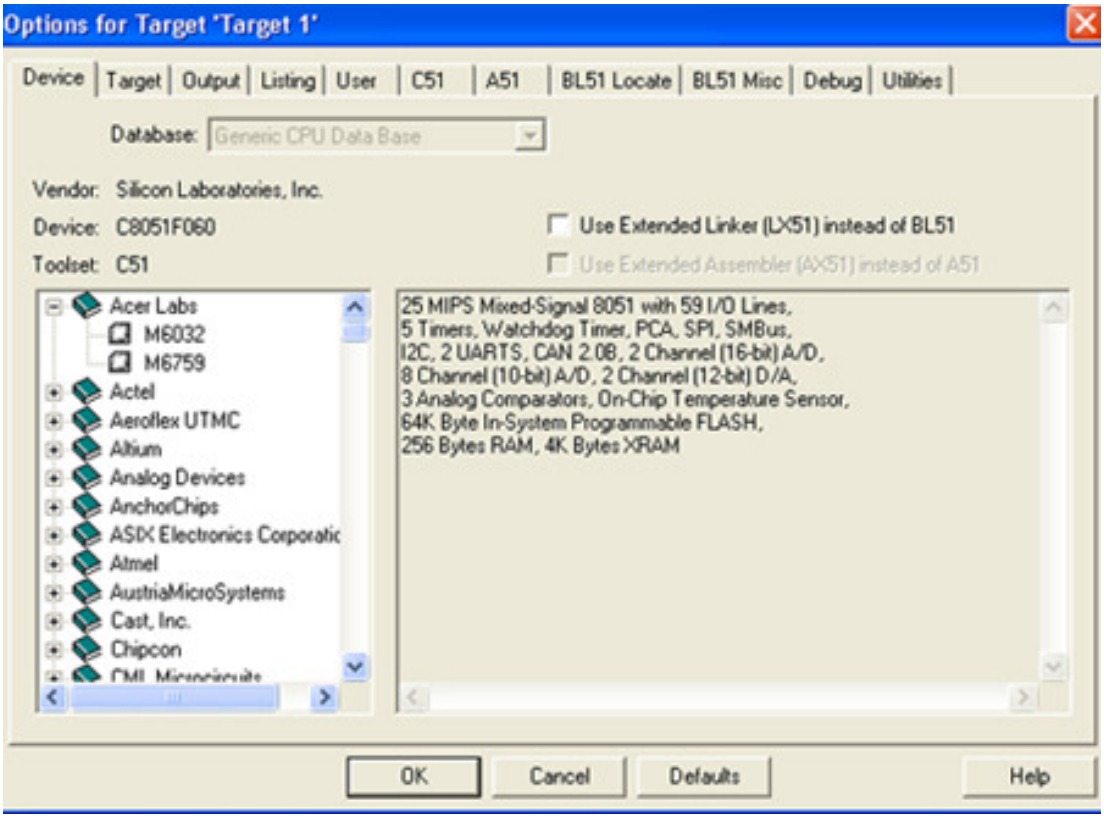
Şifreleme algoritmalarında algoritmanın güvenilirliği donanımın tükettiği güç tüketimi ile doğrudan ilişkilidir. Algoritmanın donanım üzerindeki işleyişi sırasında harcadığı gücün analizi, algoritma ile korunmaya çalışılan veri hakkında dolaylı bir bilgi verir. Güç tüketimi ise donanım içeriğinin her adımdaki durum değişimi ile ilişkilidir. Bir mikroişlemci için bir adım bir saat darbesine karşılık gelir. Durum değişimi ise mikroişlemci üzerindeki kaydedicilerin içeriklerinin değişimidir. Dolayısı ile mikroişlemcinin hangi işlemleri kaç saat darbesinde işlediğinin bilgisi güç tüketim analizinin temelini oluşturur. Gerçekleme için seçilen mikroişlemcinin gerçeklediği işlemler ve bu işlemlerin süreleri ekte verilmiştir.

2.2 Simulasyon

Simulasyon için Keil firmasının ürettiği bir programlama aracı olan C51 geliştirme aracı kullanılmıştır. Keil firması, merkezi Amerika Birleşik Devletleri ve Almanya'da olan ve mikrokontrolörler için C derleyiciler, makro çeviriciler, hata ayıklayıcıları, simulatörler ve entegre devreler üreten bir firmadır [12]. Kullanılan C51 programı, gömülü programları yazmaya, derlemeye ve hata ayıklamaya olanak sağlayan bir entegre geliştirme ortamıdır [13]. Geniş bir donanım veritabanına ve çeşitli derleyicilere sahip olduğundan tüm 8051 mikroişlemciler üzerinde yazılım gerçeklenebilir. Kullanımı kolay editörü ve debugging e verdiği olanak ile yazılan bir programın işleyişini takip etmeyi kolaylaştırır. Tüm kaynak kodu analiz ederek ortak kod blokları için ayrı fonksiyonlar oluşturur ve kodun takibini kolaylaştırır. Programın işleyişi sırasına seçili donanımın içeriğindeki değişimleri göstererek anlaşılması kolay bir simulasyon sağlar.

2.2.1 Keil C51 ile proje oluşturma ve derleme

Keil C51 ile proje oluştrmada ilk adım kullanılacak donanımın seçilmesidir. Çünkü seçili donanımın özelliklerine göre derleyici kullanılır. Daha sonra kaynak C kodu oluşturulan projenin içine eklenerek proje derlenir. Bu açamada kaynak kodda kullanılan değişken tiplerinin seçilen donanıma uygun olması büyük önem taşır. Değişken boyutlarının seçili donanıma uyumsuz olması durumunda kaynak kodta hata bulunmasa bile program derlenemez. Bu yüzden kaynak kodun donanım özelliklerine uygun olacak şekilde düzenlenmesi gerekir. Keil C51 ile simulasyon yapılacak donanımın seçilimi Şekil 2.2'de verilmiştir.



Şekil 2.2 : Keil C51 donanım seçimi

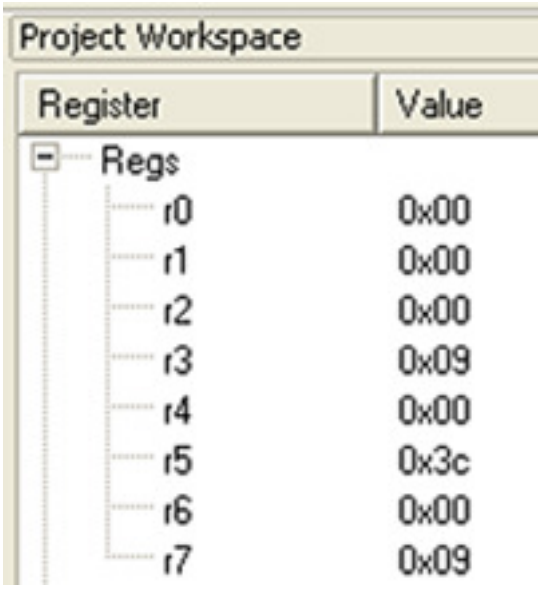
2.2.2 Keil C51 ile hata ayıklama

Derleme işleminden sonra adım adım kodun işlenmesi ve hataların ayıklanması aşamasına geçilir. Keil C51 ortamı bu aşamada kaynak C kodundan üretilen ve donanım üzerinde işlenecek olan Assembly kodunu üretir (bkz: Şekil 2.3).

```
 tempmatrix[0][1]=tempmatrix[0][2];
3D3     MOV       DPTR,#0x03D3
        MOVX      A,@DPTR
        MOV       R5,A
        INC       DPTR
        MOVX      A,@DPTR
3D1     MOV       DPTR,#0x03D1
        XCH       A,R5
        MOVX      @DPTR,A
        INC       DPTR
        MOV       A,R5
        MOVX      @DPTR,A
 tempmatrix[0][2]=tempmatrix[0][3];
3D5     MOV       DPTR,#0x03D5
```

Şekil 2.3 : Kaynak C kodundan ASM kodunun üretilmesi

Assembly kodu mikroişlemci üzerinde çalışacak olan koddur. Aynı C kaynak kodundan üretilen assembly kodları seçili donanımın özelliklerine göre farklılık gösterebilir. Keil-C51 aracının sağladığı en önemli imkanlardan biri de her satır assembly kodunun donanım üzerindeki işleyişini, donanım içerisindeki her bileşenin kodun işlenmesiyle nasıl değişime uğradığını göstermesidir. Kullanıcı mikroişlemci üzerindeki kaydedicilerin ve hafıza üzerindeki adreslerdeki değerleri işlemin her aşamasında Şekil 2.4'de görüldüğü gibi takip edebilir.



Şekil 2.4 : Mikroişlemci değişimlerinin izlenmesi

3. AES GERÇEKLEMESİ

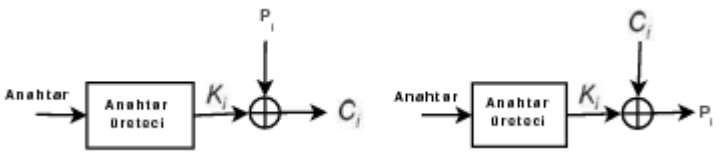
3.1 Şifreleme Ve Şifreleme Sistemleri

Şifreleme, verilerin herkese açık ortamlarda iletilirken istenmeyen kişiler tarafından kullanılmasını veya değiştirilmesini önlemek amaçlı kullanılır. Şifreleme işlemi önceleri sadece askeri amaçlı kullanılmaktaydı. Ancak günümüzde gelişen teknoloji ile veri güvenliği önemli bir sorun haline gelmiştir. Bu yüzden kriptografinin alanı genişlemiş ve şifreleme, veri haberleşmesinde yaygın olarak kullanılır hale gelmiştir. Şifreleme işleminde düz metin, şifreleme işlemine tabi tutulur ve bu işlem sonucunda elde edilen şifrelenmiş veri alıcı tarafa yollanır. Alıcı taraf şifrelenmiş veriyi şifre çözme işlemi ile düz metin haline çevirir (bkz. Şekil 3.1) [3]. Bu yolla haberleşme kanalında esas veri anlaşılmaz bir halde gönderilir ve verinin başkaları tarafından elde edilmesi ve değiştirilmesi engellenmiş olur.

Şekil 3.1 : Şifreleme ve şifre çözme işlemleri diyagramı

Şifreleme sistemleri işleyişi bakımından temel olarak iki ana başlıkta incelenebilir. Bu sistemler Dizi Şifreleme ve Blok Şifreleme sistemleridir.

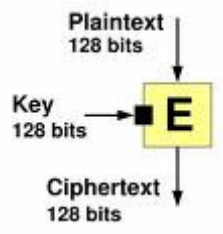
Dizi şifreleme sistemlerinde şifreleme işleminde anahtar üretilir ve verinin her bir biti ile anahtarın her bir biti exor'lanır. Verinin her bir biti sırayla şifrelenir. Alıcı taraf şifrelenmiş veriye yine anahtar ile e-xor işlemi uygular ve düz veriyi elde eder. Dizi şifreleme sistemlerinin işleyişinin blok diyagramı Şekil 3.2'de verilmiştir [14].



Şekil 3.2 : Dizi şifreleme

Blok şifreleme sistemlerinde ise veri belirli uzunluktaki veri blokları halinde şifrelenir. Şifreleme için karmaşık işlemler ve algoritmalar kullanılır. Blok

şifrelemeye örnek olarak 128 bitlik veri bloğunun aynı uzunluktaki anahtar ile şifrelenerek yine aynı uzunlukta şifrelenmiş veri elde edilmesinin blok diyagramı Şekil 3.3'de verilmiştir.



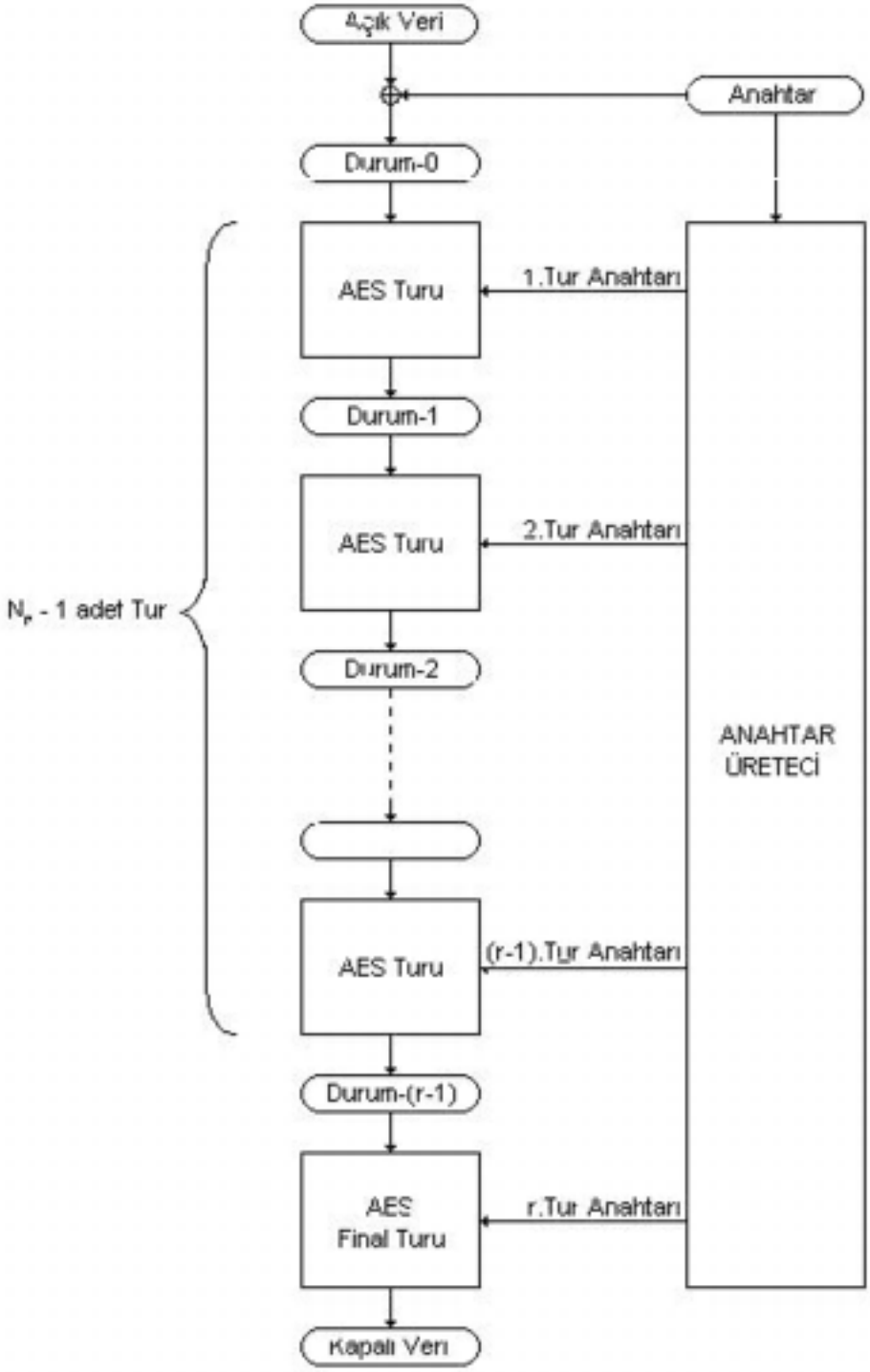
Şekil 3.3 : Blok şifreleme

3.2 AES

Gelişmiş şifreleme standardı (AES) veriyi 128 bitlik paçalar halinde şifreleyen bir blok şifreleme algoritmasıdır. Kullandığı anahtar uzunluğuna göre AES-128, AES-192 ve AES-256 olmak üzere üç çeşittir [1]. Bu tezde AES'in 128 bit uzunluğunda anahtar kullandığı çeşidi üzerine çalışılmıştır.

AES-126 10 çevrimdir. İlk olarak 128 bitlik anahtar on çevrimde farklı şekliyle kullanılması amacıyla genişletilir [1]. Daha sonra Tur Anahtarını Ekleme adımı gerçekleşir. Bu aşamadan sonra 10 çevrim gerçekleşir. Her çevrim sırasıyla Bayt Değiştirme, Satırları kaydırma, Sütunları karıştırma ve Tur Anahtarını Ekleme işlemlerinden oluşur. Son çevirm olan onuncu çevrimde Sütunları Karıştırma adımı uygulanmaz [15].

AES blok şifreleme algoritmasının blok diyagramı Şekil 3.4'de verilmiştir [14].



Şekil 3.4 : AES blok diyagramı

Adımlarda yapılan işlemler aşağıda kısaca verilmiştir.

3.2.1 Bayt değiştirme

İlk olarak 128 bitlik veri 8'er bitlik 16 parçaya ayrılır ve 4x4 boyutundaki durum matrisi oluşturulur (bkz. Şekil 3.5) [1]. Tüm işlemler bu durum matrisi üzerinden gerçekleşmektedir. Bayt değiştirme adımında her 8 bitlik parçaya matematiksel bir dönüşüm uygulanır. Bu dönüşüm iki aşamada gerçekleşir. İlk olarak indirgeme polinomu $P(x) = x^8 + x^4 + x^3 + x + 1$ kullanılarak çarpmaya göre ters alma işlemi uygulanır. Burdan elde edilen sonuç bir geçiş matrisi ile çarpılarak sabit bir matris ile toplanır [1]. Bu işlemlerin sonucunda bayt değiştirme adımı sonucu elde edilir. Bu işlemlerin sonuçları Şekil 3.6'da tablo olarak verilmiştir. Bu dönüşümler 8 bitlik 16 veriye seri olarak tekrarlandığında 128 bitlik veri bu adımdan geçmiş olur.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Şekil 3.5 : Durum matrisi

3.2.2 Satırları kaydırma

Bu adımda Bayt Değiştirme işleminden elde edilen veri yine 8'er bitlik 16 parçaya ayrılır ve 4x4 boyutunda bir matris haline getirilir. Martisin ilk satırı sabit bırakılarak ikinci, üçüncü ve son satırlar sırasıyla bir, iki ve üç kere sola kaydırılır ve bu işlemler sonucu yeni bir 128 bitlik veri elde edilir. Bu işlem blok diyagram halinde Şekil 3.7'de gösterilmektedir.

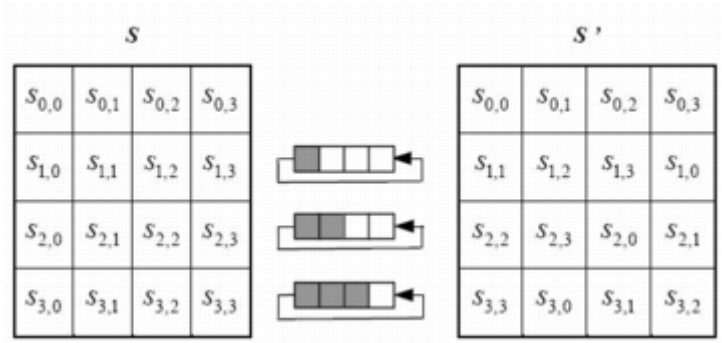
3.2.3 Sütunları karıştırma

Süt Satırları Kaydırma adımında oluşan 128 bitlik verinin 8'er bitlik 16 parçasının herbiri belirli işlemlere tabi tutularak yeni bir 128 bitlik veri elde edilir (bkz: Şekil 3.8). Bu adımda işlemler durum matrisindeki her bir sütun üzerinde bağımsız olarak gerçekleşir. Her bir sütun bir polinom olarak düşünülerek $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ polinomu ile modülo $x^4 + 1$'de çarpma işlemi

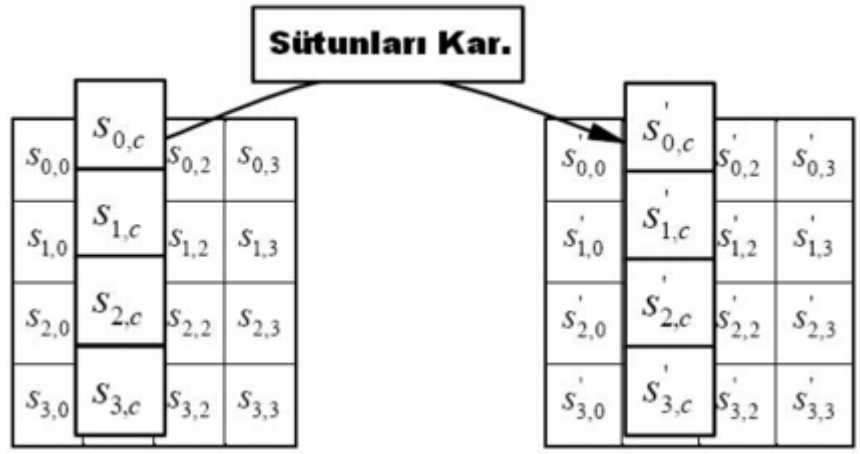
gerçekleştirilir [14]. Sütunları kaydırma işleminin diyagramı Şekil 3.8'de verilmiştir [1].

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7c	77	7b	f2	6b	6f	C5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
A	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
B	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
C	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
D	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
E	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
F	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Şekil 3.6 : S-kutusu çıkışları



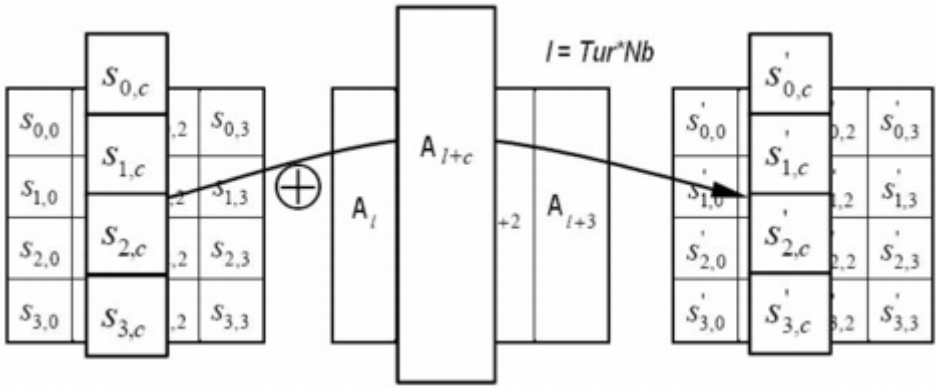
Şekil 3.7 : Satırları kaydırma



Şekil 3.8 : Sütunları karıştırma

3.2.4 Tur anahtarını ekleme

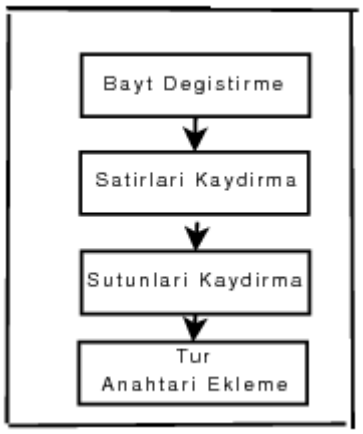
Bu aşamada bir önceki işlemin sonucunda elde edilen 128 bitlik durum matrisi ile genişletilen anahtarın o çevrimle ilgili bölümü olan 128 bitlik anahtar dizisi exorlanır. Bu aşama Şekil 3.9'da verilmiştir [1].



Şekil 3.9 : Tur anahtarını ekleme

3.2.5 AES çevrimi

AES algoritması kendini tekrarlayan bir yapıdadır. Kullanılan anahtar boyutuna göre bu tekrarların bir başka deyişle çevrimlerin sayısı değişir. Bu çevrim sayıları 128 bitlik anahtar kullanımı için 10, 192 bitlik anahtar kullanımı için 12 ve 256 bitlik anahtar kullanımı için 14'tür. AES çevrimi sırasıyla yukarıda anlatılan dört adımdan oluşur. Bir çevrime ait blok diyagram Şekil 3.10'da verilmiştir [14].



Şekil 3.10 : AES çevrimi blok diyagramı

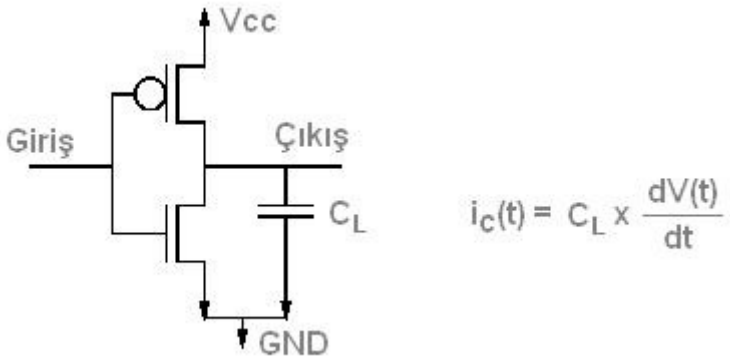
3.3 Gerçekleme

Bu çalışmada AES gerçeklemesi C programlama dili ile yapılmıştır. Çevrimdeki her adım için ayrı fonksiyonlar yazılmış, ana fonksiyonda bu alt fonksiyonlar kullanılmıştır. 10 çevrim için döngülerden yararlanılmış, Bayt Değiştirme adımı için ise giriş ve çıkışların bulunduğu bir tablo hazırlanarak bu tablodan yararlanılarak kodlama yapılmıştır.

Yazılımın doğruluğu literatürde bulunan test verilerinden yararlanılarak kontrol edilmiştir. Gerçeklemenin kaynak kodu ektedir.

C ile gerçekleme aşamasından sonra Keil C51 programında, yazılan C kodu C8051F060 tipi mikroişlemci üzerinde derlenmiştir. Program yardımıyla C kodundan, mikroişlemci üzerinde işleyecek olan makine kodu elde edilmiştir. Ancak mikroişlemcinin güç tüketimini ölçmek için, makine kodlarının işlemci üzerine çalışırken mikroişlemci içerisinde bulunan kaydedici ve hafızanın durumundaki bit değişimlerinin bilgisi gerekmektedir. Bu değişimlerden kasıt, işlemcinin her saat darbesinde kaydedici ve hafızasında bulunan verinin değişmesidir. Her saat darbesinde mikroişlemci bir işlemi gerçekleşir ve kaydedici ve hafızadaki veriler değişikliğe uğrar.

Bir elektronik devrenin harcadığı güçte en önemli paya sahip olan kısım dinamik güç harcamasıdır. Dinamik güç tüketimi tarnsistörlerin konum değiştirdiği andaki güç tüketimidir. Dinamik güç tüketimi tahmini için ihtiyaç duyulan bilgi, her saat darbesine hangi kaydedicilerde ve hafızadaki hangi adreslerde verilerin ne kadar değiştiği bilgisidir. Veriler bit bazında işlenir. Dolayısıyla verinin değişimi bit değişimidir. Bir bitin sıfırdan bire ya da birden sıfıra değişmesi, mikroişlemci üzerinde bulunan transistörlerin kaynaktan akım geçmesi anlamına gelir çünkü transistörün sürdüğü yük kapasitesinin akımı üzerindeki gerilim değişiminin hızlanmasıyla artar (Şekil 3.10) [4]. Transistörün akım çekmesi ise mikroişlemcinin güç harcaması demektir. Dolayısıyla dinamik güç tüketimini yahmin etmek için elde edilen makina kodunun çalışması ile gerçekleşen mikroişlemcideki bit değişimleri tahmin edilmelidir. Bu amaçla Bölüm 4 de anlatılacak olan ayrı bir simulasyon programı hazırlanmıştır.



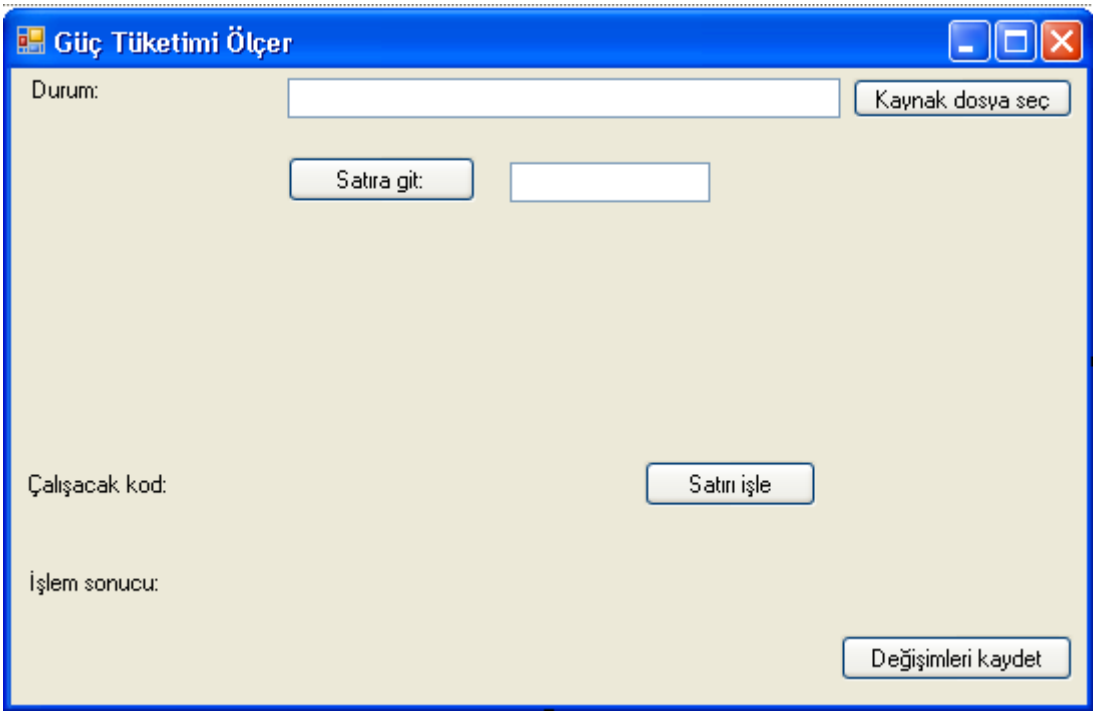
Şekil 3.11 : CMOS transistörler ile gerçeklenmiş bir evirici

4. GÜÇ TÜKETİMİ TAHMİM PROGRAMI

Keil C51 programı ile mikroişlemci üzerinde çalışacak olan makina kodu elde edildikten sonra bu kodların nasıl işlediğini, her bir makina kodunun mikroişlemci üzerinde kaç saat darbesinde ne gibi işlemler yaptığını analiz etmek amacıyla bir güç tüketimi tahmin programı hazırlanmıştır. Bu program Vb.net ile yazılmıştır. Program bir Excel dosyasından kaynak makina kodunu okuyarak ve her bir komutu işleyerek komutun gerektirdiği işlemleri yapmak mantığında tasarlanmıştır. Programın hazırlanma aşamaları aşağıdaki gibidir:

- Her bir ASM kodunun yaptığı işlemlerin eldesi.
- Mikroişlemci üzerindeki kaydedici ve hafızanın programla birleştirilmesi.
- Veri değişimi ve aritmetik işlemler yapan ASM kodlarının işlenmesi.
- Program dallanma işlemleri yapan ASM kodlarının işlenmesinin programa dahil edilmesi.
- Program ile kaynak kod dosyasının seçimi.
- Mikroişlemci durumunun program arayüzde gösterilimi
- İşlenecek kodların program arayüzünde gösterilimi
- İstenilen satıra kadar işlemi yapabilme seçeneği.
- Bit değişimlerini dosyaya kaydetme seçeneği.

Her bir ASM kodu, mikroişlemci üzerinde farklı sürelerde çalşır ve farklı işlemler yapar. Program, okuduğu kodların herbiri ile ilgili kaydedici ve bellek adreslerindeki verileri gerektiği şekilde günceller. Programın arayüzü Şekil 4.1'deki gibidir.



Şekil 4.1 : Program arayüzü

4.1 Programın İşleyişi

Mikroişlemci üzerinde çalışan komutlar ile çeşitli işlemler gerçekleşir. Komut, bir kaydedici ya da hafızada bulunan veriyi güncelleyebilir, başka bir alt fonksiyonu çalıştırabilir ya da işlemin başka bir kod satırından devam etmesini sağlayabilir [16,17,18]. Komut çeşitleri ve her bir komutun kaç saat darbesinde çalıştığının bilgisi ekte verilmiştir. Bu bilgi ölçüm programının hazırlanma aşamasında kullanılan temel bilgidir.

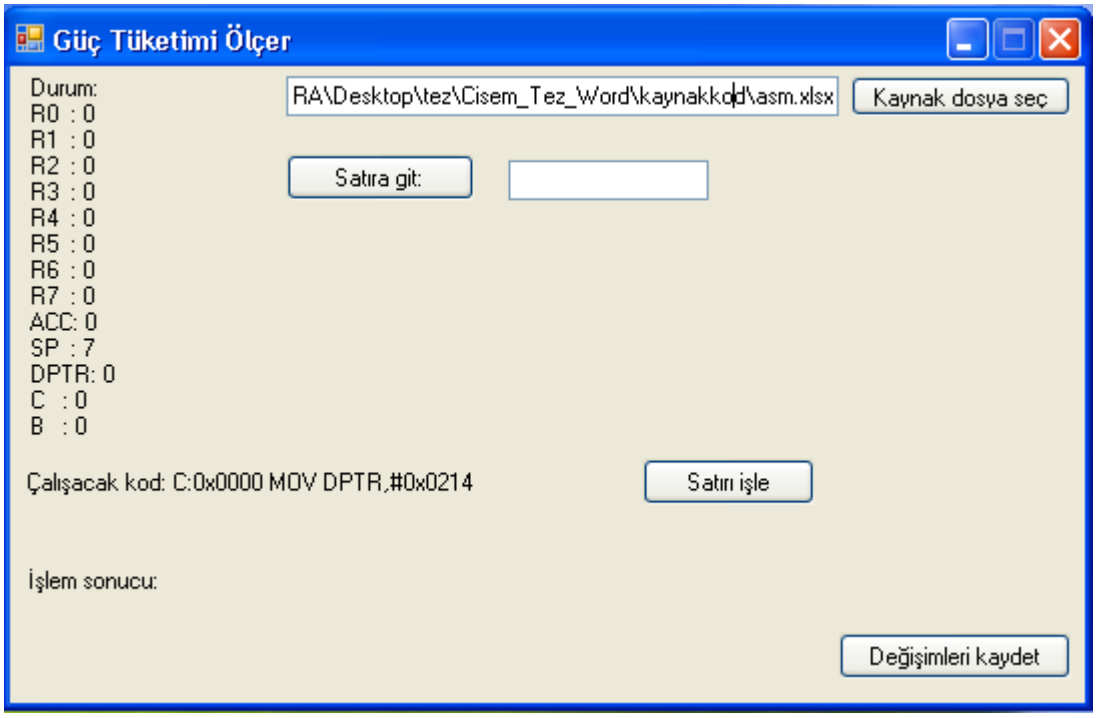
Keil C51 programından elde edilen kaynak ASM kodu, programın işleyişinde okunacak dosyadır. Bu dosya bir Excel dosyasıdır ve içerisinde işlenecek ASM kodunun sırasıyla kod sırası, işlenecek komut ve komutun işleneceği değerler bilgisini üç kolon halinde bulundurur (bkz: Şekil 4.2). Program başlatıldığında ilk olarak "Kaynak dosya seçimi" seçeneği yardımıyla bu kaynak dosyanın adresi verilir. Programın başlatılması ile birlikte verilen adresteki kaynak dosya yüklenir ve ilk satır ile işlenecek kod bilgisi kullanıcıya sunulur. Bu aşamada kodlar işlenmeye hazırdır, mikroişlemci üzerindeki kaydediciler ilk durumlarındadır ve işlenecek ilk kod satırı bilgisi kullanıcıya gösterilmektedir (bkz: Şekil 4.3).

asm.xlsx

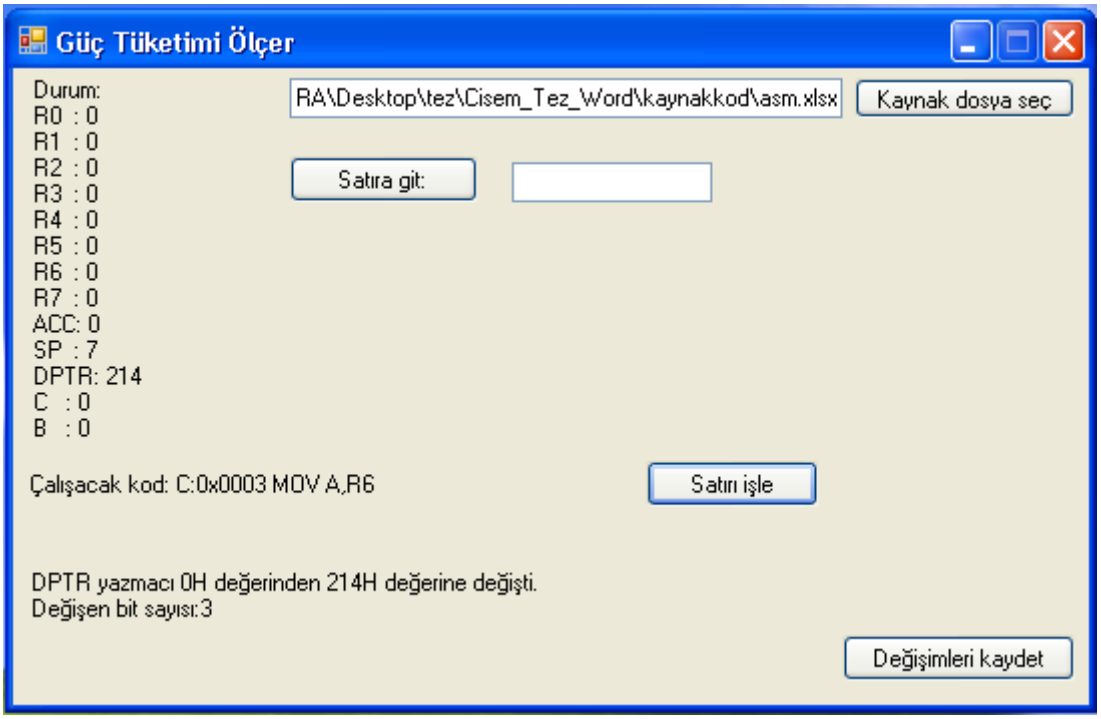
	A	B	C
1	**Satır**	**ASM kodu**	**Değer**
2	C:0x0000	MOV	DPTR,#0x0214
3	C:0x0003	MOV	A,R6
4	C:0x0004	MOVX	@DPTR,A
5	C:0x0005	INC	DPTR
6	C:0x0006	MOV	A,R7
7	C:0x0007	MOVX	@DPTR,A

Şekil 4.2 : Programın kullandığı kayna excel dosyası formatı

Satırı işle komutu ile birlikte satırdaki ASM kodu gereği işlemler yapılır [19,20]. İşlemlerin sonucu kaydedicilerin durumları güncellenir, işlem sonucunda değişen bit sayısı bilgisi ve sıradaki çalışacak kod satırının bilgisi kullanıcıya verilir (bkz:Şekil 4.4).

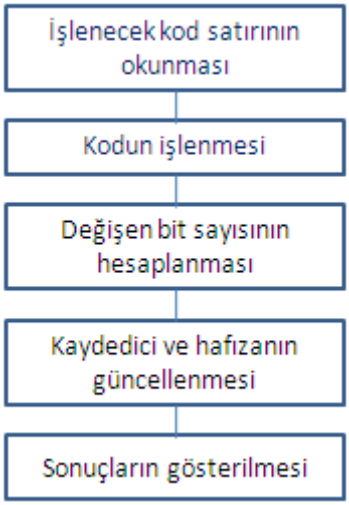


Şekil 4.3 : Dosyanın belirlenmesi



Şekil 4.4 : Kodun işlenmesi

Her bir satır kodu işlenmesi blok diyagram haline şekil 4.5'de verilmiştir.



Şekil 4.5 : Ölçüm programı komut işleyiş blok diyagramı

Kullanıcı her bir satır kodu kendi çalıştırmak yerine "Satıra Git" komutu ile bir kod satırı belirterek işlemin istediği bir aşamaya kadar otomatik çalışmasını sağlayabilir. "Satıra git" seçeneği kullanıcıya zaman kazandırma amaçlı yapılan ek bir kolaylıktır.

4.2 Güç Tüketimi Tahmini

Günümüzde tamamlayıcı metal oksitli yarı-iletken (CMOS) tranzistörler, elektronik tümdevrelerin gerçeklenmesinde yaygın olarak kullanılmaktadır. Bir CMOS tranzistörün güç tüketimindeki en büyük pay dinamik güç tüketimine (konum değiştirme anlarındaki güç tüketimi) bağlıdır. Çıkışın sabit kaldığı anlardaki güç tüketimi, çıkışın değiştiği anlara oranla çok daha azdır. Dolayısıyla gerçeklemenin donanım üzerindeki güç tüketimi ölçümü için gerekli bilgi, her saat darbesinde 1-0 ya da 0-1 geçişi yapan transistör sayısı, bir başka deyişle mikroişlemci içerisindeki kaydedici ve hafızasında bulunan verilerdeki değişen bit sayısıdır.

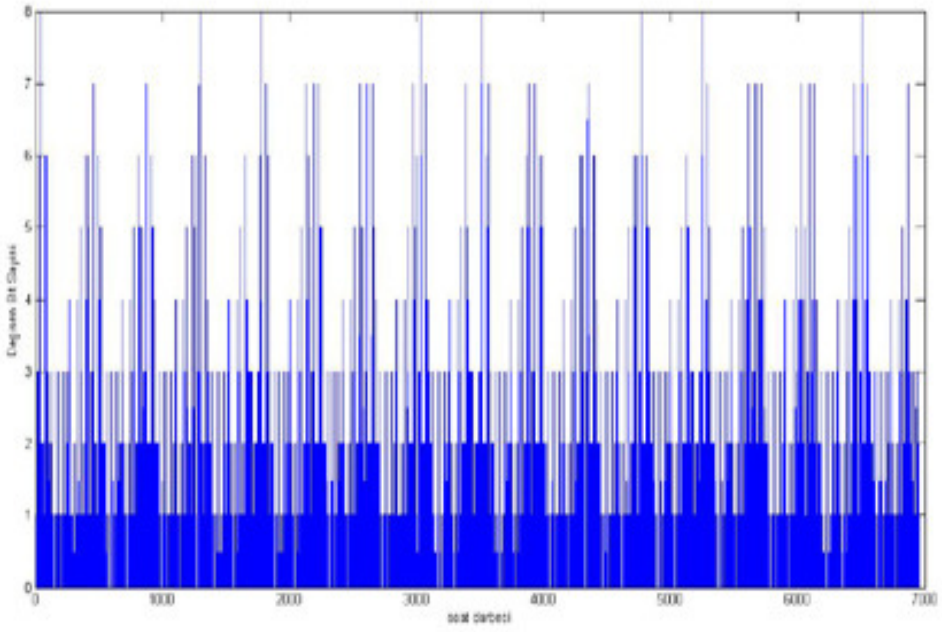
Ölçüm programında değişen bit sayılarının ölçümü bölüm 4.1'de anlatıldığı gibi mümkündür. Programda bu bit değişimlerini kaydetmek için "Değişimleri kaydet" seçeneği mevcuttur. Bu seçenek işlemin herhangi bir aşamasında başlatılıp durdurulabilir. Bu sayede işlemlerin istenilen kısımlarındaki bit değişimleri kısmı ayrı ayrı kaydedilebilir. "Değişimleri kaydet" seçeneği seçildiğinde program her satırı işledikten sonra hesapladığı bit değişi bilgisini bir yazı dosyasına kaydeder. Bu aşamada komutların kaç saat darbesinde işlendiği bilgisi önem taşır. Her komut sonrasında değişen bit sayısı, o komutun çalışmasının sürdüğü saat darbesi bilgisi ile birlikte kaydedilir. Kaydetme işlemi kullanıcı tarafından değişimleri durdurma işlemi yapılana kadar devam eder.

5. GERÇEKLEMENİN GÜÇ TÜKETİMİ

Bu bölümde AES gerçeklemesinin hazırlanan ölçüm programı ile elde edilen güç tüketimi sonuçları algoritma aşamaları ayrı ayrı belirtilerek verilmiştir..

5.1 Bayt Değiştirme

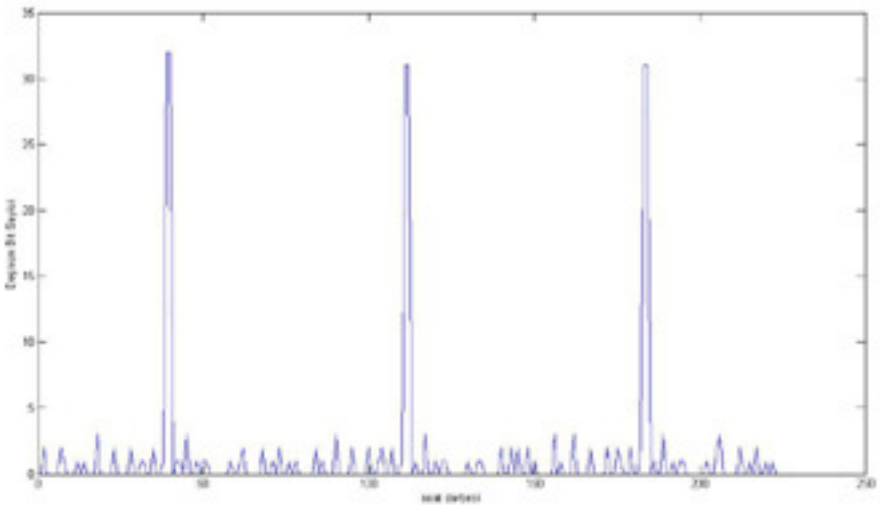
Bayt değiştirme adımı 128 bitlik veriyi 16 parçalık seri çalışma halinde değiştirir. Dolayısı ile bu adımın 16 kez tekrarlayan bir rutin halinde güç tüketmesi beklenir. Yapılan güç tükletimi analizinde de beklenildiği gibi 16 seri parçadan oluşan bir Bayt Değiştirme işlemi net bir şekilde gözlenmiştir (bkz: Şekil 5.1).



Şekil 5.1 : Bayt değiştirme güç tüketimi grafiği

5.2 Satırları Kaydırma

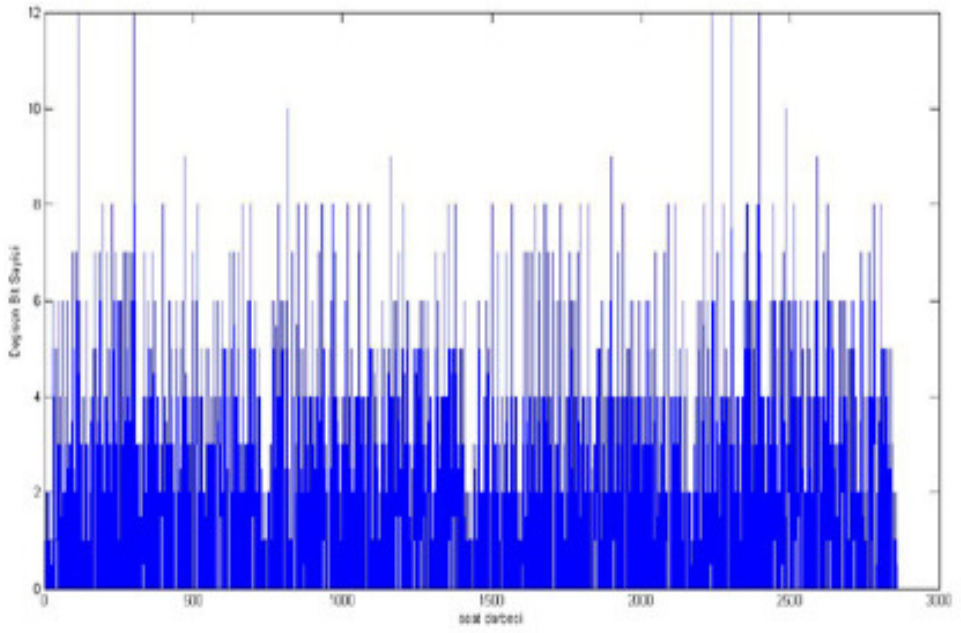
Satırları kaydırma adımında 128 bitlik veri 4 satıra ayrılır, ilk satır hariç diğar üç satıra kaydırma işlemi uygulanır. Yani bu işlemin 3 aşamadan gerçekleşir. Analizler sonucu Şekil 5.2'de görüleceği gibi 3 parçadan oluşan bir güç tüketimi grafiği elde edilmiştir.



Şekil 5.2 : Satırları kaydırma güç tüketimi grafiği

5.3 Sütunları Karıştırma

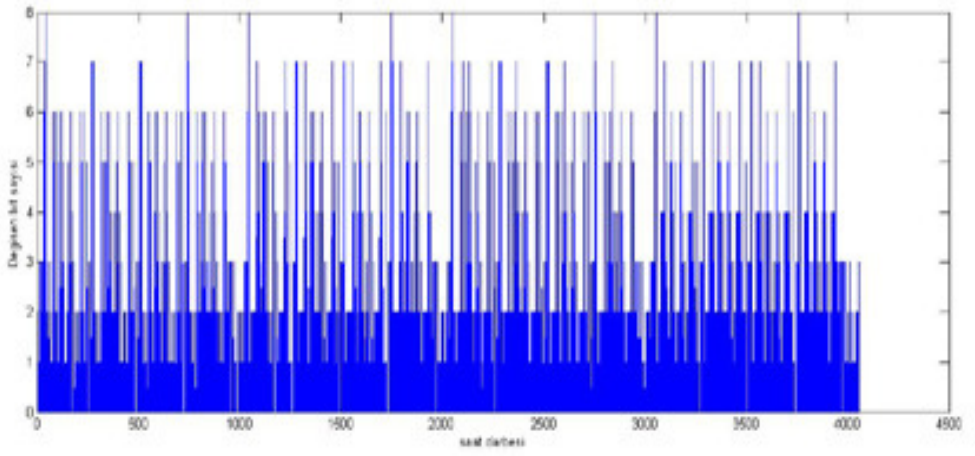
Sütunları karıştırma işlemininde yapılan işlemler 4 seri adım halindedir. 16 adet 8 bitlik veri 4 kolon haline 4 adımda işleme tabi tutulur. Şekil 5.2'deki Sütunları karıştırma adımının güç tüketim grafiğinde de seri 4 blok halinde güç tüketimi eğrileri görülmektedir.



Şekil 5.3 : Sütunları karıştırma güç tüketimi grafiği

5.4 Tur Anahtarını Ekleme

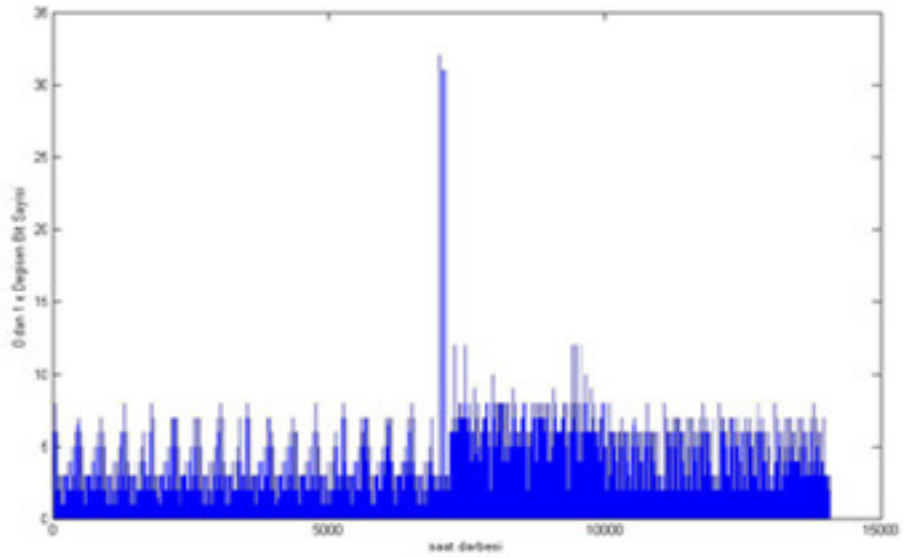
Bu adımda 128 bitlik veri ile 128 bitlik anahtar exor işlemi yapılır. Burada seri 128 işlem gerçekleşmektedir. Elde edilen güç tüketimi eğrisi de bu 128 yaklaşık eş güç tüketimli adımı gösterir niteliktedir (bkz: Şekil 5.4).



Şekil 5.4 : Tur anahtarını ekleme güç tüketimi grafiği

5.5 Çevrim

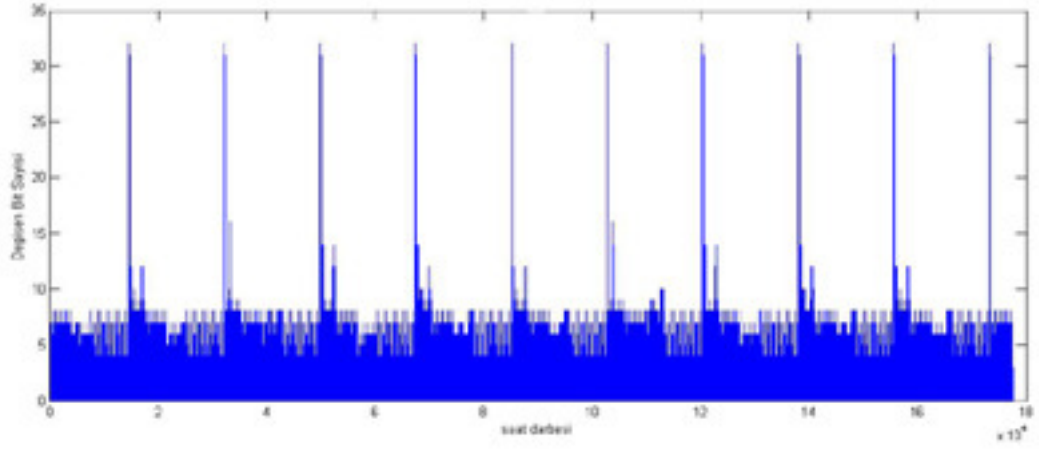
Tam bir çevrim son çevrim hariç sırasıyla yukarıda anlatılan 4 adımdan oluşur. Bir çevrimin güç tüketimi ölçüldüğünde tüm adımlar net bir şekilde güç tüketim eğrisinde görülebilmektedir (bkz: Şekil: 5.5).



Şekil 5.5 : Tüm çevrim güç tüketimi grafiği.

5.6 Tüm Şifreleme

Tüm şifreleme Tur Anahtarını Ekleme adımı ve sonrasındaki 10 çevrimden oluşur. Tüm çevrimin güç tüketim eğrisi Şekil 5.6'da verilmiştir. Toplam 10 çevrim şekilde gözlenebilmektedir.



Şekil 5.6 : Tüm şifreleme güç tüketimi grafiği

6. YAN KANAL ANALİZİ

6.1 Analiz Yöntemleri

6.1.1 Korelasyon

Korelasyon olasılık kuramı ve istatistikte iki bağımsız değişlen arasındaki doğrusal ilişkinin yönünü ve gücünü belirtir. Genel istatistiksel kullanımda korelasyon, bağımsızlık durumundan ne kadar uzaklaşıldığını gösterir [21].

Korelasyon katsayısı, bağımsız değişkenler arasındaki ilişkinin yönü ve büyüklüğünü belirten katsayıdır. Bu katsayı -1 ile +1 arasında bir değer alır. Pozitif değerler direk yönlü doğrusal ilişkiyi, negatif değerler ise ters yönlü bir doğrusal ilişkiyi belirtir.

Farklı durumlar ve sistemler için tanımlanmış farklı korelasyon katsayıları vardır. Bunlardan en çok bilinen ve kullanılanı 'Pearson Korelasyon Katsayısı'dır. E(X) X'in ortalama değerini ve Var(X) X'in sapmasını göstermek üzere, X ve Y değişken kümelerinin korelasyonu, C(X,Y), aşağıdaki denklemle bulunur [5].

$$C(X,Y) = \frac{E(X \times Y) - E(X) \times E(Y)}{\sqrt{Var(X) \times Var(Y)}} \qquad -1 \le C(X,Y) \le 1 \qquad \textbf{(6.1)}$$

Korelasyon katsayısı 0 ise sözkonusu değişkenler arasında doğrusal bir ilişki yoktur. Katsayı +1 veya -1'e ne kadar yakınsa ilişkinin doğrusallığı o kadar güçlüdür.

6.1.2 Ortalamaların farkı testi

Ortalamaların farkı testi (Distance of mean test, DOMT) aşağıdaki adımlardan oluşur:

1. Kriptografik algoritma N adet rastgele düz metin için çalıştırılır.
2. N adet düz metinin herbiri için saldırı noktasındaki değerler tahmin edilir.
3. Tahmin edilen değerler bir ayrıştıma fonksiyonuna göre iki kümeye ayırlır.
4. Her iki küme için de ortalama değerler hesaplanır.
5. Elde edilen iki ortalama değerin farkı alınır.

6. Yukarıdaki işlemler her anahtar olasılığı için yapılarak ortalama değerlerin farkının en yüksek olduğu anahtar seçeneği kullanılan anahtarın bilgisini verir.

6.2 Saldırılar

Yan kanal saldırılarıyla şifreyi elde etmek için öncelikle yeterli miktarda düz metin örneği gereklidir. Bu düz metinler 128 bit uzunluğunda ve 16'lık düzendedir. Düz metinler için örnek bir dizi aşağıdadır.

3243F6A8885A308D313198A2E0370734

1DEAD0FA92416224ED3BB685BFD20A58

3243F6A8885A308D313198A2E0370734

1DEAD0FA92416224ED3BB685BFD20A58

……

Düz metinlerin hazırlanmasından sonra saldırı yapılacak nokta belirlenir. Yeterli sayıda düz metin, her anahtar olasılığı ile şifrelenir ve seçilen saldırı noktasındaki durumları hesaplanarak "kullanılan mesaj sayısı x anahtar olasılığı" boyutunda bir tahmin matrisi (T) elde edilir. Bu tahmin matrisinin elemanları saldırılan noktadaki 1 değerli bitlerin sayısıdır. Tahmin matrisinin formatı aşağıda verilmiştir.

$$T = Mn \begin{matrix} \\ \\ \\ \end{matrix} \overset{\begin{matrix} K_0 & & K_{255} \end{matrix}}{\begin{bmatrix} 3 & \cdots & 5 \\ \vdots & \ddots & \vdots \\ 7 & \cdots & 2 \end{bmatrix}}$$

Tahmin matrisi elde edildikten sonra aynı düz metinler kullanılan anahtar ile şifrelenerek aynı saldırı noktasındaki güç tüketiminin benzetimi elde edilir ve "kullanılan mesaj sayısı x 1" boyutunda benzetim matrisi (B) elde edilir. Bu matris aşağıdaki formattadır.

$$B = Mn \overset{\begin{matrix} K \end{matrix}}{\begin{bmatrix} 2 \\ 7 \\ 3 \\ \vdots \\ 5 \end{bmatrix}}$$

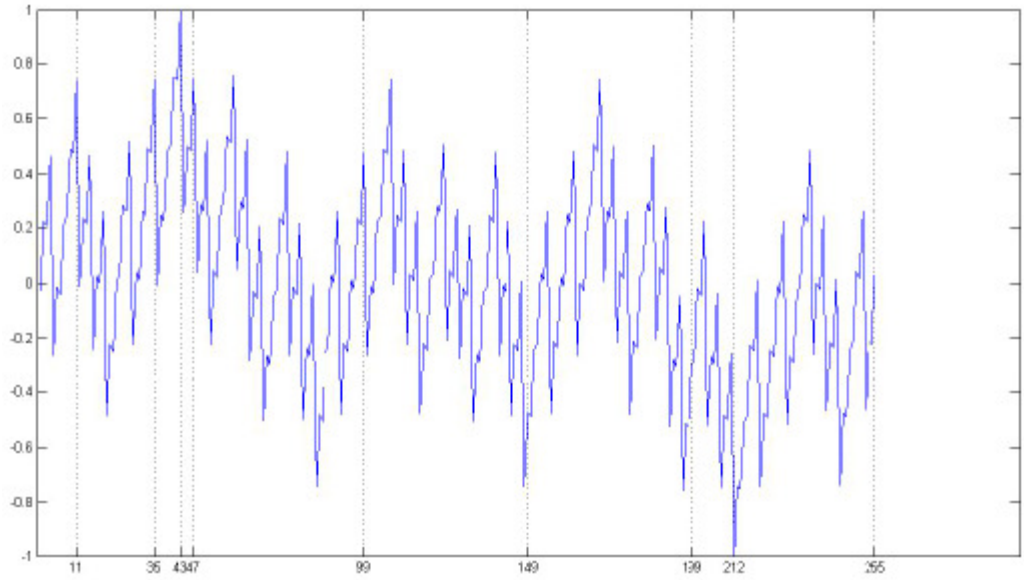
Son olarak T tahmin matrisinin her sütunu ile B benzetim matrsinin korelsayonu hesaplanır. Korelasyonun yüksek olduğu T matrisi sütunu kullanılan anahtarı verir.

Bu çalışmada AES algoritmasının mikroişlemci üzerinde gerçeklemesine farksal güç tüketimi analizi yapılmıştır. Kullanılan atak noktaları aşağıdaki gibidir:

- İlk Tur anahtarını ekleme işleminde düz metinin ilk baytı ile anahtarın ilk baytının exor işleminden geçirilme sonucu.
- İlk Bayt değiştirme işleminde durum matrisinin ilk elemanı olan ilk baytın S kutusundan çıkışı.

6.2.1 Tur anahtarını ekleme çıkışına saldırı

Tur anahtarını ekleme çıkışına saldırıda ilk bayt üzerindeki işlemin çıkışına saldırı yapıldığı için bu işlem sırasında kullanılan anahtar parçasının uzunluğu 1 bayttır. Dolayısı ile elde edilmeye çalışılan anahtar bilgisi 128 bit uzunluğundaki anahtarın ilk 8 bitidir. Bu noktada 8 bitlik anahtarın alabileceği değerler $2^8 = 256$ adettir. Saldırıda ilk olarak çıkış noktasındaki değerler 2000 adet düz metin ve 256 adet anahtar olasılığı için hesaplanarak tahmin matrisi T oluşturulmuştur. Daha sonra gerçeklemede aynı noktadaki güç tüketimi benzetim sonuçları anahtarın ilk bayt değeri için "2B" kullanılarak tespit edilmiş ve B benzetim matrisi elde edilmiştir. T matrisinin her kolonu ile B matrisi arasıdaki korelsayon değerleri hesaplandığında elde edilen sonuç aşağıdaki gibidir (bkz: Şekil 6.1).



Şekil 6.1 : Tur anahtarını ekleme saldırısı sonuçları

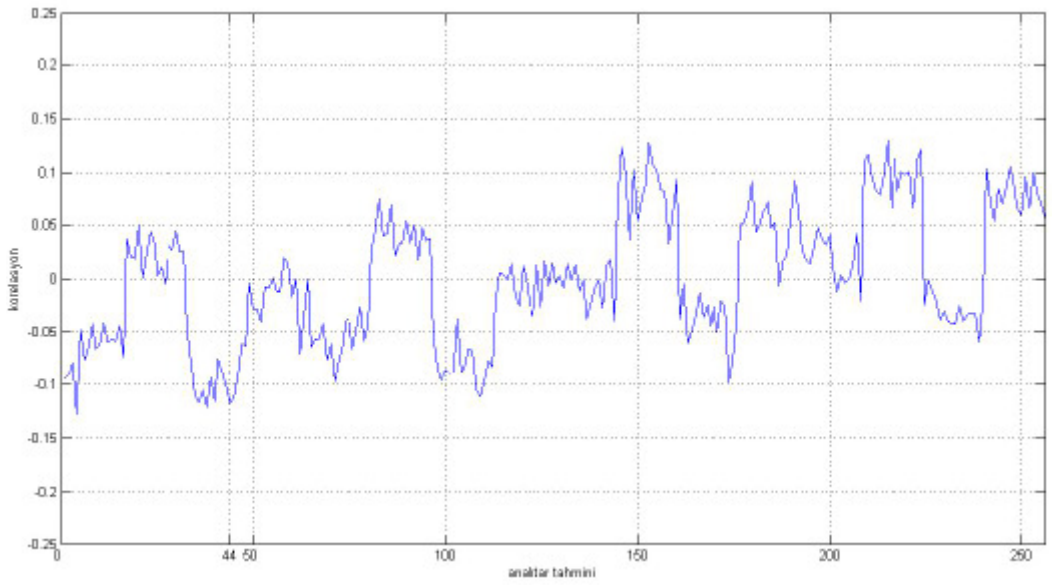
Şekil 6.1'deki korelasyon değerlerinin en yüksek olduğu nokta 43 noktasıdır. Bu noktada korelasyon olabileceği en yüksek değer olan 1 değerindedir. Kullandığımız

anahtar “2B” onluk sayı sisteminde 43’e eşittir. Bu adımda saldırı doğru sonuç vermiş, anahtar elde edilmiştir.

Tur anahtarını ekleme adımındaki işlem exor işlemi olduğundan kullanılan anahtara bir bit yakınlıktaki anahtar olasılıkları da yüksek sonuç vermiştir. Korelasyonun -1 olduğu değer için anahtar bilgisi 212’dir. 212 değeri ikilik sistemde 11010100’a eşittir. Bu da kullanılan 00101011 (2B) anahtarının bit bazında tersidir. Kullanılan anahtara bir bit yakınlıktaki 00001011 (11), 00100011 (35), 00101111 (47) anahtar değerleri için korelasyon Şekil 6.1’de görüldüğü gibi yüksek sonuç vermiştir.

6.2.2 Bayt değiştirme çıkışına saldırı

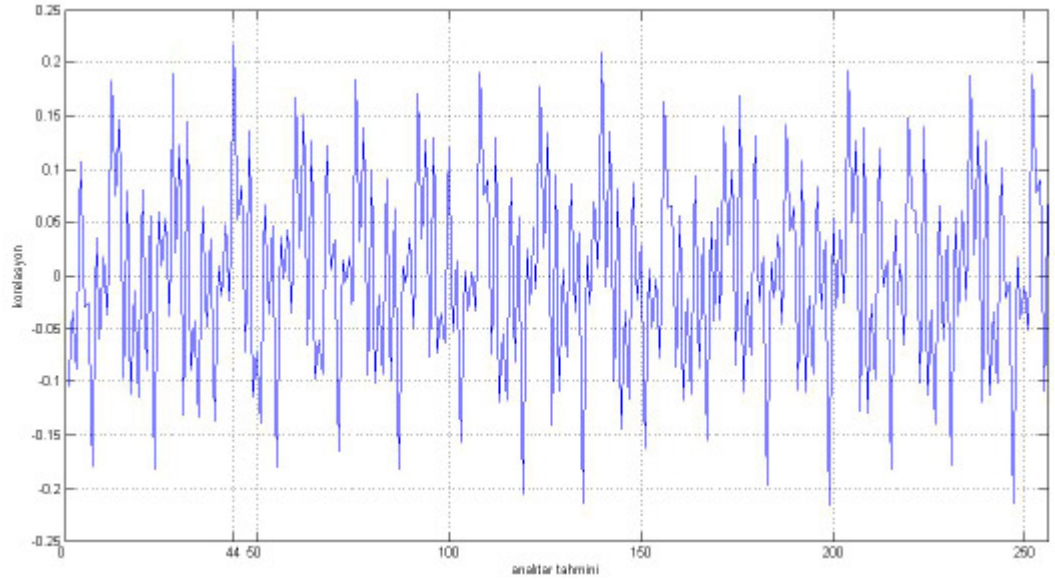
Bayt değiştirme çıkışına saldırı için öncelikle bir saldırı noktası seçilmiş ve bu nokta için tahmin ve benzetim matrisleri hesaplanmıştır. Korelasyon analizi yapıldığında korelasyon değerleri sıfır noktası civarında çıkmış, analiz anahtar hakkında bir bilgi vermemiştir. Korelasyon değerleri Şekil 6.2’de verilmiştir.



Şekil 6.2 : Bayt değiştirme ilk saldırı korelasyon sonuçları

Seçilen saldırı noktasında anahtar bilgisi elde edilemediği için saldırı noktası değiştirilmiş ve aynı ölçümler takrarlanarak yeni korelasyon bilgileri hesaplanmıştır. İkinci saldırı noktası için korelasyon bilgileri Şekil 6.3’teki gibidir.

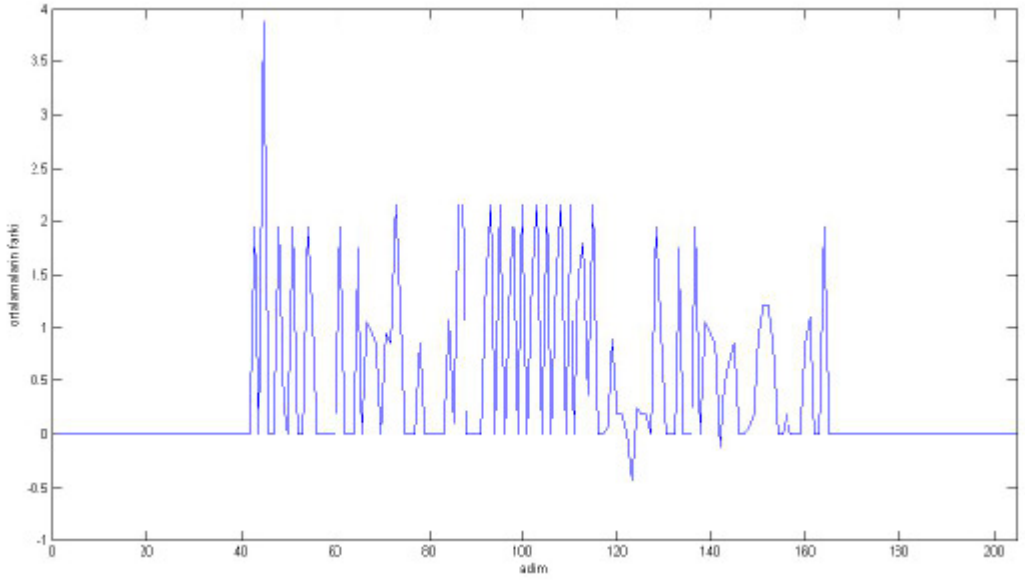
Bu noktadaki saldırıda korelasyon değerleri nispeten yükselmiş, korelasyonun en yüksek olduğu üç anahtar değeri tespit edilmiştir. Bu değerlerden biri kullanılan anahtardır. Ancak bu noktada da korelasyon analizi ile anahtar hakkında net bir bilgi



Şekil 6.3 : Bayt değiştirme ikinci saldırı korelasyon sonuçları

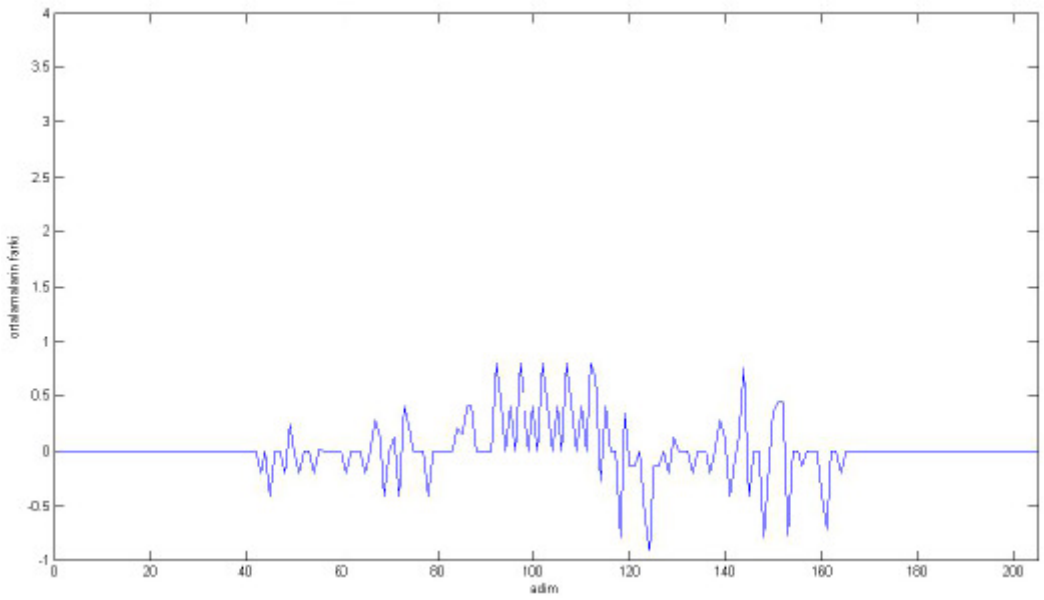
elde edilemediği için ortalamaların farkı testi yardımıyla anahtar elde edilmeye çalışılmıştır.

Test için gerekli ölçümler N=2000 düz metin örneği için yapılmış, Bölüm 6.1.2'de anlatıldığı şekilde fark değerleri hesaplanmıştır. Yüksek fark değerlerinin tespit edildiği anahtarın kullanılan anahtar olması beklenmektedir. Ancak en yüksek fark anahtar=199 değeri için elde edilmiştir (bkz: Şekil 6.4)



Şekil 6.4 : Anahtar = 199 için fark eğrisi

Kullanılan anahtar olan 43 için ise fark eğrisi beklenenden düşük değerlerde bulunmuştur (bkz: Şekil 6.5).



Şekil 6.5 : Anahtar = 43 için fark eğrisi

Bayt değiştirme çıkışına saldırılarda korelasyon analizinde kullanılan anahtar dahil olmak üzere üç anahtar olasılığı için korelasyon değerleri yüksek bulunmuştur, anahtarı elde etmeye yönelik iyi bir sonuç elde edilmiştir. Ortalamaların farkı testi ile sağlıklı bir sonuç elde edilememiştir. Bunun sebebi kullanılan düz metin sayısının yetersizliği olabilir. Bu sayı artırılarak test tekrarlanabilir.

7. SONUÇ

Bu çalışmada AES algoritması C programlama dili ile yazılmış ve bu yazılım Silicon Laboratories firmasına ait C8051F060 isimli mikroişlemci üzerinde Keil C51 simulasyon programı yardımı ile gerçeklenmiştir. Gerçeklemenin güç tüketimi analizi için Vb.Net programlama dili ile ayrı bir ölçüm programı hazırlanmış, bu program yardımı ile gerçeklemenin güç tüketimi tahmini olarak ölçülmüştür. Elde edilen ölçümler sonucu güç tüketim grafikleri çıkarılarak bu grafikler üzerinde gerçeklemenin aşamaları belirlenmiştir. Grafikler Matlab yardımı ile çizdirilmiştir. Son olarak gerçeklemenin güvenilirliği yan kanal saldırıları yapılarak test edilmiştir. Yan kanal analizlerinde korelasyon değerleri yine Matlab yardımı ile elde edilmiştir. Saldırı yapılan iki adımdan birinde anahtar elde edilmiş, diğerinde ise anahtar elde edilememiş ancak anahtar olasılığı 256 dan 3'e indirilebilmiştir.

Tez ile ilgili önemli bir nokta literatürdeki AES gerçeklemelerinin genellikle FPGA üzerinde olması, mikroişlemci üzerinde gerçekleme çalışmalarının sayıca çok az olmasıdır.

KAYNAKLAR

[1] **FIPS 197,** 2001: Advanced Encryption Standard. National Institute of Standarts and Technology (NIST).

[2] **Karakoç, F.,** 2008: "Kripto analizinde Melez Bir Yöntem: Çakışma Saldırısı", İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Türkiye.

[3] **Kula, G. Ç.,** 2006: "Mosquito Dizi Şifreleme Algoritmasının VHDL ile yazılımı ve FPGA Üzerinde Gerçeklenmesi", İstanbul Teknik Üniversitesi, Elektrik-Elektronik Fakültesi, Türkiye.

[4] **Ordu, L.,** 2006: "AES Algoritmasının FPGA Üzerinde Gerçeklenmesi ve Yen Kanal Analizi Saldırılarına Karşı Güçlendirilmesi", İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Türkiye.

[5] **Kayış, H.,** 2006: "AES Uygulamasının FPGA Gerçeklemelerine Karşı Güç analizi Saldırısı", İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Türkiye.

[6] **Şahinoğlu, M.,** 2009: "Gelişmiş Şifreleme Standardı Algoritmasının Donanım Üzerinde Gerçeklemesine Elektromanyetik Alan Saldırısı", İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Türkiye.

[7] **FIPS 46-3,** 2001: Data Encryption Standard. National Institute of Standarts and Technology (NIST).

[8] **Url-1** <*http://www.tr.wikipedia.org/wiki/Mikro%C5%9Flemciler*>*,* 18.04.2009

[9] **Url-2** <*http://www.elektrik.gen.tr/contenet/view/90/30*>*,* 17.04.2009

[10] **Url-3** <*http://www.tr.wikipedia.org/wiki/8051*>*,* 18.04.2009

[11] **Silicon Laboratories,** 2004: C8051F60/1/2/3/4/5/6/7 Mixed Signal ISP Flash MCU Family, Datasheet.

[12] **Url-4** <*http://www.keil.com*>*,* 15.04.2009

[13] **Url-5** <*http://www.keil.com/product/brochures/cx51_v8.pdf*>*,* 10.04.2009

[14] **Doğan, A. H.,** 2006: "AES Algoritmasının FPGA Üzerinde Düşük Güçlü Tasarımı", İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Türkiye.

[15] **Ferguson, N., Schneier B.,** 2003: Practical Cryptography. Wiley Pulishing, Inc., Indianapolis, Indiana.

[16] **Topaloğlu, N..** X86 Tabanlı Mikroişlemci Mimarisi ve Assembly Dili. 3. Baskı. Şubat 2008.

[17] **Wenn, D.Y., Whipple, G.G.,** 1994: Assembly Language Byte By Byte Programming In The IBM PC Environment. West Publishing Company,Minneapolis.

[18] **Antonakos, J.L.,** 1995: An Introduction to Assembly Language Programming for The Intel 8088 microporcessor. Prentice-Hall International (UK) Ltd., London, England.

[19] **Duntemann, J.,** 2000: Assembly Language: Step By Step. John Wiley & Sons Inc, New Jersey.

[20] **Ball, R., Pratt, B.,** 1986: Engineering Applications of Microcomputers: Instrumentation and Control. Prentice-Hall International (UK) Ltd., London, England.

[21] **Url-6** <*http://www.tr.wikipedia.org/wiki/korelasyon*>, 22.04.2009

EKLER

EK A : ASM kodları ve işlem süreleri

Çizelge A.1 : ASM kodu işlem süreleri (1)

Mnemonic	Description	Bytes	Clock Cycles
	Arithmetic Operations		
ADD A, Rn	Add register to A	1	1
ADD A, direct	Add direct byte to A	2	2
ADD A, @Ri	Add indirect RAM to A	1	2
ADD A, #data	Add immediate to A	2	2
ADDC A, Rn	Add register to A with carry	1	1
ADDC A, direct	Add direct byte to A with carry	2	2
ADDC A, @Ri	Add indirect RAM to A with carry	1	2
ADDC A, #data	Add immediate to A with carry	2	2
SUBB A, Rn	Subtract register from A with borrow	1	1
SUBB A, direct	Subtract direct byte from A with borrow	2	2
SUBB A, @Ri	Subtract indirect RAM from A with borrow	1	2
SUBB A, #data	Subtract immediate from A with borrow	2	2
INC A	Increment A	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	2
INC @Ri	Increment indirect RAM	1	2
DEC A	Decrement A	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	2
DEC @Ri	Decrement indirect RAM	1	2
INC DPTR	Increment Data Pointer	1	1
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	8
DA A	Decimal adjust A	1	1
	Logical Operations		
ANL A, Rn	AND Register to A	1	1
ANL A, direct	AND direct byte to A	2	2
ANL A, @Ri	AND indirect RAM to A	1	2
ANL A, #data	AND immediate to A	2	2
ANL direct, A	AND A to direct byte	2	2
ANL direct, #data	AND immediate to direct byte	3	3
ORL A, Rn	OR Register to A	1	1
ORL A, direct	OR direct byte to A	2	2
ORL A, @Ri	OR indirect RAM to A	1	2
ORL A, #data	OR immediate to A	2	2
ORL direct, A	OR A to direct byte	2	2
ORL direct, #data	OR immediate to direct byte	3	3
XRL A, Rn	Exclusive-OR Register to A	1	1
XRL A, direct	Exclusive-OR direct byte to A	2	2
XRL A, @Ri	Exclusive-OR indirect RAM to A	1	2
XRL A, #data	Exclusive-OR immediate to A	2	2
XRL direct, A	Exclusive-OR A to direct byte	2	2

Çizelge A.2 : ASM kodları işlem süreleri

Mnemonic	Description	Bytes	Clock Cycles
XRL direct, #data	Exclusive-OR immediate to direct byte	3	3
CLR A	Clear A	1	1
CPL A	Complement A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through Carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through Carry	1	1
SWAP A	Swap nibbles of A	1	1
Data Transfer			
MOV A, Rn	Move Register to A	1	1
MOV A, direct	Move direct byte to A	2	2
MOV A, @Ri	Move indirect RAM to A	1	2
MOV A, #data	Move immediate to A	2	2
MOV Rn, A	Move A to Register	1	1
MOV Rn, direct	Move direct byte to Register	2	2
MOV Rn, #data	Move immediate to Register	2	2
MOV direct, A	Move A to direct byte	2	2
MOV direct, Rn	Move Register to direct byte	2	2
MOV direct, direct	Move direct byte to direct byte	3	3
MOV direct, @Ri	Move indirect RAM to direct byte	2	2
MOV direct, #data	Move immediate to direct byte	3	3
MOV @Ri, A	Move A to indirect RAM	1	2
MOV @Ri, direct	Move direct byte to indirect RAM	2	2
MOV @Ri, #data	Move immediate to indirect RAM	2	2
MOV DPTR, #data16	Load DPTR with 16-bit constant	3	3
MOVC A, @A+DPTR	Move code byte relative DPTR to A	1	3
MOVC A, @A+PC	Move code byte relative PC to A	1	3
MOVX A, @Ri	Move external data (8-bit address) to A	1	3
MOVX @Ri, A	Move A to external data (8-bit address)	1	3
MOVX A, @DPTR	Move external data (16-bit address) to A	1	3
MOVX @DPTR, A	Move A to external data (16-bit address)	1	3
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A, Rn	Exchange Register with A	1	1
XCH A, direct	Exchange direct byte with A	2	2
XCH A, @Ri	Exchange indirect RAM with A	1	2
XCHD A, @Ri	Exchange low nibble of indirect RAM with A	1	2
Boolean Manipulation			
CLR C	Clear Carry	1	1
CLR bit	Clear direct bit	2	2
SETB C	Set Carry	1	1
SETB bit	Set direct bit	2	2
CPL C	Complement Carry	1	1
CPL bit	Complement direct bit	2	2
ANL C, bit	AND direct bit to Carry	2	2

Çizelge A.3 : ASM kodları işlem süreleri(3)

Mnemonic	Description	Bytes	Clock Cycles
ANL C, /bit	AND complement of direct bit to Carry	2	2
ORL C, bit	OR direct bit to carry	2	2
ORL C, /bit	OR complement of direct bit to Carry	2	2
MOV C, bit	Move direct bit to Carry	2	2
MOV bit, C	Move Carry to direct bit	2	2
JC rel	Jump if Carry is set	2	2/3
JNC rel	Jump if Carry is not set	2	2/3
JB bit, rel	Jump if direct bit is set	3	3/4
JNB bit, rel	Jump if direct bit is not set	3	3/4
JBC bit, rel	Jump if direct bit is set and clear bit	3	3/4
Program Branching			
ACALL addr11	Absolute subroutine call	2	3
LCALL addr16	Long subroutine call	3	4
RET	Return from subroutine	1	5
RETI	Return from interrupt	1	5
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump (relative address)	2	3
JMP @A+DPTR	Jump indirect relative to DPTR	1	3
JZ rel	Jump if A equals zero	2	2/3
JNZ rel	Jump if A does not equal zero	2	2/3
CJNE A, direct, rel	Compare direct byte to A and jump if not equal	3	3/4
CJNE A, #data, rel	Compare immediate to A and jump if not equal	3	3/4
CJNE Rn, #data, rel	Compare immediate to Register and jump if not equal	3	3/4
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal	3	4/5
DJNZ Rn, rel	Decrement Register and jump if not zero	2	2/3
DJNZ direct, rel	Decrement direct byte and jump if not zero	3	3/4
NOP	No operation	1	1

EK B:

Teze ek olarak verilmiş CD-ROM içerisinde aynı isimlerdeki klasörlerde AES C kaynak kodu, AES ASM kaynak kodu, ölçüm programı kaynak kodu ve ölçüm programı dosyaları bulunmaktadır.

ÖZGEÇMİŞ

Ad Soyad: Gizem Çisem KULA

Doğum Yeri ve Tarihi: Kırklareli, 1983

Adres: 19 Mayıs Mah. 19 Mayıs Cad. Tellioğlu apt. 16/4 Fulya Şişli/İSTANBUL

Lisans Üniversite: İstanbul Teknik Üniversitesi, Elektronik Mühendisliği, 2006.