1. GİRİŞ

Akıllı kartlar belli işlemleri gerçekleştiren küçük bilgisayarlar olarak düşünülebilir. Günümüzde akıllı kartların kullanımı gittikçe yaygınlaşmaktadır. Akıllı kartların kullanım alanlarına örnek olarak banka kartları, telekomünikasyon alanında kullanılan SIM kartlar, kimlik kartları verilebilir. Bazı ülkelerde akıllı kart temelli sağlık sistemine geçilmiştir.

Sağlık alanında akıllı kart kullanımını teşvik eden 4 önemli etken vardır [1].

- Hastanın kimliğini doğrulama
- Hasta ile hastaya ait veriyi eşleştirme
- Farklı kaynaklardan alınan verileri senkronize etme
- Güvenliği ve verilere erişim kontrolünü sağlama

Sağlık sisteminde akıllı kart kullanımının sağladığı en büyük avantajlardan biri servis kalitesini artırarak veri işlemek için harcanan maliyeti düşürmesidir [1, 2]. Hastanın kimliğinin belirlenmesi, hasta kabul işlemleri kolaylaştırılarak zaman ve maliyetten tasarruf sağlanır. Ayrıca güvenli kimlik doğrulama sayesinde kötüye kullanım ve sahtecilik azaltılır. ABD Ulusal Sağlık Sistemi Sahtecilik Önleme Kurumu'nun (The National Health Care Anti-Fraud Association) yaptığı tahmine göre [1] sağlık alanındaki sahtecilik 2007 yılında 68 milyar Amerikan Doları kadar kayba neden olmuştur. Akıllı kartların sağlık sisteminde kullanımının hasta güvenliği açısından da önemli faydaları vardır. Hastanın tıbbi geçmişine dair verilere (uygulanan tedaviler, kullanılan ilaçlar, alerjiler gibi) kolay ve hızlı ulaşım acil müdahalelerde hayat kurtarıcı rol oynar.

Sağlık alanında kullanılacak akıllı kart temelli bir sistem hastaların ve sağlık görevlilerinin sahip olacağı akıllı kartlar dışında kart okuyucular, haberleşme ağı, sunucular, kart bilgilerini güncellemek için kullanılacak terminaller gibi çeşitli birimler içerir. Kartın kısıtlı bir belleği olduğundan hastanın kişisel bilgileri ve hastaya dair acil tıbbi bilgilerin kartta saklanması, hastanın detaylı tıbbi geçmişine haberleşme ağı aracılığıyla sunuculara bağlanılarak ulaşılması uygun bir çözüm yoludur.

1.1 Çalışmanın Amacı

Hazırlanan bitirme çalışmasında sistemin akıllı kart ve kart okuyucu kısmı tasarlanmıştır. Kartın işlevi hastaya ait temel bilgileri saklamak ve kart okuyucuyla veri iletişimini güvenli bir şekilde yapmaktır.

2. KRİPTOGRAFİ

Bu bölümde kriptografi ve projede kullanılan kriptografik algoritmalar hakkında bilgi verilecektir.

Kriptografi gizli bilgileri gizli tutma bilimidir [3]. Kriptografinin temel amaçlarından biri şifreleme yöntemlerini kullanarak gizliliği (confidentiality) sağlamaktır. İletilecek mesaj şifrelenerek karşı tarafa iletilir. Karşı taraf şifreyi çözmek için gizli bir şifre çözücü anahtar kullanır. Şifreleme, şifrelenmiş mesajı ele geçiren herhangi birinin orijinal mesaja ulaşmasını önleyebilmelidir.

Gizlilik dışında kriptografiye ihtiyaç duyulma nedenleri veri bütünlüğü (data integrity), doğrulama (authentication) ve inkar edememedir (non-repudiation) [3]. Veri bütünlüğü orijinal mesajın, iletim sırasında kazayla veya bir müdahaleyle değişime uğrayıp uğramadığı kontrol edilerek doğrulanır. Doğrulama haberleşecek tarafların birbirlerinin kimliklerinden emin olmalarını sağlar. İnkar edememe ise bir mesajı gönderen kişinin daha sonrasında bu eylemi yapmadığını öne süememesini sağlar.

2.1 Şifreleme Yöntemleri

Şifreleme yöntemleri simetrik şifreleme ve açık-anahtar şifreleme olmak üzere ikiye ayrılır [3].

2.1.1 Simetrik Şifreleme

Simetrik şifrelemede haberleşen iki taraf şifreleme ve şifre çözmede aynı anahtarı kullanır [3]. Şifreleme algoritması E ve şifre çözme algoritması D herkesçe bilinir. Bu nedenle güvenlik için kullanılan anahtarın gizli olması şarttır. Güvenli bir haberleşme kanalı oluşturmak için haberleşecek iki taraf gizli bir k anahtarı üzerinde anlaşmaya varır. Taraflardan biri göndereceği m mesajını E algoritmasını kullanarak şifreler ve *c* şifreli mesajını aşağıdaki denklemdeki gibi üretir.

$$c = E(k, m) \tag{2.1}$$

Şifreli mesajı alan taraf D algoritmasını ve ortak anahtarı kullanarak şifreyi çözer [3].

$$m = D(k, c) \tag{2.2}$$

Simetrik anahtarlı şifreleme algoritmaları hem donanım hem de yazılım gerçeklemelerinde en hızlı çalışan kriptografi algoritmalarıdır [3]. Bu nedenle büyük verilerin şifrelenmesinde kullanılmaları avantajlıdır.

Şifreleme fonksiyonları blok şifreleme ve dizi şifreleme fonksiyonları olmak üzere ikiye ayrılır. Blok şifreleme fonksiyonları belli bir uzunluktaki mesajları şifreler. Blok şifreleyiciler blok halinde aldıkları belli uzunluktaki veriyi aynı uzunluktaki bloklar halinde şifrelerler. Blok şifreleyicilere örnek olarak DES ve AES verilebilir [4]. Dizi şifreleme fonksiyonları ise karakter karakter işlem yaparak rastgele uzunluklu mesajları şifreler. Dizi şifreleyiciler blok şifreleyicilere göre daha küçük uzunluklu verileri işler. Blok şifreleyicilerde tipik blok uzunlukları 64 veya 128 bit iken, tipik anahtar uzunlukları 56, 128, 192 veya 256 bittir [3].

2.1.1.1 AES

1997'de ABD Ulusal Standartlar ve Teknoloji Enstitüsü (NIST: National Institute of Standards and Technology) Gelişmiş Şifreleme Standardı (AES: Advanced Encryption Standard) adı verilecek yeni bir şifreleme standardı seçilmesi için çalışma başlattı [3, 4]. Önerilecek algoritmaların en az 128 bitlik blok uzunluğunu, ve 128, 192 ve 256 bit olmak üzere üç farklı anahtar uzunluğunu desteklemesi istendi. 2000'de adaylar arasından J. Daemen ve V. Rijmen tarafından tasarlanan Rijndael seçildi. Rijndael şifreleme dışında kriptografik özet fonksiyonlarının ve sözde-rastgele bit üreticilerinin gerçeklenmesinde de kullanılabilmektedir.

2.1.1.1.1 Algoritmanın Yapısı

AES algoritması tur adı verilen işlemleri tekrarlayarak çalışır [3, 4]. Tur sayısı N_r, anahtar uzunluğu N_k'ya bağlıdır. Tasarlanan sistemde N_k 128 bit, blok uzunluğu N_b 128 bit olduğundan tur sayısı 10'dur. Tur fonksiyonu "durum" adı verilen 16 baytlık bilgileri işler. Durum değişkeni şifrelemenin başında şifrelenecek mesaj bloğunu (plaintext), şifreleme tamamlandığında şifrelenmiş mesajı (ciphertext) içerir.

Algoritma şifreleyici/şifre çözücü ve anahtar üreteci olmak üzere iki temel bloktan oluşur [4]. Şifreleme/şifre çözme bloğu üç ana bölümden oluşur; ilk anahtar toplaması, N_r-*1* adet tur, son tur. İlk anahtar toplaması turunda algoritmaya girilen veri ile anahtar XOR'lanarak ilk durum elde edilir. Elde edilen durum anahtar üreticinin ürettiği anahtarla işleme sokulur ve yeni durum değişkeni elde edilir. Bu işlem N_r-*1* defa tekrarlandıktan sonra son tur işleme sokulur ve elde edilen son durum bilgisi şifreli mesajı verir.

Şifreleme yapılırken N_r-*1* defa tekrarlanan her tur 4 adımdan oluşur [3];

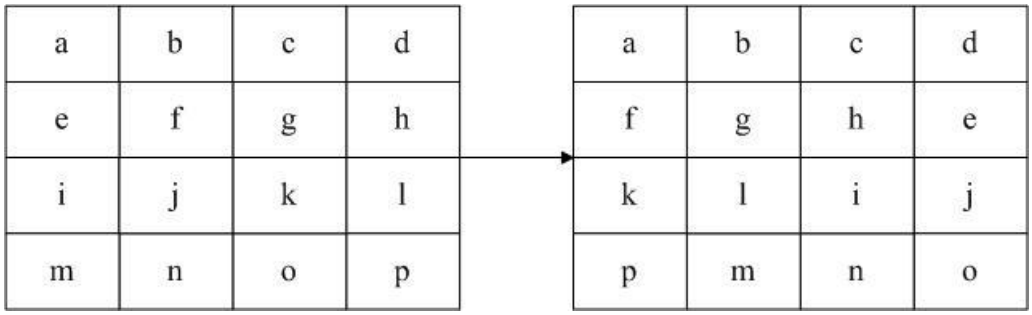
- Bayt yer değiştirme
- Satır kaydırma
- Sütun karıştırma
- Anahtar toplama

Bayt yer değiştirme adımı algoritmanın doğrusal olmayan tek adımıdır. Bu adımda durum matrisinin baytlarının yeri "S kutusu" adı verilen bir tablo yardımıyla değiştirilir. 128 bitlik blok uzunluğu için durum matrisi Tablo 2.1'deki gibidir [3].

Tablo2.1 128 bitlik blok uzunluğu için durum matrisi

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

Satır kaydırma adımında durum matrisinin satırları dairesel olarak sola kaydırılır [3]. Her satır için kaydırma miktarı farklıdır ve bu miktar blok uzunluğuna bağlıdır. 128 bitlik blok uzunluğu için birinci satır aynı bırakılır, ikinci satır elemanları sola 1 pozisyon kaydırılır. Üçüncü satır elemanları sola 2 pozisyon ve dördüncü satır elemanları sola 3 pozisyon kaydırılır. Şekil 2.1'de satır kaydırma işlemi öncesi ve sonrası durum matrisi gösterilmektedir [3, 4].



Şekil 2.1 Satır kaydırma adımı öncesi ve sonrası durum matrisi

Sütun karıştırma işlemi durum matrisinin sütunları üzerine uygulanır. Her sütun üçüncü dereceden bir polinom gibi düşünülür. Bu polinomlar sabit başka bir polinomla çarpılır ve elde edilen sonucun $(x^4 + 1)$'e göre modülü alınır [3, 4].

Anahtar toplama adımıyla her tur sonunda elde edilen durum verisi ve tur anahtarı bayt bayt XOR'lanır [4].

Anahtar üreteci her turun sonunda bir tur anahtarı üretir. Tur anahtarı üretilen ara değerle toplanır [4].

Projede kullanılan AES bloğu hazır olarak [4] numaralı tezden alınmıştır. AES bloğu güvenli veri iletişimini sağlamak için şifrelemede kullanılmıştır.

2.1.2 Açık Anahtarlı Şifreleme

Açık anahtarlı şifrelemede herkeste iki anahtar bulunur. Bunlardan biri şifrelemede kullanılan ve herkesin ulaşabileceği açık anahtar, diğeri sadece sahibinin bildiği ve şifre çözmede kullanılan gizli anahtardır.

Simetrik kriptografi kullanıcılara güvenli bir haberleşme kanalı sağlar. Güvenli bir haberleşme kanalı kurmak için tarafların ortak bir gizli anahtar üzerinde anlaşmaları gerekmektedir. Gizli anahtarların güvenli bir şekilde dağıtılması için açık anahtar tekniklerine ihtiyaç duyulmaktadır. Simetrik kriptografi gizliliği ve veri bütünlüğünü kontrol etmeyi sağlar, ancak anahtar paylaşımı, inkar edememe ve bazı doğrulama şekillerinde açık anahtarlı teknolojiye ihtiyaç duyulur [3].

1976'da W. Diffie ve M.E. Hellman açık anahtarlı kriptografi fikrini ortaya attılar [5]. R. Rivest, A. Shamir ve L. Adleman tarafından tasarlanan ve ilk açık anahtarlı kriptosistem olan RSA anahtar paylaşımı ve dijital imzalamada kullanılmaktadır [3].

Diffie ve Hellman [5] numaralı makalede bir açık anahtar paylaşım sistemi önermişlerdir. Bu önerilen teknik, q bir asal sayı olmak üzere, q adet elemanı olan GF(q) sonlu alanında logaritma hesaplamanın zorluğundan yararlanmaktadır [5]. GF(q) sonlu alanı, q asal sayısı kadar elemana sahip bir $Z_q = \{0, 1, 2 \ldots q\text{-}1\}$ kümesi üzerinde modülo q toplama ve modülo q çarpma işlemlerinin tanımlanmasıyla elde edilen, yine q sayıda elemanı olan sonlu alandır [3]. [5] numaralı makaleye göre, a GF(q) sonlu alanının sabit bir birincil elemanı olmak üzere;

$$Y = a^X mod q, \qquad 1 \leq X \leq q-1 \tag{2.3}$$

$$X = log_a Y mod q, \quad 1 \leq Y \leq q-1 \tag{2.4}$$

(2.3) ve (2.4) denklemleri kullanılarak Y'nin hesaplanması kolay; ancak Y'den X'in hesaplanması zordur.

2.1.2.1 RSA

A ve B arasında güvenli haberleşmede kullanılmak üzere çalıştırılacak RSA algoritması, [6] numaralı kaynakta belirtildiği gibi, anahtar üretimi ve açık-anahtar şifreleme olmak üzere iki kısma ayrılabilir. RSA anahtar üretimi evresinde ilk olarak

taraflardan biri, örneğin B, iki adet büyük, rastgele, asal p ve q sayılarını üretir. Bu p ve q sayıları (2.5) eşitsizliğini sağlamalıdır.

$$p \neq q \tag{2.5}$$

B, p ve q sayılarını belirledikten sonra n ve $\phi(n)$ değerlerini (2.6a) ve (2.6b) eşitliklerini kullanarak hesaplar. n değeri B'nin RSA modülüdür.

$$n = p \cdot q \tag{2.6a}$$

$$\phi(n) = (p-1) \cdot (q-1) \tag{2.6b}$$

B daha sonra şifreleme üsteli adı verilen rastgele e sayısını seçer. e'nin $\phi(n)$'le en büyük ortak böleninin 1 olması ve (2.7) eşitsizliğini sağlaması gerekir.

$$1 < e < \phi(n) \tag{2.7}$$

B, (2.8a) ve (2.8b) ifadelerini sağlayan d doğal sayısını hesaplar.

$$1 < d < \phi(n) \tag{2.8a}$$

$$e \cdot d \equiv 1 \ (mod\ \phi(n)) \tag{2.8b}$$

(n, e) B'nin açık anahtarı, d ise gizli anahtarıdır [6].

Algoritmanın açık-anahtar şifreleme kısmında B'nin açık anahtarı (n, e)'yi bilen A, $c \equiv m^e \ (mod\ n)$ işlemini yaparak şifreli mesaj c'yi orijinal mesaj m'den hesaplar. A şifreli mesajı B'ye gönderir. B, $m \equiv c^d \ (mod\ n)$ işlemini yaparak orijinal mesaja ulaşır [6].

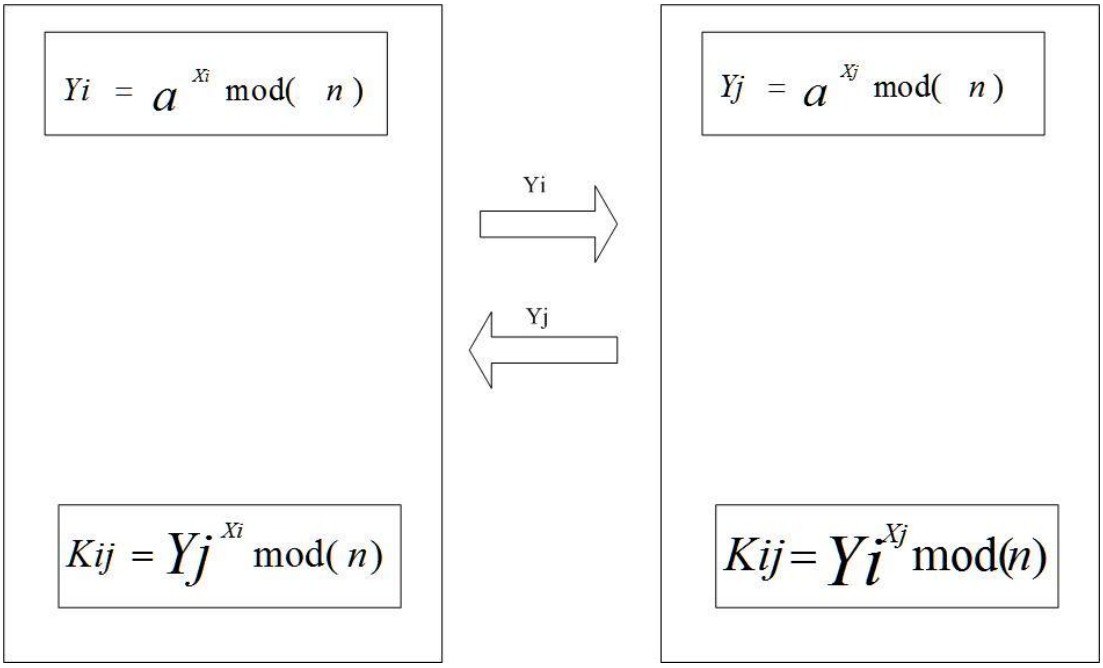
Bitirme çalışmasında RSA bloğu [7] numaralı tezden hazır olarak alınıp kullanılmıştır.

2.1.2.2 Diffie-Hellman Anahtar Paylaşımı

Şifrelemede ve şifre çözmede kullanılacak anahtarın taraflar arasında paylaşımı probleminin çözümüne yönelik olarak sunulmuştur. Her kullanıcı rastgele bir sayı seçer ve (2.9) eşitliğindeki hesabı yapar [5]. Buradaki X_i sayısı kullanıcının seçtiği rastgele sayı, a GF(n) sonlu alanının bir elemanıdır. Hesaplanan Y_i herkese açıktır. 1 ve 2 numaralı kullanıcılar güvenli bir şekilde haberleşmek istediklerinde her iki taraf diğerinin Y_i değerini alarak (2.10) işlemini yapar ve kullanılacak ortak anahtara ulaşır.

$$Y_i \equiv a^{X_i} \ (mod\ n) \tag{2.9}$$

$$K_{ij} \equiv Y_i{}^{X_j} \ (mod\ n) \tag{2.10}$$



Şekil 2.2 Diffie-Hellman anahtar paylaşımı

Şekil 2.2'de Diffie-Hellman anahtar paylaşımının iki kullanıcı arasında nasıl yapıldığı gösterilmektedir.

2.2 Rastgele Sayı Üretimi

Rastgele sayı üreticiler kriptografik mekanizmalarda ihtiyaç duyulan birimlerdir. Örneğin bir önceki bölümde anlatılan anahtar paylaşımı yapılırken rastgele sayılar kullanılmaktadır. Sistemin güvenilirliği için üretilen rastgele sayıların tahmin edilemez olması önemlidir. Yapılam çalışmada bir sözde-rastgele sayı üretici kullanılmıştır. Sözde-rastgele sayı üretici, kendisine verilen bir başlangıç değerinden (seed) yola çıkarak rastgele gibi görünen bir sayı üreten deterministik algoritmalardır [8].

Bu çalışmada sözde-rastgele sayı üretici olarak doğrusal geri-beslemeli ötelemeli kaydedici (LFSR: Linear feedback shift register) kullanılmıştır. L bit uzunluklu bir LFSR saat işareti geldiğinde her bitini bir düşük anlamlı bite kaydırır. L-1 numaralı en anlamlı bitin yerine de belli bitlerin XOR'lanmasıyla elde edilen değeri yazar. LFSR'ın belli bir periyodu tamamladıktan sonra başlangıç değerine geri döner. Bu periyodun mümkün olduğunca büyük olması güvenlik açısından önemlidir. Yapılan çalışmada 128 bitlik rastgele sayıya ihtiyaç duyulmuştur ve bu bit uzunluklu bir LFSR için maksimum periyodu sağlayan XNOR'lu bir geri-besleme fonksiyonu bulunmuştur. Gerçekleme bu XNOR'lu fonksiyon kullanılarak yapılmıştır.

3. AKILLI KARTLAR

Akıllı kartlar içlerinde tümdevre barındıran taşınabilir kartlardır. Hafıza kartları ve mikroişlemcili kartlar olmak üzere iki tür akıllı kart vardır [9].

3.1 Hafıza Kartları

Hafıza kartları ön ödemeli telefon uygulamalarında kullanılan eski tip akıllı kartlardır [9]. Bu telefon kartlarında kredi bilgisi bulunmakta ve her aramada kredi düşmektedir. Ancak bu kartların sınırlı güvenlik özelliği yüzünden sakladıkları bilgi kolayca değiştirilebilmektedir. Bu nedenle bu tip akıllı kartların işlevselliği kısıtlıdır.

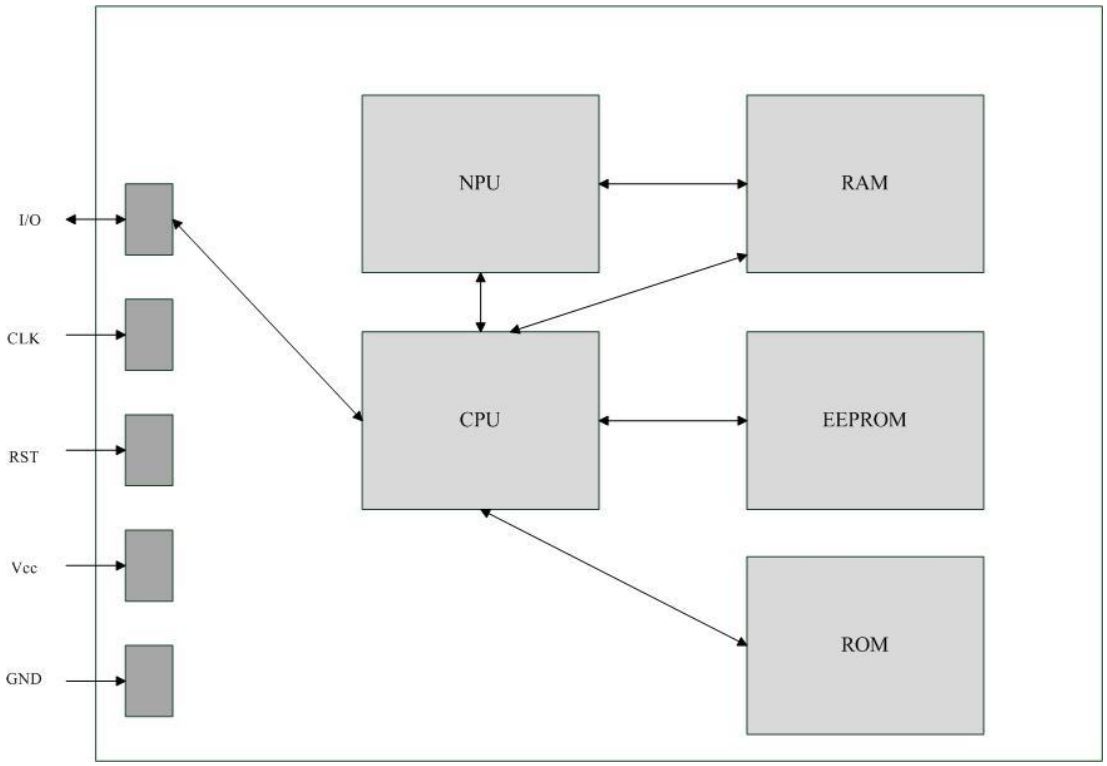
3.2 Mikroişlemcili Kartlar

Mikroişlemcili kartlar olarak adlandırılan ikinci tip akıllı kartlar ilk olarak banka kartları olarak kullanılmıştır [9]. Bu kartlar özel anahtarları güvenli bir şekilde barındırabilir ve modern kriptografik algoritmaları çalıştırabilirler. Bu kartlarla yapılabilecekler işlemcilerinin kapasitesi ve veri depolama birimlerinin büyüklükleriyle sınırlıdır. Şekil 3.1'de mikroişlemcili bir kartın içerdiği bloklar gösterilmektedir [9].

3.3 Sağlık Alanında Kullanılan Akıllı Kartlar

Sağlık alanında akıllı kartlar çeşitli ülkelerde kullanılmaya başlanmıştır. Bu kartlardan iki ana işlevi gerçekleştirmesi beklenmektedir [9]. İşlevlerden birincisi kartın sahibini doktora tanıtmasıdır. İkinci işlevi ise hasta ile ilgili verileri saklaması ve bilgisayarlarla veri alışverişi yapabilmesidir.

Bu bitirme çalışması yürütülürken şu ana kadar tasarlanmış ve uygulamaya konmuş olan sağlık kartlarıyla ilgili raporlar incelenmiştir. [10] numaralı kaynaktan Tayvan'da kullanımda olan sağlık kartı ile ilgili bilgi edinilmiştir. Bu kart mikrodenetleyici temelli bir akıllı karttır. 32 KB bellek bulundurmaktadır. Kartın belleğinin 22 KB'lık kısmı hasta ile ilgili kişisel bilgiler (kart sahibinin adı, cinsiyeti, kimlik numarası vb), sigorta bilgileri (tıbbi harcamalar, hastane ziyaretleri ve kabulleri), tıbbi hizmet bilgileri (alerji durumu, uygulanan tedaviler) ve sağlık sistemi yönetimiyle ilgili bilgilere (organ bağışı durumu) ayrılmıştır. Kalan 10 KB'lık bellek kart kullanıldıkça eklenecek bilgilere ayrılmıştır.



Şekil 3.1 Yardımcı işlemci bulunduran temaslı bir mikroişlemcili akıllı kart

Tayvan'da kullanılan sağlık kartında güvenlikle ilgili kartta tutulan verilere erişimi kontrol etmek için yetkilendirme, kişisel bilgileri korumak için kimlik doğrulama ve bilgi gizliliğini sağlamak için şifreleme yapılmıştır [10].

4. TASARIM ORTAMI

Bu bölümde izlenen tasarım yöntemi ve kullanılan araçlar anlatılacaktır. Proje yürütülürken SystemC ile VHDL dilleri kullanılmış ve Aldec Active-HDL ile CoWare tasarım ortamlarında çalışılmıştır.

4.1 Tasarım Yöntemi

Tasarım yöntemi olarak sistem düzeyinde tasarım metodu izlenmiştir. Bu yöntem yukarıdan aşağıya bir yaklaşım gerektirmektedir. Sistem düzeyinde tasarım üst seviyeli bir yöntemdir ve bütün bir sistemi yüksek seviyeli bir dilde modellenerek izlenir. [11] numaralı kaynağa göre sistem düzeyinde tasarım (ESL: Electronic System Level Design) net bir tanımı olmamakla beraber, donanım ve yazılım ilişkisi ve yüksek seviyeli soyutluk içeren bir yöntemdir.

Projede tasarlanan sistem heterojen bir sistem olduğundan donanım-yazılım ortak tasarım metotları araştırılmıştır. Donanım-yazılım ortak tasarımı yapan tasarımcılar belirlenen performans kriterleri ve gerçeklemede kullanılacak teknoloji doğrultusunda donanımsal ve yazılımsal parçalar bir arada çalışırken oluşacak ödünleşmeleri (trade-off) göz önünde bulundururlar [12]. [12] numaralı kaynakta önerilen tasarım yöntemi şekil 4.1'deki gibidir.

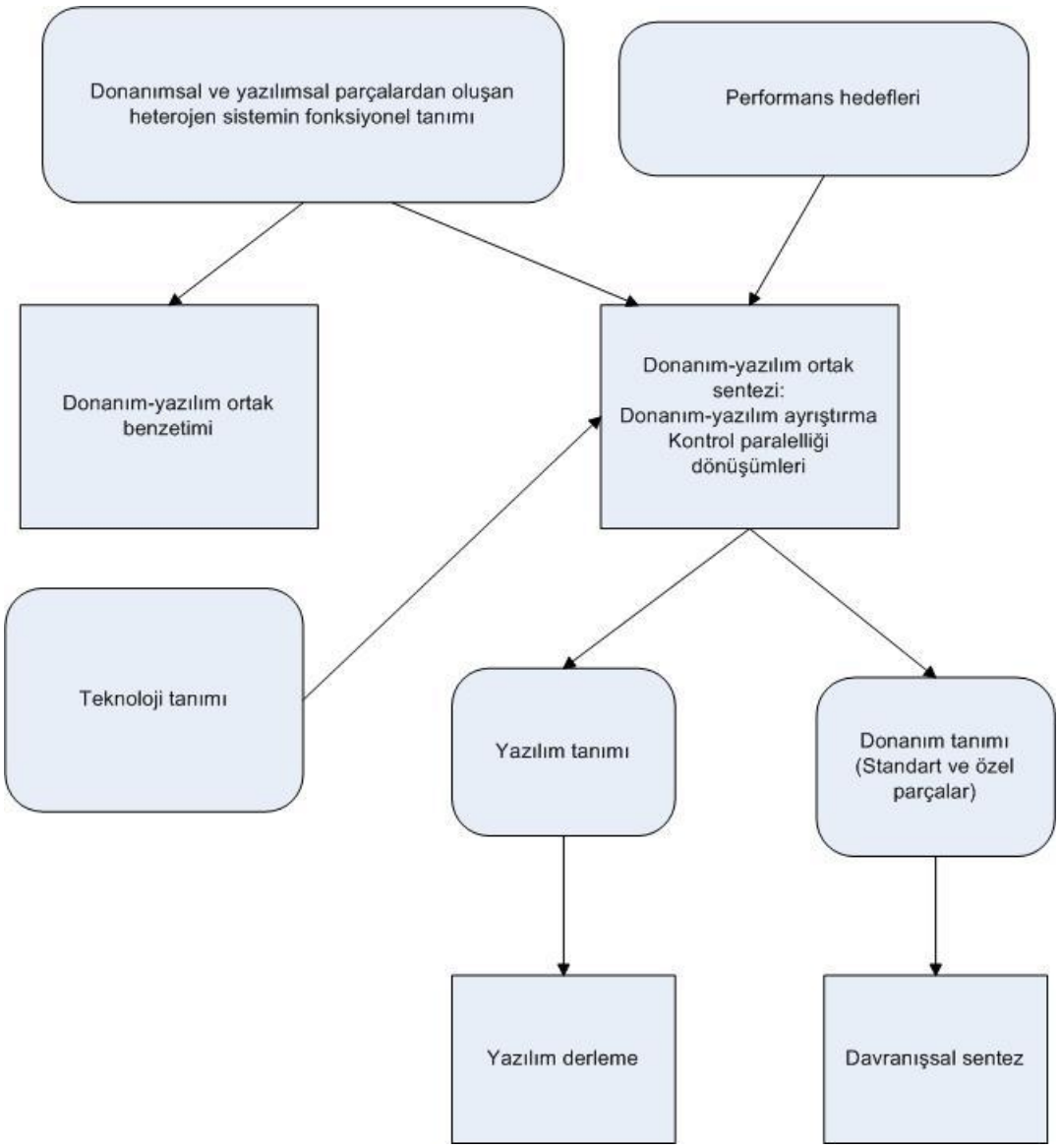
4.2 SystemC

SystemC elektronik sistem tasarımcılarının verimliliğini artırmak amacıyla geliştirilmiş bir sistem tasarım dilidir [13]. Günümüzde tasarlanan sistemler uygulamaya özgü donanımlar ve yazılımlar içermektedir. Çoğu zaman bu donanımsal ve yazılımsal parçaların kısıtlı bir sürede birlikte geliştirilmesi gerekmektedir. Büyük maddi kayıpları önlemek için bu parçaların bir bütün halinde fonksiyonel testinin yapılması gerekliliği doğmuştur [13].

SystemC donanımsal ve yazılımsal parçaları birlikte tasarlamayı mümkün kılar. Yüksek seviyeli oluşu sistemin tasarımında çalışan mühendislerin bütün sistemin işleyişini kavramalarını sağlar [13].

SystemC bir C++ sınıf kütüphanesidir [13]. C++ sentaksı kullanarak donanım tasarımına olanak sağlar. C++ derleme ve geliştirme ortamlarının sayıca çok ve oturmuş olması SystemC tasarımı yapmak isteyenler için kolaylık sağlar. Bu bölümde SystemC'yi daha iyi anlamak için SystemC ile ilgili temel kavramlara değinilecektir. Bu kavramlar Şekil 4.2'deki örnek kod parçası üzerinden örnekler

verilerek açıklanacaktır. Kod parçasındaki değişkenlere dilin anahtar kelimeleriyle uyum sağlamaları açısından İngilizce isimler verilmiştir.



Şekil 4.1 Donanım-yazılım ortak tasarımı yöntemi

4.2.1 Modüller

Modüller (Module) bir sistemin temel parçalarıdır. Şekil 4.2'deki kodda *module1* adlı bir sınıf *sc_module* kütüphane sınıfından türetilmektedir.

```
SC_MODULE(module1)
{
        sc_in <bool> clk;
        sc_in <sc_uint<32>> in1;
        sc_out<sc_uint<32>> out1;

        void process1()
        {
                out1.write(in1.read());
        }

        SC_CTOR(module1):
                clk("clk"),
                out1("out1"),
                in1("in1")
        {
                SC_THREAD(process1);
                sensitive << clk.pos();
        }
};
```

Şekil 4.2 SystemC'de yazılmış örnek bir kod parçası

4.2.2 Girişler ve Çıkışlar

Girişler ve Çıkışlar (Port) modüllerin dış dünyayla iletişimini sağlayan ara yüzlerdir [13, 14]. *sc_in* anahtar kelimesiyle giriş kapıları, *sc_out* anahtar kelimesiyle çıkış kapıları tanımlanır.

4.2.3 Prosesler

SystemC'de prosesler (Process) temel işlevsellik birimidir [14]. Programlama dillerinde işlevler ardışıl olarak çalıştırılır. Ancak elektronik sistemlerde paralel işlem yapmaya da ihtiyaç duyulduğundan SystemC prosesleri bu ihtiyacı karşılamaktadır [14].

SystemC'de metotlar (Method) ve iplikler (Thread) olarak adlandırılan iki tür proses vardır. İpliklerin metotlardan farkı bekletilip tekrar aktive edilebilmeleridir [13, 14]. Şekil 4.2'deki kod parçasında bu iki tür prosesten iplik kullanılmıştır. *process1*, prosesin üye fonksiyonudur. *SC_THREAD(process1)* ifadesi *process1* fonksiyonunu bir akış işlemine adresler. *sensitive << clk.pos()* ifadesi ise *clk* değişkeni ile gösterilen giriş kapısına gelen işaretin her yükselen kenarında *process1* fonksiyonunun çalıştırılmasını sağlar.

4.2.4 Kurucu Fonksiyonlar

Kurucu fonksiyonlar (Constructor) *SC_CTOR* makrosuyla tanımlanır. Kurucu fonksiyonlar üye fonksiyonları proseslere adresleme işini yaparlar [14].

Şekil 4.2'deki kod parçasında yapılanları bütün olarak açıklayacak olursak; bir saat girişi, 32'şer bitlik bir giriş ve bir çıkışı bulunan bir modül tanımlanmıştır. Sistem çalıştığında gelen her saat darbesinin yükselen kenarında *in1* girişindeki değer *out1* çıkışına yazılmaktadır.

Yapılan bitirme çalışmasında zaman kısıtlaması nedeniyle SystemC dili derinlemesine öğrenilmemiş, tasarlanacak sistemde duyulan ihtiyaçlara yönelik bir çalışma yolu izlenmiştir.

4.3 VHDL

VHDL bir donanım tanımlama dilidir (HDL: Hardware description language). Yapılan bitirme çalışmasında bazı bloklar VHDL kullanılarak tasarlanmıştır. Ayrıca [4] ve [7] numaralı tezlerden hazır olarak alınan iki VHDL bloğu sistemde kullanılmıştır.

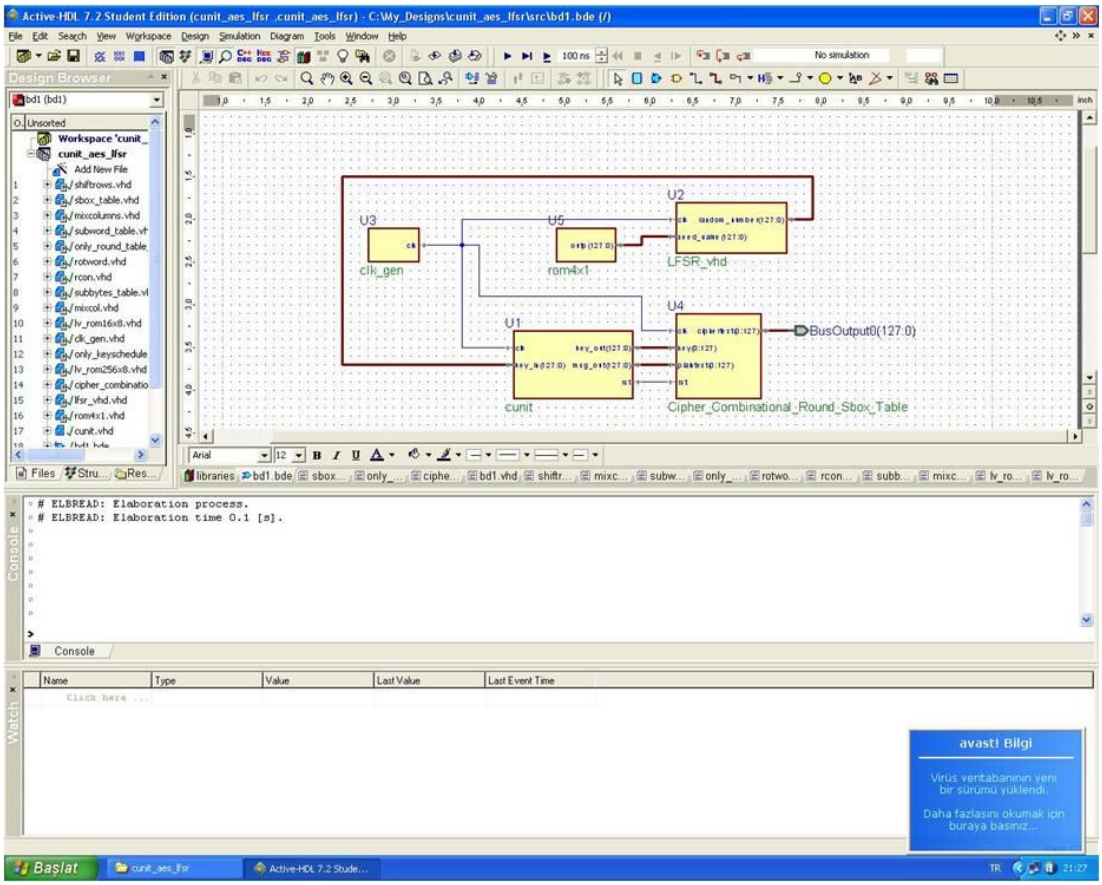
4.4 CoWare

CoWare yazılımsal ve donanımsal parçalardan oluşan heterojen sistemlerin tasarımının yapılabildiği bir ortamdır [15]. Bitirme projesine başlandığında bu yazılım paketinin SystemC IDE, Platform Architect ve Processor Designer adlı alt yazılımları kurulmuştur. Processor Designer LISA (Language for Instruction Set Architectures) dili kullanılarak işlemci tasarlamak için geliştirilmiş bir ortamdır. SystemC IDE SystemC geliştirme ortamıdır. Platform Architect ise kendi kütüphanesinde bulunan ARM işlemciler, AMBA veri yolları gibi standart donanımlarla tasarımcının SystemC'de yaptığı tasarımların kullanılarak sistem tasarımı ve benzetiminin yapılabildiği bir ortamdır.

Bitirme çalışmasında tasarlanan sistem hem SystemC'de hem VHDL'de yazılmış bloklar içerdiğinden karışık dillerle benzetim yapan bir ortama ihtiyaç duyulmuştur. CoWare bunu desteklediği HDL simülatörlerinin varlığı halinde yapabilmektedir. SystemC IDE'nin C kütüphaneleriyle ilgili verdiği derleme hatasının çözülememesi ve karışık dillerde yazılmış sistemin benzetiminin mevcut simülatörlerle yapılamaması nedeniyle çalışmaya CoWare ile devam edilmemiştir.

4.5 Aldec Active-HDL

Proje yürütülürken karşılaşılan VHDL-SystemC ortak benzetimi sorunu Aldec'in Active-HDL yazılımıyla çözülebilmiştir. Active-HDL bir benzetim ve geliştirme ortamıdır [16]. Active-HDL'de SystemC, VHDL, Verilog, SystemVerilog dilleriyle çalışılabilmektedir. Ayrıca bu yazılım bir sistemin farklı dillerde yazılan parçalarının birlikte benzetimini yapmaya da imkan vermektedir. Active-HDL'de tasarlanan modüllerin birbirleriyle bağlantıları veri yolları ve kablolar kullanılarak Şekil 4.3'te görüldüğü gibi görsel olarak yapılabilmektedir.



Şekil 4.3 Active-HDL tasarım ve benzetim ortamı

5. TASARLANAN SİSTEM

Sistem tasarlanırken ilk olarak ihtiyaçlar ve sistemin gerçekleştirmesi beklenen işlevler belirlendi. Bu işlevler şu şekilde sıralanabilir:

- Kart okuyucudan karta bir iletişim talebi geldiğinde güvenli bir haberleşme kanalı oluşturmak için anahtar paylaşımı yapmak
- Kart sahibi ile ilgili bilgileri saklamak
- Kart okuyucuya istenen bilgiyi şifreleyerek göndermek

Bu gereksinimler göz önünde bulundurularak sistemin blok diyagramı Şekil 5.1'deki gibi oluşturuldu.

Bu bölümün devamında sistemin alt blokları ve nasıl çalıştıkları anlatılacaktır.

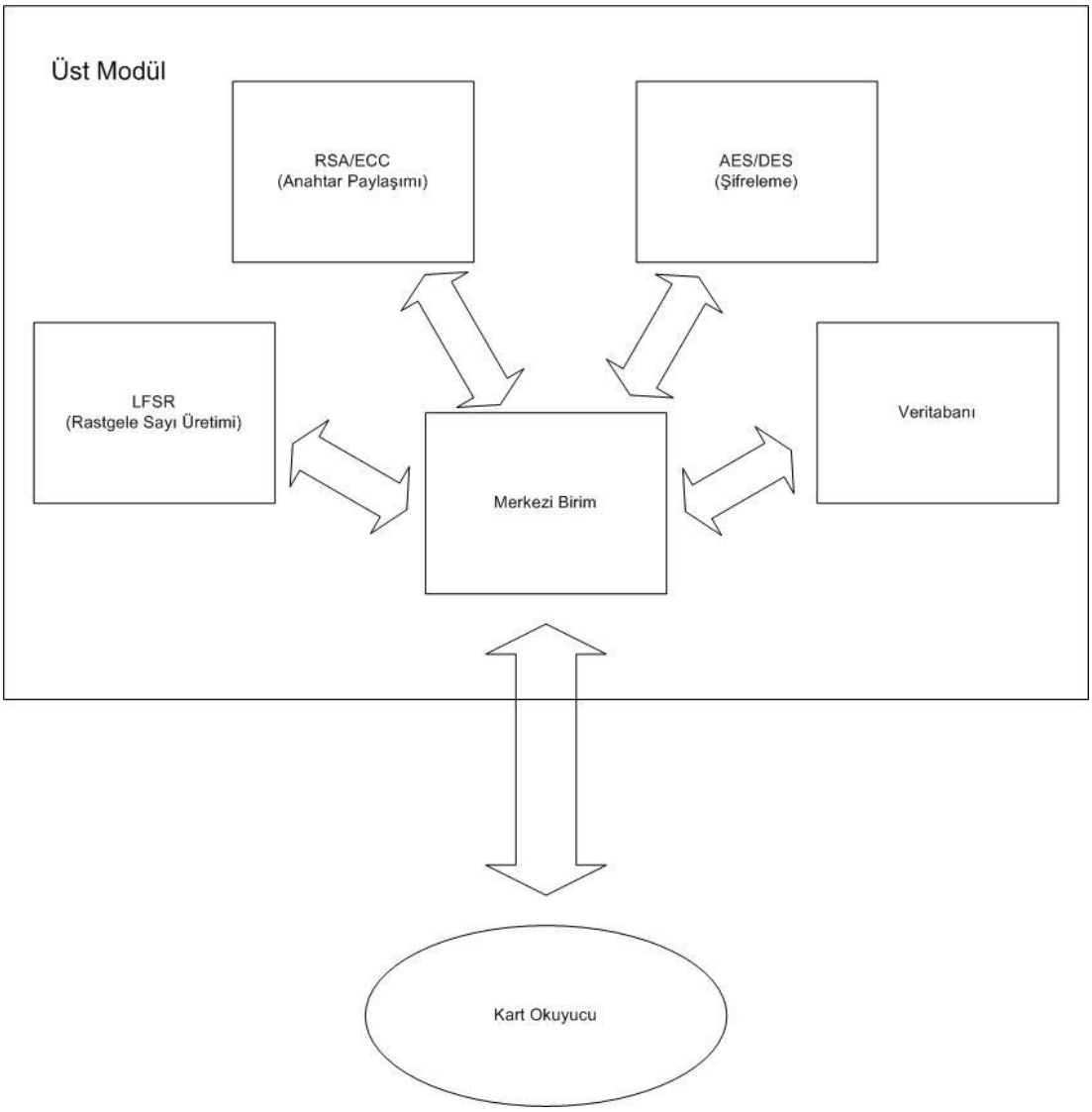
5.1 Merkezi Birim

Kartın merkezi birimi dış dünyayla ve diğer alt bloklarla iletişim kurmaktadır. Bu birim SystemC'de yazılmıştır. Merkezi birim dışarıya bir bilgi gönderileceği zaman ilk olarak sözde-rastgele sayı üreticiye bir başlangıç değeri ve başla işareti gönderir. Üretilen rastgele sayıyı Diffie-Hellman anahtar paylaşımında kullanmak üzere alır. Rastgele sayıyı (5.1) denklemindeki modülo-üstel işlemini yapacak olan RSA bloğuna *X* olarak verir.

$$Y = a^X mod q \tag{5.1}$$

a ve q değerlerinin nasıl seçileceğinden daha önceki bölümlerde bahsedilmişti. q değeri 128 bitlik bir asal sayıdır. RSA bloğunun hesapladığı Y değeri merkezi işlem birimi tarafından kart okuyucuya gönderilir. Kart okuyucudan alınan değer, q ve X aynı olmak üzere, a değeri yerine RSA bloğuna gönderilerek şifrelemede kullanılacak anahtar elde edilir. Bundan sonra bu seans boyunca kart okuyucuya gönderilecek her mesaj hesaplanan anahtar kullanılarak AES bloğuna şifreletilir. Şifrelenen mesaj okuyucuya gönderilir. Okuyucu da aynı anahtarı kullanarak aldığı her mesajı çözer. Şekil 5.7'de kart ve okuyucu tarafından güvenli bir haberleşme kanalının kurulması ve veri alışverişini anlatan bir akış diyagramı görülmektedir.

Şekil 5.2'de merkezi işlem biriminin Active-HDL'de üretilen blok halindeki görüntüsü bulunmaktadır.



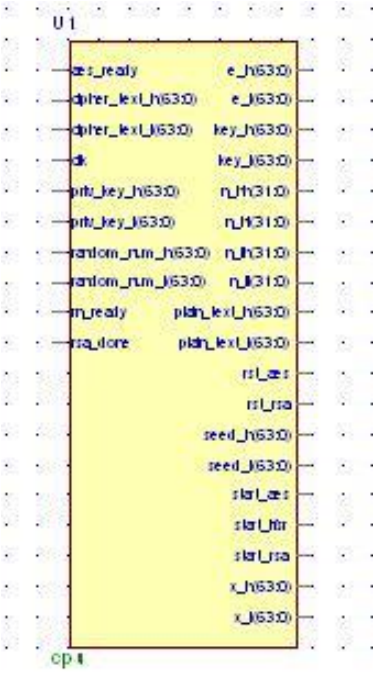
Şekil 5.1 Tasarlanan akıllı kartın blok diyagramı

5.2 Rastgele Sayı Üretici (LFSR)

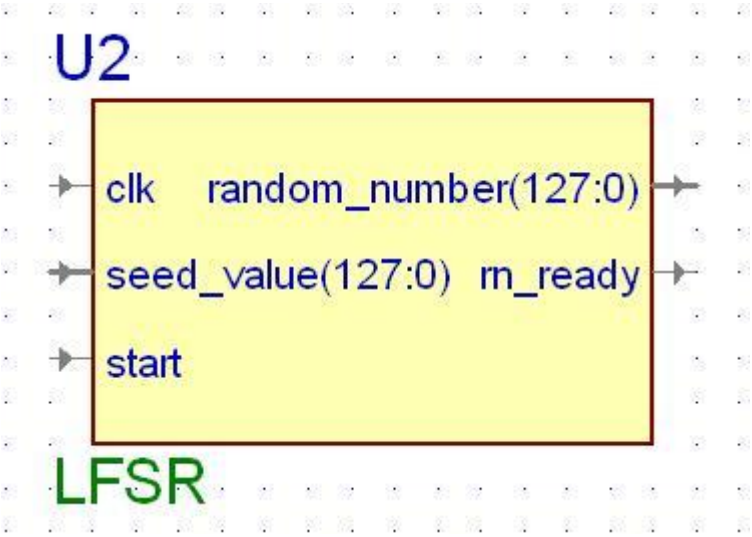
Şifrelemede kullanılacak anahtarı oluşturmak için rastgele sayıya ihtiyaç duyulduğundan daha önceki bölümlerde bahsedilmişti. Rastgele sayı üretici olarak bir LFSR bloğu gerçeklenmiş ve sistemde kullanılmıştır. Yine daha önceki bölümlerde LFSR'ın periyot büyüklüğünün üretilen rastgele sayının tahmin edilme zorluğuyla ilişkili olduğundan bahsedilmişti. Tasarımda 128 bitlik rastgele sayıya ihtiyaç duyulduğundan 128 bitlik LFSR için maksimum periyot büyüklüğünü verecek geri-besleme polinomunun (5.2) denklemindeki gibi olduğu [17] numaralı kaynaktan alınmıştır.

$$L(1) = L(128)\ XNOR\ L(126) XNOR\ L(101) XNOR\ L(99) \quad (5.2)$$

Gerçeklenen LFSR'da kaydedicinin en düşük anlamlı bitine her saat darbesinde 128, 126, 101, 99 numaralı bitler XNOR işlemine tabi tutulup, elde edilen sonuç yazılmaktadır. En yüksek anlamlı bit haricindeki tüm bitler bir üst bite kaydırılmaktadır. Şekil 5.3'te LFSR'ın Active-HDL'de üretilen blok gösterilimi görülmektedir.



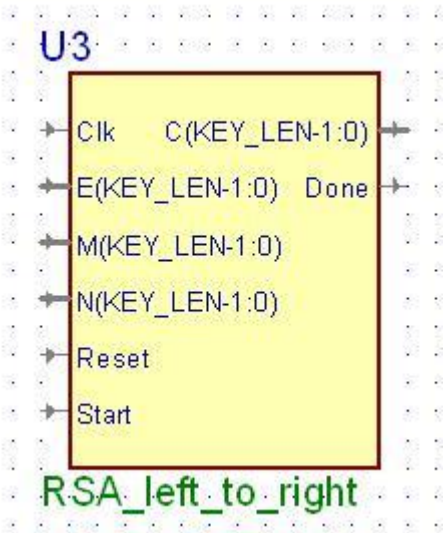
Şekil 5.2 Merkezi işlem birimi blok gösterilimi



Şekil 5.3 Sözde-rastgele sayı üretici blok gösterilimi

5.3 Anahtar üretimi (RSA)

Bu blok [7] numaralı tezden hazır olarak alınıp projede kullanılmıştır. Blok alındıktan sonra analiz edilmiştir. Blok 8 bitlik çalışma modunda yazılmış olduğundan 128 bite çevirmek için üzerinde küçük değişiklikler yapılmıştır. Bu blok (5.1) denklemindeki modülo-üstel işlemini gerçekleştirmektedir. Şekil 5.4'te RSA modülünün Active-HDL'de oluşturulan blok gösterilimi görülmektedir. Şekilde görülen *E, M, N* girişleri sırasıyla, modülo-üstel hesaplamada kullanılan, mod, üst ve taban değerleridir.



Şekil 5.4 RSA modülünün Acvite-HDL'de blok gösterilimi

5.4 Şifreleme (AES)

AES bloğu [4] numaralı tezden hazır olarak alınıp projede kullanılmıştır. Blok alındıktan sonra analiz edilmiştir. Karttan dışarıya gönderilecek bilgilerin şifrelenmesinde kullanılmıştır. Şekil 5.5'de AES bloğunun Active-HDL'de üretilen blok gösterilimi verilmiştir.

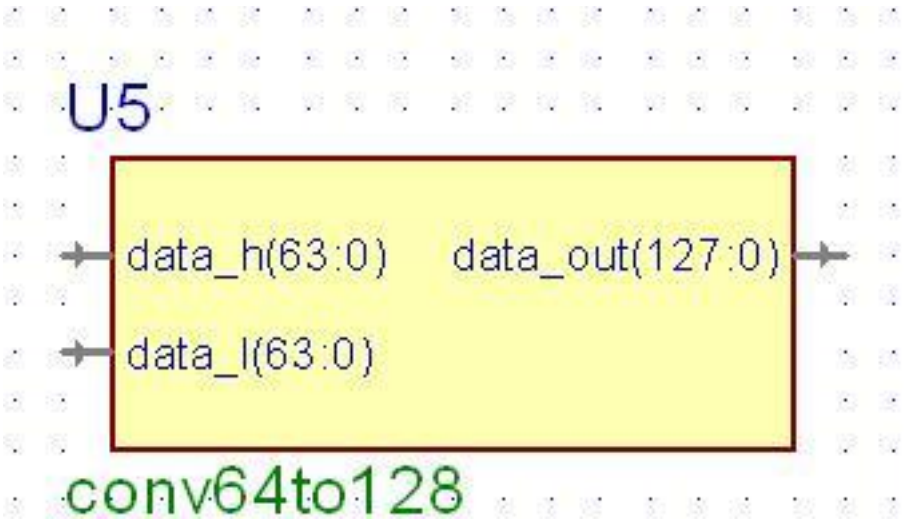
5.5 SystemC-VHDL Ara Yüzleri

SystemC'de 128 bitlik veri tipi kullanılamadığından ve tüm VHDL'de yazılmış modüllerde 128 bitlik giriş ve çıkışlar kullanıldığından VHDL'de 64 bitten 128 bite, 32 bitten 128 bite çevirme ve tam tersi işlemleri yapan modüller yazılıp tasarımda kullanılmıştır. Şekil 5.6'da 64 bitten 128 bite çeviren ara yüzün blok diyagramı verilmiştir. *key* ve *plaintext* girişleri sırasıyla anahtar ve şifrelenecek mesajı alan girişlerdir. *ciphertext* çıkışı ise şifrelenmiş mesajı veren çıkıştır.

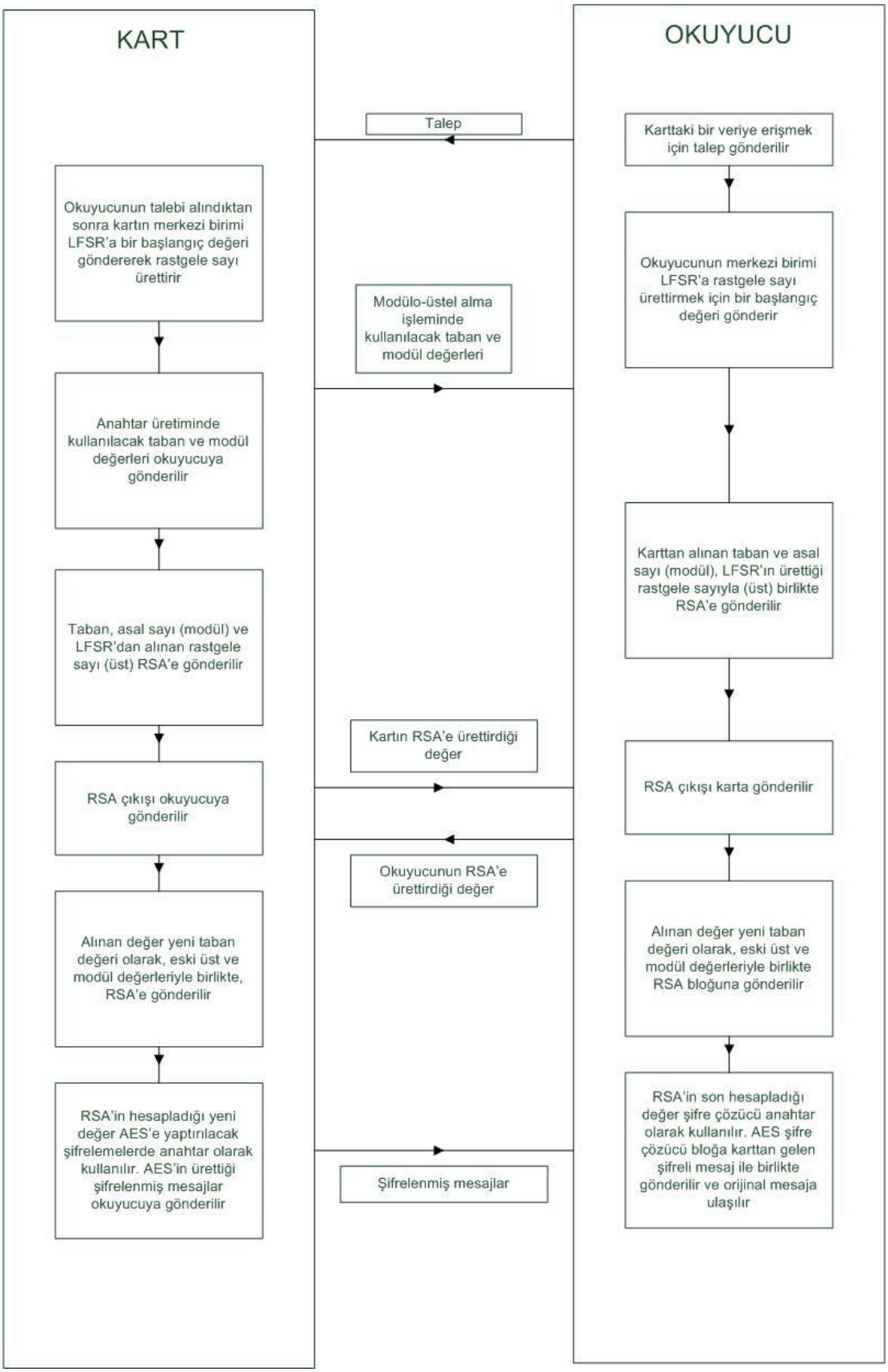
5.6 Üst Modül

En üst modül sistemin alt blokları arasındaki bağlantıları yapmak için yazılmıştır.

Şekil 5.5 AES modülünün Acvite-HDL'de blok gösterilimi



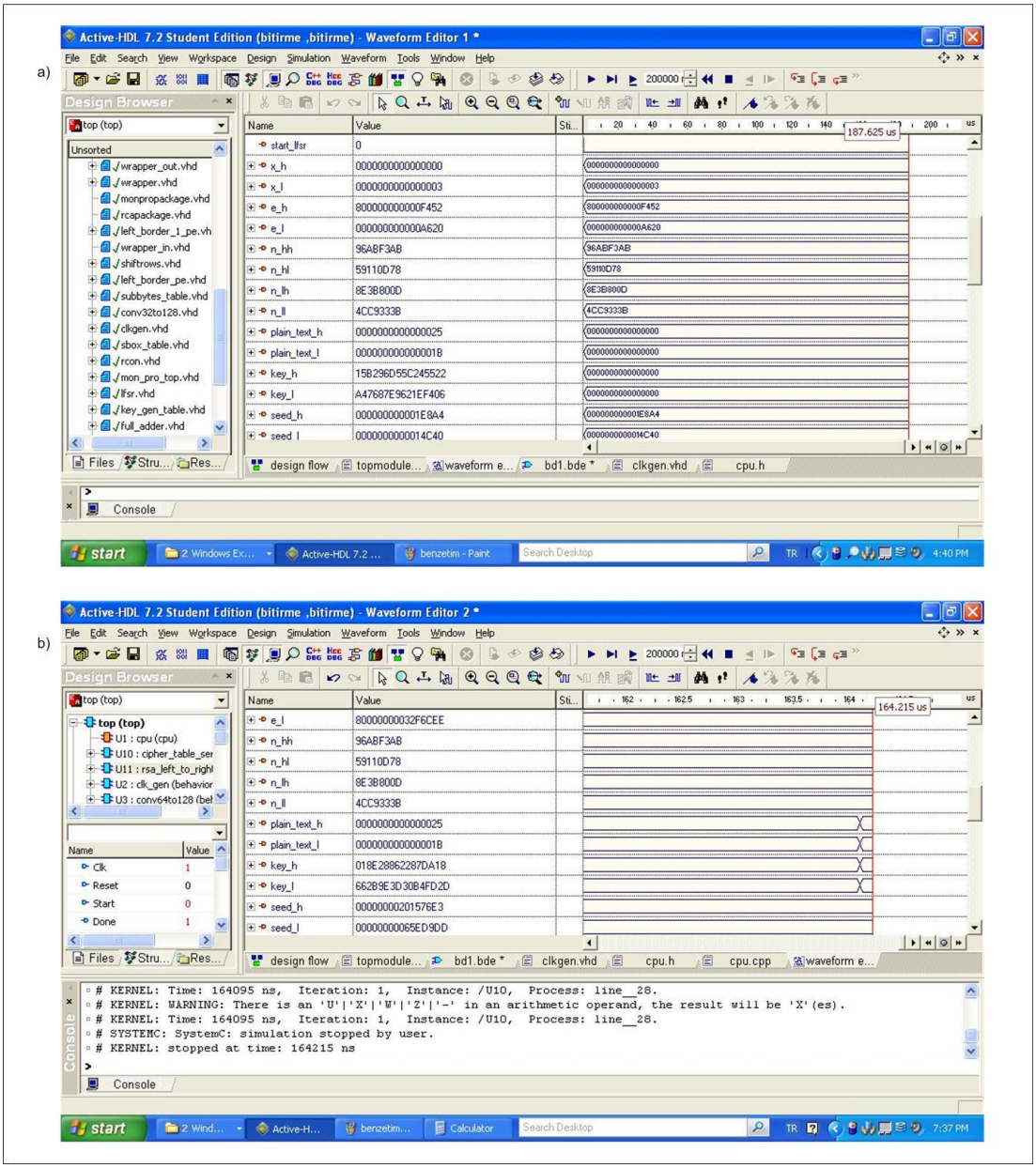
Şekil 5.6 64 bitten 128 bite dönüşüm yapan SystemC-VHDL ara yüz modülünün Acvite-HDL'de blok gösterilimi



Şekil 5.7 Kart ve okuyucu arasında veri alışverişi akış diyagramı

6. BENZETİM SONUÇLARI

Tasarlanan sistemin fonksiyonel benzetimi Active-HDL'de yapılmıştır. Kart okuyucudan bir haberleşme talebi geldiğinde sistem çalışmaya başlamakta ve güvenli haberleşme kanalı kurulmaktadır. O seans için üretilen anahtar kullanılarak 128 bitlik bir mesaj şifrelenmekte ve benzetim sonlanmaktadır. Şekil 6.2'de benzetim sonucu alt modüllerin giriş ve çıkışlarının bir kısmında gözlemlenen dalga formları görülmektedir.



Şekil 6.1 Farklı anahtarlar kullanılarak yapılan benzetimlerin sonuçları

Sistem farklı şifrelenecek mesaj sabit tutulup farklı anahtarlar kullanılarak çalıştırıldığında yapılacak işlemlerin tamamlanma sürelerinin değiştiği gözlemlenmiştir. Bu değişiklik RSA bloğundan kaynaklanmaktadır. RSA algoritmasının modülo-üstel alma işlemini tamamlama süresi kullanılan üstün değerine göre değişmektedir. Anahtar sabit tutulup şifrelenecek mesaj değiştirilerek benzetim yapıldığında ise benzetimin tamamlanma süresinde hiçbir değişiklik olmadığı görülmüştür. Bu da AES bloğunun kullanılan mesaj ve anahtara bağlı olarak işlem süresinin değişmediğini göstermiştir. Kullanılan anahtar ve şifrelenecek mesaj değiştirilerek 3 farklı durum için benzetim yapılmış ve benzetimin tamamlanması için geçen saat periyodu sayısı hesaplanarak Tablo 6.1 oluşturulmuştur. Birinci durumda *K1* anahtarı ile *P1* mesajı, ikinci durumda *K2* anahtarı ile *P1* mesajı, üçüncü durumda *K2* anahtarı ile *P2* mesajı kullanılmıştır.

Tablo 6.1 Farklı anahtar ve mesajlar kullanılması halinde işlemlerin gerçekleşme süresi

	Anahtar Üretme Süresi (RSA)	Mesaj Şifreleme Süresi (AES)	Sistemin Bütün İşlemleri Tamamlama Süresi
Birinci Durum (*K1* ve *P1*)	18747 saat periyodu	100 saat periyodu	18762 saat periyodu
İkinci Durum (*K2* ve *P1*)	16406 saat periyodu	100 saat periyodu	16432 saat periyodu
Üçüncü Durum (*K2* ve *P2*)	16406 saat periyodu	100 saat periyodu	16432 saat periyodu

7. SONUÇLAR

Yapılan bitirme projesinde hedeflenen güvenli bir akıllı kart sistemi tasarımı başarıyla tamamlanmış ve sistemin bütün olarak donanım-yazılım ortak benzetimi yapılarak fonksiyonelliği test edilmiştir. Sistemin benzetimi teknolojiden bağımsız olarak yapılmıştır. Bu aşamadan sonra ihtiyaçlara uygun teknoloji ve araçlar (FPGA ve mikroişlemci gibi) seçilerek donanımsal ve yazılımsal parçalar gerçeklenebilir.

Sistem hedeflenen veri saklama ve güvenli haberleşme görevlerini yerine getirmektedir. Mevcut haliyle sistem, kart okuyuculardan gelen iletişim taleplerini okuyucuları doğrulamadan kabul etmektedir. İleride bu çalışmaya kimlik doğrulama özelliği de eklenerek sistem geliştirilebilir.

[13] numaralı kaynakta belirtildiği gibi günümüzde büyüyen ve karmaşıklaşan mühendislik sistemlerinin tasarımı, zaman kısıtlaması ve ayrı tasarlanan parçaların bir arada çalışmaması halinde ortaya çıkabilecek mali kayıpların önüne geçilmesi için, sistem düzeyinde tasarım, benzetim ve test etme ihtiyaçlarını beraberinde getirmektedir. Bu çalışmada da sistem düzeyinde tasarım yöntemi araştırılmış ve donanım-yazılım ortak benzetimi gerçekleştirilmiştir.

Bu bitirme çalışması daha önce gerçeklenmiş olan benzer sistemlerle ilgili raporlar da incelenerek yapılmıştır. Tamamlanmış çalışmaların incelenmesi daha önce karşılaşılmış problemler hakkında fikir edinmek bakımından faydalı olmuştur.

Kişisel bilgilerin güvenliği, sağlık harcamalarının maliyetini, sağlık alanında karşılaşılan kötüye kullanım ve sahtecilik miktarı, sağlık hizmetlerinin kalitesi gibi kıstaslar göz önünde bulundurulduğunda akıllı kartlı sistemlerin geleneksel sistemlere göre avantajlı olduğu yapılan araştırmalarca gösterilmiştir [1, 2]. Akıllı kartların sağlık sistemi dışında toplu ulaşım, banka kartları gibi çeşitli alanlarda da kullanımı yaygındır. Akıllı kartların günümüzde yaygın olarak kullanılması nedeniyle yapılan bitirme çalışması önem kazanmaktadır.

KAYNAKLAR

[1] **Smart Card Alliance,** 2009. A Healthcare CFO's Guide to Smart Card Technology and Applications, *A Smart Card Alliance Healthcare Council Publication,* Princeton Junction NJ.

[2] **Smart Card Alliance,** 2003. HIPAA Compliance and Smart Cards: Solutions to Privacy and Security Requirements, *A Smart Card Alliance Report,* Princeton Junction, NJ.

[3] **Delfs, H. ve Knebl, H.,** 2007. Introduction to Cryptography: Principles and Applications, Springer-Verlag, Heidelberg.

[4] **Ordu, L.,** 2006. AES Algoritmasının FPGA Üzerinde Gerçeklenmesi ve Yan Kanal Analizi Saldırılarına Karşı Güzlendirilmesi, *Yüksek Lisans Tezi*, İTÜ Fen Bilimleri Enstitüsü, İstanbul.

[5] **Diffie, W. ve Hellman, M.E.,** 1976. New Directions in Cryptography, *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, sf. 644-654.

[6] **Mollin, R.A.,** 2003. RSA and Public-Key Cryptography, CRC Press LLC, Florida.

[7] **Bayhan, D.,** 2010. RSA Açık Anahtarlı Kriptosisteminin FPGA Üzerinde Düşük Güçlü Tasarımı ve Gerçeklemesi, *Yüksek Lisans Tezi*, İTÜ Fen Bilimleri Enstitüsü, İstanbul.

[8]**Menezes, A.J., Van Oorschot, P.C. ve Vanstone, S.A.,** 1997. Handbook of Applied Cryptography, CRC Press.

[9] **Effing, W. ve Rankl, W.,** 2003. Smart Card Hanbook, John Wiley & Sons, West Sussex.

[10] **Smart Card Alliance,** 2005. The Taiwan Healthcare Smart Card Project, Princeton Junction, NJ.

[11] **Bailey B., Martin G. ve Piziali A.,** 2007. ESL Design and Verification: A Prescription for Electronic System-Level Methodology, Elsevier, San Francisco, CA.

[12] **Thomas D.E., Adams J.K. ve Schmit H.,** 1993. A Model And Methodology For Hardwre-Software Co-Design, IEEE Design & Test of Computers.

[13] **Black, D.C., Donovan, J.,** 2004. SystemC From the Ground Up, Kluwer Academic Publishers, Boston.

[14] **Grötker, T., Liao, S., Martin, G. ve Swan, S.,** 2002. System Design with SystemC, Kluwer Academic Publishers, Dordrecht.

[15] **De Micheli, G., Ernst, R. ve Wolf, W.,** 2002. Readings in Hardware/Software Co-Design, Academic Press, USA, sf. 412-426.

[16] **Aldec,** 2011 http://www.aldec.com/activehdl/.

[17] **Alfke, P. ve George, M.,** 2007. Linear Feedback Shift Registers in Virtex Devices, *Xilinx Inc. Application Note*.