İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

Hafif Bir Kripto Algoritması Olan Boron'un Fpga Üzerinde Mikroişlemci İle Beraber İlk Kez Gerçeklenmesi

BİTİRME ÖDEVİ

Burak Acar

040130030

Bölümü: Elektronik ve Haberleşme Mühendisliği Bölümü
Programı: Elektronik ve Haberleşme Mühendisliği

Danışmanı: Doç. Dr. Sıddıka Berna ÖRS YALÇIN

MAYIS 2017

ÖNSÖZ

Öncelikle bitirme projem boyunca bana her hafta değerli vaktini ayıran, beni çok yeni alanlarda araştırmaya teşvik eden, karşılaştığım sıkıntılarda benden desteğini biran olsun esirgemeyen saygıdeğer hocam Doç. Dr. Sıddıka Berna ÖRS YALÇIN'a sonsuz teşekkürlerimi sunmayı bir borç bilirim.

İstanbul Teknik Üniversitesi lisans hayatımda eğitimime katkı sağlayan, bakış açıma vizyon katan tüm hocalarıma teşekkür ederim.

Okula geldiğimde bana işte doğru yerdeyim dedirten, hayatımdaki ilk ledi yaktığım, çok şeyler öğrendiğim OTOKON kulübüne ve bana farklı disiplinler arası çalışmayı, takım olmayı öğreten İTÜ Hedef Takımı'na ve çok değerli arkadaşlarıma teşekkür ederim. İyi ki varsınız.

Son olarak, hayatım boyunca olduğu gibi bitirme projemde de benden sevgisini esirgemeyen, sürekli destek veren canım aileme sonsuz minnettarlığımı sunarım.

MAYIS 2017 BURAK ACAR

İÇİNDEKİLER **Sayfa**

KISALTMALAR

IoT : Internet of Things
FPGA : Field Programmable Gate Array
ASIC : Application Specific Integrated Circuits
HDL : Hardware Description Language
UART : Universal Asynchronous Receiver Transmitter
ARM : Advanced RISC Machine
DDoS : Distributed Denial of Service attack
AES : Advanced Encryption Standard
DES : Data Encryption Standard
GE : Gate Equivalent
SPN : Substitution-Permutation Network
LUT : Look-up Table
VHDL : Data Encryption Standard
XOR : Exclusive Or
SDK : Software Development Kit
IP : Intellectual Property
AXI : Advanced eXtensible Interface

TABLO LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

HAFİF BİR KRİPTO ALGORİTMASI OLAN BORON'UN FPGA ÜZERİNDE MİKROİŞLEMCİ İLE BERABER İLK KEZ GERÇEKLENMESİ

ÖZET

Nesnelerin İnterneti (IoT), akıllı ev teknolojileri ve giyilebilir teknolojiler gibi yeni ortaya çıkan alanlar beraberinde güvenlik sorunlarını ortaya çıkarmıştır. Bu teknolojilerin gelecekte uygulanabilmesi için üretilen verilerin gizliliğine dikkat edilmelidir. Bunu sağlamanın en iyi yolu ise kriptografiyi kullanmaktır. Kriptografi ile birlikte bilgi, anlaşılamaz bir forma dönüştürülerek sahip olduğu anlam gizlenmiş ve korunmuş olur. Geçmişte, cihazların güvenliğini sağlamak ve kişisel verileri korumak için birçok şifre tasarlanmıştır. Bu şifrelerin donanımsal olarak büyük yer kaplaması ve yüksek güç tüketimine sahip olması sınırlı kaynaklara sahip gömülü sistemlerde uygulanmasını imkânsız hale getirmiştir. Tüm bu kısıtlamalar Hafif Kritografi alanının ortaya çıkmasına neden olmuştur.

Bu proje kapsamında, literatürdeki az yer kaplayan, düşük güç tüketimli kripto algoritmaları araştırılmış, henüz çok yeni bir algoritma olan Boron en uygun algoritma olarak belirlenmiştir. Daha sonra Boron kripto algoritması FPGA üzerinde Verilog donanım tasarlama dili (HDL) ile gerçeklenmiştir. Donanımın tamamen çalışır durumda olduğu test edildikten sonra tüm sistemi kontrol etmek için ARM işlemci üzerinde C dili ile yazılım tasarımı yapılmıştır. Son olarak donanım ve yazılım tasarımı yapılan Boron şifresi Sahada Programlanabilir Kapı Dizinleri (FPGA) üzerinde gerçeklenmiş çıktıları seri haberleşme protokolüyle (UART) beraber ekrana yazdırılmıştır.

FIRST TIME IMPLEMENTATION OF A LIGHTWEIGHT CRYPTO ALGORİTHM BORON ON FPGA WITH A MICROPROCESSOR

SUMMARY

New areas such as Internet of Things (IOT), smart home technologies and wearable technologies have brought security problems together. In order for these technologies to be implemented in the future, attention should be paid to the confidentiality of the produced data. In order for these technologies to be implemented in the future, attention should be paid to the confidentiality of the produced data. The best way to achieve this is to use cryptography. With cryptography, information is transformed into an inexplicable form, meaning that it is hidden and protected. In the past, many crypto algorithm were designed to secure the devices and protect personal data. These ciphers have a large hardware footprint and high power consumption, making them impossible to implement in embedded systems with limited resources. All these constraints led to the emergence of the field of Lightweight Cryptography.

Within the scope of this project, low capacity crypto algorithms which have a small footprint in the literature have been searched and Boron which is a very new algorithm has been determined as the most suitable algorithm. Then the Boron crypto algorithm was implemented on the FPGA with the Verilog hardware design language (HDL). After testing that the hardware is fully functional, the software is designed with the C language on the ARM processor to check the entire system. Finally, the output of the hardware and software-designed Boron code on the Field Programmable Gate Array (FPGA) is printed on the screen together with the serial communication protocol (UART).

1. GİRİŞ

Nesnelerin İnterneti (IoT), akıllı ev teknolojileri ve giyilebilir teknoloji gibi yeni ortaya çıkan alanlar insan hayatını kolaylaştırmak için ileride çok önemli bir rol oynayacaktır. Bu teknolojiler sayesinde insanlar her gün kullandığı cihazlara sürekli bağlı olacak, sensör ağlarından çok fazla miktarda veri toplanacak ve bu verilerin analizi ile birlikte günlük yaşamlarındaki problemlerine çözüm üretilecektir. Bu durum aynı zamanda, her bireyin kişisel bilgisinin aynı ortam üzerinde bulunması ve belirli yöntemlerle herkes tarafından erişilebileceği anlamına gelir. Her cihazı akıllı hale getirip internete bağlamak bilgi sızıntısı, kişisel verilerin ele geçirilmesi ve analiz edilmesi, Dağıtık Hizmet Engelleme (DDoS) saldırıları gibi sorunları beraberinde getirir. Tüm olası saldırı türlerine karşı cihazların güvenlik önlemlerinin alınmış olması gerekir. Bu teknolojilerin gelecekte uygulanabilmesi için üretilen verilerin gizliliğine dikkat edilmelidir. Geçmişte, cihazların güvenliğini sağlamak ve kişisel verileri korumak için birçok şifre tasarlanmıştır. AES [1] [2] ve Triple DES [3] [4] gibi şifreler yaygın olarak kullanılan şifrelerdir. Şimdiye kadar bu şifrelere karşı makul bir sürede başarılı olabilmiş bir saldırı mevcut değildir.

Bu şifrelerin donanımsal olarak büyük yer kaplaması ve yüksek güç tüketimine sahip olması sınırlı kaynaklara sahip gömülü sistemlerde uygulanmasını imkânsız hale getirmiştir. Tüm bu kısıtlamalar Hafif Kritografi alanının ortaya çıkmasına neden olmuştur [5]. Hafif Kriptografi, sağlam bir tasarıma sahip olan, daha az yer kaplayan, düşük güç tüketimi ve kapı eşdeğer (GE) sayısı 2200'den daha az olan şifreleme algoritmaları tasarlamayı amaçlar [6].

Bu projede düşük ölçekli gömülü sistemlerde kullanılabilecek minimum alan, minimum güç tüketimine sahip bir şifreleme donanımı tasarlamak hedeflenmiştir. Literatürdeki hafif algoritmalar incelendiğinde bazında en iyi algoritma olarak Boron belirlenmiş ve FPGA üzerinde donanımsal tasarım işlemleri gerçekleştirilmiştir. Donanımı ve tüm sistemi kontrol etmek amacıyla FPGA geliştirme kartı üzerinde

bulunan ARM işlemci kullanılmıştır. İlk kez Boron kripto algoriması şifreleme-şifre çözme işlevleriyle beraber FPGA üzerinde bir mikroişlemci ile koordineli olarak gerçeklenmiştir.

Tezin ikinci bölümünde önbilgiler olarak gerçeklenecek Boron şifresinin matematiksel algoritması ve donanım-yazılım tasarımı sırasında kullanılan araçlar anlatılmıştır.

Tezin üçüncü bölümünde gerçeklenecek algoritmanın FPGA üzerinde gerçekleme aşamaları tüm detaylarıyla anlatılmıştır.

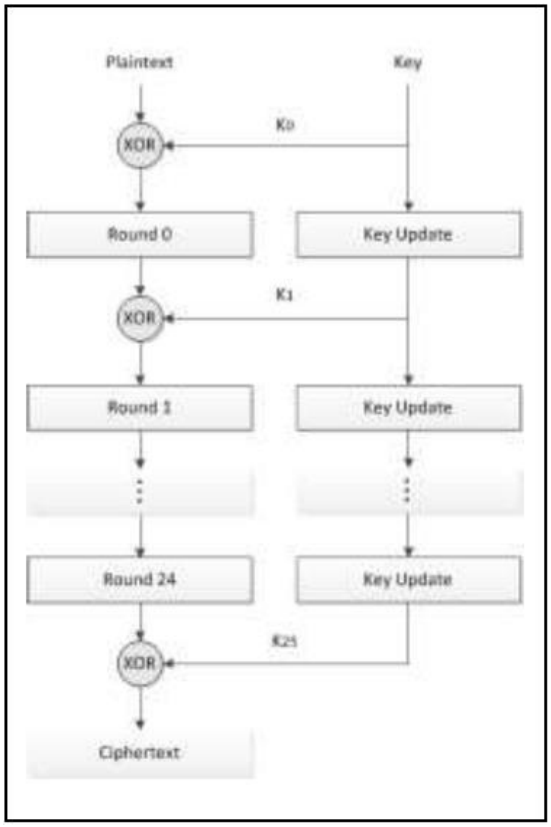
2. ÖNBİLGİLER

2.1 Gerçeklenecek Hafif Kriptografi Algoritması

Bu bölümde BORON algoritmasının özellikleri ve veri şifreleme algoritmaları açıklanmaktadır.

2.1.1 Özellikleri

BORON [7] şifreleme algoritması, hafif bir blok şifre değiştirme-permütasyon ağı (SPN) yapısına sahiptir. Diğer Feistel tabanlı şifrelere nazaran daha hızlı çalışır. Bir bloğu 25 kez tekrarlayan bir şifre olan Boron'un bu özelliği kriptografik olarak güçlü olmasını sağlar. Şifre değiştirme-permütasyon ağında (SPN) doğrusal olmayan katmana sahip olması, aktif S_box sayısında iyi sonuçlar doğurmuş, doğrusal ve diferansiyel şifreleme analizi gibi standart kriptografik saldırılara karşı güvenli olmasını sağlamıştır. Anahtar üretimi PRESENT [9] şifreleme algoritmasının anahtar üretimine dayanıyor. BORON şifreleme algoritması, 64 bitlik bir düz metin veya şifreli metin blok uzunluğuna sahipken, 80 bit veya 128 bitlik şifreleme metni uzunluğunu destekliyor. Şifreleme algoritmasına genel bir bakış Şekil 2-1'de verilmiştir. Daha ayrıntılı özellikler aşağıdaki bölümlerde verilecektir.



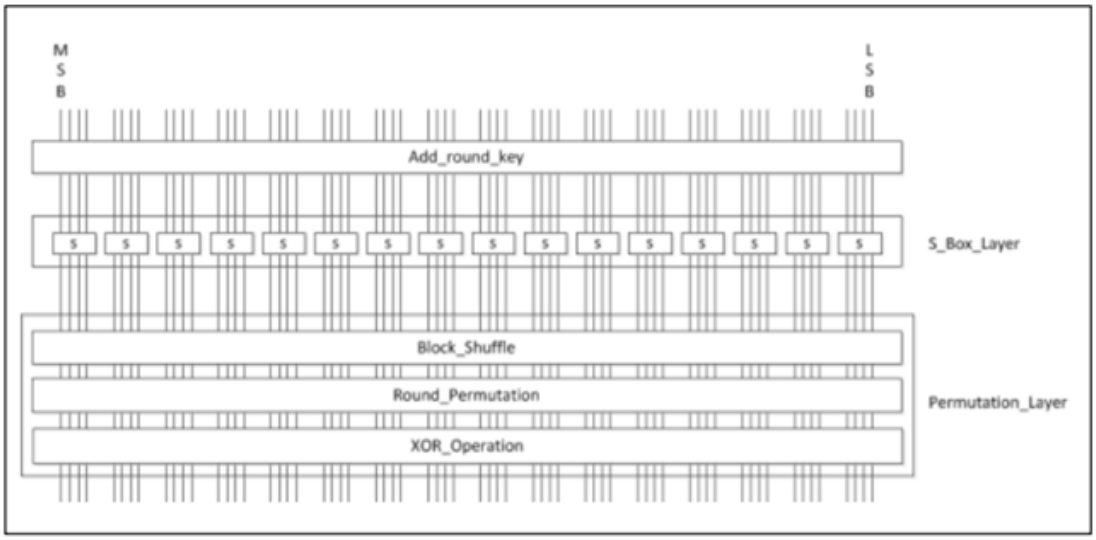
Şekil 2-1 : Boron Round Fonksiyonu [9]

2.1.2 Round Fonksiyonu

Boron şifresi kendini 25 kez tekrar eden Round fonksiyonuna sahiptir. Round fonksiyonu Şekil 2-2'de de gösterildiği gibi anahtar ekleme, doğrusal olmayan permütasyon ve doğrusal katmana sahiptir. Anahtar ekleme katmanı bit düzeyinde anahtar ekleme işlemi gerçekleştirir. Doğrusal olmayan katman S_box katmanıdır. 16 tane paralel çalışan 4 bitlik giriş ve çıkışa sahip s_boxlardan oluşur. Doğrusal katman ise sırasıyla blok karıştırma, round permütasyon ve XOR işlemlerinden oluşur.

Her bir tur (round) aşağıdaki 5 fonksiyona sahiptir:

1. Her bir turun giriş verisinin tur anahtarı ile XOR işlemine sokulması
2. S_Box ile doğrusal olmayan dönüşüm
3. Blok karıştırma
4. Round permütasyonu
5. XOR işlemi



Şekil 2-2 : Round Fonksiyonu İç Yapısı [9]

2.1.2.1 Anahtar Ekleme Katmanı (Add_Round_key)

Anahtar ekleme katmanı şimdiki durum ile tur anahtarının en düşük anlamlı 64 biti arasında basit bir XOR işlemi yapar. Her bir turda anahtar değişir. Anahtar üretimi ile ilgili detaylı bilgiler C bölümünde verilmiştir.

2.1.2.2 Doğrusal Olmayan Katman (S_Box_Layer)

S_Box katmanı 4 bit girişten 4 bit çıkış üreten birbiriyle paralel çalışan toplamda 16 s_boxtan oluşur. Boron algoritmasının şifre değiştirme katmanının her bir değere karşılık gelen çıktısı hexadecimal olarak Tablo 2-1'de verilmiştir.

Tablo 2-1 : S-box tablosu

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	E	4	B	1	7	9	C	A	D	2	0	F	8	5	3	6

2.1.2.3 Doğrusal Katman (Permutaion_Layer)

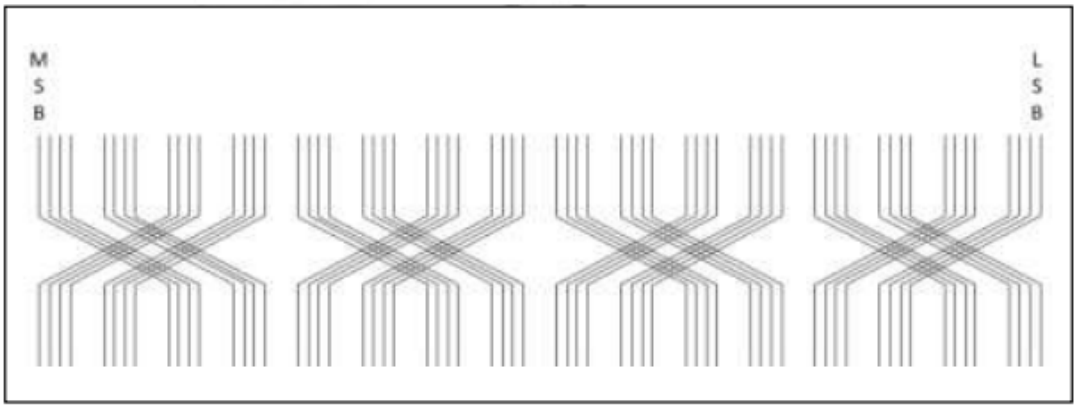
Boron şifresinin doğrusal permütasyon katmanı 3 ara katmana sahiptir.

Blok Karıştırma Katmanı (Block_Shuffle)

Blok karıştırma katmanı 16 bitlik girdisini 8 bit sola çevrimsel kaydırma yaparak 16 bit çıktı üretir. Blok karıştırma işlemi Tablo 2-2'de gösterilmiştir. 16 bitlik girdi önce 4 bitlik bloklara ayrılır. En anlamsız 4 bitlik blok (j=0) 8 bit sola kaydırılarak 3. en anlamsız bloğun yerine geçer. Aynı şekilde 2. en anlamsız blok (j=1) 8 bit sola kayarak en anlamlı bloğun yerine geçmiş olur. Şekil 2-3' ten blok karıştırma katmanı daha rahat bir şekilde anlaşılabilir.

Tablo 2-2 : Blok Karıştırma: 16 bit blokların içinde 8 bit sola çevrimsel kaydırma

j	0	1	2	3
B(j)	2	3	0	1



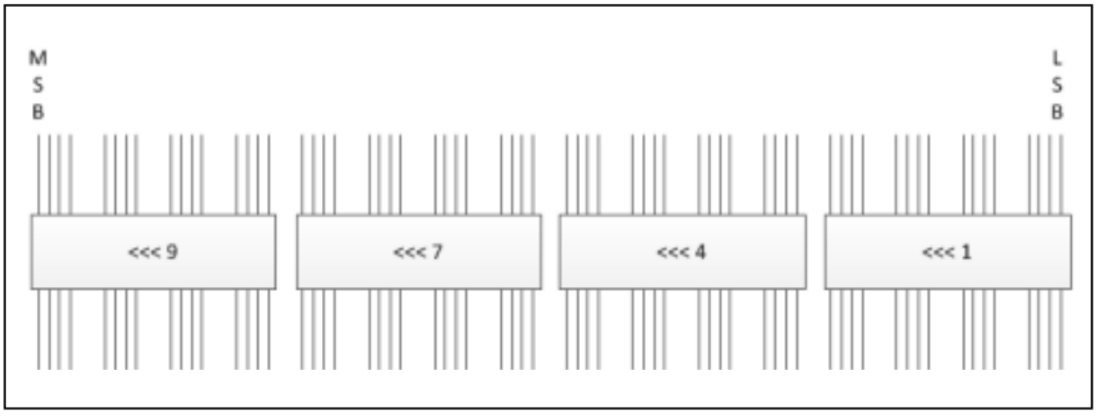
Şekil 2-3 : Blok Karıştırma [9]

Round Permütasyonu (Round_Permutation)

Round Permütasyonu her 16 bitlik bloklara Tablo 2-3'e göre sola kaydırma işlemi uygular. Round Permütasyonu'nun 64 bitlik girişi önce 16 bitlik 4 bloğa ayrılır ve bu 4 bloğa Tablo 2-3 doğrultusunda sola çevrimsel kaydırma işlemi uygulanır.
Şekil 2-4' ten Round Permütasyon katmanı daha rahat anlaşılabilir.

Tablo 2-3 : Round Permütasyonu: sola çevrimsel kaydırma değerleri

j	0	1	2	3
r(j)	1	4	7	9



Şekil 2-4 : Round Permütasyonu [9]

XOR İşlemi Katmanı (XOR_Operation)

XOR İşlemi Katmanı 16 bitlik bloklar arasında basit bir XOR işlemi gerçekleştirir. Aşağıdaki denkleme göre 4 tane 16 bitlik çıktıdan (W'3 W'2 W'1 W'0) toplamda 64 bit çıkış üreten bir katmandır.

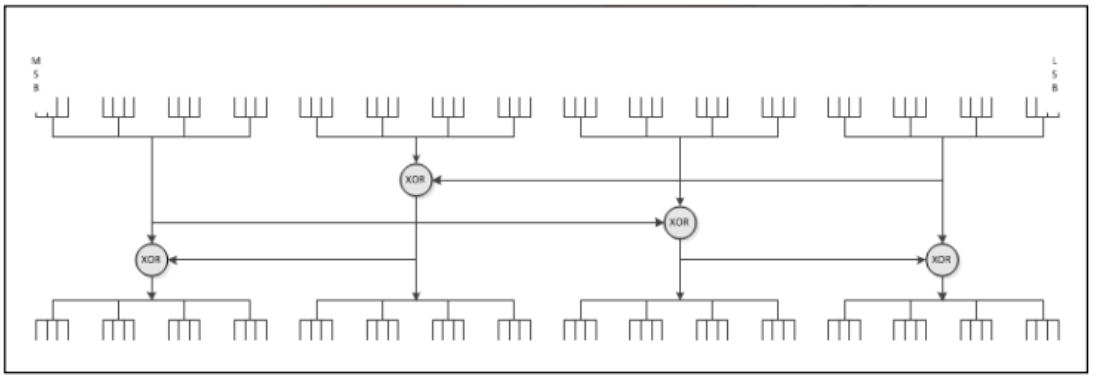
$$W'3 = (W3 \wedge W2 \wedge W0) \quad (1)$$

$$W'2 = (W2 \wedge W0) \quad (2)$$

$$W'1 = (W3 \wedge W1) \quad (3)$$

$$W'0 = (W3 \wedge W1 \wedge W0) \quad (4)$$

" ' " çıkışları, "W" 16-bit girişleri ve "^" XOR işlemini simgeler.



Şekil 2-5 : XOR İşlemi Katmanı [9]

2.1.3 Anahtar Üretimi (Key_Schedule)

Boron anahtar üretimi hafif bir kripto algoritması olan PRESENT ile çok benzerdir. Şimdiye kadar PRESENT anahtar üretimine karşı bir saldırı yayınlanmamıştır. Boron şifreleme için 26 tane 64 bitlik ara anahtar kullanır. Bu anahtarların üretimi 128 bitlik veya 80 bitlik bir kullanıcı tanımlı ana anahtarın (Master_Key) anahtar üretimi fonksiyonuna sokulup çıkışın en anlamsız 64 bitinin alınmasıyla gerçekleşir. Üretilen anahtarlar sırasıyla round fonksiyonuna dâhil olurlar. Burada dikkat edilmesi gereken konu ilk turun anahtarı (round_key[0]) üretilmeyip kullanıcı tarafından tanımlanmış ana anahtarın en anlamsız 64 bitinin kullanılması. Dolayısıyla anahtar üretimi fonksiyonunun ilk üretilmiş anahtarı (round_key[1]) ikinci turda dâhil olacak. Sonuç olarak Boron şifreleme algoritması şifreleme için gerekli olan 26 anahtarın (K0-K25) ilkini (K0) ana anahtarın son 64 bitinden diğer anahtarları da anahtar üretimi fonksiyonundan karşılar. Bu tez çalışmasında sadece 80 bitlik ana anahtardan anahtar üretimi fonksiyonu kullanılmıştır.

Kullanıcı tarafından tanımlanmış 80 bit uzunluğunda ana anahtar (Master_Key) KEY_Register'da kayıt edilir. Bu kaydın en anlamsız 64 biti her turun ara anahtarı (round_key) olarak kullanılır. Boron'un anahtar güncellemesi aşağıdaki algoritmaya göre yapılır. Önce Key_Register 13 bit sola doğru çevrimsel kaydırılır. Daha sonra bu anahtar kaydının en anlamsız 4 biti S_box'a sokulurak güncellenir. Son olarak bu anahtar kaydının 59. ve 63. bitleri arasındaki 4 bit kaçıncı turda ise o sayının 4 bitlik ikilik tabandaki (binary) haliyle XOR'lanarak güncellenir.

KEY_Register = K79K78...K2K1K0
round_key = K63K62...K2K1K0
1. KEY_Register <<<13
2. [K3K2K1K0] ←S [K3K2K1K0].
3. [K63K62K61K60K59] ← [K63K62K61K60K59] ^ RCi
<<<n sola doğru n bitlik çevrimsel kaydırmayı ve RCi i. turdaki tur sayacının değerini ifade eder.

2.1.4 Şifreleme-Şifre Çözme İşlemi

Boron şifrelemesi 0'dan 24'e kadar toplamda 25 kez tekrar eden yukarıda 2.1.2 de anlatılan tur fonksiyonundan (Round) oluşur. Her bir turun giriş anahtarı 2.1.3 bölümünde bahsedilen anahtar üretim fonksiyonu (Key_schedule) ile oluşturulur. Şekil 6'dan da görülebileceği üzere son tur fonksiyonun çıktısı 25. tur anahtarı ile XOR'lanarak şifrelenmiş metin oluşturulur. Boron şifreleme algoritmasının basit kodu aşağıdaki gibi verilebilir.

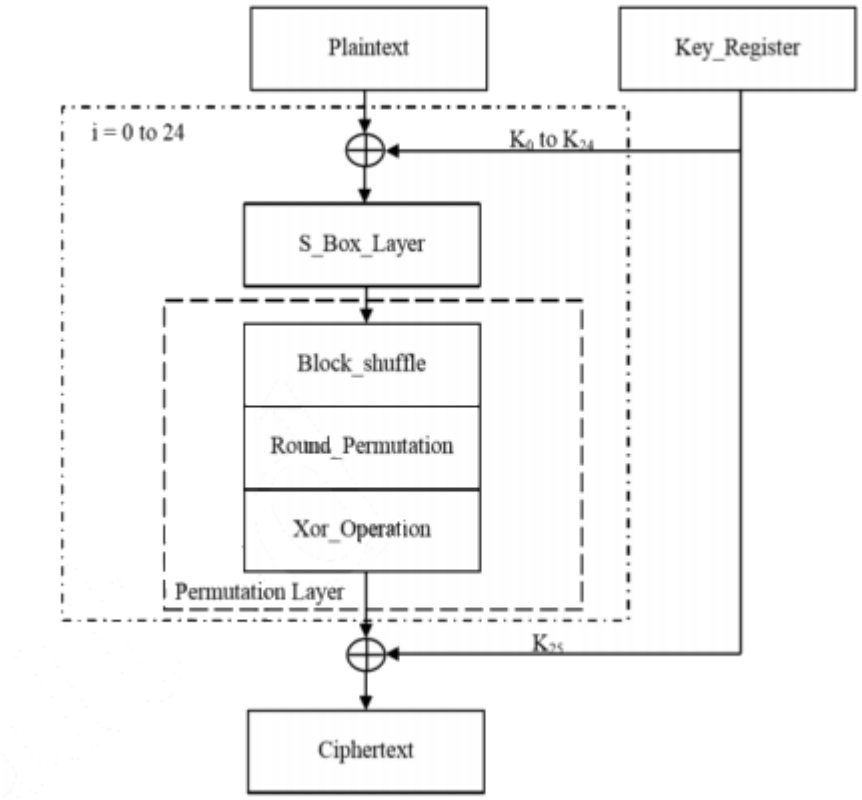
Input: plaintext, Master_key;
Output: ciphertext;

```
BORON-encrypt(plaintext, Master_key)
{
        state = plaintext;
        Key_Schedule(key);
        for (i=0; i<=24; i++) Round(state, round_key[i]);
        ciphertext = Add_round_key(state, round_key[25]);
}
```

```
Key_Schedule(Master_key)
{
        KEY_Register = Master_key;
        for (i=0; i<=25; i++)
        {
                KEY_Register<<<13;
                [K3K2K1K0] = S_Box([K3K2K1K0])
                [K63K62K61K60K59] = [K63K62K61K60K59] ^ i
                round_key[i] = KEY_Register[K63K62....K1K0];
        }
}

Round(state, rk[i])
{
        Add_round_key_Layer(state, rk[i]);
        S_Box_Layer(state);
        Block_shuffle(state);
        Round_Permutation(state);
        XOR_Operation(state);
}
```

Burada Master_key, 80 bitlik giriş şifreleme anahtarı, state ara turlardaki 64 bit veri bloğu ve cipher son turdan sonra elde edilen şifreli metni ifade eder. KEY_register ara anahtarların saklandığı 80 bitlik değişken ve round_key [i], i. turdaki 64-bit tur anahtarıdır. Bu algoritma hakkında detaylı bilgiler algoritmanın yayınlanmış orijinal metninde mevcuttur[3].



Şekil 2-6 : Boron İç Yapısı

BORON şifre çözme işlemi şifrelemede yapılan işlemlerin tam tersidir. Şifre çözme algoritmasının basit kodu aşağıdaki gibi verilebilir.

Input: ciphertext, Master_key;
Output: plaintext;

```
BORON-decrypt(ciphertext, Master_key)
{
        Key_Schedule(Master_key);
        state = Add_round_key(ciphertext, round_key[25]);
        for (i=0; i<=24; i++) Round(state, round_key[24-i]);
        plaintext = state;
}
```

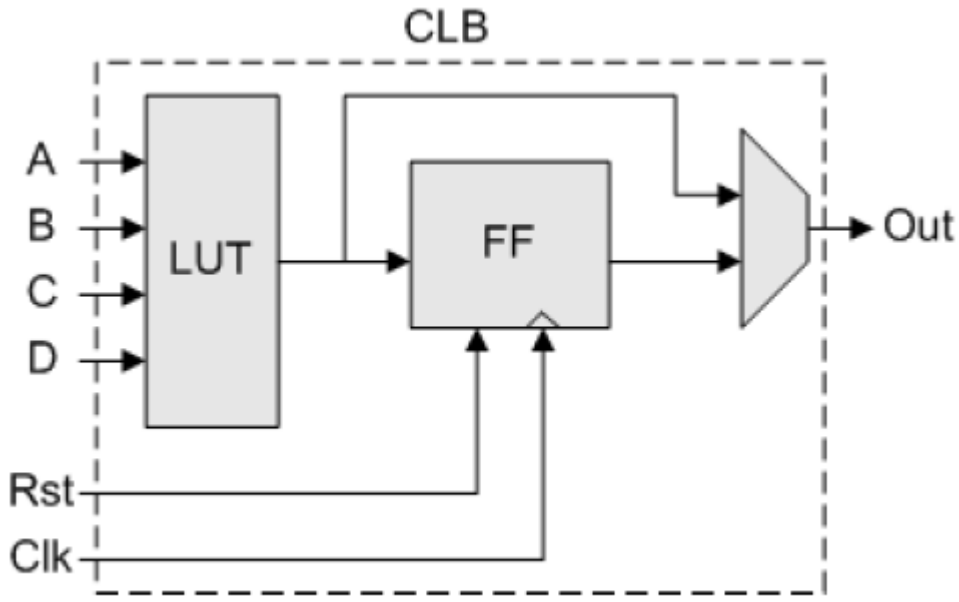
```
Key_Schedule(Master_key)
{
        KEY_Register = Master_key;
        for (i=0; i<=25; i++)
        {
                KEY_Register<<<13;
                [K3K2K1K0] = S_Box([K3K2K1K0])
                [K63K62K61K60K59] = [K63K62K61K60K59] ^ i
                round_key[i] = KEY_Register[K63K62....K1K0];
        }
}

Round(state, rk[i])
{
        XOR_Operation_dec(state);
        Round_Permutation_dec(state);
        Block_shuffle_dec(state);
        S_Box_Layer_dec(state);
        Add_round_key_Layer(state, rk[i]);
}
```

2.2 Sahada Programlanabilir Kapı Dizileri

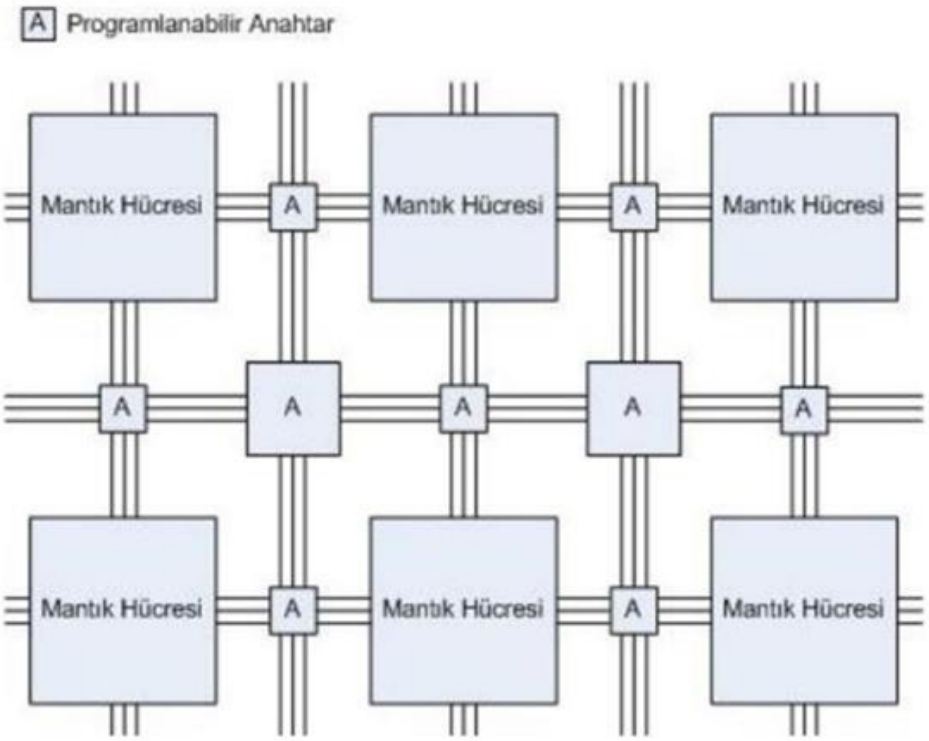
Sahada Programlanabilir Kapı Dizinleri (FPGA) üretim sonrasında herhangi bir sayısal fonksiyonu gerçekleştirebilmek için kullanıcı tarafından programlanabilen yarı iletken devrelerdir [10]. Programlanabilir mantık hücrelerinin ve yönetilebilir anahtarların iki boyutlu olarak dizilmesi ile oluşmuşlardır. Mantık hücreleri basit bir fonksiyonu gerçeklemek üzere yapılandırılabilir. Bunun yanında programlanabilir anahtarlar ile mantık hücreleri arasında bağlantılar kurulabilir. Sayısal donanımlar mantık hücreleri ve anahtarların bu şekilde programlanmasıyla gerçeklenir. Verilog, VHDL gibi donanım tanımlama dilleri kullanılarak devrenin tasarımı yapıldıktan ve sentezlenmesinin ardından istenilen lojik hücre ve anahtar yapılandırılmasının yer aldığı veri dizisi bir kablo yardımıyla FPGA'e gömülerek devre gerçeklenmiş olur [11].

Mantık hücreleri Şekil 2.7'de gösterildiği üzere programlanabilir kombinezonsal devre ve bir adet D tipi flip-flop içerir.



Şekil 2-7 : Mantık Hücresi Yapısı [11].

İçyapısı Şekil 2.7'de gösterilen mantık hücrelerinden ve programlanabilir ara bağlantılardan oluşan FPGA'in genel yapısı Şekil 2.8'de gösterilmektedir.



Şekil 2-8 : FPGA İç Yapısı [11].

Mantık hücrelerindeki LUT (Look-up Table) yapılandırılabilen kombinezonsal devreleri gerçeklemek için kullanılmaktadır. LUT'lar hangi girişe hangi çıkışın üretileceğini bir tablodan okuyan bir bellek elemanıdır. N girişli bir LUT 2^N boyutlu bellek elemanına karşılık gelmektedir. Çok sayıda LUT elemanı yan yana dizilerek daha karmaşık kombinezonsal fonksiyonlar gerçekleştirilmesine imkân tanır [12].

FPGA temel olarak içinde bulundurduğu elemanlar yardımıyla tasarımcının ihtiyaç duyduğu mantık işlevlerini gerçekleştirme amacına yönelik olarak üretilmiştir. Dolayısıyla FPGA içerisinde bulunan her bir mantık bloğunun işlevi kullanıcı tarafından düzenlenebilmektedir. FPGA isminin kaynağı olan alanda programlanabilir isminin verilmesinin nedeni, mantık bloklarının ve ara bağlantıların imalat sürecinden sonra programlanabilmesidir. Bunun yanı sıra FPGA'in birbirine paralel birçok işlemi aynı anda yapabilme kabiliyetine sahip olması çok daha hızlı sistemlerin tasarlanmasına olanak sağlamaktadır. FPGA içerisindeki yapılar sayesinde içerisine mikroişlemci de gömebilmek mümkündür. Tüm sistemin aynı yerde yer alması bağlantılar arası gecikmeleri azaltacağından FPGA üzerinde daha hızlı sistemler tasarlanabilir. ASIC tasarımlara göre yavaştırlar fakat ASIC'e göre ucuz olması tercih edilmelerini sağlamıştır. Bütün bu özellikler dikkate alındığında tasarım sırasında büyük esneklik sağlaması ve ayrıca FPGA'in paralel işlem yapabilme kapasitesine sahip olmasından ötürü bu tezin FPGA üzerinde tasarlanması tercih edilmiştir.

2.3 Zedboard Geliştirme Kiti

Bu çalışmada, Xilinx Zynq-7000 All Programable SoC'yi kullanan ZedBoard geliştirme kiti kullanılmıştır. Kart, Linux, Windows veya diğer işletim sistemleri için gerekli olan gerekli tüm arabirimleri ve destekleyici işlevleri içerir [13]. Kartın artırılabilir özellikleri, hızlı prototiplendirme için idealdir. Zynq-7000 cihazları, Watt başına mükemmel performans ve maksimum tasarım esnekliği için 28nm Artix-7 veya Kintex-7 tabanlı programlanabilir mantık ile entegre çift çekirdekli ARM Cortex-A9 işlemcileri ile donatılmıştır [14]. 6,6M'ye kadar mantık hücreleriyle ve 6.25Gb/s'den 12.5Gb/s'ye kadar haberleşme hızı sunulan Zynq-7000 cihazları, çoklu

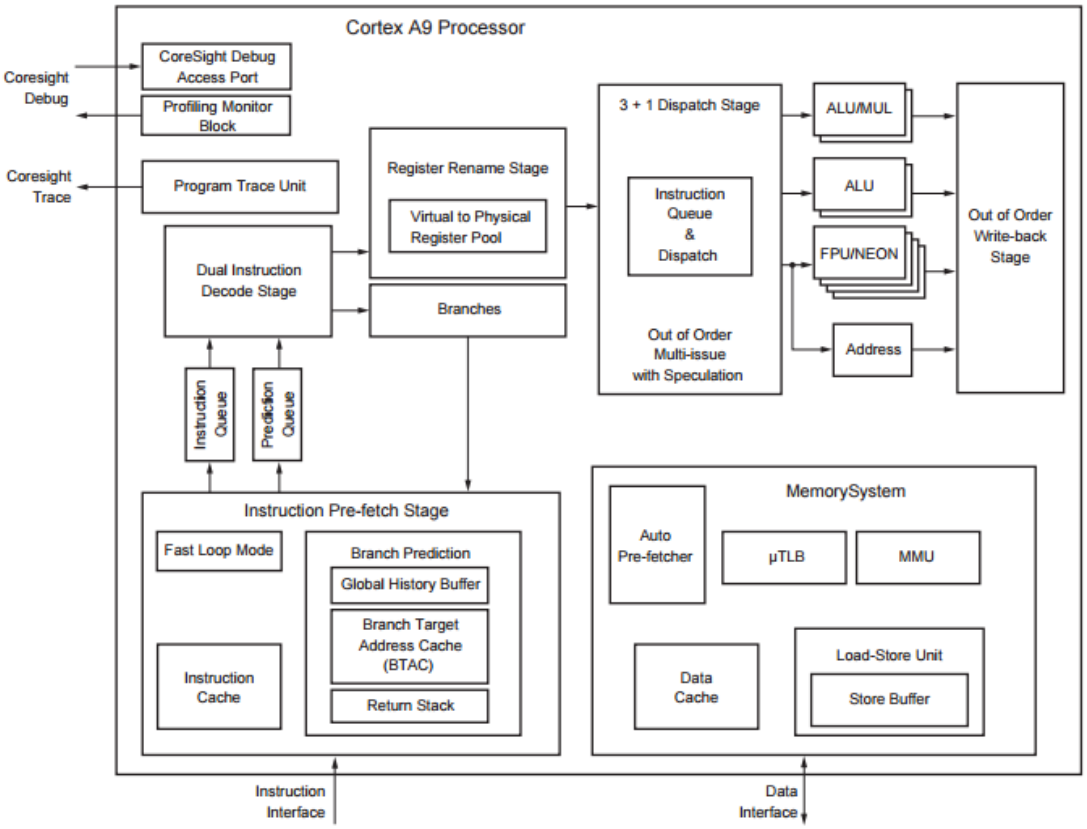
kamera sürücü destek sistemleri ve 4K2K Ultra-HDTV dâhil olmak üzere video işleme, motor kontrolü, yazılım hızlandırması, Linux / Android / RTOS geliştirme, gömülü ARM işleme gibi çok çeşitli gömülü uygulamalar için yüksek oranda farklılaşmış tasarımlar sağlar.

Şekil 2-9 : Zedboard Geliştirme Kiti

2.4 Arm İşlemci

Zynq-7000 ailesi mimari yapısı Şekil 2.10'da gösterilen 32 bitlik ARM Cortex-A9 işlemcileri ile donatılmıştır. Bu projede FPGA üzerinde yazılım ile kontrol edilebilir gömülü sistemler tasarlamaya olanak sağlamak için FPGA bloklarının uygun şekilde programlanması ile oluşturulan sanal işlemci yerine kartın üzerinde gömülü olarak bulunan ARM işlemci kullanılmıştır.

ARM işlemci çift çekirdekli çoklu işlem yapabilme (SMP, AMP), medya işleme motoru (NEON), 32KB L1, 512 KB L2 önbellekleri, iş hattı (pipeline) derinliği, çipli bellek, giriş-çıkış, bellek denetleyicisi, kayan noktalı sayı birimi (Floating Point Unit, FPU), bellek idare birimi (Memory Management Unit, MMU) gibi özelliklere sahiptir [14].



Şekil 2.10 : Kortex A9 Mimarisi [14].

2.5 Verilog Donanım Tanımlama Dili

Verilog donanım tanımlama dili, sayısal devreleri modelleme, test etme ve analiz etme amacıyla kolay, basit ve etkili bir şekilde ifade etmeyi hedefleyen bir donanım tanımlama dili olarak geliştirilmiştir. Yapısal olarak C dili ile olan benzerliği nedeniyle sayısal sistem tasarımında tercih edilecektir.

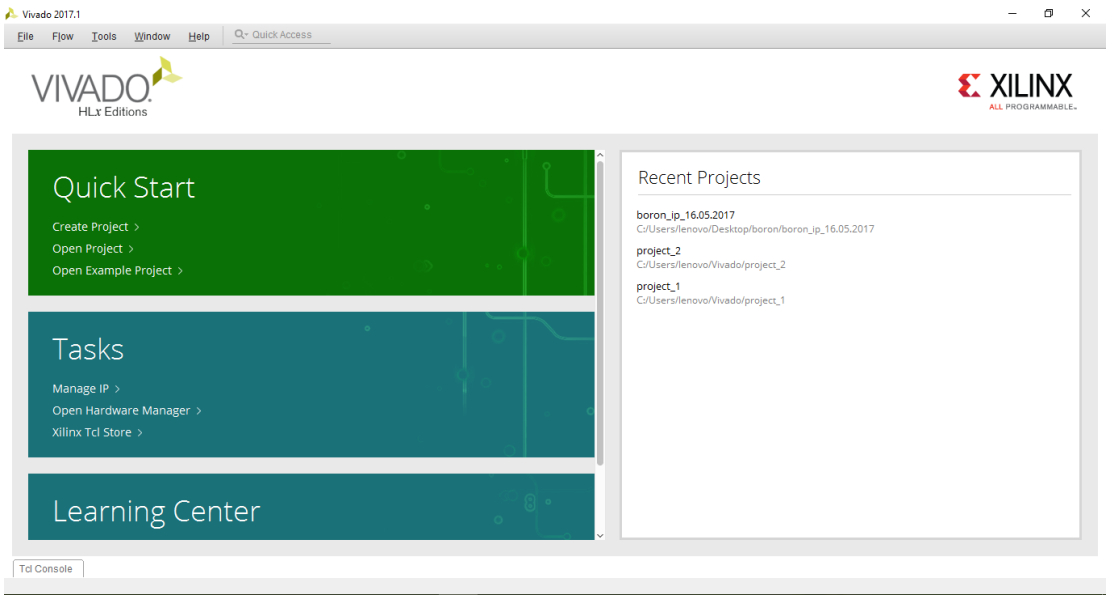
2.6 Xilinx Vivado Ortamı

Vivado Xilinx firmasının geliştirdiği FPGA'ları programlamak için kullanılan bir ara yüz yazılımıdır. Blok seviyesinde tasarım yapmayı sağlayan, tasarlanan donanımların paketlenmesini kolaylaştıran, Matlab üzerinden donanım tasarlatan, C dili ile belli bir kurala göre yazılan kodu donanıma çeviren birçok programı içinde bulunduran bu paket güncellemeler geldikçe hatalarını giderip hızlanarak FPGA tasarımı konusunda büyük bir çözüm olmayı başarmıştır. Xilinx Vivado'nun gelişmesiyle EDK'nın ve ISE'nin geliştirilmesini durdurmuştur. Böylece Xilinx, Vivado ile birçok alandaki çözümleri bir pakette toplayarak tasarımcılara büyük bir kolaylık sağlamıştır.

Bu projede Vivado'nun 2017.1 versiyonunun ücretsiz yayını olan WebPack Edition kullanılmıştır. Vivado Artix®-7, Kintex®-7, Kintex UltraScale™, Zynq®-7000 All Programmable SoC gibi FPGA kartlarını ücretsiz olarak desteklemektedir [15]. Eski seri FPGA kartlarını desteklememektedir. Kurulum yapılırken çok fazla yer kaplamaması için hangi araçların kurulmak istendiği program tarafından otomatikman sorulmaktadır. Bu projede kripto tasarımı bir işlemci ile beraber çalıştırılacağı için SDK aracı yüklenmiştir.

2.6.1 Proje Oluşturma

Vivado ISE'ye nazaran gayet basit bir arayüze sahiptir. Program çalıştırıldığında hızlı başlama, görevler ve öğrenme merkezi olmak üzere 3 bölüme ayrıldığı görülmektedir. Hızlı başlama bölümünden yeni bir proje oluşturulur. Projeye yeni Verilog/VHDL modülleri veya IP blok tasarımları ekleyebilmek için proje türü olarak RTL Projesi seçilir. Sırasıyla yeni kaynak dosyaları ve istenirse fiziksel veya zamansal kısıt dosyaları eklenir. Kaynak dosyası dili Verilog veya VHDL olarak seçilebilir. Bu projede oluşturulan modüller Verilog dilinde yazılmıştır.

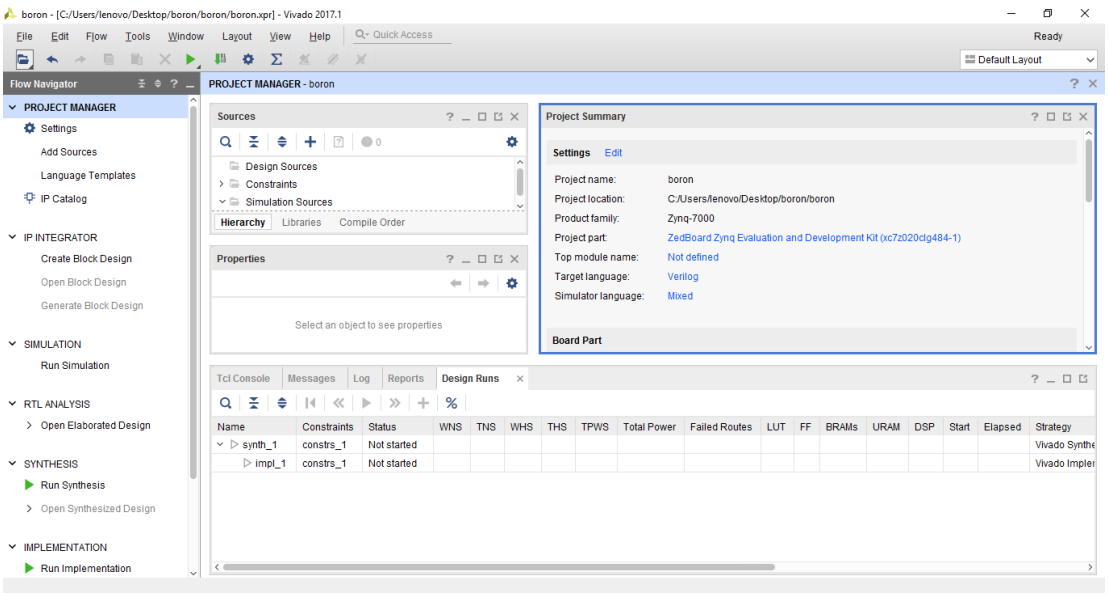


Şekil 2.11 : Vivado Programı

Sonraki bölümde projenin hangi FPGA kartı üzerinde gerçekleneceği seçilmelidir. Bu projede oluşturulan tasarım ZedBoard Zynq geliştirme kiti üzerinde gerçeklenecektir.

2.6.2 Akış Gezgini

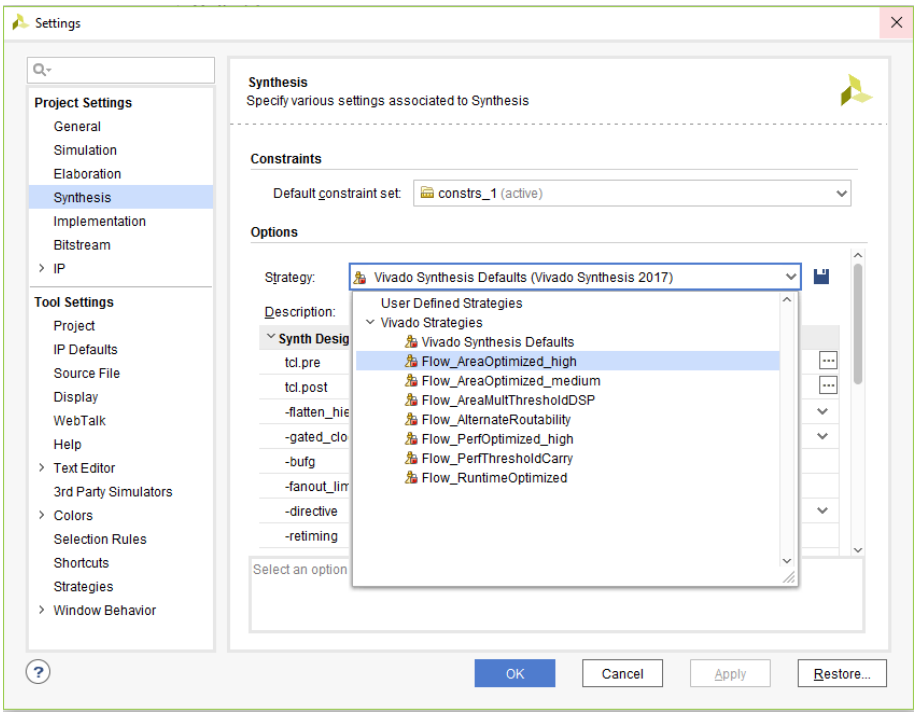
Vivado 2017.1 sürümünde kullanıcı arayüzüne önem verilmiştir. ISE'deki tooların dağınıklığı giderilmiş ve basit kullanışlı bir arayüz oluşturulmuştur. Vivado'nun 2016.4 versiyonunda Akış Gezgini(Flow Navigator)'nde çok fazla işlem bulunurken, 2017.1 versiyonunda simgeler ve tekrar eden işlemler kaldırılarak basitleştirmeye gidilmiştir. Proje için gerekli olan işlemler Akış Gezgini'nde sıralanmaktadır.



Şekil 2-10 : Vivado Ortamı

Proje oluşturulduktan sonra sol tarafta Flow Navigator altında yapılabilecek işlemler sıralanmıştır. Project Manager altında Settings>General sekmesinde projenin hangi FPGA kartında gerçekleneceği ve hangi donanım tanımlama dili ile derleneceği ayarlanabilir. Bu proje ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1) üzerinde Verilog 2001 donanım tanımlama dili ile gerçeklenecektir.

Project Settings altında Synthesis sekmesinden sentez işleminde projenin hangi kısıtlara göre optimizasyon yapılacağı belirlenebilir. Bu proje minimum alanda düşük güç tüketimine sahip bir tasarım oluşturulmaya çalışılmış dolayısıyla Flow_AreaOptimized_high stratejisi seçilmiştir. FPGA üzerinde gerçekleme stratejisi olarak daha az sayıda LUT ve register kullanmak için Area_ExploreSequential (Vivado Implementation 2017) seçilmiştir.



Şekil 2-11 : Sentezleme Stratejisi

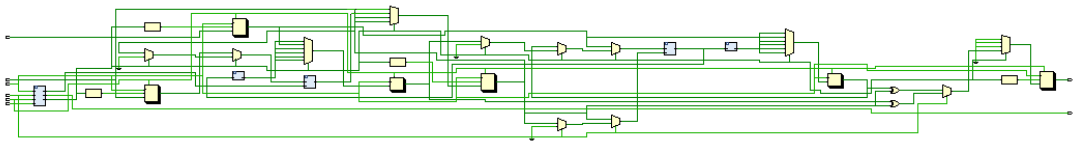
Project Manager>Add Sources kısmından yeni veya halihazırda varolan tasarım, simülasyon veya kısıt dosyaları projeye dahil edilir. ISE programında benzetim dosyaları tasarım dosyalarından otomatik olarak üretilebilirken Vivado programı bu desteği vermemektedir. Simülasyon dosyalarının el ile sıfırdan yazılması gerekmektedir.

Project Manager>IP Catolog sekmesinde Xilinx tarafından hazır olarak kullanıcılara hazır olarak sunulan Akıllı Özellikler(IP) bulunmaktadır. Bu projede mikroişlemci olarak ZYNQ7 Processing System hazır IP si kullanılacaktır.

IP Integrator bölümünde yeni IPler oluşturabilir Xilinx tarafından sunulan hazır IPler ile beraber blok tasarımları üretilebilir. Ayrıca üretilmiş IP'ler üzerinde değişiklikler yapılabilir.

Simulation bölümünde Run Simulation sekmesi altında davranışsal, sentez sonrası ve implementasyon sonrası benzetimler yapılarak tasarımın doğru çalışıp çalışmadığı test edilir. Davranışsal sentezlemede tasarımın sadece fonksiyonel olarak doğru çalışıp çalışmadığı test edilir.

RTL Analysis bölümünde tasarımın RTL olarak bağlantıları gerçekleştirilir. Tasarımın ne kadar hücreye sığdığı, giriş-çıkış portu sayısı belirtilir. Tasarım kuralı denetim raporları ve gürültü raporları oluşturulabilir.



Şekil 2-12 : Boron RTL şematiği

Synthesis bölümünde tasarım kısıtlamalara uygun olarak, yüksek seviye ve düşük seviye optimizasyonlar yapılarak sentezlenir. Open Synthesis Design bölümünde projeye zaman kısıtları eklenebilir. Aynı şekilde Project Manager>Add Sources içinden zaman kısıtları dosyası eklenebilir. Bu projede tüm girişler çıkışlar arasında maximum 10 ns gecikme olması istenmiştir.

Implementation bölümünde oluşturulan tasarımın kısıtlamalar dikkate alınarak FPGA üzerine haritalandırma işlemi yapılır. Kullanıcıya tasarladığı sistemlerin donanım üzerinde ne kadar yer kapladığını, ne kadar hat gecikmesine sahip olduğunu ve yapılan tasarımın FPGA'in hangi bölgesine yerleştirileceğine kadar detaylı bilgiler sunmaktadır.

Program and Debug bölümünde Generate Bitstream ile tasarımın FPGA'e gömülecek bit dosyası oluşturulur. Open Hardware sekmesinden FPGA kartı bağlantısı yapılarak FPGA programlanır.

2.7 Xilinx SDK Ortamı

Yazılım Geliştirme Kiti (Software Development Kit, SDK) Xilinx firması tarafından Vivado ortamında tasarlanan mikroişlemci merkezli sayısal sistem tasarımlarının yazılım tasarımını gerçeklemek için geliştirilen ara yüz ortamıdır. Xilinx'in tasarım ortamlarının eski sürümlerinde SDK yazılım tasarımı, XPS ise donanım geliştirmek için beraber kullanılıyordu. Daha sonra Xilinx firması XPS ortamını da Vivado'ya dahil ederek donanım tasarlama işlemlerinin hepsini tek bir yerde topladı. SDK ise sadece tasarlanan donanımlara yazılım tasarımı yapmak amacıyla kullanılmaktadır. Vivado ortamında tasarlanan sisteme ait kullanıcı donanımları ve çevre birimlerinin kütüphaneleri üretilerek yazılım tasarımına ilk adımın atılması sağlamaktadır. Aynı zamanda, SDK tarafından üretilen kütüphanelerin söz konusu söz konusu yazılım projesine eklenmesiyle kullanıcıya mikroişlemciyi kolayca kontrol etme olanağı sağlanmaktadır.

- Zengin özellikli C/C++ kod editörü ve derleme ortamı
- Proje yönetimi
- Tasarım yapılandırması uygulaması ve otomatik Makefile üretimi
- Hata navigasyonu
- Kaynak düzeyinde hata ayıklama ve gömülü hedeflerin görünüşü için iyi tümleştirilmiş ortam
- Kaynak kodu sürümü kontrolü

SDK tarafından kullanıcılarına sunulmuş başlıca özelliklerdir [16].

Vivado ortamında donanım tanımlama dilleri ya da şematik çizimlerle tasarlanan donanımlar yine Vivado ortamında özel IP olarak paketlenerek mikroişlemci merkezli sayısal sistem tasarımına eklenmek istenen diğer hazır IP'lerle birlikte eklenmektedir. Vivado ortamında donanım yapısı tamamlanan sistem SDK ortamına gönderilerek bu aşamada otomatik olarak kütüphaneleri üretildikten sonra yazılım tasarımı yapılmaktadır. Son olarak, donanım ve bu donanımları kontrol etmek için yapılan yazılım da tamamlandıktan sonra SDK aracılığıyla, donanım bilgilerini içeren "bit" uzantılı donanım dosyası ve "elf" uzantılı yazılım dosyası birleştirilerek FPGA'e gönderilir.

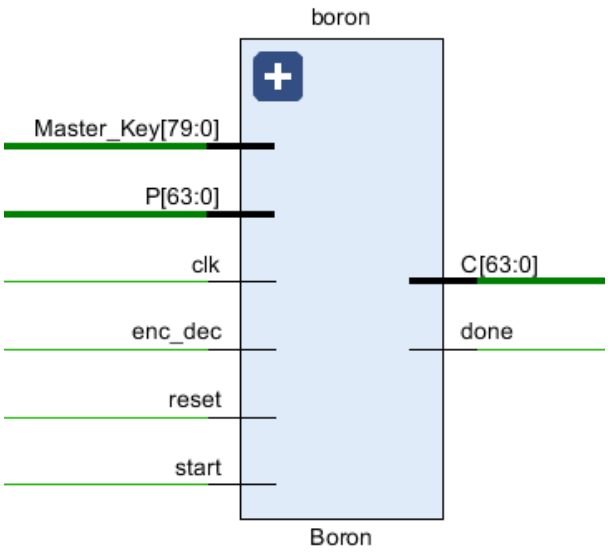
3. DONANIM TASARIMI

Boron şifreleme algoritmasının gerçeklenmesi aşamasına ilk olarak algoritmada yer alan donanımların Verilog donanım tanımlama dili kullanılarak Vivado programında tasarlanmasıyla başlanmıştır. Algoritmada bulunan donanımlar; Boron modülü altında toplanmıştır.

Donanım tasarımı aşaması tamamlandıktan sonra bu donanımları ve sistemi kontrol etmek amacıyla FPGA içerisindeki gömülü olarak bulunan ARM mikroişlemcisi üzerinde yazılım tasarımı yapılmıştır.

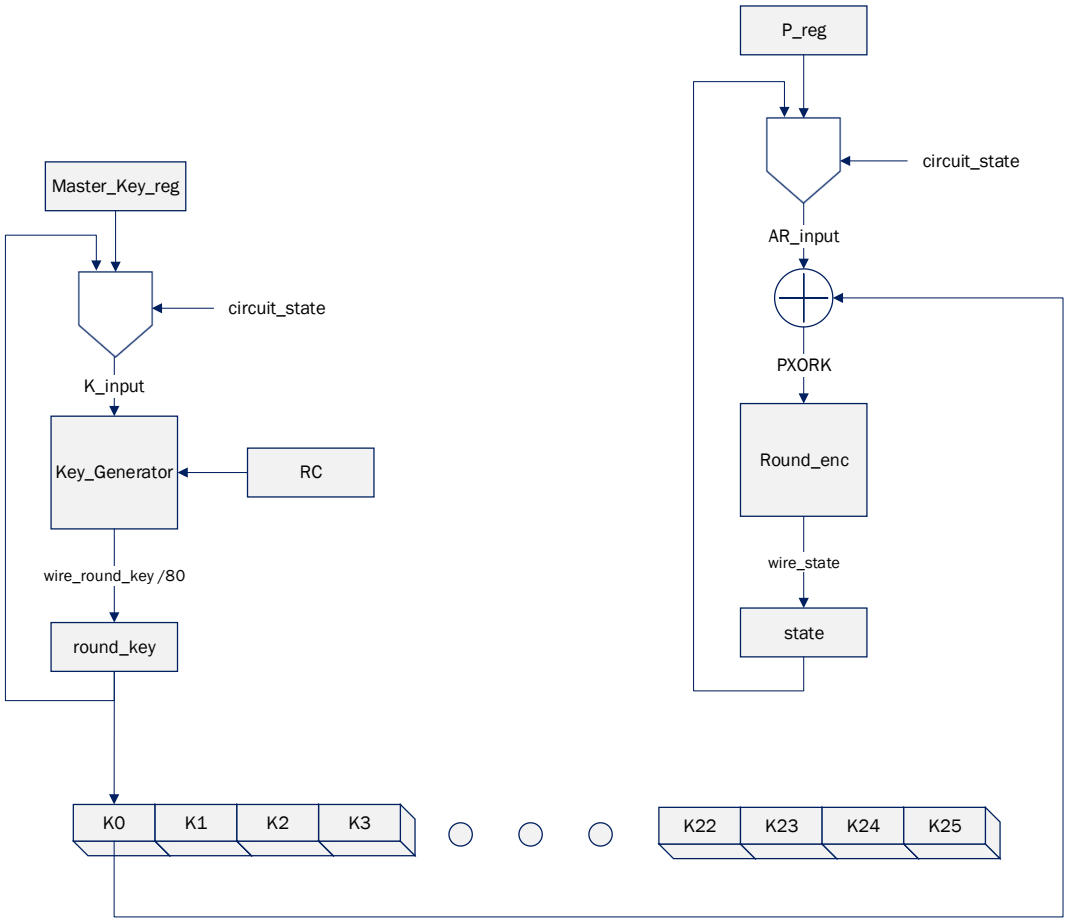
3.1 Boron Modülü

Boron modülünün 64 bit metin girişi vardır. Ana anahtar giriş uzunluğu 80 bit seçilmiş ve buna göre tasarım yapılmıştır. Boron modülü kendi içinde hem şifreleme hem şifre çözme işlemi gerçekleştirebilecek şekilde tasarlanmıştır. 1 bitlik enc_dec girişi modülün giriş metnini şifreleyeceğini mi yoksa çözeceğini mi belirtir. Boron modülünde kullanılan start ve done işaretleri, yapının mikroişlemci ile haberleşmesi amacıyla kullanılmıştır.

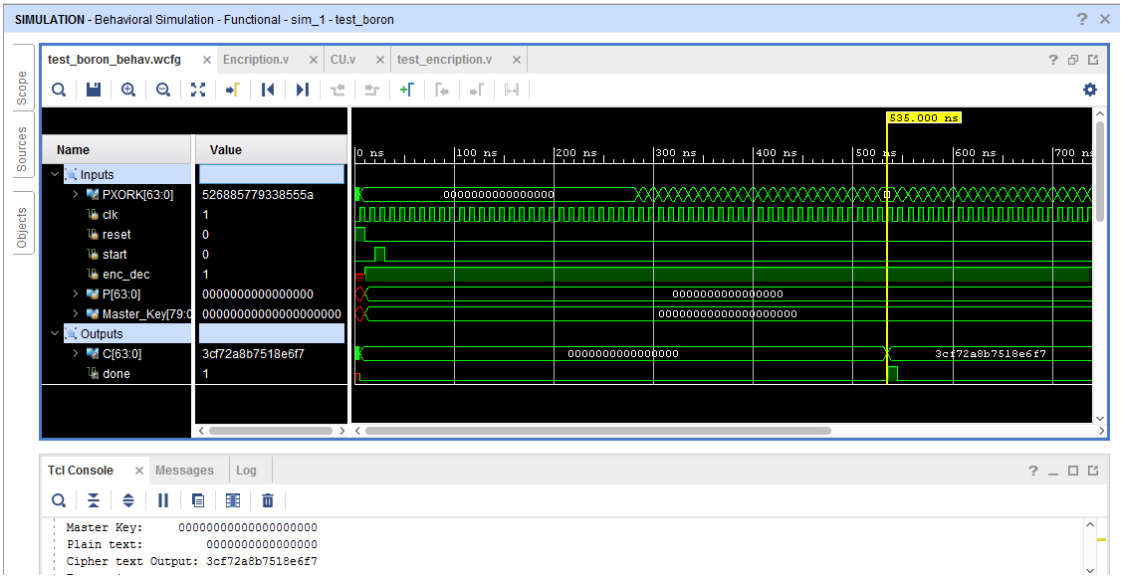


Şekil 3-1 : Boron modülü RTL şematiği

Boron kendisini 25 defa tekrar eden Round fonksiyonundan oluşmaktadır. Aynı zamanda anahtar üretimi de birbirini tekrar eden bir yapıya sahiptir. İçerisinde bulunan CU sonlu durum makinesi sayesinde Boron modülünün senkron bir şekilde işlevlerini yerine getirmesi sağlanmıştır. Modülde şifreleme çözme işleminde tekrar tekrar anahtar üretme yerine anahtarların bir kez üretilip kayıt edilmesi tercih edilmiştir. Bu durum ilk aşamada fazla alan kullanma dezavantajı getirse de, sürekli çalışan bir sistem düşünüldüğünde tekrar tekrar anahtar üretme işleminin olmamasının toplam çalışma süresini ciddi manada düşüreceği, dolayısıyla güç tüketimini minimize edeceği düşünülmüştür.



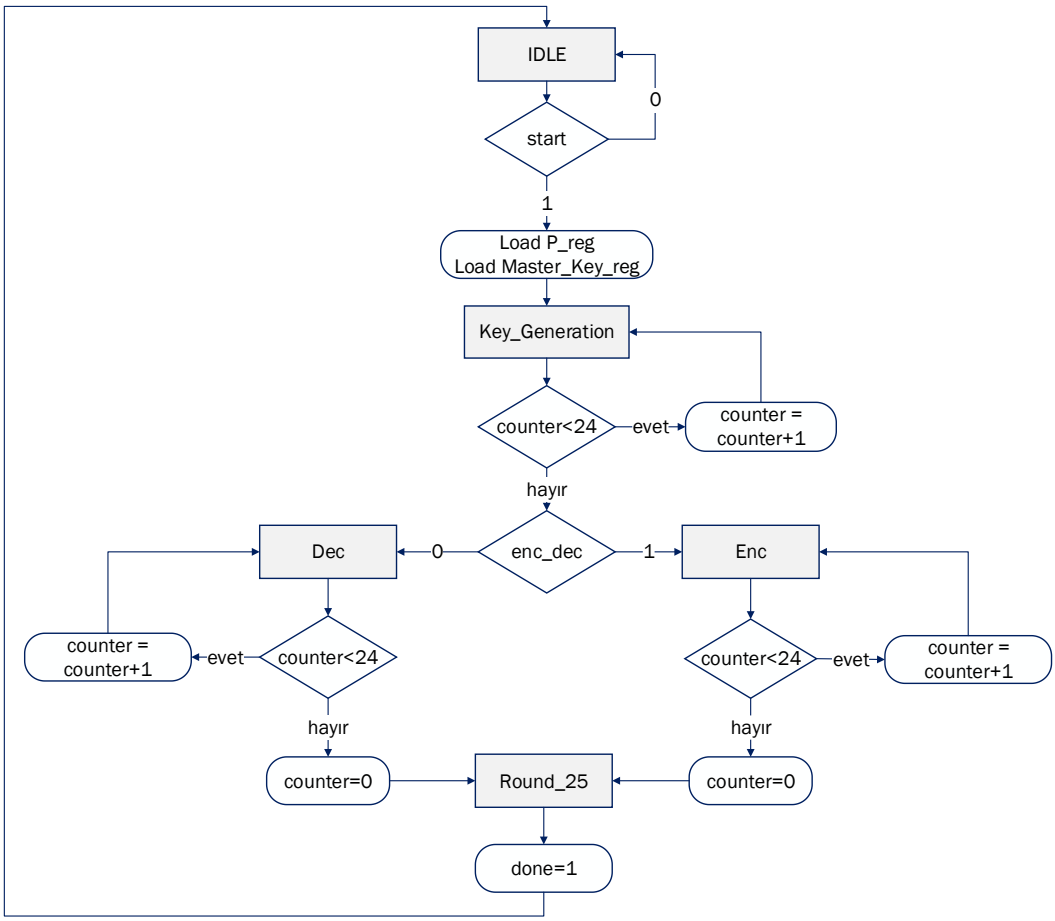
Şekil 3-2 : Boron modülü çalışma diyagramı



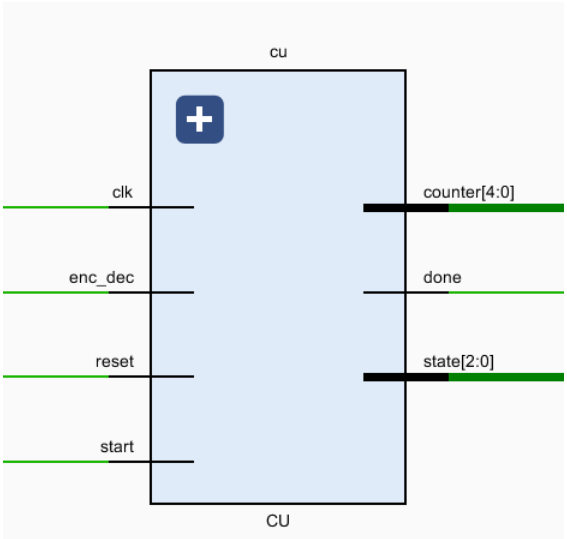
Şekil 3-3 : Boron modülünün davranışsal benzetim sonucu

3.1.1 CU Modülü

Boron modülünü yönetmek için bir sonlu durum makinası tasarlanmıştır. IDLE, Key_Generation, Enc, Round25, Dec olmak üzere 5 farklı çalışma durumu belirlenmiştir. IDLE durumunda işlemciden start işlemi gelene kadar beklenir. Start işaretinin gelmesi durumunda Key_Generation durumuna geçilir ve sayacımız (0'dan başlıyor) 24 olana kadar bu durumda kalarak Round fonksiyonumuz için gerekli olan anahtarlar üretilir. Sayacımız 24 olduğunda enc_dec işaretine bakılarak Enc veya Dec durumlarına geçiş yapılır. Enc durumu şifreleme işlemlerinin, Dec ise şifre çözme işlemlerinin yapıldığı durumdur. Bir sayaç tutularak 24 tur bu durumda kalınır ve sayaç 24 olduğunda Round_25 durumuna geçilir. Artık Boron modülü görevini yerine getirmiştir ve done çıkışını yükseğe çekerek başlangıç (IDLE) durumuna geri döner. Bu işlemi yaparken sayacı da sıfırlayarak yeni işlemlere hazır hale getirir.



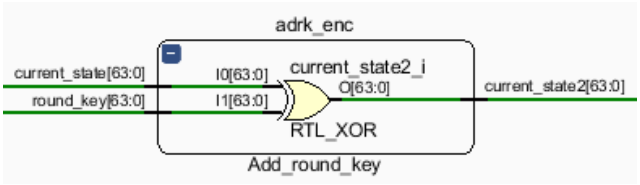
Şekil 3-4 : CU modülü akış diyagramı



Şekil 3-5 : CU modülü RTL şematiği

3.1.2 Add_round_key Modülü

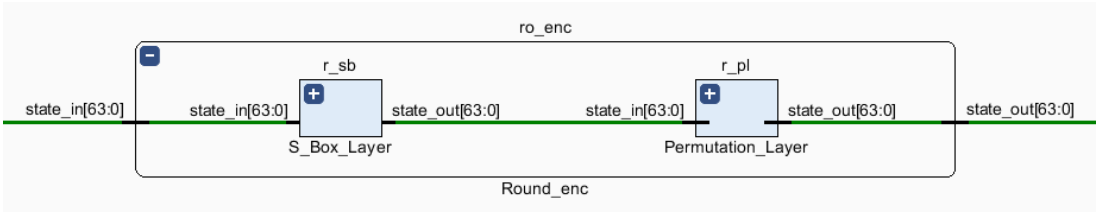
Add_round_key modülü şimdiki durum ile tur anahtarının en düşük anlamlı 64 biti arasında basit bir XOR işlemi yapar.



Şekil 3-6 : Add_round_key modülü RTL şematiği

3.1.3 Round_enc Modülü

Boron modülü kendini 25 kez tekrar eden Round fonksiyonuna sahiptir. Round fonksiyonunun anahtar ekleme bölümü Add_round_key modülünde gerçekleştirilirken doğrusal olmayan permütasyon (S_Box_layer) ve doğrusal katman (Permütation_Layer) bölümü Round modülü içerisinde gerçeklenmiştir.



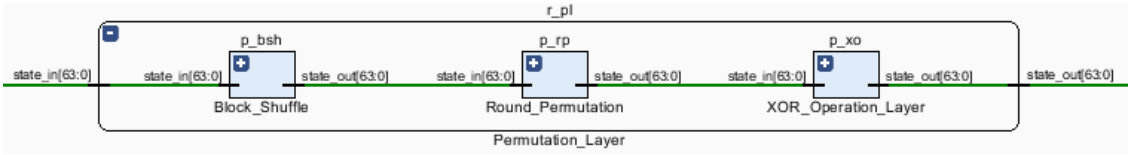
Şekil 3-7 : Round_enc modülü RTL şematiği

3.1.3.1 S_Box_Layer Modülü

S_Box modülü 4 bit girişten 4 bit çıkış üreten birbiriyle paralel çalışan toplamda 16 s_boxtan oluşur. S_box modülleri Tablo 1'de verilen değerlere göre doğrusal olmayan şifre değiştirme işlemi yapar. Modül tasarlanırken gerçeklenecek fonksiyonda herhangi bir indirgeme yapılmamış, girdiler ve çıktılar direk switch-case yapısında yazılmıştır.

3.1.3.2 Permutation_Layer Modülü

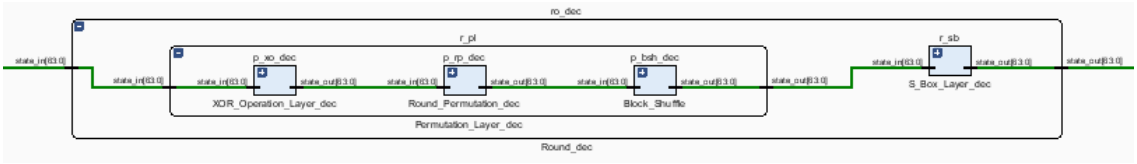
Permutation_Layer modülü birbirine seri olarak bağlanmış 3 ara modüle sahiptir. 2.1.2.3 bölümünde verilen işlevleri kombinezonsal olarak yerine getirir.



Şekil 3-8 : Permutation_Layer modülü RTL şematiği

3.1.4 Round_dec Modülü

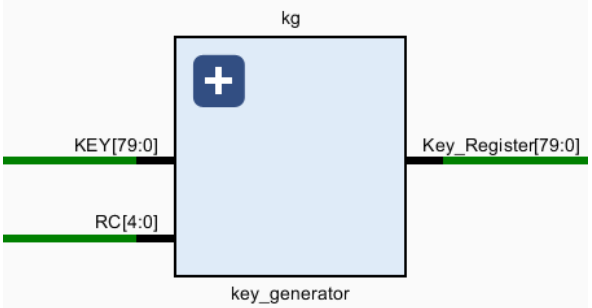
Round_enc modülünün tam tersi şeklinde tasarlamıştır. Şifre çözme işleminde bu modül kullanılacaktır.



Şekil 3-9 : Round_dec modülü RTL şematiği

3.1.5 key_generator Modülü

Boron modülü 26 adet ara anahtara ihtiyaç duymaktadır. Bu tasarımda ara anahtarlar 2080 (26*80) bitlik Key_Register'a kayıt edilir. Kullanıcı tarafından tanımlanan 80 bitlik anahtar key_generator modülüne sokularak bir takım matematiksel işlemler yapıldıktan sonra üretilen ara anahtar Key_Register'ın en anlamlı 80 bitine kayıt edilir. Her turda Key_Register 80 bit sağa doğru kaydırılarak yeni oluşturulmuş anahtarın en anlamlı 80 bitte kalması sağlanır. Toplamda 26 turun sonunda ilk oluşturulan K0 anahtarımız Key_Register'ın en anlamsız[79:0] 80 bitine yerleşirken son üretilen anahtar (K25) Key_Register'ın en anlamlı[2079:2000] 80 bitine yerleşir. Add_round_key modülü şifreleme için kullanılacak ise Key_Register'ın en anlamsız 80 bitinden (K0) başlayarak her bir turunda Key_Register'ı 80 bit sola kaydırarak üretilen bu anahtarları kullanır. Eşer şifre çözme için kullanılacaksa tam tersi biçimde Key_Register'ın en anlamlı 80 bitinden (K25) başlayarak her bir turunda 80 bit sağa kaydırma yaparak üretilen anahtarları kullanır.



Şekil 3-10 : key_generator modülü RTL şematiği

3.2 Blok Tasarımı

3.2.1 Özel IP (Custom IP) Oluşturma ve Paketleme

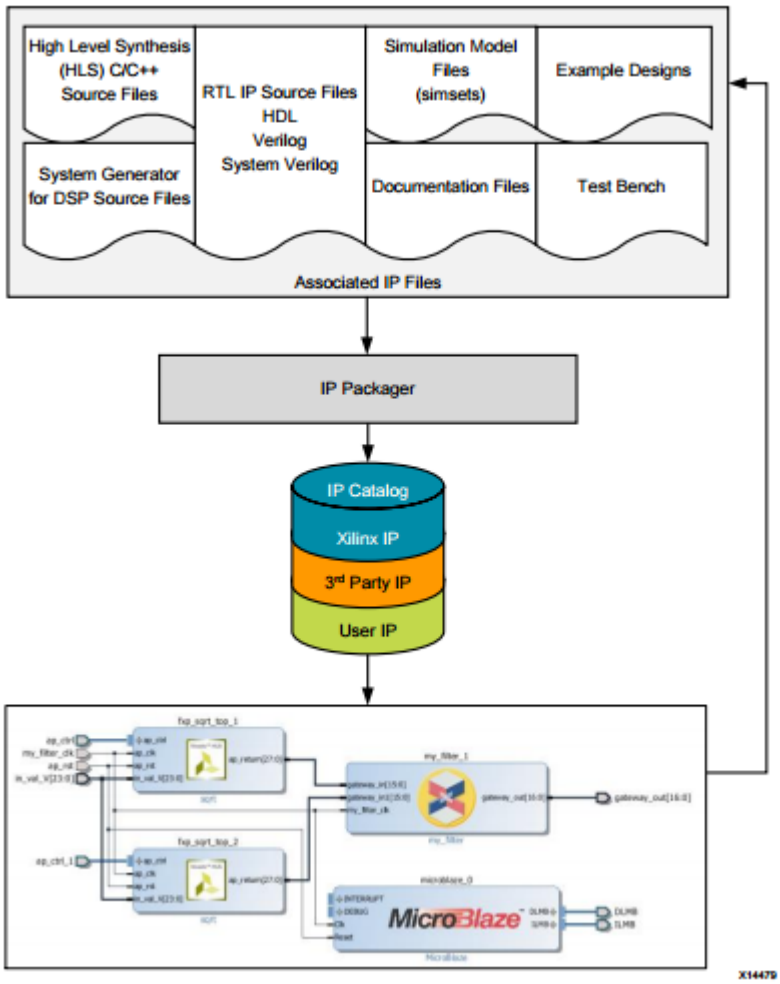
Şekil 3.11, Akıllı Özellik (IP) paketleme ve kullanım modelindeki akışı göstermektedir. Vivado IP Paketleyiciyi ile yapılabilecekler şunlardır:

• Dosyaları ve ilişkili verileri IP-XACT standart formatında oluşturup paketleme.

• Vivado IP Kataloğuna IP ekleme.

• Bir paket deposu dizinindeki bir son kullanıcıya veya bir arşiv (.zip) dosyasına paketlenmiş IP sunma.

IP dış ortama dağıtıldığında, bir son kullanıcı tasarımlarında bu IP'nin özelleştirilmesini sağlayabilir [15].



Şekil 3-11 : IP paketleme ve blok tasarımda kullanımı

IP paketleyicisi, IP'ye uygun birçok dosya grubunu dizayn edebilir. IP paketleyicisi, IP-XACT standardı olan component.xml'e dayalı bir XML dosyası ve bir XGUI özelleştirme Tcl dosyası oluşturur. Bu iki dosya, IP kök dizini konumunda oluşturulur.

3.2.2 Custom IP Oluşturma ve IP Paketleme Temelleri

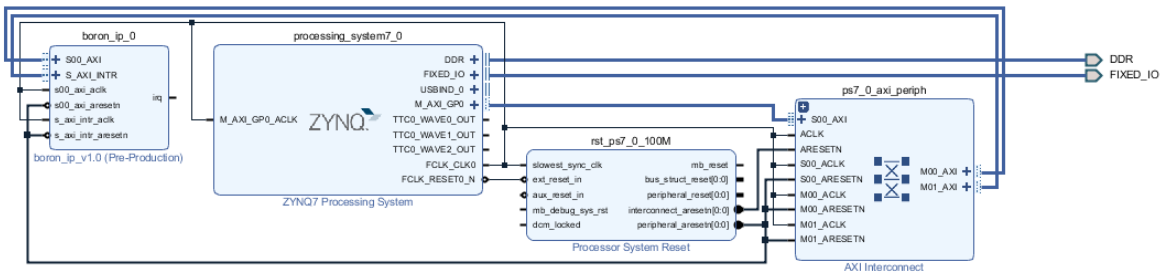
IP paketleyicisi, IP üst düzey HDL dosyası için HDL sentez dili yapılarını destekler. Vivado IP paketleyicisi, IP portlarının kendinden bağımsız olmasını gerektirir. Bu nedenle IP Paketleyici, HDL dosyalarında tanımlanan özel fonksiyonları desteklemez. Vivado IP paketleyicisi, arzulanan matematiksel ifadeleri oluşturmak için standart aritmetik ve mantıksal operatörleri destekler. Karmaşık matematiksel ifadeleri desteklemek için IP Paketleyici XPATH fonksiyonunu destekler.

3.2.3 AXI Sinyallerini Anlamdırma

Xilinx, Akıllı Özellik (IP) çekirdekleri için Gelişmiş Genişletilebilir Arayüz (AXI) protokolünü benimsemiştir. Özel IP'mizde bu protokolü kullanmak tasarımımızın Vivado IP Kataloğu'ndaki diğer IP'ler ile bağlanmasında esneklik sağlar. Özel IP'nin port sinyalleri AXI'ye bağlıysa Vivado IP paketleyici, AXI arayüzünü otomatik olarak anlamdıracaktır.

3.2.4 IP Integrator'u Kullanarak Vivado'da Özel bir IP Çekirdeği Oluşturma

Vivado'da özel bir AXI IP bloğu oluşturulmuş ve Boron modülü Verilog kodları entegre edilmiştir. Boron IP'sinin paketlenmesinin ardından blok tasarımına geçilmiştir. Zynq-7000 donanım platformunun da blok tasarımına eklenmesiyle Boron IP'si ve işlemci arasında AXI haberleşme protokolü bağlantıları program tarafından otomatik olarak gerçekleştirilmiştir. Bu projede, Zynq-7000 platformunun ARM işlemci sistemi kullanılarak, işlemci için yazılmış C kodu ile beraber Boron şifreleme donanımının kontrolü sağlanmıştır.

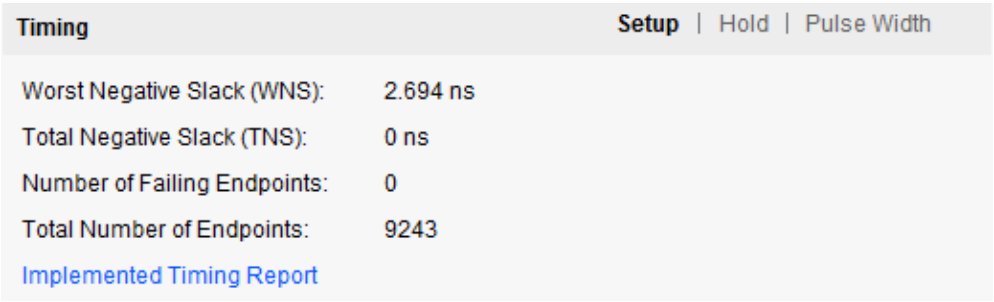


Şekil 3-12 : Boron IP'sinin ZYNQ ile bağlanması

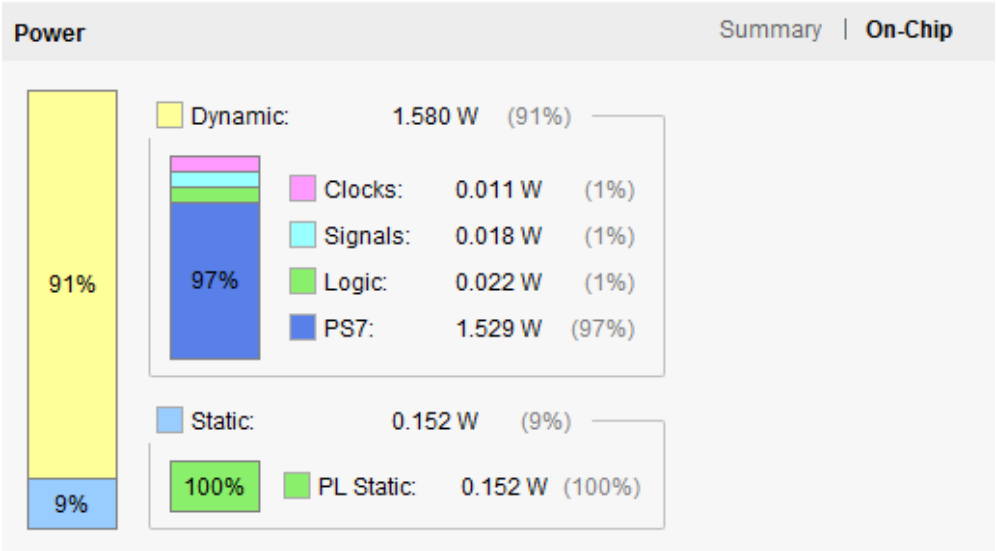
3.3 Tasarlanan Donanımın Benzetim Sonuçları

Resource	Utilization	Available	Utilization %
LUT	2399	53200	4.51
LUTRAM	62	17400	0.36
FF	3320	106400	3.12
BUFG	1	32	3.13

Şekil 3-13 : Alan Kullanım Sonuçları



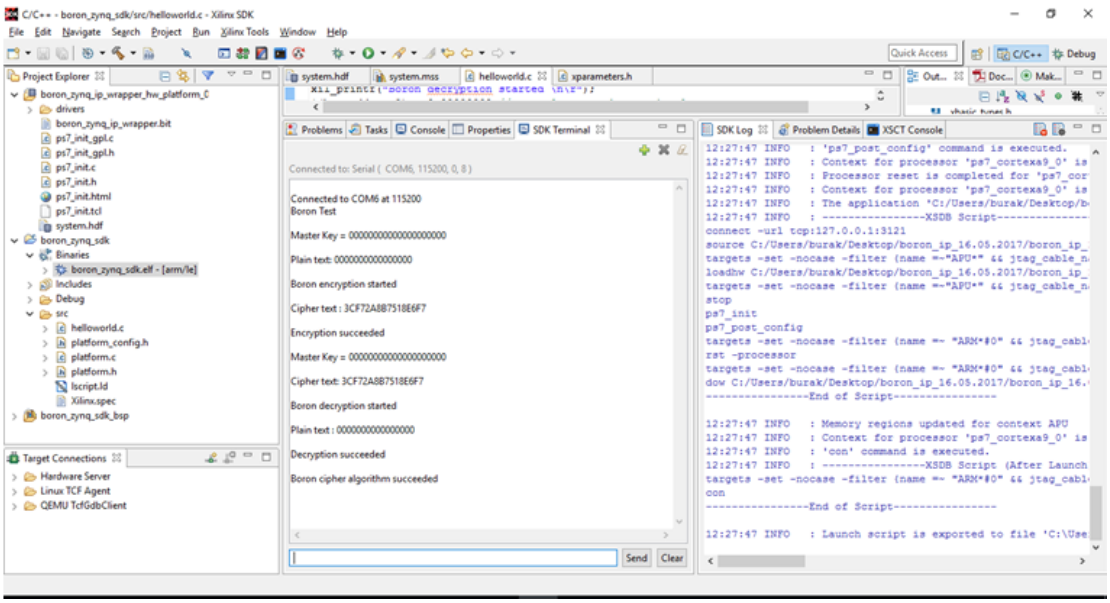
Şekil 3-14 : Kritik Zaman Sonuçları



Şekil 3-15 : Güç Tüketimi Sonuçları

4. YAZILIM TASARIMI

Boron şifreleme-şifre çözme donanımını kontrol etmek, donanımlar arası veri akışını sağlamak amacıyla bir mikroişlemci kullanılmıştır. Xilinx tarafından sağlanan Mikroblaze sanal işlemcisini FPGA üzerine gömmek yerine, kullanılan Zedboard kartında donanımsal olarak hâlihazırda bulunan Zynq ARM işlemci tercih edilmiştir. Böylece FPGA üzerinde işlemci için gereksiz yere alan kullanımından kaçınılmıştır. İşlemciye SDK arayüzünde yazılan C kodu ile Boron modülüne giriş metni, anahtarı, şifeleme mi şifre çözme mi yapacağı söylenmiştir. İlk olarak reset ile Boron modülü registerları sıfırlanır. start işareti ile beraber girişler modüle aktarılır. Modülün işlemlerine tekrar tekrar devam etmemesi için belirli bir süre sonunda start işareti düşük seviyeye çekilir. Boron modülü görevlerini yerine getirdiğinde UART haberleşmesiyle terminale ürettiği çıktıyı yazar. Yazılan kodda önce bir metnin şifreleme işlemi gerçekleştirilir. Üretilen sonuç registerlara kayıt edilir. Daha sonra şifrelenmiş metin tekrar modüle sokularak bu sefer şifre çözme işlemi gerçekleştirilir. Elde edilen metin başlangıçtaki giriş metni ile örtüşüyorsa tasarımın başarıyla gerçekleştirildiği UART üzerinden terminale yazdırılır.



Şekil 4-1 : SDK arayüzünde tasarımın çıktıları

5. SONUÇLAR

Bu bitirme çalışması kapsamında düşük ölçekli gömülü sistemler için düşük güç tüketimli, az yer kaplayan hafif bir kripto donanımı tasarlamak hedeflenmiştir. Bu amaçla seçilen kripto algoritması FPGA üzerinde ARM işlemci ile beraber gerçeklenmiş, yazılım donanım ortaklı bir sistem elde edilmiştir. Tasarımda gerek yer gerek güç tüketimi bakımından daha verimli bir yöntem olduğundan Boron şifreleme algoritması tercih edilmiştir.

Çalışma kapsamında şifreleme ve şifre çözme işlemleri için gerekli olan anahtarların her seferinde tekrar tekrar üretilmesi yerine her bir turda üretilen anahtarların bir kayıt bloğunda saklanması tercih edilmiştir. Bu durum ilk etapta kullanılan LUT sayısını ister istemez artırmıştır. Fakat sürekli çalışan bir sistem düşünüldüğünde tekrar tekrar ara anahtarlar üretilmeyeceği için sistemin çalışma süresinde dolayısıyla toplam güç tüketiminde azalma olacağı düşünülmüştür.

ARM işlemci ve Boron donanımının haberleşmesi arasında problemler oluşmuş, problem Boron donanımının sentez sonrası benzetim çıktılarına bakılarak çözülmüştür.

Boron kripto modülü şifreleme şifre çözme işlemleri ile beraber FPGA üzerinde 2399 dilim işgal etmiştir. Bu sonuçlar AES gibi yaygın şifreleme algoritmaları ile kıyaslandığında, tasarlanan donanımda hedeflenen düşük yer kaplama ve düşük güç tüketimine ulaşılmıştır.

İleri çalışmalarda tasarımda iyileştirmeler yapılıp olası yan kanal saldırılarına karşı nasıl önlemler alınabileceği araştırılarak daha gelişmiş bir sistem tasarlanabileceği tespit edilmiştir.

KAYNAKLAR

[1] NIST (National Institute of Standards and Technology), “Advanced Encryption Standard (AES),” Federal Information Processing Standards Publication 197, November 2000.

[2] J. Daemen and V. Rijmen, “The Design of Rijndael, AES - The Advanced Encryption Standard”, Springer-Verlag 2002 (238 pp.)

[3] Barker, William Curt. Recommendation for the triple data encryption algorithm (TDEA) block cipher. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2004.

[4] Hemme, Ludger. "A differential fault attack against early rounds of (triple-) DES." In Cryptographic Hardware and Embedded Systems-CHES 2004, pp. 254-267. Springer Berlin Heidelberg, 2004.

[5] Petroulakis, Nikolaos E., Ioannis G. Askoxylakis, and Theo Tryfonas. "Lifelogging in smart environments: challenges and security threats." Communications (ICC), 2012 IEEE International Conference on. IEEE, 2012.

[6] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification. Hoboken, NJ, USA: Wiley, 2003.

[7] G. BANSOD, N. PISHAROTY, A. PATIL3, “BORON: an ultra lightweight and low power encryption design for pervasive computing”, Frontiers of Information Technology & Electronic Engineering,

http://www.zju.edu.cn/jzus/iparticle.php?doi=10.1631/FITEE.1500415

[8] A. Bogdanov et al., “PRESENT—An ultra-lightweight block cipher,” in Cryptographic Hardware and Embedded Systems (Lecture Notes in Computer

Science), vol. 4727, P. Paillier and I. Verbauwhede, Eds. Berlin, Germany:

Springer-Verlag, 2007, pp. 450–466.

[9] T. Okabe, ‘‘FPGA Implementation and Evaluation of lightweight

block cipher – BORON’’, Tokyo Metropolitan Industrial Technology Research Institute, 2017

[10] Koca, H., 2007. “Robot Manipülatör Denetimi”, Yüksek Lisans Tezi, Gazi

Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

[11] Chu, Pong P., 2008. FPGA Prototyping by VHDL Examples. Wiley-

Interscience, New Jersay.

[12] Bağbaba, A., 2013. “Güvenli Bir Yakın Haberleşme Sisteminin FPGA Üzerinde Gerçeklenmesi”, Lisans Tezi, İstanbul Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği, İstanbul.

[13] Avnet zedboard. http://zedboard.org/product/zedboard.

[14] Xilinx, 2016. Zynq-7000 All Programmable SoC Technical Reference Manual.

[15] Xilinx, 2017. Vivado Design Suite User Guide.

[16] Xilinx, 2017. Software Development Kit (SDK) User Guide

ÖZGEÇMİŞ

Adı Soyadı: Burak Acar

Doğum Yeri ve Tarihi: İstanbul, 1994

E-Mail: burakacaritu@gmail.com

Lise: Kadıköy Anadolu Lisesi; 2008-2013

Lisans: İstanbul Teknik Üniversitesi, Elektronik ve Haberleşme Mühendisliği; 2013-2017